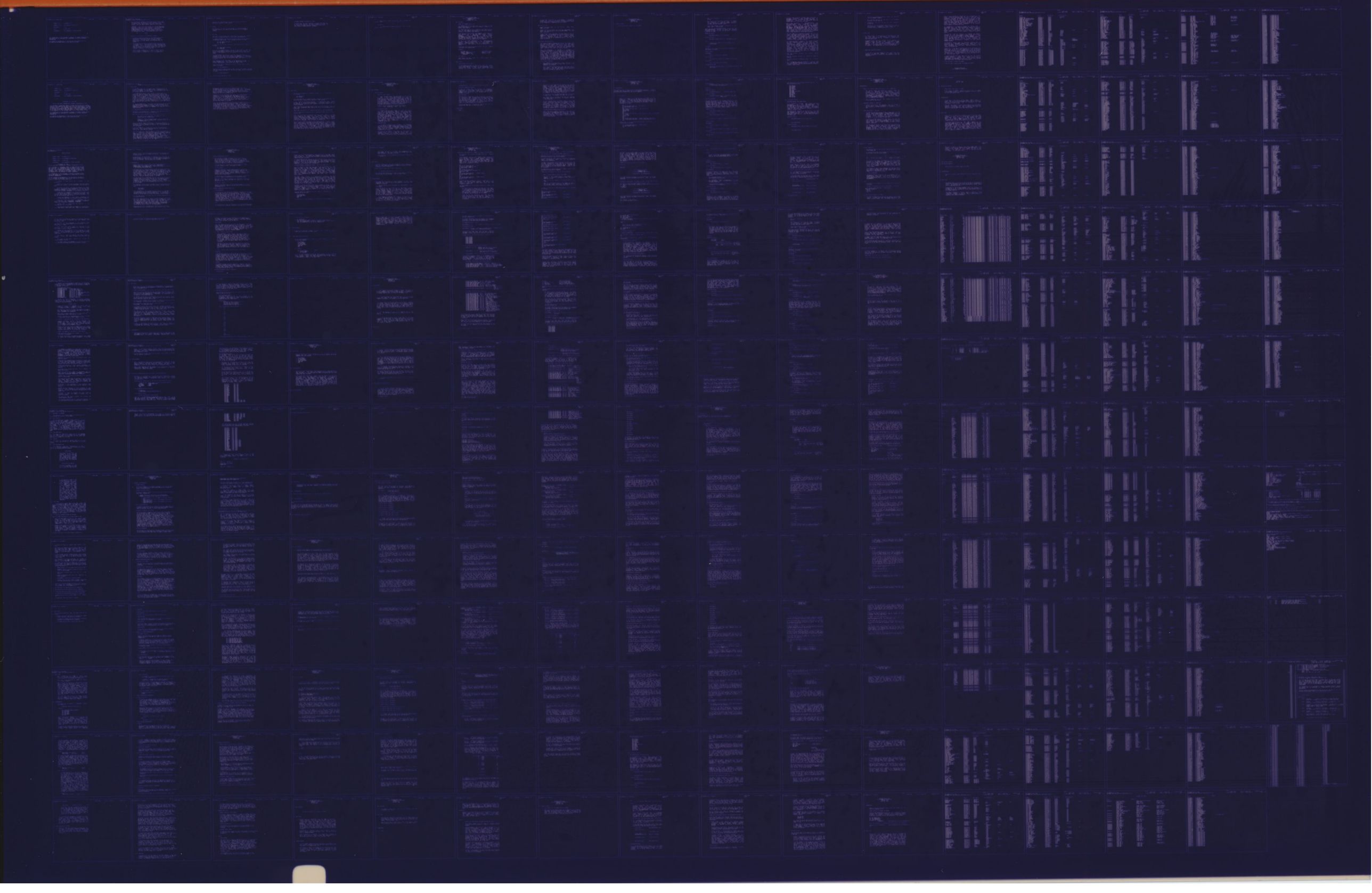
ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
1 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

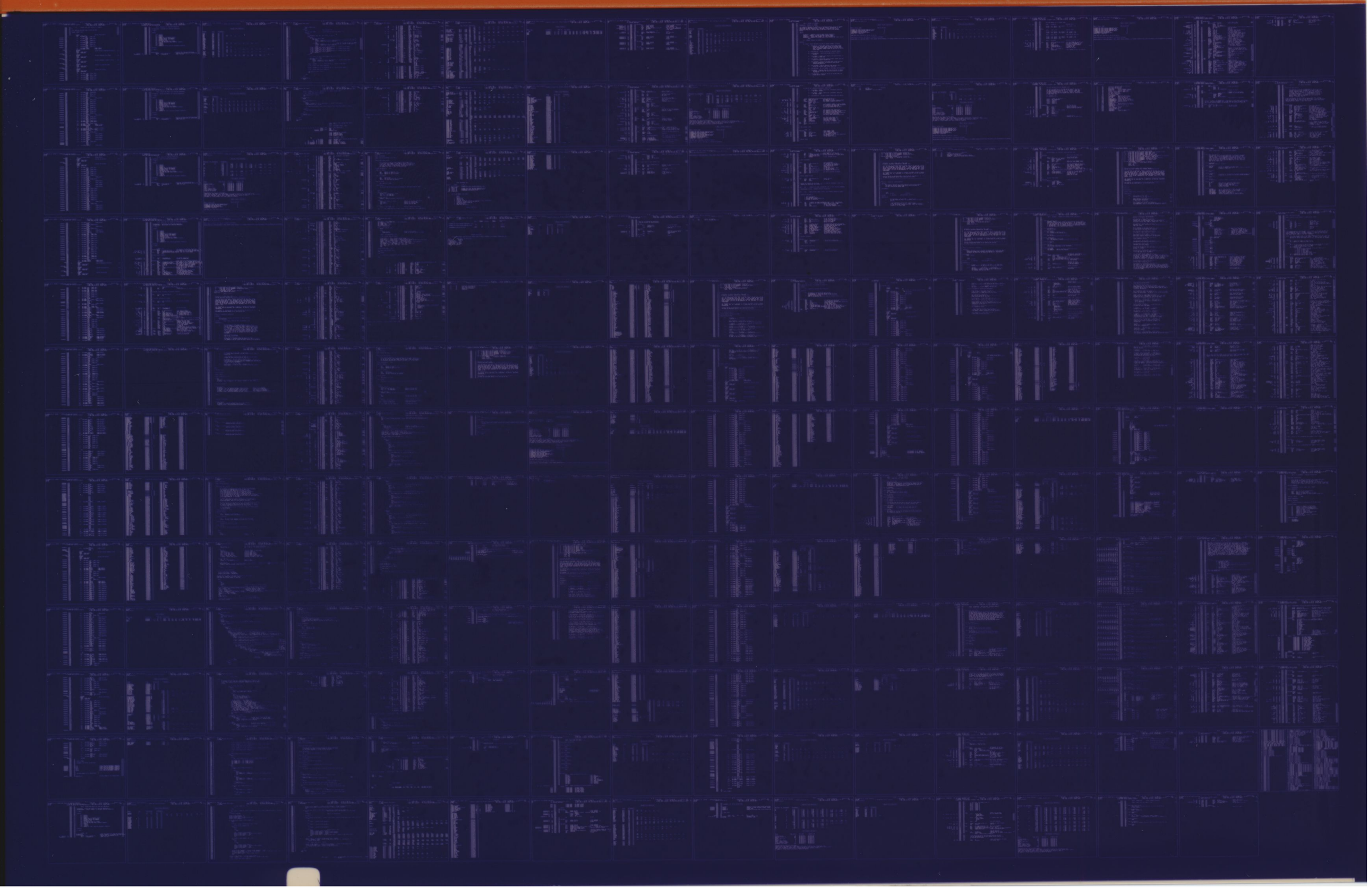


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
2 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

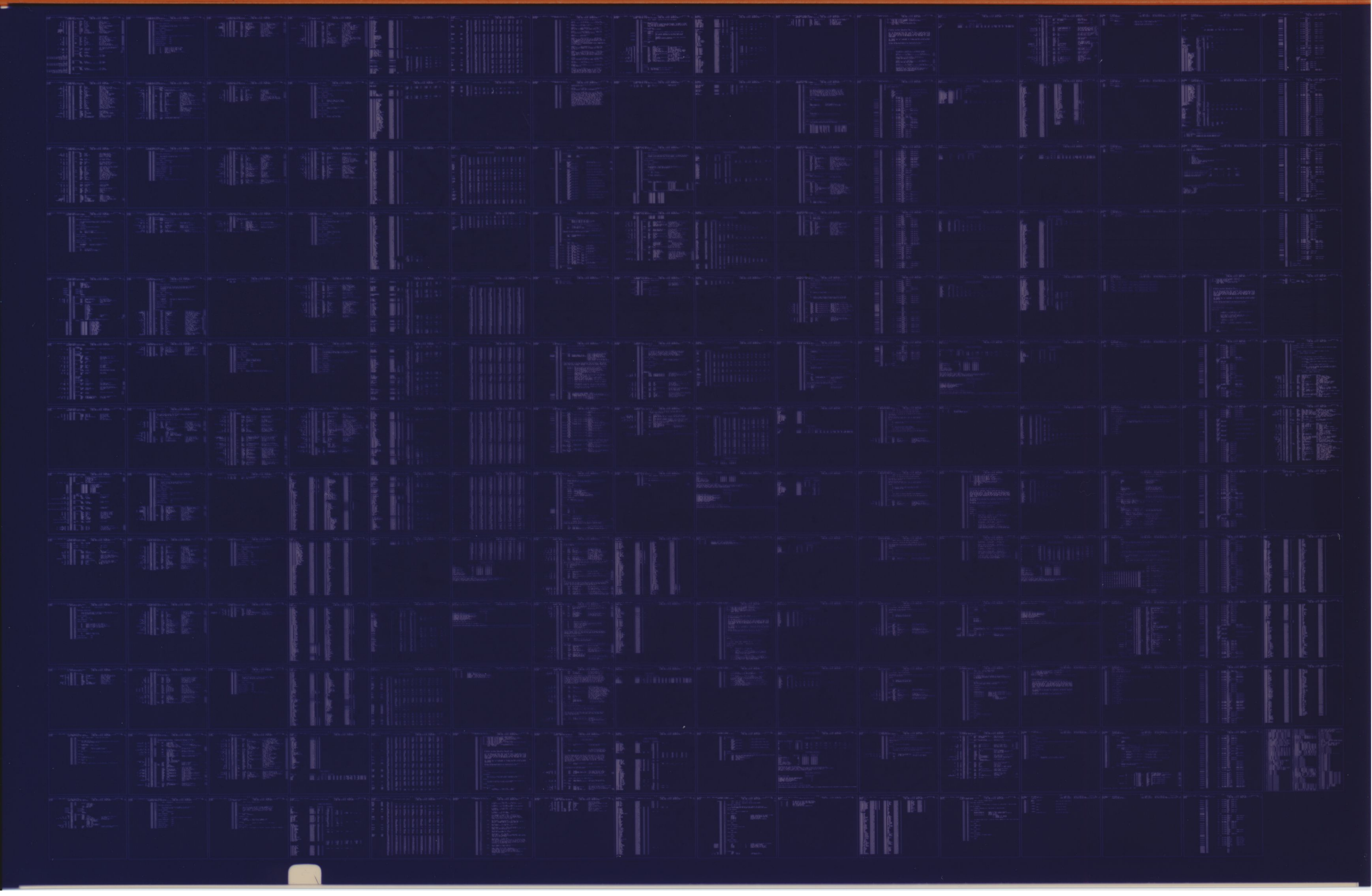


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
3 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

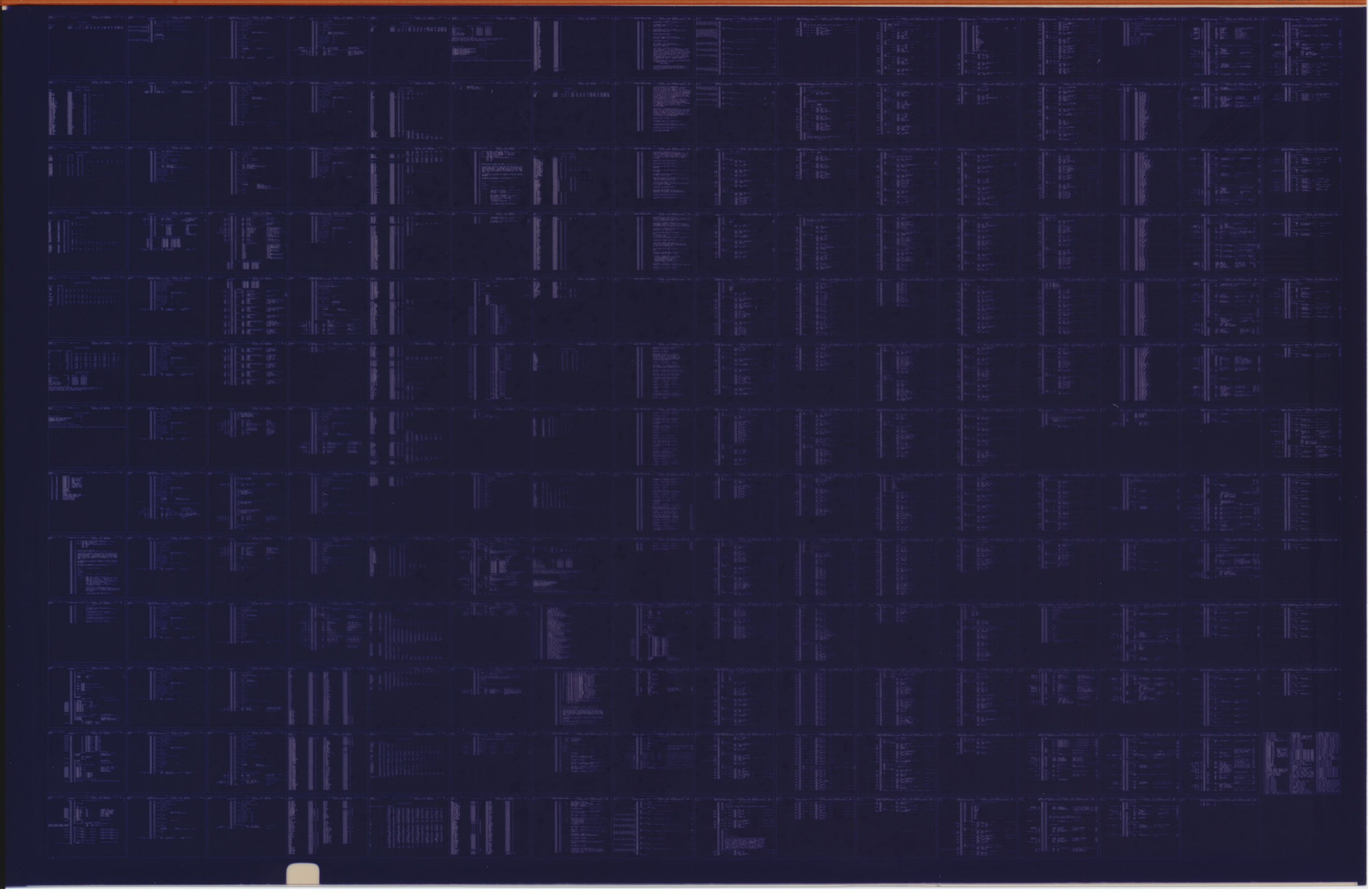


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
4 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

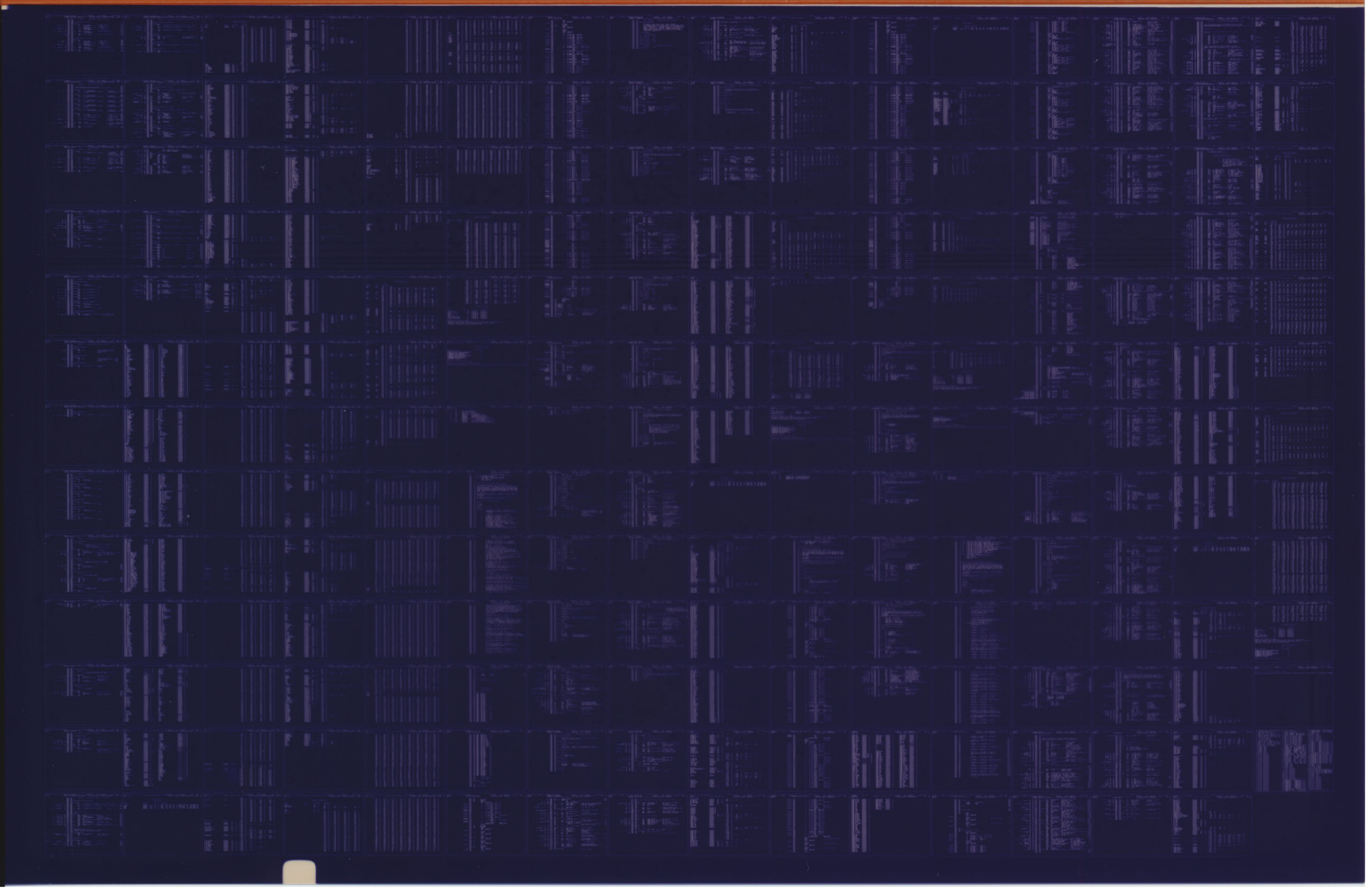


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
5 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

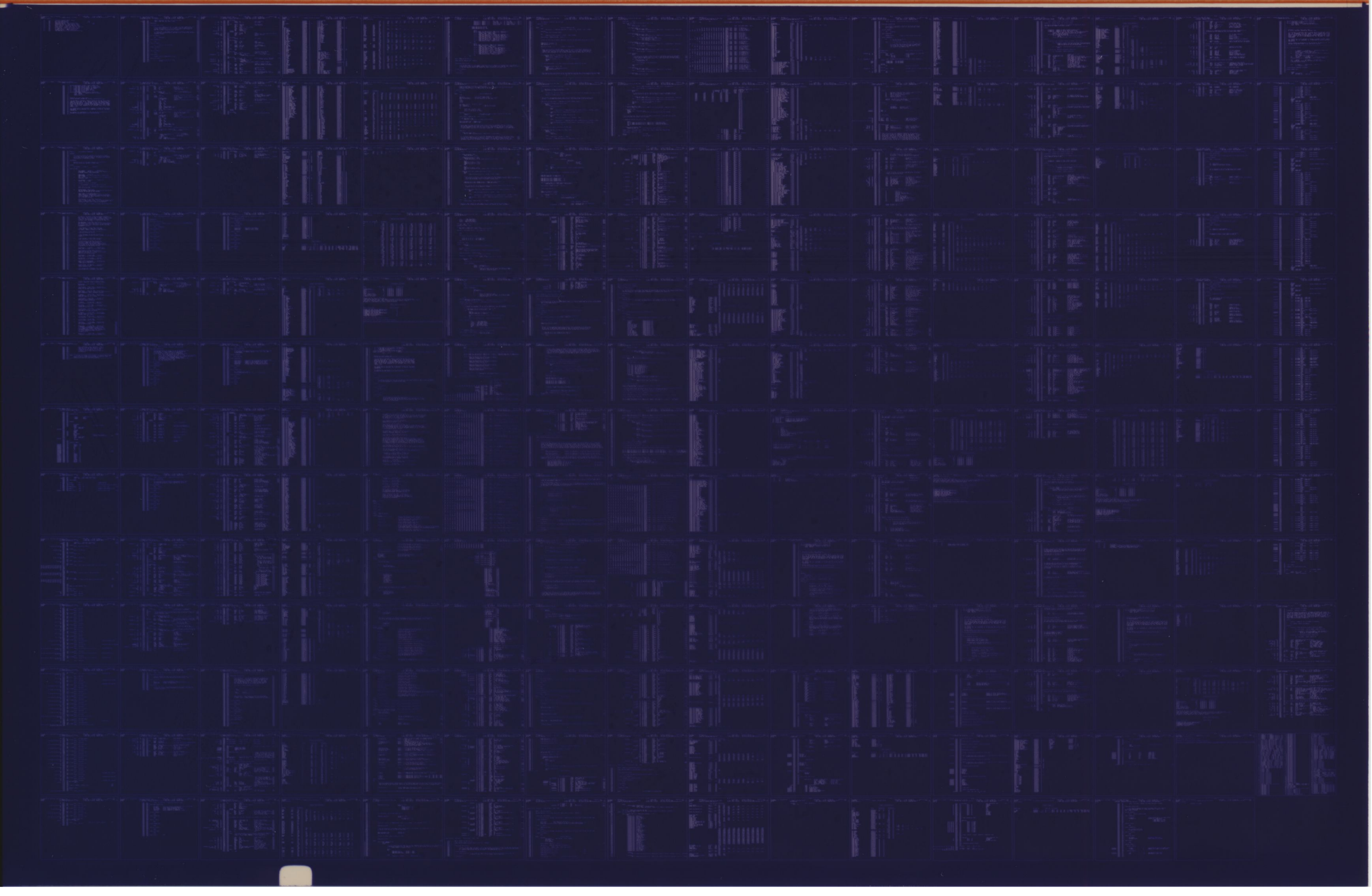


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
6 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

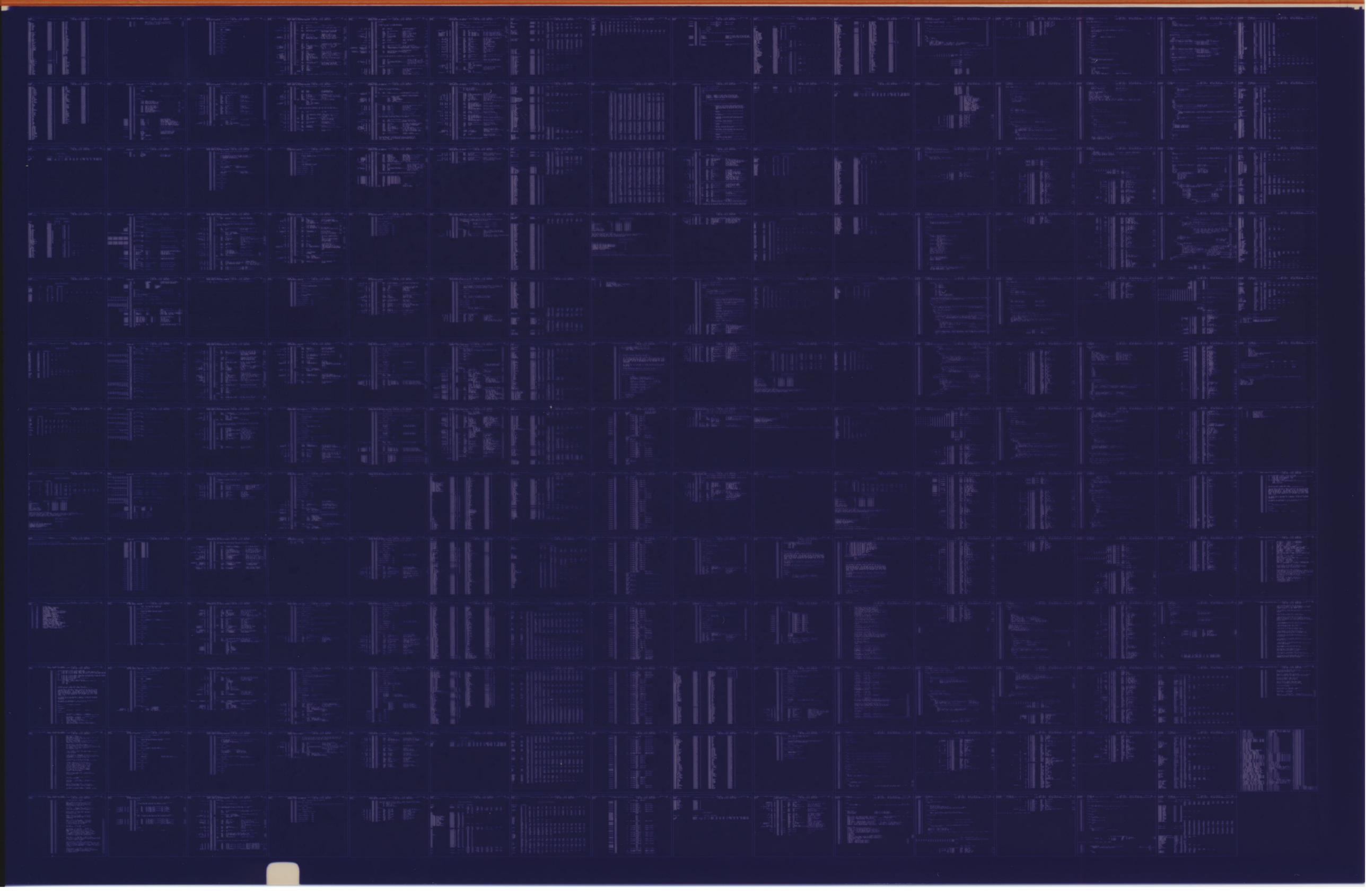


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
7 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

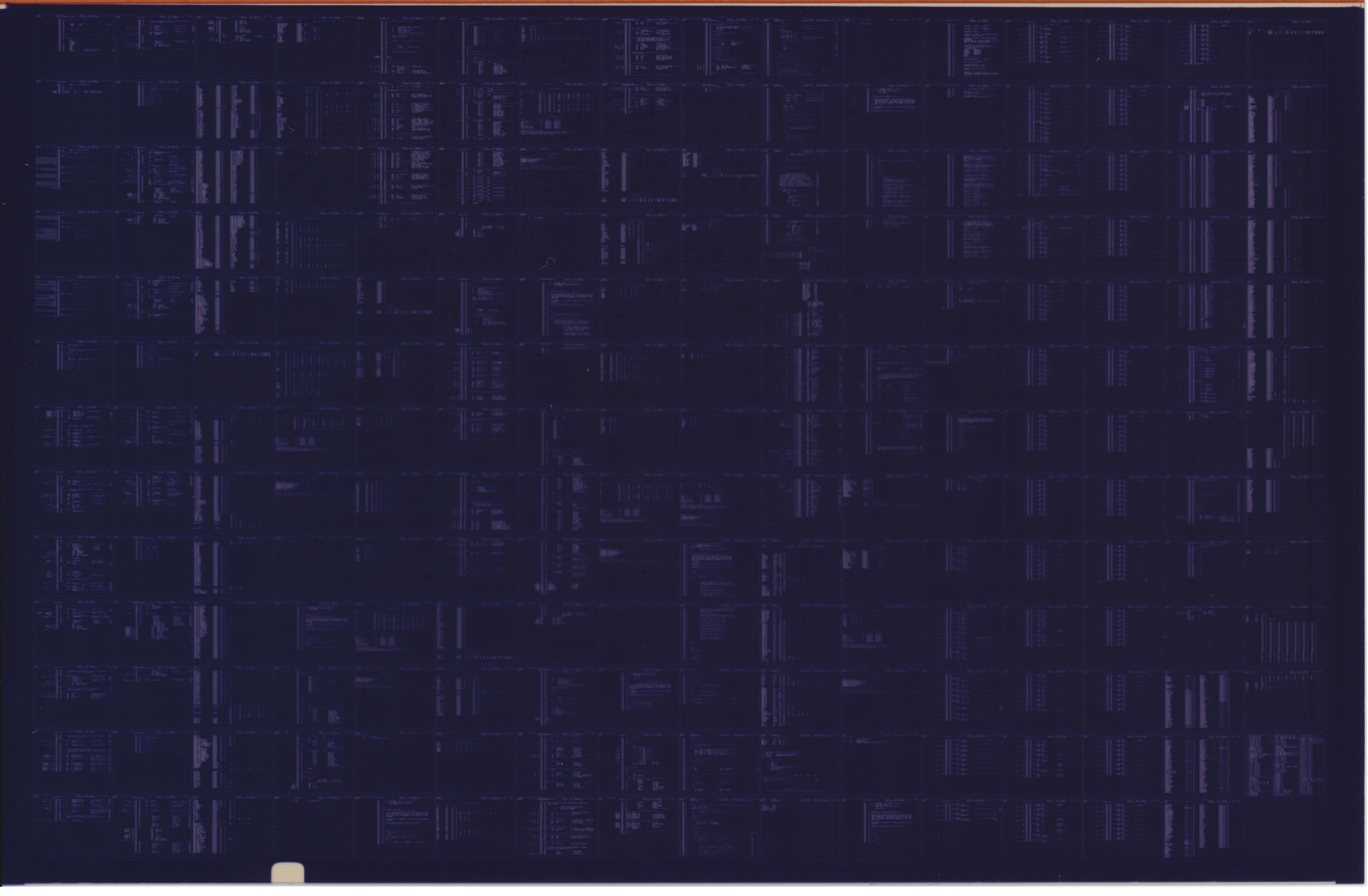


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
8 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

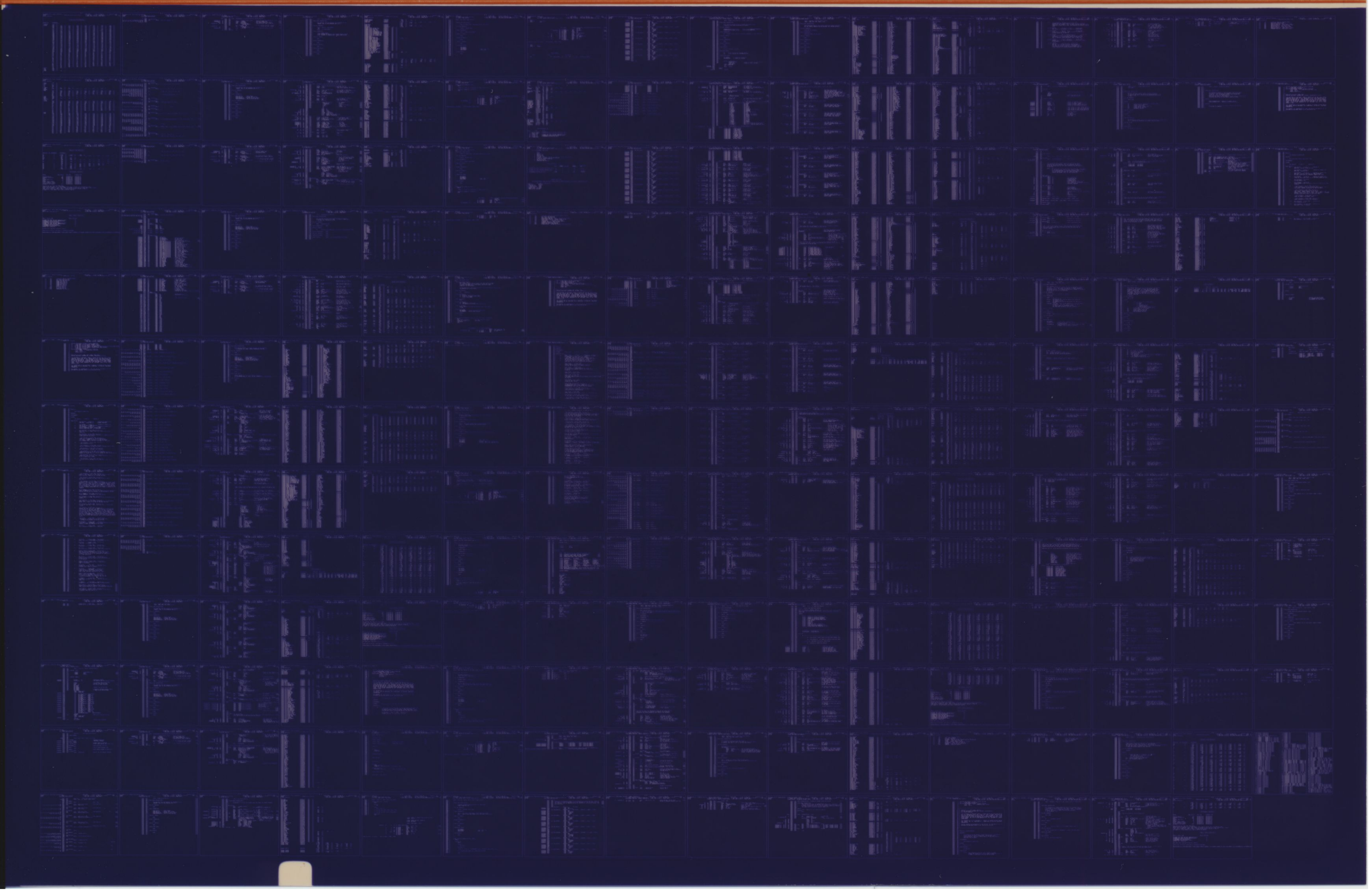


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
9 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

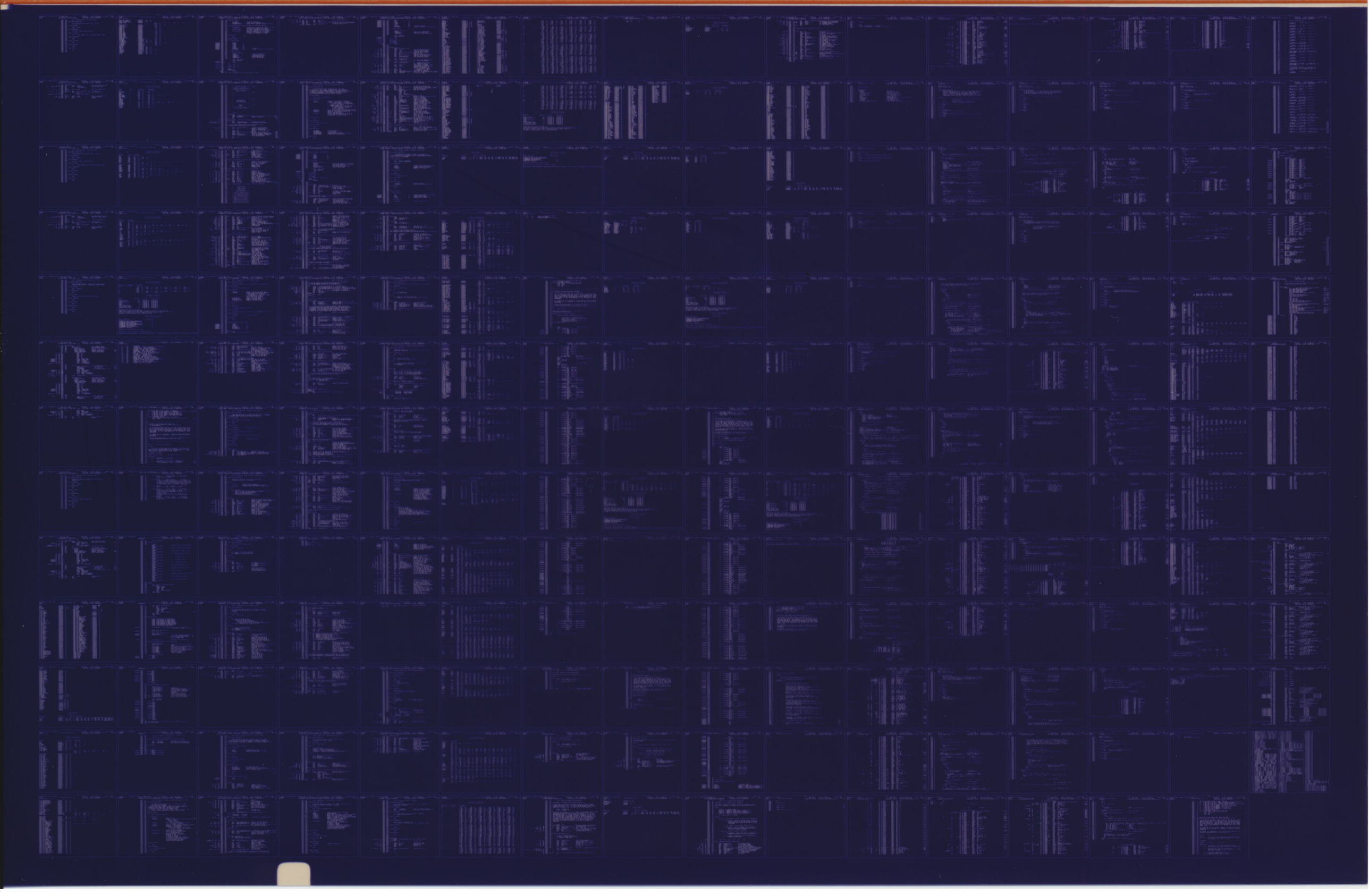


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
10 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

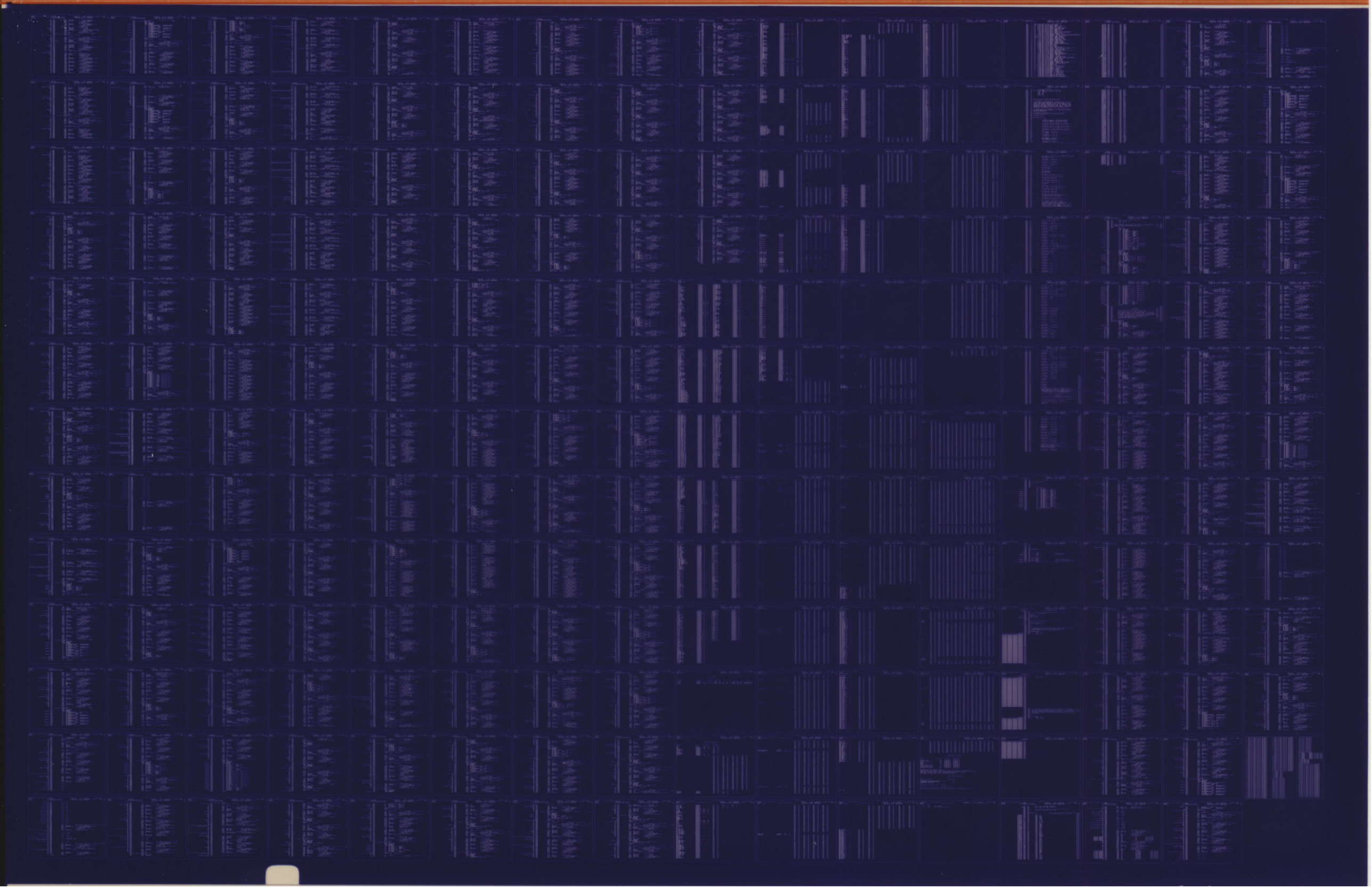


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
11 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

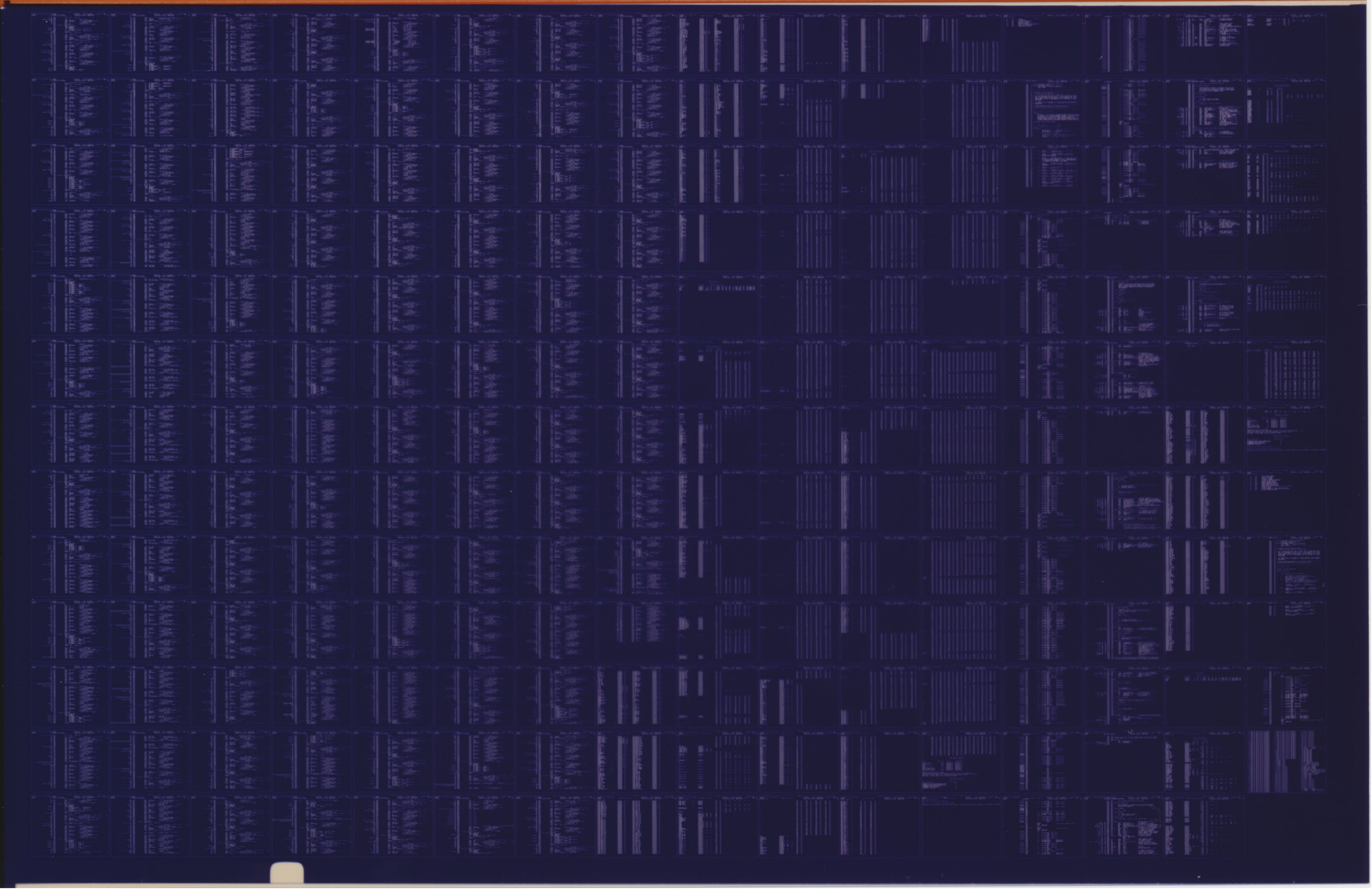


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
12 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

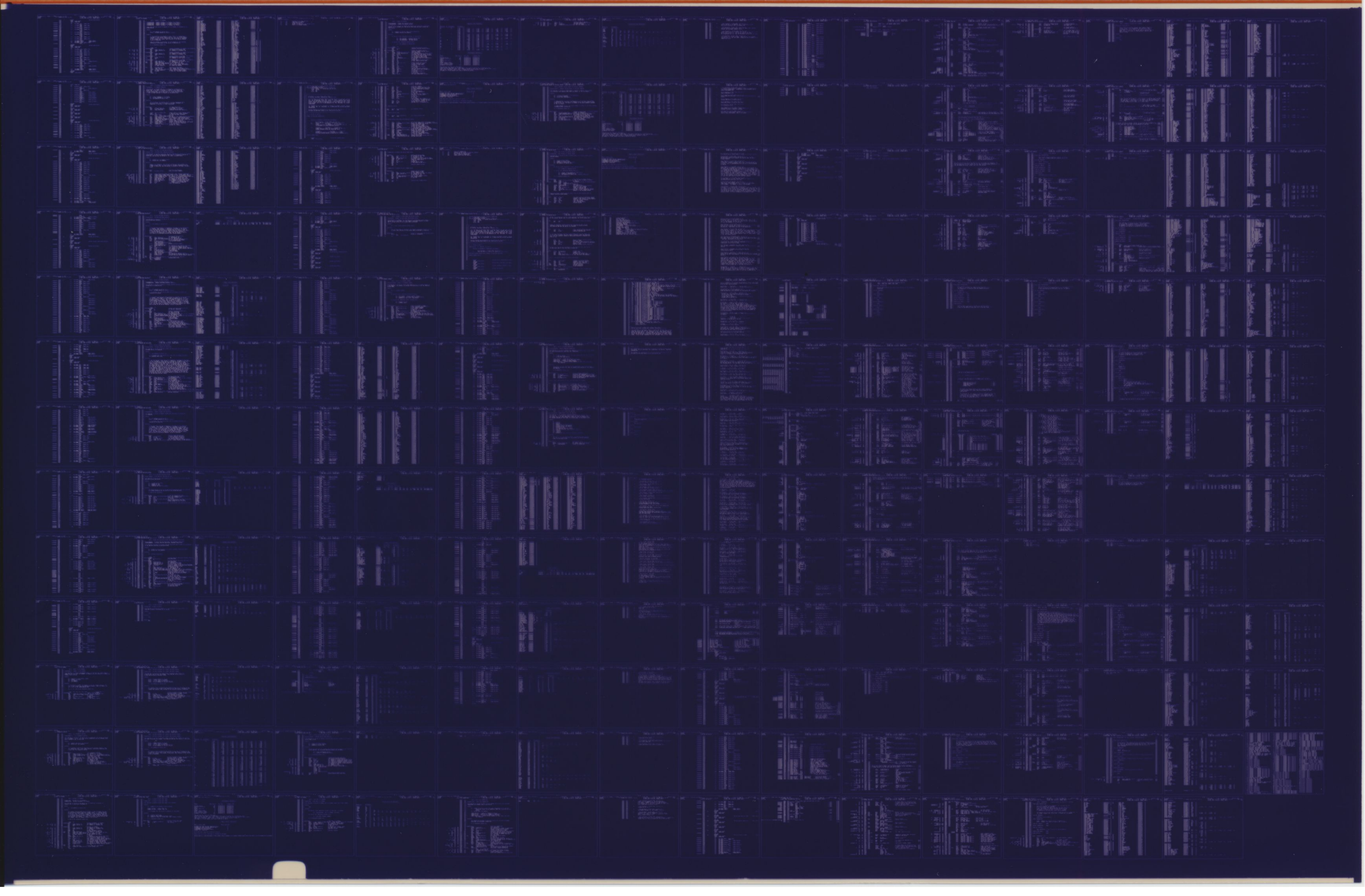


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
13 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

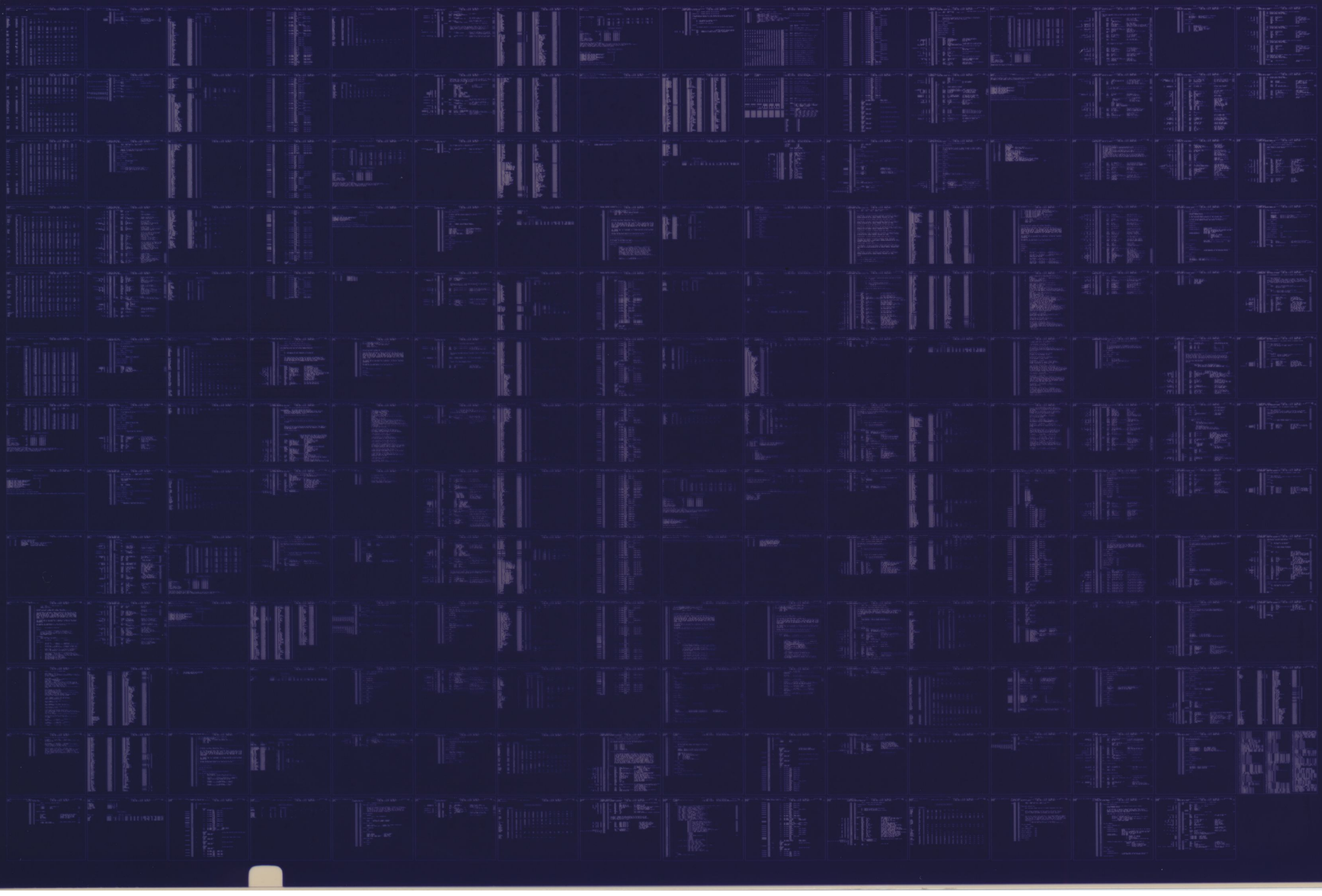


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
14 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

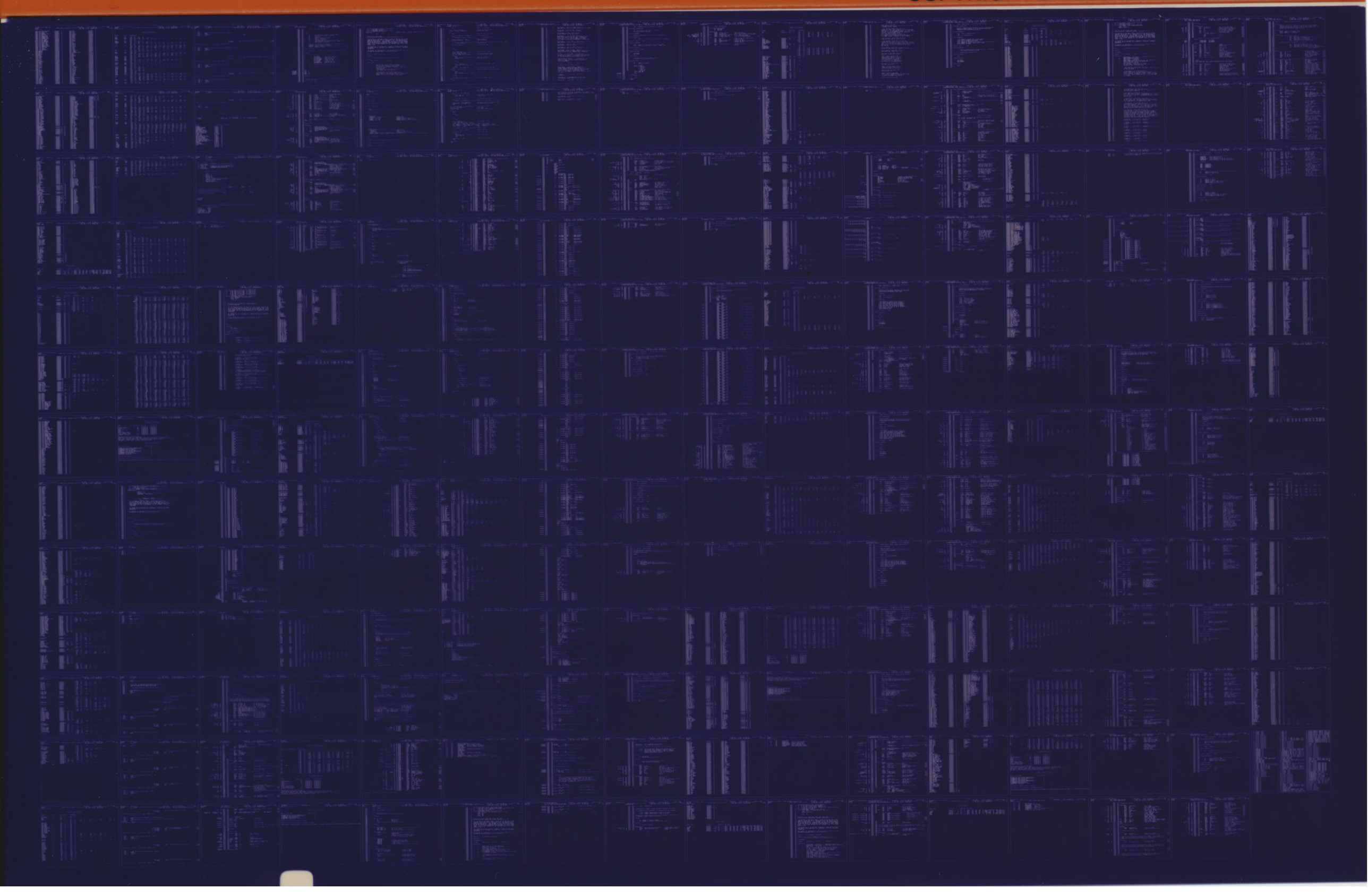


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
15 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

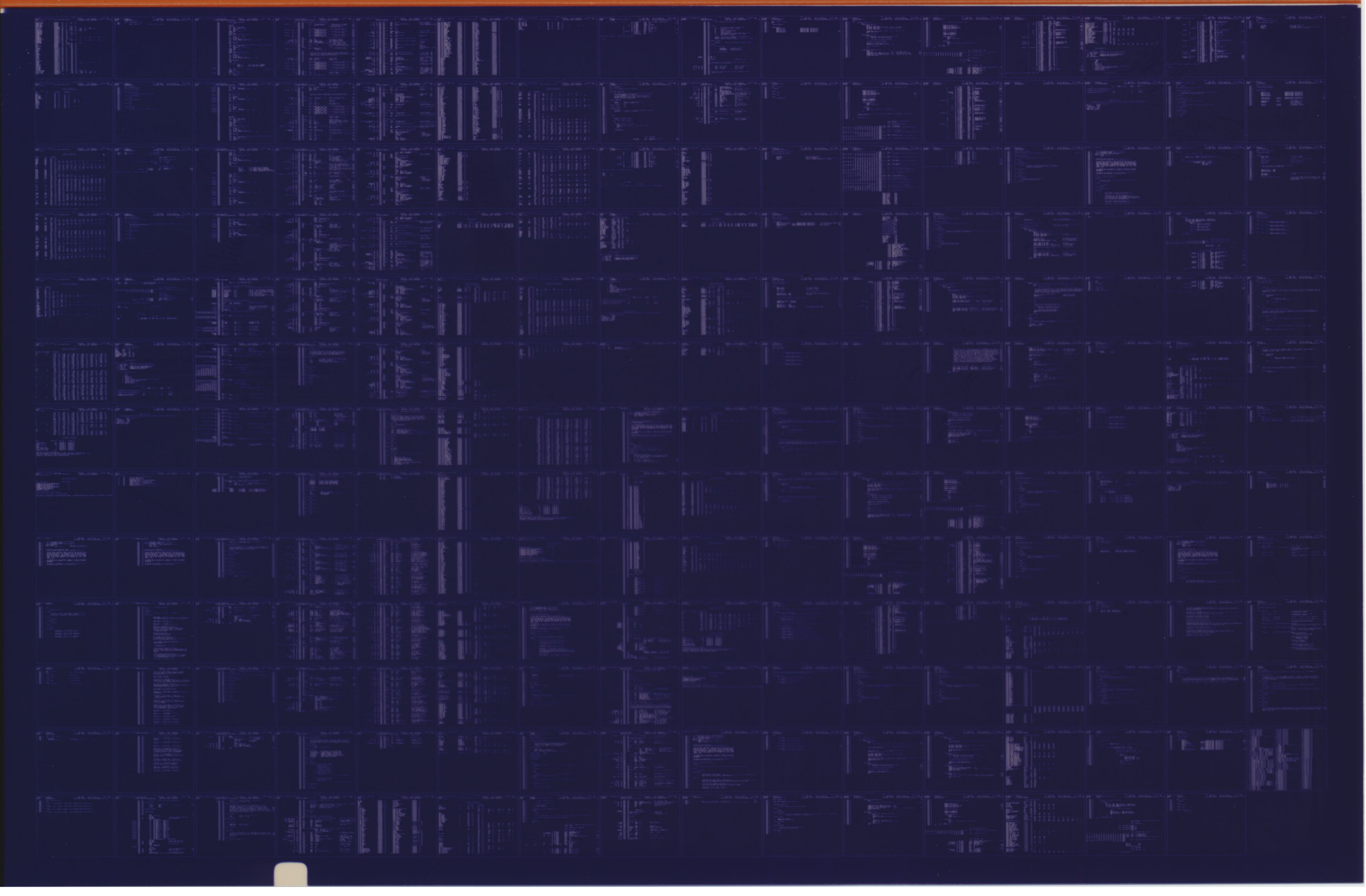


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
16 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

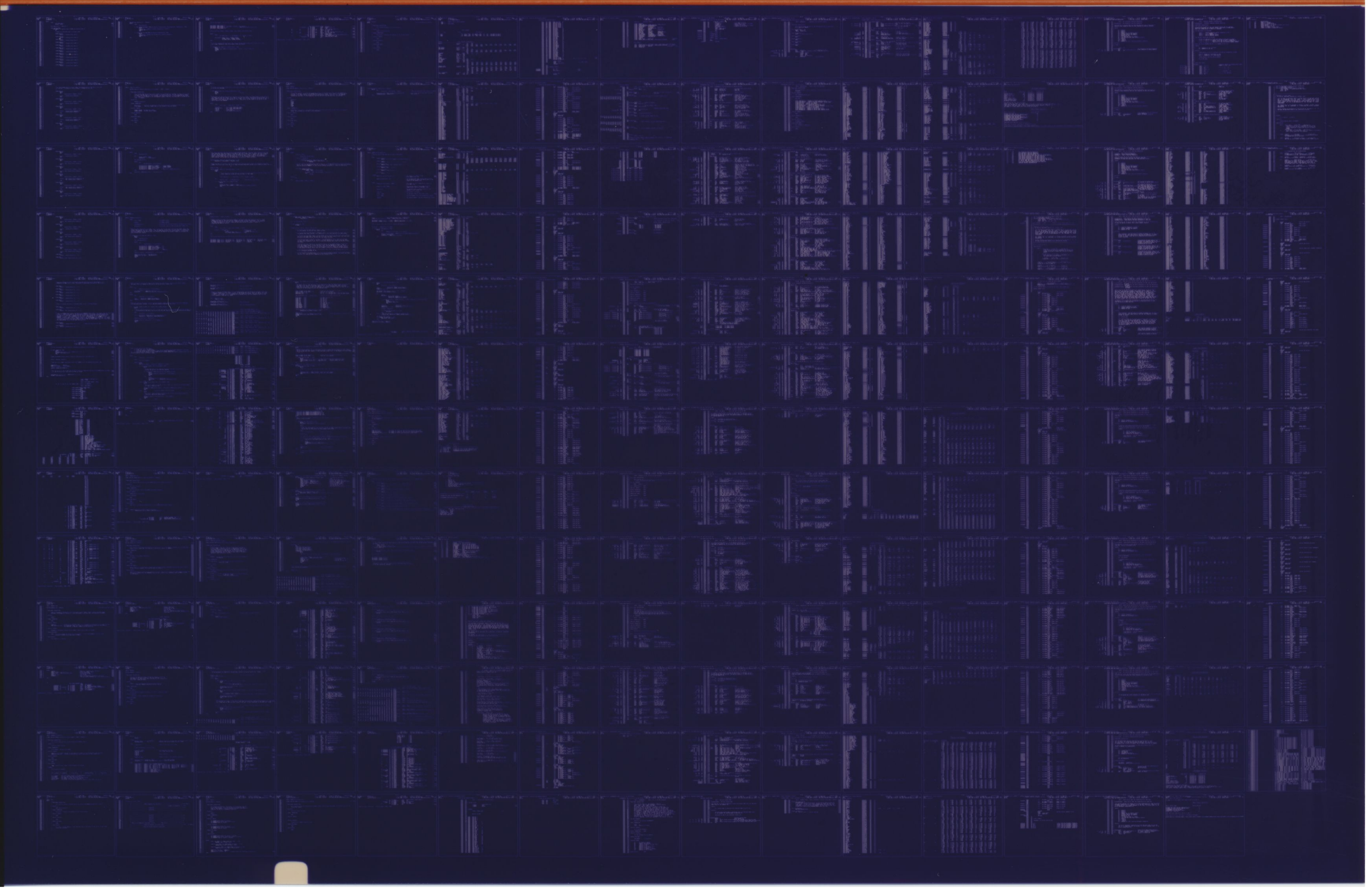


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
17 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

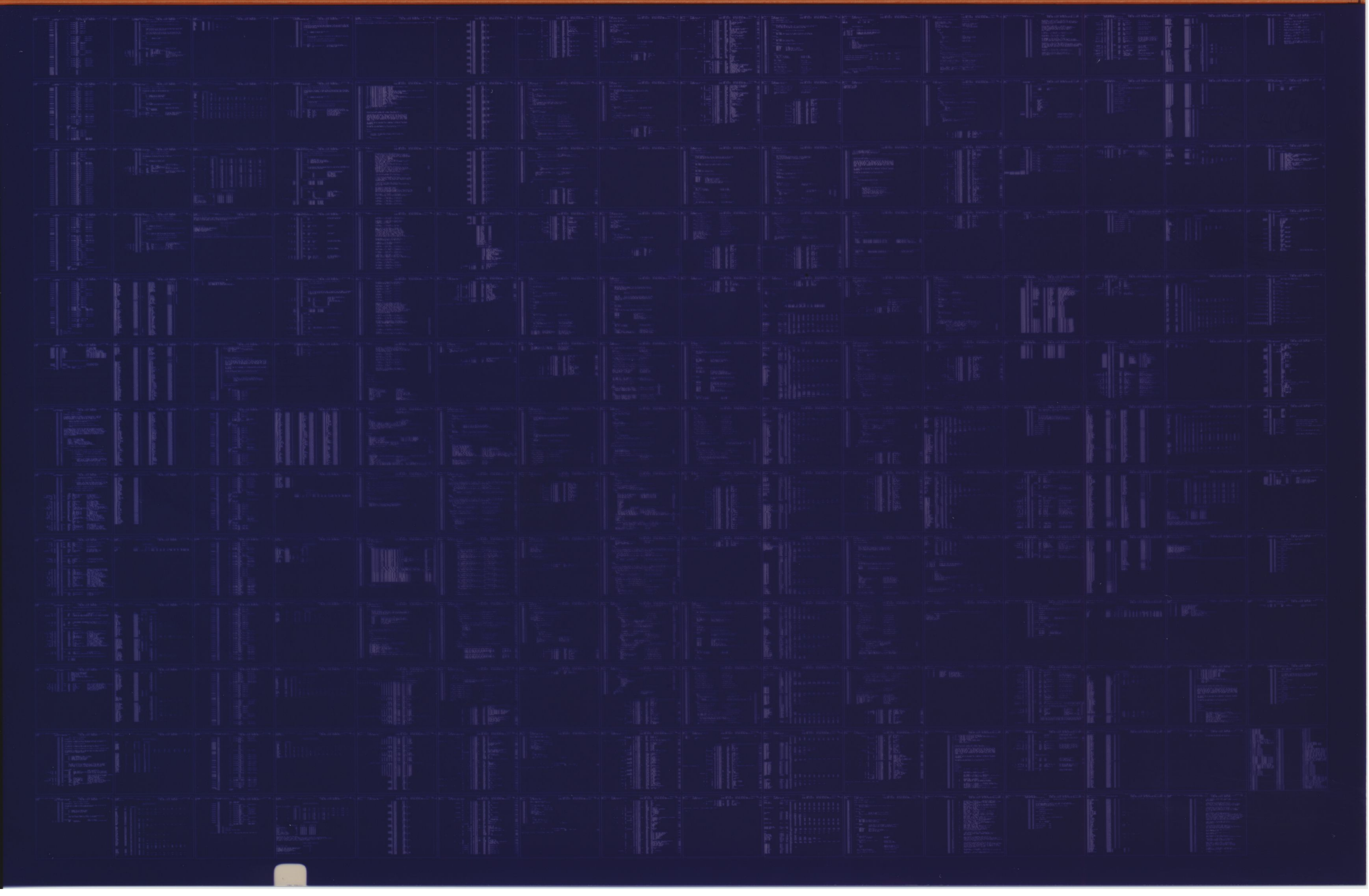


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
18 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA

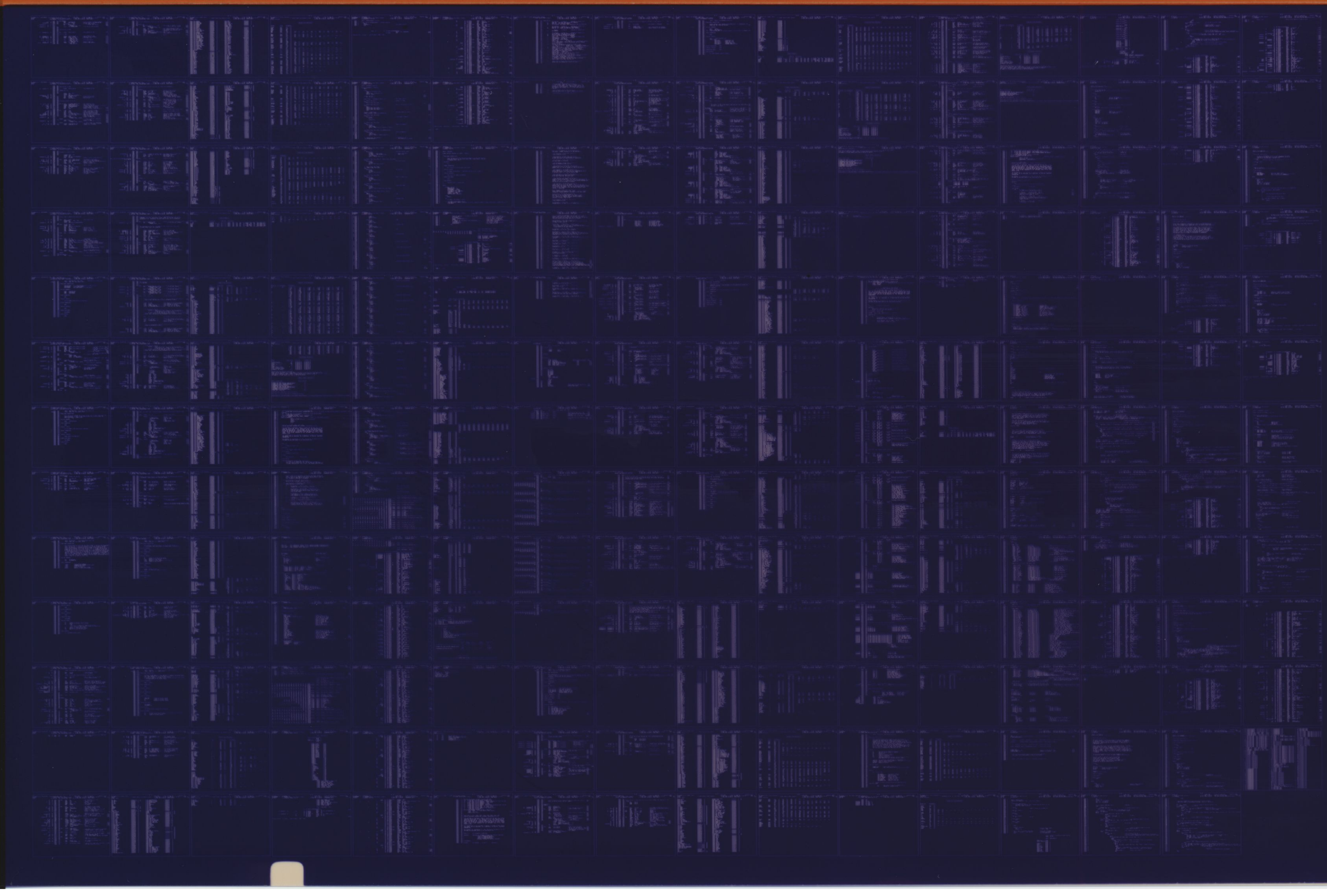


ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
19 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA



ENSAA-10.10

VAX 11/730
DIAG SUPERVISOR

EP-ENSAA-DL-10.10
20 OF 20 MAR 1988
COPYRIGHT© 1978-87

digital
MADE IN USA



PRODUCT ID: ZZ-ENSAA-10.10

PRODUCT TITLE: VAX Diagnostic Supervisor

DECO/DEPO: 10.10

DATE: 21-DEC-1987

DEPARTMENT: VAX DIAGNOSTIC SYSTEM SOFTWARE

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

PRODUCT ID: ZZ-ENSAA-10.10

PRODUCT TITLE: VAX Diagnostic Supervisor

DECO/DEPO: 10.10

DATE: 21-DEC-1987

DEPARTMENT: VAX DIAGNOSTIC SYSTEM SOFTWARE

COPYRIGHT (C) 1978,1987

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

IDENTIFICATION

PRODUCT CODE: ZZ-ENSAA-10.10

PRODUCT TITLE: VAX DIAGNOSTIC SUPERVISOR

PRODUCT DATE: 21-DEC-1987

DEPARTMENT: VAX DIAGNOSTIC SYSTEMS SOFTWARE

COPYRIGHT (C) 1987
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERED.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.

Diagnostic Supervisor release notes V5.3

A buffer in show device was increased to 256 characters.

Channel services was fixed to disable interrupts before releasing the vectors and the INVALIDATE function made to work.

SET MM ON, SET MM OFF, and SHOW MM commands were added to enable and disable memory management. At the same time DS$MMON and DS$MMOFF were modified to prevent the program from disabling memory management if the operator enabled it.

SYS$SETIMR and SYS$CANTIM were enhanced to change mode to kernel before doing MTPR's or disabling interrupts. This resolved some problems caused by interrupting programs that had set the processor mode to user.

Cleanup was enhanced to reset memory management and re-initialize the SCB after the program cleanup is complete.

Dummy entry points were added to $DS_DSSDEF and the entry vectors to accomodate SYS$LKWSET, SYS$SETAST, SYS$SETPRI, SYS$SETRWM, SYS$ULKPAG, SYS$ULWSET.

SYS$FAO used in stand-alone was modified to fix problems with !*? directives.

The known device data base was updated to include DL11, LP25, RX02, RX211, TM78, TS04, TS11, TU58, and TU78 support.

The descriptor for KA780 has changed, the optional arguments are now G-floating, H-floating, WCS last address, and accelerator type.

An error is now generated if the processor type specified in the system identification register is not 1, indicating an 11/780.

Error text and status codes are returned if the supervisor is run from a command file. The next release will support running in batch.

DS$LOAD has been enhanced to allow programs to read parts of files indicated by the starting block number argument. The name of the console storage device is now always CSan.

The count of loops on subtests has been fixed to correctly reflect the number of passes thru the specified subtest.

The logout for SBI WRITE timeouts now includes the timeout address register.

The ABORT command and DS$ABORT now reset the stacks and reset the processor mode before doing program cleanup.

Diagnostic Supervisor release notes V5.4

Release 5.4 of the diagnostic supervisor has been made. ESSAA.EXE 5.4;214 and ECSAA.EXE 5.4;121 represent 11/780 and 11/750 supervisors. Relevent files can be found on PROTO 3 in Tewksbury as follows:

```
SYS$LIBRARY:DIAG.MLB;611
[DS]ESSAA.EXE;214     11/780 STAR supervisor
[DS]ECSAA.EXE;121     11/750 COMET supervisor
[DS]EVQDB.EXE         QIO driver for RP04,5,6
[DS]EVQDL.EXE         QIO driver for RL02
[DS]EVQDR.EXE         QIO driver for RM80,RP07,RM03,RM05
[DS]EVQDM.EXE         QIO driver for RK06,RK07
[DS]EVQXF.EXE         QIO driver for DR780
[DS]EVQTM.EXE         QIO driver for TM03/TE16,TU77,TU45
[DS]EVQTS.EXE         QIO driver for TS11
[DS]DIAGBOOT.EXE      Secondary Bootstrap
```

Please use these components in testing programs. Needless to say if any problems are encountered please report them to me immediately so their solutions can be included in the next release.

Release notes

- Support for drivers was upgraded to reflect VMS 2.0 interface changes. Loadable drivers have been re-released to reflect Release 2.0 of VMS.

- The supervisor can be run from a VMS command proceedure. When the supervisor is running under VMS it uses RMS to access SYS$INPUT and SYS$OUTPUT.

- Bug fixed to remove supervisor error when supervisor is started again after interrupting a script.

- The APT interface will now generate an error if no PTEXT is found when a process PTEXT command is given.

- A buffer in show device was increased to 256 characters.

- Internal processor registers can be examined with EXAMINE Pxx, where xx is a register number in the current radix. If only the decimal values are available Try EX P%D127 to get register ^X7F. Internal registers can be changed with the DEPOSIT command.

- Channel services was fixed to disable interrupts before releasing the vectors and the INVALIDATE function made to work. Also SETMAP now sets and clears all map registers and allows all to be used for transfers.

- Support was added for the RL02 as a boot device.

- Time conversion routines were changed to fix loss of precision in hundredths.

- SET MM ON, SET MM OFF, and SHOW MM commands were added to enable and disable memory management. At the same

time DS$MMON and DS$MMOFF were modified to prevent the program from disabling memory management if the operator enabled it. Also the return code in R0 now indicates the previous state of memory management and an error return if DS$MMOFF was trying to disable memory management and the operator had enabled it. If memory management is enabled the stack for each mode is write protected against lower access modes.

- DS$ASKSTR now returns 0 in R1 if the default answer was given to a query.

- SYS$SETIMR and SYS$CANTIM were enhanced to change mode to kernel before doing MTPR's or disabling interrupts. This resolved some problems caused by interrupting programs that had set the processor mode to user.

- Cleanup was enhanced to reset memory management and re-initialize the SCB after the program cleanup is complete.

- Corrected errors induced when more the 2 megabyte of memory was present.

- Dummy entry points were added to $DS_DSSDEF and the entry vectors to accomodate SYS$LKWSET, SYS$SETAST, SYS$SETPRI, SYS$SETRWM, SYS$ULKPAG, SYS$ULWSET.

- SYS$FAO used in stand-alone was modified to fix problems with !*? directives.

- The known device data base was updated to include DL11, LP25, RX02, RX211, TM78, TS04, TS11, TU58, and TU78 support. The descriptor for KA780 has changed, the optional arguments are now G-floating, H-floating, WCS last address, and accelerator type.

- An warning is now generated if the processor type specified in the system identification register is not correct.

- DS$LOAD has been enhanced to allow programs to read parts of files indicated by the starting block number argument. The name of the console storage device is now always CSan.

- The count of loops on subtests has been fixed to correctly reflect the number of passes thru the specified subtest.

- The logout for SBI WRITE timeouts now includes the timeout address register.

- The processor mode and and all stacks and registers are now reset before the users cleanup code is executed.

1.0 BUFFERED I/O SUPPORT

The supervisor now includes support for QIO drivers using buffered I/O (such as line printers).

2.0 HELP FACILITY

The new Supervisor includes a HELP facility. It is invoked via the "HELP" command. In general it is identical functionally with the VMS HELP command. Additionally, it supports gathering HELP on diagnostics via typing the diagnostic code (ES, EV, EC pre-fix followed by diagnostic name, as "EVRAA", etc.) as the first keyword after HELP (e.g., "HELP EVRAA"). In standalone mode the HELP command will function off of any load media (including magtape and console media). In user mode however, it will not function from magtape or the console media, as these are not fully supported by VMS RMS (see below, description of Supervisor RMS).

3.0 LOAD AND SCRIPT

The Supervisor LOAD (which loads files for the commands RUN and LOAD as well as for diagnostics via the $DS_LOAD service call) and SCRIPT file handling services have been modified to use RMS file services.

4.0 MAGTAPE

The supervisor now supports magtape (TM03 only) as load media. TS-11 and TU-78 will be supported in a future release.

5.0 RMS IMPLEMENTATION

The diagnostic supervisor implementation of RMS is similar to the VMS implementation, but is specifically adapted to the standalone DS environment.

DISCLAIMER

Since there are currently no diagnostics which use RMS services, we can not guarantee 100% functionality of RMS code, though we have attempted thorough QA testing. However, services within the Supervisor which use RMS will function correctly.

Console Device Naming

Previous to release 6.0, due to a file name-handling loophole in the file load routine,

it was possible to specify the console device as "CS:". This is not consistent with Supervisor convention, which requires complete device names (i.e., "DMA0:" not "DM:" even if only one RK06/07 is attached). In Version 6.0, "CS:" will not function on COMET (11/730); it will fail when the TU58 driver attempts to access a non-existent drive. The solution is that full device names MUST be typed. In future versions of the Supervisor, this naming "loophole" will be closed, and omitting the controller letter or drive number in the console device specification will return an illegal device error (RMS$_DEV).

5.1 User Mode

To consistently support RMS in user mode (where RMS calls are passed through to VMS), the Supervisor SET LOAD command will function correctly in user mode (as will SHOW LOAD). This is done by translating or reassigning SYS$DISK for device names, and via an RMS housekeeping routine to alter default directory. The Supervisor exit handling routine will restore the original values: however for this to occur, the Supervisor must be terminated either by ^Z or by a ^Y, EXIT sequence. If the (VMS) STOP command is used, the original default directory is not (and CAN not be) restored.

5.2 Differences Between DS RMS And VMS RMS

Supervisor RMS functionality not duplicated by VMS:

- Supervisor RMS supports files on the console media (floppy or TU-58). These files are in RT-11 format, but the Supervisor RMS $GET facility will provide file records as if the file were an ODS CR-attribute, variable length record file. The FAB's RFM field (FAB$B_RFM) will have the value 4.

- Random by RFA (record's file address) access is supported for magtape as well as for conventionally random-access devices. VMS does NOT support magtape random access. In general, random file access should not be used by any level 2 diagnostic, since if the load device is magtape, the function is unsupported in user mode.

VMS RMS functionality not duplicated by Supervisor:

- The Supervisor does not support NAM blocks in any form.

- The Supervisor supports only one form of XAB (the FHC XAB, returning file header characteristics). Any other XAB in the FAB's XAB chain will be ignored, though no error will result. This is because the information is not meaningfull in the diagnostic environment.

- In the FHC XAB, the LRL (longest actual record length) field is not supported. It is meaningful only for ODS files in any case. It will be set to maximum record length always.

- Only sequential files are supported. An attempt to open a file with INDEX or RELATIVE organization results in error RMS$_ORG (illegal file organization).

- Only READ-type functions are supported, since the supervisor does not support write access to any media. To be specific, the functions which the Supervisor RMS package implements is:

  1. $OPEN (access a file on media and initialize FAB)

  2. $CONNECT (Define a record stream for access to a file, initialize RAB)

  3. $GET (Copy a record within the file to a user buffer)

  4. $READ (Unformatted read from file, beginning on block boundary)

  5. $DISCONNECT (Close out a record stream)

  6. $CLOSE (Close out any record streams associated with FAB, and de-access file attached to FAB)

- The bits in the FOP and ROP fields are not significant, except for the BIO bit in the ROP field (specifying block I/O via the $READ call).

- Asynchronous file operations are not supported.

- RMS operations to the terminal are not supported.

- A user may have up to three files open simultaneously.

- Multibuffering and multiblocking are not supported.

- Random access by key or record number is not supported.

- The RAB TMO (time out) field is ignored.

6.0 BUG FIXES

The following Supervisor "bugs" have been fixed for the 6.0 release:

- Channel services incorrectly calculated the number of map registers required for a transfer.
- The operator request routine incorrectly defaulted parameters when prompting was disabled.
- The COMET machine check log-out was incorrect.

1.0 BUG FIXES

- BUG: "EXAMINE PSL" with "/WORD" or "/BYTE" would use only low-order bits of PSL, but the high-order fields would still be decoded and typed as 0.

  FIX: PSL is now unaffected by "/WORD" and "/BYTE"; it is always longword.

- BUG: Previous to Version 6.0, the console storage device could be specified by "CS:". This is at odds with Supervisor convention which requires full device name given in attach command. It results from an accidental "loophole" in that only the first two characters were checked, and an unspecified unit number was converted to 0. In Version 6.0, the unit number conversion was (also accidentally) removed, and using the "CS:" form to Version 6.0 causes a device error on COMET (the STAR console load routine does not use unit number) when the TU58 attempts to access a non-existant (negative) unit.

  FIX: The "CS:" short form is no longer allowed. It will result in an "invalid or illegal device" RMS error (RMS$_DEV) during name parsing. You must type the full device name (e.g., "CSA0:").

2.0 ENHANCEMENTS

- The Supervisor now supports a DIRECTORY command. The format is:

  ```
  DIRECTORY [device:][[directory]]
  ```

  Examples:

  ```
  DS> DIRECTORY DBB3:[SYSMAINT]
  DS> DIR MTA0:
  DS> DIR CSA0:
  DS> DIR [DS]
  DS> DIR DBB3:
  DS> DIR
  ```

  Device and directory default to the current load path (original default is "<load device>: [SYSMAINT]" or last SET LOAD specification). All load media are supported. The format is identical to the standard default VMS directory format. There are no qualifiers. Only file name and version (if applicable, i.e., not on RT-11 media) are typed, and there is no provision for typing dates, sizes, etc.

- To support the AUTOSIZER program, the supervisor's ATTACH command processor has been made externally callable. The entry point is DS$ATTACH

(cmdline,[prompt]) where cmdline is the address of a string descriptor for the ATTACH command to be parsed, and prompt is the address of a descriptor for a buffer (MUST be at least 1 byte long if prompt is given!) to receive the ASCIC text with which to prompt the operator if the command line is incomplete or incorrect (on error, the prompt string is set, and the cmdline descriptor length is decreased to include only the correct part of the input line). The prompt argument is optional and in general is useless except within the Supervisor.

Format:

```
$DS_ATTACH (cmd, [pmt])          ! Bliss
$DS_ATTACH_x    cmd, [pmt]       ; Macro
```

o The Supervisor HELP routine has been made callable, to allow programs implementing a conversation mode to include a HELP command. You pass the address of a descriptor of a command string; this string is just what you'd type on a Supervisor HELP command line, except leave out the "HELP" keyword. You can access the program's usual HELP file (preferable) by prefixing whatever keys the operator types with the program name. Keep in mind that HELP will fiddle with the adapter and controller, etc., in the path to the load device.

Format:

```
$DS_HELP (keylst)                ! Bliss
$DS_HELP_x      keylst           ; Macro
```

o User control of ^C

. In some diagnostics, it may be necessary to call DS services which do implicit "$DS_BREAK" calls during sections of code which must not be interrupted by "^C". To solve this problem, new functionality has been added to the $DS_CNTRLC service. A second argument, DISABL, has been implemented. The value of this argument defaults to "0", and the DS$CNTRLC routine accepts argment lists of only one argument to maintain compatibility with currently released diagnostics. If the low bit (bit 0) of DISABL is SET, ^C will be ignored by the Supervisor. If a user ^C handler is set, it will not be called, nor will the Supervisor itself take any special action beyond setting it's ^C flag bit. Note that if ^C recognition is disabled, and a ^C character is typed, the ^C will be honored as soon as ^C recognition is re-enabled. Therefore, the ^C is effectively postponed beyond the non-interruptable segment of code, but is not lost.

Format:

```
$DS_CNTRLC ([astadr], [disabl]) ! Bliss
```

```
$DS_CNTRLC      [astadr], [disabl]      ; Macro
```

. Previously, the user ^C handler routine had no value. Now, the user handler has the option of telling the Supervisor to handle the ^C normally even though there was a user handler defined. To do this, the user's handler should return a failure status code (R0 bit 0 clear). If the user handler returns a success code, the Supervisor will do nothing further.

o It is possible for messages to be typed out asynchronously, by AST routines for example. This typeout may occur while the operator is keying in commands and/or data. In such a circumstance, VMS will retype the input line automatically (as if a ^R had been typed), but the Supervisor would just let the line "dangle". Now, the Supervisor will perform the "^R"

3.0 NEW LOAD DEVICE SUPPORT

o Supervisor RMS (and therefore all Supervisor file utilities including HELP, DIRECTORY, DS$LOAD, RUN, and LOAD) now will function with the TS-11/TS-04 tape drive sub-system. This addition is transparent to users of the utilities, syntax and programming-wise.

- A user control-C handler returning failure status under APT would cause the Control-C to be ignored, since status information was cleared before the user routine was CALLed. This has been fixed by deferring the status clear.

- DIRECTORY - due to an inconsistency in string descriptor handling, access violations could occur during DIRECTORY depending on previous stack contents. Specifically, if a HELP command with keywords was entered previously, DIRECTORY would fail, but would succeed if HELP without any keywords was typed. The problem has been fixed.

NOTE

Addition to directions for DIRECTORY command:

1) In order to get a directory in user mode, the target device must be mounted (do NOT use the /FOREIGN qualifier!). You can not get a directory from the console device in user mode, since VMS RMS does not support it.

2) The target device must be attached before you can get a directory of it. This applies in both standalone AND user modes. This is due to the fact that the Supervisor does not enforce device naming conventions, and therefore requires access to a Ptable to figure out what type of directory structure it is accessing.

3) The directory command will not work on ODS structure level 1 disks in standalone. (This is a bug, and will be corrected in a future version of the Supervisor!)

1.0 BUG FIXES

- If the COMET supervisor is loaded from a Massbus device, it currently attaches the load MBA as RH0, irregardless of the actual unit number. In version 6.3, this has been fixed, and it will attach the correct unit number.

- In the $DS_DEVTYP macro, under certain circumstances the supervisor could be fooled into thinking that a device name string was part of a Ptable descriptor when it actually was only a name. This is because it looks 5 bytes beond the end of the string for a byte 80(x), and if the byte IS 80, the supervisor thinks it is a Ptable descriptor. This has been fixed by adding a null byte to the end of the DEVTYP string list. This does not affect any diagnostic already assembled as it is transparent to both diagnostic and supervisor, occuring only within the library DIAG.

- For the $DS_ABORT macro, there was a difference in syntax between the MACRO and BLISS forms, as BLISS required 'PROGRAM' while MACRO would accept PROGRAM or program. BLISS will now accept PROGRAM and program, as well as 'PROGRAM' for compatibility with older versions.

- The $DS_WAITMS macro has been fixed so it will work correctly.

- The DV11 is now ATTACH-able. Previously the Supervisor knew of the device but would not allow ATTACH-ing it.

- RMS fixes (to match VMS RMS functionality more closely):

  - When $GET returns RMS$_RTB, the RAB$L_STV field will contain the full record size.

  - $READ will return the block number read in the first longword of the RFA (RAB$L_RFA0). The byte offset of the RFA (RFA$W_RFA4) is 0.

- SBI silo register dump (11/780 only). Due to an incorrect assumption as to instruction context in indexed addressing mode, the wrong field was decoded for function ID. This problem has been corrected.

- A fix to DS$WAITUS to promote more correct and consistent behaviour. This involves ensuring that the interrupt and error bits in the timer CSR are cleared.

- It is now legal for a section name to include the characters "$" and "_". Previously, they could be used in a diagnostic section name definition, but the Supervisor /SECTION qualifier (RUN and START commands) would not recognize them as legal. The Supervisor will now accept these characters. In conjunction with this, the function code CLI$K_SYMBOL has been added to the $DS_CLIDEF macro, and may be used by any diagnostic. This code causes PARSE to recognize a string composed of the characters "A" to "Z", "a" to "z", "0" to "9", "$", and "_".

- In standalone, the $SETPRT service would return an incorrect

address vector. The start address would be 1FF hex, due to a reversal of operands in a BICL3 instruction. The address will now be correct.

- In standalone, the $DS_RELBUF (or $CNTREG) would fail with an ERRSUP (Supervisor fatal error) if the page geing released was in P1 or SYS space and had been modified with memory management on (M bit set in PTE). V6.3 corrects this problem.

2.0 ENHANCEMENTS

- Several QIO driver service entry points have been added to support level 2R to level 2 conversion: IOC$ALLOSPT, EXE$MAXACMODE, and EXE$READLOCK. These entry points are defined in the DIAG.MLB $DS_QIOVEC macro.

- Previous to V6.3, it was possible to attach a device to any of the predefined "main buses" recognised by the supervisor. Now, an error message will result if the main bus name does not match the CPU type; for example, if a device is attached to an SBI on a COMET processor the error message is "there is no SBI on this type processor." Also, two new "main bus" names have been defined. These can be used on any VAX implementation. They are PHAD (for "Physical Address Decoder") and HUB.

- The $DS_SETVEC macro (DS$SETVEC routine) now allows setting the processor mode in which an interrupt will be handled, without changing the handler address. To do this, set the SRVADR argument to 0, and use the CODE argument as usual. Bits 2 to 31 of the SCB vector will be unchanged, but bits 0 and 1 will still be set to CODE.

- New commands. These support customer service and manufacturing in using terminals narrower than 132 columns, without losing data.

  - SET WIDTH n
  - SHOW WIDTH

  These commands will be accepted and carried out in user mode, but will have no effect on terminal characteristics (use VMS SET TERMINAL/WIDTH=n before running Supervisor). In standalone, a newline is forced when the cursor (printhead) passes column n.

- In user mode (only), the Supervisor will accept subdirectories in file specifications.

- HELP FILE: Where possible, I have included standard (fixed) CSR and vector addresses in the DEVICE portion of EVSAA.HLP. Also, the lines which used to read "Device:" now read "Description:" to avoid possible confusion over actual names. Finally, the recommended generic names have been brought into agreement with the "MASTER LIST OF PDP-11/VAX-11 DEVICE NAME CONVENTIONS" maintained by Kerby Altmann of VMS Development.

- Beginning with V6.3, the Supervisor .DOC file will contain a concatination of all release note files (from V5.3 on). This

will allow easier reference to warnings on use of Supervisor features, as well as tracking enhancements/bug fixes.

1.0 BUG FIXES

- QIO driver entry points IOC$ALLOSPT, EXE$MAXACMODE, and EXE$READLOCK, which were supposed to have appeared in version 6.3, somehow didn't. They are present in 6.4.

- ECSAA attached MASSbus load device incorrectly. This has been fixed.

- Most bits of the 11/750 UBI register are undefined, specified as MBZ (Must Be Zero) but are actually unpredictable. This is confusing in the output of a $DS_SHOWCHAN call. Therefore, the Supervisor will now mask out these undefined bits.

- The $SETPRT system service (standalone) would not work if the caller was not already in Kernel mode. Version 6.4 will do a CHMK to insure kernel mode privilege for the routine.

- The byte context PRVPRT output argument of $SETPRT was being written as a longword. Version 6.4 will write it as a byte.

- The diagnostic cleanup code was not called when a loop on subtest terminated. This has been fixed.

- If a range of tests were used with the /SUBTEST qualifier, the diagnostic would loop on the first subtest with the specified number, irregardless of which test it was in. Now, it will only loop on the correctly numbered subtest in the last test number given.

- The VMS console device non-interrupt I/O drivers have been incorporated into the Diagnostic Supervisor. This means console reads which fail due to media bad spots will be re-tried. (This is more of an enhancement, but is listed under bugs because Manufacturing has complained of trouble with console bad spots under APT control).

- The bliss diagnostic library (DIAG.L32) had several bugs in it, including incorrect macros for $DS_$INITIALIZE and $DS_$STRING, many errors in %ERROR statements to point out bad macro useage, and incorrect compilation of $DS_PRINTSIG macro. These have all been fixed.

- A bug in $DS_SETVEC prevented setting the mode in which a 'soft-vectored' exception was handled. Soft vectored exceptions include BPT and T-bit traps. This problem has been fixed.

2.0 ENHANCEMENTS

- FIELD definitions for all Ptable types known to the Supervisor have been added to the Bliss library (DIAG.L32). They can be used via the $DS_HPODEF macro; for example, "$DS_HPODEF ($DS_DZ11_DEF)" allows

access to the basic Ptable fields, plus the fields of the DZ11 Ptable. Also, literals representing the length of these Ptables have been added. There are no FIELD definitions for Ptables which have no device dependent fields.

- Correct spelling of many Supervisor messages, and change to mixed case for greater legibility.

3.0 MISCILLANEOUS

- Note: The $SETAST system service (as documented in the VMS System Service manual) is supported by the Supervisor in standalone. It was added in version 5.4 for Supervisor internal use and was not documented. It may be used by level 2 and 3 programs freely.

- Note: The Diagnostic Design Guide is in error over the PRINT ($DS_PRINTX, $DS_PRINTB, $DS_PRINTF) format argument. It should be counted ascii (ASCIC) not ASCID.

3.1 Previously Undocumented Gems Of Wisdom: Linking Diagnostics

- Bliss-32 => If a diagnostic is written entirely in Bliss-32, without any Macro-32 modules, the diagnostic must be linked with a DS.STB file defining Diagnostic Supervisor entry points (for system services, etc.).

  DS.STB can be created easily from the following Macro-32 source:

```
        .TITLE  DS.STB  Symbol table to link with BLISS modules
        .LIBRARY        'SYS$LIBRARY:DIAG'
        $DS_DSADEF      GLOBAL
        $DS_DSSDEF      GLOBAL
        .END
```

  And assembled/linked as follows:

```
        $ MACRO DS
        $ LINK/SYMBOL_TABLE=DS/NOEXECUTABLE DS
```

  Link <diag> as such:

```
        $ LINK <qualifiers> <diag>+DS.STB
```

- Macro-32 => The $DS_BGNMOD and $DS_ENDMOD macros use a label called $M0 for checking. The $DS_BGNMOD macro defines "$M0 == 0", and the $DS_ENDMOD defines "$M0 == $M0 + 1". Therefore, a missing $DS_BGNMOD macro will cause an "undefined symbol" error at assembly time in that

module, and a missing $DS_ENDMOD will cause a "multiply defined global" error at link time, due to the fact that correct module(s) define $M0 as 1 and modules missing $DS_ENDMOD macros define $M0 as 0.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.5 4-DEC-1981

A. GENERAL ENCHANCEMENTS

1. UNIBUS Mapping

The entire UNIBUS is now mapped, instead of just the I/O pages. This allows diagnostics to get UNIBUS NXM errors instead of access violation errors.

2. Channel Services status

By popular demand, channel services has been enhanced to return additional information:

- o UNIBUS NXM errors reported by bit CHS$M_BUSNXM
- o MASSbus writecheck errors reported by three bits:
  - a. CHS$M_CHNDPE is set if either of the other two are set:
  - b. CHS$M_MBAWCKLWR is set if the MASSbus write check lower bit is set.
  - c. CHS$M_MBAWCKUPR is set if the MASSbus write check upper bit is set.

3. APT troubleshooting

To enable troubleshooting of diagnostic problems under APT, the hardware context will not be initialized when an ABORT command code is received from APT, if errors have occurred.

4. Automatic diagnosis assistance

There are several programs (including APT, RD, and EVXBB) which act as operators to VDS. Currently, this process requires parsing of the VDS ASCII output. With Version 6.5, VDS is growing to make the task of automated diagnosis easier: A new flag has been implemented, called BINARY. When this flag is set, VDS will output a binary 'type byte' as the first byte of every message (including prompts and diagnostic PRINTS). This type byte will indicate the message class (e.g., command error, program start, harderror, etc.) The type codes are defined in macro $DS_TYPECODE in DIAG.MLB and DIAG.L32. Currently, the implementation is incomplete. Most messages for this release will have a type code of 0 (ds$k_type_general). Only prompts, program sequence messages (start, first_pass, end, etc.), and command messages will generate meaningfull type codes. The BINARY flag is off by default, and is not turned on by SET FLAGS ALL (or turned off by CLEAR FLAGS ALL).

5. DEVICE UNIQUENESS

The concept of a unique device has been altered. Previously two devices with the same name were considered the same, and could not be ATTACHed at the same time. To support network nodes

(which may be accessible through several different network adapters), TWO DEVICES WITH THE SAME NAME BUT DIFFERENT LINKS ARE NOW CONSIDERED DIFFERENT. This means that a device attached to the wrong adapter can not be 'edited' by re-attaching it. The new DEATTACH command gives the ability to correct any ATTACH errors.

6. Error messages

All error messages (ERRHARD, ERRSOFT, ERRDEV, ERRSYS, and "Software detected error") now include a trailing "End of error" line heralded by "********". This makes it easier to separate errors and assorted intervening text on printouts. It was particularly devised in conjunction with the BINARY flag (see item 4 in this section), and has its own unique type code.

B. COMMAND ENHANCEMENTS/ADDITIONS

1. New Qualifier /BRIEF

This qualifier can be used on the SHOW DEVICE and SHOW SELECT commands. It will cause only the device name and type fields to be printed.

2. New qualifier /ADAPTER=name

This qualifier can be used on the DEATTACH, SELECT, DESELECT, and SHOW DEVICE commands. It will limit the command to devices attached to the adapter with device name 'name'. E.g., for select, "SELECT/ADAPTER=DW0 ALL" will cause all devices attached to DW0 (but not devices attached in turn to those devices) to be selected for test. The special adapter name "HUB" is recognised, but not "SBI", "CMI", etc.

3. DEATTACH

This command reverses the ATTACH command for a device. It can be used to remove incorrectly attached devices. The /ADAPTER=name qualifier is REQUIRED on this command. Since Ptables are linked in a tree structure, it makes no sense to de-attach a device which has other devices attached to it, unless those devices are also de-attached. Therefore, the DEATTACH command will recursively "seek and destroy" all lower level devices for each device it de-attaches. E.g., if DW0 has devices DMA and TTA attached to it, TTA has device TTA0 attached to it and DMA has DMA0 attached to it, typing "DEATTACH/ADAPTER=HUB DW0" will cause DW0, DMA, TTA, DMA0, and TTA0 to be de-attached. a message will be typed for each as it is de-attached. (note that the message order may appear strange, since the de-attach routine does a post-order search).

4. EXIT

In on-line mode, this command will execute a VMS $EXIT system service to terminate the VDS image. In standalone, it will do a HALT after re-initializing the machine (a console CONTINUE will return to VDS command mode, however).

5. SHOW BASE

   Displays the current base value. (as in "SET BASE").

6. SHOW DEFAULT

   Displays the current default radix and data size.

7. SHOW SECTIONS

   Lists the SECTION names supported by the current diagnostic. An error results if no diagnostic is loaded.

8. SHOW STATUS

   Prints the name, revision, current time, error count, section, pass count, test, subtest, and PC for a CNTRL/C'd diagnostic. If there is not a running diagnostic, an error is typed.

9. SHOW SUPPORT

   Prints the list of devices supported by the current diagnostic. An error results if no diagnostic is loaded.

C. GENERAL CLI/PARSE ENHANCEMENTS

1. New PARSE features

   Special function codes have been added to PARSE to improve command parsing for the VDS CLI (they can also be used by diagnostics):

   o CLI$K_EOL: Accepts a string consisting of blanks (any number of spaces and/or tabs; including none of either) followed by either end of line (no more characters in string) or a "!" character (comment delimiter). It will call the action routine and branch through the miss label, as appropriate.

   o CLI$K_SLASH: Accepts blanks, followed by the character "/" followed by blanks.

   o CLI$L_COMMA: Like CLI$K_SLASH, but expects a "," instead of a "/".

   o CLI$K_VALUE: like CLI$K_SLASH, but expects either an "=" or a ":" instead of a "/". "=" and ":" are exactly equivalent (as in VMS DCL).

2. CLI improvements:

   o CLI uses the features listed above to make command syntax more flexible. For example, the qualifier "/PASSES:1" is the same as " / PASSES = 1".

   o The old CLI had two error messages: "Illegal command" and "Ambiguous command"; both of which were usually rather ambiguous themselves. The latter has been replaced by "Incomplete command" and the rules for generating each have

been tightened:

a. A command is illegal if something unexpected is found. E.g., the command "XYZZY".

b. A command is incomplete if something expected is NOT found. E.g., "CLEAR" lacking any arguments.

Furthermore, both error messages include some information on what was in error. "Illegal command" displays the command line UP THROUGH THE FIRST ILLEGAL CHARACTER. "Incomplete command" displays up to where the expected data should have been. E.g., the command "XYZZY" results in:

```
?? Illegal command
   \X\
```

while the command "CLEAR" results in:

```
?? Incomplete command line
   \CLEAR\
```

D. PTABLE DESCRIPTOR ENHANCEMENTS

1. New directives

- Previously, to request logical (yes/no) data, it was necessary to use the $DS_$STRING directive, supplying prompt, "NO", and "YES" strings. I have implemented the new directive $DS_$LOGICAL. It's only argument is the prompt string. This saves program space, and also allows ABBREVIATION of the words "Yes" and "No". All Ptable descriptors within VDS which require "Yes/No" input have been modified to use this directive.

- Value translation can now be accomplished by the directive $DS_$CASE. The argument is a list of "case_pairs"s. The current VALUE register is compared to the first value of each pair until a match is found; the second value of that pair becomes the new VALUE, and the case directive is exited. VALUE is not altered if no match is found.

  Bliss-32:

```
$ds_$case (<1,%x'FFF'>, <2,%x'1CE0'>)
```

  MACRO-32

```
$ds_$case <<1,<^xFFF>><2,<^x1CE0>>>
```

E. MISCELLANEOUS ENHANCEMENTS

1. VERIFY

VDS now supports a mechanism similar to the VMS "SET [NO]VERIFY". The flag VERIFY can be set or cleared via "SET FLAG VERIFY" or "CLEAR FLAG VERIFY". When the flag is set, commands read from VDS scripts are echoed to the console. When the flag

is clear, command lines are not echoed. This does not affect error or information messages, and has no effect unless commands are being processed from a VDS SCRIPT file : commands read from a VMS command file are considered by VDS to be coming from the console!

!!The flag is CLEAR by default!!

2. MIXED CASE MESSAGES

Many messages throughout VDS have been altered from all caps to mixed case. We feel that this is much easier to read, as well as appearing "warmer", in the line of "human engineering".

F. BUG FIXES

1. Ptable descriptor location

A bug has existed which could cause errors when a program's DEVTYP list existed but had 0 elements. This has been fixed.

2. ATTACH

It is now possible to CNTRL/C out of an ATTACH prompt. Furthermore, ATTACH will not attempt to prompt a SCRIPT or APT for a missing or invalid parameter: the command will be aborted. From a SCRIPT, the error "Prompt not found" is generated. From APT mode, a "Software detected error" is generated.

3. Fixes to DIAG.L32 macros

- $DS_ENDMOD malfunctioned when a module did not contain tests (incorrectly setting an overlayed PSECT which at runtime should contain highest test number). This has been fixed.
- $DS_HPO_DECL has been fixed to correctly accept multiple arguments.
- $DS_$STRING and $DS_$INITIALIZE did not function properly. Now they will.

4. Event flag services

All standalone event flag services have been re-written to correspond to the VMS services. The $SETEF and $CLREF services will return SS$_WASSET or SS$_WASCLR when they should, as will $READEF.

G AUTOMATED DIAGNOSTIC QUALITY ASSURANCE

G.1 Background

Enhancements have been made to the Diagnostic Supervisor to enable it to automatically perform Quality Assurance checks on diagnostic programs.

This version of the Supervisor performs the Normal Start check

(Section 3.1 in the "VAX Diagnostic Quality Assurance Checklist"), the Multiple Pass check (Section 3.2), the Loop on Test check (Section 3.7), and the Run Backwards check (Section 3.5). Other sections in the checklist will be added to the Supervisor as time and resources permit.

The Normal Start check does the following: saves the current setting of the Supervisor flags; turns on the TRACE flag; turns off the following flags: BELL, IE1, IE2, IE3, IES, LOOP, OPER, PROMPT, QUICK, and SEARCH; performs a normal start of the diagnostic program, running all the tests in the specified section; turns the TRACE flag off; and either stops when a QA error is detected or proceeds to the next check.

The types of errors that the Normal Start check will detect are asking for operator input when the no-default flag is set (in the $DS_ASK macro) and reporting a diagnostic error via one of the $DS_ERR macros. When a QA error is detected, the Supervisor will print out information that should help the diagnostic engineer determine why the program failed QA. Then, an overall error summary table is printed, the Supervisor flags are restored to their previous settings, and the diagnostic is aborted.

If the Normal Start check succeeds, the Multiple Pass check is performed. This check performs a number of passes of the diagnostic program. The current setting of the QAMULTIPLEPASS flag is the number of passes that will be executed (see Section G.4). The types of QA errors that can occur are the same as those in the Normal Start check.

If the Multiple Pass check succeeds, the Loop on Test check is executed. This check runs each test QATESTLOOPS times (see Section G.4). Assuming a diagnostic has tests 1, 2, and 3, and QATESTLOOPS has been set to 4, this check will run the initialization code, followed by tests 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, and 3. The types of QA errors that can occur are the same as those in the Normal Start check.

If the Loop on Test check succeeds, the Run Backwards test is executed. This check runs the tests in reverse order. If a diagnostic contained tests 1, 2, 3, and 4, this check would execute the initialization code followed by test 4, test 3, test 2, and then test 1. The types of QA errors that can occur are the same as those in the Normal Start check.

If the Run Backwards check succeeds, the overall error summary table is printed, a line is printed saying that the program passed Auto-QA, the Supervisor flags are restored to their previous settings, and the diagnostic is aborted.

G.2 START/QA

A new qualifier has been added to the START command. If the START command contains the /QA qualifier, QA will be performed on the currently loaded diagnostic program. If the /QA qualifier is present, the /PASSES, /SUBTEST, and the /TEST qualifiers are

ignored. The /SECTION qualifier may be used to run QA on a particular section in the program.

G.3 RUN/QA

The /QA qualifier has also been added to the RUN command. It functions exactly the same as the /QA qualifier on the START command.

G.4 QA Flags

There are five new Supervisor flags that will affect QA. However, only two of these are used in the current implementation.

The QAMULTIPLEPASS flag controls how many passes are performed in the Multiple Pass check. This flag may be set as follows:

```
DS> SET QAMULTIPLEPASS 10
DS> SET QAM 3
```

In addition, the current setting of the flag may be printed as follows:

```
DS> SHOW QAMULTIPLEPASS
DS> SHOW QAM
```

Note that the QAMULTIPLEPASS flag may be abbreviated to QAM. The initial (and default) setting for this flag is 10. That is, 10 passes of the diagnostic will be executed in the Multiple Pass check.

The QATESTLOOPS flag controls now many times each test is executed in the Loop on Test check. For example, if a diagnostic program contains three tests and QATESTLOOPS equals 2, this check will first execute the initialization code, then tests 1, 1, 2, 2, 3, and 3.

This flag may be SET and SHOWn in the same manner as the QAMULTIPLEPASS flag. This flag may be abbreviated to QAT. The initial (and default) setting for this flag is 100.

Also, the following command will set all the QA flags to their default settings:

```
DS> SET QADEFAULTS
```

QADEFAULTS may be abbreviated to QAD. To show the default settings (not the current settings) of the five QA flags, the following may be used:

```
DS> SHOW QADEFAULTS
```

G.5 Modifications Necessary For QA

Diagnostic programs wishing to undergo QA must call the $DS_ENDPASS macro in the initialization code. This is necessary for the internal Supervisor sequencing of QA checks. The branch macros discussed in the "Design Specification for Automatic Quality Assurance for Diagnostic Programs" need not be used in the diagnostics for this version of QA.

Because of the way the Supervisor works, the code executed in the diagnostic's initialization code when pass0 is true (in conjunction with the $DS_BPASS0 or $DS_BNPASS0 macros) will be done only once. It is done at the start of the Normal Start check. Also, the cleanup section will be done only once. It is done at the conclusion of the Normal Start check (before the Multiple Pass check is started).

Thus, the whole QA sequence of checks may be thought of as executing the diagnostic a number-of-passes times, with different things happening in different passes. That is, one pass is executed for the Normal Start check, then QAMULTIPLEPASS passes are executed in the Multiple Pass check, then one pass is executed in the Loop on Test check (with the tests executing more than once), and then one pass is executed for the Run Backwards check (with the tests being run in reverse order).

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.7 03-May-1982

A. GENERAL ENCHANCEMENTS/NOTES

1. This release is almost identical to the Version 6.6 VDS which was released in the previous cycle. However, due to technical difficulties, particularly due to the support of VMS V2-style QIO interface in standalone, while linking VDS under V3 of VMS, it was not possible to release version 6.6 VDS for 11/780 and 11/750 processors.

   This release (6.7) includes all three processors; ENSAA (11/730), ECSAA (11/750), and ESSAA (11/780).

2. VMS V3a support

   Support of VMS V3a includes support for the new system directory structure, where [SYS0.SYSMAINT] replaces the old [SYSMAINT] directory. Earlier versions of VDS will not operate correctly with a 'rooted' SYSMAINT directory. This support requires DIAGBOOT.EXE V4, which was released with the previous release cycle.

   A side-effect of the support of rooted systems is that VDS now fully supports subdirectories in standalone mode.

   NOTE that while VDS 6.7 supports the directory structure of VMS V3a, it is linked under VMS V2.5, and so will run under VMS V2.5 or V3a.

3. New Device support

   VDS V6.7 includes support for the UDA-50 UNIBUS to SI disk adapter/controller, and for the RA- series disk drives which can attach to it.

B. COMMAND ENHANCEMENTS/ADDITIONS/FIXES

1. SELECT/DESELECT

   In version 6.5, the SELECT and DESELECT commands did not handle the new definitions of 'device uniqueness' correctly. If the device name specified was attached to more than one adapter, the commands would simply act upon the first such device found (unless the /Adapter qualifier was used). Since the SELECT command affects the order in which device names are searched, this often meant that it was impossible to DESELECT the device which had been SELECTed; yet no error message was given. Now, such a situation will result in an error message saying that the /Adapter qualifier must be used.

2. HELP

   In the process of major restructuring of the VDS command parser for version 6.5, a bug was inserted in the parsing of the HELP

command. Specifically, any command line which began with 'H' was taken as a HELP command; for example, a 'SHOW' command mis-typed as 'HSOW' was interpreted as if 'HELP SOW' had been typed. This has been fixed for VDS 6.6.

3. SHOW MEMORY

This new command displays the layout of VDS memory. In default mode, the start and end addresses of the diagnostic, VDS, and dynamic memory (used by VDS) are typed, as well as the physical memory size of the system, in both HEX and DECIMAL radix. There are several qualifiers to this command. Note that you can not use these qualifiers prior to the MEMORY keyword. E.g., 'SHOW MEMORY/ALL' but not 'SHOW/ALL MEMORY'.

a. /MAP : same as default mode

b. /DATA_STRUCTURE : shows internal VDS data structure addresses, including all stacks, page tables, SCB, and various simulated VMS data structures (PCB, PHB, etc.). knowing the addresses of these data structures can be usefull in debugging. Note that the memory layout in on-line mode is quite different than in standalone; in on-line mode, none of the aforementioned data structures are present within VDS memory space, and will not be displayed.

c. /BUFFER : Show available space for diagnostic buffers, both above and below VDS (this command is not meaningfull when no diagnostic is loaded). Note that since on-line mode allocates buffer space above VDS via $EXPREG as it is requested, /BUFFER on-line will not show space above VDS.

c. /ALL : Combine /Map, /Data_Structure, and /Buffer

4. DIRECTORY

The DIRECTORY command has been enhanced. Previously, it would accept as arguments only the device and directory parts of a filespec, and would take only one level of directory in standalone mode. Now, you can specify the filename, type, and version fields of the filespec; additionally, both the filename and type fields may include both "*" and "%" wildcards (with the same meaning as under VMS). Furthermore, subdirectory specification is now legal in standalone as well as on-line.

5. SUMMARY

The SUMMARY command has been enhanced. Whether or not the diagnostic program contains summary code, VDS will generate a header containing the name of the diagnostic and a summary of all HARD, SOFT, DEVICE, SYSTEM, and SUPERVISOR DETECTED errors which occurred during the run.

C. AUTOMATED DIAGNOSTIC QUALITY ASSURANCE

1. Introduction

These sections will explain what functionality I have added to the Automatic Quality Assurance enhancement to the Diagnostic Supervisor, and what problems I have fixed since the 6.5 version of the Supervisor. Note that when I speak of versions of QA or versions of the Diagnostic Supervisor, I am actually speaking of the same thing.

2. Additional Functionality

The following gives the two new QA checks that have been added to the 6.6 version:

a. Loop On Subtest check:
This check performs a loop on every subtest in the diagnostic. The number of loops is user-controlled via the QASUBTESTLOOPS Supervisor flag. The following is an example of how this check works:

Assumptions:
Diagnostic EXXXX contains two tests.
Test one has three subtests.
Test two has two subtests.
QASUBTESTLOOPS has been SET to 3.

Order of Execution (Test#.Subtest#):
1.1
1.1
1.1
1.1

1.1
1.2
1.2
1.2

1.1
1.2
1.3
1.3
1.3

1.1
1.2
1.3
2.1
2.1
2.1

2.1
2.2
2.2
2.2

2.1
2.2
End of Pass

In the above, the blank lines represent places where QA

internally aborts the diagnostic and restarts it so as to start looping on the next subtest. The ENDPASS routine is only executed at the very end of the check. However, the initialization code and the cleanup code are both executed once for each of the above groupings.

b. Error Phase One check:
This QA check forces as many error reports as possible to be printed. This is a way of insuring that the parameters passed to the error routines are correctly coded, as well as allowing the user to check the format of the error messages. The following lists some points that must be considered when writing a diagnostic that will be QA'ed using Auto-QA:

i. Any error call that was not forced (i.e., was not preceeded by a branch macro in which QA forced the branch to change sense) will be treated as a QA error, resulting in termination of Auto-QA.

ii. No two branch macros should occur in sequence without an intervening error macro call. That is, the sequence should always be one branch macro followed by one error macro call.

iii. No branch macros should occur in a subroutine. Branch macros in subroutines may cause QA to go into an infinite loop and print error reports ad infinitum! This may be fixed in a later version of QA.

iv. The branch macros for use in MACRO diagnostic modules are listed below. For an explanation of the parameters, see Revision 1.1 of the "Functional Specification for Automatic Quality Assurance for Diagnostic Programs" memo. Contact Jack Stansbury (TW/F18) for a copy of this memo.

```
$DS_BLSS        ADR, NUM
$DS_BLSSU       ADR, NUM
$DS_BLEQ        ADR, NUM
$DS_BLEQU       ADR, NUM
$DS_BEQL        ADR, NUM
$DS_BEQLU       ADR, NUM
$DS_BGEQ        ADR, NUM
$DS_BGEQU       ADR, NUM
$DS_BGTR        ADR, NUM
$DS_BGTRU       ADR, NUM
$DS_BGTRU       ADR, NUM
$DS_BNEQ        ADR, NUM
$DS_BNEQU       ADR, NUM
$DS_BVS         ADR, NUM
$DS_BVC         ADR, NUM
$DS_BCS         ADR, NUM
$DS_BCC         ADR, NUM
$DS_BBS         POS, BASE, ADR, NUM
$DS_BBC         POS, BASE, ADR, NUM
$DS_BBSS        POS, BASE, ADR, NUM
```

```
$DS_BBSC        POS, BASE, ADR, NUM
$DS_BBCS        POS, BASE, ADR, NUM
$DS_BBCC        POS, BASE, ADR, NUM
$DS_BBCCI       POS, BASE, ADR, NUM
$DS_BBSSI       POS, BASE, ADR, NUM
$DS_BLBS        SRC, ADR, NUM
$DS_BLBC        SRC, ADR, NUM
$DS_BERROR      ADR, NUM
$DS_BNERROR     ADR, NUM
```

v. The branch macros for use in BLISS diagnostic modules are as follows (note that the BCS, BCC, BVS, and BVC branch macros are not implementable in BLISS):

```
$DS_BLSS    (VAL, NUM)
$DS_BLSSU   (VAL, NUM)
$DS_BLEQ    (VAL, NUM)
$DS_BLEQU   (VAL, NUM)
$DS_BEQL    (VAL, NUM)
$DS_BEQLU   (VAL, NUM)
$DS_BGEQ    (VAL, NUM)
$DS_BGEQU   (VAL, NUM)
$DS_BGTR    (VAL, NUM)
$DS_BGTRU   (VAL, NUM)
$DS_BGTRU   (VAL, NUM)
$DS_BNEQ    (VAL, NUM)
$DS_BNEQU   (VAL, NUM)
$DS_BBS     (POS, BASE, NUM)
$DS_BBC     (POS, BASE, NUM)
$DS_BBSS    (POS, BASE, NUM)
$DS_BBSC    (POS, BASE, NUM)
$DS_BBCS    (POS, BASE, NUM)
$DS_BBCC    (POS, BASE, NUM)
$DS_BBCCI   (POS, BASE, NUM)
$DS_BBSSI   (POS, BASE, NUM)
$DS_BLBS    (SRC, NUM)
$DS_BLBC    (SRC, NUM)
$DS_BERROR  (VAL, NUM)
$DS_BNERROR (VAL, NUM)
```

After all the errors are forced, a table is printed that summarizes the error numbers. This is the Forced-Error Summary table. This table looks like the following:

Summary of Forced-Errors

Test X1	Subtest Y1
aaaaa-bbbb	ccccc-dddd ...
mmmmm-nnnn	...

Test X2	Subtest Y2

```
AAAAA-BBBB CCCCC-DDDD EEEEE-FFFF GGGGG-HHHH ...
MMMMM-NNNN OOOOO-PPPP QQQQQ-RRRR ...
...
```

The error numbers are printed in a test/subtest order, with the error numbers sorted within each test/subtest.

The Xi and Yi are the test and subtest numbers, respectively. The 'aaaa', 'cccc', ..., 'AAAA', 'CCCC', etc. are the error numbers that are passed to the $DS_ERRxxxx macro.

The 'bbbb', 'dddd', ..., 'BBBB', 'DDDD', etc. are the number of times that error was forced. This number should be one except in those cases where the same error number is used more than once in one subtest.

If the diagnostic runs on a subset of the total number of tests in the DEFAULT section (by using the /TEST qualifier), the Overall Error Summary table will not be printed.

However, if the /TEST qualifier is not used, the Overall Error Summary table will be printed. This table gives the number of errors that occurred in each QA check. For the 6.6 version of the Supervisor, these numbers should be either all zeroes (in which case a line is printed saying the program passed Auto-QA), or else there will be at most one error (since Auto-QA is aborted upon detection of a single QA error).

3. Auto-QA Problems in the 6.5 Version

The following is a list of problems with QA that were discovered in the 6.5 version of the Supervisor:

a. The Halt-On-Error flag was not cleared before starting QA. I had meant to clear all the Supervisor flags when QA started, and then restore them when QA ended. However, I forgot this one flag. Although it probably will not cause any problems in the 6.5 version, it would have caused problems in this new version.

b. If a previously-started diagnostic is running, a Control-C is typed, and then a CONTINUE Supervisor command is executed, Auto-QA will start behaving strangely (it will set the number of passes to 0, and start executing forever).

c. The cleanup code in the diagnostic was executed at the end of the first QA check, the Normal Start check. After executing the cleanup code, QA caused the diagnostic to continue running by starting the Multiple Pass check. That is, the /QA qualifier on the START or RUN commands cause the following to happen:

   i. The diagnostic is started and executes one pass.

This means it executes the initialization code (doing the pass-zero things, such as allocating buffers and assigning channels), then executes all the tests in the appropriate section (the DEFAULT section by default).

ii. Then, after executing the last test in that section, the Supervisor calls the initialization code again. This time, the pass-zero things will not be done and the ENDPASS routine will be called.

iii. The ENDPASS routine calls the diagnostic's cleanup code (where the buffers are usually deallocated and channels deassigned).

iv. Normally at this point (i.e., when NOT under control of QA), the Supervisor would return to command mode. However, QA will cause the ENDPASS routine to set the number of passes to QAMULTIPLEPASS (settable via a "SET QAMULTIPLEPASS" DS command) and start the Multiple Pass QA check.

v. This causes test one to execute again. Note that the initialization code is NOT executed in between the Normal Start check and Multiple Pass check. Thus, the buffers that were deallocated at the end of the Normal Start check (in the cleanup code) are no longer available to the diagnostic in the Multiple Pass check. This was causing several diagnostics (including the Cluster Exerciser) to fail.

d. The branch macros (i.e., $DS_BERROR, $DS_BLBS, $DS_BBS, $DS_BNEQ, etc.) will destroy the contents of R0. This happens because the branch macros expand into a Supervisor service routine, which is executed with a CALLS instruction. Since R0 is never saved across CALL instructions, the contents can not be guaranteed upon return from this routine.

e. The register values that are printed out when a QA error is detected are incomplete. That is, at most three of the registers are correct, with the others having a default value of zero. This led to some confusion.

f. The /TEST qualifier was ignored on the START or RUN commands when used in conjunction with the /QA qualifier.

g. The /SECTION qualifier did not work properly when used in conjunction with the /QA qualifier. The DEFAULT section is the only section that can be run when using the /QA qualifier.

4. Fixes to the Above Problems

The following describes the status of the above problems:

a. The HALT flag is now cleared when QA is running. As with the other Supervisor flags (e.g., IE1, IE2, IE3, TRACE, BELL, etc.), it is restored to its previous state (i.e., to the same setting it had before the START or RUN command was executed) after QA finishes.

b. A Control-C will now cause the diagnostic currently being QA'ed to be aborted only if Control-C has NOT been disabled by the diagnostic. That is, if the diagnostic has disabled C's, C's will be ignored when running under QA. If the diagnostic has not disabled C's, (they are enabled by default), the Supervisor will gain control (before the diagnostic's Control-C handler) and will abort the diagnostic. Note that a CONTINUE Supervisor command in this context will not work.

c. The way the QA check routines execute the diagnostic has been fundamentally changed. Whereas before the sequence of checks could be thought of as executing a variable number of passes of the diagnostic, the sequence is now like STARTing the diagnostic a fixed number of times (the number of times depends on the number of QA checks).

For example, ASSUMING (note that even though there is none, this has been suggested as an enhancement) there was a new qualifier on the START (and RUN) command that would allow one to select which particular QA check routine to execute (such as /QA=NORMAL_START to execute ONLY the Normal Start QA check on the diagnostic), then the sequence of checks can now be thought of as being:

```
DS> START/QA=NORMAL_START
DS> START/QA=MULTIPLE_PASS
DS> START/QA=LOOP_ON_TEST
DS> START/QA=LOOP_ON_SUBTEST
DS> START/QA=RUN_BACKWARDS
DS> START/QA=ERROR_PHASE_ONE
```

That is, the diagnostic program is reSTARTed at the beginning of each QA check and executes as though a normal START Supervisor command had been executed. This implies that the initialization code is always done at the beginning of each check routine, then all the tests execute (a varying number of times depending on which QA check is executing), then the ENDPASS routine is executed causing the cleanup code to execute, and then the next QA check routine performs the same way. This new implementation should alleviate the deallocation problems encountered by several people.

d. The implementation of the branch macros has changed somewhat. Some diagnostic engineers felt that the $DS_BERROR and $DS_BNERROR macros should NOT alter the contents of R0 both when QA is not running, and when QA is running but not forcing errors (as happens in the Error Phase One QA check). In this version of the Supervisor and of the DIAG library, the $DS_BERROR and $DS_BNERROR macros will alter the contents of R0 only

when QA is running and forcing errors.

For branch macros other than $DS_BERROR and $DS_BNERROR, the contents of R0 may change across the branch macro call. The main reason an exception was made in the case of the above two macros is that they existed prior to the QA enhancement and did not alter the contents of R0. Therefore, to be compatible with old diagnostics, it was felt that this should not change. However, with the new branch macros, R0 will change only in the low bit position (i.e., it may change from a one to a zero, or vice-versa). Note that this bit may change even when the /QA qualifier is not being used (i.e., in normal running of the diagnostic).

e. The register values printed out when a QA error is detected are now correct. The values printed are the values at the time of the call to the Supervisor Service routine that produced the QA error (e.g., a $DS_ERRHARD call).

f. The /TEST qualifier is no longer ignored when used in conjunction with the /QA qualifier. This allows one to run QA on a particular test in a diagnostic, or on a range of tests. However, if the /TEST qualifier is used, the Overall Error Summary table is not printed (see later section).

g. The /SECTION qualifier is still essentially ignored. This will be fixed in the next version of the Supervisor.

D. DIAGBOOT (DIAGNOSTIC SECONDARY BOOTSTRAP PROGRAM)

(DIAGBOOT has not changed from release 6.6 to 6.7 of VDS; this section is included for the convenience of those who have not seen the 6.6 release notes).

DIAGBOOT is loaded by VMB when a BOOT command is issued with register 5 bit 4 set. DIAGBOOT will sense the type of processor (11/780, 11/750, or 11/730) and load the appropriate VDS image (ESSAA, ECSAA, or ENSAA respectively). This new version of DIAGBOOT (version 4) supports the new VMS rooted systems (the old [SYSMAINT] becomes [SYS0.SYSMAINT], or [SYS1.SYSMAINT], etc.). The high nibble (bits 28 to 31) of R5 at boot time represent the hex digit which should be converted to ASCII and appended to the root directory name 'SYS'; this allows system root directories of 'SYS0' to 'SYSF' to co-exist on the same disk pack.

This new version of DIAGBOOT also allows the VDS image being booted to be non-contiguous. With previous versions of DIAGBOOT, VDS had to be contiguous.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.8 6-Jul-1982

A. GENERAL ENCHANCEMENTS

1. ATTACH enforces device naming conventions

An additional Ptable descriptor statement ($DS_$NAME; see VAX-11 Diagnostic Design Guide, chapter 17 for information) allows VDS to verify the names given to devices when they are attached. For example, when an RK07 is ATTACHed, it must have a name of the generic format 'ggan', where 'gg' is the device generic prefix, 'DM', 'a' is the controller letter of the RK611, and 'n' is the unit number. Previously, VDS imposed no constraints on device names other than on whether a unit number was allowed, and on the magnitude of the unit number if allowed.

Because this is new functionality, and some old SCRIPTS or even diagnostics may not function under these new constraints (e.g., scripts which ATTACH an RK07 as 'DM0'), new commands have been implemented which will enable and disable this functionality under operator control. The commands are:

- SET ENFORCE : This command enables enforcement of device name enforcement. It is the default condition when VDS starts, EXCEPT when running under APT.

- CLEAR ENFORCE : This command disables enforcement of device name enforcment. It will cause VDS to behave as it did prior to this enhancement. CLEAR ENFORCE is the default condition when VDS is run under control of APT.

****NOTE****
These commands are intended only for EMERGENCY use. You should fix your script files and/or diagnostic programs immediately to conform to the device naming conventions.

2. AUTOSIZER (EVSBA) changes

The autosizer (EVSBA V1.4) has been modified to generate the correct names for V6.8 of the Diagnostic Supervisor. The entire attach command line generated for each device is described in a separated document.

3. $DS_ERRPREP_x [num], [unit], [msgadr], [prlink], [p1--6]

A new error message facility has been added to the Supervisor

for detecting device preparation errors. You should use this macro after detecting a device preparation error (e.g. disk not spinning) instead of another error message facility. The parameter list is the same as for the other error macros and is fully described in section 8.7 of the Vax-11 Diagnostic Design Guide (EK-1VAXD-TM-002).

B. GENERAL BUG FIXES

1. KERNEL STACK NOT VALID ABORTS

Previously, VDS allocated only 3 pages for KERNEL mode stack. In certain commands, notably a directory of an ODS stuctured disk (standalone only, with MM ON), these 3 pages could be overflowed, resulting in a KERNEL STACK NOT VALID ABORT, due to the page below the kernel stack being marked 'no access' to catch stack overflows. To minimize the chances of this sort of behavior, the kernel stack has been expanded to 10 pages for V6.8. This problem can not occur in on-line mode (under VMS).

2. TM03/TE16/TU45/TU77

The problem with not being able to access magtapes on this controller through the Supervisor has been solved in this released.

3. MACHINE CHECK LOGOUT

The machine check logout for the 11/780 (ESSAA) has been changed to show the error summary register in addition to the other parameters.

C. NOTES

1. TM78 SUPPORT

VDS theoretically supports MAGTAPE devices as file load devices. Currently, the only Magtape devices supported are TM03 and TS11 controllers. This release includes the beginnings of TM78 (TU78 drive) support: however, due to a lack of standalone hardware, it is not debugged or supported. This note is just to warn those who may try a file command (DIRECTORY, HELP, LOAD, RUN) from a TM78 magtape, or who look at the VDS map listing and see 'TFBTDRIVR' (which is the VDS TM78 readonly driver) that the TM78 code is undebugged and probably will not work as-is. However, TM78 support will be completed as soon as possible, and documented in a future release.

2. INTERRUPT STACK

The INTERRUPT stack size has be increased 4 pages so the priviledge exerciser can print out errors without failing.

3. CUSTOMER RUNNABLE DIAGNOSTICS (CRD)

The Supervisor has been enhanced to support CRD AUTO TEST on the 11/730 machine. For more information on CRD, see the "Nebula Customer Runnable Diagnostics Functional Specification" DTM-82-001-01, document.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.9 8-Nov-1982

1.0 BUG FIXES

- The following sequence of commands caused a "?? Translation not valid Fault ..." :

  ```
  DS> SET MM ON
  DS> ATTACH ...
  ```

  This occurred only in standalone mode, and only if a program had not been previously loaded with a supervisor LOAD command.

- A "^Z", typed at the console terminal in stand-alone mode, echoed garbage as a non-printing character. It has been fixed so that it now imitates the supervisor EXIT command, and returns control to the console program.

- The following error message appears when any file services (such as LOAD, RUN, DIRECTORY, HELP, and script files) are attempted with an RL11/RL02:

  %RMS-F-DEV, bad device, or inappropriate device type (R0=000184C4)

  This is a bug only in version 6.8 in standalone mode, and is fixed in version 6.9.

- Also, only in version 6.8 and only for the 11/750, there is a bug which prevents the supervisor from loading large files from the console TU58 tape cartridge. Some small files could be loaded. Typically, the tape cartridge indicator light would remain lit for an excessively long time, and eventually the supervisor would respond with an error message. This is fixed in version 6.9.

- New versions of VMS which modify the system service transfer vector table experienced a problem running the supervisor in on-line mode. This occurred because the supervisor copied three pages of system service vectors for its own use, but the transfer vectors had been modified to branch into a fourth page. In order to accomodate other possible modifications to the VMS system service transfer vectors, the supervisor's transfer table was modified to jump directly to the actual VMS system service transfer vector table.

2.0 ENHANCEMENTS

- Support for the TM78/TU78 magnetic tape device as a file load device is now functional. This means that commands such as DIRECTORY, HELP, LOAD, and RUN, will now work when the TU78 is ATTACHed and a SET LOAD command has been executed.

- The SHOW MEMORY/BUFFER command has been enhanced to show the amount of buffer space in use by the diagnostic. The command displays the amounts in use in P0, P1, and S0 space (the number of pages in use). Previously, only the available buffer space was displayed. The SHOW MEMORY/ALL also displays this information.

- CUSTOMER RUNNABLE DIAGNOSTICS (CRD): VAX-11/730 MENU TEST

  The VAX Diagnostic Supervisor now supports VAX-11/730 MENU TEST (CRD MENU) as part of the CRD Package. CRD MENU is an off-line Menu-driven test feature of CRD. The purpose of CRD MENU is to provide the customer, or inexperienced operator, stand-alone VAX Diagnostic test capability for the VAX-11/730 'base' system package and, in particular, for VAX-11/730 Add-on options. CRD MENU, therefore, represents a complementary test package to VAX-11/730 AUTO TEST (see DTM-82-001-01 Nebula CRD Auto Test Functional Specification).

  CRD MENU, when invoked, will automatically determine CRD MENU supported hardware on the system and satisfy the necessary software requirements required for executing VAX Diagnostics. Through the Menu interface, the operator may select and test one supported hardware unit for test, or with the 'Test All' mode feature, select and test all supported system hardware. CRD MENU, therefore, provides full functional verification and fault isolation to the subsystem level of the system.

  CRD MENU is invoked by VAX-11/730 Console Command (typing 'T/M' to console prompt), at the successful completion of VAX-11/730 AUTO TEST (invoked by typing 'T' to console prompt), or from VAX Diagnostic Supervisor Command Level (typing 'CRD' to VDS CLI command prompt). However, the 'CRD' Supervisor command is not supported for use in Supervisor command files.

  Four new Supervisor commands have been added for CRD MENU Test. The commands are as follows:

  ```
  SET CRD/TRACE
  CLEAR CRD/TRACE
  SHOW CRD
  CRD
  ```

  The functions of these commands are explained in the ENSAB.DOC file.

For additional information:

- see ENSAB.DOC
- see the Functional Specification for VAX-11/730 MENU TEST
- Type "HELP ENSAB MENU" to VDS CLI Command prompt (this requires that the ENSAB.HLP file be on the load device).

3.0 MISCELLANEOUS

- A bug in the $D1_ERR_L MACRO in DIAG, which caused an invalid conditional directive, has been fixed.

4.0 KNOWN BUGS AND DEFICIENCIES

- The following commands will cause an error message if a comment follows the command:

```
SET ENFORCE
CLEAR ENFORCE
SET CRD/TRACE
CLEAR CRD/TRACE
SHOW CRD
CRD
```

For example,

```
DS> SET ENFORCE           ! Comment
?? Illegal command
   \SET ENFORCE           ! C\
```

All of the above commands except the 'CRD' command will still perform their correct function even though the Supervisor responds with "?? Illegal command". NOTE: the 'CRD' command will FAIL if the command is followed by a comment on the same line.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.10 3-Jan-1982

1.0 BUG FIXES

- The 6.9 VDS bug concerning comments on the following Supervisor commands has been fixed. All of the following commands now accept comments after the command:
  - SET ENFORCE
  - CLEAR ENFORCE
  - SET CRD/TRACE
  - CLEAR CRD/TRACE
  - SHOW CRD
  - CRD

2.0 ENHANCEMENTS

- Support for the TM80 magnetic tape device as a file load device is now functional. This means that commands such as DIRECTORY, HELP, LOAD, and RUN, will now work when the TU80 is ATTACHed and a SET LOAD command has been executed.

- A new supervisor service call, which loads the COMET PCS microcode and patch bits, has been added. The service is called by a supervisor macro, $DS_LOADPCS_x ($DS_LOADPCS_DEF, $DS_LOADPCS_G, $DS_LOADPCS_L, and $DS_LOADPCS_S for MACRO and $DS_LOADPCS for BLISS), with no arguments. The service returns DS$_NORMAL if the service is completed successfully, and DS$_NOPCS if PCS is not available on this CPU. Currently, the only diagnostic program which uses this call is ECKAX. This release does not contain PCS microcode which affects any instructions. It merely restores the PCS to a known state. If the CPU attempts to execute any of the PCS microcode, an error message will be printed out.

3.0 MISCELLANEOUS

-

4.0 KNOWN BUGS AND DEFICIENCIES

- o

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.11 23-May-1983

1.0 BUG FIXES

- Implementation of Control-Z recognition in standalone mode was done incorrectly and has been removed, although no symptoms have been discovered.

2.0 ENHANCEMENTS

1. VDS 6.11 supports booting and file loading from HSC50 on CI780.

3.0 MISCELLANEOUS

2. The device name for booting from the RP07 disk will now be DRan and not DBan.

3. The driver for the TM78/TU78 was changed to have a longer delay between accesses of the ATTENTION register during the wait for command completion to prevent spurious operations caused by false signals. The driver was enhanced to work with tapes written at 6250 b.p.i and be usable on drives other that number 0.

-

4.0 KNOWN BUGS AND DEFICIENCIES

-

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.12 21-Jul-1983

1.0 NEW FEATURES

- The SET WIDTH command has been changed to work in online mode. The default is the same as for standalone mode, 80 characters.

- The SET PAGE command has been added to freeze terminal output after a specified number of lines. The user is then prompted for more output. The default value is zero, which indicates that no paging will be done. In order to determine a maximum value for a video terminal such that no terminal output will be lost, subtract 2 from the number of lines capable of being displayed at one time by the terminal. Two is the number of lines taken by the prompt message. Thus, for a VT100 with a 24 line screen, SET PAGE 22, will print the maximum amount of lines without any being lost.

- A change has been made so that a person creating a script can force the answer to a prompting question to be fetched from the user terminal instead of from the script. If, in the script, a prompting message is followed by "" instead of the answer to the prompt, then the vds will print that prompt on the terminal and wait for a response to be typed. then the vds will continue executing the script. full documentation on this feature will be available in the next version of the vax-11 diagnostic design guide.

- the rc11 p-table descriptor has been added, so that aztec is now supported with p-table descriptors for the rc11, rc25, and rcf25.

- a new vds service has been added, called $dsGetterm. this service, "get terminal characteristics", can be used in both standalone and user mode to obtain characteristics about the user terminal. you can determine the terminal type, whether it is a video or hardcopy terminal, other information. full documentation on this feature will be available in the next version of the vax-11 diagnostic design guide.

2.0 BUG FIXES

- A problem with ECKAX hanging has been fixed. The CLRVEC routine was changed to reset the clock vector before resetting the clock. This compensates for the TOK hardware bug on 11/750s, which caused interrupts once in a blue moon when resetting the clock.

- $DSPRINT macros now return an error status. SS$NORMAL, SS$BUFFEROVF, or SS$BADPARAM will be returned.

- $DSINLOOP will return DS$NORMAL or DS$ERROR, instead of just setting or clear the low bit of R0.

3.0 MISCELLANEOUS

- LCN Low Cost Nebula will be supported with ENSAA diagnostic Supervisor.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.13 21-Nov-1983

1.0 NEW FEATURES

o

This is the first Diagnostic Supervisor to be linked under version 3 of VMS. Therefore it will not function correctly under those systems running version 2 or earlier versions of VMS.

o

The method of attaching the load path to the CI network has been modified to allow access to a user specified disk on the HSC50 controller. An example of the attach command lines for the 11/780 follows:

```
DS> ATTACH CI780 HUB PAA0 14 5 4
DS> ATTACH CI_NODE PAA0 CIA10 HSC50 10
DS> DISK CIA10 HSC50$DUA1
```

Please note the addition an extra attach command line for DISK and the new form of the generic name. The name must be of the form 'name$DUannn' where 'name' is any alphanumeric string 1 to 6 characters long, and the '$DU' is required. The controller, 'a', is usually 'A' and 'nnn' is the disk unit number.

o A "SET MEMORY n" command has been added which allows the operator to change the amount of total physical memory available to the supervisor and diagnostic programs. This feature is useful when it is desired to verify that a diagnostic program and the supervisor will run in a smaller amount of memory than is actually available on the system.

The unit of measurement for "n" is pages. The value of "n" must be greater than or equal to zero, but less than the last value specified for total available physical memory. Total available physical memory may be reset to the actual available physical memory by using a value of zero for "n". A value greater than the actual available physical memory will be accepted but not used.

The command will not be accepted in user mode. Also note that any devices attached or selected will be deselected and deattached after performing the command.

o An "INITPCS" command has been added. When this command is executed, the supervisor searches for the PCS750.BIN file on the console TU58. If it is found, the supervisor onboard image of the PCS750.BIN is

overwritten with the image found on the TU58. Otherwise, the supervisor onboard image remains the same. Then, the PCS is reloaded with the current PCS image.

This command will be useful when the supervisor being run does not contain the latest PCS750.BIN file, but the file is available on the console TU58. The command is not accepted when running the supervisor under VMS, and will only be accepted when running on an 11/750.

Also, this supervisor contains version CMT098 of PCS microcode patches.

2.0 BUG FIXES

- A bug in the $DS_EXIT TEST macro, which caused an "[aborted]" message to be printed if TRACE was set and the low bit in R0 was clear, has been fixed. Now, the "[aborted]" message will only appear if the program does a $DS_ABORT TEST

3.0 MISCELLANEOUS

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 6.14 20-Feb-1984

1 NEW FEATURES

- New device support is provided for DMZ32.

2 BUG FIXES

- Several supervisor BOOTDRIVRs use the TODR to perform a timeout function. For the 11/750, the TODR remains at zero if the battery backup has failed, and the system has been powered off. Thus, the TODR in this case was not able to perform the timeout function correctly. However, the BOOTDRIVRs were written so that the code would always appear to succeed. This has been fixed by writing a 1 into the TODR (which starts the clock ticking) if the battery backup has failed and the system was powered down. This may fix some of the 'unkown error' problems with supervisor file services for 11/750s.

- The problem of terminal output from a user's control-C handler being suppressed has been fixed. This bug occurred only in standalone mode.

- When a diagnostic is aborted, the supervisor attempts to print out a user PC. When run under VMS, sometimes a PC in VMS system space was printed. This has been fixed.

3 MISCELLANEOUS

- The SET MEMORY command has been enhanced so that ptables for the console load device and the boot path devices are saved. Previously, all ptables were lost.

- The format of supervisor messages which inform the user that an interrupt, trap, or exception has occurred, has been changed to state that the interrupt, exception or trap occurred through an SCB vector rather than just a vector. For example,

?? Reserved/privileged instruction Fault through SCB
vector: 10(X)

instead of,

?? Reserved/privileged instruction Fault through
vector: 10(X)

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 7.0 18-Jul-1984

1 NEW FEATURES

o VMS V4 support

1. Long file name support.

Long filename support has been added to this version of VDS. A full file specification will have the capability for the filename and filetype each to have up to 39 characters maximum. The version number can have up to 5 characters. The directory names can have up to 39 characters and up to 7 nested sub-directories, each with up to 39 characters. The device names can have up to 15 characters and overall the full file specification is not to exceed 252 characters including punctuation.

EXAMPLE: DEVICE:[DIRECTORY]FILENAME.TYPE;VERSION
15 39.39.etc 39 39 5

2. Long device names.

Support code has been added to allow for device names up to 15 characters. VDS will scan 'device name' line for the long device name format. If found VDS will dynamically add device name buffer space to the Ptable,otherwise device names of the format 'ggan' will be handled as before. In both cases, ,the quadword descriptor ,HP$Q_DEVICE, will contain the length and string address of the device name and HP$T_DEVICE will contain the device name with the format of 'ggan'. The new device name format will be as follows: 'name$ggan',the total combination of which must not exceed 15 characters.

3. DIRECTORY/WIDE

In order to display long filenames, the DIRECTORY command now has a /WIDE switch, which causes one filename per line to be printed. A DIRECTORY command without the /WIDE switch will display filenames in four 20-character columns, but for filenames longer than 20 characters, truncation will occur.

4. The value of the symbols IPL$_TIMER and $IPL_SYNCH is changed from 7 to 8 in VMS V4 library LIB. The supervisor, however, will continue to use the old value (7), because of possible conflicts between QIO services

and timer services. In order that diagnostics are encouraged to use the old value, the $IPLDEF macro has been included in the DIAG libraries.

5. This supervisor is not supported on versions of VMS prior to version 4.

2 BUG FIXES

- The $DS_CI_NODE ptable has been changed so that a response of KL10 to the Node_type question produces a value of 6 and a response of CINT a value of 7.

- Some internal supervisor errors of the form:

    ?? Software detected error in ...

  were causing the supervisor to hang in the middle of printing out the rest of this message. This has been fixed.

3 MISCELLANEOUS

- The supervisor now displays a legal statement at boot time regarding the right to use of the diagnostic supervisor and diagnostic programs.

  Two new VDS services, $DS_MOUNT and $DS_DISMOUNT, have been added.

- As of VMS Version 4, you cannot issue a VMS MOUNT command before running the VDS. If you do, and if the device's p-table indicates that the device should be allocated, then the VDS SELECT command will fail. This failure occurs because the SELECT command will try to allocate the device, but you cannot allocate a device that has already been mounted (in VMS V3 you could, if the device had been mounted by you and not someone else).

  Any diagnostic programs that require a device to be allocated AND mounted before it is tested will not work with this release. We know of only one such diagnostic, EVRMD. We are in the process of correcting the problem. The correction will be in a future release.

- When the supervisor is run in online mode (under VMS), the default load path has been changed to be equivalent to SYS$SYSROOT:[SYSMAINT]. Previously, a bug in the supervisor caused the default load device to be the translated value of SYS$DISK from the process name table rather than from the system name table. Normally, SYS$SYSROOT:[SYSMAINT] is the equivalent of SYS$MAINTENANCE, and the supervisor will be changed in some future release to fully translate SYS$MAINTENANCE.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 7.1 18-Jul-1984

1 NEW FEATURES

o

A new VDS macro has been added to return the size of physical memory, expressed in number of pages. The macro is called DS$MEMSIZE and it accepts one argument, the address of a storage location that will receive the memory size.

o

A new VDS command has been added which will print the number and name of the tests in the currently loaded diagnostic program. The command is called SHOW TESTS.

o

A new qualifiers has been added to the SELECT command: /NOALLOCATE. When this qualifier is used the device selected will not be allocated to the VDS user. This parameter should be used when selecting a device that is already allocated to another user. Also, the SHOW DEVICES and SHOW SELECT will display the allocation status of the devices for your current process.

o

The TU81 magtape is now fully supported as a load device. You can now do operation with tapes at 1600 bpi and 6250 bpi.

o

A new naming convention which allows name of the form $m$ggan is now permitted for generic names when you do ATTACH commands. The '$' signs are required and the allocation class 'm' must be a decimal number between 1 and 255. The 'gg' is the normal name with 'a' the controller and 'n' the unit number if required.

2 BUG FIXES

o

A flaw in the operation of the EXAMINE command has been corrected. The fix involved incorrect parsing of an EXAMINE of storage locations 'A' and 'F', which were being confused with the registers AP and FP. The EXAMINE command now works as it should.

o

If the TODR contained a value less than ^X10000000, the supervisor was not recalibrating its software clock from the TODR. This has been fixed by writing the value ^X10000000 to the TODR if the TODR contained less than ^X10000000. Thus, the supervisor will set the TODR at supervisor boot time to ^X10000000 if the TODR contained less than ^X10000000 (^X10000000 is the bias added to the current time of day so that powerfails where the battery back-up also fails can be detected.)

This bug may have caused the supervisor's software time count to become inaccurate when diagnostic programs, which used many $DS_WAITUS or $HIBER calls, were run when the battery back-up had failed.

o Corrected memory data and write bus timeout interrupts were not being forced to use the interrupt stack for the 11/750. This release forces those interrupts to use the interrupt stack. No known symptom had been reported.

o

The command SET PROMPT works properly now as it had not in the past.

3 KNOWN BUGS AND DEFICIENCIES

o

If the 11/750 diagnostic supervisor is booted from a non-console device (a disk), the supervisor is started with cache turned off. This will cause some diagnostic programs, which use supervisor timing services (such as $DS_WAITUS or $DS_WAITMS), to fail in bizarre ways. Therefore, we recommend that you either boot the supervisor from the console TU58, or boot from a disk and then turn on cache with the following supervisor command:

```
DS> DEPOSIT P25 0
```

This will be fixed in the next release of the diagnostic supervisor.

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 8.0
Jan. 7, 1985

1 ANNOUNCING A NEW SUPERVISOR, EDSAA, TO SUPPORT THE VAX-8600!

1.1 New Features Include:

o Dual SBI system, SBIA0 and SBIA1, that must be attached to the hub, which is an ABUS, for the 8600. MBA, UBA and CI adapters are then attached to either SBIA0(SI0), or SBIA1(SI1).

For example, ATTACH SBIA HUB SI0
ATTACH SBIA HUB SI1

Then any adapter is attached to either SI0 or SI1.

With a dual SBI configuration on the 8600, it is possible to have a maximum of 8 Unibus's, 4 on each SBI. To accomodate this situation, changes were made to the DW780 Ptable macro. The number of units was changed to accept 8 maximum. Since 3 bits are needed to specify attaching of DW0-DW7, and the DW780 macro only stores the lower two bits of _DRIVE into the _DVA field, the Unibus address space translates as follows: (using a CSR of 760000)

Unibus 0 and 4 = 2013E000

Unibus 1 and 5 = 2017E000

Unibus 2 and 6 = 201BE000

Unibus 3 and 7 = 201FE000

The recommended way of attaching 8 DW780'S:

for SBIA 0, DW0, DW1, DW2, or DW3,

for SBIA 1, DW4, DW5, DW6, or DW7.

WARNING: DW0 and DW4 should not be attached to the same SBIA since they translate to the same Unibus address space. Likewise for DW1 and 5, DW2 and 6, and DW3 and 7.

o Support for the Environmental Monitoring Module, EMM, is included in this supervisor. If any alerts are received from the EMM module from a power supply, temperature sensor, or air blower, the VDS will report a message on the console and

then resume operation.

o This supervisor also supports control from a remote terminal. To make a remote connection, the terminal control switch of the CPU must be set to REMOTE, and a modem must be connected to the rear async-panel port. Then the user can dial-up the cpu and remote connection will occur. You may dial up at any time, either before booting the supervisor, or while the supervisor is up and running.

  Note that terminal input from local or remote consoles will be echoed on both! Thus, if you are using the remote terminal, you can send messages to a user at the local terminal, and vice versa, by prefixing the message with a "!", which indicates a comment line to the supervisor's command parser.

  Also note that command input from both console terminals is not separated by the diagnostic supervisor, so if both users type a command at the same time, the diagnostic supervisor will see a garbled command and report that an illegal command was typed.

1.2 New Features

1. The "delete" key for video console terminals will now do a backspace, space, backspace sequence, which results in erasure of the previous character.

1.3 Bug Fixes

1. Previously, the DIRECTORY command returned "Total of 0 files." when used on the console storage device and a filename or wildcarded filename was specified as an argument. This has been partially fixed, such that if you specify a filename * extension, or a wildcard filename * extension, or a filename * wildcard extension, or some other combination, the desired result will be correct. Note that not all combinations of wildcarding work, and that this is only a problem for the console storage device.

2. Previously, when the 11/750 supervisor was booted from a non-console disk, it came up with cache turned off (VMB was turning it off). The supervisor now turns the cache on when it is booted or started from 10000.

3. If you turned memory management on (SET MM ON), a RUN or START command occasionally would fail with an "Interrupt stack not valid" halt code (04). This would have occurred most on the slower VAXes (11/730). This has been fixed.

1.4 Known Bugs And Deficiencies

1. When running the supervisor in the on-line mode (under VMS), the supervisor's default load device/directory is set to the first equivalence value returned for the SYS$SYSROOT logical name, plus the string [SYSMAINT]. Normally, this will equate to the system disk plus [SYS0.SYSMAINT], e.g. DRA0:[SYS0.SYSMAINT]. If you have arranged diagnostics in a search list structure, you can SET LOAD to a logical name which is equivalent to the desired search list. For example,

SET LOAD SYS$MAINTENANCE:

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 8.1
Mar. 31, 1985

1 ANNOUNCING A NEW SUPERVISOR, EDSAA, TO SUPPORT THE VAX-8600!

1.1 New Features Include:

o Dual SBI system, SBIA0 and SBIA1, that must be attached to the hub, which is an ABUS, for the 8600. MBA, UBA and CI adapters are then attached to either SBIA0(SI0), or SBIA1(SI1).

For example, ATTACH SBIA HUB SI0
ATTACH SBIA HUB SI1

Then any adapter is attached to either SI0 or SI1.

With a dual SBI configuration on the 8600, it is possible to have a maximum of 8 Unibus's, 4 on each SBI. To accomodate this situation, changes were made to the DW780 Ptable macro. The number of units was changed to accept 8 maximum. Since 3 bits are needed to specify attaching of DW0-DW7, and the DW780 macro only stores the lower two bits of _DRIVE into the _DVA field, the Unibus address space translates as follows: (using a CSR of 760000)

Unibus 0 and 4 = 2013E000

Unibus 1 and 5 = 2017E000

Unibus 2 and 6 = 201BE000

Unibus 3 and 7 = 201FE000

The recommended way of attaching 8 DW780'S:

for SBIA 0, DW0, DW1, DW2, or DW3,

for SBIA 1, DW4, DW5, DW6, or DW7.

WARNING: DW0 and DW4 should not be attached to the same SBIA since they translate to the same Unibus address space. Likewise for DW1 and 5, DW2 and 6, and DW3 and 7.

o Support for the Environmental Monitoring Module, EMM, is included in this supervisor. If any alerts are received from the EMM module from a power supply, temperature sensor, or air blower, the VDS will report a message on the console and

then resume operation.

o This supervisor can be booted and controlled from a remote terminal. To make a remote connection, the terminal control switch of the CPU must be set to REMOTE, and a modem must be connected to the rear async-panel port. Then the user can dial-up the cpu and remote connection will occur. The supervisor can be booted from the remote terminal by the usual commands. If the supervisor has already been booted from the CTY, make the remote connection, type "^P" from the remote terminal, then "CONT".

Note that terminal input from local or remote consoles will be echoed on both! Thus, if you are using the remote terminal, you can send messages to a user at the local terminal, and vice versa, by prefixing the message with a "!", which indicates a comment line to the supervisor's command parser.

Also note that command input from both console terminals is not separated by the diagnostic supervisor, so if both users type a command at the same time, the diagnostic supervisor will see a garbled command and report that an illegal command was typed.

1.2 New Features

1. Several new p-tables have been added. They are: DHU11, DHV11, DR11C, DX20, TU70, and TU72.

2. The number of terminals that can be attached has been increased from 8 to 255 to better support LAT configurations.

1.3 Bug Fixes

1. The /NOALLOCATE switch has been fixed to modify the complete list of devices specified, not just the first device in the list.

2. Previously, when SET P was entered, the supervisor would assume that this was short for SET PROMPT. This is an ambiguity however, since there is also a SET PAGE command. Now, more than just the leading P must be entered to make the command unique.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 8.2 June 24, 1985

1 NEW FEATURES

1. The number of UNA11 units that can be attached has been increased from 1 to 8.

2 BUG FIXES

1. When booting from a disk connected to an HSC50 which is connected to a CI780 or CI750, the ptable for the CI750/CI780 would sometimes contain the incorrect values for BR and slot number. This has been fixed.

2. When booting from a CI750 disk, the CI750 ptable (PAa0) device name may have been incorrect. This has been fixed.

3 KNOWN BUGS

1. None.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 8.3 September 23, 1985

1. NEW FEATURES:

1. /BASE and /NOBASE Qualifiers:

The /NOBASE qualifier allows zero to be used in the address calculation for the given command, instead of the default base address.

The /BASE:temp_base allows a temporary base address to be used in the address calculation for the given command, instead of the default base address.

These new qualifiers affect the commands which use the base address set by the SET BASE command: DEPOSIT, EXAMINE, SET BREAKPOINT, and CLEAR BREAKPOINT.(q(_.PARAGRAPH On SET, and CLEAR BREAKPOINT, the /NOBASE qualifier can be abbreviated to /N. For EXAMINE and DEPOSIT, /NOBASE can be abbreviated to /NO, thus keeping /N as the abbreviation for /NEXT:n.

On SET, and CLEAR BREAKPOINT, the /BASE:temp_base qualifier can be abbreviated to /B. For EXAMINE and DEPOSIT, /BASE:temp_base can be abbreviated to /BA:temp_base, thus keeping /B as the abbreviation for /BYTE.

These qualifiers may be placed anywhere on the command line after the command name.

EXAMPLE:

```
SET BASE 100                        ! Sets Default Base.
DEPOSIT /BASE:200 44 555            ! Address is 244.
EXAMINE /NOBASE 244                 ! Address is 244.
   00000244:     555
```

2. New Ptables:

New Ptables have been added for: LN03, LA210, VT61, VT71, VT131, and VT132.

2. Bug Fixes:

1. Console Boot Driver HALT:

If an attempt was made to load a file from the console device when mapping is enabled, the VDS would halt in the console boot driver in the routine DO_MAPPING where physical to virtual mapping would occur, but is not

implemented yet. This problem has been corrected.

2. Online as Subprocess Hang:

   A bug was fixed that was causing the Diagnostic Supervisor to hang when run on-line as a sub-process, communicating via a mailbox. Additionally, the SET WIDTH command will return an error message when run with a mailbox as output.

3. HSC Failover:

   VMB uses bits <15:08> and <07:00> to test for failover to an "alternate" HSC. Diagboot and the Diag Super expect the low byte [<7:0>] to be the "first try" HSC, but VMB uses the second byte [<15:08>] as the "first try" HSC, if the second byte is NON ZERO. If there is a non-zero value in the second byte in R2, the Diag Super will not load, run or get directories in Sysmaint.

   The problem has been resolved by having VDS use the method that VMB uses, as described above.

3. Known Bugs:

   1. NONE.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 9.0 Dec. 23, 1985

1. Announcing a new supervisor, EBSAA, to support the VAX-8200!

This version, 9.0, contains the 8200 supervisor. Support includes:

o A new processor (device type KA820) which contains 4 SLU's (Serial Line Units), a new backplane interconnect called the VAXBI, and a UNIBUS adapter (device type DWBUA) and a BI to SDI adapter (device type KDB50). The KDB50 is used to interface to RA60, RA80, RA81, and RA82 type disks. Since the VAXBI is known as the HUB, these adapters are attached as follows:

```
DS>
DS> ATTACH KA820 HUB KA0    ! cpu
K-bytes of Main Memory? 1024
BI Node Number (HEX)? B
DS>
DS> ATTACH SLU KA0 TCA0
Serial Line Externally Wrapped? Yes
Baud Rate? 9600
DS>
DS> ATTACH DWBUA HUB DW0    ! UNIBUS adapter
BI Node Number (HEX)? 2
B? 4
DS>
DS> ATTACH KDB50 HUB DUA    ! KDB50 adapter
BI Node Number (HEX)? 7
B? 5
```

o Customer Runnable Diagnostics are not supported on the 8200. The console boot command: T/M, which would ordinarily start CRD in Menu mode, boots the Vax Diagnostic Supervisor.

o The $DS_CHANNEL service has been enhanced to support BI adapters.

- The INITA function supports the BUA and the KDB50. It also accepts as an optional argument, time, which is the number of ten-milliseconds to wait for the BI node self-test to complete. If the time argument is not specified, a default of 100 milliseconds is used for the BUA, and 10 seconds for the KDB50.

- There are two new functions, SELF_TEST and STOP, which are used only for BI adapters. SELF_TEST will initiate the self-test for any BI adapter. This function also accepts the time argument (default is 10 seconds).

The STOP function stops a BI adapter from issuing any more BI transactions.

- The STATUS and ENINT functions accept a new argument (bistsadr) which specifies the address of a quadword which returns the BIIC CSR and the BIIC BER.

- The MACRO-32 format is now:

  $DS_CHANNEL_x, unit, func, [vecadr], [stsadr], [time], [bistsadr]

- The BLISS-32 format is:

  $DS_CHANNEL (UNIT=unit, FUNC=func, [VECADR=vecadr], [STSADR=stsadr],
  [TIME=time], [BISTSADR=bistsadr]);

- Two new return codes have been defined: DS$_BIIC indicates that the BIIC self-test has failed, and DS$_NODE indicates that the BI node self-test failed.

- In addition to the following fields in status-1,

  CHS$V_SYSERR
  CHS$V_CHNERR
  CHS$V_DEVERR
  CHS$V_PGMERR
  CHS$V_DEVBUS
  CHS$V_PGMHDE
  CHS$V_BUSPDN
  CHS$V_BUSNXM,

  the following new bits have been defined:

  CHS$V_UIE - Bit 28 - UNIBUS Interlock Error; set if a DATO(B) does not follow a DATIP transaction on the UNIBUS.

  CHS$V_BADBDP - Bit 29 - Bad Buffered Data Path; set if data path 6 or 7 is selected.

  CHS$V_BADPAR - Bit 30 - Set if RAM parity error occurred.

- The bistsadr argument points to a quadword which contains 2 longwords. The first is the BIIC Control and Status Register. The second is the BIIC Bus Error Register. Bit definitions are defined by the $BIICDEF macro. Bit definitions are as follows:

  BIIC CSR: BIIC Control and Status Register

  BIIC$V_NODE_ID,BIIC$S_NODE_ID - Bits 0-3 - Node ID
  BIIC$V_ARBCNTL,BIIC$S_ARBCNTL - Bits 4-5 - Arbitration Mode
  BIIC$V_SEIE,BIIC$M_SEIE - Bit 6 - Soft Error Interrupt Enable
  BIIC$V_HEIE,BIIC$M_HEIE - Bit 7 - Hard Error Interrupt Enable
  BIIC$V_UWP,BIIC$M_UWP - Bit 8 - Unlock Write Pending

```
BIIC$V_SST,BIIC$M_SST - Bit 10 - Start Self_Test
BIIC$V_STS,BIIC$M_STS - Bit 11 - Self-Test Status
BIIC$V_BROKE,BIIC$M_BROKE - Bit 12 - Broke
BIIC$V_INIT,BIIC$M_INIT - Bit 13 - Init
BIIC$V_SES,BIIC$M_SES - Bit 14 - Soft Error Summary
BIIC$V_HES,BIIC$M_HES - Bit 15 - Hard Error Summary
BIIC$V_BIICTYPE,BIIC$S_BIICTYPE - Bits 16-23 - BIIC Interface
                                              Type
BIIC$V_BIICREVN,BIIC$S_BIICREVN - Bits 24-31 - BIIC Interface
                                              Revision

BIIC BER:  BIIC Bus Error Register

BIIC$V_NPE,BIIC$M_NPE - Bit 0 - Null Bus Parity Error
BIIC$V_CRD,BIIC$M_CRD - Bit 1 - Corrected Read Data
BIIC$V_IPE,BIIC$M_IPE - Bit 2 - ID Parity Error
BIIC$V_UPEN,BIIC$M_UPEN - Bit 3 - User Parity Enable
BIIC$V_ICE,BIIC$M_ICE - Bit 16 - Illegal Confirmation Error
BIIC$V_NEX,BIIC$M_NEX - Bit 17 - Non-Existent Address
BIIC$V_BTO,BIIC$M_BTO - Bit 18 - Bus Timeout
BIIC$V_STO,BIIC$M_STO - Bit 19 - Stall Timeout
BIIC$V_RTO,BIIC$M_RTO - Bit 20 - Retry Timeout
BIIC$V_RDS,BIIC$M_RDS - Bit 21 - Read Data Substitute
BIIC$V_SPE,BIIC$M_SPE - Bit 22 - Slave Parity Error
BIIC$V_CPE,BIIC$M_CPE - Bit 23 - Command Parity Error
BIIC$V_IVE,BIIC$M_IVE - Bit 24 - IDENT Vector Error
BIIC$V_TDF,BIIC$M_TDF - Bit 25 - Transmitter During Fault
BIIC$V_ISE,BIIC$M_ISE - Bit 26 - Interlock Sequence Error
BIIC$V_MPE,BIIC$M_MPE - Bit 27 - Master Parity Error
BIIC$V_CTE,BIIC$M_CTE - Bit 28 - Control Transmit Error
BIIC$V_MTCE,BIIC$M_MTCE - Bit 29 - Master Loopback Error
BIIC$V_NMR,BIIC$M_NMR - Bit 30 - NOACK to multi-responder
                                 command Received
```

- The message which is printed when an unexpected interrupt occurs from a BI adapter (SCB vectors 100 - 1FC) has been changed to include the BI node number, and the contents of that node's Bus Error Register with a bit-to-text translation.

- When a machine check stack frame is printed, if the BI ERROR bit (4) in the machine check typecode longword is set, the cpu's Bus Error Register is also included.

- The duration of the boot process may vary because it runs self-test on all adapters on the system except the cpu and memory.

2. Vax Diagnostic Supervisor Support for the 8650.

EDSAA has been updated to include support for the 8650. The 8650 represents a 44% speed improvement over the 8600, plus larger 32 M memory boards.

3. Vax Diagnostic Supervisor Support under ULTRIX.

- Standalone ULTRIX is fully supported in all Supervisors linked now. Any disk for which VDS supports the ODS file structure is automatically supported for ULTRIX. Operation is the same except for the following changes made for the DIRECTORY, LOAD, SET LOAD and RUN commands. ALL VDS COMMAND MAY BE UPPER OR LOWER CASE LETTERS.

- Legal filename for VDS ULTRIX directories and files can include the symbols A-z, 0-9,. , and * or % for wildcard matching.

- BOOTING VDS SYSTEMS:

  Booting is the same as now except that your default directory will be ddan:(0,p)field unless the directory value is found in the RPB$T_FILE file of the RPB in which case that will be used. The exception to this is the 11/750 system. The command 'B/60000010' is used instead of 'B/10' in order to set the partition to '6'.

  For the 11750 you use "B/60000010" and that boots the device selected by the switch on the front panel. For the 8200 you use "B/R5:60000010". The boot cmd files for the 11/730 has a "D RB 60000010" command in it which does a deposit to R11 to boot VDS. The 11/780/785 boot cmd files have a "D R11 60000010" command in them to boot VDS under ULTRIX.

- SET LOAD:

  To run from an ULTRIX disk you did not boot from you will have to issue a SET LOAD command of the form ddan:(0,p)directory. The parentheses and enclosed information are required if you are currently running from an ODS disk. The '0' is a ULTRIX unit number which is not currently implemented and therefore set to 0. The p is a partition number and is generally set to 6 for partition 'G'. The 'directory' value is generally set to 'field' for diagnostics but can be any values (for examples, usr, WORD, field/user/source). The '/' is a delimiter and is use to separate directory names. If you are currently in ULTRIX mode, you can return to the VMS compatible mode by giving a SET LOAD with ddan:[name] where the brackets and directory are required to change over. Once in ULTRIX mode any of the following are valid:

  SET LOAD ddan:(0,p)field/user
  SET LOAD (0,p)field/user

SET LOAD field/user

If a device or '(0,p)' is not specified they are not changed.

o DIRECTORY:

The directory command has several items to note. Assume you have the files evdba.exe and evkab.hlp in the directory DMA0:(0,6)field If you enter DIRECTORY you will get:

evkab.exe evkab.hlp

total of 2 files.

As output. If you do a SET LOAD to field/evkab.exe and then enter the DIRECTORY command you will get

evkab.exe

total of 1 files.

The system will automatically check if the last link given is a valid directory and if not will back up to the previous link and assume you are looking for one particular file which it finds. This is what you would also get if your directory was 'field' and you entered any of the following:

DIRECTORY evkab.exe
DIRECTORY field/evkab.exe
DIRECTORY (0,6)field/evkab.exe
DIRECTORY DMA0:(0,6)field/evkab.exe

Wildcard are support as for ODS disks, but you should be aware that NO DEFAULTS ARE ASSUMED. An ULTRIX file spec does not have the required format of a VMS type file. You will have to enter the name and extension with either actual or wildcard values. The wildcard name may be any combination of upper or lower case letters. The results from a DIRECTORY command will yield all matching files without regards to upper or lower case letters.

o LOAD or RUN:

These commands work similar as before. If you specify a directory link with the file name, you must give the entire string of links. For example if your file eckam.exe is in DMA0:(0,6)field/usr/word directory, and you have done a SET LOAD to this, the command LOAD word/eckam.exe will not work. Any of the following commands will work.

LOAD eckam.exe

```
LOAD field/usr/word/eckam.exe
LOAD (0,6)field/usr/word/eckam.exe
LOAD DMA0:(0,6)field/usr/word/eckam.exe
```

The same format is valid for RUN also.

o RMS protocols:

ULTRIX files do not have a structure like VMS files. However support has been added to make all the old commands be compatible. Sequential and Random (by RFA) access is supported.

1. $OPEN

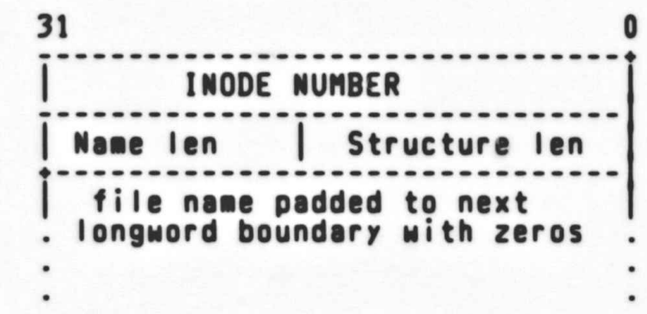
   If you do an $OPEN on a file with the last 4 characters of its $FAB DNM set to .dir (letters any case) the file you get will be the directory the file is in unless it is a directory is which case you will get it.

2. $READ

   This will read in the file to the location you specified. You will have to do any necessary file structure interpretation.

3. $GET

   If the opened file has the last 4 characters of its $FAB DNM set to .hlp or to the length of the line. If the opened file has the last 4 characters of its $FAB DNM set to .dir (letters any case) this will yield a 'record' of the form:



   If anything else is specified for DNM you will get 512 bytes or remaining bytes in the files and you interpret the structure.

This information is subject to change for compatibility with the online support and updates to ULTRIX. Welcome to the world of ULTRIX.

4. A new supervisor service, $DS_PRINTREV has been added.

The Print Revision Level service may be used in diagnostic programs that test devices for which a hardware, firmware, and/or patch revision level is accessible to the program. Such programs may want to indicate whether the device's revision levels are compatible with the revision levels expected by the program.

This service will:

- Compare actual and expected device revision levels specified by the diagnostic program
- Display a message indicating the device's revision level (hardware firmware, or patch), and also indicating whether the revision levels being compared are hardware levels, firmware levels, or patch levels. A module number may be printed, and hardware revision numbers may be converted to a letter code.
- Cause the hardware revision level, firmware revision level and/or patch level to be included in all error messages (generated via $DS_ERRxxxx macros) that are subsequently displayed for the unit in question.

If the device being tested has a software-readable hardware, firmware, or patch level, the $DS_PRINTREV service should be called for each selected device. This call(s) should be made in the "pass 0" section of the initialization code.

If the device has all three (hardware, firmware, and patch) revision types, then the service should be called three times for each selected device; once to report the hardware revision, once for the firmware revision, and once for the patch level. For each logical unit, $DS_PRINTREV can be called four times for each type of revision level. For example, it would be desirable to call $DS_PRINTREV twice for hardware revision levels if a logical unit was a 2-board device, and each board had an associated hardware revision level.

When called, the service will (at the programmer's option) immediately display a message indicating the actual revision level and/or messages indicating whether the actual revision level matches the revision level expected by the diagnostic program. One of the following messages may be printed. (DEVNAME will be replaced by the generic name of the unit in question.)

```
Hardware revision level of DEVNAME (T1002):    ******
Hardware revision level of DEVNAME (T1002) is less than that expected by
diagnostic program.
       Hardware revision level:                ******
       Revision level expected by program:     ******

Hardware revision level of DEVNAME (T1002) is equal to that expected by
diagnostic program.
       Hardware revision level:                ******
       Revision level expected by program:     ******

Hardware revision level of DEVNAME (T1002) is greater than that expected by
diagnostic program.
       Hardware revision level:                ******
       Revision level expected by program:     ******
If firmware or patch levels are being reported, the word
"hardware" in the messages will be replaced with the word
"firmware" or "patch".

If the modulenum argument has been passed, the module number
will be printed for the hardware, firmware, and patch
messages associated with that module number. If the number
is not passed, it will not appear in the above messages.

Each time an error service is called ($DS_ERRxxxx) for a unit
for which the $DS_PRINTREV macro has been called, the
hardware revision level, and/or firmware revision level
and/or patch level will be displayed as part of the error
message. The message will be displayed AFTER all of the
error text has been printed. It will be inhibited if the IE1
flag is set. The format of this display will be:
******  program title          program rev. ******
Pass #, test #, subtest #, error #, date
Hard error while testing DEVNAME:  text

       text

Revision level(s) for DEVNAME:
       (T1002): Hardware = X; Firmware = X; Patch = X;
       (T1003): Hardware = Y; Firmware = Y; Patch = Y;

****** End of hard error number # ******

(If any of the levels or the module number are not being
reported, they will not appear in the revision level
message.)

MACRO-32 Format:

      $DS_PRINTREV_x log_unit, actual_rev, expected_rev, rev_type,
                          [printmask], [prlink], [modulenum]
```

BLISS Format:

```
$DS_PRINTREV   (LOG_UNIT=log_unit, ACTUAL_UNIT=actual_rev,
                        EXPECTED_REV=expected_rev, REV_TYPE=rev_type,
                        [PRINTMASK=printmask], [PRLINK=prlink],
                        [MODULENUM=modulenum]);
```

log_unit

Logical unit number of device whose revision level is to be checked.

actual_rev

Longword containing the actual hardware, firmware, or patch revision level for the device whose logical unit number is specified with the "logunit" parameter. See note 4. (The diagnostic program determines this value by accessing the hardware.)

expected_rev

Longword containing the hardware, firmware, or patch revision level which was expected by the diagnostic program. See note 4. (This value is placed in the diagnostic program at compilation time.)

rev_type

Longword mask indicating the type of revision code (hardware, firmware, or patch) being compared. Bit definitions are:

- Bit 0 - PRVV_HWREV,PRVM_HWREV - Set this bit to indicate a hardware revision level.
- Bit 1 - PRVV_FWREV,PRVM_FWREV - Set this bit to indicate a firmware revision level.
- Bit 2 - PRVV_PREV,PRVM_PREV - Set this bit to indicate a patch level.

(Only one bit should be set per service call.)

printmask

Longword mask used to control $DS_PRINTREV functions. Bit definitions are:

- Bit 0 - PRMSK$V_LSS,PRMSK$M_LSS - if set, inhibits message that would normally be displayed if the actual revision is less than expected.

- Bit 1 - PRMSK$V_EQL,PRMSK$M_EQL - if set, inhibits message that would normally be displayed if the actual revision is equal to that expected.

- Bit 2 - PRMSK$V_GTR,PRMSK$M_GTR - if set, inhibits message that would normally be displayed if the actual revision is greater than expected.

- Bit 3 - PRMSK$V_ALWAYS,PRMSK$M_ALWAYS - if set, displays the message which lists only the actual revision level.

- Bit 4 - PRMSK$V_TRANSL,PRMSK$M_TRANSL - if set, will convert a hardware revision number to a letter code which represents the functional revision as defined by DEC STD 012. Setting this bit will not result in conversion for firmware or patch revisions. The conversion pattern is shown in table 1. See note 6.

Table 1: Revision Number to Letter Conversion

Field contents [bits <n:0>]	Functional Rev.
0.....0000	pre-release
0.....0001	A
0.....0010	B
0.....0011	C
.	.
.	.
0....11010	Z
0....11011	AA
0....11100	AB
0....11101	AC
.	.
.	.
0...110100	AZ
0...110101	BA
.	.
.	.
0..1001110	BZ
.	.
.	.
1010111110	ZZ

For example, if only bits 1 and 2 are set then a message will be displayed only if the actual revision is less than that expected by the program.

By default, bits 0, 1, 2, 3, and 4 are clear, in which case no conversion will be performed on revision numbers. In addition, a message will be displayed stating whether the actual revision is greater than, less than, or equal to the revision expected by the program. Inhibiting messages does not affect the inclusion of the hardware, firmware or patch revision levels in error messages generated by $DS_ERRxxxx macros.

prlink

Address of a message routine. This optional routine will have the same format as an error reporting routine used with a $DS_ERRxxxx macro. That is, the routine should be delineated with $DS_BGNMESSAGE and $DS_ENDMESSAGE macros and should use $DS_PRINTB and $DS_PRINTX macros for displaying text.

modulenum

Address of a counted ASCII string which represents the unique module number for a board (e.g. "T1002"). If you supply this argument for a hardware revision, then you must supply the same argument for any firmware or patch revision associated with that module.

Notes:

1. Symbols are defined by the $DS_PRINTREV_DEF macro.

2. This macro should only be used for devices having software-accessible hardware, firmware, or patch revision numbers.

3. Some programmer's may want the text displayed only in certain circumstances. For example, perhaps it is the concensus of a particular program's users that the text only be displayed if the actual revision level is lower than that expected by the diagnostic program. In such a case, the "printmask" parameter may be used to inhibit one or more of the possible warning messages.

4. The format of the revision level is irrelevant, but the values passed via the "actual_rev" and "expected_rev" parameters must be in the same format. The service will perform an unsigned comparison of the two values to determine which message to display. Revision numbers will be displayed as unsigned decimal values unless conversion to a letter code is desired.

5. This macro should be used to determine whether current versions of diagnostics are incompatible with old hardware. Since new diagnostics may only run on hardware above a certain revision level, the diagnostic should only set bits 1 and 2 in printmask. This causes a message to be displayed only if the actual revision level is less than that expected by the program.

6. Normally, the hardware revision level should be converted to the corresponding functional revision letter code by setting bit 4 in printmask.

7. If you do not set bit 2 in printmask, a misleading message may result if the actual hardware, firmware, or patch levels were updated without a corresponding change to the diagnostic's expected revision levels.

5. Additions & Fixes.

   1. Support has been added for new devices: RD52, RD53, RA82, and LUA11.

   2. Channel service for DSINT was returning SS$_NORMAL when DSINT was documented as returning DS$_NORMAL. This has been corrected.

   3. A new control flag, AskPrompt, has been added in order to support the new Automated Test and Verification system, SAVES/AVS. When this flag is set, a special prompt, "ASK>", is appended to the user's prompt which was specified for any of the $DSASKxxxx macros. The flag is clear by default.

6. Known Bugs and Deficiencies:

   1. The console command language uses the device names CSA1 and CSA2 for the console floppy drives. The supervisor, however, always refers to drive CSA1 as CSA0. Furthermore, although the supervisor can be booted from either floppy drive, files can only be accessed from CSA1. This will be fixed in a future release.

   2. When the supervisor is booted, it attempts to automatically ATTACH all of the devices in the boot path. In order to do this, a certain amount of sizing must be done in order to determine the controller letter and drive number of the device name for the disk which was booted from. If you have more than one UDA50 configured on the system, or a combination of UDA50s and KDB50s, and you boot from a disk other than a disk which is connected to the adapter with the lowest node id, the controller letter portion of the device name will not agree with the controller letter assigned by the autosizer.

   3. Ultrix mode problems:

      1. Some commands are recognized in upper case only. All commands are recognized in upper case, so this is the temporary work-around.

      2. Ultrix differentiates between upper and lower case in file names. The VDS maintains this distinction. Therefore, the case of each letter must be correct in order for the VDS to find the file.

3. The Ultrix file system does not support file version numbers as does VMS. Like-wise, the VDS does not support version numbers when referencing Ultrix file systems. Version numbers are sometimes created by the DECnet-Ultrix dcp command. Any files with version numbers must be renamed. Under Ultrix, this is done with the mv command; specifying the file name in single quotes for it to accept the version number. For example: mv 'ECSAA.EXE;1' ecsaa.exe.

4. All scripts run by the VDS must have their file names in upper case. For example, ATT.COM, not att.com.

5. When specifying a file name to the VDS, the complete file name must be specified. For example, @ATT.COM or RUN evsba.exe.

6. If a non-system disk is being used as the load device, that disk must have file systems on both the a and g partitions, with the files wanted in the g partition's file system. No links have to exist between these file systems, however. A file system is created with the Ultrix newfs command. Care must be used, as this command is similar to the VMS INITIALIZE command and will destroy data on the disk.

7. The VDS INITPCS command does not work.

8. When specifying command modifiers, they must be placed on the command verb and not on the file name. For example, use RUN/TEST:3:4 evkab.exe, not RUN evkab.exe/TEST:3:4. The / used to indicate a command modifier to the VDS is also used in Ultrix as part of a file specification.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 9.1 Jan 20,1986

1. Announcing a new supervisor, EZSAA, to support the VAX-8800!

This version, 9.1, contains the 8800 supervisor

o NAUTILUS I/O consists of the new corporate interconnects: BI,NI,SI,CI and will have 4 I/O channels.

The NAUTILUS NMI-BI (NBI) adapter interfaces the NAUTILUS Memory Interconnect (NMI) bus to two backplane Interconnect (BI)busses.

The NBI is partitioned into three main sections,the NMI port,NBI transaction buffer and two BI users ports,the first two sections are contained on an extended hex module mounted in the cpu backplane (NBIA). Each BI user port is contained on a BI Eurocard (NBIB) located in the BI backplane.

Any NAUTILUS will have a maximum of 2 NBIA's,permitting a maximum of 4 seperate BI channels.

EZSAA has also support for the following adapters:

- BUA, a unibus adapter,device type DWBUA
- BLA, BI to LESI adapter,device type DWBLA. This adapter is primarily intended for TAPE interface on NAUTILUS systems.
- BCI, CI to BI computer interconnect,device type CIBCI.
- BDA, BI to SDI adapter,device type KDB50 The BDA is the primary disk controller for NAUTILUS and will provide the pathway to SI based disks only.

The NMI is known as the HUB,these adapters are attached as follows:

```
DS>
DS> ATTACH KA880 HUB KA0            ! cpu
PRIMARY ? YES
DS>
DS> ATTACH NBIA HUB NBIA0           ! NBI adapter
LOGICAL ADAPTER # 0
DS>
DS> ATTACH NBIB NBIA0 NBIB0         ! NBI adapter
BI # ? 0
```

```
BI Node Number (HEX) ? 2
DS>
DS> ATTACH NBIB NBIA0 NBIB1          ! NBI adapter
BI # ? 1
BI Node Number (HEX) ? 3
DS>
DS>
DS> ATTACH NBIA HUB NBIA1            ! NBI adapter
LOGICAL ADAPTER # 1
DS>
DS> ATTACH NBIB NBIA1 NBIB2          ! NBI adapter
BI # ? 0
BI Node Number (HEX) ? 0
DS>
DS> ATTACH NBIB NBIA1 NBIB3          ! NBI adapter
BI # ? 1
BI Node Number (HEX) ? 1
DS>
DS>
DS> ATTACH DWBUA NBIB0 DW0           ! BUA adapter
BI Node Number (HEX) ? 5
BR ? 4
DS>
DS> ATTACH DWBLA NBIB0 BLA0          ! BLA adapter
BI Node Number (HEX) ? 6
DS>
DS> ATTACH CIBCI NBIB0 PAA0          ! BCI adapter
BI Node Number (HEX) ? 7
BR ? 5
CI Node ? 2
DS>
DS> ATTACH KDB50 NBIB0 DUA ! BDA adapter
BI Node Numbe (HEX) 8
BR ? 6
DS>
```

The LOGICAL ADAPTER #,BI # and the BI Node Number are not arbitrary values, but are based upon the NAUTILUS system I/O configuration, these values are used by the supervisor to construct the device address for each NBIA and BI node that is attached and therefore must be known at attach time.

However,the adapter's drive numbers don't require that restriction,but, because of Vax Diagnostic Supervisor convention,it is advisable that these numbers are assigned in a sequential order.

- Prior to the loading of diagnostics,a console ASSIGN command must be issued ,this will associate the logical name CSA1 with the winchester RD disk. This ASSIGN command can be done prior to the booting of the supervisor or as described below if the supervisor has already been booted.

```
DS> ^p                      ; back to console mode
>>> halt                    ; halt the processor
>>> assign csa1 dw1:        ; do the assign command
>>> continue                ; back to supervisor control
DS> load EVXXX              ; proceed to load diagnostic
```

o Customer Runnable Diagnostics are not supported on the 8800.

o The $DS_CHANNEL service has been enhanced to support BI adapters.

- The INITA function supports the BUA and the KDB50. It also accepts as an optional argument, time, which is the number of ten-milliseconds to wait for the BI node self-test to complete. If the time argument is not specified, a default of 320 milliseconds is used for the BUA, and 10 seconds for the KDB50.

- There are two new functions, SELF_TEST and STOP, which are used only for BI adapters. SELF_TEST will initiate the self-test for any BI adapter. This function also accepts the time argument (default is 10 seconds).

  The STOP function stops a BI adapter from issuing any more BI transactions.

- The STATUS and ENINT functions accept a new argument (bistsadr) which specifies the address of a quadword which returns the BIIC CSR and the BIIC BER.

- The MACRO-32 format is now:

  ```
  $DS_CHANNEL_x, unit, func, [vecadr], [stsadr], [time],
                 [bistsadr]
  ```

- The BLISS-32 format is:

  ```
  $DS_CHANNEL (UNIT=unit, FUNC=func, [VECADR=vecadr], [STSADR=
               stsadr], [TIME=time], [BISTSADR=bistsadr]);
  ```

- Two new return codes have been defined: DS$_BIIC indicates that the BIIC self-test has failed, and DS$_NODE indicates that the BI node self-test failed.

- In addition to the following fields in status-1,

  ```
  CHS$V_SYSERR
  CHS$V_CHNERR
  CHS$V_DEVERR
  CHS$V_PGMERR
  CHS$V_DEVBUS
  CHS$V_PGMHDE
  CHS$V_BUSPDN
  ```

CHS$V_BUSNXM,

the following new bits have been defined:

CHS$V_UIE - Bit 28 - UNIBUS Interlock Error; set if a DATO(B) does not follow a DATIP transaction on the UNIBUS.

CHS$V_BADBDP - Bit 29 - Bad Buffered Data Path; set if data path 6 or 7 is selected.

CHS$V_BADPAR - Bit 30 - Set if RAM parity error occurred.

- The bistsadr argument points to a quadword which contains 2 longwords. The first is the BIIC Control and Status Register. The second is the BIIC Bus Error Register. Bit definitions are defined by the $BIICDEF macro. Bit definitions are as follows:

BIIC CSR: BIIC Control and Status Register

BIIC$V_NODE_ID,BIIC$S_NODE_ID - Bits 0-3 - Node ID
BIIC$V_ARBCNTL,BIIC$S_ARBCNTL - Bits 4-5 - Arbitration Mode
BIIC$V_SEIE,BIIC$M_SEIE - Bit 6 - Soft Error Interrupt Enable
BIIC$V_HEIE,BIIC$M_HEIE - Bit 7 - Hard Error Interrupt Enable
BIIC$V_UWP,BIIC$M_UWP - Bit 8 - Unlock Write Pending
BIIC$V_SST,BIIC$M_SST - Bit 10 - Start Self_Test
BIIC$V_STS,BIIC$M_STS - Bit 11 - Self-Test Status
BIIC$V_BROKE,BIIC$M_BROKE - Bit 12 - Broke
BIIC$V_INIT,BIIC$M_INIT - Bit 13 - Init
BIIC$V_SES,BIIC$M_SES - Bit 14 - Soft Error Summary
BIIC$V_HES,BIIC$M_HES - Bit 15 - Hard Error Summary
BIIC$V_BIICTYPE,BIIC$S_BIICTYPE - Bits 16-23 - BIIC Interface Type
BIIC$V_BIICREVN,BIIC$S_BIICREVN - Bits 24-31 - BIIC Interface Revision

BIIC BER: BIIC Bus Error Register

BIIC$V_NPE,BIIC$M_NPE - Bit 0 - Null Bus Parity Error
BIIC$V_CRD,BIIC$M_CRD - Bit 1 - Corrected Read Data
BIIC$V_IPE,BIIC$M_IPE - Bit 2 - ID Parity Error
BIIC$V_UPEN,BIIC$M_UPEN - Bit 3 - User Parity Enable
BIIC$V_ICE,BIIC$M_ICE - Bit 16 - Illegal Confirmation Error
BIIC$V_NEX,BIIC$M_NEX - Bit 17 - Non-Existent Address
BIIC$V_BTO,BIIC$M_BTO - Bit 18 - Bus Timeout
BIIC$V_STO,BIIC$M_STO - Bit 19 - Stall Timeout
BIIC$V_RTO,BIIC$M_RTO - Bit 20 - Retry Timeout
BIIC$V_RDS,BIIC$M_RDS - Bit 21 - Read Data Substitute

BIIC$V_SPE,BIIC$M_SPE - Bit 22 - Slave Parity Error
BIIC$V_CPE,BIIC$M_CPE - Bit 23 - Command Parity Error
BIIC$V_IVE,BIIC$M_IVE - Bit 24 - IDENT Vector Error
BIIC$V_TDF,BIIC$M_TDF - Bit 25 - Transmitter During Fault
BIIC$V_ISE,BIIC$M_ISE - Bit 26 - Interlock Sequence Error
BIIC$V_MPE,BIIC$M_MPE - Bit 27 - Master Parity Error
BIIC$V_CTE,BIIC$M_CTE - Bit 28 - Control Transmit Error
BIIC$V_MTCE,BIIC$M_MTCE - Bit 29 - Master Loopback Error
BIIC$V_NMR,BIIC$M_NMR - Bit 30 - NOACK to multi-responder command Received

2. A new supervisor service, $DS_PRINTREV has been added.

The Print Revision Level service may be used in diagnostic programs that test devices for which a hardware, firmware, and/or patch revision level is accessible to the program. Such programs may want to indicate whether the device's revision levels are compatible with the revision levels expected by the program.

This service will:

- Compare actual and expected device revision levels specified by the diagnostic program
- Display a message indicating the device's revision level (hardware firmware, or patch), and also indicating whether the revision levels being compared are hardware levels, firmware levels, or patch levels. A module number may be printed, and hardware revision numbers may be converted to a letter code.
- Cause the hardware revision level, firmware revision level and/or patch level to be included in all error messages (generated via $DS_ERRxxxx macros) that are subsequently displayed for the unit in question.

If the device being tested has a software-readable hardware, firmware, or patch level, the $DS_PRINTREV service should be called for each selected device. This call(s) should be made in the "pass 0" section of the initialization code.

If the device has all three (hardware, firmware, and patch) revision types, then the service should be called three times for each selected device; once to report the hardware revision, once for the firmware revision, and once for the patch level. For each logical unit, $DS_PRINTREV can be called four times for each type of revision level. For example, it would be desirable to call $DS_PRINTREV twice for hardware revision levels if a logical unit was a 2-board device, and each board had an associated hardware revision level.

When called, the service will (at the programmer's option) immediately display a message indicating the actual revision level and/or messages indicating whether the actual revision level matches the revision level expected by the diagnostic program. One of the following messages may be printed. (DEVNAME will be replaced by the generic name of the unit in question.)

```
Hardware revision level of DEVNAME (T1002):    ******
Hardware revision level of DEVNAME (T1002) is less than that expected
by diagnostic program.
       Hardware revision level:                ******
       Revision level expected by program:     ******

Hardware revision level of DEVNAME (T1002) is equal to that expected
by diagnostic program.
       Hardware revision level:                ******
       Revision level expected by program:     ******

Hardware revision level of DEVNAME (T1002) is greater than that
expected by diagnostic program.
       Hardware revision level:                ******
       Revision level expected by program:     ******
```

If firmware or patch levels are being reported, the word "hardware" in the messages will be replaced with the word "firmware" or "patch".

If the modulenum argument has been passed, the module number will be printed for the hardware, firmware, and patch messages associated with that module number. If the number is not passed, it will not appear in the above messages.

Each time an error service is called ($DS_ERRxxxx) for a unit for which the $DS_PRINTREV macro has been called, the hardware revision level, and/or firmware revision level and/or patch level will be displayed as part of the error message. The message will be displayed AFTER all of the error text has been printed. It will be inhibited if the IE1 flag is set. The format of this display will be:

```
******  program title          program rev. ******
Pass #, test #, subtest #, error #, date
Hard error while testing DEVNAME:  text

       text

Revision level(s) for DEVNAME:
       (T1002): Hardware = X; Firmware = X; Patch = X;
       (T1003): Hardware = Y; Firmware = Y; Patch = Y;

****** End of hard error number # ******
```

(If any of the levels or the module number are not being reported, they will not appear in the revision level message.)

MACRO-32 Format:

```
$DS_PRINTREV_x log_unit, actual_rev, expected_rev, rev_type,
                        [printmask], [prlink], [modulenum]
```

BLISS Format:

```
$DS_PRINTREV   (LOG_UNIT=log_unit, ACTUAL_UNIT=actual_rev,
                        EXPECTED_REV=expected_rev, REV_TYPE=rev_type,
                        [PRINTMASK=printmask], [PRLINK=prlink],
                        [MODULENUM=modulenum]);
```

log_unit

Logical unit number of device whose revision level is to be checked.

actual_rev

Longword containing the actual hardware, firmware, or patch revision level for the device whose logical unit number is specified with the "logunit" parameter. See note 4. (The diagnostic program determines this value by accessing the hardware.)

expected_rev

Longword containing the hardware, firmware, or patch revision level which was expected by the diagnostic program. See note 4. (This value is placed in the diagnostic program at compilation time.)

rev_type

Longword mask indicating the type of revision code (hardware, firmware, or patch) being compared. Bit definitions are:

- Bit 0 - PRVV_HWREV,PRVM_HWREV - Set this bit to indicate a hardware revision level.
- Bit 1 - PRVV_FWREV,PRVM_FWREV - Set this bit to indicate a firmware revision level.
- Bit 2 - PRVV_PREV,PRVM_PREV - Set this bit to indicate a patch level.

(Only one bit should be set per service call.)

printmask

Longword mask used to control $DS_PRINTREV functions. Bit definitions are:

- Bit 0 - PRMSK$V_LSS,PRMSK$M_LSS - if set, inhibits message that would normally be displayed if the actual revision is less than expected.
- Bit 1 - PRMSK$V_EQL,PRMSK$M_EQL - if set, inhibits message that would normally be displayed if the actual revision is equal to that expected.
- Bit 2 - PRMSK$V_GTR,PRMSK$M_GTR - if set, inhibits message that would normally be displayed if the actual revision is greater than expected.
- Bit 3 - PRMSK$V_ALWAYS,PRMSK$M_ALWAYS - if set, displays the message which lists only the actual revision level.
- Bit 4 - PRMSK$V_TRANSL,PRMSK$M_TRANSL - if set, will convert a hardware revision number to a letter code which represents the functional revision as defined by DEC STD 012. Setting this bit will not result in conversion for firmware or patch revisions. The conversion pattern is shown in table 1. See note 6.

Table 1: Revision Number to Letter Conversion

Field contents [bits <n:0>]	Functional Rev.
0.....0000	pre-release
0.....0001	A
0.....0010	B
0.....0011	C
.	.
.	.
0....11010	Z
0....11011	AA
0....11100	AB
0....11101	AC
.	.
.	.
0...110100	AZ
0...110101	BA
.	.
.	.
0..1001110	BZ
.	.
.	.
1010111110	ZZ

For example, if only bits 1 and 2 are set then a message will be displayed only if the actual revision is less than that expected by the program.

By default, bits 0, 1, 2, 3, and 4 are clear, in which case no conversion will be performed on revision numbers. In addition, a message will be displayed stating whether the actual revision is greater than, less than, or equal to the revision expected by the program. Inhibiting messages does not affect the inclusion of the hardware, firmware or patch revision levels in error messages generated by $DS_ERRxxxx macros.

prlink

Address of a message routine. This optional routine will have the same format as an error reporting routine used with a $DS_ERRxxxx macro. That is, the routine should be delineated with $DS_BGNMESSAGE and $DS_ENDMESSAGE macros and should use $DS_PRINTB and $DS_PRINTX macros for displaying text.

modulenum

Address of a counted ASCII string which represents the unique module number for a board (e.g. "T1002"). If you supply this argument for a hardware revision, then you must supply the same argument for any firmware or patch revision associated with that module.

Notes:

1. Symbols are defined by the $DS_PRINTREV_DEF macro.

2. This macro should only be used for devices having software-accessible hardware, firmware, or patch revision numbers.

3. Some programmer's may want the text displayed only in certain circumstances. For example, perhaps it is the concensus of a particular program's users that the text only be displayed if the actual revision level is lower than that expected by the diagnostic program. In such a case, the "printmask" parameter may be used to inhibit one or more of the possible warning messages.

4. The format of the revision level is irrelevant, but the values passed via the "actual_rev" and "expected_rev" parameters must be in the same format. The service will perform an unsigned comparison of the two values to determine which message to display. Revision numbers will be displayed as unsigned decimal values unless conversion to a letter code is desired.

5. This macro should be used to determine whether current versions of diagnostics are incompatible with old hardware. Since new diagnostics may only run on hardware above a certain revision level, the diagnostic should only set bits 1 and 2 in printmask. This causes a message to be displayed only if the actual revision level is less than that expected by the program.

6. Normally, the hardware revision level should be converted to the corresponding functional revision letter code by setting bit 4 in printmask.

7. If you do not set bit 2 in printmask, a misleading message may result if the actual hardware, firmware, or patch levels were updated without a corresponding change to the diagnostic's expected revision levels.

3. Known Bugs and Deficiencies: The console command language uses the device names CSA1 and CSA2 for the console floppy drives. The Supervisor,however,always refers to either drive as CSA0.

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 9.2
March 25,1986

1. EZSAA-9.2-292

   - o The p-table for the KA850 has been added .

   - o This version has support code for attaching the floppies, CSA1,CSA2 and the winchester RD disk,CSA3. Upon boot of the supervisor the default disk will be CSA3,the winchester RD disk. I/O can be done from any of the afor-mentioned devices after doing a SET LOAD to that particular device.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 9.3 April 4,1986

1. EZSAA-9.3-305

   o This version has support for the DMB32

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 10.0
April 30, 1986

1 ENHANCEMENTS

This version, 10.0, contains only the 8200/8300 supervisor. Support includes booting the supervisor on the attached processor, running of multiprocessor diagnostics, and new BI adapters.

1.1 New BI Adapters

1. DWBLA - This adapter is a combination of a DWBUA and LESI controller. TU81 tape drives and RC25/RC25F disk drives are connected to it. Note that the TU81 is attached directly to the DWBLA, while the RC25 and RCF25 must be attached to a LESI which in turn must be attached to the DWBLA. They are attached as follows:

```
DS>
DS> ATTACH DWBLA HUB BLA0 ! BLA adapter
BI Node Number (HEX) ? 6
DS>
DS> ATTACH TU81 BLA0 MUA0 ! TU81 tape drive
CSR? 774500
VECTOR? 260
BR? 5
DS>
DS> ATTACH LESI BLA0 DAA  ! LESI controller (part of the DWBLA)
IP? 774500
Vector? 260
BR? 5
DS>
DS> ATTACH RC25 DAA DAA0  ! AZTEC removable disk drive
DS> ATTACH RCF25 DAA DAA1 ! AZTEC fixed head disk drive
DS>
```

2. DMB32 - This is a synchronous/asynchronous multiplexer. It is attached as follows:

```
DS> ATTACH DMB32 HUB TXA  ! DMB32 adapter
BI Node Number (HEX)? 3
DS>
```

1.2 New Features For The 8300 Systems.

- Attached processor (AP) support. Allows running the VDS on the AP. A new command, BOOT node (where node = node number of AP), allows the Supervisor to execute on the AP. The primary processor (PP) continues to run in order to provide console I/O (both terminal and floppy), but it does not execute any diagnostics. The Supervisor operates on the AP as it would on the PP. To return control to the PP, use the EXIT command. WARNING: Before any transition between the AP and the PP, ensure that the diagnostic has completed or has been ABORTed.

  EXAMPLE of running the VDS on an AP (node F):

```
>>>b csa1

0 . . 3 4 . . 7 . 8 9 A . . . . F
00600000

                    VAX DIAGNOSTIC SOFTWARE
                          PROPERTY OF
                    DIGITAL EQUIPMENT CORPORATION

                    ***CONFIDENTIAL AND PROPRIETARY***


Use Authorized Only Pursuant to a Valid Right-to-Use License
Copyright, Digital Equipment Corporation, 1986. All Rights
Reserved.


DIAGNOSTIC SUPERVISOR. ZZ-EBSAA-10.0-618 28-APR-1986 17:09:36

DS> !VDS is running on the Primary Processor (PP).
DS> !We now issue the BOOT command to start the Attached Processor
    !(AP).
DS> BOOT F
S 0001F420

                    VAX DIAGNOSTIC SOFTWARE
                          PROPERTY OF
                    DIGITAL EQUIPMENT CORPORATION

                    ***CONFIDENTIAL AND PROPRIETARY***


Use Authorized Only Pursuant to a Valid Right-to-Use License
Copyright, Digital Equipment Corporation, 1986. All Rights
Reserved.

DS> !The AP is now running.
DS> !ATTACH and SELECT the AP.
```

```
DS> ATTACH KA820 HUB KAF 6144 F
DS> SELECT KAF
DS> !Run a diagnostic.
DS> RUN EVKAB
.. Program: ZZ-EVKAB, VAX Basic Inst's Exrczr, revision 3.4, 161
tests, at 17:19:08.68.

Testing:
_KAF

.. End of run, 0 errors detected, pass count is 1,
   time is 28-APR-1986 17:19:57.13

DS> !Diagnostic is finished.
DS> !Stop the AP.
DS> EXIT

 *** AP has stopped ***

DIAGNOSTIC SUPERVISOR.  ZZ-EBSAA-10.0-618  28-APR-1986 17:09:36

DS> !The VDS is running on the PP again.
DS>
```

o VAX Diagnostic Supervisor support for multi-processor diagnostics.

* Multiprocessor Concepts:

  A few diagnostic programs are designed to run concurrently on multiple processors. In a multiprocessing environment, you boot the VDS in one of the processors. The processor running the VDS is called the primary processor. If you load and start a multiprocessing diagnostic program, it will cause one or more of the other processors to execute code simultaneously with the primary processor. These other processors are called attached processors. New services have been added to the VAX Diagnostic Supervisor to support multi-processor diagnostics.

* VDS in a Multiprocessor Environment:

  This section describes the VDS features that facilitate the execution of diagnostic programs on multiprocessor systems.

  General Concepts

  In a multiprocessor environment, the processor used to boot the VDS is called the "primary processor". All other processors are called "attached processors". Labeling processors as primary and attached is just a software definition and does not imply any particular hardware configuration. "Attached" processors bear no

relation to the VDS ATTACH command.

When the VDS is booted, it always assumes there is only one processor.

When operating on a multiprocessor system, there is one copy of the VDS software. Control portions of the VDS, such as the command line interpreter and the command dispatcher, are executed only by the primary processor. Some portions of the code, such as system services and exception handlers, may be executed by any processor.

It is assumed that a common memory is shared by all processors.

Diagnostic programs are loaded by and initially executed by the primary processor. A diagnostic program may consist of one or more secondary portions that can be executed by one or more attached processors. In this document, the code executing in the primary processor is referred to as the "primary process". Code executing in an attached processor is called an "attached process".

* Attaching and Selecting Attached Processors:

If a diagnostic program is to use more than one processor, ATTACH commands must be used to describe each processor. You specify which processors to test by ATTACHing and SELECTing them. The diagnostic program calls the $DS_GPHARD service to find out which processors to use.

* Multiprocessing Macros:

The VDS provides the following system services for use in multiprocessor environments:

- o $DS_BOOTATTACHED boots an attached processor and leaves the processor in an execution loop called an "idle loop", which is a "JMP" instruction.

- o $DS_BGNATTACHED and $DS_ENDATTACHED delineate the code to be executed in an attached processor.

- o $DS_STARTATTACHED causes an attached processor to begin executing at a specified address. The code to be executed must be delimited by the $DS_BGNATTACHED and $_ENDATTACHED macros. When execution of this code is complete, the processor is automatically returned to its idle loop.

- o $DS_HALTATTACHED halts an attached processor.

- $DS_SHOWIDLE indicates which attached processors are currently executing "idle loops".

- A new value for the $DS_EXIT argument, ATTACHED, lets you exit from the currently attached process. You must locate the $DS_EXIT macro between the $DS_BGNATTACHED and $DS_ENDATTACHED macros.

* Executing in an Attached Processor:

In order for a diagnostic program to execute an attached process, the following steps must occur:

- The attached processor must be booted using the $DS_BOOTATTACHED service.

- The code to be executed must be loaded either by including the code within the source file of the code executing in the primary or by including it in a separate file. If you use a separate file, you must:

  a Call the $DS_GETBUF service to allocate memory space for the attached process.

  a Use the $DS_LOAD service or RMS services to load the file into the space assigned by the $DS_GETBUF service.

- The $DS_STARTATTACHED service passes the attached processor the address where the code begins. In the case where the attached process is loaded into a buffer, the address is the same as the buffer itself.

- The attached process begin executing; VDS system services may be called.

You must repeat the process for each attached processor.

* Using System Services;

Most system services may be called from code executing in attached processors. These services provide an interlocking mechanism, transparent to the diagnostic program, that assures that only one processor executes the service routine at a time. If two or more processors often issue calls to the same service, a large amount of system time may be spent waiting for one processor to finish with the service so that the other one can use it. If two or more processors are locked out of a service because it is in use, the next processor to be serviced is completely arbitrary, as there is no enqueueing of service requests.

The following services may not be called from code

executing in an attached processor:

$DS_ASKxxxx

$DS_CHANNEL

$DS_CLRVEC

$DS_ERRxxx

$DS_INITSCB

$DS_LOAD

$DS_MMON, $DS_MMOFF

$DS_PARSE

$DS_PRINTx

$DS_SETMAP

$SETIMR

$DS_SETVEC

$DS_WAITMS

Additionally, the $DS_MMON and $DS_MMOFF may not be called from code executing in the primary processor after an attached processor has been booted (with the $DS_BOOTATTACHED service).

* Memory Management:

Memory mapping for all processors is identical. Any code within the VDS that alters the page tables alters the tables for each processor because there is one set of page tables for all processors. Page table base registers for each processor simply point to the same places.

However, consider the following scenario:

- You run a diagnostic program that creates attached processes.
- Execution of the diagnostic program is stopped by control-C, a breakpoint, or an exception condition.
- You issue either a SET MM ON or SET MM OFF command and then issue the CONTINUE command.

In this scenario, the state of memory management in the primary processor changes when you issue the SET MM

command. The state of memory management in the attached processors, however, does not change until you issue the CONTINUE command.

If each processor executes $DS_GETBUF macros, $DS_RELBUF cannot be used to separately release buffers allocated to each processor because $DS_RELBUF deallocates the last allocated blocks no matter who requested them. All $DS_GETBUF and $DS_RELBUF calls should be made by the primary process. Alternately, you can use a globally-referenced location to track total buffer allocation. Then you can issue one $DS_RELBUF call to deallocate all allocated space at once.

* Timing:

The only timing service available is the $DS_WAITUS. Any other timing scheme must be designed by the user.

It is important to note that only one $DS_WAITUS request may be serviced at any time. If one processor, for example, requests a $DS_WAITUS service and the $DS_WAITUS service routine is already in use by another processor, the requesting processor is forced (by the VDS) to wait until the service routine has completed its execution for the first processor.

The result will be that the actual amount of time the second processor has to wait could be considerably longer than the actual wait time requested. Be aware that if more than one processor is waiting for service, it is arbitrary as to which is serviced first, as there is no enqueueing of requests.

* The SCB:

Each processor has its own SCB that is initialized when the processor is bootstrapped. All SCBs are initialized as follows:

- All vectors in the first half page of the SCB point to the proper exception handlers. Note that all handlers are the same for each processor.

- The rest of the SCB (the device interrupt vector area) points to the VDS's unexpected interrupt handler. Only the primary processor, however, receives device interrupts (See the section entitled, "Input/Output").

The $DS_SETVEC, $DS_CLRVEC, and $DS_INITSCB services can only be called by the primary process. If an attached process wants to modify its SCB, it must do so directly. The SCB base address is returned by the $DS_BOOTATTACHED service.

* Input/Output:

  Only the primary processor may receive device interrupts. (For some processor types, this is a hardware restriction, for others, it is not. For consistency's sake, VDS implementation is the same for all processor types.)

  The $DS_CHANNEL service may be called only by the primary processor.

  Device interrupt service routines only execute in the primary processor and are considered a part of the main program. The main program may notify an attached process that an interrupt has been received by setting an event flag or by delivering an interprocessor interrupt to the attached processor.

  It is important to remember that the $DS_CHANNEL service only allows one device interrupt service routine in use at a time. Therefore if several devices are to be active at once, they must all be serviced by the same interrupt service routine. (The routine can determine which device caused the most recent interrupt, since the vector address is passed to the routine.)

* ASTs:

  Only the primary processor can execute a service that provides AST delivery. (These services, for level 3 programs, include $SETIMR, $DS_CNTRLC, and, indirectly, $DS_WAITMS.) AST requests from all processors are enqueued into a common AST queue.

* Interprocessor Interrupts:

  Diagnostic programs may implement interprocessor communication by using interprocessor interrupts. Diagnostic programs wishing to make use of interprocessor interrupts may modify the SCBs of the various processors so that the IP interrupt vector points to an interrupt service routine specified by the program.

  The VDS does not use interprocessor interrupts.

* Exceptions and Unexpected Interrupts:

  The SCB of each processor is initialized so that exceptions vector into VDS condition handlers that provide interlocking. The last chance handler stops program execution on all processors, no matter which processor "trapped out". The primary processor re-enters VDS CLI and issues the DS> prompt. All attached processors re-enter their idle loops.

Diagnostic programs can override the VDS last chance handler by specifying their own condition handlers.

Unexpected device interrupts are fielded by the primary processor. The primary processor's unexpected interrupt handler reports the interrupt to the user and re-enters VDS CLI, issuing the DS> prompt. All attached processors are forced to re-enter their idle loops.

* Control-Cs:

You can return to the VDS CLI and the DS> prompt by typing control-C. Attached processors return to their idle loops the next time they call the $DS_BREAK service.

As is currently the case, a diagnostic program may declare its own control-C handler to take precedence over the VDS control-C handler.

* Communication Between the Main and Attached Processes:

Since the VDS does not provide specific services to facilitate communication between the attached processes and the primary process, you can use any of the following techniques:

- o Event Flags - Very useful for passing status information between the primary and various attached processes. The $SETEF, $CLREF, $READEF, $WAITFR, $WFLAND, and $WFLOR services may all be used by any process.

- o Interprocessor Interrupts - Available for use by the diagnostic program; you may design any scheme you wish for making use of interprocessor interrupts.

- o Common "mailbox" - You can specify a common data area in which to pass information back and forth between the main process and the attached processes.

- o Dispatch vectors - Attached processes can call routines in the main process via dispatch vectors, stored in a table in the main process, that point to the routines. If these vectors are assigned to absolute addresses, then attached processes not linked with the main process can reference them. (If the code for attached processes is linked with the main process, dispatch vectors are unnecessary, since addressing references may be relative and can be resolved at link time.)

* Restrictions:

The following restrictions apply to diagnostic programs using the multiprocessing features of the VDS:

- As with single processor systems, code executing in attached processors must periodically call the $DS_BREAK service. THIS RULE IS VERY IMPORTANT, as breakpoints, control-C's, and exception handling depend upon this rule being followed.

- With the exception of $DS_BGNATTACHED and $DS_ENDATTACHED, code executing in attached processors may not use any of the program structure macros. (These macros include $DS_BGNTEST, $DS_ENDTEST, $DS_BGNSUB, $DS_ENDSUB, and the macros $DS_HEADER and $DS_DISPATCH that define such data structures as the diagnostic header and the dispatch table, respectively.) All initialization and clean-up code, looping, and error reporting must be contained within code executed by the primary processor.

- The load image for a diagnostic program may not be larger than approximately 63.5 Kbytes. If the total size of the code to be executed by the primary and all attached processes exceeds the maximum, you have to store the code for attached processors in separate loadable images. (Refer to the section entitled "Executing in an Attached Processor".)

- As stated previously, requests for system services are not enqueued (except in the case of ASTs). Therefore if several attached processes are simultaneously requesting the same service, there is no way to determine which process will be serviced next. It is assumed (but not guaranteed by the software) that all requests will eventually be serviced. All services have an interlocking mechanism, so that the next processor to execute the service is the first one to reference the interlocking flag after the service is released (by the previous processor.)

- Only the primary processor can receive device interrupts.

- Use the standard methods for declaring condition handlers.

- All multi-processor diagnostics must run in Kernel mode.

- It is recommended that the diagnostic clean-up code call the $DS_HALTATTACHED service in order that each Attached Processor that was bootattached be placed in a known, static state.

- Code executing in an attached processor may not call the following services:

  $DS_ASKxxxx
  $DS_CHANNEL
  $DS_CLRVEC
  $DS_ERRxxx
  $DS_INITSCB
  $DS_LOAD
  $DS_MMON, $DS_MMOFF
  $DS_PARSE
  $DS_PRINTx
  $DS_SETVEC
  $SETIMR
  $DS_SETMAP
  $DS_WAITMS

* Multi-processor Macros:

- $DS_BGNATTACHED, $DS_ENDATTACHED

  In a diagnostic program that tests multiple processors, use the $DS_BGNATTACHED and $DS_ENDATTACHED macros to delineate code that is to be executed in an attached processor. These macros are used whether the code is included in the loadable image of the main diagnostic program or it is a separate loadable image.

  $DS_BGNATTACHED indicates the beginning of the code and creates a label that can be used with the $DS_STARTATTACHED service. The $DS_ENDATTACHED macro generates code that will send the processor back to its idle loop.

  MACRO-32 Format:

```
$DS_BGNATTACHED routine_name, mask
        .
        .
        .
$DS_ENDATTACHED
```

  BLISS-32 Format:

```
$DS_BGNATTACHED (ROUTINE_NAME=routine_name);
        .
        .
        .
$DS_ENDATTACHED;
```

  routine_name

  Labels the routine and points to its first instruction.

mask
List of register names used in the entry mask.

Notes:

1. You can include code that is contained in an attached process in any number of separate executable files. The code in each file, however, must be position-independent. You can only have one attached process, delimited by one set of $DS_BGNATTACHED and $DS_ENDATTACHED macros, per file.

2. If you want to place the code in a separate image, request a buffer using the $DS_GETBUF service, load the image into the buffer, and use the address of the buffer as the "start_addr" argument for the $DS_STARTATTACHED macro.

3. You can enter the code using a CALL instruction.

4. It is recommended that you place data structures for the code in a separate psect. If you must include the data structures

   in the same psect as the code, place them (data structures) after the code and end the executable section with a $DS_EXIT macro as shown:

```
.psect data                 $DS_BGNATTACHED RTN2
                                        .
<data structures>                       .
        .                               .
        .                   <executable code>
        .                               .
.psect code                             .
$DS_BGNATTACHED RTN1        $DS_EXIT ATTACHED
        .                               .
        .                               .
<executable code>           <data structures>
        .                               .
        .                               .
        .
$DS_ENDATTACHED             $DS_ENDATTACHED
```

o $DS_BOOTATTACHED

In a multiprocessing environment, use the Boot Attached CPU system service to bootstrap an attached processor on a multiprocessor system. It will perform the following functions for the target processor:

1. Halt the processor, if it is not currently halted.

2. Perform all initialization necessary to enable the processor to execute code, including initializing stacks, memory management registers, the SCB, and the interval clock.

3. Cause the processor to begin executing an idle loop ("JMP ." instruction).

After you call the $DS_BOOTATTACHED, you can use the $DS_STARTATTACHED service to cause the attached processor to leave its idle loop and begin executing a section of code.

MACRO-32 Format:

```
$DS_BOOTATTACHED_x unit, scb_addr
```

BLISS-32 Format:

```
$DS_BOOTATTACHED (UNIT=unit, SCB_ADDR=scb_addr);
```

unit

Logical unit number of the processor to be bootstrapped.

scb_addr

Address of the longword to receive the SCB address of the target processor.

Return Status:

DS$_NORMAL

Service successfully completed.

DS$_ILLUNIT

The specified logical unit number is too large.

DS$_INVCPU

Can't boot specified processor.

DS$_MEM_ALLOC_ERR

Could not allocate memory for attached processor's SCB and stacks.

DS$_INIT_FAIL
Attached processor did not return prompt from Control P or Initialize. (VAX 8200 only)

Examples:

MACRO-32 Example:

```
$DS_BOOTATTACHED_S      LOG_UNIT, PROC2_SCB
```

BLISS-32 Example:

```
$DS_BOOTATTACHED (UNIT = .LOG_UNIT, SCB_ADDR =
                  PROC2_SCB);
```

o $DS_HALTATTACHED

Use the Halt Attached CPU system service to stop program execution in an attached processor or a multiprocessor environment. In order to use this service, you must bootstrap the processor with the $DS_BOOTATTACHED service. This service should be used in the clean-up code of a multi-processor diagnostic to place each Attached Processor that was bootattached into a known, static state.

MACRO-32 Format:

```
$DS__HALTATTACHED__x unit
```

BLISS-32 Format:

```
$DS__HALTATTACHED (UNIT=unit);
```

unit

Logical unit number of the CPU to be halted.

Return Status:

DS$_NORMAL

Service successfully completed.

DS$_ILLUNIT

The specified logical unit number is too large.

DS$_INVCPU

The specified processor is the primary processor.

DS$_INIT_FAIL

The processor failed to send console prompt only).

Notes:

1. In order to restart a halted processor, you must reboot using the $DS_BOOTATTACHED service and then use the $DS_STARTATTACHED service.

Examples:

MACRO-32 Example:

```
$DS__HALTATTACHED__S    LOG_UNIT
```

BLISS-32 Example:

```
$DS__HALTATTACHED (UNIT = .LOG_UNIT);
```

o $DS_SHOWIDLE

In a multiprocessor environment, use the Show Idle Processors service to determine which attached processors are currently executing their "idle loops". Attached processors are placed in their idle loops when:

1. The $DS_BOOTATTACHED service has completed bootstrapping the processor.

2. An attached process, delimited by the $DS_BGNATTACHED and $DS_ENDATTACHED macros, finishes executing.

3. A multiprocessor diagnostic program is stopped by a control-C or an exception.

MACRO-32 Format:

```
$DS_SHOWIDLE_x  bitmap
```

BLISS-32 Format:

```
$DS_SHOWIDLE (BITMAP=bitmap);
```

bitmap

Address of the longword to receive a bitmap indicating which attached processors are currently executing their idle loops. There are 32 bits (0 through 31); each bit corresponding to a logical unit number. You need to look at the bits that correspond

to logical unit numbers actually associated with attached processors. (See Note 1.)

Return Status:

DS$_NORMAL

Service successfully completed.

Notes:

1. One method for using this service is to create a bit mask. Each time you issue a $DS_GPHARD call and receive the address of a p-table for an attached processor, set the bit in the mask corresponding to the logical unit number for that p-table. When you call the $DS_SHOWIDLE service, test only the bits that are set in the mask you created.

Examples:

MACRO-32 Example:

```
IDLE_MASK:      .LONG 0
IDLE_PROC:      .LONG 0

                $DS_SHOWIDLE_S (IDLE_PROC)  ; Get idling
                                            ; processors.
                CMPL    IDLE_MASK,IDLE_PROC ; All idling?
                BEQL    300$                ; Yes, branch.
                        .
                        .
```

BLISS-32 Example:

```
                        .
                        .
        $DS_SHOWIDLE (BITMAP=IDLE_PROC); ! Get idling
                                         ! processors.
        IF (.IDLE_MASK NEQ .IDLE_PROC)   ! If not idling
        THEN                             ! then...
                        .
                        .
```

o $DS_STARTATTACHED

In a multiprocessor environment, you can use the Start Attached CPU system service to cause an attached processor to begin executing a section of code. Before you can use this service, you must bootstrap the specified processor with the $DS_BOOTATTACHED service.

You must delimit the section of code to be executed with the $DS_BGNATTACHED and $DS_ENDATTACHED program structure macros. The service, in conjunction with these macros, causes the specified processor to leave the idle loop that was created by the $DS_BOOTATTACHED service and jump to the code delimited by the macros. When it finishes executing, the processor re-enters the idle loop. (Refer to the description of the $DS_BGNATTACHED and $DS_ENDATTACHED macros for details.)

MACRO-32 Format:

```
$DS_STARTATTACHED_x unit, start_addr
```

BLISS-32 Format:

```
$DS_STARTATTACHED (UNIT=unit, START_ADDR=start_addr);
```

unit

Logical unit number of the processor to be bootstrapped.

start_addr

Address of the code to be executed in the attached processor. This argument must be the address of the code generated by a $DS_BGNATTACHED macro.

Return Status:

DS$_NORMAL

Service successfully completed.

DS$_ILLUNIT

The specified logical unit number is too large.

DS$_INVCPU

Can't start the specified processor. May mean the processor doesn't exist or the processor was not executing its idle loop (see $DS_BOOTATTACHED).

Examples:

MACRO-32 Example:

```
.
.
.
```

```
            $DS_BGNATTACHED ROUTINE1
                    .
                    .
                    .
            $DS_ENDATTACHED
                    .
                    .
                    .
            $DS_STARTATTACHED_S     LOG_UNIT, ROUTINE1
```

BLISS-32 Example:

```
                    .
                    .
                    .
            $DS_BGNATTACHED (ROUTINE_NAME=ROUTINE1);
                    .
                    .
                    .
            $DS_ENDATTACHED;
                    .
                    .
                    .
            $DS_STARTATTACHED (UNIT = .LOG_UNIT,
                               START_ADDR=ROUTINE1);
```

2 KNOWN BUGS AND DEFICIENCIES

1. Version 9.0 has a bug whereby depositing into an Internal Processor Register at the Diagnostic Supervisor prompt will yield incorrect results. Version 10.0 fixes this problem.

2. The bug in version 9.0 which prevented the diagnostic supervisor from being booted from a CIBCI disk has been fixed. Previously a machine check would result.

3. In version 9.0, the supervisor was inadvertently clearing the BROKE bit for memory adapters at boot time, if the adapter had previously set the BROKE bit. This has been fixed.

4. In version 9.0, after an ULTRIX system had previously been running, an "Unexpected BI bus error interrupt" would occur while running EBUCA. This has been fixed.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 10.1 April 4, 1986

1 ENHANCEMENTS

This version, 10.1, contains only the 8500/8700/8800 supervisor. Support includes running of multiprocessor diagnostics.

1.1 Multiprocessing

- VAX Diagnostic Supervisor support for multi-processor diagnostics.

  * Multiprocessor Concepts:

    A few diagnostic programs are designed to run concurrently on multiple processors. In a multiprocessing environment, you boot the VDS in one of the processors. The processor running the VDS is called the primary processor. If you load and start a multiprocessing diagnostic program, it will cause one or more of the other processors to execute code simultaneously with the primary processor. These other processors are called attached processors. New services have been added to the VAX Diagnostic Supervisor to support multi-processor diagnostics.

  * VDS in a Multiprocessor Environment:

    This section describes the VDS features that facilitate the execution of diagnostic programs on multiprocessor systems.

    General Concepts

    In a multiprocessor environment, the processor used to boot the VDS is called the "primary processor". All other processors are called "attached processors". Labeling processors as primary and attached is just a software definition and does not imply any particular hardware configuration. "Attached" processors bear no relation to the VDS ATTACH command.

    When the VDS is booted, it always assumes there is only one processor.

When operating on a multiprocessor system, there is one copy of the VDS software. Control portions of the VDS, such as the command line interpreter and the command dispatcher, are executed only by the primary processor. Some portions of the code, such as system services and exception handlers, may be executed by any processor.

It is assumed that a common memory is shared by all processors.

Diagnostic programs are loaded by and initially executed by the primary processor. A diagnostic program may consist of one or more secondary portions that can be executed by one or more attached processors. In this document, the code executing in the primary processor is referred to as the "primary process". Code executing in an attached processor is called an "attached process".

* Attaching and Selecting Attached Processors:

  If a diagnostic program is to use more than one processor, ATTACH commands must be used to describe each processor. You specify which processors to test by ATTACHing and SELECTing them. The diagnostic program calls the $DS_GPHARD service to find out which processors to use.

* Multiprocessing Macros:

  The VDS provides the following system services for use in multiprocessor environments:

  - $DS_BOOTATTACHED boots an attached processor and leaves the processor in an execution loop called an "idle loop", which is a "JMP" instruction.

  - $DS_BGNATTACHED and $DS_ENDATTACHED delineate the code to be executed in an attached processor.

  - $DS_STARTATTACHED causes an attached processor to begin executing at a specified address. The code to be executed must be delimited by the $DS_BGNATTACHED and $_ENDATTACHED macros. When execution of this code is complete, the processor is automatically returned to its idle loop.

  - $DS_HALTATTACHED halts an attached processor.

  - $DS_SHOWIDLE indicates which attached processors are currently executing "idle loops".

  - A new value for the $DS_EXIT argument, ATTACHED, lets you exit from the currently attached process. You must locate the $DS_EXIT macro between the $DS_BGNATTACHED and $DS_ENDATTACHED macros.

* Executing in an Attached Processor:

  In order for a diagnostic program to execute an attached process, the following steps must occur:

  - The attached processor must be booted using the $DS_BOOTATTACHED service.
  - The code to be executed must be loaded either by including the code within the source file of the code executing in the primary or by including it in a separate file. If you use a separate file, you must:
    - Call the $DS_GETBUF service to allocate memory space for the attached process.
    - Use the $DS_LOAD service or RMS services to load the file into the space assigned by the $DS_GETBUF service.
  - The $DS_STARTATTACHED service passes the attached processor the address where the code begins. In the case where the attached process is loaded into a buffer, the address is the same as the buffer itself.
  - The attached process begin executing; VDS system services may be called.

  You must repeat the process for each attached processor.

* Using System Services;

  Most system services may be called from code executing in attached processors. These services provide an interlocking mechanism, transparent to the diagnostic program, that assures that only one processor executes the service routine at a time. If two or more processors often issue calls to the same service, a large amount of system time may be spent waiting for one processor to finish with the service so that the other one can use it. If two or more processors are locked out of a service because it is in use, the next processor to be serviced is completely arbitrary, as there is no enqueueing of service requests.

  The following services may not be called from code executing in an attached processor:

  $DS_ASKxxxx

  $DS_CHANNEL

$DS_CLRVEC

$DS_ERRxxx

$DS_INITSCB

$DS_LOAD

$DS_MMON, $DS_MMOFF

$DS_PARSE

$DS_PRINTx

$DS_SETMAP

$SETIMR

$DS_SETVEC

$DS_WAITMS

Additionally, the $DS_MMON and $DS_MMOFF may not be called from code executing in the primary processor after an attached processor has been booted (with the $DS_BOOTATTACHED service).

* Memory Management:

Memory mapping for all processors is identical. Any code within the VDS that alters the page tables alters the tables for each processor because there is one set of page tables for all processors. Page table base registers for each processor simply point to the same places.

However, consider the following scenario:

- You run a diagnostic program that creates attached processes.
- Execution of the diagnostic program is stopped by control-C, a breakpoint, or an exception condition.
- You issue either a SET MM ON or SET MM OFF command and then issue the CONTINUE command.

In this scenario, the state of memory management in the primary processor changes when you issue the SET MM command. The state of memory management in the attached processors, however, does not change until you issue the CONTINUE command.

If each processor executes $DS_GETBUF macros, $DS_RELBUF cannot be used to separately release buffers allocated to

each processor because $DS_RELBUF deallocates the last allocated blocks no matter who requested them. All $DS_GETBUF and $DS_RELBUF calls should be made by the primary process. Alternately, you can use a globally-referenced location to track total buffer allocation. Then you can issue one $DS_RELBUF call to deallocate all allocated space at once.

* Timing:

  The only timing service available to an Attached Processor is the $DS_WAITUS. Any other timing scheme must be designed by the user.

  It is important to note that only one $DS_WAITUS request may be serviced at any time. If one processor, for example, requests a $DS_WAITUS service and the $DS_WAITUS service routine is already in use by another processor, the requesting processor is forced (by the VDS) to wait until the service routine has completed its execution for the first processor.

  The result will be that the actual amount of time the second processor has to wait could be considerably longer than the actual wait time requested. Be aware that if more than one processor is waiting for service, it is arbitrary as to which is serviced first, as there is no enqueueing of requests.

* The SCB:

  Each processor has its own SCB that is initialized when the processor is bootstrapped. All SCBs are initialized as follows:

  o All vectors in the first half page of the SCB point to the proper exception handlers. Note that all handlers are the same for each processor.

  o The rest of the SCB (the device interrupt vector area) points to the VDS's unexpected interrupt handler. Only the primary processor, however, receives device interrupts (See the section entitled, "Input/Output").

  The $DS_SETVEC, $DS_CLRVEC, and $DS_INITSCB services can only be called by the primary process. If an attached process wants to modify its SCB, it must do so directly. The SCB base address is returned by the $DS_BOOTATTACHED service.

* Input/Output:

Only the primary processor may receive device interrupts. (For some processor types, this is a hardware restriction, for others, it is not. For consistency's sake, VDS implementation is the same for all processor types.)

The $DS_CHANNEL service may be called only by the primary processor.

Device interrupt service routines only execute in the primary processor and are considered a part of the main program. The main program may notify an attached process that an interrupt has been received by setting an event flag or by delivering an interprocessor interrupt to the attached processor.

It is important to remember that the $DS_CHANNEL service only allows one device interrupt service routine in use at a time. Therefore if several devices are to be active at once, they must all be serviced by the same interrupt service routine. (The routine can determine which device caused the most recent interrupt, since the vector address is passed to the routine.)

* ASTs:

Only the primary processor can execute a service that provides AST delivery. (These services, for level 3 programs, include $SETIMR, $DS_CNTRLC, and, indirectly, $DS_WAITMS.) AST requests from all processors are enqueued into a common AST queue.

* Interprocessor Interrupts:

Diagnostic programs may implement interprocessor communication by using interprocessor interrupts. Diagnostic programs wishing to make use of interprocessor interrupts may modify the SCBs of the various processors so that the IP interrupt vector points to an interrupt service routine specified by the program.

The VDS does not use interprocessor interrupts.

* Exceptions and Unexpected Interrupts:

The SCB of each processor is initialized so that exceptions vector into VDS condition handlers that provide interlocking. The last chance handler stops program execution on all processors, no matter which processor "trapped out". The primary processor re-enters VDS CLI and issues the DS> prompt. All attached processors re-enter their idle loops.

Diagnostic programs can override the VDS last chance handler by specifying their own condition handlers.

Unexpected device interrupts are fielded by the primary processor. The primary processor's unexpected interrupt handler reports the interrupt to the user and re-enters VDS CLI, issuing the DS> prompt. All attached processors are forced to re-enter their idle loops.

* Control-Cs:

  You can return to the VDS CLI and the DS> prompt by typing control-C. Attached processors return to their idle loops the next time they call the $DS_BREAK service.

  As is currently the case, a diagnostic program may declare its own control-C handler to take precedence over the VDS control-C handler.

* Communication Between the Main and Attached Processes:

  Since the VDS does not provide specific services to facilitate communication between the attached processes and the primary process, you can use any of the following techniques:

  o Event Flags - Very useful for passing status information between the primary and various attached processes. The $SETEF, $CLREF, $READEF, $WAITFR, $WFLAND, and $WFLOR services may all be used by any process.

  o Interprocessor Interrupts - Available for use by the diagnostic program; you may design any scheme you wish for making use of interprocessor interrupts.

  o Common "mailbox" - You can specify a common data area in which to pass information back and forth between the main process and the attached processes.

  o Dispatch vectors - Attached processes can call routines in the main process via dispatch vectors, stored in a table in the main process, that point to the routines. If these vectors are assigned to absolute addresses, then attached processes not linked with the main process can reference them. (If the code for attached processes is linked with the main process, dispatch vectors are unnecessary, since addressing references may be relative and can be resolved at link time.)

* Restrictions:

  The following restrictions apply to diagnostic programs using the multiprocessing features of the VDS:

- As with single processor systems, code executing in attached processors must periodically call the $DS_BREAK service. THIS RULE IS VERY IMPORTANT, as breakpoints, control-C's, and exception handling depend upon this rule being followed.

- With the exception of $DS_BGNATTACHED and $DS_ENDATTACHED, code executing in attached processors may not use any of the program structure macros. (These macros include $DS_BGNTEST, $DS_ENDTEST, $DS_BGNSUB, $DS_ENDSUB, and the macros $DS_HEADER and $DS_DISPATCH that define such data structures as the diagnostic header and the dispatch table, respectively.) All initialization and clean-up code, looping, and error reporting must be contained within code executed by the primary processor.

- The load image for a diagnostic program may not be larger than approximately 63.5 Kbytes. If the total size of the code to be executed by the primary and all attached processes exceeds the maximum, you have to store the code for attached processors in separate loadable images. (Refer to the section entitled "Executing in an Attached Processor".)

- As stated previously, requests for system services are not enqueued (except in the case of ASTs). Therefore if several attached processes are simultaneously requesting the same service, there is no way to determine which process will be serviced next. It is assumed (but not guaranteed by the software) that all requests will eventually be serviced. All services have an interlocking mechanism, so that the next processor to execute the service is the first one to reference the interlocking flag after the service is released (by the previous processor.)

- Only the primary processor can receive device interrupts.

- Use the standard methods for declaring condition handlers.

- All multi-processor diagnostics must run in Kernel mode.

- It is recommended that the diagnostic clean-up code call the $DS_HALTATTACHED service in order that each Attached Processor that was bootattached be placed in a known, static state.

- Code executing in an attached processor may not call the following services:

  $DS_ASKxxxx
  $DS_CHANNEL
  $DS_CLRVEC
  $DS_ERRxxx
  $DS_INITSCB
  $DS_LOAD
  $DS_MMON, $DS_MMOFF
  $DS_PARSE
  $DS_PRINTx
  $DS_SETVEC
  $SETIMR
  $DS_SETMAP
  $DS_WAITMS

* Multi-processor Macros:

  - $DS_BGNATTACHED, $DS_ENDATTACHED

    In a diagnostic program that tests multiple processors, use the $DS_BGNATTACHED and $DS_ENDATTACHED macros to delineate code that is to be executed in an attached processor. These macros are used whether the code is included in the loadable image of the main diagnostic program or it is a separate loadable image.

    $DS_BGNATTACHED indicates the beginning of the code and creates a label that can be used with the $DS_STARTATTACHED service. The $DS_ENDATTACHED macro generates code that will send the processor back to its idle loop.

    MACRO-32 Format:

```
$DS_BGNATTACHED routine_name, mask
        .
        .
        .
$DS_ENDATTACHED
```

    BLISS-32 Format:

```
$DS_BGNATTACHED (ROUTINE_NAME=routine_name);
        .
        .
        .
$DS_ENDATTACHED;
```

routine_name

Labels the routine and points to its first instruction.

mask

List of register names used in the entry mask.

Notes:

1. You can include code that is contained in an attached process in any number of separate executable files. The code in each file, however, must be position-independent. You can only have one attached process, delimited by one set of $DS_BGNATTACHED and $DS_ENDATTACHED macros, per file.

2. If you want to place the code in a separate image, request a buffer using the $DS_GETBUF service, load the image into the buffer, and use the address of the buffer as the "start_addr" argument for the $DS_STARTATTACHED macro.

3. You can enter the code using a CALL instruction.

4. It is recommended that you place data structures for the code in a separate psect. If you must include the data structures

   in the same psect as the code, place them (data structures) after the code and end the executable section with a $DS_EXIT macro as shown:

```
.psect data                 $DS_BGNATTACHED RTN2
        .                           .
<data structures>                   .
        .                   <executable code>
        .                           .
.psect code                         .
$DS_BGNATTACHED RTN1        $DS_EXIT ATTACHED
        .                           .
        .                           .
<executable code>           <data structures>
        .                           .
        .                           .
        .                           .
$DS_ENDATTACHED             $DS_ENDATTACHED
```

o $DS_BOOTATTACHED

In a multiprocessing environment, use the Boot Attached CPU system service to bootstrap an attached processor on a multiprocessor system. It will perform the following functions for the target processor:

1. Halt the processor, if it is not currently halted.

2. Perform all initialization necessary to enable the processor to execute code, including initializing stacks, memory management registers, the SCB, and the interval clock.

3. Cause the processor to begin executing an idle loop ("JMP ." instruction).

After you call the $DS_BOOTATTACHED, you can use the $DS_STARTATTACHED service to cause the attached processor to leave its idle loop and begin executing a section of code.

MACRO-32 Format:

```
$DS_BOOTATTACHED_x unit, scb_addr
```

BLISS-32 Format:

```
$DS_BOOTATTACHED (UNIT=unit, SCB_ADDR=scb_addr);
```

unit

Logical unit number of the processor to be bootstrapped.

scb_addr

Address of the longword to receive the SCB address of the target processor.

Return Status:

DS$_NORMAL

Service successfully completed.

DS$_ILLUNIT

The specified logical unit number is too large.

DS$_INVCPU

Can't boot specified processor.

DS$_MEM_ALLOC_ERR

Could not allocate memory for attached processor's SCB and stacks.

DS$_INIT_FAIL
Attached processor did not return prompt from Control P or Initialize. (VAX 8200 only)

Examples:

MACRO-32 Example:

```
$DS_BOOTATTACHED_S      LOG_UNIT, PROC2_SCB
```

BLISS-32 Example:

```
$DS_BOOTATTACHED (UNIT = .LOG_UNIT, SCB_ADDR =
                  PROC2_SCB);
```

o $DS_HALTATTACHED

Use the Halt Attached CPU system service to stop program execution in an attached processor or a multiprocessor environment. In order to use this service, you must bootstrap the processor with the $DS_BOOTATTACHED service. This service should be used in the clean-up code of a multi-processor diagnostic to place each Attached Processor that was bootattached into a known, static state.

MACRO-32 Format:

```
$DS__HALTATTACHED__x unit
```

BLISS-32 Format:

```
$DS__HALTATTACHED (UNIT=unit);
```

unit

Logical unit number of the CPU to be halted.

Return Status:

DS$_NORMAL

Service successfully completed.

DS$_ILLUNIT

The specified logical unit number is too large.

DS$_INVCPU

The specified processor is the primary processor.

DS$_INIT_FAIL

The processor failed to send console prompt
only).

Notes:

1. In order to restart a halted processor, you must reboot using the $DS_BOOTATTACHED service and then use the $DS_STARTATTACHED service.

Examples:

MACRO-32 Example:

```
$DS__HALTATTACHED__S    LOG_UNIT
```

BLISS-32 Example:

```
$DS__HALTATTACHED (UNIT = .LOG_UNIT);
```

- $DS_SHOWIDLE

  In a multiprocessor environment, use the Show Idle Processors service to determine which attached processors are currently executing their "idle loops". Attached processors are placed in their idle loops when:

  1. The $DS_BOOTATTACHED service has completed bootstrapping the processor.

  2. An attached process, delimited by the $DS_BGNATTACHED and $DS_ENDATTACHED macros, finishes executing.

  3. A multiprocessor diagnostic program is stopped by a control-C or an exception.

  MACRO-32 Format:

  ```
  $DS_SHOWIDLE_x  bitmap
  ```

  BLISS-32 Format:

  ```
  $DS_SHOWIDLE (BITMAP=bitmap);
  ```

  bitmap

Address of the longword to receive a bitmap indicating which attached processors are currently executing their idle loops. There are 32 bits (0 through 31); each bit corresponding to a logical unit number. You need to look at the bits that correspond to logical unit numbers actually associated with attached processors. (See Note 1.)

Return Status:

DS$_NORMAL

Service successfully completed.

Notes:

1. One method for using this service is to create a bit mask. Each time you issue a $DS_GPHARD call and receive the address of a p-table for an attached processor, set the bit in the mask corresponding to the logical unit number for that p-table. When you call the $DS_SHOWIDLE service, test only the bits that are set in the mask you created.

Examples:

MACRO-32 Example:

```
IDLE_MASK:      .LONG 0
IDLE_PROC:      .LONG 0
                .
                .
        $DS_SHOWIDLE_S (IDLE_PROC)  ; Get idling
                                    ; processors.
        CMPL    IDLE_MASK,IDLE_PROC ; All idling?
        BEQL    300$                ; Yes, branch.
                .
                .
```

BLISS-32 Example:

```
                .
                .
        $DS_SHOWIDLE (BITMAP=IDLE_PROC); ! Get idling
                                         ! processors.
        IF (.IDLE_MASK NEQ .IDLE_PROC)   ! If not idling
        THEN                             ! then...
                .
                .
```

o $DS_STARTATTACHED

In a multiprocessor environment, you can use the Start Attached CPU system service to cause an attached processor to begin executing a section of code. Before you can use this service, you must bootstrap the specified processor with the $DS_BOOTATTACHED service.

You must delimit the section of code to be executed with the $DS_BGNATTACHED and $DS_ENDATTACHED program structure macros. The service, in conjunction with these macros, causes the specified processor to leave the idle loop that was created by the $DS_BOOTATTACHED service and jump to the code delimited by the macros. When it finishes executing, the processor re-enters the idle loop. (Refer to the description of the $DS_BGNATTACHED and $DS_ENDATTACHED macros for details.)

MACRO-32 Format:

```
$DS_STARTATTACHED_x unit, start_addr
```

BLISS-32 Format:

```
$DS_STARTATTACHED (UNIT=unit, START_ADDR=start_addr);
```

unit

Logical unit number of the processor to be bootstrapped.

start_addr

Address of the code to be executed in the attached processor. This argument must be the address of the code generated by a $DS_BGNATTACHED macro.

Return Status:

DS$_NORMAL

Service successfully completed.

DS$_ILLUNIT

The specified logical unit number is too large.

DS$_INVCPU

Can't start the specified processor. May mean the processor doesn't exist or the processor was not executing its idle loop (see $DS_BOOTATTACHED).

Examples:

MACRO-32 Example:

```
                .
                .
                .
        $DS_BGNATTACHED ROUTINE1
                .
                .
                .
        $DS_ENDATTACHED
                .
                .
                .
        $DS_STARTATTACHED_S     LOG_UNIT, ROUTINE1
```

BLISS-32 Example:

```
                .
                .
                .
        $DS_BGNATTACHED (ROUTINE_NAME=ROUTINE1);
                .
                .
                .
$DS_ENDATTACHED;
                .
                .
                .
$DS_STARTATTACHED (UNIT = .LOG_UNIT,
                   START_ADDR=ROUTINE1);
```

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 10.2 June 23, 1986

1 ENHANCEMENTS

* A new qualifier has been added to the START and RUN commands:

/TIME=[dddd-][hh:mm:ss.cc]

Indicates the length of time which the diagnostic program will run before execution is stopped. When the time limit expires, the diagnostic program's clean-up code is executed.

In the standalone mode, the amount of time which the program will run is the same as real time. When running under VMS, the program will run for the specified amount of cpu time rather than elapsed time.

If you also specify a number of passes to be run, the diagnostic program will stop when either the specified number of passes have been run, or the specified amount of execution time has elapsed, whichever occurs first.

If the diagnostic program's execution is interrupted by a Control-C and a CONTINUE command is typed, the diagnostic program will continue to execute for a period of time equal to the original time specified in the START or RUN command minus the time for which the diagnostic has already executed.

If the TIME qualifier is not used, or a value of zero is specified, the diagnostic program will not have a time limit placed upon its execution.

The rules for specifying a time value follow the VMS rules for specifying a delta time. The variable fields are as follows:

Field	Meaning
dddd	Number of days, 24-hour units (0 through 9999)
hh	Number of hours (0 through 23)
mm	Number of minutes (0 through 59)
ss	Number of seconds (0 through 59)
cc	Number of hundredths of seconds (0 through 99)

When you specify a time value, you can truncate the time field. You can also omit any of the variable fields, as long as you supply the punctuation marks.

When any field is omitted from a delta time value, the system supplies a value of zero for the field.

Examples:

Time Specification	Result
3-	3 days from now (72 hours)
1	1 hour from now
:20	20 minutes from now
2-:20	2 days and 20 minutes from now
::35	35 seconds from now

Notes:

1. This qualifier should not be used when the HALT flag is set, or when setting breakpoints.

2. The accuracy of this feature can be affected by the design of a diagnostic program. The diagnostic supervisor checks for an expired time limit whenever it checks for control-Cs. Since diagnostic programs are required to call a system service which checks for control-Cs at least every 3 seconds, the actual runtime should not deviate by more than 3 seconds from the desired runtime.

* Multiprocessor Enhancements

  o Breakpoints

    You can optionally set breakpoints in the code to be executed in attached processors. Whenever a breakpoint is reached by any processor, all processors are placed in their idle loops (except for the primary processor, which enters VDS command mode). Typing a CONTINUE command causes all processors to continue from the point they were stopped. Typing a NEXT command executes the next instruction only if the breakpoint was executed by the primary processor, so that you can only single step through code being executed in the primary processor.

    You can set one or more breakpoints on an Attached or Primary Processor but not both (no mixed bpt's).

    In addition, to be assured of hitting breakpoints in an Attached Processor; test code must be synchronized between the Primary and Attached CPU's such that if the Attached processor hits a breakpoint, the Primary processor would be able to report that breakpoint by

issuing periodic calls (at least every three seconds) to the $DSBREAK service.

One simple method of accomplishing this is shown in the following test structure. Of course, other methods for synchronization using the $DSBREAK service may exist.

```
Primary Processor code          Attached Processor Code
----------------------          -----------------------
    Set flag
    Start Attached Processor -----> BGNATTACHED
5$: BBC flag 10$                    Execute test code
    $DS_BREAK                                .
    BRB 5$
                                    $DS_BREAK
                                    Clear Flag
10$: Continue                             Here the Attached could
                                    wait for another flag from
                                    the Primary to continue.
```

In this structure the attached can hit any breakpoint set between the BGNATTACHED and the clearing of the flag. The Primary processor needs to loop on $DS_BREAK to be able to report an Ap breakpoint or condition and also to respond to control-C's. A sort of stop/go flag mechanism can be designed for the Primary and Attached to execute sections of test code where the Ap can perform a test, perhaps write results, and then signal the Primary, then the primary can continue and perhaps start another section of test code. It is important that $DS_BREAK 's be called from both the Primary and Attached Processors.

When an Attached Processor hits a breakpoint, the PC of the Primary Processor (that detected the breakpoint) will point to the instruction following the $DSBREAK service that resulted in the Ap BPT message.

o Mp Diagnostic Users

If a diagnostic program not using the multiprocessor features of the VDS is executed under the new version of the VDS, the user will see no differences from current versions of the VDS. There will be two differences that a user will see when executing a program designed to run on multiple processors.

1. He/she will have to attach each processor he/she desires to be executed by the diagnostic program. However, this process is identical to the attaching sequences that are currently necessary for attaching any device before testing it.

2. In order to examine or deposit into the general purpose registers (GPRs) or internal processor registers (IPRs) of an attached processor, the user must specify the generic name of the processor (used in the ATTACH command) to indicate which processor's registers are to be examined or deposited into.

   It will not be possible to examine registers in an attached processor unless a diagnostic program has previously called the $DS_BOOTATTACHED service to boot that processor, and that attached processor has hit a breakpoint.

   A new command, SET CPU, will be added. This command will change the default processor referenced by the EXAMINE and DEPOSIT commands. For example, the command sequence

```
SET CPU KA1
EXAMINE R5
DEPOSIT R3 4
```

   will cause R5 to be examined and the value "4" to be deposited into R3 of the processor whose logical unit number is KA1.

   To keep track of which registers are being examined or deposited to, another new command called SHOW CPU has been added. Typing a SHOW CPU will indicate either PRIMARY or the generic name of the Ap used in the Attach sequence.

o Mp Restrictions

  The following restrictions apply to users of diagnostic programs which make use of the multiprocessing facilities of the VDS:

  1. It will not be possible to use the VDS EXAMINE and DEPOSIT commands to examine or deposit into the IPRs of an attached processor unless that processor has previously been booted with the $DS_BOOTATTACHED service, and that processor has hit a breakpoint.

  2. The NEXT command (single-stepping) cannot be used for code executing in an attached processor. If it is desirable to single-step the Ap, then the user should deposit a halt at the point you wish to single-step the Ap, and set a breakpoint in the Primary Processor at a convenient place, such as after calling the $DS_STARTATTACHED service. When you start the test and hit the Ap halt instruction, you can then use console commands to single-step the Ap.

3. The PC of an attached processor cannot be modified with the DEPOSIT command.

4. Only the PSW portion of the PSL in an attached processor can be examined.

5. Breakpoints can only be set for one processor at-at-time.

6. The PC will not be indicated when continuing from a multiprocessor diagnostic since more that one processor may be resuming execution of code.

2 BUG FIXES

- For 8200 systems, if the VALID bit in the WATCH chip CSR was clear (due to a bad battery, for instance), the supervisor would not boot up. Instead it would halt. This has been fixed.

- A problem with starting selftest on BI adapters on SCORPIO systems has been fixed. Previously, on some occasions, when selftest was attempted on a KDB50 adapter, a BI cycle was being truncated, resulting in ICE, RDS, or corrupted memory. This could happen to any BI adapter. The symptom was seen while running EVRLF.

3 KNOWN BUGS AND DEFICIENCIES

- If you control-C an mp diagnostic and then start and control-C a SHOW MEM command for example, the next CONTINUE command will result in a continuation of the mp diagnostic.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 10.4 December 22,1986

1 ENHANCEMENTS

* Support for logical names and search lists in USER mode

  o VMS logical names and searchlist logical names are allowed to represent file specifications. If the equivalence string of the logical name includes only part of a directory specification, it will be concatenated with the supervisor's default directory. For example, if SYS$SYSROOT is equivalent to 1DUS0:[SYS3.], and you specify SYS$SYSROOT:[SYSMAINT], the resulting filespec will be 1DUS0:[SYS3.SYSMAINT].

  o This is supported for all VDS commands and services involving file operations.

* DIRECTORY command in USER mode

  o Previous versions of VDS required that you ATTACH a device before doing a DIRECTORY of that device. It is not necessary to ATTACH the device in Version 10.4 of VDS.

* USER mode default load path

  o The VDS now sets its initial default load path to the logical name SYS$MAINTENANCE. SYS$MAINTENANCE should be defined in the system logical name table to the disk and directory which contains the system's diagnostic programs. If it is not defined, see your system manager, or define it in your process's name table.

  o In order to change the default load path, use the SET LOAD command as before. You may use DCL to define an equivalence string for SYS$MAINTENANCE in your process logical name table in order to change the VDS's initial default load path.

  o The VDS sets the default disk and directory by redefining the SYS$DISK logical name with the $CRELOG system service, and changing the default directory with the $SETDDIR service. Thus, if you attempt to SET LOAD to SYS$DISK, the VDS will redefine the logical SYS$DISK to be equivalent to itself, and file services will fail.

* Volume Shadowing

  o The VDS now supports booting and loading diagnostics from a VMS shadow set.

  o In order to boot VDS from a shadow set, R3 should be set up for VMB in the following format:



    The virtual unit number corresponds to the unit number assigned to the shadow set. The member unit can be the physical drive number of any member of the shadow set. It is used by VMB and SYSBOOT during booting in case access via the shadow set virtual number fails. The VDS itself only attempts file services via the shadow set virtual number.

    For example, to boot from a shadowed set known as DUS19, which contains members DUA2 and DUA9, and use DUA9 as the boot path in case booting via the shadow set virtual number fails, load R3 as follows:

```
>>> D/G 3 80130009
```

  o The device naming convention for a shadowed disk is HSC50$DUSnnn, where nnn is the shadow set virtual unit number between 0 and 255. This is in contrast to a non-shadowed HSC50 disk which has a device name format of HSC50$DUAnnn.

* BCA

  o The BCA, BI to CI Adapter, is now supported in a boot and load path.

  o The BCA is ATTACHed with device type CIBCA, as follows:

```
DS> ATTACH CIBCA HUB PAA0   ! BCA adapter
BI Node Number? 6
BR? 4
CI Node? 2
DS>
```

* Global Section Support for VSDP

  o The VDS now maps part of its control region into a global section, in order that VSDP can examine pass count, test number, subtest number, etc. The global section is only

established when the VDS is being run from a mailbox, and does not affect the operation of the supervisor or diagnostic programs.

2 BUG FIXES

* In command mode, the DELETE key was not being processed correctly by the supervisor in standalone mode on video terminals. The symptom would appear when a command was typed, and the user tried to erase the command by typing the delete key. Although the screen appeared correct, the VDS's internal command buffer was not being cleared, and the originally typed command would be executed. This has been fixed.

* When a SHOW MEMORY/BUFFER command was given just after a program had called $DS_RELBUF and no buffers had been allocated, the VDS displayed a negative number of pages allocated. This has been fixed.

* If memory management had previously been turned on via a $DS_MMON call, a SHOW MM command would not indicate that memory management was on. This has been fixed.

* In user mode, an access control violation occurred as a result of a DIRECTORY command to a disk whose device name did not conform to the ggan or name$ggan format. This has been fixed.

3 KNOWN BUGS AND DEFICIENCIES

1. If memory management is turned on with the SET MM ON command, and file access to an HSC50 disk is attempted, you may experience file read errors on 86xx machines. The message you may see is:

   %RMS-F-RER, file read error (R0=0001C0F4)

   or the file access may cause the VDS to hang.

RELEASE NOTES FOR
VAX-11 DIAGNOSTIC SUPERVISOR
VERSION 10.5, EZSAA only
January 26, 1987

1 ENHANCEMENTS

* Support for logical names and search lists in USER mode

  o VMS logical names and searchlist logical names are allowed to represent file specifications. If the equivalence string of the logical name includes only part of a directory specification, it will be concatenated with the supervisor's default directory. For example, if SYS$SYSROOT is equivalent to 1DUS0:[SYS3.], and you specify SYS$SYSROOT:[SYSMAINT], the resulting filespec will be 1DUS0:[SYS3.SYSMAINT].

  o This is supported for all VDS commands and services involving file operations.

* DIRECTORY command in USER mode

  o Previous versions of VDS required that you ATTACH a device before doing a DIRECTORY of that device. It is not necessary to ATTACH the device in Version 10.4 of VDS.

* USER mode default load path

  o The VDS now sets its initial default load path to the logical name SYS$MAINTENANCE. SYS$MAINTENANCE should be defined in the system logical name table to the disk and directory which contains the system's diagnostic programs. If it is not defined, see your system manager, or define it in your process's name table.

  o In order to change the default load path, use the SET LOAD command as before. You may use DCL to define an equivalence string for SYS$MAINTENANCE in your process logical name table in order to change the VDS's initial default load path.

  o The VDS sets the default disk and directory by redefining the SYS$DISK logical name with the $CRELOG system service, and changing the default directory with the $SETDDIR service. Thus, if you attempt to SET LOAD to SYS$DISK, the VDS will redefine the logical SYS$DISK to be equivalent to itself, and file services will fail.

* Volume Shadowing

  o The VDS now supports booting and loading diagnostics from a VMS shadow set.

  o In order to boot VDS from a shadow set, R3 should be set up for VMB in the following format:



  The virtual unit number corresponds to the unit number assigned to the shadow set. The member unit can be the physical drive number of any member of the shadow set. It is used by VMB and SYSBOOT during booting in case access via the shadow set virtual number fails. The VDS itself only attempts file services via the shadow set virtual number.

  For example, to boot from a shadowed set known as DUS19, which contains members DUA2 and DUA9, and use DUA9 as the boot path in case booting via the shadow set virtual number fails, load R3 as follows:

```
>>> D/G 3 80130009
```

  o The device naming convention for a shadowed disk is HSC50$DUSnnn, where nnn is the shadow set virtual unit number between 0 and 255. This is in contrast to a non-shadowed HSC50 disk which has a device name format of HSC50$DUAnnn.

* BCA

  o The BCA, BI to CI Adapter, is now supported in a boot and load path.

  o The BCA is ATTACHed with device type CIBCA, as follows:

```
DS> ATTACH NBIA HUB NBIA0          ! NBI adapter
LOGICAL ADAPTER # 0
DS>
DS> ATTACH NBIB NBIA0 NBIB0        ! NBI adapter
BI # ? 0
BI Node Number (HEX) ? 2
DS>
DS> ATTACH CIBCA NBIB0 PAA0        ! BCA adapter
BI Node Number (HEX) ? 6
BR? 4
CI Node? 2
DS>
```

* Global Section Support for VSDP

  o The VDS now maps part of its control region into a global section, in order that VSDP can examine pass count, test number, subtest number, etc. The global section is only established when the VDS is being run from a mailbox, and does not affect the operation of the supervisor or diagnostic programs.

* Multiprocessor Enhancements

  o Breakpoints

    You can optionally set breakpoints in the code to be executed in attached processors. Whenever a breakpoint is reached by any processor, all processors are placed in their idle loops (except for the primary processor, which enters VDS command mode). Typing a CONTINUE command causes all processors to continue from the point they were stopped. Typing a NEXT command executes the next instruction only if the breakpoint was executed by the primary processor, so that you can only single step through code being executed in the primary processor.

    You can set one or more breakpoints on an Attached or Primary Processor but not both (no mixed bpt's).

    In addition, to be assured of hitting breakpoints in an Attached Processor, test code must be synchronized between the Primary and Attached CPU's such that if the Attached processor hits a breakpoint, the Primary processor would be able to report that breakpoint by issuing periodic calls (at least every three seconds) to the $DS_BREAK service.

    One simple method of accomplishing this is shown in the following test structure. Of course, other methods for synchronization using the $DS_BREAK service may exist.

```
Primary Processor code          Attached Processor Code
----------------------          -----------------------
    Set flag
    Start Attached Processor -----> BGNATTACHED
5$: BBC flag 10$                Execute test code
    $DS_BREAK                               .
    BRB 5$                                  .
                                $DS_BREAK
                                Clear Flag
10$: Continue                           Here the Attached could
                                wait for another flag from
                                the Primary to continue.
```

    In this structure the attached can hit any breakpoint set between the BGNATTACHED and the clearing of the flag.

The Primary processor needs to loop on $DS_BREAK to be able to report an Ap breakpoint or condition and also to respond to control-C's. A sort of stop/go flag mechanism can be designed for the Primary and Attached to execute sections of test code where the Ap can perform a test, perhaps write results, and then signal the Primary, then the primary can continue and perhaps start another section of test code. It is important that $DS_BREAK 's be called from both the Primary and Attached Processors.

When an Attached Processor hits a breakpoint, the PC of the Primary Processor (that detected the breakpoint) will point to the instruction following the $DS_BREAK service that resulted in the Ap BPT message.

o Mp Diagnostic Users

If a diagnostic program not using the multiprocessor features of the VDS is executed under the new version of the VDS, the user will see no differences from current versions of the VDS. There will be two differences that a user will see when executing a program designed to run on multiple processors.

1. He/she will have to attach each processor he/she desires to be executed by the diagnostic program. However, this process is identical to the attaching sequences that are currently necessary for attaching any device before testing it.

2. In order to examine or deposit into the general purpose registers (GPRs) or internal processor registers (IPRs) of an attached processor, the user must specify the generic name of the processor (used in the ATTACH command) to indicate which processor's registers are to be examined or deposited into.

   It will not be possible to examine registers in an attached processor unless a diagnostic program has previously called the $DS_BOOTATTACHED service to boot that processor, and that attached processor has hit a breakpoint.

   A new command, SET CPU, will be added. This command will change the default processor referenced by the EXAMINE and DEPOSIT commands. For example, the command sequence

```
SET CPU KA1
EXAMINE R5
DEPOSIT R3 4
```

   will cause R5 to be examined and the value "4" to be deposited into R3 of the processor whose logical unit number is KA1.

To keep track of which registers are being examined or deposited to, another new command called SHOW CPU has been added. Typing a SHOW CPU will indicate either PRIMARY or the generic name of the Ap used in the Attach sequence.

o Mp Restrictions

The following restrictions apply to users of diagnostic programs which make use of the multiprocessing facilities of the VDS:

1. It will not be possible to use the VDS EXAMINE and DEPOSIT commands to examine or deposit into the IPRs of an attached processor unless that processor has previously been booted with the $DS_BOOTATTACHED service, and that processor has hit a breakpoint.

2. The NEXT command (single-stepping) cannot be used for code executing in an attached processor. If it is desirable to single-step the Ap, then the user should deposit a halt at the point you wish to single-step the Ap, and set a breakpoint in the Primary Processor at a convenient place, such as after calling the $DS_STARTATTACHED service. When you start the test and hit the Ap halt instruction, you can then use console commands to single-step the Ap.

3. The PC of an attached processor cannot be modified with the DEPOSIT command.

4. Only the PSW portion of the PSL in an attached processor can be examined.

5. Breakpoints can only be set for one processor at a time.

6. The PC will not be indicated when continuing from a multiprocessor diagnostic since more that one processor may be resuming execution of code.

2 BUG FIXES

* In standalone mode, the time stamp in some VDS messages was appearing as "Jan. 1, 1986 0:0:0" even though the correct time and date would appear in other messages. This has been fixed.

* In command mode, the DELETE key was not being processed correctly by the supervisor in standalone mode on video terminals. The symptom would appear when a command was typed, and the user tried to erase the command by typing the delete key. Although the screen appeared correct, the VDS's internal command buffer was not being cleared, and the originally typed command would be executed. This has been fixed.

* When a SHOW MEMORY/BUFFER command was given just after a program had called $DS_RELBUF and no buffers had been allocated, the VDS displayed a negative number of pages allocated. This has been fixed.

* If memory management had previously been turned on via a $DS_MMON call, a SHOW MM command would not indicate that memory management was on. This has been fixed.

* In user mode, an access control violation occurred as a result of a DIRECTORY command to a disk whose device name did not conform to the ggan or name$ggan format. This has been fixed.

* When booting from an HSC disk connected to a CIBCI or CIBCA, the CI_NODE ptable contained the incorrect value in the functionality field (HP$L_CI_FUNC) and the CI node field (HP$B_CI_NODE). No symptom was reported other than a SHOW DEVICE command displaying the wrong information for the CI node. This has been fixed.

* If you control-C an mp diagnostic and then start and control-C a SHOW MEM command for example, the next CONTINUE command will result in a continuation of the mp diagnostic.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 10.6, EBSAA only Jan. 29, 1987

1 ENHANCEMENTS

* This version provides support for the 8250/8350 systems. These systems contain faster cpus than the original 8200/8300 systems. The VDS which supports the new systems is EBSAA.EXE. Note that the processor is still ATTACHed with device type KA820 for the 8250/8350 systems. The only services which are affected are the $DS_WAITMS and $DS_WAITUS, and these have been adjusted to account for the increased processor speed.

RELEASE NOTES FOR VAX-11 DIAGNOSTIC SUPERVISOR VERSION 10.7 March 23, 1987

1 ENHANCEMENTS

* This version provides support for the 8250/8350 systems. These systems contain faster cpus than the original 8200/8300 systems. The VDS which supports the new systems is EBSAA.EXE. Note that the processor is still ATTACHed with device type KA820 for the 8250/8350 systems. The only services which are affected are the $DS_WAITMS and $DS_WAITUS, and these have been adjusted to account for the increased processor speed.

* Ptables have been added for the LG01, LG02, and LG03 lineprinters.

2 BUG FIXES

* A bit field in the SBIA SILO register was being reported by the 86xx EDSAA.EXE as "Reserved" when it contained a value of 6. This occurred when the VDS interpreted SILO information after receiving SBIA Fault Interrupts. This has been corrected to report "Interrupt Summary Read".

* When a VDS command, which involved accessing a file, was executed while running under VMS, and the file was protected, VDS would report "Unknown error". This has been changed to report "Insufficient privilege or file protection violation".

* When using the VDS NEXT command to single step over an instruction which had a breakpoint set, the VDS display after the next command would show the 03 (BREAKPOINT) opcode instead of the original opcode. This has been fixed so that the "real" opcode is displayed.

RELEASE NOTES FOR VAX DIAGNOSTIC SUPERVISOR VERSION 10.8 June 22, 1987

1 ENHANCEMENTS

* Ptables and boot and load support have been added for the RA70 non-removable disk.

* Added ptables for DEBNA and DEBNK.

* Updated SHOW BREAKPOINT display and the breakpoint message (primary and attached processor) which is displayed when a breakpoint is reached, to include the base address and offset which were used when setting the breakpoint. For example,

```
DS> SET BASE 9010
DS> SET BREAKPOINT E
DS> SHOW BREAKPOINT
Breakpoints located at:
                0000901E(X)     (Base=00009010(X) Offset=0000000E(X))

when the breakpoint is reached the following will be displayed:

Breakpoint at: 0000901E(X)     (Base=00009010(X) Offset=0000000E(X))
DS>
```

2 BUG FIXES

* When a zero value was passed for one of the $DS_LOAD parameters (FILE, DEFAULT, ADDRESS, RETLEN, or RETREC), an access violation would occur when VDS tried to access the location with memory management turned on in the standalone mode, or when running under VMS. The supervisor was then changed so that the $DS_LOAD service checked these parameters for zero, and returned an error code if so. Unfortunately, this change caused several diagnostics to fail by not allowing them to load files (usually microcode) into memory. So far, EVMAE and EVDFD are known to experience this problem. This problem exists in VDS versions 10.0 (EBSAA prerelease), 10.1 (EZSAA prerelease), 10.2 (release 25 and 26), 10.4 (rel. 27), 10.5 (EZSAA prerelease), 10.6 (EBSAA 8250 prerelease) and 10.7 (rel. 28).

Some of these diagnostic programs were not affected by the access violation because they do not run under VMS or in standalone with memory management turned on. So that these diagnostic programs do not have to be rereleased, the VDS will no longer return an error code when a parameter of zero is passed. For those programs which are required to run with memory management enabled, the access violations should be caught by developers when executing the QA checklist in the section which requires memory management to be enabled. Those diagnostics should be fixed.

* If a breakpoint in a diagnostic program was reached after a $DS_SETVEC macro had been performed to capture the machine check (04), translation not valid (24), access violation (20), or reserved operand (18) vectors, any EXAMINE or DEPOSIT command which resulted in one of these faults would cause control to pass to the diagnostic program's exception handler.

  This has been fixed by changing the EXAMINE and DEPOSIT commands to reset these vectors in the system control block for the duration of the EXAMINE or DEPOSIT command. A side effect of this change is that if the VDS EXAMINE command is used to EXAMINE these SCB vectors in this scenario, the normal VDS SCB vectors will be returned. To examine the vectors as set by the diagnostic program, use the console EXAMINE command.

* The parameter generic-name for DEATTACH command MUST be specified now. DEATTACH command also takes 'ALL' as a keyword parameter to de-attach all devices attached to the link, excluding the link itself. VDS User's Guide will be updated correspondingly.

* Previously, if the VDS was booted from a disk connected to an HSC50, you could not load diagnostics from a different disk/HSC50 (one other than an HSC50 specified as the failover path in R2 at boot time) by ATTACHing and setting the default load path to it. The VDS would still try to access the disk via the primary or failover HSC50 path which was specified in R2. This has been fixed.

* A hanging problem which was amplified by Rev 22E console program appeared in EZSAA while running in a script file involving a lots of terminal I/O. EZSAA console modules CONSOLE8800 and CSDUBTDRV are modified to catch a missing XON sent from console to resume from holding data.

* Fixed the bug in supporting DHB32.

* Fixed the bug in booting VDS from KDB50 and UDA50.

RELEASE NOTES FOR
VAX DIAGNOSTIC SUPERVISOR

VERSION 10.9
Sep. 21, 1987

1 NEW DEVICE SUPPORT

* Added ptable descriptor for RV20.

* Bliss support for DHB32. Added Bliss definitions for DHB32 in Bliss DIAG library so that any diagnostic written in Bliss can use those definitions.

2 ENHANCEMENTS

* In the past, a boot device, either DWBUA, KDB50, CIBCI, or KFBTA, had to be selftested again after the initialization done by its device driver in VMB. This redundancy has been eliminated in a BI system.

* Boot and load support for 2nd console RX50 drive on 82xx/83xx systems. Previously, VDS could be booted from one console RX50 drive called CSA0 only. Now, an operator can have the options of booting VDS from either CSA1 or CSA2.

3 BUG FIXES

* When a device is specified without a colon for the 'SET LOAD' command, a colon will be automatically postfixed to the device name. Prior to the fix, a VDS command 'SET LOAD DBA2', for example, appeared to succeed, but did not change the default to device DBA2 because it did not actually recognize DBA2 as a device without a postfixed colon.

* A directory specification for 'SET LOAD' command was not parsed the same way as by VMS. For example, VDS did not take 'SET LOAD RELD$1:[28]' as a legal command while 'SET DEFAULT RELD$1:[28]' is perfectly fine with VMS. This was because VDS treated a number in a directory specification as an octal format of UIC, which invalidated a number with digits higher than seven. VDS is now made consistent with VMS by generalizing any number as alphanumeric format of either a UIC or a name of a directory.

- VDS used to reject a device with unit number 255. For example, a VDS attach command 'ATTACH RA80 HUB DUA255' would cause an error message of 'Unit number too big or not allowed'. This was due to an error in treating a maximum unit number. It has been fixed.

RELEASE NOTES FOR
VAX DIAGNOSTIC SUPERVISOR
VERSION 10.10
Dec. 21, 1987

1 NEW SYSTEM SUPPORT

2 NEW DEVICE SUPPORT

- Added ptables for DEBNA, DEBNT and LANCE.

3 ENHANCEMENTS

- Supervisor onboard image PCS750.BIN has been upgraded to Rev. 99.

4 BUG FIXES

- $DS_ASKSTR macro could not be used by BLISS source because R1 containing returned pointer could not be easily referenced by caller. The register can now be referenced by BLISS source with Linkage-Declarations.
- Encountered a 'Reserved/privileged instruction fault' while attempting to do a 'SHOW MEM/DAT' after running EVKAE. It has been soloved by checking for proper mode before accessing privileged register.
- The bug occasionally causing access violation during 'DIRECTORY' command in online mode has been fixed. This problem has existed since VDS Ver. 10.3. Its PC at error is somewhere near address 00026000.

Object Module Synopsis

Module Name	Ident	Bytes	File	Creation Date	Creator
KERNEL	10.9-36	7377	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	18-NOV-1987 15:15	VAX/VMS Macro V04-00
ACPFDT	6.1-3	251	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:48	VAX/VMS Macro V04-00
ANSI	7.0-7	2482	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-Dec-1987 11:49	VAX Bliss-32 V4.3-808
APBOOT	7.0-0	91	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:49	VAX/VMS Macro V04-00
APT	10	505	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:49	VAX/VMS Macro V04-00
ASSIGN	8.3-3	306	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:49	VAX/VMS Macro V04-00
ASTCON	7.0-2	44	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:50	VAX/VMS Macro V04-00
ASTDEL	6.7-7	436	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:50	VAX/VMS Macro V04-00
ATTACH	10.10-9	5797	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:50	VAX/VMS Macro V04-00
BOOTDRIVR	10.0-28	770	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:50	VAX/VMS Macro V04-00
BOOTIO	6.12-2	84	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:51	VAX/VMS Macro V04-00
BUFCTL	5.4-3	100	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:51	VAX/VMS Macro V04-00
BUFFER	10.2-10	286	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:51	VAX/VMS Macro V04-00
CALLFRAME	01-01	464	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-Dec-1987 11:51	VAX Bliss-32 V4.3-808
CANCEL	05-04	265	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:51	VAX/VMS Macro V04-00
CHAN730	06-11	1330	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	9-DEC-1987 11:52	VAX/VMS Macro V04-00
CHMK	10-09	260	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:26	VAX/VMS Macro V04-00
CLI	10.10-28	8617	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:26	VAX/VMS Macro V04-00
CLOCK	07-36	1238	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:27	VAX/VMS Macro V04-00
COMDRVSUB	06.01	245	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:28	VAX/VMS Macro V04-00
CONFIG	10.10-45	3103	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	17-DEC-1987 14:09	VAX/VMS Macro V04-00
CONSOLE	09-35	2082	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:29	VAX/VMS Macro V04-00
CRDIMAGE	08-19	5860	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:30	VAX Bliss-32 V4.3-808
CSDDBTDRV	06-12	649	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:30	VAX/VMS Macro V04-00
DSVECS	01-10	112	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:34	VAX/VMS Macro V04-00
CVRTIM	V02-04	1062	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:31	VAX/VMS Macro V04-00
CVTFILNAM	06-02	214	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:31	VAX/VMS Macro V04-00
DASSGN	05-02	189	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:31	VAX/VMS Macro V04-00
DEBUG	10.10-8	5226	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:32	VAX/VMS Macro V04-00
DEVALC	05-05	353	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:32	VAX/VMS Macro V04-00
DEVICE	06-01	175	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:32	VAX/VMS Macro V04-00
DIRECTORY	10.10-12	5627	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:32	VAX Bliss-32 V4.3-808
DISPAT	10.10-04	2313	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:32	VAX/VMS Macro V04-00
DLBTDRIVR	V02-002	312	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:33	VAX/VMS Macro V04-00
DMBTDRIVR	V02-002	315	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:33	VAX/VMS Macro V04-00
DRINT	V01	29	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:33	VAX/VMS Macro V04-00
DSLOAD	10.8-12	941	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:33	VAX Bliss-32 V4.3-808
DQBTDRIVR	V02-001	333	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:33	VAX/VMS Macro V04-00
ENTRY	10-43	3057	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:34	VAX/VMS Macro V04-00
ERROR	10-5	4080	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:34	VAX/VMS Macro V04-00
EVENT	06-01	241	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:34	VAX Bliss-32 V4.3-808
EXCEPT	07-34	4148	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:35	VAX/VMS Macro V04-00
FAO	03-02	865	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:35	VAX/VMS Macro V04-00
FLAGS	07-14	601	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:35	VAX/VMS Macro V04-00
FILEREAD	10-03	2137	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:35	VAX/VMS Macro V04-00
FRKCTL	06-03	102	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:35	VAX/VMS Macro V04-00
GETTIM	01	26	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:36	VAX/VMS Macro V04-00
GTCHAN	02-01	136	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:36	VAX/VMS Macro V04-00
HELP	10.10-2	2718	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:36	VAX Bliss-32 V4.3-808

Module Name	Ident	Bytes	File	Creation Date	Creator
IOBASE	10.10-47	14448	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	18-NOV-1987 15:12	VAX/VMS Macro V04-00
IOPOST	05-04	979	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:38	VAX/VMS Macro V04-00
IOSNPG	06-04	770	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:38	VAX/VMS Macro V04-00
IOSPGD	05-02	402	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:38	VAX/VMS Macro V04-00
IOSRAM	05-02	373	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:38	VAX/VMS Macro V04-00
LOAD	10.9-03	862	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:39	VAX/VMS Macro V04-00
LODMAP	05-02	255	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:39	VAX/VMS Macro V04-00
LOOP	07-19	815	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:39	VAX/VMS Macro V04-00
MCHK730	06-07	1163	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:40	VAX Bliss-32 V4.3-808
MBAINT	01	180	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:39	VAX/VMS Macro V04-00
MEMALC	07-09	826	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:41	VAX/VMS Macro V04-00
MEMMGT	10-39	2531	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:41	VAX/VMS Macro V04-00
MPDUMMY	8.0	64	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	12-Nov-1987 14:45	VAX Bliss-32 V4.3-808
ODS	01-04	972	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:44	VAX Bliss-32 V4.3-808
ONLY730	10.10-3	1036	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:44	VAX/VMS Macro V04-00
PARAM	9.1-17	1717	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:47	VAX/VMS Macro V04-00
PARSE	07-20	1446	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:47	VAX/VMS Macro V04-00
PCSLOAD	06-03	77	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:48	VAX Bliss-32 V4.3-808
PRINT	07-34	3860	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:48	VAX/VMS Macro V04-00
PROBE	01-05	189	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:48	VAX Bliss-32 V4.3-808
PUBTDRIVR	10.10-2	768	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:49	VAX/VMS Macro V04-00
QACHECKS	10.10-02	2107	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:49	VAX Bliss-32 V4.3-808
QAFLAGS	6.6-04	484	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:49	VAX Bliss-32 V4.3-808
QAMAIN	1-08	2807	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:49	VAX Bliss-32 V4.3-808
QIO	09-28	3247	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:50	VAX/VMS Macro V04-00
QIOFDT	05-03	479	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:50	VAX/VMS Macro V04-00
QIOREQ	01-02	564	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:50	VAX/VMS Macro V04-00
RELOCDRV	01	164	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:50	VAX/VMS Macro V04-00
RMS	10.10-20	3903	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	18-Nov-1987 15:17	VAX Bliss-32 V4.3-808
RT11	01-04	835	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:51	VAX Bliss-32 V4.3-808
SCB	07-34	1132	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	18-NOV-1987 15:15	VAX/VMS Macro V04-00
SCRIPT	10-27	1877	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:51	VAX/VMS Macro V04-00
SHOWMEMORY	10.10-3	2323	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:52	VAX Bliss-32 V4.3-808
START	10.1-10	2450	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:52	VAX/VMS Macro V04-00
TSBTDRIVR	V01-02	522	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:53	VAX/VMS Macro V04-00
MUBTDRIVR	06-12	1019	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:43	VAX/VMS Macro V04-00
ULTRIX	01-05	3201	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-Nov-1987 17:53	VAX Bliss-32 V4.3-808
UNWIND	01-1	297	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	11-NOV-1987 17:53	VAX/VMS Macro V04-00
VER730	10.10	108	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	17-DEC-1987 15:30	VAX/VMS Macro V04-00
VMSDUMMY	05-02	30	[DS.LOCAL.RELEASE.WORK]GOOD.OLB;1	5-DEC-1986 12:44	VAX/VMS Macro V04-00
SYS$IODEF	X-1	0	SYS$COMMON:[SYSLIB]STARLET.OLB;8	22-MAY-1987 19:42	VAX/VMS Macro V04-00
LIB$CVT_ATB	X-1	122	SYS$COMMON:[SYSLIB]STARLET.OLB;8	22-MAY-1987 21:58	VAX/VMS Macro V04-00
LIB$SIGNAL	X-1	220	SYS$COMMON:[SYSLIB]STARLET.OLB;8	22-MAY-1987 19:57	VAX/VMS Macro V04-00
SYS$PRDEF	X-1	0	SYS$COMMON:[SYSLIB]STARLET.OLB;8	22-MAY-1987 20:24	VAX/VMS Macro V04-00
SYS$SSDEF	X-1	0	SYS$COMMON:[SYSLIB]STARLET.OLB;8	22-MAY-1987 20:48	VAX/VMS Macro V04-00
SYS$P1_VECTOR	X-5M3	0	SYS$COMMON:[SYSLIB]STARLET.OLB;8	22-MAY-1987 20:10	VAX/VMS Macro V04-00

Image Section Synopsis

Cluster	Type	Pages	Base Addr	Disk VBN	PFC	Protection and Paging	Global Sec. Name	Match	Majorid	Minorid
DEFAULT_CLUSTER	0	79	00010000	2	0	READ ONLY				
	0	21	00019E00	81	0	READ WRITE COPY ON REF				
	0	152	0001C800	102	0	READ ONLY				
	0	24	0002F800	254	0	READ WRITE COPY ON REF				
	0	1	00032800	278	0	READ WRITE FIXUP VECTORS				
	253	20	7FFFD800	0	0	READ WRITE DEMAND ZERO				

Key for special characters above:

! R - Relocatable !
! P - Protected !

```
                                    +---------------------------+
                                    ! Program Section Synopsis !
                                    +---------------------------+

Psect Name      Module Name       Base     End          Length            Align                 Attributes
----------      -----------       ----     ---          ------            -----                 ----------

$BASE                           00010000 00010B27 00000B28 (      2856.) QUAD 3 NOPIC,USR,CON,REL,LCL,  SHR,NOEXE,  RD,NOWRT,NOVEC
                ENTRY           00010000 00010B27 00000B28 (      2856.) QUAD 3

DATA                            00010B40 00019CE3 000091A4 (     37284.) 2 ** 5 NOPIC,USR,CON,REL,LCL,  SHR,NOEXE,  RD,NOWRT,NOVEC
                KERNEL          00010B40 00010CEE 000001AF (       431.) BYTE 0
                ANSI            00010CF0 00010D07 00000018 (        24.) LONG 2
                APBOOT          00010D08 00010D42 0000003B (        59.) LONG 2
                APT             00010D44 00010D68 00000025 (        37.) LONG 2
                ASTDEL          00010D6C 00010D7E 00000013 (        19.) LONG 2
                ATTACH          00010D80 0001113C 000003BD (       957.) LONG 2
                CALLFRAME       00011140 00011220 000000E1 (       225.) LONG 2
                CHAN730         00011224 0001127C 00000059 (        89.) LONG 2
                CHMK            00011280 00011284 00000005 (         5.) LONG 2
                CLI             00011285 000121F1 00000F6D (      3949.) BYTE 0
                CLOCK           000121F2 000121F7 00000006 (         6.) BYTE 0
                CONFIG          000121F8 00012297 000000A0 (       160.) LONG 2
                CONSOLE         00012298 000124BD 00000226 (       550.) LONG 2
                CRDIMAGE        000124C0 00013112 00000C53 (      3155.) LONG 2
                DEBUG           00013113 00013445 00000333 (       819.) BYTE 0
                DEVICE          00013446 0001344C 00000007 (         7.) BYTE 0
                DIRECTORY       00013450 00013768 00000319 (       793.) LONG 2
                DISPAT          0001376C 00013A68 000002FD (       765.) LONG 2
                DSLOAD          00013A6C 00013ACB 00000060 (        96.) LONG 2
                ENTRY           00013AD0 00013B5F 00000090 (       144.) QUAD 3
                ERROR           00013B60 000144D7 00000978 (      2424.) BYTE 0
                EXCEPT          000144D8 00014C84 000007AD (      1965.) LONG 2
                FAO             00014C88 00014CC4 0000003D (        61.) LONG 2
                FLAGS           00014CC5 00014DAD 000000E9 (       233.) BYTE 0
                HELP            00014DB0 00014E63 000000B4 (       180.) LONG 2
                IOBASE          00014E64 00018087 00003224 (     12836.) BYTE 0
                IOPOST          00018088 00018098 00000011 (        17.) LONG 2
                LOAD            00018099 000180FC 00000064 (       100.) BYTE 0
                LOOP            000180FD 000181A0 000000A4 (       164.) BYTE 0
                MCHK730         000181A4 000185C3 00000420 (      1056.) LONG 2
                MEMALC          000185C4 000185DF 0000001C (        28.) BYTE 0
                MEMMGT          000185E0 0001861D 0000003E (        62.) LONG 2
                ONLY730         00018620 00018638 00000019 (        25.) LONG 2
                PARAM           00018639 00018775 0000013D (       317.) BYTE 0
                PCSLOAD         00018778 000187AA 00000033 (        51.) LONG 2
                PRINT           000187AB 000187BA 00000010 (        16.) BYTE 0
                QACHECKS        000187BC 00018B56 0000039B (       923.) LONG 2
                QAFLAGS         00018B58 00018BFE 000000A7 (       167.) LONG 2
                QAMAIN          00018C00 00019086 00000487 (      1159.) LONG 2
                QIO             00019088 0001918A 00000103 (       259.) LONG 2
                RMS             0001918C 000195B3 00000428 (      1064.) LONG 2
                SCB             000195B4 000195B7 00000004 (         4.) LONG 2
                SCRIPT          000195B8 00019628 00000071 (       113.) BYTE 0
                SHOWMEMORY      0001962C 000199BD 00000392 (       914.) LONG 2
```

Psect Name	Module Name	Base	End	Length		Align	Attributes
DATA		00010B40	00019CE3	000091A4 (	37284.)	2 ** 5	NOPIC,USR,CON,REL,LCL, SHR,NOEXE, RD,NOWRT,NOVEC
	START	000199BE	00019C59	0000029C (	668.)	BYTE 0	
	TSBTDRIVR	00019C60	00019C7F	00000020 (	32.)	2 ** 5	
	VER730	00019C80	00019CE3	00000064 (	100.)	LONG 2	
$GLOBAL$		00019E00	00019E03	00000004 (	4.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	MPDUMMY	00019E00	00019E03	00000004 (	4.)	LONG 2	
WORK		00019E04	0001C60B	00002808 (	10248.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	KERNEL	00019E04	0001A9E5	00000BE2 (	3042.)	LONG 2	
	APBOOT	0001A9E8	0001A9EF	00000008 (	8.)	LONG 2	
	ATTACH	0001A9F0	0001A9FF	00000010 (	16.)	LONG 2	
	BUFFER	0001AA00	0001AA1F	00000020 (	32.)	LONG 2	
	CHAN730	0001AA20	0001AB10	000000F1 (	241.)	LONG 2	
	CHMK	0001AB14	0001AB1B	00000008 (	8.)	LONG 2	
	CLI	0001AB1C	0001AF9B	00000480 (	1152.)	LONG 2	
	CLOCK	0001AF9C	0001B007	0000006C (	108.)	LONG 2	
	CONFIG	0001B008	0001B008	00000001 (	1.)	LONG 2	
	CONSOLE	0001B00C	0001B013	00000008 (	8.)	LONG 2	
	CRDIMAGE	0001B014	0001B097	00000084 (	132.)	LONG 2	
	DSVECS	0001B098	0001B107	00000070 (	112.)	LONG 2	
	DEBUG	0001B108	0001B309	00000202 (	514.)	LONG 2	
	DIRECTORY	0001B30C	0001B6C8	000003BD (	957.)	LONG 2	
	DISPAT	0001B6CC	0001B6D7	0000000C (	12.)	LONG 2	
	DSLOAD	0001B6D8	0001B713	0000003C (	60.)	LONG 2	
	ERROR	0001B714	0001B753	00000040 (	64.)	LONG 2	
	EXCEPT	0001B754	0001B76B	00000018 (	24.)	LONG 2	
	HELP	0001B76C	0001B799	0000002E (	46.)	LONG 2	
	IOBASE	0001B79C	0001BDCF	00000634 (	1588.)	LONG 2	
	LOAD	0001BDD0	0001BE27	00000058 (	88.)	LONG 2	
	MEMALC	0001BE28	0001BE43	0000001C (	28.)	LONG 2	
	MEMMGT	0001BE44	0001BE60	0000001D (	29.)	LONG 2	
	ONLY730	0001BE64	0001BF33	000000D0 (	208.)	LONG 2	
	PARAM	0001BF34	0001BF37	00000004 (	4.)	BYTE 0	
	PARSE	0001BF38	0001BF8F	00000058 (	88.)	LONG 2	
	PRINT	0001BF90	0001C3B9	0000042A (	1066.)	LONG 2	
	QACHECKS	0001C3BC	0001C3BD	00000002 (	2.)	LONG 2	
	QAFLAGS	0001C3C0	0001C3C5	00000006 (	6.)	LONG 2	
	QAMAIN	0001C3C8	0001C3FC	00000035 (	53.)	LONG 2	
	QIO	0001C400	0001C40D	0000000E (	14.)	LONG 2	
	RMS	0001C410	0001C41F	00000010 (	16.)	LONG 2	
	SCB	0001C420	0001C470	00000051 (	81.)	LONG 2	
	SCRIPT	0001C474	0001C548	000000D5 (	213.)	LONG 2	
	SHOWMEMORY	0001C54C	0001C553	00000008 (	8.)	LONG 2	
	START	0001C554	0001C56F	0000001C (	28.)	LONG 2	
	ULTRIX	0001C570	0001C60B	0000009C (	156.)	LONG 2	
$CODE$		0001C800	0001C83B	0000003C (	60.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	MPDUMMY	0001C800	0001C83B	0000003C (	60.)	LONG 2	
CODE		0001C83C	0002F5E6	00012DAB (	77227.)	LONG 2	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	KERNEL	0001C83C	0001D77B	00000F40 (	3904.)	LONG 2	

Psect Name	Module Name	Base	End	Length		Align	Attributes
CODE		0001C83C	0002F5E6	00012DAB	(77227.)	LONG 2	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	ANSI	0001D77C	0001E115	0000099A	(2458.)	LONG 2	
	APBOOT	0001E118	0001E12F	00000018	(24.)	LONG 2	
	APT	0001E130	0001E303	000001D4	(468.)	LONG 2	
	ASTCON	0001E304	0001E32F	0000002C	(44.)	LONG 2	
	ASTDEL	0001E330	0001E4D0	000001A1	(417.)	LONG 2	
	ATTACH	0001E4D4	0001F7AB	000012D8	(4824.)	LONG 2	
	BOOTIO	0001F7AC	0001F7FF	00000054	(84.)	LONG 2	
	BUFFER	0001F800	0001F8FD	000000FE	(254.)	BYTE 0	
	CALLFRAME	0001F900	0001F9EE	000000EF	(239.)	LONG 2	
	CHAN730	0001F9F0	0001FDD7	000003E8	(1000.)	LONG 2	
	CHMK	0001FDD8	0001FECE	000000F7	(247.)	LONG 2	
	CLI	0001FECF	00020C8A	00000DBC	(3516.)	BYTE 0	
	CLOCK	00020C8C	000210EF	00000464	(1124.)	LONG 2	
	CONFIG	000210F0	00021C6D	00000B7E	(2942.)	LONG 2	
	CONSOLE	00021C70	00022263	000005F4	(1524.)	LONG 2	
	CRDIMAGE	00022264	00022C70	00000A0D	(2573.)	LONG 2	
	CVTFILNAM	00022C74	00022D49	000000D6	(214.)	LONG 2	
	DEBUG	00022D4A	00023C7E	00000F35	(3893.)	BYTE 0	
	DEVICE	00023C7F	00023D26	000000A8	(168.)	BYTE 0	
	DIRECTORY	00023D28	00024C4C	00000F25	(3877.)	LONG 2	
	DISPAT	00024C50	0002524F	00000600	(1536.)	LONG 2	
	DSLOAD	00025250	00025560	00000311	(785.)	LONG 2	
	ENTRY	00025561	00025599	00000039	(57.)	BYTE 0	
	ERROR	0002559A	00025BD1	00000638	(1592.)	BYTE 0	
	EVENT	00025BD4	00025CC4	000000F1	(241.)	LONG 2	
	EXCEPT	00025CC8	00026536	0000086F	(2159.)	LONG 2	
	FAO	00026538	0002685B	00000324	(804.)	LONG 2	
	FLAGS	0002685C	000269CB	00000170	(368.)	BYTE 0	
	FILEREAD	000269CC	00027224	00000859	(2137.)	LONG 2	
	HELP	00027228	00027BE3	000009BC	(2492.)	LONG 2	
	IOPOST	00027BE4	00027FA5	000003C2	(962.)	LONG 2	
	LOAD	00027FA6	00028247	000002A2	(674.)	BYTE 0	
	LOOP	00028248	000284D2	0000028B	(651.)	BYTE 0	
	MCHK730	000284D4	0002853E	0000006B	(107.)	LONG 2	
	MEMALC	0002853F	00028840	00000302	(770.)	BYTE 0	
	MEMMGT	00028844	000291CB	00000988	(2440.)	LONG 2	
	ODS	000291CC	00029597	000003CC	(972.)	LONG 2	
	ONLY730	00029598	000298BA	00000323	(803.)	LONG 2	
	PARAM	000298BC	00029E2F	00000574	(1396.)	LONG 2	
	PARSE	00029E30	0002A37D	0000054E	(1358.)	BYTE 0	
	PCSLOAD	0002A380	0002A399	0000001A	(26.)	LONG 2	
	PRINT	0002A39A	0002AE73	00000ADA	(2778.)	BYTE 0	
	QACHECKS	0002AE74	0002B311	0000049E	(1182.)	LONG 2	
	QAFLAGS	0002B314	0002B44A	00000137	(311.)	LONG 2	
	QAMAIN	0002B44C	0002BA86	0000063B	(1595.)	LONG 2	
	QIO	0002BA88	0002C625	00000B9E	(2974.)	LONG 2	
	RMS	0002C628	0002D12E	00000B07	(2823.)	LONG 2	
	RT11	0002D130	0002D472	00000343	(835.)	LONG 2	
	SCB	0002D474	0002D777	00000304	(772.)	LONG 2	
	SCRIPT	0002D778	0002DD86	0000060F	(1551.)	BYTE 0	
	SHOWMEMORY	0002DD88	0002E300	00000579	(1401.)	LONG 2	

Psect Name	Module Name	Base	End	Length	Align	Attributes
CODE		0001C83C	0002F5E6	00012DAB (77227.)	LONG 2	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	START	0002E301	0002E9DA	000006DA (1754.)	BYTE 0	
	ULTRIX	0002E9DC	0002F5C0	00000BE5 (3045.)	LONG 2	
	VER730	0002F5C1	0002F5C8	00000008 (8.)	BYTE 0	
	VMSDUMMY	0002F5C9	0002F5E6	0000001E (30.)	BYTE 0	
_LIB$CODE		0002F5E8	0002F73F	00000158 (344.)	LONG 2	PIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	LIB$CVT_ATB	0002F5E8	0002F661	0000007A (122.)	LONG 2	
	LIB$SIGNAL	0002F664	0002F73F	000000DC (220.)	LONG 2	
$SCB		0002F800	0002F8FF	00000100 (256.)	PAGE 9	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD, WRT,NOVEC
	SCB	0002F800	0002F8FF	00000100 (256.)	PAGE 9	
BOOTDRIVR_1		0002F900	0002FB8C	0000028D (653.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	0002F900	0002FB8C	0000028D (653.)	LONG 2	
BOOTDRIVR_2		0002FC00	00030BE2	00000FE3 (4067.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	CSDDBTDRV	0002FC00	0002FE70	00000271 (625.)	BYTE 0	
	DLBTDRIVR	0002FE71	0002FF90	00000120 (288.)	BYTE 0	
	DMBTDRIVR	0002FF91	000300B3	00000123 (291.)	BYTE 0	
	DQBTDRIVR	000300B4	000301E8	00000135 (309.)	BYTE 0	
	PUBTDRIVR	00030200	000304E7	000002E8 (744.)	PAGE 9	
	TSBTDRIVR	000304E8	000306B9	000001D2 (466.)	BYTE 0	
	MUBTDRIVR	00030800	00030BE2	000003E3 (995.)	PAGE 9	
BOOTDRIVR_3		00030BE3	00030BE3	00000000 (0.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	00030BE3	00030BE3	00000000 (0.)	BYTE 0	
BOOTDRIVR_4		00030BE3	00030C8A	000000A8 (168.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	CSDDBTDRV	00030BE3	00030BFA	00000018 (24.)	BYTE 0	
	DLBTDRIVR	00030BFB	00030C12	00000018 (24.)	BYTE 0	
	DMBTDRIVR	00030C13	00030C2A	00000018 (24.)	BYTE 0	
	DQBTDRIVR	00030C2B	00030C42	00000018 (24.)	BYTE 0	
	PUBTDRIVR	00030C43	00030C5A	00000018 (24.)	BYTE 0	
	TSBTDRIVR	00030C5B	00030C72	00000018 (24.)	BYTE 0	
	MUBTDRIVR	00030C73	00030C8A	00000018 (24.)	BYTE 0	
BOOTDRIVR_5		00030C8B	00030C8E	00000004 (4.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	00030C8B	00030C8E	00000004 (4.)	BYTE 0	
BOOTDRIVR_6		00030C8F	00030CFF	00000071 (113.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	BOOTDRIVR	00030C8F	00030CFF	00000071 (113.)	BYTE 0	
SEP		00030D00	000327A0	00001AA1 (6817.)	LONG 2	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD, WRT,NOVEC
	ACPFDT	00030D00	00030DFA	000000FB (251.)	LONG 2	
	ASSIGN	00030DFC	00030F2D	00000132 (306.)	LONG 2	
	BUFCTL	00030F30	00030F93	00000064 (100.)	LONG 2	
	CANCEL	00030F94	0003109C	00000109 (265.)	LONG 2	
	COMDRVSUB	000310A0	00031194	000000F5 (245.)	LONG 2	
	CVRTIM	00031198	000315BD	00000426 (1062.)	LONG 2	
	DASSGN	000315C0	0003167C	000000BD (189.)	LONG 2	
	DEVALC	00031680	000317E0	00000161 (353.)	LONG 2	

```
Psect Name      Module Name      Base     End          Length            Align                 Attributes
----------      -----------      ----     ---          ------            -----                 ----------
SEP                              00030D00 000327A0 00001AA1 (      6817.) LONG 2 NOPIC,USR,CON,REL,LCL,  SHR,  EXE,  RD,  WRT,NOVEC
                DRINT            000317E4 00031800 0000001D (        29.) LONG 2
                FRKCTL           00031804 00031869 00000066 (       102.) LONG 2
                GETTIM           0003186C 00031885 0000001A (        26.) LONG 2
                GTCHAN           00031888 0003190F 00000088 (       136.) LONG 2
                IOBASE           00031910 00031927 00000018 (        24.) LONG 2
                IOSNPG           00031928 00031C29 00000302 (       770.) LONG 2
                IOSPGD           00031C2C 00031DBD 00000192 (       402.) LONG 2
                IOSRAM           00031DC0 00031F34 00000175 (       373.) LONG 2
                LODMAP           00031F38 00032036 000000FF (       255.) LONG 2
                MBAINT           00032038 000320EB 000000B4 (       180.) LONG 2
                PROBE            000320EC 000321A8 000000BD (       189.) LONG 2
                QIOFDT           000321AC 0003238A 000001DF (       479.) LONG 2
                QIOREQ           0003238C 000325BF 00000234 (       564.) LONG 2
                RELOCDRV         000325C0 00032663 000000A4 (       164.) LONG 2
                SCB              00032664 00032676 00000013 (        19.) LONG 2
                UNWIND           00032678 000327A0 00000129 (       297.) LONG 2

Y$EXEPAGED                       000327A1 000327A1 00000000 (         0.) BYTE 0 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,  WRT,NOVEC
                ASTCON           000327A1 000327A1 00000000 (         0.) BYTE 0

_LAST                            00032800 00032800 00000000 (         0.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,  WRT,NOVEC
                ONLY730          00032800 00032800 00000000 (         0.) PAGE 9
```

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

Symbol                          Value          Defined By        Referenced By ...
------                          -----          ----------        -----------------
AAT_ARITH                       00014643-R      EXCEPT
ACC$HANDLER                     00032164-R      PROBE
ACP$ACCESS                      00030D00-R      ACPFDT             ENTRY
ACP$ACCESSNET                   00030D00-R      ACPFDT
ACP$DEACCESS                    00030D0B-R      ACPFDT             ENTRY
ACP$MODIFY                      00030D16-R      ACPFDT             ENTRY
ACP$MOUNT                       00030D21-R      ACPFDT             ENTRY
ACP$READBLK                     00030D2C-R      ACPFDT             ENTRY
ACP$WRITEBLK                    00030D49-R      ACPFDT             ENTRY
ACT_CONT                        0002022A-R      CLI
ACT_DEFAULT_DBG                 0002023D-R      CLI                START
ACT_NO_ALLOC                    000203A7-R      CLI
ANSI$CACHEVBN                   0001DD55-R      ANSI
ANSI$CLOSE                      0001DD24-R      ANSI
ANSI$GET                        0001DF1B-R      ANSI
ANSI$OPEN                       0001D77C-R      ANSI               RMS
ANSI$READ                       0001DE5A-R      ANSI
AP$B_PROMPT_BUFFER_COUNT        00000000                       WK-KERNEL
AP$B_PROMPT_COUNT               00000000                       WK-KERNEL
AP$GK_NO_PROMPT                 00000003        APBOOT
AP$GK_PROMPT                    00000001        APBOOT
AP$GK_PROMPT_WITH_STATUS        00000002        APBOOT
AP$GK_REQ_CONSRX                00000001        APBOOT
AP$GR_SHARED_DATA               0001A9EC-R      APBOOT             KERNEL
AP$GW_AP_NODES                  0001A9EA-R      APBOOT
AP$GW_DATA                      0001A9E8-R      APBOOT             CHMK               CLI                KERNEL
AP$I_PROMPT_CHECK               00000000                       WK-KERNEL
AP$_REQ_CONSRX                  00000000        KERNEL
AP$_REQ_RXCDVEC                 00000001        KERNEL
APSLMNE                         00013356-R      DEBUG              EXCEPT             MCHK730
APT                             0001C915-R      KERNEL             ENTRY
APT_COM                         0001E130-R      APT                CLI
APT_DONE                        0001E2F5-R      APT
APT_MSG                         0001E2EC-R      APT                CLI                ERROR
AX_BUFF                         00032800-R      ONLY730            KERNEL             SHOWMEMORY
BDA_MAPREGS                     00000000                       WK-BOOTDRIVR
BD_INIT                         00000000                       WK-CONFIG
BEGIN                           0001D0A9-R      KERNEL             CLI                DISPAT             DSVECS
                                                                   LOAD               LOOP               SCRIPT
                                                                   START
BEGIN_BLISS                     0001D0A9-R      KERNEL             CRDIMAGE           DIRECTORY          SHOWMEMORY
BEGIN_TIME                      0001C55C-R      START              KERNEL
BIN_TIME                        0001C554-R      START              CLI                KERNEL
BI_BER_BIT                      00000000                       WK-EXCEPT
BOO$AL_VECTOR                   0002F900-R      BOOTDRIVR          DIRECTORY          RMS
BOO$DRIVER                      0002FB8D-R      BOOTDRIVR
BOO$GB_UMR_DP                   0002F938-R      BOOTDRIVR
BOO$GL_UCODE                    0002F928-R      BOOTDRIVR
BOO$GL_UMR_DIS                  0002F924-R      BOOTDRIVR
```

Symbol	Value	Defined By	Referenced By ...		
BOO$GL_UMR_TMPL	0002F934-R	BOOTDRIVR			
BOO$IMAGE_ATT	0001F7CC-R	BOOTIO			
BOO$MAP	0002FAA8-R	BOOTDRIVR			
BOO$QIO	0002F950-R	BOOTDRIVR	ANSI	DIRECTORY	ODS
			RT11	ULTRIX	
BOO$RESELECT	00030C8F-R	BOOTDRIVR			
BOO$SELECT	00030C8F-R	BOOTDRIVR	DIRECTORY	RMS	
BOOT	0001C83C-R	KERNEL	ENTRY		
BOOT$A_CCA	0001A68D-R	KERNEL			
BOOT$L_ADP	0001A669-R	KERNEL	CONFIG		
BOOT$L_CSR	0001A66D-R	KERNEL	CONFIG		
BOOT$L_DEVTYP	0001A681-R	KERNEL	CONFIG		
BOOT$L_R1	0001A675-R	KERNEL	CONFIG		
BOOT$L_R2	0001A679-R	KERNEL	CONFIG		
BOOT$L_R5	0001A67D-R	KERNEL			
BOOT$L_SCORPIO_R3	0001A685-R	KERNEL			
BOOT$L_UNIT	0001A671-R	KERNEL	CONFIG		
BPTCODE	00000003	DEBUG			
BPTMAX	0000000F	DEBUG	EXCEPT		
BREAKUP_FILE	0002A251-R	PARSE	DIRECTORY	LOAD	RMS
BUG$CHECK	0002C19E-R	QIO	FRKCTL	IOPOST	IOSNPG
			IOSRAM	LODMAP	MBAINT
			MEMALC	ONLY730	QIOREQ
CDB$BLOCK	0001AA20-R	CHAN730			
CHR$MCHK	000284D4-R	MCHK730	EXCEPT		
CLI_ACTION	0002000C-R	CLI	APT		
CLK$K_TCR0	00000000		WK-SCB		
CLK$RE_INIT	00021098-R	CLOCK	CSDDBTDRV	KERNEL	SCB
			SCRIPT		
CMK$APBOOT	0001E118-R	APBOOT	CHMK	CLI	
CMK$ASTEXIT	0001FE94-R	CHMK	ASTDEL		
CMK$CANTIM	00020E12-R	CLOCK	CHMK		
CMK$HIBER	00020DE4-R	CLOCK	CHMK		
CMK$INITSCB	0002D49A-R	SCB	CHMK		
CMK$MMENABLE	00028F33-R	MEMMGT	CHMK		
CMK$RCONSOLE	00021DA2-R	CONSOLE	CHMK	KERNEL	ONLY730
CMK$REFRESH	0001D1B7-R	KERNEL	CHMK		
CMK$SETIMR	00020EB0-R	CLOCK	CHMK		
CMK$SETIPL	0002D6DA-R	SCB	CHMK		
CMK$SETPRT	00028F72-R	MEMMGT	CHMK		
CMK$SGIPR	00023AC4-R	DEBUG	CHMK	CLI	KERNEL
			SHOWMEMORY		
CMK$TCONSOLE	00021F5E-R	CONSOLE	CHMK	ONLY730	
COM$DELATTNAST	000310A0-R	COMDRVSUB	ENTRY		
COM$DRVDEALMEM	00031113-R	COMDRVSUB	ENTRY		
COM$FLUSHATTNS	000310D3-R	COMDRVSUB	ENTRY		
COM$POST	00031103-R	COMDRVSUB	ENTRY		
COM$SETATTNAST	00031138-R	COMDRVSUB	ENTRY		
COMMAND_TREE	0001151A-R	CLI	APT		
COND_DEBUG	000234F5-R	DEBUG	EXCEPT	KERNEL	
COND_HANDLR	00025CC8-R	EXCEPT	KERNEL		
CONSOLEAAA_FLAGS	00000000		WK-QIO		

Symbol	Value	Defined By	Referenced By ...		
CRD$EXPREG	00028C72-R	MEMMGT	CRDIMAGE		
CRD$GL_CRD_COMMAND_DEBUG	0001B09C-R	DSVECS	CLI		
CRD$K_CRD_INITIALIZATION	00000000	KERNEL			
CRD$K_DS_CONTROL_C	00000001	KERNEL			
CRD$K_DS_FLAGS	00000000	KERNEL			
CRD$K_HOOK_POINT	00000000	KERNEL			
CRD$K_LAST_ACCESS_CODE	00000002	KERNEL			
CRD$K_LAST_DS_VARIABLE	00000002	KERNEL			
CRD$K_LAST_FUNCTION	00000002	KERNEL			
CRD$K_LAST_HOOK_POINT	00000001	KERNEL			
CRD$K_READ	00000000	KERNEL			
CRD$K_TYPECODE	00000001	KERNEL			
CRD$K_WRITE	00000001	KERNEL			
CTL$GL_CCB	0001A231-R	KERNEL			
CTL$GL_CCBBASE	0001A631-R	KERNEL	ASSIGN	IOPOST	IOSPGD
CTL$GW_CHINDX	000190C2-R	QIO	IOSPGD		
CTL$GW_NMIOCH	000190C0-R	QIO	IOSPGD		
CVT$T_MBAMAP	0001AA74-R	CHAN730	PRINT		
CVT$T_MBASR	0001AA70-R	CHAN730	PRINT		
CVT$T_UBADPR	0001AA78-R	CHAN730	PRINT		
CVT$T_UBAMAP	0001AA7C-R	CHAN730	PRINT		
DATETABLE	00031198-R	CVRTIM			
DBG_ADDR_RTN	0001B2D5-R	DEBUG			
DEBUG$GB_FLAGS	0001B108-R	DEBUG	CLI	EXCEPT	
DEBUG_HI	0001B2E1-R	DEBUG	MEMMGT	SHOWMEMORY	
DEBUG_LO	0001B2DD-R	DEBUG	MEMMGT	SHOWMEMORY	
DEBUG_THE_SUCKER	0001C3B9-R	PRINT	SCRIPT		
DEF$C_BKSTP_LEN	00000011	RMS	DIRECTORY		
DEF$C_ULTRIX_LEN	0000000F	RMS	DIRECTORY		
DEF$Q_BACKSTOP	0001918C-R	RMS	DIRECTORY		
DEF$Q_DEV	0001BE8B-R	ONLY730	DIRECTORY	KERNEL	LOAD
			RMS		
DEF$Q_DIR	0001BEB1-R	ONLY730	DIRECTORY	KERNEL	LOAD
			RMS		
DEF$Q_ULTRIX	0001919D-R	RMS	DIRECTORY		
DEV$M_DMT	00200000	SYS$IODEF	DASSGN		
DEV$M_FOR	01000000	SYS$IODEF	DASSGN		
DEV$M_MNT	00080000	SYS$IODEF	DASSGN		
DEV$M_RCK	40000000	SYS$IODEF	DASSGN		
DEV$M_SWL	02000000	SYS$IODEF	DASSGN		
DEV$M_WCK	80000000	SYS$IODEF	DASSGN		
DEV$V_ALL	00000017	SYS$IODEF	DASSGN		
DEV$V_DMT	00000015	SYS$IODEF	DASSGN		
DEV$V_FOD	0000000E	SYS$IODEF	ASSIGN	QIOREQ	
DEV$V_FOR	00000018	SYS$IODEF	DASSGN	QIOREQ	
DEV$V_MBX	00000014	SYS$IODEF	ASSIGN		
DEV$V_MNT	00000013	SYS$IODEF	QIOREQ		
DEV$V_NET	0000000D	SYS$IODEF	ASSIGN		
DEV$V_SHR	00000010	SYS$IODEF	ASSIGN		
DEV$V_SPL	00000006	SYS$IODEF	ASSIGN	QIOREQ	
DEV$V_SQD	00000005	SYS$IODEF	IOPOST		
DEVREV$T	0001BF90-R	PRINT	ERROR	START	

Symbol	Value	Defined By	Referenced By ...		
DEVREV$T_X	0001C160-R	PRINT	START		
DIAG_FILE_NAME	0001BDD1-R	LOAD	DEBUG		
DIR_FLAGS	0001B6C8-R	DIRECTORY			
DIR_V_CMM	00000000	DIRECTORY			
DIR_V_DRVCLR	00000001	DIRECTORY			
DISPATCH	00024C50-R	DISPAT	START		
DR$INITIAL	000317FB-R	DRINT	QIO		
DR$INT	000317E4-R	DRINT	QIO		
DS$AA_BPTADDR	0001B11C-R	DEBUG	ERROR	EXCEPT	LOOP
DS$AA_BPTBASE	0001B15C-R	DEBUG			
DS$ABORT	00010020-R	ENTRY	EXCEPT	PARAM	SCRIPT
DS$ABORTWAIT	00010070-R	ENTRY			
DS$AB_BPTINST	0001B109-R	DEBUG	ERROR	LOOP	
DS$AL_DS_DISPATCH_VECTORS	0001B0A0-R	DSVECS	CRDIMAGE		
DS$AQ_SSEND	000107FF-R	ENTRY	KERNEL		
DS$AQ_SYSSRV	00010200-R	ENTRY	KERNEL	MEMMGT	
DS$ASKADR	00010090-R	ENTRY			
DS$ASKDATA	00010080-R	ENTRY			
DS$ASKLGCL	00010098-R	ENTRY			
DS$ASKSTR	000100A0-R	ENTRY			
DS$ASKVLD	00010088-R	ENTRY			
DS$ATTACH	000101A8-R	ENTRY	DSVECS		
DS$AX_ARB	00032A74-R	ONLY730	KERNEL	SHOWMEMORY	
DS$AX_JIB	00032A00-R	ONLY730	KERNEL	SHOWMEMORY	
DS$AX_SOFTPCB	00032AEC-R	ONLY730	ASSIGN	ASTCON	CANCEL
			CHMK	DASSGN	DEVALC
			EVENT	KERNEL	QIOREQ
			SHOWMEMORY		
DS$A_PRGBGN	00000200	KERNEL	APT	CLI	LOAD
			MEMMGT	SHOWMEMORY	
DS$BGNSUB	00010030-R	ENTRY			
DS$BRANCH	000100A8-R	ENTRY			
DS$BREAK	00010058-R	ENTRY	QIO		
DS$CANWAIT	00010070-R	ENTRY	CLOCK		
DS$CHANNEL	00010180-R	ENTRY	CONFIG	PROBE	
DS$CKLOOP	00010040-R	ENTRY			
DS$CLI	0001FECF-R	CLI	DEBUG	EXCEPT	KERNEL
			SCRIPT		
DS$CLRVEC	00010168-R	ENTRY	SCRIPT	START	
DS$CNTRLC	00010078-R	ENTRY	ATTACH	CRDIMAGE	DIRECTORY
			DISPAT	KERNEL	QAMAIN
DS$CVTREG	000100B0-R	ENTRY	CHAN730	DEBUG	EXCEPT
			FLAGS		
DS$DOSUMMARY	00010028-R	ENTRY			
DS$ENDPASS	00010010-R	ENTRY			
DS$ENDSUB	00010038-R	ENTRY			
DS$ENTRY	0002F5C1-R	VER730			
DS$ERRDEV	000100C8-R	ENTRY			
DS$ERRHARD	000100D0-R	ENTRY			
DS$ERRPREP	00010108-R	ENTRY			
DS$ERRSOFT	000100D8-R	ENTRY			
DS$ERRSYS	000100C0-R	ENTRY			

Symbol	Value	Defined By	Referenced By ...		
DS$ESCAPE	00010050-R	ENTRY			
DS$FAB_INPUT	00019E70-R	KERNEL			
DS$FAB_OUTPUT	00019F04-R	KERNEL			
DS$FREEDBGSYM	000101B8-R	ENTRY			
DS$GA_ALLOCATED	0001B9C8-R	IOBASE			
DS$GA_ALLOCATES	0001B7B0-R	IOBASE	ATTACH		
DS$GA_BREAKVEC	0001C425-R	SCB	EXCEPT		
DS$GA_BUFPTR	0001A6F1-R	KERNEL			
DS$GA_CHKLPPC	0001A6D1-R	KERNEL	DISPAT	ERROR	LOOP
DS$GA_CHMKVEC	0001C421-R	SCB	CHMK		
DS$GA_CRD_DS_INTERFACE	0001B0A4-R	DSVECS	CLI	KERNEL	PRINT
			SCRIPT		
DS$GA_DS_CTRL_C_FIRST	00019F98-R	KERNEL	CLI	CRDIMAGE	DSVECS
			QAMAIN		
DS$GA_DS_CTRL_C_SECOND	00019F9C-R	KERNEL	CLI	CRDIMAGE	DSVECS
DS$GA_LASTADR	0001A699-R	KERNEL	CRDIMAGE	MEMALC	MEMMGT
			SHOWMEMORY		
DS$GA_LOOPADR	0001A6D5-R	KERNEL	LOOP		
DS$GA_PTABLE	0001BBCC-R	IOBASE	ATTACH	DEVICE	DSVECS
			KERNEL		
DS$GA_PTDESC	00014E64-R	IOBASE	ATTACH	DSVECS	
DS$GA_SELECTED	0001B7C4-R	IOBASE	CONFIG	DEVICE	
DS$GA_SELECTS	0001B79C-R	IOBASE	ATTACH	DEVICE	KERNEL
DS$GA_SOFTINT	0001C431-R	SCB	CLOCK		
DS$GA_SUPPORT_TABLE	00019374-R	RMS			
DS$GA_TBITVEC	0001C429-R	SCB	EXCEPT		
DS$GA_USER_ERROR_TEXT	0001B0BC-R	DSVECS	ERROR		
DS$GB_BYTEBUF	0001A6F5-R	KERNEL			
DS$GB_CI_INIT	00000000		WK-RMS		
DS$GB_INHIBIT_NAMING	0001A6F7-R	KERNEL	ATTACH	CLI	
DS$GB_KDB50_INIT	00000000		WK-RMS		
DS$GB_MM_ENB	0001BE60-R	MEMMGT	CLI	DISPAT	KERNEL
DS$GB_PB_INIT	00000000		WK-RMS		
DS$GB_PRIMARY_CPU_IDENTIFIER	00018630-R	ONLY730	CLOCK	DEBUG	EXCEPT
DS$GB_QIOACT	0001C548-R	SCRIPT	PRINT		
DS$GB_STOPPING_THE_TIMER	0001C420-R	SCB	CLOCK		
DS$GB_SYSTYPE	0001A210-R	KERNEL	CONSOLE	DIRECTORY	MEMMGT
			RMS		
DS$GB_TYPECODE	0001A6F6-R	KERNEL	DISPAT	ERROR	EXCEPT
			MCHK730	PRINT	SHOWMEMORY
			START		
DS$GB_UDA50_INIT	000302E4-R	PUBTDRIVR	WK-RMS		
DS$GB_VDS_DEBUG	0001C3B9-R	PRINT			
DS$GB_WIDTH	0001B00E-R	CONSOLE	PRINT	QAMAIN	
DS$GETADDRESS	00010090-R	ENTRY			
DS$GETBUF	00010120-R	ENTRY			
DS$GETDATA	00010080-R	ENTRY			
DS$GETLOGICAL	00010098-R	ENTRY			
DS$GETSTRING	000100A0-R	ENTRY			
DS$GETTERM	000101C8-R	ENTRY			
DS$GETVIELD	00010088-R	ENTRY			
DS$GK_SID	03000000	ONLY730	ERROR	KERNEL	PARAM

Symbol	Value	Defined By	Referenced By ...		
			QIO	START	
DS$GK_SUPPORT_TABLE_ROWS	00000024	RMS			
DS$GK_USOVR	00000200	ONLY730	CLOCK		
DS$GL_ACTUAL_MEMSIZE	0001BE58-R	MEMMGT			
DS$GL_BUFCNT	0001AA08-R	BUFFER	KERNEL	MEMMGT	SHOWMEMORY
DS$GL_BUFLEN	0001A6ED-R	KERNEL	PARAM	PRINT	
DS$GL_CLIBASE	0001AB1C-R	CLI	APT	ATTACH	CONFIG
			DEBUG	DIRECTORY	DISPAT
			KERNEL	LOAD	PARSE
			QACHECKS	QAMAIN	SCRIPT
			START		
DS$GL_CRD_DEBUG	0001B0D0-R	DSVECS	CLI	CRDIMAGE	
DS$GL_CRD_FLAGS	0001B098-R	DSVECS	ATTACH	CONSOLE	CRDIMAGE
DS$GL_DEVERR_COUNT	0001A6B5-R	KERNEL	DISPAT	ERROR	
DS$GL_EFC0	0001A691-R	KERNEL	FLAGS		
DS$GL_EFC1	0001A695-R	KERNEL			
DS$GL_ERRCNT	0001A6A5-R	KERNEL	DISPAT	ERROR	LOOP
DS$GL_ERRSUP_COUNT	0001A6BD-R	KERNEL	DISPAT	ERROR	
DS$GL_FLAGS	0001A69D-R	KERNEL	APT	ATTACH	CALLFRAME
			CLI	CLOCK	CONSOLE
			CRDIMAGE	CSDDBTDRV	DISPAT
			DSLOAD	DSVECS	ERROR
			EXCEPT	LOAD	LOOP
			MEMMGT	PRINT	QIO
			SCB	SCRIPT	SHOWMEMORY
			START		
DS$GL_FSTTEST	0001A6C5-R	KERNEL	DISPAT	QACHECKS	QAMAIN
			START		
DS$GL_HARDERR_COUNT	0001A6A9-R	KERNEL	DISPAT	ERROR	
DS$GL_IPR_VALUE	0001C550-R	SHOWMEMORY	DEBUG		
DS$GL_LOOKASIDE	0001BE38-R	MEMALC	SHOWMEMORY		
DS$GL_LSTTEST	0001A6C9-R	KERNEL	DISPAT	LOOP	QACHECKS
			QAMAIN	START	
DS$GL_MEMSIZE	0001BE54-R	MEMMGT	DEBUG	MEMALC	SHOWMEMORY
DS$GL_NUMTEST	0001A6C1-R	KERNEL	START		
DS$GL_PAGE	0001C16D-R	PRINT			
DS$GL_PCBVEC	0001A635-R	KERNEL	ENTRY		
DS$GL_PREPERR_COUNT	0001A6AD-R	KERNEL	ERROR		
DS$GL_RADIX	0001A6E5-R	KERNEL	DEBUG	PARSE	
DS$GL_RUNTIM	0001A6D9-R	KERNEL			
DS$GL_SET_MEMSIZE	0001BE5C-R	MEMMGT	SHOWMEMORY		
DS$GL_SIZE	0001A6E9-R	KERNEL			
DS$GL_SOFTERR_COUNT	0001A6B1-R	KERNEL	DISPAT	ERROR	
DS$GL_SUBTEST	0001A6CD-R	KERNEL	DISPAT	LOOP	QACHECKS
			START		
DS$GL_SYSERR_COUNT	0001A6B9-R	KERNEL	DISPAT	ERROR	
DS$GL_TERMCHAR	0001A987-R	KERNEL	CONSOLE		
DS$GL_TIMESEC	0001A6DD-R	KERNEL			
DS$GL_WAITIM	0001A6E1-R	KERNEL			
DS$GPHARD	00010018-R	ENTRY	CHAN730	CLI	DEVICE
			ERROR	PRINT	QIO
			START		

Symbol	Value	Defined By	Referenced By ...		
DS$GQ_BUFQWD	0001A6ED-R	KERNEL			
DS$GQ_CURYEAR	0001AFA4-R	CLOCK	KERNEL		
DS$GQ_DAYTIM	00019C80-R	VER730	KERNEL		
DS$GQ_EFL	0001A691-R	KERNEL	EVENT		
DS$GQ_PHYADR	0001AA00-R	BUFFER	MEMMGT		
DS$GQ_TESTID	0001B71C-R	ERROR	START		
DS$GT_ASCIBUF	0001A6F8-R	KERNEL			
DS$GT_BUFFER	0001A707-R	KERNEL	PRINT	SCRIPT	
DS$GT_COPYEAR	00019E32-R	KERNEL			
DS$GT_COPYRIGHT	00019E04-R	KERNEL			
DS$GT_LEGAL	00010BAE-R	KERNEL			
DS$GT_LEGAL_END	00019E51-R	KERNEL			
DS$GT_NEWYEAR	0001AFAC-R	CLOCK	KERNEL		
DS$GT_PROMPT	00019CDF-R	VER730	APT	CLI	KERNEL
			SCRIPT		
DS$GT_START	00019C9F-R	VER730	KERNEL		
DS$GT_STRBUF	0001A907-R	KERNEL	FLAGS	PARAM	
DS$GT_VALPHA	0002A04E-R	PARSE			
DS$GT_VALPHANUM	0002A060-R	PARSE			
DS$GT_VDS_VERSION	00019CD5-R	VER730	ERROR		
DS$GT_VFILE	0002A072-R	PARSE			
DS$GT_VHEX	0002A0D3-R	PARSE			
DS$GT_VSPACE	0002A084-R	PARSE			
DS$GT_VSYMBOL	0002A03C-R	PARSE			
DS$GW_TTIN	0001A6A1-R	KERNEL	PRINT	SCRIPT	
DS$GW_TTOUT	0001A6A3-R	KERNEL	CONSOLE	ERROR	EXCEPT
			PRINT	SCRIPT	
DS$HELP	000101B0-R	ENTRY			
DS$INITSCB	00010170-R	ENTRY	DISPAT	KERNEL	
DS$INLOOP	00010048-R	ENTRY			
DS$K_ASCIBUF	0000000F	KERNEL			
DS$K_BBC	0000001D	DISPAT			
DS$K_BBCC	00000021	DISPAT			
DS$K_BBCCI	00000023	DISPAT			
DS$K_BBCS	0000001F	DISPAT			
DS$K_BBS	0000001C	DISPAT			
DS$K_BBSC	00000020	DISPAT			
DS$K_BBSS	0000001E	DISPAT			
DS$K_BBSSI	00000022	DISPAT			
DS$K_BCC_M	0000001B	DISPAT			
DS$K_BCS_M	0000001A	DISPAT			
DS$K_BEQLU_B	00000005	DISPAT			
DS$K_BEQLU_M	00000011	DISPAT			
DS$K_BEQL_B	00000004	DISPAT			
DS$K_BEQL_M	00000010	DISPAT			
DS$K_BERROR	00000026	DISPAT			
DS$K_BGEQU_B	00000007	DISPAT			
DS$K_BGEQU_M	00000013	DISPAT			
DS$K_BGEQ_B	00000006	DISPAT			
DS$K_BGEQ_M	00000012	DISPAT			
DS$K_BGTRU_B	00000009	DISPAT			
DS$K_BGTRU_M	00000015	DISPAT			

Symbol	Value	Defined By	Referenced By ...		
DS$K_BGTR_B	00000008	DISPAT			
DS$K_BGTR_M	00000014	DISPAT			
DS$K_BLBC	00000025	DISPAT			
DS$K_BLBS	00000024	DISPAT			
DS$K_BLEQU_B	00000003	DISPAT			
DS$K_BLEQU_M	0000000F	DISPAT			
DS$K_BLEQ_B	00000002	DISPAT			
DS$K_BLEQ_M	0000000E	DISPAT			
DS$K_BLSSU_B	00000001	DISPAT			
DS$K_BLSSU_M	0000000D	DISPAT			
DS$K_BLSS_B	00000000	DISPAT			
DS$K_BLSS_M	0000000C	DISPAT			
DS$K_BNEQU_B	0000000B	DISPAT			
DS$K_BNEQU_M	00000017	DISPAT			
DS$K_BNEQ_B	0000000A	DISPAT			
DS$K_BNEQ_M	00000016	DISPAT			
DS$K_BNERROR	00000027	DISPAT			
DS$K_BUFSIZ	00000200	KERNEL	PRINT	SCRIPT	
DS$K_BVC_M	00000019	DISPAT			
DS$K_BVS_M	00000018	DISPAT			
DS$K_CCB	00000040	KERNEL	QIO		
DS$K_NYSIZ	00000014	CLOCK	KERNEL		
DS$K_PRGSIZ	0000F800	KERNEL	APT	LOAD	MEMMGT
DS$K_SSSIZE	00000600	ENTRY	KERNEL		
DS$K_STRBUF	00000080	KERNEL	FLAGS	PARAM	
DS$LOAD	00010198-R	ENTRY	CRDIMAGE	QIO	SCRIPT
DS$LOADPCS	000101C0-R	ENTRY	KERNEL		
DS$L_USERCNTRLC	00019FA0-R	KERNEL	CLI	CRDIMAGE	
DS$MAPDBGBLOCK	00010118-R	ENTRY			
DS$MEMSIZE	000101D8-R	ENTRY			
DS$MMOFF	00010158-R	ENTRY	CRDIMAGE		
DS$MMON	00010150-R	ENTRY			
DS$M_TODR_ACTIVE	00000000		WK-QIO		
DS$PARSE	000100B8-R	ENTRY			
DS$PRINTB	000100E0-R	ENTRY	CHAN730		
DS$PRINTF	000100F0-R	ENTRY	APBOOT	ATTACH	DEBUG
			DEVICE	DIRECTORY	DSLOAD
			HELP	QACHECKS	QAFLAGS
			QAMAIN	SHOWMEMORY	START
DS$PRINTS	000100F8-R	ENTRY			
DS$PRINTSIG	00010100-R	ENTRY			
DS$PRINTX	000100E8-R	ENTRY			
DS$PROBE	000101A0-R	ENTRY			
DS$RAB_INPUT	00019EC0-R	KERNEL	SCRIPT		
DS$RAB_OUTPUT	00019F54-R	KERNEL	PRINT	SCRIPT	
DS$RELBUF	00010128-R	ENTRY			
DS$SERVICE_DISPATCHER	000101D0-R	ENTRY	CVRTIM	DEBUG	EXCEPT
			MEMALC		
DS$SETIPL	00010178-R	ENTRY			
DS$SETMAP	00010188-R	ENTRY			
DS$SETPRIEXV	00010110-R	ENTRY			
DS$SETVEC	00010160-R	ENTRY			

Symbol	Value	Defined By	Referenced By ...		
DS$SHOCHAN	00010190-R	ENTRY			
DS$SHOWCHAN	00010190-R	ENTRY			
DS$SOFTIPL5	0001CB00-R	KERNEL	SCB		
DS$SRVDISP_TABLE	00013B08-R	ENTRY			
DS$SUMMARY	00010028-R	ENTRY	DISPAT	LOOP	
DS$USERMODE	0001CB0B-R	KERNEL	VER730		
DS$WAITMS	00010060-R	ENTRY			
DS$WAITUS	00010068-R	ENTRY	CLOCK	CONFIG	ONLY730
DSA$M_DFLTFLGS	00003000	KERNEL	FLAGS		
DSI$ABORT	0002E7C2-R	START	KERNEL		
DSI$CHANNEL_VEC	0001FD70-R	CHAN730	SCB		
DSI$TIMER	00020FA8-R	CLOCK	SCB		
DSI$TIMSRV	00020F24-R	CLOCK	SCB		
DSP$CONFIG_ADP	00029598-R	ONLY730	CONFIG		
DSP$CONFIG_LOAD	00021131-R	CONFIG			
DSP$GEN_PTABLES	00023CB1-R	DEVICE	START		
DSP$REMOVEBREAK	00023441-R	DEBUG			
DSP$SHOWMBA	0001FC71-R	CHAN730			
DSP$SHOWUBI	0001FC7B-R	CHAN730			
DSR$ALLOCATE_PSEUDO_RPB	0002C628-R	RMS	CONFIG	DIRECTORY	
DSR$CHECKLOAD	0002DD88-R	SHOWMEMORY	CALLFRAME	DISPAT	MEMMGT
DSR$CHECK_AUTOTEST_OFF_SET	0001D74F-R	KERNEL	CRDIMAGE	DSVECS	ERROR
DSR$CHECK_MENUTEST_OFF_SET	0001D72B-R	KERNEL	CRDIMAGE	DSVECS	ERROR
DSR$COMPLETION	00025B46-R	ERROR	ATTACH	CLI	DEBUG
			DIRECTORY	LOAD	QIO
			SCRIPT		
DSR$DEALLOCATE_PSEUDO_RPB	0002C6A3-R	RMS	CONFIG	DIRECTORY	
DSR$FAULT_CLEAR	0002853B-R	MCHK730	EXCEPT	SCB	
DSR$FIND_USER_PC	0001F9BF-R	CALLFRAME	DISPAT	EXCEPT	START
DSR$IFLOADDEV	000280FD-R	LOAD	ATTACH		
DSR$KEY_EQUAL	00027A60-R	HELP	DIRECTORY		
DSR$LOAD_CRD	00022264-R	CRDIMAGE	CLI	KERNEL	
DSR$MMENABLE	00028F20-R	MEMMGT	DISPAT	SHOWMEMORY	
DSR$ONLY_CONSOLE_CONFIG	00029843-R	ONLY730	CONFIG		
DSR$PRINT_TYPECODE_NAME	00022B20-R	CRDIMAGE			
DSR$RESTORE_CRD_VECTORS	00022636-R	CRDIMAGE	KERNEL		
DSR$SETIMR_INI	00020E4A-R	CLOCK	KERNEL	MEMMGT	
DSR$UNLOAD_CRD	00022546-R	CRDIMAGE	CLI	KERNEL	
DSV$ATTACH	0001E4D4-R	ATTACH	CLI		
DSV$DEATTACH	0001F114-R	ATTACH	CLI		
DSV$DEBUG	00023B09-R	DEBUG	CLI	START	
DSV$DESELECT	0001F3BC-R	ATTACH	CLI		
DSV$DIRECTORY	00024C24-R	DIRECTORY	CLI		
DSV$EXIT	0001D62B-R	KERNEL	CLI	CRDIMAGE	DSVECS
DSV$INITPCS	0002A38A-R	PCSLOAD	CLI		
DSV$LOAD	00028128-R	LOAD	CLI		
DSV$RUN	00028127-R	LOAD	CLI		
DSV$SELECT	0001F1DC-R	ATTACH	CLI		
DSV$SETLOAD	00027FA6-R	LOAD	CLI	CONFIG	KERNEL
DSV$SETPAGE	0002A39A-R	PRINT	CLI		
DSV$SHOWDEVICE	0001F082-R	ATTACH	CLI	QAMAIN	
DSV$SHOWDEVICEB	0001F07B-R	ATTACH	CLI		

Symbol	Value	Defined By	Referenced By ...		
DSV$SHOWLOAD	000280DD-R	LOAD	CLI		
DSV$SHOWMEMORY	0002DDA7-R	SHOWMEMORY	CLI		
DSV$SHOWPAGE	0002A3C1-R	PRINT	CLI		
DSV$SHOWSELECT	0001F0E3-R	ATTACH	CLI	QAMAIN	
DSV$SHOWSELECTB	0001F0DC-R	ATTACH	CLI		
DSV$SHOWTESTS	0002E91D-R	START	CLI		
DSX$ABORT	0002E798-R	START	ENTRY	QACHECKS	QAMAIN
DSX$ATTACH	0001E5DE-R	ATTACH	ENTRY		
DSX$BGNSUB	0002825D-R	LOOP	ENTRY		
DSX$BOOTATTACHED	0001C806-R	MPDUMMY	ENTRY		
DSX$BRANCH	0002B44C-R	QAMAIN	ENTRY		
DSX$CANWAIT	00020C8C-R	CLOCK	ENTRY		
DSX$CHANNEL	0001F9F0-R	CHAN730	ENTRY		
DSX$CHECK_SUPPORT	0002C6B6-R	RMS			
DSX$CKLOOP	0002846C-R	LOOP	ENTRY		
DSX$CLRVEC	0002D61A-R	SCB	ENTRY		
DSX$CNTRLC	0001D600-R	KERNEL	ENTRY		
DSX$CVTREG	0002ACB8-R	PRINT	ENTRY		
DSX$DIRECTORY	000248EF-R	DIRECTORY			
DSX$DOSUMMARY	000250F7-R	DISPAT	ENTRY		
DSX$ENDPASS	00024F08-R	DISPAT	ENTRY		
DSX$ENDSUB	000282FB-R	LOOP	ENTRY		
DSX$ERRDEV	0002560F-R	ERROR	ENTRY		
DSX$ERRHARD	000255C1-R	ERROR	ENTRY		
DSX$ERRPREP	000255E8-R	ERROR	ENTRY		
DSX$ERRSOFT	0002559A-R	ERROR	ENTRY		
DSX$ERRSYS	0002563D-R	ERROR	ENTRY		
DSX$ESCAPE	00028248-R	LOOP	ENTRY		
DSX$FREEDBGSYM	000290EE-R	MEMMGT	ENTRY		
DSX$GETADDRESS	000299B2-R	PARAM	ENTRY		
DSX$GETBUF	0001F80B-R	BUFFER	ENTRY		
DSX$GETDATA	000298BC-R	PARAM	ENTRY		
DSX$GETLOGICAL	00029A0B-R	PARAM	ENTRY		
DSX$GETSTRING	00029B2B-R	PARAM	ENTRY		
DSX$GETSYSBUF	0001F800-R	BUFFER	WK-ENTRY		
DSX$GETTERM	0001D76E-R	KERNEL	ENTRY		
DSX$GETVIELD	00029941-R	PARAM	ENTRY		
DSX$GPHARD	00023C7F-R	DEVICE	ENTRY		
DSX$HALTATTACHED	0001C812-R	MPDUMMY	ENTRY		
DSX$HELP	00027228-R	HELP	CLI	ENTRY	
DSX$INITSCB	0002D474-R	SCB	ENTRY		
DSX$INLOOP	000284AC-R	LOOP	ENTRY		
DSX$LOAD	00025250-R	DSLOAD	DEBUG	ENTRY	LOAD
DSX$LOADPCS	0002A380-R	PCSLOAD	ENTRY		
DSX$MAPDBGBLOCK	000290A0-R	MEMMGT	ENTRY		
DSX$MEMSIZE	0002881F-R	MEMALC	ENTRY		
DSX$MMOFF	00028EFA-R	MEMMGT	CLI	ENTRY	
DSX$MMON	00028EE0-R	MEMMGT	CLI	ENTRY	
DSX$PARSE	00029E37-R	PARSE	ENTRY		
DSX$PPSCB	0001E126-R	APBOOT	ENTRY		
DSX$PRINT	0002A66D-R	PRINT	ATTACH	CALLFRAME	CLI
			CONSOLE	CRDIMAGE	DISPAT

Symbol	Value	Defined By	Referenced By ...		
			DSVECS	ERROR	FLAGS
			KERNEL	LOAD	LOOP
			MEMMGT	PARAM	QIO
			SCRIPT	START	
DSX$PRINTB	0002A6D5-R	PRINT	ENTRY	EXCEPT	
DSX$PRINTF	0002A6ED-R	PRINT	ENTRY	LOAD	
DSX$PRINTI	0002A702-R	PRINT	DEBUG	EXCEPT	KERNEL
			PCSLOAD		
DSX$PRINTREV	0002A83C-R	PRINT	ENTRY		
DSX$PRINTS	0002A6BB-R	PRINT	ENTRY		
DSX$PRINTSIG	00025E10-R	EXCEPT	ENTRY		
DSX$PRINTSIGI	00025E22-R	EXCEPT			
DSX$PRINTX	0002A6C8-R	PRINT	ENTRY	EXCEPT	
DSX$PROBE	000320EC-R	PROBE	ENTRY		
DSX$RELBUF	0001F87B-R	BUFFER	ENTRY		
DSX$RELSYSBUF	0001F873-R	BUFFER	WK-ENTRY		
DSX$SERVICE_DISPATCHER	00025561-R	ENTRY			
DSX$SETIPL	0002D6BE-R	SCB	ENTRY		
DSX$SETMAP	0001FAA1-R	CHAN730	ENTRY		
DSX$SETPRIEXV	000264FA-R	EXCEPT	ENTRY		
DSX$SETVEC	0002D53B-R	SCB	ENTRY		
DSX$SHOWCHAN	0001FC8C-R	CHAN730	ENTRY		
DSX$SHOWIDLE	0001C818-R	MPDUMMY	ENTRY		
DSX$STARTATTACHED	0001C80C-R	MPDUMMY	ENTRY		
DSX$SUPERPARSE	00029E30-R	PARSE	APT	CLI	
DSX$TYPE_OUT	0002A3E1-R	PRINT	SCRIPT		
DSX$WAITMS	00020CB5-R	CLOCK	ENTRY		
DSX$WAITUS	00020D40-R	CLOCK	ENTRY		
DSX$WAITUS_USOVR	00020D5B-R	CLOCK			
DSX$_CLEAR_INTERLOCK	0001C839-R	MPDUMMY	WK-ENTRY		
DSX$_INTERLOCK_TEST	0001C800-R	MPDUMMY	WK-ENTRY		
DSX$_SET_INTERLOCK	0001C836-R	MPDUMMY	WK-ENTRY		
DS_CLEANUP	00025022-R	DISPAT	CLI	CRDIMAGE	DIRECTORY
			KERNEL	LOAD	LOOP
			SHOWMEMORY	START	
DS_ERRSUP	00025A19-R	ERROR	APT	ASTDEL	ATTACH
			CHMK	CLI	CLOCK
			CONFIG	DEVICE	DISPAT
			ENTRY	EXCEPT	IOPOST
			KERNEL	LOOP	MEMALC
			MEMMGT	ONLY730	QIO
			SCRIPT		
DS_GETLINE	0002D959-R	SCRIPT	PARAM		
DS_SUPERLINE	0002D952-R	SCRIPT	ATTACH	CLI	
DS_TESTID	000259AD-R	ERROR	START		
DUE_TIME	0001C56C-R	START	KERNEL		
DUMP_SELF	00000000		WK-CLI		
DX20$B_DRIVE	00000032	IOBASE			
DX20$K_LEN	00000033	IOBASE			
DYN$C_ACB	00000002	QIO			
DYN$C_ADP	00000001	QIO			
DYN$C_AQB	00000003	QIO			

Symbol	Value	Defined By	Referenced By ...
DYN$C_BRDCST	0000001A	QIO	
DYN$C_BUFIO	00000013	QIO	
DYN$C_CEB	00000004	QIO	
DYN$C_CRB	00000005	QIO	
DYN$C_CXB	0000001B	QIO	
DYN$C_DDB	00000006	QIO	
DYN$C_DPT	0000001E	QIO	
DYN$C_EXTGSD	00000028	QIO	
DYN$C_FCB	00000007	QIO	
DYN$C_FRK	00000008	QIO	
DYN$C_GSD	00000015	QIO	
DYN$C_IDB	00000009	QIO	
DYN$C_IRP	0000000A	QIO	
DYN$C_IRPE	0000002C	QIO	
DYN$C_JIB	0000002F	QIO	
DYN$C_JPB	0000001F	QIO	
DYN$C_KFH	00000026	QIO	
DYN$C_KFI	00000018	QIO	
DYN$C_LOG	0000000B	QIO	
DYN$C_MBX	0000002B	QIO	
DYN$C_MTL	00000019	QIO	
DYN$C_MVL	00000016	QIO	
DYN$C_NDB	0000001C	QIO	
DYN$C_NET	00000017	QIO	
DYN$C_PBH	00000020	QIO	
DYN$C_PCB	0000000C	QIO	
DYN$C_PDB	00000021	QIO	
DYN$C_PFL	00000023	QIO	
DYN$C_PIB	00000022	QIO	
DYN$C_PQB	0000000D	QIO	
DYN$C_PTR	00000025	QIO	
DYN$C_RBM	00000031	QIO	
DYN$C_RVT	0000000E	QIO	
DYN$C_RVX	00000027	QIO	
DYN$C_SFT	00000024	QIO	
DYN$C_SHB	0000002A	QIO	
DYN$C_SHMCEB	0000002E	QIO	
DYN$C_SHMGSD	00000029	QIO	
DYN$C_SHRBUFIO	00000080	QIO	
DYN$C_SLAVCEB	0000002D	QIO	
DYN$C_SPECIAL	00000080	QIO	
DYN$C_SSB	0000001D	QIO	
DYN$C_TQE	0000000F	QIO	
DYN$C_TWP	00000030	QIO	
DYN$C_TYPAHD	00000014	QIO	
DYN$C_UCB	00000010	QIO	
DYN$C_VCA	00000032	QIO	
DYN$C_VCB	00000011	QIO	
DYN$C_WCB	00000012	QIO	
END_SIZE	0001B316-R	DIRECTORY	ULTRIX
ERL$DEVICERR	0002F5CB-R	VMSDUMMY	ENTRY
ERL$DEVICTMO	0002F5CB-R	VMSDUMMY	ENTRY

Symbol	Value	Defined By	Referenced By ...		
ERL$RELEASEMB	0002F5CB-R	VMSDUMMY	ENTRY	IOSNPG	
ESTKPTR	00035800-R	ONLY730	EXCEPT	KERNEL	MEMMGT
			SHOWMEMORY		
ET$INIT	00000000		WK-CONFIG		
EXCHANGE_DISPATCH_VECTORS	00022681-R	CRDIMAGE			
EXE$ABORTIO	00032526-R	QIOREQ	ACPFDT	COMDRVSUB	ENTRY
			QIOFDT		
EXE$ACVIOLAT	0002621C-R	EXCEPT			
EXE$ALLOC	0003168A-R	DEVALC	ENTRY		
EXE$ALLOCATE	0002861D-R	MEMALC			
EXE$ALLOCBUF	0002853F-R	MEMALC	COMDRVSUB	ENTRY	QIOREQ
EXE$ALLOCCEB	00028543-R	MEMALC			
EXE$ALLOCIRP	00028550-R	MEMALC	ENTRY	QIOREQ	
EXE$ALLOCJIB	00028547-R	MEMALC			
EXE$ALLOCPCB	00028554-R	MEMALC			
EXE$ALLOCPQB	0002855C-R	MEMALC			
EXE$ALLOCTQE	00028565-R	MEMALC	CLOCK		
EXE$ALONONPAGED	0002859F-R	MEMALC	ATTACH	CANCEL	CONFIG
			CRDIMAGE	DSVECS	ENTRY
			ODS	ONLY730	QAMAIN
			QIO	RMS	SCRIPT
			ULTRIX		
EXE$ALTQUEPKT	00032551-R	QIOREQ	ENTRY		
EXE$ASCTIM	000311FD-R	CVRTIM	ENTRY		
EXE$ASSIGN	00030E00-R	ASSIGN	ENTRY		
EXE$BINTIM	0003129E-R	CVRTIM	ENTRY		
EXE$BREAK	00026244-R	EXCEPT			
EXE$BUFFRQUOTA	0002F5CC-R	VMSDUMMY	ENTRY		
EXE$BUFQUOPRC	0002F5CC-R	VMSDUMMY	ENTRY		
EXE$CANCEL	00030F9B-R	CANCEL	ENTRY		
EXE$CANTIM	00020DFA-R	CLOCK	ENTRY		
EXE$CARRIAGE	00032343-R	QIOFDT	ENTRY		
EXE$CLREF	00025C0D-R	EVENT	ENTRY		
EXE$CNTREG	00028DAA-R	MEMMGT	CRDIMAGE	ENTRY	
EXE$DALLOC	00031726-R	DEVALC			
EXE$DASSGN	000315CA-R	DASSGN	ENTRY		
EXE$DEALLOCATE	000286CE-R	MEMALC			
EXE$DEANONPAGED	00028645-R	MEMALC	ASTDEL	ATTACH	CLOCK
			COMDRVSUB	CONFIG	CRDIMAGE
			DASSGN	DSVECS	ENTRY
			IOPOST	ONLY730	QAMAIN
			QIO	QIOFDT	RMS
			SCRIPT		
EXE$EXPREG	00028C8F-R	MEMMGT	ENTRY		
EXE$FAO	00026538-R	FAO	ENTRY		
EXE$FAOL	00026540-R	FAO	ENTRY		
EXE$FINISHIO	00032535-R	QIOREQ	ENTRY	IOSRAM	QIOFDT
EXE$FINISHIOC	00032533-R	QIOREQ	ACPFDT	ENTRY	
EXE$FORK	00031809-R	FRKCTL	COMDRVSUB	ENTRY	
EXE$FORKDSPTH	00031828-R	FRKCTL	QIO	SCB	
EXE$GB_CPUDATA	0002F93A-R	BOOTDRIVR			
EXE$GB_CPUTYPE	0002F939-R	BOOTDRIVR	CONFIG	CSDDBTDRV	DISPAT

Symbol	Value	Defined By	Referenced By ...		
			EXCEPT	KERNEL	SCB
EXE$GETTIM	0003186C-R	GETTIM	ENTRY		
EXE$GL_ABSTIM	00010810-R	ENTRY	CLOCK	IOSNPG	
EXE$GL_PWRDONE	0001908C-R	QIO			
EXE$GL_SCB	0001C42D-R	SCB			
EXE$GL_SPLITADR	0001BE34-R	MEMALC	SHOWMEMORY		
EXE$GL_TENUSEC	0002F93E-R	BOOTDRIVR	KERNEL	MUBTDRIVR	PUBTDRIVR
EXE$GL_UBDELAY	0002F942-R	BOOTDRIVR	KERNEL	MUBTDRIVR	PUBTDRIVR
EXE$GL_UMR_PHY	0002F948-R	BOOTDRIVR			
EXE$GL_UMR_VIR	0002F94C-R	BOOTDRIVR			
EXE$GQ_SYSTIME	00010B00-R	ENTRY	CLOCK	CVRTIM	GETTIM
			IOSNPG	KERNEL	
EXE$GTCHAN	00031888-R	GTCHAN	ENTRY		
EXE$HIBER	00020DC1-R	CLOCK	ENTRY		
EXE$INSERTIRP	000325A9-R	QIOREQ	ENTRY		
EXE$INSIOQ	0003258E-R	QIOREQ	ENTRY	IOPOST	
EXE$INSTIMQ	00020E69-R	CLOCK			
EXE$IOFORK	00031804-R	FRKCTL	ENTRY		
EXE$MAXACMODE	0002C18F-R	QIO	ASSIGN	COMDRVSUB	DEVALC
			ENTRY		
EXE$MODIFY	000321B8-R	QIOFDT	ENTRY		
EXE$MODIFYLOCK	000321F0-R	QIOFDT	ENTRY		
EXE$MODIFYLOCKR	000321F3-R	QIOFDT	ENTRY		
EXE$NUMTIM	00031494-R	CVRTIM	ENTRY		
EXE$ONEPARM	000321AC-R	QIOFDT	ENTRY		
EXE$PWRTIMCHK	0002C5B1-R	QIO	ENTRY		
EXE$QIO	0002BA88-R	QIO	ENTRY		
EXE$QIOACPPKT	00032573-R	QIOREQ	CANCEL		
EXE$QIODRVPKT	0003254D-R	QIOREQ	ACPFDT	ENTRY	QIOFDT
EXE$QIORETURN	00032585-R	QIOREQ	ENTRY		
EXE$READ	000321BE-R	QIOFDT	ENTRY		
EXE$READCHK	0003227A-R	QIOFDT	ENTRY		
EXE$READCHKR	0003228E-R	QIOFDT	ENTRY		
EXE$READEF	00025C3A-R	EVENT	ENTRY		
EXE$READLOCK	000321EA-R	QIOFDT	ACPFDT	ENTRY	
EXE$READLOCKR	000321FD-R	QIOFDT	ENTRY		
EXE$REFLECT	0002632F-R	EXCEPT			
EXE$ROPRAND	0002620C-R	EXCEPT			
EXE$SENSEMODE	0003231B-R	QIOFDT	ENTRY		
EXE$SETAST	0001E304-R	ASTCON	ENTRY	VMSDUMMY	
EXE$SETCHAR	000322ED-R	QIOFDT	ENTRY		
EXE$SETEF	00025BF3-R	EVENT	ENTRY		
EXE$SETIMR	00020E9B-R	CLOCK	ENTRY		
EXE$SETMODE	0003230B-R	QIOFDT	ENTRY		
EXE$SETPRT	00028F5D-R	MEMMGT	ENTRY		
EXE$SNDEVMSG	0002F5CB-R	VMSDUMMY	ENTRY		
EXE$TBIT	000262D8-R	EXCEPT			
EXE$UNWIND	00032678-R	UNWIND	ENTRY		
EXE$WAITFR	00025C5E-R	EVENT	ENTRY		
EXE$WAKE	00020CA7-R	CLOCK	ENTRY		
EXE$WFLAND	00025C9C-R	EVENT	ENTRY		
EXE$WFLOR	00025C7B-R	EVENT	ENTRY		

Symbol	Value	Defined By	Referenced By ...		
EXE$WRITE	000321C7-R	QIOFDT	ENTRY		
EXE$WRITECHK	00032280-R	QIOFDT	ENTRY		
EXE$WRITECHKR	000322B8-R	QIOFDT	ENTRY		
EXE$WRITELOCK	000321ED-R	QIOFDT	ACPFDT	ENTRY	
EXE$WRITELOCKR	00032204-R	QIOFDT	ENTRY		
EXE$ZEROPARM	000321B2-R	QIOFDT	ENTRY		
EXE_ACCVIO	0002621C-R	EXCEPT	SCB		
EXE_ARITH	00026238-R	EXCEPT	SCB		
EXE_BREAK	00026244-R	EXCEPT	SCB		
EXE_CHME	00026310-R	EXCEPT	SCB		
EXE_CHMK	00026308-R	EXCEPT			
EXE_CHMS	00026318-R	EXCEPT	SCB		
EXE_CHMU	00026320-R	EXCEPT	SCB		
EXE_CMODKRNL	0001FDD8-R	CHMK	SCB		
EXE_COMPAT	00026230-R	EXCEPT	SCB		
EXE_KRNLSTK	000261E4-R	EXCEPT	SCB		
EXE_MCHK	000261C4-R	EXCEPT	SCB		
EXE_OPCCUS	00026204-R	EXCEPT	SCB		
EXE_OPCDEC	000261FC-R	EXCEPT	SCB		
EXE_POWER	000261F0-R	EXCEPT	SCB		
EXE_RADRMOD	00026214-R	EXCEPT	SCB		
EXE_ROPRAND	0002620C-R	EXCEPT	SCB		
EXE_TBIT	000262D8-R	EXCEPT	SCB		
EXE_TRANSL	00026224-R	EXCEPT	SCB		
EXE_UNEXPINT	000262FE-R	EXCEPT	ONLY730	SCB	
FETCH_LONG	000296EA-R	ONLY730			
FETCH_STORE	000238CD-R	DEBUG	ONLY730		
FETCH_STOR_PREG	00023ADA-R	DEBUG			
FIL$CACHE_INIT	00026ADE-R	FILEREAD			
FIL$CACHE_TRUNC	00026B52-R	FILEREAD			
FIL$CVTFILNAM	00022C74-R	CVTFILNAM	FILEREAD	RMS	RT11
FIL$C_DIR_SIZE	00000024	FILEREAD			
FIL$C_SIZE	00000218	FILEREAD			
FIL$FINDFILID	00026CBD-R	FILEREAD			
FIL$GQ_CACHE	00000000		WK-FILEREAD		
FIL$GT_DDDEV	000180A7-R	LOAD	WK-FILEREAD		
FIL$GT_DDSTRING	000180A8-R	LOAD	FILEREAD		
FIL$GT_TOPSYS	00000000		WK-FILEREAD		
FIL$MOUNT	00026C29-R	FILEREAD			
FIL$OPENFILE	000269D8-R	FILEREAD	ODS		
FIL$RDCHKFILHDR	00026FC6-R	FILEREAD			
FIL$RDWRTLBN	0001F7AC-R	BOOTIO	FILEREAD		
FIL$READVBN	000270A5-R	FILEREAD	ODS		
FIL$STATBLK	0002715E-R	FILEREAD			
FIL$WRITEVBN	0002709E-R	FILEREAD			
FIND_USERPC	00026168-R	EXCEPT			
FIRMEXIST$T	0001C130-R	PRINT			
FMTBPT	00013198-R	DEBUG			
FMT_BER	00000000		WK-EXCEPT		
GET_BLOCK_LENGTH	0001E0E8-R	ANSI	ANSI		
GET_TIME	0001D4D5-R	KERNEL	START		
GT_MENU_ERROR	00011470-R	CLI	KERNEL		

Symbol	Value	Defined By	Referenced By ...		
HARDEXIST$T	0001C134-R	PRINT			
HDW_CLOCK_SET	000297BB-R	ONLY730	KERNEL		
HELPINFO	0001B76C-R	HELP	CLI		
HP$B_CPU_ID	0000004F	IOBASE			
HP$B_DHB32_NODE_ID	00000032	IOBASE			
HP$B_DMB32_NODE_ID	00000032	IOBASE			
HP$B_DX20_DRIVE	00000032	IOBASE			
HP$C_NAMELEN	00000014	ATTACH			
HP$K_DHB32_LEN	00000033	IOBASE			
HP$K_DMB32_LEN	00000033	IOBASE			
HP$K_DX20_LEN	00000033	IOBASE			
HP$K_TU70_LEN	00000032	IOBASE			
HP$K_TU72_LEN	00000032	IOBASE			
HP$L_KAA_SLOT	0000004B	IOBASE			
HPE$C_EPB_SIZE	0000001A	ATTACH	CONFIG	ONLY730	
HSC50_BOOT	00012297-R	CONFIG			
IMAGE_HEADER	0001A00C-R	KERNEL	DEBUG	DSLOAD	LOAD
IMAGE_HEADER_END	0001A20C-R	KERNEL	DSLOAD	LOAD	
INI$BRK	0001CAFE-R	KERNEL	CLI		
INI$LOAD_DEVICE	000210F0-R	CONFIG	KERNEL	SHOWMEMORY	
INI$PTABLE	0001F569-R	ATTACH	KERNEL	SHOWMEMORY	
INI$RDONLY	000291CB-R	MEMMGT			
INI$WRITABLE	000291CB-R	MEMMGT			
INIT_CONTEXT	0001D0C6-R	KERNEL	CLI	DIRECTORY	DISPAT
			LOAD	LOOP	SCRIPT
			START		
INS$PTABLE	0001F714-R	ATTACH	CONFIG	ONLY730	
IO$GQ_PHYSICAL	00018628-R	ONLY730	MEMMGT	PROBE	
IO$M_CANCTRLO	00000040	SYS$IODEF	EXCEPT		
IO$M_CTRLCAST	00000100	SYS$IODEF	KERNEL	PRINT	
IO$M_FCODE	0000003F	SYS$IODEF	QIOREQ		
IO$M_NOW	00000040	SYS$IODEF	KERNEL		
IO$M_TRMNOECHO	00001000	SYS$IODEF	PRINT		
IO$TESTCSR_L	00029741-R	ONLY730	QIO		
IO$TESTCSR_W	00029760-R	ONLY730	QIO		
IO$V_CANCTRLO	00000006	SYS$IODEF	CONSOLE		
IO$_LOGICAL	0000002F	SYS$IODEF	QIOREQ		
IO$_PHYSICAL	0000001F	SYS$IODEF	QIOREQ		
IO$_READLBLK	00000021	SYS$IODEF	QIOREQ		
IO$_READPROMPT	00000037	SYS$IODEF	CONSOLE	SCRIPT	
IO$_READVBLK	00000031	SYS$IODEF	KERNEL	PRINT	QIOREQ
			SCRIPT		
IO$_SENSEMODE	00000027	SYS$IODEF	CONSOLE		
IO$_SETMODE	00000023	SYS$IODEF	CONSOLE	KERNEL	PRINT
IO$_WRITEVBLK	00000030	SYS$IODEF	ERROR	EXCEPT	PRINT
			QIOREQ	SCRIPT	
IOC$ALLOSPT	00031C1F-R	IOSNPG	ENTRY		
IOC$ALOUBAMAP	00031B76-R	IOSNPG	ENTRY		
IOC$ALOUBAMAPN	00031B72-R	IOSNPG	ENTRY		
IOC$ALTUBAMAP	00031B54-R	IOSNPG	ENTRY		
IOC$AL_SYSUCB	00031910-R	IOBASE			
IOC$APPLYECC	00031DC0-R	IOSRAM	ENTRY		

Symbol	Value	Defined By	Referenced By ...		
IOC$CANCELIO	00031928-R	IOSNPG	ENTRY		
IOC$CVTLOGPHY	00031E34-R	IOSRAM	ACPFDT	IOPOST	
IOC$CVT_DEVNAM	00031C35-R	IOSPGD	DEVALC		
IOC$DIAGBUFILL	0003193F-R	IOSNPG	ENTRY		
IOC$DIRPOST1	00027EE6-R	IOPOST			
IOC$FFCHAN	00031C69-R	IOSPGD	ASSIGN		
IOC$FILSPT	00030F89-R	BUFCTL			
IOC$GETBYTE	00030F30-R	BUFCTL	ENTRY		
IOC$GL_ADPLIST	0003191C-R	IOBASE	ONLY730	QIO	
IOC$GL_DEVLIST	00031918-R	IOBASE	CLOCK	DEVALC	IOSPGD
			QIO		
IOC$GL_DPTLIST	00031920-R	IOBASE	QIO		
IOC$GL_IRPFL	0001BE3C-R	MEMALC	QIOREQ		
IOC$GL_PSBL	0001081C-R	ENTRY	CANCEL	COMDRVSUB	IOPOST
			IOSNPG	QIOREQ	
IOC$GL_PSFL	00010818-R	ENTRY	IOPOST		
IOC$INITBUFWIND	00030F52-R	BUFCTL	ENTRY		
IOC$INITDRV	000325C4-R	RELOCDRV	QIO		
IOC$INITIATE	00031A48-R	IOSNPG	ENTRY	QIOREQ	
IOC$IOPOST	00027BE8-R	IOPOST	QIO	SCB	
IOC$K_DIAGPID	00660000	KERNEL	CLOCK		
IOC$K_IOSPACE	40000000	ONLY730	CONFIG	DISPAT	
IOC$LOADMBAMAP	00031F38-R	LODMAP	ENTRY		
IOC$LOADUBAMAP	00031F99-R	LODMAP	ENTRY		
IOC$LOADUBAMAPA	00031F82-R	LODMAP	ENTRY		
IOC$MAPVBLK	00031E5B-R	IOSRAM	IOPOST		
IOC$MOVFRUSER	00030F59-R	BUFCTL	ENTRY		
IOC$MOVFRUSER1	00030F6A-R	BUFCTL	ENTRY		
IOC$MOVFRUSER2	00030F5D-R	BUFCTL	ENTRY		
IOC$MOVTOUSER	00030F71-R	BUFCTL	ENTRY		
IOC$MOVTOUSER1	00030F82-R	BUFCTL	ENTRY		
IOC$MOVTOUSER2	00030F75-R	BUFCTL	ENTRY		
IOC$PTETOPFN	00032015-R	LODMAP	ENTRY		IOSRAM
IOC$PURGDATAP	0002971E-R	ONLY730	ENTRY		
IOC$PUTBYTE	00030F41-R	BUFCTL	ENTRY		
IOC$QNXTSEG	00027D30-R	IOPOST			
IOC$QNXTSEG1	00027D3C-R	IOPOST			
IOC$REINITDRV	000325CA-R	RELOCDRV	QIO		
IOC$RELCHAN	0003197A-R	IOSNPG	ENTRY		
IOC$RELDATAP	00031A7B-R	IOSNPG	ENTRY		
IOC$RELMAPREG	00031B08-R	IOSNPG	ENTRY		
IOC$RELSCHAN	00031970-R	IOSNPG	ENTRY		
IOC$REQCOM	00031A05-R	IOSNPG	ENTRY		
IOC$REQDATAP	00031ACF-R	IOSNPG	ENTRY		
IOC$REQDATAPNW	00031ACB-R	IOSNPG	ENTRY		
IOC$REQMAPREG	00031B42-R	IOSNPG	ENTRY		
IOC$REQPCHANH	000319D1-R	IOSNPG	ENTRY		
IOC$REQPCHANL	000319DA-R	IOSNPG	ENTRY		
IOC$REQSCHANH	000319BD-R	IOSNPG	ENTRY		
IOC$REQSCHANL	000319C7-R	IOSNPG	ENTRY		
IOC$RETURN	00031BD8-R	IOSNPG	ENTRY		
IOC$RTN	00010930-R	ENTRY	QIO		

Symbol	Value	Defined By	Referenced By ...		
IOC$SEARCHDEV	00031C90-R	IOSPGD	ASSIGN	DEVALC	
IOC$SEARCHGEN	00031C95-R	IOSPGD	DEVALC		
IOC$SENSEDISK	00031F2A-R	IOSRAM	ENTRY		
IOC$UNLOCK	00031D87-R	IOSPGD	ASSIGN	DASSGN	DEVALC
IOC$UPDATRANSP	00031EEE-R	IOSRAM	ENTRY		
IOC$VERIFYCHAN	00031D8B-R	IOSPGD	CANCEL	DASSGN	GTCHAN
			QIOREQ		
IOC$WAKACP	00027D9A-R	IOPOST			
IOC$WFIKPCH	00031BD9-R	IOSNPG	ENTRY		
IOC$WFIRLCH	00031BFB-R	IOSNPG	ENTRY		
IOGEN$CNTRL_INI	0002C4DB-R	QIO			
IOGEN$CONN_VEC	0002977F-R	ONLY730	QIO		
ISTKPTR	00035200-R	ONLY730	EXCEPT	KERNEL	MEMMGT
			SHOWMEMORY		
KA$B_ACC_TYPE	00000042	IOBASE			
KA$B_NODE_ID	00000046	IOBASE			
KA$B_RESERVED	00000043	IOBASE			
KA$B_SID_TYPE	0000003D	IOBASE			
KA$K_LEN	00000050	IOBASE			
KA$L_FLAGS	00000032	IOBASE			
KA$L_MEM_SIZE	00000047	IOBASE			
KA$L_SID	0000003A	IOBASE			
KA$M_CHAR	00000100	IOBASE			
KA$M_CRC	00000010	IOBASE			
KA$M_D_FLOAT	00000002	IOBASE			
KA$M_EDIT	00000080	IOBASE			
KA$M_F_FLOAT	00000001	IOBASE			
KA$M_G_FLOAT	00000004	IOBASE			
KA$M_H_FLOAT	00000008	IOBASE			
KA$M_PACKED	00000020	IOBASE			
KA$M_PACK_MUL	00000040	IOBASE			
KA$M_SB_ERR	02000000	IOBASE			
KA$M_TODR	00010000	IOBASE			
KA$M_WCS_LD	01000000	IOBASE			
KA$V_CHAR	00000008	IOBASE			
KA$V_CRC	00000004	IOBASE			
KA$V_D_FLOAT	00000001	IOBASE			
KA$V_EDIT	00000007	IOBASE			
KA$V_F_FLOAT	00000000	IOBASE			
KA$V_G_FLOAT	00000002	IOBASE			
KA$V_H_FLOAT	00000003	IOBASE			
KA$V_PACKED	00000005	IOBASE			
KA$V_PACK_MUL	00000006	IOBASE			
KA$V_SB_ERR	00000019	IOBASE			
KA$V_TODR	00000010	IOBASE			
KA$V_WCS_LD	00000018	IOBASE			
KA$W_LATENCY	0000003E	IOBASE			
KA$W_MEM_SIZE	00000044	IOBASE			
KA$W_PROGRESS	00000040	IOBASE			
KA$W_RESERVED	00000038	IOBASE			
KA$W_WCS	00000036	IOBASE			
KB_CHECK	0001D2C4-R	KERNEL	ANSI	APT	ATTACH

Symbol	Value	Defined By	Referenced By ...		
			BUFFER	CLI	CLOCK
			DIRECTORY	DISPAT	ENTRY
			EVENT	LOOP	PRINT
			SCB		
KB_CHECK_APT	0001D322-R	KERNEL	CLI		
KB_POLL	00021D35-R	CONSOLE	CSDDBTDRV	KERNEL	
KDB50_RESPONSE	00000000		WK-CONFIG		
KSTKPTR	00034A00-R	ONLY730	KERNEL	MEMMGT	SHOWMEMORY
LIB$CVT_DTB	0002F5F1-R	LIB$CVT_ATB	CVTFILNAM	FILEREAD	
LIB$CVT_HTB	0002F5FF-R	LIB$CVT_ATB			
LIB$CVT_OTB	0002F5F8-R	LIB$CVT_ATB			
LIB$SIGNAL	0002F66A-R	LIB$SIGNAL	DIRECTORY		
LIB$STOP	0002F664-R	LIB$SIGNAL			
LINE_COUNT	0001C169-R	PRINT	CLI		
LOAD_SELF	00000000		WK-DSLOAD		
LOC$ADAPTER	0001F616-R	ATTACH			
LOC$PTABLE	0001F66B-R	ATTACH	DIRECTORY	RMS	
LOC$PTDESC	0001F6BA-R	ATTACH	QIO		
LOGNUM	0001C15C-R	PRINT			
MAP$IOSPACE	00029696-R	ONLY730	MEMMGT		
MAPFREE	00028BD6-R	MEMMGT	CLI	CRDIMAGE	LOAD
			START		
MAPMEM	00028844-R	MEMMGT	KERNEL	SHOWMEMORY	
MAPMEM$IOSPACE	00028A9E-R	MEMMGT	ONLY730		
MAP_DEBUGGER	00029118-R	MEMMGT	DEBUG	START	
MBA$INITIAL	000320E3-R	MBAINT	QIO		
MBA$INT	00032038-R	MBAINT	QIO		
MEMPOOL$INIT	00028724-R	MEMALC	KERNEL	MEMMGT	
MMG$GL_POBASE	00010808-R	ENTRY			
MMG$GL_SPTBASE	00010804-R	ENTRY	IOSNPG	IOSRAM	MEMMGT
MMG$IOLOCK	00029092-R	MEMMGT	QIOFDT		
MMG$PAGEFAULT	00026224-R	EXCEPT			
MMG$UNLOCK	0002909C-R	MEMMGT	IOPOST		
MODELST	00013397-R	DEBUG	EXCEPT	MCHK730	
MP$GB_FUNCTION	0001B2E9-R	DEBUG			
MP$GB_SETCPU	0001AF8E-R	CLI	DEBUG		
MP$GL_AP_BASE_ADDR_BKPT	00000000		WK-DEBUG		
MP$GL_AP_BKPT_STORE	00000000		WK-DEBUG		
MP$GL_AP_OFFSET_BKPT	00000000		WK-DEBUG		
MP$GL_AP_SIGNAL_ARRAY	00000000		WK-EXCEPT		
MP$GL_CONDITION_FLAGS	00019E00-R	MPDUMMY	WK-CLI	WK-START	
MP$GL_FLAGS	00000000		WK-START		
MP$GL_IPR_STATUS	0001B2EA-R	DEBUG			
MP$GL_IPR_VALUE	0001B2EE-R	DEBUG			
MP$GL_LUN	0001AF8F-R	CLI			
MP$GL_REG_ADDRESS	0001B2F2-R	DEBUG			
MP$GL_REG_VALUE	0001B2F6-R	DEBUG			
MP$GL_RET_STATUS	0001B2FA-R	DEBUG			
MP$GL_SAVED_PSL	00000000		WK-EXCEPT		
MP$GL_SAVED_R1	0001B2FE-R	DEBUG			
MP$GR_BOOTED_PROCESSORS	0001A20C-R	KERNEL	CLI	CLOCK	DEBUG
			EXCEPT	MEMMGT	PRINT

Symbol	Value	Defined By	Referenced By ...		
			WK-START		
MP$GR_IDLING_PROCESSORS	00000000		WK-START		
MP$GR_SAVED_GPRS	00000000		WK-DEBUG	WK-EXCEPT	
MP$GT_DEVICE_NAME	0001AF93-R	CLI	DEBUG		
MP$R_ACTIVE_SERVICES	00000000		WK-KERNEL		
MP$R_INTERLOCK_END	00000000		WK-KERNEL		
MP$_ATTACHED_CPU_CHECK	0001C82D-R	MPDUMMY			
MP$_CLEAR_LOCKS	00000000		WK-START		
MP$_COND_DEBUG	00000000		WK-DEBUG		
MP$_COND_HANDLR	00000000		WK-EXCEPT		
MP$_DEALL_MEMORY	0001C81E-R	MPDUMMY	DISPAT		
MP$_GET_GPR	00000000		WK-DEBUG		
MP$_GET_IPR	00000000		WK-DEBUG		
MP$_GET_PROCESSOR_ID	0001C821-R	MPDUMMY	CLOCK	WK-DEBUG	EXCEPT
MP$_PRIMARY_CPU_CHECK	0001C830-R	MPDUMMY	WK-KERNEL		
MP$_RESUME_ATTACHED_PROCESSORS	0001C833-R	MPDUMMY	WK-DEBUG	EXCEPT	WK-KERNEL
MP$_STOP_EVERYTHING	0001C82A-R	MPDUMMY			
MS$CMD	00019C60-R	TSBTDRIVR			
MT$CHECK_ACCESS	0002F5C9-R	VMSDUMMY	ENTRY		
NDS$OPEN	00000000		WK-KERNEL		
NI$B_LOOPBACK	00000032	IOBASE			
NI$GW_RPB_BUFFER	00000000		WK-CONFIG		
NI$K_LEN	00000033	IOBASE			
ODS$GET	000293BC-R	ODS			
ODS$OPEN	000292CB-R	ODS	RMS		
ODS$READ	000294E5-R	ODS			
ONLY$GL_TENUSEC	00018631-R	ONLY730	KERNEL		
ONLY$GL_UBDELAY	00018635-R	ONLY730	KERNEL		
ONLY_ATTACH	000297BC-R	ONLY730	ATTACH		
ONLY_CPU	000298BA-R	ONLY730	KERNEL		
ONLY_SCB	000297BA-R	ONLY730	SCB		
POPT_BASE	00036400-R	ONLY730	ENTRY	KERNEL	MEMMGT
			SHOWMEMORY		
PARSE_DEVICE	0001EA31-R	ATTACH			
PATCHEXIST$T	0001C138-R	PRINT			
PA_INIT	00000000		WK-CONFIG		
PB$INIT	00000000		WK-CONFIG		
PCB_BASE	00032878-R	ONLY730	EXCEPT	KERNEL	SHOWMEMORY
PFN$AB_STATE	0001BE54-R	MEMMGT			
PFN$AB_TYPE	00000000	MEMMGT			
PFN$AL_BAK	0001BE4C-R	MEMMGT			
PFN$AL_PTE	0001BE48-R	MEMMGT			
PFN$AW_BLINK	00000000	MEMMGT			
PFN$AW_FLINK	00000000	MEMMGT			
PFN$AW_REFCNT	0001BE50-R	MEMMGT			
PFN$AW_SWPVBN	0001BE44-R	MEMMGT			
PHD_BASE	00032800-R	ONLY730	KERNEL	SHOWMEMORY	
POLL_TIME	0001C564-R	START	KERNEL		
PPA$CLOSE	00000000		WK-RMS		
PPA$DIRECTORY	00000000		WK-DIRECTORY		
PPA$GB_CPU	00000000		WK-KERNEL		
PPA$GB_MID	00000000		WK-KERNEL		

Symbol	Value	Defined By	Referenced By ...		
PPA$LOOKUP	00000000		WK-RMS		
PPA$READ	00000000		WK-RMS		
PR$_MAPEN	00000038	SYS$PRDEF	APT		
PR$_SID_TYPRTUV2	00000010	KERNEL	BOOTDRIVR	MEMMGT	
PR$_SYS_TYP800	00000002	KERNEL	CONSOLE		
PR$_SYS_TYP900	00000003	KERNEL	DIRECTORY	MEMMGT	RMS
PR$_SYS_TYP_LLL	0000000A	KERNEL			
PREADVBLK	00021F89-R	CONSOLE	QIO		
PRNT_AP_NAME	0001326C-R	DEBUG			
PRNT_PRMT	0001BF34-R	PARAM	SCRIPT		
PRT$C_UR	0000000F	SYS$PRDEF	DEBUG		
PRT$C_UW	00000004	SYS$PRDEF	DEBUG		
PT$EXTRACT	0001EEB6-R	ATTACH			
PT$INTERPRET	0001EBE0-R	ATTACH			
QA$ADD_FORCED_ERROR	0002B5CA-R	QAMAIN	QACHECKS		
QA$ADD_QA_ERROR	0002B5CD-R	QAMAIN	QACHECKS		
QA$AOB_CHECK_STATE	0001C3F8-R	QAMAIN	DISPAT	LOOP	QACHECKS
			START		
QA$AOB_FLAGS	0001C3FC-R	QAMAIN	DISPAT	PARAM	QACHECKS
			START		
QA$AOB_ROUTINE_STATE	0001C3F4-R	QAMAIN	QACHECKS		
QA$FIND_USER_CALL_FRAME	0002B76B-R	QAMAIN	QACHECKS		
QA$LOOP_ON_SUBTEST	0002B110-R	QACHECKS	QAMAIN		
QA$LOOP_ON_TEST	0002B00E-R	QACHECKS	QAMAIN		
QA$L_SAVED_DSA_FLAGS	0001C3F0-R	QAMAIN			
QA$MAIN	0002B7EB-R	QAMAIN	DISPAT	ERROR	KERNEL
			LOOP	PARAM	START
QA$MULTIPLE_PASS	0002AF3F-R	QACHECKS	QAMAIN		
QA$NORMAL_START	0002AE74-R	QACHECKS	QAMAIN		
QA$PRINT_FORCED_ERRORS	0002B9CC-R	QAMAIN	QACHECKS		
QA$RUN_BACKWARDS	0002B239-R	QACHECKS	QAMAIN		
QA$T_HEADER_1	0001894B-R	QACHECKS	QAMAIN		
QA$T_HEADER_2	00018971-R	QACHECKS	QAMAIN		
QA$T_HEADER_3	00018998-R	QACHECKS	QAMAIN		
QA$T_OPERATOR_REQUEST_MESSAGE	000187C5-R	QACHECKS	QAMAIN		
QA$W_BRANCH	0001C3E6-R	QAMAIN	QACHECKS		
QA$W_ERROR_NUMBER	0001C3E8-R	QAMAIN	QACHECKS		
QA$W_MULTIPLEPASS	0001C3C4-R	QAFLAGS	QACHECKS	QAMAIN	
QA$W_SAVE_LAST	0001C3EC-R	QAMAIN	DISPAT	QACHECKS	
QA$W_SAVE_TEST	0001C3EA-R	QAMAIN	QACHECKS		
QA$W_SUBTESTLOOPS	0001C3C2-R	QAFLAGS	QACHECKS	QAMAIN	
QA$W_TESTLOOPS	0001C3C0-R	QAFLAGS	QACHECKS	QAMAIN	
QIO$ADDUNIT	0002BCE1-R	QIO	START		
QIO$CLEANUP	0002BC1A-R	QIO	DIRECTORY	DISPAT	SCRIPT
			START		
QIO$DEALLOCATE	0002C56C-R	QIO	ONLY730		
QIO$INITIALIZE	0002BBC6-R	QIO	KERNEL		
QIO$MINCORE	00005E3C	KERNEL	MEMALC		
QIOCLEAN_CPU	0002978A-R	ONLY730	QIO		
Q_ADAPTER	0001A9F4-R	ATTACH	CLI		
RCTRLC	0001D225-R	KERNEL	PRINT		
READVBLK	00021F89-R	CONSOLE	QIO		

Symbol	Value	Defined By	Referenced By ...		
RECEIVE_POLL	00000000		WK-KERNEL		
REMOTE_NODE_CONTROL_LEVEL	00000000		WK-QIO		
RMS$ALLOC_CACHE	0002C9E2-R	RMS	ANSI	ODS	RT11
			ULTRIX		
RMS$CLEANUP	0002CA41-R	RMS	DISPAT		
RMS$CLOSE	0002D0BE-R	RMS	ENTRY		
RMS$CONNECT	0002CE62-R	RMS	ENTRY		
RMS$DISCONNECT	0002D064-R	RMS	ENTRY		
RMS$ERROR	0002C901-R	RMS			
RMS$FILE_TABLE	0001C410-R	RMS			
RMS$GET	0002CEE4-R	RMS	ENTRY		
RMS$INIT	0002CA32-R	RMS	KERNEL		
RMS$LOAD_XAB	0002C993-R	RMS			
RMS$OPEN	0002CAC5-R	RMS	ENTRY		
RMS$READ	0002CFA6-R	RMS	ENTRY		
RPTEADR	00029048-R	MEMMGT			
RRD50$B_BR	00000036	IOBASE			
RRD50$K_LEN	00000037	IOBASE			
RRD50$L_CSR	00000032	IOBASE			
RT11$GET	0002D2FB-R	RT11			
RT11$OPEN	0002D205-R	RT11	RMS		
RT11$READ	0002D40B-R	RT11			
RX50_DISK_SELECT	0001A689-R	KERNEL			
SCAN$ALPHA	0002A04C-R	PARSE	ATTACH	QIO	
SCAN$ALPHANUM	0002A05E-R	PARSE	ATTACH		
SCAN$BINARY	0002A0FE-R	PARSE			
SCAN$COMPARE	0002A0B1-R	PARSE	SCRIPT		
SCAN$DECIMAL	0002A0F4-R	PARSE	ATTACH		
SCAN$DEVICE	0002A1F1-R	PARSE	ATTACH	LOAD	RMS
SCAN$FILE	0002A070-R	PARSE			
SCAN$HEX	0002A103-R	PARSE			
SCAN$NUMERIC	0002A106-R	PARSE	ANSI	ATTACH	DIRECTORY
			DSVECS	PARAM	
SCAN$OCTAL	0002A0F9-R	PARSE			
SCAN$SEARCHLIST	0002A1A2-R	PARSE	QIO		
SCAN$SPACE	0002A082-R	PARSE			
SCAN$SPACES	0002A082-R	PARSE	ATTACH	PARAM	SCRIPT
SCAN$SPANC	0002A096-R	PARSE			
SCAN$SYMBOL	0002A03A-R	PARSE	ATTACH		
SCBVECTOR_000	000297C1-R	ONLY730	SCB		
SCBVECTOR_038	000297C5-R	ONLY730	SCB		
SCBVECTOR_03C	000297C9-R	ONLY730	SCB		
SCBVECTOR_050	000297CD-R	ONLY730	SCB		
SCBVECTOR_054	000297D1-R	ONLY730	SCB		
SCBVECTOR_058	000297D5-R	ONLY730	SCB		
SCBVECTOR_05C	000297D9-R	ONLY730	SCB		
SCBVECTOR_060	000297DD-R	ONLY730	SCB		
SCBVECTOR_064	000297E1-R	ONLY730	SCB		
SCBVECTOR_068	000297E5-R	ONLY730	SCB		
SCBVECTOR_06C	000297E9-R	ONLY730	SCB		
SCBVECTOR_070	000297ED-R	ONLY730	SCB		
SCBVECTOR_074	000297F1-R	ONLY730	SCB		

Symbol	Value	Defined By	Referenced By ...		
SCBVECTOR_078	000297F5-R	ONLY730	SCB		
SCBVECTOR_07C	000297F9-R	ONLY730	SCB		
SCBVECTOR_080	000297FD-R	ONLY730	SCB		
SCBVECTOR_0C4	00029801-R	ONLY730	SCB		
SCBVECTOR_0C8	00029805-R	ONLY730	SCB		
SCBVECTOR_0CC	00029809-R	ONLY730	SCB		
SCBVECTOR_0D0	0002980D-R	ONLY730	SCB		
SCBVECTOR_0D4	00029811-R	ONLY730	SCB		
SCBVECTOR_0D8	00029815-R	ONLY730	SCB		
SCBVECTOR_0DC	00029819-R	ONLY730	SCB		
SCBVECTOR_0E0	0002981D-R	ONLY730	SCB		
SCBVECTOR_0E4	00029821-R	ONLY730	SCB		
SCBVECTOR_0E8	00029825-R	ONLY730	SCB		
SCBVECTOR_0EC	00029829-R	ONLY730	SCB		
SCBVECTOR_0F0	0002982D-R	ONLY730	SCB		
SCBVECTOR_0F4	00029831-R	ONLY730	SCB		
SCBVECTOR_0F8	00029835-R	ONLY730	SCB		
SCBVECTOR_0FC	00029839-R	ONLY730	SCB		
SCB_BASE	00032C00-R	ONLY730	CHAN730	CONFIG	DEBUG
			DISPAT	QIO	SCB
			SHOWMEMORY		
SCB_IMAGE	0002F800-R	SCB	CONFIG	DEBUG	EXCEPT
			KERNEL	ONLY730	QIO
SCB_UNKINT	00033000-R	ONLY730	CHAN730	MEMMGT	SCB
SCH$ASTDEL	0001E330-R	ASTDEL	SCB		
SCH$GL_CURPCB	0001A635-R	KERNEL	ASTDEL		
SCH$GL_PCBVEC	0001080C-R	ENTRY	ASSIGN	ASTDEL	IOPOST
SCH$LOCKW	0002F5D0-R	VMSDUMMY	ENTRY		
SCH$NEWLVL	0001E35D-R	ASTDEL	ASTCON	CHMK	
SCH$POSTEF	00025C27-R	EVENT	ENTRY	IOPOST	
SCH$QAST	0001E454-R	ASTDEL	CLOCK	COMDRVSUB	ENTRY
			IOPOST		
SCH$UNLOCK	0002F5D7-R	VMSDUMMY	ENTRY		
SCH$WAKE	0002F5CC-R	VMSDUMMY			
SCRIPT$CONT	0002D91A-R	SCRIPT	CLI		
SCRIPT$FLUSH	0002D8CF-R	SCRIPT	ATTACH	CLI	DIRECTORY
			LOAD	SHOWMEMORY	START
SCRIPT$INIT	0002D778-R	SCRIPT	KERNEL		
SCRIPT$MINCORE	00000400	SCRIPT	MEMALC		
SCRIPT$OPEN	0002D782-R	SCRIPT	CLI		
SCRIPT$STOP	0002D911-R	SCRIPT	DEBUG	EXCEPT	KERNEL
SEC_TICK	00000064	CLOCK			
SEND_CONSOLE_RESPONSE	00000000		WK-QIO		
SET_ABORT	0001E5D1-R	ATTACH			
SGN$C_IRPCNT	00000040	KERNEL	MEMALC		
SGN$C_NPAGEDYN	0000263C	KERNEL			
SHOWMEMORYFLAGS	0001C54C-R	SHOWMEMORY	CLI		
SIZE_TERM	0002218C-R	CONSOLE	KERNEL		
SLU$B_ACTIV	00000033	IOBASE			
SLU$B_EXT	00000032	IOBASE			
SLU$B_SPEED	00000034	IOBASE			
SLU$K_LEN	00000035	IOBASE			

Symbol	Value	Defined By	Referenced By ...		
SS$_ACCVIO	0000000C	SYS$SSDEF	DEBUG	ERROR	IOPOST
			QIOREQ		
SS$_BADPARAM	00000014	SYS$SSDEF	ACPFDT		
SS$_BREAK	00000414	SYS$SSDEF	DEBUG		
SS$_CANCEL	00000830	SYS$SSDEF	CANCEL		
SS$_CONTINUE	00000001	SYS$SSDEF	DEBUG	ONLY730	
SS$_CONTROLC	00000651	SYS$SSDEF	CONSOLE	ONLY730	SCRIPT
SS$_CTRLERR	00000054	SYS$SSDEF	ERROR	ONLY730	
SS$_DATACHECK	0000005C	SYS$SSDEF	ONLY730		
SS$_DEVOFFLINE	00000084	SYS$SSDEF	QIOREQ		
SS$_ENDOFFILE	00000870	SYS$SSDEF	ACPFDT	CLI	SCRIPT
SS$_FILNOTACC	000000AC	SYS$SSDEF	ACPFDT		
SS$_ILLBLKNUM	000000DC	SYS$SSDEF	ACPFDT	IOPOST	
SS$_ILLIOFUNC	000000F4	SYS$SSDEF	QIOREQ	SCRIPT	
SS$_ILLPAGCNT	000000FC	SYS$SSDEF	BUFFER		
SS$_INSFMEM	00000124	SYS$SSDEF	CANCEL	QIO	
SS$_IVCHAN	0000013C	SYS$SSDEF	QIO		
SS$_NOPRIV	00000024	SYS$SSDEF	ACPFDT		
SS$_NORMAL	00000001	SYS$SSDEF	ACPFDT	BUFFER	CANCEL
			COMDRVSUB	IOSNPG	IOSRAM
			QIO	QIOREQ	SCB
			SCRIPT		
SS$_NOSUCHDEV	00000908	SYS$SSDEF	ERROR		
SS$_NOSUCHFILE	00000910	SYS$SSDEF	ERROR		
SS$_PAGOWNVIO	000001EC	SYS$SSDEF	BUFFER		
SS$_RESIGNAL	00000918	SYS$SSDEF	DEBUG	ONLY730	
SS$_ROPRAND	00000454	SYS$SSDEF	DEBUG		
SS$_TBIT	00000464	SYS$SSDEF	DEBUG		
SS$_WASSET	00000009	SYS$SSDEF	CONSOLE		
SSTKPTR	00035E00-R	ONLY730	EXCEPT	KERNEL	MEMMGT
			SHOWMEMORY		
STACK_BASE	00033400-R	ONLY730	EXCEPT	MEMMGT	SHOWMEMORY
SWI$GL_FQBL	0001A63D-R	KERNEL			
SWI$GL_FQFL	0001A639-R	KERNEL	FRKCTL		
SYMTAB_HI	0001B2E5-R	DEBUG	MEMMGT		
SYS$ALLOC	00010238-R	ENTRY	ATTACH		
SYS$ASCTIM	00010248-R	ENTRY	FAO		
SYS$ASSIGN	00010250-R	ENTRY	KERNEL		
SYS$BINTIM	00010258-R	ENTRY	CLI	CONSOLE	KERNEL
SYS$CALL_HANDL	00010210-R	ENTRY	EXCEPT	UNWIND	
SYS$CANCEL	00010260-R	ENTRY	DASSGN	PRINT	
SYS$CANTIM	00010268-R	ENTRY	CLOCK	DISPAT	
SYS$CLOSE	000105B8-R	ENTRY	CRDIMAGE	DIRECTORY	DSLOAD
			HELP	SCRIPT	
SYS$CLREF	00010298-R	ENTRY	CLOCK	FLAGS	KERNEL
			LOAD	QIOREQ	
SYS$CNTREG	000102A0-R	ENTRY	BUFFER	CRDIMAGE	
SYS$CONNECT	000105C0-R	ENTRY	DIRECTORY	DSLOAD	HELP
			KERNEL	SCRIPT	
SYS$CREATE	7FFEE1C8	SYS$P1_VECTOR	KERNEL		
SYS$CRELOG	7FFEDEB0	SYS$P1_VECTOR	KERNEL	LOAD	
SYS$CRETVA	800000C8	KERNEL	DEBUG		

Symbol	Value	Defined By	Referenced By ...		
SYS$CRMPSC	7FFEDE78	SYS$P1_VECTOR	KERNEL		
SYS$DALLOC	000102D8-R	ENTRY	ATTACH	KERNEL	
SYS$DASSGN	000102E0-R	ENTRY			
SYS$DCLEXH	7FFEDEF0	SYS$P1_VECTOR	KERNEL		
SYS$DELTVA	80000110	KERNEL			
SYS$DISCONNECT	000105D0-R	ENTRY	DSLOAD	SCRIPT	
SYS$DISK	0001BE7B-R	ONLY730	KERNEL	LOAD	
SYS$EXIT	00010340-R	ENTRY	CLI	KERNEL	
SYS$EXPREG	00010348-R	ENTRY	BUFFER	CRDIMAGE	MEMALC
SYS$FAO	00010350-R	ENTRY	ANSI	ATTACH	CVRTIM
			DEBUG	DIRECTORY	ERROR
			IOSPGD	KERNEL	PARAM
			PRINT	SCRIPT	
SYS$FAOL	00010358-R	ENTRY	ATTACH	DEBUG	PRINT
SYS$GET	00010580-R	ENTRY	DIRECTORY	HELP	SCRIPT
SYS$GETCHN	000104C8-R	ENTRY	KERNEL		
SYS$GETDVI	7FFEE410	SYS$P1_VECTOR	DIRECTORY		
SYS$GETJPI	7FFEE0D8	SYS$P1_VECTOR	KERNEL		
SYS$GETJPIW	7FFEE420	SYS$P1_VECTOR	KERNEL		
SYS$GETSYI	7FFEE3F8	SYS$P1_VECTOR	DIRECTORY	KERNEL	
SYS$GETTIM	00010378-R	ENTRY	CLOCK		
SYS$GL_OPRMBX	00010800-R	ENTRY			
SYS$GTCHAN	00010380-R	ENTRY			
SYS$HIBER	00010388-R	ENTRY	CLOCK	SCRIPT	
SYS$LKWSET	000103A0-R	ENTRY			
SYS$NUMTIM	000103B8-R	ENTRY	CVRTIM		
SYS$OPEN	00010608-R	ENTRY	CRDIMAGE	DIRECTORY	DSLOAD
			HELP	KERNEL	SCRIPT
SYS$PARSE	7FFEE230	SYS$P1_VECTOR	DIRECTORY		
SYS$PUT	7FFEE188	SYS$P1_VECTOR	PRINT		
SYS$QIO	000103C8-R	ENTRY	EXCEPT	SCRIPT	
SYS$QIOW	00010200-R	ENTRY	CONSOLE	ERROR	KERNEL
			PRINT	SCRIPT	
SYS$READ	00010590-R	ENTRY	DSLOAD		
SYS$READEF	000103D0-R	ENTRY	CONSOLE	FLAGS	SCRIPT
SYS$SEARCH	7FFEE248	SYS$P1_VECTOR	DIRECTORY		
SYS$SETAST	000103F8-R	ENTRY	KERNEL		
SYS$SETDDIR	7FFEE250	SYS$P1_VECTOR	KERNEL	LOAD	
SYS$SETEF	00010400-R	ENTRY	CLOCK	CONSOLE	FLAGS
			QIOREQ		
SYS$SETEXV	80000208	KERNEL	EXCEPT		
SYS$SETIMR	00010420-R	ENTRY	CLOCK	CONSOLE	
SYS$SETPRI	00010428-R	ENTRY			
SYS$SETPRT	00010430-R	ENTRY	DEBUG	ONLY730	
SYS$SETRWM	00010438-R	ENTRY			
SYS$SPACE	7FFEE218	SYS$P1_VECTOR	DSLOAD		
SYS$SRCHANDLER	7FFEE118	SYS$P1_VECTOR	LIB$SIGNAL		
SYS$SYSROOT	0001BE68-R	ONLY730			
SYS$TRNLOG	00010458-R	ENTRY	KERNEL		
SYS$ULKPAG	00010460-R	ENTRY			
SYS$ULWSET	00010468-R	ENTRY			
SYS$UNWIND	00010470-R	ENTRY	CONFIG	DEBUG	EXCEPT

Symbol	Value	Defined By	Referenced By ...		
			MEMMGT	ONLY730	PROBE
SYS$WAITFR	00010478-R	ENTRY			
SYS$WAKE	00010480-R	ENTRY	CLOCK	SCRIPT	
SYS$WFLAND	00010488-R	ENTRY			
SYS$WFLOR	00010490-R	ENTRY			
TICK_NS	000186A0	CLOCK	KERNEL		
TODR_BIAS	10000000	CLOCK	KERNEL		
TODR_BYTE_CNT	00000000		WK-KERNEL	WK-QIO	
TODR_STORE	00000000		WK-KERNEL	WK-QIO	
TS11_DRIVER	000304E8-R	TSBTDRIVR			
TST$MCHK	000261AC-R	EXCEPT	DISPAT	ONLY730	
TU70$K_LEN	00000032	IOBASE			
TU72$K_LEN	00000032	IOBASE			
TU_INIT	00030800-R	MUBTDRIVR			
TXCS$V_RDY	00000000		WK-KERNEL	WK-QIO	
T_EMESSG1	00011342-R	CLI	DEBUG		
UBA$INITIAL	0002973E-R	ONLY730	WK-QIO		
UBA$UNEXINT	00029740-R	ONLY730	QIO		
UBADP_CPU	00029731-R	ONLY730	QIO		
UBINTSZ	000000B4	ONLY730	WK-QIO		
UBINT_DISP	000190CC-R	QIO	ONLY730		
UDA_RESPONSE	00030450-R	PUBTDRIVR	WK-CONFIG		
UD_INIT	00030200-R	PUBTDRIVR	WK-CONFIG		
ULTRIX$B	00002000	ULTRIX	MEMALC		
ULTRIX$GET	0002F23D-R	ULTRIX			
ULTRIX$IOB	000040D6	ULTRIX	MEMALC		
ULTRIX$OPEN	0002EAC2-R	ULTRIX	RMS		
ULTRIX$READ	0002F54A-R	ULTRIX			
ULTRIX_B_ADD	0001C5E8-R	ULTRIX	MEMALC		
ULTRIX_FLAGS	0001BDD0-R	LOAD	CLI	DIRECTORY	HELP
			PARSE	RMS	SCRIPT
ULTRIX_IOB_ADD	0001C5E4-R	ULTRIX	MEMALC	SHOWMEMORY	
ULTRIX_V_MODE	00000000	LOAD	CLI	PARSE	SCRIPT
UNMAP_DEBUGGER	00029131-R	MEMMGT	DEBUG		
USTKPTR	00036400-R	ONLY730	EXCEPT	KERNEL	MEMMGT
			SHOWMEMORY		
VAX$MODIFY_EXCEPTION	00000000		WK-EXCEPT		
VMS$QIO	00032393-R	QIOREQ	QIO		
VRABORT	0002E7BE-R	START	APT	CLI	
VRCLRBRK	000233D4-R	DEBUG	APT	CLI	
VRCLREF	000268B8-R	FLAGS	CLI		
VRCLRFLG	00026881-R	FLAGS	CLI		
VRDEPOSIT	000231BA-R	DEBUG	CLI		
VREXAMINE	00022E14-R	DEBUG	CLI		
VRRESTART	0002E3A4-R	START	APT	DISPAT	LOOP
VRSETBASE	00022D4A-R	DEBUG	CLI		
VRSETBRK	00023355-R	DEBUG	CLI		
VRSETDFLT	00022D67-R	DEBUG	CLI		
VRSETEF	00026897-R	FLAGS	CLI		
VRSETFLG	0002685C-R	FLAGS	CLI		
VRSETMEM	0002E2A3-R	SHOWMEMORY	CLI		
VRSETQA	0002B314-R	QAFLAGS	CLI		

Symbol	Value	Defined By	Referenced By ...		
VRSETWIDTH	00021C70-R	CONSOLE	CLI		
VRSHOWBASE	00022D53-R	DEBUG	CLI		
VRSHOWBRK	00023474-R	DEBUG	CLI		
VRSHOWDFLT	00022DAA-R	DEBUG	CLI		
VRSHOWEF	0002696D-R	FLAGS	CLI	QAMAIN	
VRSHOWFLG	000268D9-R	FLAGS	CLI	QAMAIN	
VRSHOWQA	0002B3C1-R	QAFLAGS	CLI		
VRSHOWSECTIONS	0002E989-R	START	CLI		
VRSHOWSTATUS	00025181-R	DISPAT	CLI		
VRSHOWWIDTH	00021D15-R	CONSOLE	CLI		
VRSHOW_CALLS	0001F900-R	CALLFRAME	CLI		
VRSTART	0002E301-R	START	APT	CLI	LOAD
VRSUMMARY	00025109-R	DISPAT	CLI		
VRSUPPORT	0002E8C3-R	START	CLI		
WIDE_FLAG	0001B315-R	DIRECTORY	CLI		
WRITEVBLK	00021F6D-R	CONSOLE	QIO		
XDELBPT	00000000		WK-CLI	WK-KERNEL	
XDELTBIT	00000000		WK-KERNEL		
XDS$INIT	00000000	MEMMGT			

Symbols By Value

Value	Symbols...		
00000000	AP$_REQ_CONSRX	CRD$K_CRD_INITIALIZATION	CRD$K_DS_FLAGS
	CRD$K_HOOK_POINT	CRD$K_READ	DIR_V_CMM
	DS$K_BLSS_B	KA$V_F_FLOAT	PFN$AB_TYPE
	PFN$AW_BLINK	PFN$AW_FLINK	ULTRIX_V_MODE
	XDS$INIT		
00000001	AP$GK_PROMPT	AP$GK_REQ_CONSRX	AP$_REQ_RXCDVEC
	CRD$K_DS_CONTROL_C	CRD$K_LAST_HOOK_POINT	CRD$K_TYPECODE
	CRD$K_WRITE	DIR_V_DRVCLR	DS$K_BLSSU_B
	DYN$C_ADP	KA$M_F_FLOAT	KA$V_D_FLOAT
	SS$_CONTINUE	SS$_NORMAL	
00000002	AP$GK_PROMPT_WITH_STATUS	CRD$K_LAST_ACCESS_CODE	CRD$K_LAST_DS_VARIABLE
	CRD$K_LAST_FUNCTION	DS$K_BLEQ_B	DYN$C_ACB
	KA$M_D_FLOAT	KA$V_G_FLOAT	PR$_SYS_TYP800
00000003	AP$GK_NO_PROMPT	BPTCODE	DS$K_BLEQU_B
	DYN$C_AQB	KA$V_H_FLOAT	PR$_SYS_TYP900
00000004	DS$K_BEQL_B	DYN$C_CEB	KA$M_G_FLOAT
	KA$V_CRC	PRT$C_UW	
00000005	DEV$V_SQD	DS$K_BEQLU_B	DYN$C_CRB
	KA$V_PACKED		
00000006	DEV$V_SPL	DS$K_BGEQ_B	DYN$C_DDB
	IO$V_CANCTRLO	KA$V_PACK_MUL	
00000007	DS$K_BGEQU_B	DYN$C_FCB	KA$V_EDIT
00000008	DS$K_BGTR_B	DYN$C_FRK	KA$M_H_FLOAT
	KA$V_CHAR		
00000009	DS$K_BGTRU_B	DYN$C_IDB	SS$_WASSET
0000000A	DS$K_BNEQ_B	DYN$C_IRP	PR$_SYS_TYP_LLL
0000000B	DS$K_BNEQU_B	DYN$C_LOG	
0000000C	DS$K_BLSS_M	DYN$C_PCB	SS$_ACCVIO
0000000D	DEV$V_NET	DS$K_BLSSU_M	DYN$C_PQB
0000000E	DEV$V_FOD	DS$K_BLEQ_M	DYN$C_RVT
0000000F	BPTMAX	DEF$C_ULTRIX_LEN	DS$K_ASCIBUF
	DS$K_BLEQU_M	DYN$C_TQE	PRT$C_UR
00000010	DEV$V_SHR	DS$K_BEQL_M	DYN$C_UCB
	KA$M_CRC	KA$V_TODR	PR$_SID_TYPRTUV2
00000011	DEF$C_BKSTP_LEN	DS$K_BEQLU_M	DYN$C_VCB
00000012	DS$K_BGEQ_M	DYN$C_WCB	
00000013	DEV$V_MNT	DS$K_BGEQU_M	DYN$C_BUFIO
00000014	DEV$V_MBX	DS$K_BGTR_M	DS$K_NYSIZ
	DYN$C_TYPAHD	HP$C_NAMELEN	SS$_BADPARAM
00000015	DEV$V_DMT	DS$K_BGTRU_M	DYN$C_GSD
00000016	DS$K_BNEQ_M	DYN$C_MVL	
00000017	DEV$V_ALL	DS$K_BNEQU_M	DYN$C_NET
00000018	DEV$V_FOR	DS$K_BVS_M	DYN$C_KFI
	KA$V_WCS_LD		
00000019	DS$K_BVC_M	DYN$C_MTL	KA$V_SB_ERR
0000001A	DS$K_BCS_M	DYN$C_BRDCST	HPE$C_EPB_SIZE
0000001B	DS$K_BCC_M	DYN$C_CXB	
0000001C	DS$K_BBS	DYN$C_NDB	
0000001D	DS$K_BBC	DYN$C_SSB	

Value	Symbols...		
0000001E	DS$K_BBSS	DYN$C_DPT	
0000001F	DS$K_BBCS	DYN$C_JPB	IO$_PHYSICAL
00000020	DS$K_BBSC	DYN$C_PBH	KA$M_PACKED
00000021	DS$K_BBCC	DYN$C_PDB	IO$_READLBLK
00000022	DS$K_BBSSI	DYN$C_PIB	
00000023	DS$K_BBCCI	DYN$C_PFL	IO$_SETMODE
00000024	DS$GK_SUPPORT_TABLE_ROWS	DS$K_BLBS	DYN$C_SFT
	FIL$C_DIR_SIZE	SS$_NOPRIV	
00000025	DS$K_BLBC	DYN$C_PTR	
00000026	DS$K_BERROR	DYN$C_KFH	
00000027	DS$K_BNERROR	DYN$C_RVX	IO$_SENSEMODE
00000028	DYN$C_EXTGSD		
00000029	DYN$C_SHMGSD		
0000002A	DYN$C_SHB		
0000002B	DYN$C_MBX		
0000002C	DYN$C_IRPE		
0000002D	DYN$C_SLAVCEB		
0000002E	DYN$C_SHMCEB		
0000002F	DYN$C_JIB	IO$_LOGICAL	
00000030	DYN$C_TWP	IO$_WRITEVBLK	
00000031	DYN$C_RBM	IO$_READVBLK	
00000032	DX20$B_DRIVE	DYN$C_VCA	HP$B_DHB32_NODE_ID
	HP$B_DMB32_NODE_ID	HP$B_DX20_DRIVE	HP$K_TU70_LEN
	HP$K_TU72_LEN	KA$L_FLAGS	NI$B_LOOPBACK
	RRD50$L_CSR	SLU$B_EXT	TU70$K_LEN
	TU72$K_LEN		
00000033	DX20$K_LEN	HP$K_DHB32_LEN	HP$K_DMB32_LEN
	HP$K_DX20_LEN	NI$K_LEN	SLU$B_ACTIV
00000034	SLU$B_SPEED		
00000035	SLU$K_LEN		
00000036	KA$W_WCS	RRD50$B_BR	
00000037	IO$_READPROMPT	RRD50$K_LEN	
00000038	KA$W_RESERVED	PR$_MAPEN	
0000003A	KA$L_SID		
0000003D	KA$B_SID_TYPE		
0000003E	KA$W_LATENCY		
0000003F	IO$M_FCODE		
00000040	DS$K_CCB	IO$M_CANCTRLO	IO$M_NOW
	KA$M_PACK_MUL	KA$W_PROGRESS	SGN$C_IRPCNT
00000042	KA$B_ACC_TYPE		
00000043	KA$B_RESERVED		
00000044	KA$W_MEM_SIZE		
00000046	KA$B_NODE_ID		
00000047	KA$L_MEM_SIZE		
0000004B	HP$L_KAA_SLOT		
0000004F	HP$B_CPU_ID		
00000050	KA$K_LEN		
00000054	SS$_CTRLERR		
0000005C	SS$_DATACHECK		
00000064	SEC_TICK		
00000080	DS$K_STRBUF	DYN$C_SHRBUFIO	DYN$C_SPECIAL
	KA$M_EDIT		

```
Value                                   Symbols...
-----                                   ----------
00000084        SS$_DEVOFFLINE
000000AC        SS$_FILNOTACC
000000B4        UBINTSZ
000000DC        SS$_ILLBLKNUM
000000F4        SS$_ILLIOFUNC
000000FC        SS$_ILLPAGCNT
00000100        IO$M_CTRLCAST                   KA$M_CHAR
00000124        SS$_INSFMEM
0000013C        SS$_IVCHAN
000001EC        SS$_PAGOWNVIO
00000200        DS$A_PRGBGN                     DS$GK_USOVR                     DS$K_BUFSIZ
00000218        FIL$C_SIZE
00000400        SCRIPT$MINCORE
00000414        SS$_BREAK
00000454        SS$_ROPRAND
00000464        SS$_TBIT
00000600        DS$K_SSSIZE
00000651        SS$_CONTROLC
00000830        SS$_CANCEL
00000870        SS$_ENDOFFILE
00000908        SS$_NOSUCHDEV
00000910        SS$_NOSUCHFILE
00000918        SS$_RESIGNAL
00001000        IO$M_TRMNOECHO
00002000        ULTRIX$B
0000263C        SGN$C_NPAGEDYN
00003000        DSA$M_DFLTFLGS
000040D6        ULTRIX$IOB
00005E3C        QIO$MINCORE
0000F800        DS$K_PRGSIZ
00010000        KA$M_TODR
00010010      R-DS$ENDPASS
00010018      R-DS$GPHARD
00010020      R-DS$ABORT
00010028      R-DS$DOSUMMARY                    R-DS$SUMMARY
00010030      R-DS$BGNSUB
00010038      R-DS$ENDSUB
00010040      R-DS$CKLOOP
00010048      R-DS$INLOOP
00010050      R-DS$ESCAPE
00010058      R-DS$BREAK
00010060      R-DS$WAITMS
00010068      R-DS$WAITUS
00010070      R-DS$ABORTWAIT                    R-DS$CANWAIT
00010078      R-DS$CNTRLC
00010080      R-DS$ASKDATA                      R-DS$GETDATA
00010088      R-DS$ASKVLD                       R-DS$GETVIELD
00010090      R-DS$ASKADR                       R-DS$GETADDRESS
00010098      R-DS$ASKLGCL                      R-DS$GETLOGICAL
000100A0      R-DS$ASKSTR                       R-DS$GETSTRING
000100A8      R-DS$BRANCH
000100B0      R-DS$CVTREG
```

```
Value                                   Symbols...
-----                                   ----------
000100B8     R-DS$PARSE
000100C0     R-DS$ERRSYS
000100C8     R-DS$ERRDEV
000100D0     R-DS$ERRHARD
000100D8     R-DS$ERRSOFT
000100E0     R-DS$PRINTB
000100E8     R-DS$PRINTX
000100F0     R-DS$PRINTF
000100F8     R-DS$PRINTS
00010100     R-DS$PRINTSIG
00010108     R-DS$ERRPREP
00010110     R-DS$SETPRIEXV
00010118     R-DS$MAPDBGBLOCK
00010120     R-DS$GETBUF
00010128     R-DS$RELBUF
00010150     R-DS$MMON
00010158     R-DS$MMOFF
00010160     R-DS$SETVEC
00010168     R-DS$CLRVEC
00010170     R-DS$INITSCB
00010178     R-DS$SETIPL
00010180     R-DS$CHANNEL
00010188     R-DS$SETMAP
00010190     R-DS$SHOCHAN                       R-DS$SHOWCHAN
00010198     R-DS$LOAD
000101A0     R-DS$PROBE
000101A8     R-DS$ATTACH
000101B0     R-DS$HELP
000101B8     R-DS$FREEDBGSYM
000101C0     R-DS$LOADPCS
000101C8     R-DS$GETTERM
000101D0     R-DS$SERVICE_DISPATCHER
000101D8     R-DS$MEMSIZE
00010200     R-DS$AQ_SYSSRV                     R-SYS$QIOW
00010210     R-SYS$CALL_HANDL
00010238     R-SYS$ALLOC
00010248     R-SYS$ASCTIM
00010250     R-SYS$ASSIGN
00010258     R-SYS$BINTIM
00010260     R-SYS$CANCEL
00010268     R-SYS$CANTIM
00010298     R-SYS$CLREF
000102A0     R-SYS$CNTREG
000102D8     R-SYS$DALLOC
000102E0     R-SYS$DASSGN
00010340     R-SYS$EXIT
00010348     R-SYS$EXPREG
00010350     R-SYS$FAO
00010358     R-SYS$FAOL
00010378     R-SYS$GETTIM
00010380     R-SYS$GTCHAN
00010388     R-SYS$HIBER
```

```
Value                                   Symbols...
-----                                   ----------
000103A0     R-SYS$LKWSET
000103B8     R-SYS$NUMTIM
000103C8     R-SYS$QIO
000103D0     R-SYS$READEF
000103F8     R-SYS$SETAST
00010400     R-SYS$SETEF
00010420     R-SYS$SETIMR
00010428     R-SYS$SETPRI
00010430     R-SYS$SETPRT
00010438     R-SYS$SETRWM
00010458     R-SYS$TRNLOG
00010460     R-SYS$ULKPAG
00010468     R-SYS$ULWSET
00010470     R-SYS$UNWIND
00010478     R-SYS$WAITFR
00010480     R-SYS$WAKE
00010488     R-SYS$WFLAND
00010490     R-SYS$WFLOR
000104C8     R-SYS$GETCHN
00010580     R-SYS$GET
00010590     R-SYS$READ
000105B8     R-SYS$CLOSE
000105C0     R-SYS$CONNECT
000105D0     R-SYS$DISCONNECT
00010608     R-SYS$OPEN
000107FF     R-DS$AQ_SSEND
00010800     R-SYS$GL_OPRMBX
00010804     R-MMG$GL_SPTBASE
00010808     R-MMG$GL_P0BASE
0001080C     R-SCH$GL_PCBVEC
00010810     R-EXE$GL_ABSTIM
00010818     R-IOC$GL_PSFL
0001081C     R-IOC$GL_PSBL
00010930     R-IOC$RTN
00010B00     R-EXE$GQ_SYSTIME
00010BAE     R-DS$GT_LEGAL
00011342     R-T_EMESSG1
00011470     R-GT_MENU_ERROR
0001151A     R-COMMAND_TREE
00012297     R-HSC50_BOOT
00013198     R-FMTBPT
0001326C     R-PRNT_AP_NAME
00013356     R-APSLMNE
00013397     R-MODELST
00013B08     R-DS$SRVDISP_TABLE
00014643     R-AAT_ARITH
00014E64     R-DS$GA_PTDESC
000180A7     R-FIL$GT_DDDEV
000180A8     R-FIL$GT_DDSTRING
00018628     R-IO$GQ_PHYSICAL
00018630     R-DS$GB_PRIMARY_CPU_IDENTIFIER
00018631     R-ONLY$GL_TENUSEC
```

```
Value                                   Symbols...
-----                                   ----------
00018635     R-ONLY$GL_UBDELAY
000186A0       TICK_NS
000187C5     R-QA$T_OPERATOR_REQUEST_MESSAGE
0001894B     R-QA$T_HEADER_1
00018971     R-QA$T_HEADER_2
00018998     R-QA$T_HEADER_3
0001908C     R-EXE$GL_PWRDONE
000190C0     R-CTL$GW_NMIOCH
000190C2     R-CTL$GW_CHINDX
000190CC     R-UBINT_DISP
0001918C     R-DEF$Q_BACKSTOP
0001919D     R-DEF$Q_ULTRIX
00019374     R-DS$GA_SUPPORT_TABLE
00019C60     R-MS$CMD
00019C80     R-DS$GQ_DAYTIM
00019C9F     R-DS$GT_START
00019CD5     R-DS$GT_VDS_VERSION
00019CDF     R-DS$GT_PROMPT
00019E00     R-MP$GL_CONDITION_FLAGS
00019E04     R-DS$GT_COPYRIGHT
00019E32     R-DS$GT_COPYEAR
00019E51     R-DS$GT_LEGAL_END
00019E70     R-DS$FAB_INPUT
00019EC0     R-DS$RAB_INPUT
00019F04     R-DS$FAB_OUTPUT
00019F54     R-DS$RAB_OUTPUT
00019F98     R-DS$GA_DS_CTRL_C_FIRST
00019F9C     R-DS$GA_DS_CTRL_C_SECOND
00019FA0     R-DS$L_USERCNTRLC
0001A00C     R-IMAGE_HEADER
0001A20C     R-IMAGE_HEADER_END                R-MP$GR_BOOTED_PROCESSORS
0001A210     R-DS$GB_SYSTYPE
0001A231     R-CTL$GL_CCB
0001A631     R-CTL$GL_CCBBASE
0001A635     R-DS$GL_PCBVEC                    R-SCH$GL_CURPCB
0001A639     R-SWI$GL_FQFL
0001A63D     R-SWI$GL_FQBL
0001A669     R-BOOT$L_ADP
0001A66D     R-BOOT$L_CSR
0001A671     R-BOOT$L_UNIT
0001A675     R-BOOT$L_R1
0001A679     R-BOOT$L_R2
0001A67D     R-BOOT$L_R5
0001A681     R-BOOT$L_DEVTYP
0001A685     R-BOOT$L_SCORPIO_R3
0001A689     R-RX50_DISK_SELECT
0001A68D     R-BOOT$A_CCA
0001A691     R-DS$GL_EFC0                      R-DS$GQ_EFL
0001A695     R-DS$GL_EFC1
0001A699     R-DS$GA_LASTADR
0001A69D     R-DS$GL_FLAGS
0001A6A1     R-DS$GW_TTIN
```

```
Value                                   Symbols...
-----                                   ----------
0001A6A3      R-DS$GW_TTOUT
0001A6A5      R-DS$GL_ERRCNT
0001A6A9      R-DS$GL_HARDERR_COUNT
0001A6AD      R-DS$GL_PREPERR_COUNT
0001A6B1      R-DS$GL_SOFTERR_COUNT
0001A6B5      R-DS$GL_DEVERR_COUNT
0001A6B9      R-DS$GL_SYSERR_COUNT
0001A6BD      R-DS$GL_ERRSUP_COUNT
0001A6C1      R-DS$GL_NUMTEST
0001A6C5      R-DS$GL_FSTTEST
0001A6C9      R-DS$GL_LSTTEST
0001A6CD      R-DS$GL_SUBTEST
0001A6D1      R-DS$GA_CHKLPPC
0001A6D5      R-DS$GA_LOOPADR
0001A6D9      R-DS$GL_RUNTIM
0001A6DD      R-DS$GL_TIMESEC
0001A6E1      R-DS$GL_WAITIM
0001A6E5      R-DS$GL_RADIX
0001A6E9      R-DS$GL_SIZE
0001A6ED      R-DS$GL_BUFLEN                    R-DS$GQ_BUFQWD
0001A6F1      R-DS$GA_BUFPTR
0001A6F5      R-DS$GB_BYTEBUF
0001A6F6      R-DS$GB_TYPECODE
0001A6F7      R-DS$GB_INHIBIT_NAMING
0001A6F8      R-DS$GT_ASCIBUF
0001A707      R-DS$GT_BUFFER
0001A907      R-DS$GT_STRBUF
0001A987      R-DS$GL_TERMCHAR
0001A9E8      R-AP$GW_DATA
0001A9EA      R-AP$GW_AP_NODES
0001A9EC      R-AP$GR_SHARED_DATA
0001A9F4      R-Q_ADAPTER
0001AA00      R-DS$GQ_PHYADR
0001AA08      R-DS$GL_BUFCNT
0001AA20      R-CDB$BLOCK
0001AA70      R-CVT$T_MBASR
0001AA74      R-CVT$T_MBAMAP
0001AA78      R-CVT$T_UBADPR
0001AA7C      R-CVT$T_UBAMAP
0001AB1C      R-DS$GL_CLIBASE
0001AF8E      R-MP$GB_SETCPU
0001AF8F      R-MP$GL_LUN
0001AF93      R-MP$GT_DEVICE_NAME
0001AFA4      R-DS$GQ_CURYEAR
0001AFAC      R-DS$GT_NEWYEAR
0001B00E      R-DS$GB_WIDTH
0001B098      R-DS$GL_CRD_FLAGS
0001B09C      R-CRD$GL_CRD_COMMAND_DEBUG
0001B0A0      R-DS$AL_DS_DISPATCH_VECTORS
0001B0A4      R-DS$GA_CRD_DS_INTERFACE
0001B0BC      R-DS$GA_USER_ERROR_TEXT
0001B0D0      R-DS$GL_CRD_DEBUG
```

```
Value                                   Symbols...
-----                                   ----------
0001B108      R-DEBUG$GB_FLAGS
0001B109      R-DS$AB_BPTINST
0001B11C      R-DS$AA_BPTADDR
0001B15C      R-DS$AA_BPTBASE
0001B2D5      R-DBG_ADDR_RTN
0001B2DD      R-DEBUG_LO
0001B2E1      R-DEBUG_HI
0001B2E5      R-SYMTAB_HI
0001B2E9      R-MP$GB_FUNCTION
0001B2EA      R-MP$GL_IPR_STATUS
0001B2EE      R-MP$GL_IPR_VALUE
0001B2F2      R-MP$GL_REG_ADDRESS
0001B2F6      R-MP$GL_REG_VALUE
0001B2FA      R-MP$GL_RET_STATUS
0001B2FE      R-MP$GL_SAVED_R1
0001B315      R-WIDE_FLAG
0001B316      R-END_SIZE
0001B6C8      R-DIR_FLAGS
0001B71C      R-DS$GQ_TESTID
0001B76C      R-HELPINFO
0001B79C      R-DS$GA_SELECTS
0001B7B0      R-DS$GA_ALLOCATES
0001B7C4      R-DS$GA_SELECTED
0001B9C8      R-DS$GA_ALLOCATED
0001BBCC      R-DS$GA_PTABLE
0001BDD0      R-ULTRIX_FLAGS
0001BDD1      R-DIAG_FILE_NAME
0001BE34      R-EXE$GL_SPLITADR
0001BE38      R-DS$GL_LOOKASIDE
0001BE3C      R-IOC$GL_IRPFL
0001BE44      R-PFN$AW_SWPVBN
0001BE48      R-PFN$AL_PTE
0001BE4C      R-PFN$AL_BAK
0001BE50      R-PFN$AW_REFCNT
0001BE54      R-DS$GL_MEMSIZE                   R-PFN$AB_STATE
0001BE58      R-DS$GL_ACTUAL_MEMSIZE
0001BE5C      R-DS$GL_SET_MEMSIZE
0001BE60      R-DS$GB_MM_ENB
0001BE68      R-SYS$SYSROOT
0001BE7B      R-SYS$DISK
0001BE8B      R-DEF$Q_DEV
0001BEB1      R-DEF$Q_DIR
0001BF34      R-PRNT_PRMT
0001BF90      R-DEVREV$T
0001C130      R-FIRMEXIST$T
0001C134      R-HARDEXIST$T
0001C138      R-PATCHEXIST$T
0001C15C      R-LOGNUM
0001C160      R-DEVREV$T_X
0001C169      R-LINE_COUNT
0001C16D      R-DS$GL_PAGE
0001C3B9      R-DEBUG_THE_SUCKER                R-DS$GB_VDS_DEBUG
```

```
Value                                   Symbols...
-----                                   ----------
0001C3C0      R-QA$W_TESTLOOPS
0001C3C2      R-QA$W_SUBTESTLOOPS
0001C3C4      R-QA$W_MULTIPLEPASS
0001C3E6      R-QA$W_BRANCH
0001C3E8      R-QA$W_ERROR_NUMBER
0001C3EA      R-QA$W_SAVE_TEST
0001C3EC      R-QA$W_SAVE_LAST
0001C3F0      R-QA$L_SAVED_DSA_FLAGS
0001C3F4      R-QA$AOB_ROUTINE_STATE
0001C3F8      R-QA$AOB_CHECK_STATE
0001C3FC      R-QA$AOB_FLAGS
0001C410      R-RMS$FILE_TABLE
0001C420      R-DS$GB_STOPPING_THE_TIMER
0001C421      R-DS$GA_CHMKVEC
0001C425      R-DS$GA_BREAKVEC
0001C429      R-DS$GA_TBITVEC
0001C42D      R-EXE$GL_SCB
0001C431      R-DS$GA_SOFTINT
0001C548      R-DS$GB_QIOACT
0001C54C      R-SHOWMEMORYFLAGS
0001C550      R-DS$GL_IPR_VALUE
0001C554      R-BIN_TIME
0001C55C      R-BEGIN_TIME
0001C564      R-POLL_TIME
0001C56C      R-DUE_TIME
0001C5E4      R-ULTRIX_IOB_ADD
0001C5E8      R-ULTRIX_B_ADD
0001C800      R-DSX$_INTERLOCK_TEST
0001C806      R-DSX$BOOTATTACHED
0001C80C      R-DSX$STARTATTACHED
0001C812      R-DSX$HALTATTACHED
0001C818      R-DSX$SHOWIDLE
0001C81E      R-MP$_DEALL_MEMORY
0001C821      R-MP$_GET_PROCESSOR_ID
0001C82A      R-MP$_STOP_EVERYTHING
0001C82D      R-MP$_ATTACHED_CPU_CHECK
0001C830      R-MP$_PRIMARY_CPU_CHECK
0001C833      R-MP$_RESUME_ATTACHED_PROCESSORS
0001C836      R-DSX$_SET_INTERLOCK
0001C839      R-DSX$_CLEAR_INTERLOCK
0001C83C      R-BOOT
0001C915      R-APT
0001CAFE      R-INI$BRK
0001CB00      R-DS$SOFTIPL5
0001CB0B      R-DS$USERMODE
0001D0A9      R-BEGIN                           R-BEGIN_BLISS
0001D0C6      R-INIT_CONTEXT
0001D1B7      R-CMK$REFRESH
0001D225      R-RCTRLC
0001D2C4      R-KB_CHECK
0001D322      R-KB_CHECK_APT
0001D4D5      R-GET_TIME
```

```
Value                                   Symbols...
-----                                   ----------
0001D600      R-DSX$CNTRLC
0001D62B      R-DSV$EXIT
0001D72B      R-DSR$CHECK_MENUTEST_OFF_SET
0001D74F      R-DSR$CHECK_AUTOTEST_OFF_SET
0001D76E      R-DSX$GETTERM
0001D77C      R-ANSI$OPEN
0001DD24      R-ANSI$CLOSE
0001DD55      R-ANSI$CACHEVBN
0001DE5A      R-ANSI$READ
0001DF1B      R-ANSI$GET
0001E0E8      R-GET_BLOCK_LENGTH
0001E118      R-CMK$APBOOT
0001E126      R-DSX$PPSCB
0001E130      R-APT_COM
0001E2EC      R-APT_MSG
0001E2F5      R-APT_DONE
0001E304      R-EXE$SETAST
0001E330      R-SCH$ASTDEL
0001E35D      R-SCH$NEWLVL
0001E454      R-SCH$QAST
0001E4D4      R-DSV$ATTACH
0001E5D1      R-SET_ABORT
0001E5DE      R-DSX$ATTACH
0001EA31      R-PARSE_DEVICE
0001EBE0      R-PT$INTERPRET
0001EEB6      R-PT$EXTRACT
0001F07B      R-DSV$SHOWDEVICEB
0001F082      R-DSV$SHOWDEVICE
0001F0DC      R-DSV$SHOWSELECTB
0001F0E3      R-DSV$SHOWSELECT
0001F114      R-DSV$DEATTACH
0001F1DC      R-DSV$SELECT
0001F3BC      R-DSV$DESELECT
0001F569      R-INI$PTABLE
0001F616      R-LOC$ADAPTER
0001F66B      R-LOC$PTABLE
0001F6BA      R-LOC$PTDESC
0001F714      R-INS$PTABLE
0001F7AC      R-FIL$RDWRTLBN
0001F7CC      R-BOO$IMAGE_ATT
0001F800      R-DSX$GETSYSBUF
0001F80B      R-DSX$GETBUF
0001F873      R-DSX$RELSYSBUF
0001F87B      R-DSX$RELBUF
0001F900      R-VRSHOW_CALLS
0001F9BF      R-DSR$FIND_USER_PC
0001F9F0      R-DSX$CHANNEL
0001FAA1      R-DSX$SETMAP
0001FC71      R-DSP$SHOWMBA
0001FC7B      R-DSP$SHOWUBI
0001FC8C      R-DSX$SHOWCHAN
0001FD70      R-DSI$CHANNEL_VEC
```

```
Value                                   Symbols...
-----                                   ----------
0001FDD8      R-EXE_CMODKRNL
0001FE94      R-CMK$ASTEXIT
0001FECF      R-DS$CLI
0002000C      R-CLI_ACTION
0002022A      R-ACT_CONT
0002023D      R-ACT_DEFAULT_DBG
000203A7      R-ACT_NO_ALLOC
00020C8C      R-DSX$CANWAIT
00020CA7      R-EXE$WAKE
00020CB5      R-DSX$WAITMS
00020D40      R-DSX$WAITUS
00020D5B      R-DSX$WAITUS_USOVR
00020DC1      R-EXE$HIBER
00020DE4      R-CMK$HIBER
00020DFA      R-EXE$CANTIM
00020E12      R-CMK$CANTIM
00020E4A      R-DSR$SETIMR_INI
00020E69      R-EXE$INSTIMQ
00020E9B      R-EXE$SETIMR
00020EB0      R-CMK$SETIMR
00020F24      R-DSI$TIMSRV
00020FA8      R-DSI$TIMER
00021098      R-CLK$RE_INIT
000210F0      R-INI$LOAD_DEVICE
00021131      R-DSP$CONFIG_LOAD
00021C70      R-VRSETWIDTH
00021D15      R-VRSHOWWIDTH
00021D35      R-KB_POLL
00021DA2      R-CMK$RCONSOLE
00021F5E      R-CMK$TCONSOLE
00021F6D      R-WRITEVBLK
00021F89      R-PREADVBLK                       R-READVBLK
0002218C      R-SIZE_TERM
00022264      R-DSR$LOAD_CRD
00022546      R-DSR$UNLOAD_CRD
00022636      R-DSR$RESTORE_CRD_VECTORS
00022681      R-EXCHANGE_DISPATCH_VECTORS
00022B20      R-DSR$PRINT_TYPECODE_NAME
00022C74      R-FIL$CVTFILNAM
00022D4A      R-VRSETBASE
00022D53      R-VRSHOWBASE
00022D67      R-VRSETDFLT
00022DAA      R-VRSHOWDFLT
00022E14      R-VREXAMINE
000231BA      R-VRDEPOSIT
00023355      R-VRSETBRK
000233D4      R-VRCLRBRK
00023441      R-DSP$REMOVEBREAK
00023474      R-VRSHOWBRK
000234F5      R-COND_DEBUG
000238CD      R-FETCH_STORE
00023AC4      R-CMK$SGIPR
```

```
Value                                   Symbols...
-----                                   ----------
00023ADA     R-FETCH_STOR_PREG
00023B09     R-DSV$DEBUG
00023C7F     R-DSX$GPHARD
00023CB1     R-DSP$GEN_PTABLES
000248EF     R-DSX$DIRECTORY
00024C24     R-DSV$DIRECTORY
00024C50     R-DISPATCH
00024F08     R-DSX$ENDPASS
00025022     R-DS_CLEANUP
000250F7     R-DSX$DOSUMMARY
00025109     R-VRSUMMARY
00025181     R-VRSHOWSTATUS
00025250     R-DSX$LOAD
00025561     R-DSX$SERVICE_DISPATCHER
0002559A     R-DSX$ERRSOFT
000255C1     R-DSX$ERRHARD
000255E8     R-DSX$ERRPREP
0002560F     R-DSX$ERRDEV
0002563D     R-DSX$ERRSYS
000259AD     R-DS_TESTID
00025A19     R-DS_ERRSUP
00025B46     R-DSR$COMPLETION
00025BF3     R-EXE$SETEF
00025C0D     R-EXE$CLREF
00025C27     R-SCH$POSTEF
00025C3A     R-EXE$READEF
00025C5E     R-EXE$WAITFR
00025C7B     R-EXE$WFLOR
00025C9C     R-EXE$WFLAND
00025CC8     R-COND_HANDLR
00025E10     R-DSX$PRINTSIG
00025E22     R-DSX$PRINTSIGI
00026168     R-FIND_USERPC
000261AC     R-TST$MCHK
000261C4     R-EXE_MCHK
000261E4     R-EXE_KRNLSTK
000261F0     R-EXE_POWER
000261FC     R-EXE_OPCDEC
00026204     R-EXE_OPCCUS
0002620C     R-EXE$ROPRAND                  R-EXE_ROPRAND
00026214     R-EXE_RADRMOD
0002621C     R-EXE$ACVIOLAT                 R-EXE_ACCVIO
00026224     R-EXE_TRANSL                   R-MMG$PAGEFAULT
00026230     R-EXE_COMPAT
00026238     R-EXE_ARITH
00026244     R-EXE$BREAK                    R-EXE_BREAK
000262D8     R-EXE$TBIT                     R-EXE_TBIT
000262FE     R-EXE_UNEXPINT
00026308     R-EXE_CHMK
00026310     R-EXE_CHME
00026318     R-EXE_CHMS
00026320     R-EXE_CHMU
```

```
Value                                   Symbols...
-----                                   ----------
0002632F     R-EXE$REFLECT
000264FA     R-DSX$SETPRIEXV
00026538     R-EXE$FAO
00026540     R-EXE$FAOL
0002685C     R-VRSETFLG
00026881     R-VRCLRFLG
00026897     R-VRSETEF
000268B8     R-VRCLREF
000268D9     R-VRSHOWFLG
0002696D     R-VRSHOWEF
000269D8     R-FIL$OPENFILE
00026ADE     R-FIL$CACHE_INIT
00026B52     R-FIL$CACHE_TRUNC
00026C29     R-FIL$MOUNT
00026CBD     R-FIL$FINDFILID
00026FC6     R-FIL$RDCHKFILHDR
0002709E     R-FIL$WRITEVBN
000270A5     R-FIL$READVBN
0002715E     R-FIL$STATBLK
00027228     R-DSX$HELP
00027A60     R-DSR$KEY_EQUAL
00027BE8     R-IOC$IOPOST
00027D30     R-IOC$QNXTSEG
00027D3C     R-IOC$QNXTSEG1
00027D9A     R-IOC$WAKACP
00027EE6     R-IOC$DIRPOST1
00027FA6     R-DSV$SETLOAD
000280DD     R-DSV$SHOWLOAD
000280FD     R-DSR$IFLOADDEV
00028127     R-DSV$RUN
00028128     R-DSV$LOAD
00028248     R-DSX$ESCAPE
0002825D     R-DSX$BGNSUB
000282FB     R-DSX$ENDSUB
0002846C     R-DSX$CKLOOP
000284AC     R-DSX$INLOOP
000284D4     R-CHR$MCHK
0002853B     R-DSR$FAULT_CLEAR
0002853F     R-EXE$ALLOCBUF
00028543     R-EXE$ALLOCCEB
00028547     R-EXE$ALLOCJIB
00028550     R-EXE$ALLOCIRP
00028554     R-EXE$ALLOCPCB
0002855C     R-EXE$ALLOCPQB
00028565     R-EXE$ALLOCTQE
0002859F     R-EXE$ALONONPAGED
0002861D     R-EXE$ALLOCATE
00028645     R-EXE$DEANONPAGED
000286CE     R-EXE$DEALLOCATE
00028724     R-MEMPOOL$INIT
0002881F     R-DSX$MEMSIZE
00028844     R-MAPMEM
```

```
Value                                   Symbols...
-----                                   ----------
00028A9E     R-MAPMEM$IOSPACE
00028BD6     R-MAPFREE
00028C72     R-CRD$EXPREG
00028C8F     R-EXE$EXPREG
00028DAA     R-EXE$CNTREG
00028EE0     R-DSX$MMON
00028EFA     R-DSX$MMOFF
00028F20     R-DSR$MMENABLE
00028F33     R-CMK$MMENABLE
00028F5D     R-EXE$SETPRT
00028F72     R-CMK$SETPRT
00029048     R-RPTEADR
00029092     R-MMG$IOLOCK
0002909C     R-MMG$UNLOCK
000290A0     R-DSX$MAPDBGBLOCK
000290EE     R-DSX$FREEDBGSYM
00029118     R-MAP_DEBUGGER
00029131     R-UNMAP_DEBUGGER
000291CB     R-INI$RDONLY                       R-INI$WRITABLE
000292CB     R-ODS$OPEN
000293BC     R-ODS$GET
000294E5     R-ODS$READ
00029598     R-DSP$CONFIG_ADP
00029696     R-MAP$IOSPACE
000296EA     R-FETCH_LONG
0002971E     R-IOC$PURGDATAP
00029731     R-UBADP_CPU
0002973E     R-UBA$INITIAL
00029740     R-UBA$UNEXINT
00029741     R-IO$TESTCSR_L
00029760     R-IO$TESTCSR_W
0002977F     R-IOGEN$CONN_VEC
0002978A     R-QIOCLEAN_CPU
000297BA     R-ONLY_SCB
000297BB     R-HDW_CLOCK_SET
000297BC     R-ONLY_ATTACH
000297C1     R-SCBVECTOR_000
000297C5     R-SCBVECTOR_038
000297C9     R-SCBVECTOR_03C
000297CD     R-SCBVECTOR_050
000297D1     R-SCBVECTOR_054
000297D5     R-SCBVECTOR_058
000297D9     R-SCBVECTOR_05C
000297DD     R-SCBVECTOR_060
000297E1     R-SCBVECTOR_064
000297E5     R-SCBVECTOR_068
000297E9     R-SCBVECTOR_06C
000297ED     R-SCBVECTOR_070
000297F1     R-SCBVECTOR_074
000297F5     R-SCBVECTOR_078
000297F9     R-SCBVECTOR_07C
000297FD     R-SCBVECTOR_080
```

```
Value                                   Symbols...
-----                                   ----------
00029801      R-SCBVECTOR_0C4
00029805      R-SCBVECTOR_0C8
00029809      R-SCBVECTOR_0CC
0002980D      R-SCBVECTOR_0D0
00029811      R-SCBVECTOR_0D4
00029815      R-SCBVECTOR_0D8
00029819      R-SCBVECTOR_0DC
0002981D      R-SCBVECTOR_0E0
00029821      R-SCBVECTOR_0E4
00029825      R-SCBVECTOR_0E8
00029829      R-SCBVECTOR_0EC
0002982D      R-SCBVECTOR_0F0
00029831      R-SCBVECTOR_0F4
00029835      R-SCBVECTOR_0F8
00029839      R-SCBVECTOR_0FC
00029843      R-DSR$ONLY_CONSOLE_CONFIG
000298BA      R-ONLY_CPU
000298BC      R-DSX$GETDATA
00029941      R-DSX$GETVIELD
000299B2      R-DSX$GETADDRESS
00029A0B      R-DSX$GETLOGICAL
00029B2B      R-DSX$GETSTRING
00029E30      R-DSX$SUPERPARSE
00029E37      R-DSX$PARSE
0002A03A      R-SCAN$SYMBOL
0002A03C      R-DS$GT_VSYMBOL
0002A04C      R-SCAN$ALPHA
0002A04E      R-DS$GT_VALPHA
0002A05E      R-SCAN$ALPHANUM
0002A060      R-DS$GT_VALPHANUM
0002A070      R-SCAN$FILE
0002A072      R-DS$GT_VFILE
0002A082      R-SCAN$SPACE                      R-SCAN$SPACES
0002A084      R-DS$GT_VSPACE
0002A096      R-SCAN$SPANC
0002A0B1      R-SCAN$COMPARE
0002A0D3      R-DS$GT_VHEX
0002A0F4      R-SCAN$DECIMAL
0002A0F9      R-SCAN$OCTAL
0002A0FE      R-SCAN$BINARY
0002A103      R-SCAN$HEX
0002A106      R-SCAN$NUMERIC
0002A1A2      R-SCAN$SEARCHLIST
0002A1F1      R-SCAN$DEVICE
0002A251      R-BREAKUP_FILE
0002A380      R-DSX$LOADPCS
0002A38A      R-DSV$INITPCS
0002A39A      R-DSV$SETPAGE
0002A3C1      R-DSV$SHOWPAGE
0002A3E1      R-DSX$TYPE_OUT
0002A66D      R-DSX$PRINT
0002A6BB      R-DSX$PRINTS
```

```
Value                                   Symbols...
-----                                   ----------
0002A6C8     R-DSX$PRINTX
0002A6D5     R-DSX$PRINTB
0002A6ED     R-DSX$PRINTF
0002A702     R-DSX$PRINTI
0002A83C     R-DSX$PRINTREV
0002ACB8     R-DSX$CVTREG
0002AE74     R-QA$NORMAL_START
0002AF3F     R-QA$MULTIPLE_PASS
0002B00E     R-QA$LOOP_ON_TEST
0002B110     R-QA$LOOP_ON_SUBTEST
0002B239     R-QA$RUN_BACKWARDS
0002B314     R-VRSETQA
0002B3C1     R-VRSHOWQA
0002B44C     R-DSX$BRANCH
0002B5CA     R-QA$ADD_FORCED_ERROR
0002B5CD     R-QA$ADD_QA_ERROR
0002B76B     R-QA$FIND_USER_CALL_FRAME
0002B7EB     R-QA$MAIN
0002B9CC     R-QA$PRINT_FORCED_ERRORS
0002BA88     R-EXE$QIO
0002BBC6     R-QIO$INITIALIZE
0002BC1A     R-QIO$CLEANUP
0002BCE1     R-QIO$ADDUNIT
0002C18F     R-EXE$MAXACMODE
0002C19E     R-BUG$CHECK
0002C4DB     R-IOGEN$CNTRL_INI
0002C56C     R-QIO$DEALLOCATE
0002C5B1     R-EXE$PWRTIMCHK
0002C628     R-DSR$ALLOCATE_PSEUDO_RPB
0002C6A3     R-DSR$DEALLOCATE_PSEUDO_RPB
0002C6B6     R-DSX$CHECK_SUPPORT
0002C901     R-RMS$ERROR
0002C993     R-RMS$LOAD_XAB
0002C9E2     R-RMS$ALLOC_CACHE
0002CA32     R-RMS$INIT
0002CA41     R-RMS$CLEANUP
0002CAC5     R-RMS$OPEN
0002CE62     R-RMS$CONNECT
0002CEE4     R-RMS$GET
0002CFA6     R-RMS$READ
0002D064     R-RMS$DISCONNECT
0002D0BE     R-RMS$CLOSE
0002D205     R-RT11$OPEN
0002D2FB     R-RT11$GET
0002D40B     R-RT11$READ
0002D474     R-DSX$INITSCB
0002D49A     R-CMK$INITSCB
0002D53B     R-DSX$SETVEC
0002D61A     R-DSX$CLRVEC
0002D6BE     R-DSX$SETIPL
0002D6DA     R-CMK$SETIPL
0002D778     R-SCRIPT$INIT
```

```
Value                                   Symbols...
-----                                   ----------
0002D782      R-SCRIPT$OPEN
0002D8CF      R-SCRIPT$FLUSH
0002D911      R-SCRIPT$STOP
0002D91A      R-SCRIPT$CONT
0002D952      R-DS_SUPERLINE
0002D959      R-DS_GETLINE
0002DD88      R-DSR$CHECKLOAD
0002DDA7      R-DSV$SHOWMEMORY
0002E2A3      R-VRSETMEM
0002E301      R-VRSTART
0002E3A4      R-VRRESTART
0002E798      R-DSX$ABORT
0002E7BE      R-VRABORT
0002E7C2      R-DSI$ABORT
0002E8C3      R-VRSUPPORT
0002E91D      R-DSV$SHOWTESTS
0002E989      R-VRSHOWSECTIONS
0002EAC2      R-ULTRIX$OPEN
0002F23D      R-ULTRIX$GET
0002F54A      R-ULTRIX$READ
0002F5C1      R-DS$ENTRY
0002F5C9      R-MT$CHECK_ACCESS
0002F5CB      R-ERL$DEVICERR                  R-ERL$DEVICTMO                  R-ERL$RELEASEMB
              R-EXE$SNDEVMSG
0002F5CC      R-EXE$BUFFRQUOTA                R-EXE$BUFQUOPRC                 R-SCH$WAKE
0002F5D0      R-SCH$LOCKW
0002F5D7      R-SCH$UNLOCK
0002F5F1      R-LIB$CVT_DTB
0002F5F8      R-LIB$CVT_OTB
0002F5FF      R-LIB$CVT_HTB
0002F664      R-LIB$STOP
0002F66A      R-LIB$SIGNAL
0002F800      R-SCB_IMAGE
0002F900      R-BOO$AL_VECTOR
0002F924      R-BOO$GL_UMR_DIS
0002F928      R-BOO$GL_UCODE
0002F934      R-BOO$GL_UMR_TMPL
0002F938      R-BOO$GB_UMR_DP
0002F939      R-EXE$GB_CPUTYPE
0002F93A      R-EXE$GB_CPUDATA
0002F93E      R-EXE$GL_TENUSEC
0002F942      R-EXE$GL_UBDELAY
0002F948      R-EXE$GL_UMR_PHY
0002F94C      R-EXE$GL_UMR_VIR
0002F950      R-BOO$QIO
0002FAA8      R-BOO$MAP
0002FB8D      R-BOO$DRIVER
00030200      R-UD_INIT
000302E4      R-DS$GB_UDA50_INIT
00030450      R-UDA_RESPONSE
000304E8      R-TS11_DRIVER
00030800      R-TU_INIT
```

```
Value                                   Symbols...
-----                                   ----------
00030C8F     R-BOO$RESELECT                     R-BOO$SELECT
00030D00     R-ACP$ACCESS                       R-ACP$ACCESSNET
00030D0B     R-ACP$DEACCESS
00030D16     R-ACP$MODIFY
00030D21     R-ACP$MOUNT
00030D2C     R-ACP$READBLK
00030D49     R-ACP$WRITEBLK
00030E00     R-EXE$ASSIGN
00030F30     R-IOC$GETBYTE
00030F41     R-IOC$PUTBYTE
00030F52     R-IOC$INITBUFWIND
00030F59     R-IOC$MOVFRUSER
00030F5D     R-IOC$MOVFRUSER2
00030F6A     R-IOC$MOVFRUSER1
00030F71     R-IOC$MOVTOUSER
00030F75     R-IOC$MOVTOUSER2
00030F82     R-IOC$MOVTOUSER1
00030F89     R-IOC$FILSPT
00030F9B     R-EXE$CANCEL
000310A0     R-COM$DELATTNAST
000310D3     R-COM$FLUSHATTNS
00031103     R-COM$POST
00031113     R-COM$DRVDEALMEM
00031138     R-COM$SETATTNAST
00031198     R-DATETABLE
000311FD     R-EXE$ASCTIM
0003129E     R-EXE$BINTIM
00031494     R-EXE$NUMTIM
000315CA     R-EXE$DASSGN
0003168A     R-EXE$ALLOC
00031726     R-EXE$DALLOC
000317E4     R-DR$INT
000317FB     R-DR$INITIAL
00031804     R-EXE$IOFORK
00031809     R-EXE$FORK
00031828     R-EXE$FORKDSPTH
0003186C     R-EXE$GETTIM
00031888     R-EXE$GTCHAN
00031910     R-IOC$AL_SYSUCB
00031918     R-IOC$GL_DEVLIST
0003191C     R-IOC$GL_ADPLIST
00031920     R-IOC$GL_DPTLIST
00031928     R-IOC$CANCELIO
0003193F     R-IOC$DIAGBUFILL
00031970     R-IOC$RELSCHAN
0003197A     R-IOC$RELCHAN
000319BD     R-IOC$REQSCHANH
000319C7     R-IOC$REQSCHANL
000319D1     R-IOC$REQPCHANH
000319DA     R-IOC$REQPCHANL
00031A05     R-IOC$REQCOM
00031A48     R-IOC$INITIATE
```

```
Value                                   Symbols...
-----                                   ----------
00031A7B     R-IOC$RELDATAP
00031ACB     R-IOC$REQDATAPNW
00031ACF     R-IOC$REQDATAP
00031B08     R-IOC$RELMAPREG
00031B42     R-IOC$REQMAPREG
00031B54     R-IOC$ALTUBAMAP
00031B72     R-IOC$ALOUBAMAPN
00031B76     R-IOC$ALOUBAMAP
00031BD8     R-IOC$RETURN
00031BD9     R-IOC$WFIKPCH
00031BFB     R-IOC$WFIRLCH
00031C1F     R-IOC$ALLOSPT
00031C35     R-IOC$CVT_DEVNAM
00031C69     R-IOC$FFCHAN
00031C90     R-IOC$SEARCHDEV
00031C95     R-IOC$SEARCHGEN
00031D87     R-IOC$UNLOCK
00031D8B     R-IOC$VERIFYCHAN
00031DC0     R-IOC$APPLYECC
00031E34     R-IOC$CVTLOGPHY
00031E5B     R-IOC$MAPVBLK
00031EEE     R-IOC$UPDATRANSP
00031F2A     R-IOC$SENSEDISK
00031F38     R-IOC$LOADMBAMAP
00031F82     R-IOC$LOADUBAMAPA
00031F99     R-IOC$LOADUBAMAP
00032015     R-IOC$PTETOPFN
00032038     R-MBA$INT
000320E3     R-MBA$INITIAL
000320EC     R-DSX$PROBE
00032164     R-ACC$HANDLER
000321AC     R-EXE$ONEPARM
000321B2     R-EXE$ZEROPARM
000321B8     R-EXE$MODIFY
000321BE     R-EXE$READ
000321C7     R-EXE$WRITE
000321EA     R-EXE$READLOCK
000321ED     R-EXE$WRITELOCK
000321F0     R-EXE$MODIFYLOCK
000321F3     R-EXE$MODIFYLOCKR
000321FD     R-EXE$READLOCKR
00032204     R-EXE$WRITELOCKR
0003227A     R-EXE$READCHK
00032280     R-EXE$WRITECHK
0003228E     R-EXE$READCHKR
000322B8     R-EXE$WRITECHKR
000322ED     R-EXE$SETCHAR
0003230B     R-EXE$SETMODE
0003231B     R-EXE$SENSEMODE
00032343     R-EXE$CARRIAGE
00032393     R-VMS$QIO
00032526     R-EXE$ABORTIO
```

```
Value                                   Symbols...
-----                                   ----------
00032533     R-EXE$FINISHIOC
00032535     R-EXE$FINISHIO
0003254D     R-EXE$QIODRVPKT
00032551     R-EXE$ALTQUEPKT
00032573     R-EXE$QIOACPPKT
00032585     R-EXE$QIORETURN
0003258E     R-EXE$INSIOQ
000325A9     R-EXE$INSERTIRP
000325C4     R-IOC$INITDRV
000325CA     R-IOC$REINITDRV
00032678     R-EXE$UNWIND
00032800     R-AX_BUFF                          R-PHD_BASE
00032878     R-PCB_BASE
00032A00     R-DS$AX_JIB
00032A74     R-DS$AX_ARB
00032AEC     R-DS$AX_SOFTPCB
00032C00     R-SCB_BASE
00033000     R-SCB_UNKINT
00033400     R-STACK_BASE
00034A00     R-KSTKPTR
00035200     R-ISTKPTR
00035800     R-ESTKPTR
00035E00     R-SSTKPTR
00036400     R-POPT_BASE                        R-USTKPTR
00080000       DEV$M_MNT
00200000       DEV$M_DMT
00660000       IOC$K_DIAGPID
01000000       DEV$M_FOR                          KA$M_WCS_LD
02000000       DEV$M_SWL                          KA$M_SB_ERR
03000000       DS$GK_SID
10000000       TODR_BIAS
40000000       DEV$M_RCK                          IOC$K_IOSPACE
7FFEDE78       SYS$CRMPSC
7FFEDEB0       SYS$CRELOG
7FFEDEF0       SYS$DCLEXH
7FFEE0D8       SYS$GETJPI
7FFEE118       SYS$SRCHANDLER
7FFEE188       SYS$PUT
7FFEE1C8       SYS$CREATE
7FFEE218       SYS$SPACE
7FFEE230       SYS$PARSE
7FFEE248       SYS$SEARCH
7FFEE250       SYS$SETDDIR
7FFEE3F8       SYS$GETSYI
7FFEE410       SYS$GETDVI
7FFEE420       SYS$GETJPIW
80000000       DEV$M_WCK
800000C8       SYS$CRETVA
80000110       SYS$DELTVA
80000208       SYS$SETEXV
```

```
Value                                   Symbols...
-----                                   ----------


          Key for special characters above:
                +-------------------+
                ! *  - Undefined    !
                ! U  - Universal    !
                ! R  - Relocatable  !
                ! X  - External     !
                ! V  - Vectored     !
                +-------------------+
```

```
                                    +-----------------+
                                    ! Image Synopsis !
                                    +-----------------+

Virtual memory allocated:            00010000 000329FF 00022A00 (141824. bytes, 277. pages)
Stack size:                                20. pages
Image header virtual block limits:          1.       1. (    1. block)
Image binary virtual block limits:          2.     278. (  277. blocks)
Image name and identification:       ENSAA 10.10
Number of files:                            5.
Number of modules:                         95.
Number of program sections:                20.
Number of global symbols:                1248.
Number of cross references:              2962.
Number of image sections:                   6.
User transfer address:               0002F5C1
Image type:                          EXECUTABLE.
Map format:                          FULL WITH CROSS REFERENCE in file DISK$DS:[DS.LOCAL.RELEASE.WORK]ENSAA.MAP;811
Estimated map length:                374. blocks
                                  +----------------------+
                                  ! Link Run Statistics !
                                  +----------------------+

Performance Indicators               Page Faults   CPU Time        Elapsed Time
----------------------               -----------   --------        ------------
    Command processing:                       87   00:00:00.12     00:00:03.28
    Pass 1:                                  382   00:00:02.49     00:00:20.82
    Allocation/Relocation:                    92   00:00:00.14     00:00:00.36
    Pass 2:                                  299   00:00:02.38     00:00:21.18
    Map data after object module synopsis:    19   00:00:01.48     00:00:02.62
    Symbol table output:                       0   00:00:00.01     00:00:00.17
Total run values:                            879   00:00:06.62     00:00:48.43

Using a working set limited to 2512 pages and 739 pages of data storage (excluding image)

Total number object records read (both passes):   12130
    of which 6065 were in libraries and 3694 were DEBUG data records containing 1369321 bytes

Number of modules extracted explicitly              = 89
    with 6 extracted to resolve undefined symbols

219 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

LINK/NODEBUG/NOTRACEBACK/NOSYSSHR/CROSS/FULL/MAP=ENSAA.MAP;811/EXE=DS$R$W:ENSAA.EXE;811 DISK$DS:[DS.LOCAL.OPTIONS]ENSAA/OPT
OBJECT_LIBRARY/INCLUDE=( -
KERNEL, -
ACPFDT, ANSI, APBOOT, APT, ASSIGN, ASTCON, ASTDEL, ATTACH, -
BOOTDRIVR, BOOTIO, BUFCTL, BUFFER, -
CALLFRAME, CANCEL, CHAN730, CHMK, CLI, CLOCK, COMDRVSUB, CONFIG, CONSOLE, -
CRDIMAGE, CSDDBTDRV, DSVECS, CVRTIM, CVTFILNAM, -
DASSGN, DEBUG, DEVALC, DEVICE, DIRECTORY, DISPAT, DLBTDRIVR, DMBTDRIVR, DRINT, -
DSLOAD, DQBTDRIVR, -
ENTRY, ERROR, EVENT, EXCEPT, -
```

```
FAO, FLAGS, FILEREAD, FRKCTL, -
GETTIM, GTCHAN, -
HELP, -
IOBASE, IOPOST, IOSNPG, IOSPGD, IOSRAM, -
LOAD, LODMAP, LOOP, -
MCHK730, MBAINT, MEMALC, MEMMGT, MPDUMMY, -
ODS, ONLY730, -
PARAM, PARSE, PCSLOAD, PRINT, PROBE, PUBTDRIVR, -
QACHECKS, QAFLAGS, QAMAIN, QIO, QIOFDT, QIOREQ, -
RELOCDRV, RMS, RT11, -
SCB, SCRIPT, SHOWMEMORY, START, -
TSBTDRIVR, MUBTDRIVR, -
ULTRIX, UNWIND, -
VER730, VMSDUMMY)
!*SYS$LIBRARY:MOUNTSHR/SHAREABLE
!*SYS$LIBRARY:DISMNTSHR/SHAREABLE
BASE=%X10000
DZRO_MIN=100
```

Table of contents

(1) 100 ACCESS AND CREATE ACP FUNCTION PROCESSING ;G:4
(1) 134 DEACCESS ACP FUNCTION PROCESSING ;G:4
(1) 165 DELETE AND MODIFY ACP FUNCTION PROCESSING ;G:4
(1) 196 MOUNT ACP FUNCTION PROCESSING ;G:4
(1) 228 READ AND WRITE BLOCK ACP FUNCTION PROCESSING ;G:4

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ACPFDT.MAR
                        0000     2 ; *4    12-MAY-1986 09:46:45 BARANSKI "Cleaning up Object Libraries."
                        0000     3 ; *3     7-MAY-1986 12:56:53 BARANSKI "<no reason given>"
                        0000     4 ; *2    27-MAR-1986 09:59:50 MUSE "<no reason given>"
                        0000     5 ; *1     6-FEB-1986 15:13:21 DS ""
                        0000     6 ; DEC/CMS REPLACEMENT HISTORY, Element ACPFDT.MAR
                        0000     7         .TITLE  ACPFDT  *** ACPFDT ACP function dispatch                    ;G:4
                        0000     8         .IDENT  "6.1-3"                                                    ;G:4
                    00000000     9         .PSECT  SEP, SHR, EXE, WRT, LONG                                   ;G:4
                        0000    10                                                                            ;G:4
                        0000    11 ;                                                                          ;G:4
                        0000    12 ;*************************************************************************** ;G:4
                        0000    13 ;*                                                                         * ;G:4
                        0000    14 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                   * ;G:4
                        0000    15 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                        * ;G:4
                        0000    16 ;*                                                                         * ;G:4
                        0000    17 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED  * ;G:4
                        0000    18 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  * ;G:4
                        0000    19 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   * ;G:4
                        0000    20 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  * ;G:4
                        0000    21 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY  * ;G:4
                        0000    22 ;*  TRANSFERRED.                                                           * ;G:4
                        0000    23 ;*                                                                         * ;G:4
                        0000    24 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE  * ;G:4
                        0000    25 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  * ;G:4
                        0000    26 ;*  CORPORATION.                                                           * ;G:4
                        0000    27 ;*                                                                         * ;G:4
                        0000    28 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS  * ;G:4
                        0000    29 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 * ;G:4
                        0000    30 ;*                                                                         * ;G:4
                        0000    31 ;*************************************************************************** ;G:4
                        0000    32 ;                                                                          ;G:4
                        0000    33 ; D. N. CUTLER 22-NOV-76                                                   ;G:4
                        0000    34 ;                                                                          ;G:4
                        0000    35 ; MODIFIED BY:                                                             ;G:4
                        0000    36 ;                                                                          ;G:4
                        0000    37 ;       V0002   ACG23542        Andrew C. Goldstein,  4-May-1979  14:43    ;G:4
                        0000    38 ;               Check LBN of mapped virtual I/O against UCB$L_MAXBLOCK     ;G:4
                        0000    39 ;                                                                          ;G:4
                        0000    40 ;       V0003   ACG0047         Andrew C. Goldstein,    8-Aug-1979  17:23    ;G:4
                        0000    41 ;               Interface change to protection check routines              ;G:4
                        0000    42 ;                                                                          ;G:4
                        0000    43 ;       V0004   RIH0033         Richard I. Hustvedt     16-Oct-1979     15:55 ;G:4
                        0000    44 ;               Change PCB$W_BYTCNT to JIB$L_BYTCNT; PCB$W_BYTLM to JIB$L_BYTLM; ;G:
                        0000    45 ;               PCB$W_FILCNT to JIB$W_FILCNT.                              ;G:4
                        0000    46 ;                                                                          ;G:4
                        0000    47 ;       V0005   RIH0036         Richard I. Hustvedt     02-Nov-1979        ;G:4
                        0000    48 ;               Set IRP$V_FILACP for file acp requests and bump DIO count  ;G:4
                        0000    49 ;                                                                          ;G:4
                        0000    50 ;       V0006   ACG0100         Andrew C. Goldstein,    3-Jan-1980  17:19    ;G:4
                        0000    51 ;               Permit partial dismount of volume sets                     ;G:4
                        0000    52 ;                                                                          ;G:4
                        0000    53 ;               Dave Butenhof   13-may-1980, Version 5.4                   ;G:4
                        0000    54 ;       02      Modify VMS V2.0 source for Supervisor environment.         ;G:4
                        0000    55 ;                                                                          ;G:4
                        0000    56 ;                                                                          ;G:4
                        0000    57 ;               Dave Butenhof, 08-oct-1980, version 6.1                    ;G:4
```

```
                        0000     58 ;       03      Changed some word offsets to longword.                          ;G:4
                        0000     59 ;**                                                                             ;G:4
                        0000     60 ;                                                                               ;G:4
                        0000     61 ; ACP FUNCTION DECISION TABLE ACTION ROUTINES                                   ;G:4
                        0000     62 ;                                                                               ;G:4
                        0000     63 ; MACRO LIBRARY CALLS                                                           ;G:4
                        0000     64 ;                                                                               ;G:4
                        0000     65                                                                                 ;G:4
                        0000     66          $CCBDEF                             ;DEFINE CCB OFFSETS            ;G:4
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,CCB$B_STS,   CCB$B_STS:
                00000009 0008                        .BLKB   1
                        0009                 .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
                0000000A 0009                        .BLKB   1
                        000A                 .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
                0000000C 000A                        .BLKW   1
                        000C                 .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
                00000010 000C                        .BLKL   1
                        0010                 .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                        0010                 .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                      00000000               .RESTORE
                        0000     67          $DEVDEF                             ;DEFINE DEVICE CHARACTERISTICS BITS ;G:4
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 0010                .=0
                      00000000               .RESTORE
                        0000     68          $DYNDEF                             ;DEFINE STRUCTURE TYPE CODES   ;G:4
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 0010                .=0
                      00000000               .RESTORE
                        0000     69          $IPLDEF                             ;DEFINE INTERRUPT PRIORITY LEVELS ;G:4
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 0010                .=0
                      00000000               .RESTORE
                        0000     70          $IODEF                              ;DEFINE I/O FUNCTION CODES     ;G:4
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 0010                .=0
                      00000000               .RESTORE
                        0000     71          $IRPDEF                             ;DEFINE IRP OFFSETS            ;G:4
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 0010                .=0
                        0000                 .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
                0000000B 000A                        .BLKB   1
```

```
                    000B            .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
0000000C  000B                              .BLKB   1
                    000C            .IIF    NB,IRP$L_PID,   IRP$L_PID:
00000010  000C                              .BLKL   1
                    0010            .IIF    NB,IRP$L_AST,   IRP$L_AST:
00000014  0010                              .BLKL   1
                    0014            .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
00000018  0014                              .BLKL   1
                    0018            .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
0000001C  0018                              .BLKL   1
                    001C            .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
00000020  001C                              .BLKL   1
                    0020            .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
00000022  0020                              .BLKW   1
                    0022            .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
00000023  0022                              .BLKB   1
                    0023            .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
00000024  0023                              .BLKB   1
                    0024            .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
00000028  0024                              .BLKL   1
                    0028            .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
0000002A  0028                              .BLKW   1
                    002A            .IIF    NB,IRP$W_STS,   IRP$W_STS:
0000002C  002A                              .BLKW   1
                    002C            .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
00000030  002C                              .BLKL   1
                    0030            .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
00000032  0030                              .BLKW   1
                    0032            .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
00000034  0032                              .BLKW   1
                    0034            .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
                    0034            .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
00000038  0034                              .BLKL   1
                    0038            .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
                    0038            .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
                    0038            .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
00000039  0038                              .BLKB   1
0000003C  0039                              .BLKB   3
                    003C            .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
                    003C            .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
                    003C            .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                              .BLKW   1
                    003E            .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                              .BLKW   1
                    0040            .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                              .BLKL   1
                    0044            .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                              .BLKL   1
                    0048            .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                              .BLKL   1
                    004C            .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                              .BLKL   1
                    0050            .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                              .BLKL   1
00000058  0054                              .BLKL   1
0000005C  0058                              .BLKL   1
                    005C            .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
```

```
                        005C                .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
                    00000000                .RESTORE
                        0000    72          $JIBDEF                         ;DEFINE JIB OFFSETS                 ;G:4
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                    00000000  005C          .=0
                    00000000                .RESTORE
                        0000    73          $PCBDEF                         ;DEFINE PCB OFFSETS                 ;G:4
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                    00000000  005C          .=0
                        0000                .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
                    00000004  0000                      .BLKL   1
                        0004                .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
                    00000008  0004                      .BLKL   1
                        0008                .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
                    0000000A  0008                      .BLKW   1
                        000A                .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
                    0000000B  000A                      .BLKB   1
                        000B                .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
                    0000000C  000B                      .BLKB   1
                        000C                .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
                    0000000D  000C                      .BLKB   1
                        000D                .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
                    0000000E  000D                      .BLKB   1
                        000E                .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
                    00000010  000E                      .BLKW   1
                        0010                .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
                    00000014  0010                      .BLKL   1
                        0014                .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
                    00000018  0014                      .BLKL   1
                        0018                .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
                    0000001C  0018                      .BLKL   1
                        001C                .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
                    00000020  001C                      .BLKL   1
                        0020                .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
                    00000024  0020                      .BLKL   1
                        0024                .IIF    NB,PCB$L_STS,   PCB$L_STS:
                    00000028  0024                      .BLKL   1
                        0028                .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
                    0000002C  0028                      .BLKL   1
                        002C                .IIF    NB,PCB$W_STATE, PCB$W_STATE:
                    0000002E  002C                      .BLKW   1
                        002E                .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
                    0000002F  002E                      .BLKB   1
                        002F                .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
                    00000030  002F                      .BLKB   1
                        0030                .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
                    00000032  0030                      .BLKW   1
                        0032                .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
                    00000034  0032                      .BLKW   1
                        0034                .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
                    00000036  0034                      .BLKW   1
                        0036                .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
                    00000038  0036                      .BLKW   1
                        0038                .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
                    0000003A  0038                      .BLKW   1
```

```
                        003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                                  .BLKW   1
                        003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                                  .BLKW   1
                        003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                                  .BLKW   1
                        0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                                  .BLKW   1
                        0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                                  .BLKW   1
                        0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                                  .BLKB   8
                        004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                        004C            .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                                  .BLKL   1
                        0050            .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                                  .BLKL   1
                        0054            .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                                  .BLKL   1
                        0058            .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                                  .BLKL   1
                        005C            .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                                  .BLKL   1
                        0060            .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                                  .BLKL   1
                        0064            .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                                  .BLKL   1
                        0068            .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                                  .BLKB   16
                        0078            .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                                  .BLKL   1
                        007C            .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084  007C                                  .BLKQ   1
                        0084            .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                                  .BLKL   1
                        0088            .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                        0088            .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                                  .BLKW   1
                        008A            .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                                  .BLKW   1
                        008C            .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                        008C            .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                00000000                .RESTORE
                        0000    74      $PRDEF                          ;DEFINE PROCESSOR REGISTERS             ;G:4
                        0000            .SAVE   LOCAL_BLOCK
                        0000            .PSECT  $ABS$,ABS
00000000  008C                          .=0
                00000000                .RESTORE
                        0000    75      $PRVDEF                         ;DEFINE PRIVILEGE BITS                  ;G:4
                        0000            .SAVE   LOCAL_BLOCK
                        0000            .PSECT  $ABS$,ABS
00000000  008C                          .=0
                00000000                .RESTORE
                        0000    76      $UCBDEF                         ;DEFINE UCB OFFSETS                     ;G:4
                        0000            .SAVE   LOCAL_BLOCK
                        0000            .PSECT  $ABS$,ABS
00000000  008C                          .=0
```

```
                    0000            .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                    0000            .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004            0000                    .BLKL   1
                    0004            .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                    0004            .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008            0004                    .BLKL   1
                    0008            .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
0000000A            0008                    .BLKW   1
                    000A            .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
0000000B            000A                    .BLKB   1
                    000B            .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
0000000C            000B                    .BLKB   1
                    000C            .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C            .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C            .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
0000000D            000C                    .BLKB   1
00000010            000D                    .BLKB   3
                    0010            .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010            .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014            0010                    .BLKL   1
                    0014            .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014            .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014            .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016            0014                    .BLKW   1
                    0016            .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018            0016                    .BLKW   1
                    0018            .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018            .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
0000001A            0018                    .BLKW   1
                    001A            .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A            .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
0000001C            001A                    .BLKW   1
                    001C            .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
00000020            001C                    .BLKL   1
                    0020            .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
00000024            0020                    .BLKL   1
                    0024            .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
00000028            0024                    .BLKL   1
                    0028            .IIF    NB,UCB$L_PID,   UCB$L_PID:
0000002C            0028                    .BLKL   1
                    002C            .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030            002C                    .BLKL   1
                    0030            .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034            0030                    .BLKL   1
                    0034            .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038            0034                    .BLKL   1
                    0038            .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039            0038                    .BLKB   1
                    0039            .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A            0039                    .BLKB   1
                    003A            .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C            003A                    .BLKW   1
                    003C            .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C            .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C            .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D            003C                    .BLKB   1
                    003D            .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
```

```
                     003D                .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E             003D                        .BLKB   1
                     003E                .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                     003E                .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F             003E                        .BLKB   1
                     003F                .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040             003F                        .BLKB   1
                     0040                .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044             0040                        .BLKL   1
                     0044                .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048             0044                        .BLKL   1
                     0048                .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A             0048                        .BLKW   1
                     004A                .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                     004A                .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B             004A                        .BLKB   1
                     004B                .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C             004B                        .BLKB   1
                     004C                .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050             004C                        .BLKL   1
                     0050                .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052             0050                        .BLKW   1
                     0052                .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                     0052                .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053             0052                        .BLKB   1
                     0053                .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054             0053                        .BLKB   1
                     0054                .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058             0054                        .BLKL   1
                     0058                .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A             0058                        .BLKW   1
                     005A                .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C             005A                        .BLKW   1
                     005C                .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                     005C                .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060             005C                        .BLKL   1
                     0060                .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064             0060                        .BLKL   1
                     0064                .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                     0064                .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068             0064                        .BLKL   1
                     0068                .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C             0068                        .BLKL   1
                     006C                .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E             006C                        .BLKW   1
                     006E                .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070             006E                        .BLKW   1
                     0070                .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071             0070                        .BLKB   1
                     0071                .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072             0071                        .BLKB   1
                     0072                .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074             0072                        .BLKW   1
                     0074                .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                     0074                .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000             0074                . = 0
                     0000                .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
```

```
00000002  0000                                  .BLKW   1
00000074  0002                  . = 116
          0074                  .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                                  .BLKL   1
          0080                  .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                                  .BLKL   1
          0090                  .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
          0090                  .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                  . = 116
          0074                  .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                                  .BLKB   1
          0075                  .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                                  .BLKB   1
          0076                  .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                                  .BLKB   1
          0077                  .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                                  .BLKB   1
          0078                  .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                                  .BLKL   1
0000007E  007C                                  .BLKW   1
          007E                  .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                                  .BLKW   1
          0080                  .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                                  .BLKL   1
00000080  0084                  . = 128
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                                  .BLKW   1
          008A                  .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                                  .BLKW   1
          008C                  .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C                  .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                  .BLKW   1
          008E                  .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                  .BLKW   1
          0090                  .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                  .BLKW   1
          0092                  .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                  .BLKW   1
          0094                  .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                  .BLKB   1
          0095                  .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                  .BLKB   1
          0096                  .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                  .BLKW   1
```

```
00000096  0098                . = 150
00000098  0096                        .BLKW   1
          0098                .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                        .BLKL   1
          009C                .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                        .BLKL   1
          00A0                .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                        .BLKL   1
          00A4                .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                        .BLKW   1
          00A6                .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                        .BLKB   1
000000A8  00A7                        .BLKB   1
00000074  00A8                . = 116
00000078  0074                        .BLKL   1
0000007C  0078                        .BLKL   1
00000080  007C                        .BLKL   1
00000084  0080                        .BLKL   1
00000088  0084                        .BLKL   1
0000008C  0088                        .BLKL   1
          008C                .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                        .BLKL   1
00000094  0090                        .BLKL   1
00000098  0094                        .BLKL   1
0000009C  0098                        .BLKL   1
          009C                .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                        .BLKB   1
          009D                .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                        .BLKB   1
          009E                .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                        .BLKB   1
000000A0  009F                        .BLKB   1
          00A0                .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                        .BLKB   1
          00A1                .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                        .BLKB   1
          00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                        .BLKB   1
000000A4  00A3                        .BLKB   1
          00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                        .BLKB   1
          00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                        .BLKW   1
000000A8  00A7                        .BLKB   1
          00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                        .BLKL   1
000000B0  00AC                        .BLKL   1
000000B4  00B0                        .BLKL   1
000000B8  00B4                        .BLKL   1
          00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                        .BLKL   1
          00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC                . = 116
          0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                        .BLKL   1
          0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
```

```
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                  .BLKW   1
00000080  007E                                  .BLKW   1
      00000000                  .RESTORE
          0000    77            $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS        ;G:4
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
      00000000                  .RESTORE
          0000    78            $VCBDEF                         ;DEFINE VCB OFFSETS                   ;G:4
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .IIF    NB,VCB$L_FCBFL, VCB$L_FCBFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,VCB$L_FCBBL, VCB$L_FCBBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,VCB$W_SIZE,  VCB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,VCB$B_TYPE,  VCB$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,VCB$C_MRKLEN,        VCB$C_MRKLEN:
          000B                  .IIF    NB,VCB$K_MRKLEN,        VCB$K_MRKLEN:
          000B                  .IIF    NB,VCB$B_STATUS,        VCB$B_STATUS:
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,VCB$W_TRANS, VCB$W_TRANS:
0000000E  000C                                  .BLKW   1
          000E                  .IIF    NB,VCB$W_RVN,   VCB$W_RVN:
00000010  000E                                  .BLKW   1
          0010                  .IIF    NB,VCB$L_AQB,   VCB$L_AQB:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,VCB$T_VOLNAME,       VCB$T_VOLNAME:
00000020  0014                                  .BLKB   12
          0020                  .IIF    NB,VCB$L_RVT,   VCB$L_RVT:
00000024  0020                                  .BLKL   1
          0024                  .IIF    NB,VCB$C_COMLEN,        VCB$C_COMLEN:
          0024                  .IIF    NB,VCB$K_COMLEN,        VCB$K_COMLEN:
          0024                  .IIF    NB,VCB$L_HOMELBN,       VCB$L_HOMELBN:
00000028  0024                                  .BLKL   1
          0028                  .IIF    NB,VCB$L_HOME2LBN,      VCB$L_HOME2LBN:
0000002C  0028                                  .BLKL   1
          002C                  .IIF    NB,VCB$L_IXHDR2LBN,     VCB$L_IXHDR2LBN:
00000030  002C                                  .BLKL   1
          0030                  .IIF    NB,VCB$L_IBMAPLBN,      VCB$L_IBMAPLBN:
00000034  0030                                  .BLKL   1
          0034                  .IIF    NB,VCB$L_SBMAPLBN,      VCB$L_SBMAPLBN:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,VCB$B_IBMAPSIZE,     VCB$B_IBMAPSIZE:
00000039  0038                                  .BLKB   1
          0039                  .IIF    NB,VCB$B_SBMAPSIZE,     VCB$B_SBMAPSIZE:
0000003A  0039                                  .BLKB   1
          003A                  .IIF    NB,VCB$B_IBMAPVBN,      VCB$B_IBMAPVBN:
0000003B  003A                                  .BLKB   1
          003B                  .IIF    NB,VCB$B_SBMAPVBN,      VCB$B_SBMAPVBN:
0000003C  003B                                  .BLKB   1
          003C                  .IIF    NB,VCB$W_CLUSTER,       VCB$W_CLUSTER:
```

```
0000003E  003C                          .BLKW   1
          003E          .IIF    NB,VCB$W_EXTEND,        VCB$W_EXTEND:
00000040  003E                          .BLKW   1
          0040          .IIF    NB,VCB$L_FREE,  VCB$L_FREE:
00000044  0040                          .BLKL   1
          0044          .IIF    NB,VCB$L_MAXFILES,      VCB$L_MAXFILES:
00000048  0044                          .BLKL   1
          0048          .IIF    NB,VCB$B_WINDOW,        VCB$B_WINDOW:
00000049  0048                          .BLKB   1
          0049          .IIF    NB,VCB$B_LRU_LIM,       VCB$B_LRU_LIM:
0000004A  0049                          .BLKB   1
          004A          .IIF    NB,VCB$W_FILEPROT,      VCB$W_FILEPROT:
0000004C  004A                          .BLKW   1
          004C          .IIF    NB,VCB$W_MCOUNT,        VCB$W_MCOUNT:
0000004E  004C                          .BLKW   1
          004E          .IIF    NB,VCB$B_EOFDELTA,      VCB$B_EOFDELTA:
0000004F  004E                          .BLKB   1
          004F          .IIF    NB,VCB$B_RESFILES,      VCB$B_RESFILES:
00000050  004F                          .BLKB   1
          0050          .IIF    NB,VCB$W_RECORDSZ,      VCB$W_RECORDSZ:
00000052  0050                          .BLKW   1
          0052          .IIF    NB,VCB$B_BLOCKFACT,     VCB$B_BLOCKFACT:
00000053  0052                          .BLKB   1
          0053          .IIF    NB,VCB$B_STATUS2,       VCB$B_STATUS2:
00000054  0053                          .BLKB   1
          0054          .IIF    NB,VCB$L_QUOTAFCB,      VCB$L_QUOTAFCB:
00000058  0054                          .BLKL   1
          0058          .IIF    NB,VCB$L_CACHE, VCB$L_CACHE:
0000005C  0058                          .BLKL   1
          005C          .IIF    NB,VCB$L_QUOCACHE,      VCB$L_QUOCACHE:
00000060  005C                          .BLKL   1
          0060          .IIF    NB,VCB$W_QUOSIZE,       VCB$W_QUOSIZE:
00000062  0060                          .BLKW   1
          0062          .IIF    NB,VCB$W_PENDERR,       VCB$W_PENDERR:
00000064  0062                          .BLKW   1
          0064          .IIF    NB,VCB$C_LENGTH,        VCB$C_LENGTH:
          0064          .IIF    NB,VCB$K_LENGTH,        VCB$K_LENGTH:
00000000  0064          . = 0
          0000          .IIF    NB,VCB$L_BLOCKFL,       VCB$L_BLOCKFL:
00000004  0000                          .BLKL   1
          0004          .IIF    NB,VCB$L_BLOCKBL,       VCB$L_BLOCKBL:
00000008  0004                          .BLKL   1
0000000B  0008          . = 11
00000024  000B          . = 36
          0024          .IIF    NB,VCB$L_CUR_FID,       VCB$L_CUR_FID:
          0024          .IIF    NB,VCB$W_CUR_NUM,       VCB$W_CUR_NUM:
00000026  0024                          .BLKW   1
          0026          .IIF    NB,VCB$W_CUR_SEQ,       VCB$W_CUR_SEQ:
00000028  0026                          .BLKW   1
          0028          .IIF    NB,VCB$L_START_FID,     VCB$L_START_FID:
          0028          .IIF    NB,VCB$W_START_NUM,     VCB$W_START_NUM:
0000002A  0028                          .BLKW   1
          002A          .IIF    NB,VCB$W_START_SEQ,     VCB$W_START_SEQ:
0000002C  002A                          .BLKW   1
          002C          .IIF    NB,VCB$W_MODE,  VCB$W_MODE:
0000002E  002C                          .BLKW   1
          002E          .IIF    NB,VCB$B_TM,    VCB$B_TM:
```

```
0000002F  002E                          .BLKB   1
                  002F          .IIF    NB,VCB$B_CUR_RVN,       VCB$B_CUR_RVN:
00000030  002F                          .BLKB   1
                  0030          .IIF    NB,VCB$L_ST_RECORD,     VCB$L_ST_RECORD:
00000034  0030                          .BLKL   1
                  0034          .IIF    NB,VCB$L_MVL,   VCB$L_MVL:
00000038  0034                          .BLKL   1
                  0038          .IIF    NB,VCB$L_WCB,   VCB$L_WCB:
0000003C  0038                          .BLKL   1
                  003C          .IIF    NB,VCB$L_VPFL,  VCB$L_VPFL:
00000040  003C                          .BLKL   1
                  0040          .IIF    NB,VCB$L_VPBL,  VCB$L_VPBL:
00000044  0040                          .BLKL   1
                  0044          .IIF    NB,VCB$L_USRLBLAST,     VCB$L_USRLBLAST:
00000048  0044                          .BLKL   1
0000000B  0048          . = 11
                  000B          .IIF    NB,VCB$B_QNAMECNT,      VCB$B_QNAMECNT:
0000000C  000B                          .BLKB   1
                  000C          .IIF    NB,VCB$T_QNAME, VCB$T_QNAME:
00000020  000C                          .BLKB   20
        00000000          .RESTORE
                  0000    79    $WCBDEF                         ;DEFINE WCB OFFSETS                     ;G:4
                  0000          .SAVE   LOCAL_BLOCK
                  0000          .PSECT  $ABS$,ABS
00000000  00BC          .=0
                  0000          .IIF    NB,WCB$L_WLFL,  WCB$L_WLFL:
00000004  0000                          .BLKL   1
                  0004          .IIF    NB,WCB$L_WLBL,  WCB$L_WLBL:
00000008  0004                          .BLKL   1
                  0008          .IIF    NB,WCB$W_SIZE,  WCB$W_SIZE:
0000000A  0008                          .BLKW   1
                  000A          .IIF    NB,WCB$B_TYPE,  WCB$B_TYPE:
0000000B  000A                          .BLKB   1
                  000B          .IIF    NB,WCB$B_ACCESS,        WCB$B_ACCESS:
0000000C  000B                          .BLKB   1
                  000C          .IIF    NB,WCB$L_PID,   WCB$L_PID:
0000000E  000C                          .BLKB   2
                  000E          .IIF    NB,WCB$W_REFCNT,        WCB$W_REFCNT:
00000010  000E                          .BLKW   1
                  0010          .IIF    NB,WCB$L_ORGUCB,        WCB$L_ORGUCB:
00000014  0010                          .BLKL   1
                  0014          .IIF    NB,WCB$W_ACON,  WCB$W_ACON:
00000016  0014                          .BLKW   1
                  0016          .IIF    NB,WCB$W_NMAP,  WCB$W_NMAP:
00000018  0016                          .BLKW   1
                  0018          .IIF    NB,WCB$L_FCB,   WCB$L_FCB:
0000001C  0018                          .BLKL   1
                  001C          .IIF    NB,WCB$L_RVT,   WCB$L_RVT:
00000020  001C                          .BLKL   1
                  0020          .IIF    NB,WCB$L_STVBN, WCB$L_STVBN:
00000024  0020                          .BLKL   1
                  0024          .IIF    NB,WCB$C_MAP,   WCB$C_MAP:
                  0024          .IIF    NB,WCB$K_MAP,   WCB$K_MAP:
                  0024          .IIF    NB,WCB$C_LENGTH,        WCB$C_LENGTH:
                  0024          .IIF    NB,WCB$K_LENGTH,        WCB$K_LENGTH:
                  0024          .IIF    NB,WCB$W_P1_COUNT,      WCB$W_P1_COUNT:
00000026  0024                          .BLKW   1
```

```
                     0026                .IIF    NB,WCB$L_P1_LBN,        WCB$L_P1_LBN:
0000002A             0026                                .BLKL   1
                     002A                .IIF    NB,WCB$W_P2_COUNT,      WCB$W_P2_COUNT:
0000002C             002A                                .BLKW   1
                     002C                .IIF    NB,WCB$L_P2_LBN,        WCB$L_P2_LBN:
00000030             002C                                .BLKL   1
00000000             0030                . = 0
                     0000                .IIF    NB,WCB$W_COUNT, WCB$W_COUNT:
00000002             0000                                .BLKW   1
                     0002                .IIF    NB,WCB$L_LBN,   WCB$L_LBN:
00000006             0002                                .BLKL   1
00000000             0006                . = 0
FFFFFFFA             0000                                .BLKB   -6
                     FFFA                .IIF    NB,WCB$W_PREVCOUNT,     WCB$W_PREVCOUNT:
FFFFFFFC             FFFA                                .BLKW   1
                     FFFC                .IIF    NB,WCB$L_PREVLBN,       WCB$L_PREVLBN:
00000000             FFFC                                .BLKL   1
                00000000                 .RESTORE
                     0000     80                                                                 ;G:4
                     0000     81 ;                                                               ;G:4
                     0000     82 ; LOCAL SYMBOLS                                                 ;G:4
                     0000     83 ;                                                               ;G:4
                     0000     84 ; ARGUMENT LIST OFFSET DEFINITIONS                              ;G:4
                     0000     85 ;                                                               ;G:4
                     0000     86                                                                 ;G:4
00000000             0000     87 P1=0                                ;FIRST FUNCTION DEPENDENT PARAMETER   ;G:4
00000004             0000     88 P2=4                                ;SECOND FUNCTION DEPENDENT PARAMETER  ;G:4
00000008             0000     89 P3=8                                ;THIRD FUNCTION DEPENDENT PARAMETER   ;G:4
0000000C             0000     90 P4=12                               ;FOURTH FUNCTION DEPENDENT PARAMETER  ;G:4
00000010             0000     91 P5=16                               ;FIFTH FUNCTION DEPENDENT PARAMETER   ;G:4
00000014             0000     92 P6=20                               ;SIXTH FUNCTION DEPENDENT PARAMETER   ;G:4
                     0000     93                                                                 ;G:4
                     0000     94 ;                                                               ;G:4
                     0000     95 ; MAXIMUM NUMBER OF ACP DESCRIPTORS                             ;G:4
                     0000     96 ;                                                               ;G:4
                     0000     97                                                                 ;G:4
00000016             0000     98 MXDESCR=5+17                        ;MAXIMUM NUMBER OF ACP DESCRIPTORS    ;G:4
```

```
                     0000    100          .SBTTL  ACCESS AND CREATE ACP FUNCTION PROCESSING                    ;G:4
                     0000    101 ;+                                                                           ;G:4
                     0000    102 ; ACP$ACCESS - ACCESS AND CREATE ACP FUNCTION PROCESSING                     ;G:4
                     0000    103 ; ACP$ACCESSNET - ACCESS (CONNECT) TO NETWORK FUNCTION PROCESSING            ;G:4
                     0000    104 ;                                                                            ;G:4
                     0000    105 ; ***TBS***                                                                  ;G:4
                     0000    106 ;                                                                            ;G:4
                     0000    107 ; INPUTS:                                                                    ;G:4
                     0000    108 ;                                                                            ;G:4
                     0000    109 ;       R0 = SCRATCH.                                                        ;G:4
                     0000    110 ;       R1 = SCRATCH.                                                        ;G:4
                     0000    111 ;       R2 = SCRATCH.                                                        ;G:4
                     0000    112 ;       R3 = ADDRESS OF I/O REQUEST PACKET.                                  ;G:4
                     0000    113 ;       R4 = CURRENT PROCESS PCB ADDRESS.                                    ;G:4
                     0000    114 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.                                    ;G:4
                     0000    115 ;       R6 = ADDRESS OF CCB.                                                 ;G:4
                     0000    116 ;       R7 = I/O FUNCTION CODE BIT NUMBER.                                   ;G:4
                     0000    117 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.                       ;G:4
                     0000    118 ;       R9 = SCRATCH.                                                        ;G:4
                     0000    119 ;       R10 = SCRATCH.                                                       ;G:4
                     0000    120 ;       R11 = SCRATCH.                                                       ;G:4
                     0000    121 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.                  ;G:4
                     0000    122 ;                                                                            ;G:4
                     0000    123 ; OUTPUTS:                                                                   ;G:4
                     0000    124 ;                                                                            ;G:4
                     0000    125 ; ***TBS***                                                                  ;G:4
                     0000    126 ;-                                                                           ;G:4
                     0000    127                                                                              ;G:4
                     0000    128          .ENABL  LSB                                                         ;G:4
                     0000    129 ACP$ACCESS::                        ;ACCESS AND CREATE ACP FUNCTION PROCESSING ;
                     0000    130 ACP$ACCESSNET::                     ;ACCESS (CONNECT) TO NETWORK FUNCTION PROCES
50   0000'8F   3C   0000    131          MOVZWL  #SS$_FILNOTACC,R0      ; ROUTINE NOT IMPLEMENTED IN S/A     ;G:4
 00000000'EF   16   0005    132          JSB     L^EXE$ABORTIO          ; ABORT THIS QIO                     ;G:4
```

```
                        000B   134          .SBTTL  DEACCESS ACP FUNCTION PROCESSING                                    ;G:4
                        000B   135 ;+                                                                                    ;G:4
                        000B   136 ; ACP$DEACCESS - DEACCESS ACP FUNCTION PROCESSING                                    ;G:4
                        000B   137 ;                                                                                    ;G:4
                        000B   138 ; ***TBS***                                                                          ;G:4
                        000B   139 ;                                                                                    ;G:4
                        000B   140 ; INPUTS:                                                                            ;G:4
                        000B   141 ;                                                                                    ;G:4
                        000B   142 ;       R0 = SCRATCH.                                                                ;G:4
                        000B   143 ;       R1 = SCRATCH.                                                                ;G:4
                        000B   144 ;       R2 = SCRATCH.                                                                ;G:4
                        000B   145 ;       R3 = ADDRESS OF I/O REQUEST PACKET.                                          ;G:4
                        000B   146 ;       R4 = CURRENT PROCESS PCB ADDRESS.                                            ;G:4
                        000B   147 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.                                            ;G:4
                        000B   148 ;       R6 = ADDRESS OF CCB.                                                         ;G:4
                        000B   149 ;       R7 = I/O FUNCTION CODE BIT NUMBER.                                           ;G:4
                        000B   150 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.                               ;G:4
                        000B   151 ;       R9 = SCRATCH.                                                                ;G:4
                        000B   152 ;       R10 = SCRATCH.                                                               ;G:4
                        000B   153 ;       R11 = SCRATCH.                                                               ;G:4
                        000B   154 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.                          ;G:4
                        000B   155 ;                                                                                    ;G:4
                        000B   156 ; OUTPUTS:                                                                           ;G:4
                        000B   157 ;                                                                                    ;G:4
                        000B   158 ; ***TBS***                                                                          ;G:4
                        000B   159 ;-                                                                                   ;G:4
                        000B   160                                                                                      ;G:4
                        000B   161 ACP$DEACCESS::                                ;DEACCESS ACP FUNCTION PROCESSING      ;G:4
     50   0000'8F   3C  000B   162          MOVZWL  #SS$_FILNOTACC,R0          ; FUNCTION NOR SUPPORTED BY SUPERVISOR ;G:4
      00000000'EF   16  0010   163          JSB     L^EXE$ABORTIO              ; ABORT THIS QIO                       ;G:4
```

```
                     0016   165          .SBTTL  DELETE AND MODIFY ACP FUNCTION PROCESSING                    ;G:4
                     0016   166 ;+                                                                           ;G:4
                     0016   167 ; ACP$MODIFY - DELETE AND MODIFY ACP FUNCTION PROCESSING                     ;G:4
                     0016   168 ;                                                                           ;G:4
                     0016   169 ; ***TBS***                                                                 ;G:4
                     0016   170 ;                                                                           ;G:4
                     0016   171 ; INPUTS:                                                                   ;G:4
                     0016   172 ;                                                                           ;G:4
                     0016   173 ;       R0 = SCRATCH.                                                      ;G:4
                     0016   174 ;       R1 = SCRATCH.                                                      ;G:4
                     0016   175 ;       R2 = SCRATCH.                                                      ;G:4
                     0016   176 ;       R3 = ADDRESS OF I/O REQUEST PACKET.                                ;G:4
                     0016   177 ;       R4 = CURRENT PROCESS PCB ADDRESS.                                  ;G:4
                     0016   178 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.                                  ;G:4
                     0016   179 ;       R6 = ADDRESS OF CCB.                                               ;G:4
                     0016   180 ;       R7 = I/O FUNCTION CODE BIT NUMBER.                                 ;G:4
                     0016   181 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.                     ;G:4
                     0016   182 ;       R9 = SCRATCH.                                                      ;G:4
                     0016   183 ;       R10 = SCRATCH.                                                     ;G:4
                     0016   184 ;       R11 = SCRATCH.                                                     ;G:4
                     0016   185 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.                ;G:4
                     0016   186 ;                                                                           ;G:4
                     0016   187 ; OUTPUTS:                                                                  ;G:4
                     0016   188 ;                                                                           ;G:4
                     0016   189 ; ***TBS***                                                                 ;G:4
                     0016   190 ;-                                                                          ;G:4
                     0016   191                                                                            ;G:4
                     0016   192 ACP$MODIFY::                                ;DELETE AND MODIFY ACP FUNCTION PROCESSING ;
 50   0000'8F   3C   0016   193          MOVZWL  #SS$_FILNOTACC,R0          ; ROUTINE NOT SUPPORTED BY SUPERVISOR ;G:4
  00000000'EF   16   001B   194          JSB     L^EXE$ABORTIO              ; ABORT THIS QIO                      ;G:4
```

```
                    0021          .SBTTL  MOUNT ACP FUNCTION PROCESSING                                   ;G:4
                    0021  ;+                                                                              ;G:4
                    0021  ; ACP$MOUNT - MOUNT ACP FUNCTION PROCESSING                                     ;G:4
                    0021  ;                                                                               ;G:4
                    0021  ; ***TBS***                                                                     ;G:4
                    0021  ;                                                                               ;G:4
                    0021  ; INPUTS:                                                                       ;G:4
                    0021  ;                                                                               ;G:4
                    0021  ;       R0 = SCRATCH.                                                           ;G:4
                    0021  ;       R1 = SCRATCH.                                                           ;G:4
                    0021  ;       R2 = SCRATCH.                                                           ;G:4
                    0021  ;       R3 = ADDRESS OF I/O REQUEST PACKET.                                     ;G:4
                    0021  ;       R4 = CURRENT PROCESS PCB ADDRESS.                                       ;G:4
                    0021  ;       R5 = ASSIGNED DEVICE UCB ADDRESS.                                       ;G:4
                    0021  ;       R6 = ADDRESS OF CCB.                                                    ;G:4
                    0021  ;       R7 = I/O FUNCTION CODE BIT NUMBER.                                      ;G:4
                    0021  ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.                          ;G:4
                    0021  ;       R9 = SCRATCH.                                                           ;G:4
                    0021  ;       R10 = SCRATCH.                                                          ;G:4
                    0021  ;       R11 = SCRATCH.                                                          ;G:4
                    0021  ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.                     ;G:4
                    0021  ;                                                                               ;G:4
                    0021  ; OUTPUTS:                                                                      ;G:4
                    0021  ;                                                                               ;G:4
                    0021  ; ***TBS***                                                                     ;G:4
                    0021  ;-                                                                              ;G:4
                    0021                                                                                  ;G:4
                    0021  ACP$MOUNT::                                   ;MOUNT ACP FUNCTION PROCESSING    ;G:4
50    0000'8F   3C  0021          MOVZWL  #SS$_FILNOTACC,R0             ; FUNCTION NOT SUPPORTED BY SUPERVISOR ;G:4
 00000000'EF    16  0026          JSB     L^EXE$ABORTIO                 ; ABORT THIS QIO                  ;G:4
                    002C          .DSABL  LSB                                                             ;G:4
```

```
                     002C    228          .SBTTL  READ AND WRITE BLOCK ACP FUNCTION PROCESSING                    ;G:4
                     002C    229 ;+                                                                              ;G:4
                     002C    230 ; ACP$READBLK - READ BLOCK ACP FUNCTION PROCESSING                              ;G:4
                     002C    231 ; ACP$WRITEBLK - WRITE BLOCK ACP FUNCTION PROCESSING                            ;G:4
                     002C    232 ;                                                                               ;G:4
                     002C    233 ; ***TBS***                                                                     ;G:4
                     002C    234 ;                                                                               ;G:4
                     002C    235 ; INPUTS:                                                                       ;G:4
                     002C    236 ;                                                                               ;G:4
                     002C    237 ;       R0 = SCRATCH.                                                           ;G:4
                     002C    238 ;       R1 = SCRATCH.                                                           ;G:4
                     002C    239 ;       R2 = SCRATCH.                                                           ;G:4
                     002C    240 ;       R3 = ADDRESS OF I/O REQUEST PACKET.                                     ;G:4
                     002C    241 ;       R4 = CURRENT PROCESS PCB ADDRESS.                                       ;G:4
                     002C    242 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.                                       ;G:4
                     002C    243 ;       R6 = ADDRESS OF CCB.                                                    ;G:4
                     002C    244 ;       R7 = I/O FUNCTION CODE BIT NUMBER.                                      ;G:4
                     002C    245 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.                          ;G:4
                     002C    246 ;       R9 = SCRATCH.                                                           ;G:4
                     002C    247 ;       R10 = SCRATCH.                                                          ;G:4
                     002C    248 ;       R11 = SCRATCH.                                                          ;G:4
                     002C    249 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.                     ;G:4
                     002C    250 ;                                                                               ;G:4
                     002C    251 ; OUTPUTS:                                                                      ;G:4
                     002C    252 ;                                                                               ;G:4
                     002C    253 ; ***TBS***                                                                     ;G:4
                     002C    254 ;-                                                                              ;G:4
                     002C    255                                                                                 ;G:4
                     002C    256          .ENABL  LSB                                                            ;G:4
                     002C    257 ACP$READBLK::                                   ;READ BLOCK ACP FUNCTION PROCESSING ;G:4
  5A   0000'CF   9E  002C    258          MOVAB   W^EXE$READLOCK,R10             ;SET ADDRESS OF READ CHECK AND LOCK ROUTINE
22 34 A5   1E   E1  0031    259          BBC     S^#DEV$V_RCK,UCB$L_DEVCHAR(R5),20$ ;IF CLR, NO DEVICE READ CHECK ;G:
20 A3   4000 8F   A8  0036    260 10$:     BISW    #IO$M_DATACHECK,IRP$W_FUNC(R3) ;SET DATA CHECK ENABLE          ;G:4
            1A   11  003C    261          BRB     20$                            ;                               ;G:4
                     003E    262 ;                                                                               ;G:4
                     003E    263 ; ZERO LENGTH TRANSFER                                                          ;G:4
                     003E    264 ;                                                                               ;G:4
                     003E    265                                                                                 ;G:4
  50   0000'8F   3C  003E    266 15$:     MOVZWL  #SS$_NORMAL,R0                 ;SET NORMAL COMPLETION          ;G:4
   00000000'EF   16  0043    267          JSB     L^EXE$FINISHIOC                ;FINISH I/O OPERATION           ;G:4
                     0049    268                                                                                 ;G:4
                     0049    269 ACP$WRITEBLK::                                  ;WRITE BLOCK ACP FUNCTION PROCESSING ;G:4
46 34 A5   19   E0  0049    270          BBS     S^#DEV$V_SWL,UCB$L_DEVCHAR(R5),60$ ;IF SET, SOFTWARE WRITE LOCKED ;G
  5A   0000'CF   9E  004E    271          MOVAB   W^EXE$WRITELOCK,R10            ;SET ADDRESS OF WRITE CHECK AND LOCK ROUTINE
DE 34 A5   1F   E0  0053    272          BBS     S^#DEV$V_WCK,UCB$L_DEVCHAR(R5),10$ ;IF SET, DEVICE LEVEL WRITE CHECK
     50   6C   D0  0058    273 20$:     MOVL    P1(AP),R0                      ;GET STARTING ADDRESS OF TRANSFER ;G:4
   51   04 AC   3C  005B    274          MOVZWL  P2(AP),R1                      ;GET NUMBER OF BYTES TO TRANSFER ;G:4
           DD   13  005F    275          BEQL    15$                            ;IF EQL ZERO LENGTH TRANSFER    ;G:4
05 34 A5   05   E1  0061    276          BBC     S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),30$ ;IF CLR, NOT SEQUENTIAL DEVICE ;G
     51   0E   D1  0066    277          CMPL    #14,R1                         ;RECORD TOO SHORT?              ;G:4
           1B   14  0069    278          BGTR    45$                            ;IF GTR YES                     ;G:4
           6A   16  006B    279 30$:     JSB     (R10)                          ;CHECK BUFFER AND LOCK IN MEMORY ;G:4
   50   08 AC   D0  006D    280          MOVL    P3(AP),R0                      ;GET STARTING MEDIA ADDRESS     ;G:4
   51   32 A3   3C  0071    281          MOVZWL  IRP$W_BCNT(R3),R1              ;RETRIEVE TRANSFER BYTE COUNT   ;G:4
3E A3   51   B0  0075    282          MOVW    R1,IRP$W_OBCNT(R3)             ;SET ORIGINAL BYTE COUNT        ;G:4
     3C A3   B4  0079    283          CLRW    IRP$W_ABCNT(R3)                ;CLEAR ACCUMULATED BYTE COUNT   ;G:4
     06   00   ED  007C    284          CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;LOGICAL OR PHYSICAL I/O FUNCTION? ;G:4
```

```
                2F   20 A3        007F   285                 IRP$W_FUNC(R3),#IO$_LOGICAL ;                                   ;G:4
                       1B   15    0082   286          BLEQ   120$                        ;IF LEQ YES                          ;G:4
                       07   11    0084   287          BRB    50$                         ; Supervisor does not support Virtual ;G:4
                                  0086   288                                                                                    ;G:4
                                  0086   289 ;                                                                                  ;G:4
                                  0086   290 ; RECORD LENGTH TOO SHORT FOR MAGTAPE TRANSFER                                    ;G:4
                                  0086   291 ;                                                                                  ;G:4
                                  0086   292                                                                                    ;G:4
          50    0000'8F    3C     0086   293 45$:    MOVZWL  #SS$_BADPARAM,R0           ;SET BAD PARAMETER STATUS            ;G:4
                       0C   11    008B   294          BRB    70$                         ;                                    ;G:4
                                  008D   295                                                                                    ;G:4
                                  008D   296 ;                                                                                  ;G:4
                                  008D   297 ; FILE NOT ACCESSED ON CHANNEL                                                     ;G:4
                                  008D   298 ;                                                                                  ;G:4
                                  008D   299                                                                                    ;G:4
          50    0000'8F    3C     008D   300 50$:    MOVZWL  #SS$_FILNOTACC,R0          ;SET FILE NOT ACCESSED               ;G:4
                       05   11    0092   301          BRB    70$                         ;                                    ;G:4
                                  0094   302                                                                                    ;G:4
                                  0094   303 ;                                                                                  ;G:4
                                  0094   304 ; PRIVILEGE VIOLATION                                                             ;G:4
                                  0094   305 ;                                                                                  ;G:4
                                  0094   306                                                                                    ;G:4
          50    0000'8F    3C     0094   307 60$:    MOVZWL  #SS$_NOPRIV,R0             ;SET NO PRIVILEGE                    ;G:4
           00000000'EF     16     0099   308 70$:    JSB     L^EXE$ABORTIO              ;                                    ;G:4
                                  009F   309                                                                                    ;G:4
            34 A3    50    D0     009F   310 120$:   MOVL    R0,IRP$L_MEDIA(R3)         ;SAVE MEDIA ADDRESS                  ;G:4
               59    50    7D     00A3   311          MOVQ    R0,R9                      ;SAVE TRANSFER PARAMETERS            ;G:4
          50    0084 C4    D0     00A6   312          MOVL    PCB$L_ARB(R4),R0           ;GET ACCESS RIGHTS BLOCK             ;G:4
            51    1A A5    3C     00AB   313          MOVZWL  UCB$W_VPROT(R5),R1         ;GET VOLUME PROTECTION MASK          ;G:4
            52    1C A5    D0     00AF   314          MOVL    UCB$L_OWNUIC(R5),R2        ;GET VOLUME OWNER UIC                ;G:4
               06   00    ED     00B3   315          CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;PHYSICAL I/O FUNCTION?              ;G:4
            1F   20 A3          00B6   316                  IRP$W_FUNC(R3),#IO$_PHYSICAL ;                                    ;G:4
                       28   15    00B9   317          BLEQ   150$                        ;IF LEQ YES                          ;G:4
                       15   A2    00BB   318          SUBW   #IO$_WRITELBLK-IO$_WRITEPBLK,- ;CONVERT TO PHYSICAL I/O FUNCTION ;G:
                  20 A3          00BD   319                  IRP$W_FUNC(R3)             ;                                    ;G:4
          19 34 A5    05    E0     00BF   320          BBS    S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),140$ ;IF SET, SEQUENTIAL DEVICE ;G:4
                       5A   D7    00C4   321 130$:   DECL    R10                        ;ROUND BYTE COUNT DOWN               ;G:4
       5A   5A    F7   8F    78     00C6   322          ASHL   #-VA$S_BYTE,R10,R10         ;CONVERT TO BLOCK COUNT              ;G:4
               5A   59    C0     00CB   323          ADDL   R9,R10                     ;CALCULATE ENDING BLOCK NUMBER       ;G:4
                       19   1F    00CE   324          BCS    170$                       ;IF CS ILLEGAL BLOCK NUMBER          ;G:4
          0084 C5    5A   D1     00D0   325          CMPL   R10,UCB$L_MAXBLOCK(R5)     ;LEGAL BLOCK NUMBER?                 ;G:4
                       12   1E    00D5   326          BGEQU  170$                       ;IF GEQU NO                          ;G:4
               50   59    D0     00D7   327          MOVL   R9,R0                      ;RETRIEVE STARTING MEDIA ADDRESS     ;G:4
                  FF23'     30     00DA   328          BSBW   IOC$CVTLOGPHY              ;CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS ;
               1800 8F    AA     00DD   329 140$:   BICW   #IO$M_INHERLOG!IO$M_INHSEEK,- ;CLEAR INHIBIT ERROR LOGGING      ;G:4
                 20 A3          00E1   330                  IRP$W_FUNC(R3)             ;AND EXPLICIT SEEK                   ;G:4
           00000000'EF     16     00E3   331 150$:   JSB     L^EXE$QIODRVPKT            ;QUEUE I/O PACKET TO DRIVER          ;G:4
          50    0000'8F    3C     00E9   332 170$:   MOVZWL  #SS$_ILLBLKNUM,R0          ;SET ILLEGAL BLOCK NUMBER STATUS     ;G:4
                       05   11    00EE   333          BRB    190$                                                             ;G:4
          50    0000'8F    3C     00F0   334 180$:   MOVZWL  #SS$_ENDOFFILE,R0          ;SET END OF FILE STATUS              ;G:4
           00000000'EF     16     00F5   335 190$:   JSB     L^EXE$FINISHIOC            ;FINISH I/O OPERATION                ;G:4
                                  00FB   336          .DSABL  LSB                                                              ;G:4
```

```
                        00FB   338          .END                                                    ;G:4
```

ACP$ACCESS		00000000	RG	D	01	IRP$Q_NT_PRVMSK		0000003C	D
ACP$ACCESSNET		00000000	RG	D	01	IRP$S_FCODE	=	00000006	D
ACP$DEACCESS		0000000B	RG	D	01	IRP$V_FCODE	=	00000000	D
ACP$MODIFY		00000016	RG	D	01	IRP$W_ABCNT		0000003C	D
ACP$MOUNT		00000021	RG	D	01	IRP$W_BCNT		00000032	D
ACP$READBLK		0000002C	RG	D	01	IRP$W_BOFF		00000030	D
ACP$WRITEBLK		00000049	RG	D	01	IRP$W_CHAN		00000028	D
CCB$B_AMOD		00000009		D		IRP$W_FUNC		00000020	D
CCB$B_STS		00000008		D		IRP$W_OBCNT		0000003E	D
CCB$C_LENGTH		00000010		D		IRP$W_SIZE		00000008	D
CCB$K_LENGTH		00000010		D		IRP$W_STS		0000002A	D
CCB$L_DIRP		0000000C		D		IRP$W_TT_PRMPT		0000003C	D
CCB$L_UCB		00000000		D		MXDESCR	=	00000016	D
CCB$L_WIND		00000004		D		P1	=	00000000	D
CCB$W_IOC		0000000A		D		P2	=	00000004	D
DEV$V_RCK	=	0000001E		D		P3	=	00000008	D
DEV$V_SQD	=	00000005		D		P4	=	0000000C	D
DEV$V_SWL	=	00000019		D		P5	=	00000010	D
DEV$V_WCK	=	0000001F		D		P6	=	00000014	D
EXE$ABORTIO		********	X		01	PCB$B_ASTACT		0000000C	D
EXE$FINISHIOC		********	X		01	PCB$B_ASTEN		0000000D	D
EXE$QIODRVPKT		********	X		01	PCB$B_PRI		0000000B	D
EXE$READLOCK		********	X		01	PCB$B_PRIB		0000002F	D
EXE$WRITELOCK		********	X		01	PCB$B_TYPE		0000000A	D
IO$M_DATACHECK	=	00004000		D		PCB$B_WEFC		0000002E	D
IO$M_INHERLOG	=	00000800		D		PCB$C_LENGTH		0000008C	D
IO$M_INHSEEK	=	00001000		D		PCB$K_LENGTH		0000008C	D
IO$_LOGICAL	=	0000002F		D		PCB$L_ARB		00000084	D
IO$_PHYSICAL	=	0000001F		D		PCB$L_ASTQBL		00000014	D
IO$_WRITELBLK	=	00000020		D		PCB$L_ASTQFL		00000010	D
IO$_WRITEPBLK	=	0000000B		D		PCB$L_EFC2P		00000058	D
IOC$CVTLOGPHY		********	X		01	PCB$L_EFC3P		0000005C	D
IRP$B_CARCON		00000038		D		PCB$L_EFCS		00000050	D
IRP$B_EFN		00000022		D		PCB$L_EFCU		00000054	D
IRP$B_PRI		00000023		D		PCB$L_EFWM		0000004C	D
IRP$B_RMOD		0000000B		D		PCB$L_JIB		00000078	D
IRP$B_TYPE		0000000A		D		PCB$L_OWNER		0000001C	D
IRP$C_LENGTH		0000005C		D		PCB$L_PHD		00000064	D
IRP$K_LENGTH		0000005C		D		PCB$L_PHYPCB		00000018	D
IRP$L_ARB		00000050		D		PCB$L_PID		00000060	D
IRP$L_AST		00000010		D		PCB$L_PQB		0000004C	D
IRP$L_ASTPRM		00000014		D		PCB$L_SQBL		00000004	D
IRP$L_DIAGBUF		00000044		D		PCB$L_SQFL		00000000	D
IRP$L_EXTEND		0000004C		D		PCB$L_STS		00000024	D
IRP$L_IOQBL		00000004		D		PCB$L_UIC		00000088	D
IRP$L_IOQFL		00000000		D		PCB$L_WSSWP		00000020	D
IRP$L_IOSB		00000024		D		PCB$L_WTIME		00000028	D
IRP$L_IOST1		00000034		D		PCB$Q_PRIV		0000007C	D
IRP$L_IOST2		00000038		D		PCB$T_LNAME		00000068	D
IRP$L_MEDIA		00000034		D		PCB$T_TERMINAL		00000044	D
IRP$L_PID		0000000C		D		PCB$W_APTCNT		00000030	D
IRP$L_SEGVBN		00000040		D		PCB$W_ASTCNT		00000038	D
IRP$L_SEQNUM		00000048		D		PCB$W_BIOCNT		0000003A	D
IRP$L_SVAPTE		0000002C		D		PCB$W_BIOLM		0000003C	D
IRP$L_TT_TERM		00000038		D		PCB$W_DIOCNT		0000003E	D
IRP$L_UCB		0000001C		D		PCB$W_DIOLM		00000040	D
IRP$L_WIND		00000018		D		PCB$W_GPGCNT		00000034	D

PCB$W_GRP	0000008A	D			UCB$L_DEVCHAR	00000034	D
PCB$W_MEM	00000088	D			UCB$L_DEVDEPEND	0000003C	D
PCB$W_MTXCNT	0000000E	D			UCB$L_DPC	00000080	D
PCB$W_PPGCNT	00000036	D			UCB$L_DUETIM	0000005C	D
PCB$W_PRCCNT	00000042	D			UCB$L_DX_BFPNT	0000009C	D
PCB$W_SIZE	00000008	D			UCB$L_DX_BUF	00000098	D
PCB$W_STATE	0000002C	D			UCB$L_DX_RXDB	000000A0	D
PCB$W_TMBU	00000032	D			UCB$L_EMB	00000078	D
SIZ...	= 00000001	D			UCB$L_FIRST	00000014	D
SS$_BADPARAM	********	X	01		UCB$L_FPC	0000000C	D
SS$_ENDOFFILE	********	X	01		UCB$L_FQBL	00000004	D
SS$_FILNOTACC	********	X	01		UCB$L_FQFL	00000000	D
SS$_ILLBLKNUM	********	X	01		UCB$L_FR3	00000010	D
SS$_NOPRIV	********	X	01		UCB$L_FR4	00000014	D
SS$_NORMAL	********	X	01		UCB$L_IOQBL	00000044	D
UCB$B_AMOD	00000053	D			UCB$L_IOQFL	00000040	D
UCB$B_CEX	00000077	D			UCB$L_IRP	0000004C	D
UCB$B_CM1	0000004A	D			UCB$L_LINK	0000002C	D
UCB$B_CM2	0000004B	D			UCB$L_LOGADR	00000064	D
UCB$B_DEVCLASS	00000038	D			UCB$L_MAXBLOCK	00000084	D
UCB$B_DEVTYPE	00000039	D			UCB$L_MB_MBX	0000007C	D
UCB$B_DIPL	00000052	D			UCB$L_MB_PORT	0000008C	D
UCB$B_DX_SCTCNT	000000A6	D			UCB$L_MB_RAST	00000078	D
UCB$B_ERTCNT	00000070	D			UCB$L_MB_SHB	00000080	D
UCB$B_ERTMAX	00000071	D			UCB$L_MB_WAST	00000074	D
UCB$B_FEX	00000076	D			UCB$L_MB_WIOQBL	00000088	D
UCB$B_FIPL	0000000B	D			UCB$L_MB_WIOQFL	00000084	D
UCB$B_LOCSRV	0000003C	D			UCB$L_MEDIA	0000008C	D
UCB$B_OFFNDX	00000094	D			UCB$L_NT_DATSSB	00000074	D
UCB$B_OFFRTC	00000095	D			UCB$L_NT_INTSSB	00000078	D
UCB$B_REMSRV	0000003D	D			UCB$L_OPCNT	00000060	D
UCB$B_SECTORS	0000003C	D			UCB$L_OWNUIC	0000001C	D
UCB$B_SLAVE	00000074	D			UCB$L_PID	00000028	D
UCB$B_SPR	00000075	D			UCB$L_RQBL	00000004	D
UCB$B_STATE	00000052	D			UCB$L_RQFL	00000000	D
UCB$B_TRACKS	0000003D	D			UCB$L_SVAPTE	00000068	D
UCB$B_TT_CRFILL	0000009D	D			UCB$L_SVPN	00000064	D
UCB$B_TT_DECRF	000000A1	D			UCB$L_TT_DECHAR	000000A8	D
UCB$B_TT_DELFF	000000A2	D			UCB$L_TT_RDUE	0000008C	D
UCB$B_TT_DESPEE	000000A0	D			UCB$L_TT_RTIMOU	000000B8	D
UCB$B_TT_DETYPE	000000A4	D			UCB$L_VCB	00000030	D
UCB$B_TT_LFFILL	0000009E	D			UCB$T_PARTNER	0000000C	D
UCB$B_TT_SPEED	0000009C	D			UCB$W_BCNT	0000006E	D
UCB$B_TYPE	0000000A	D			UCB$W_BCR	00000096	D
UCB$B_VERTSZ	0000003F	D			UCB$W_BOFF	0000006C	D
UCB$C_LENGTH	00000074	D			UCB$W_BUFQUO	00000018	D
UCB$C_MB_LENGTH	00000090	D			UCB$W_BYTESTOGO	0000003E	D
UCB$C_TT_LENGTH	000000BC	D			UCB$W_CHARGE	0000004A	D
UCB$K_LENGTH	00000074	D			UCB$W_CYLINDERS	0000003E	D
UCB$K_MB_LENGTH	00000090	D			UCB$W_DA	0000008C	D
UCB$K_TT_LENGTH	000000BC	D			UCB$W_DC	0000008E	D
UCB$L_AMB	00000054	D			UCB$W_DEVBUFSIZ	0000003A	D
UCB$L_ASTQBL	00000010	D			UCB$W_DEVSTS	0000005A	D
UCB$L_ASTQFL	0000000C	D			UCB$W_DIRSEQ	00000088	D
UCB$L_CPID	0000005C	D			UCB$W_DSTADDR	00000018	D
UCB$L_CRB	00000020	D			UCB$W_DX_BCR	000000A4	D
UCB$L_DDB	00000024	D			UCB$W_ECI	00000090	D

UCB$W_EC2	00000092	D	VCB$L_USRLBLAST	00000044	D		
UCB$W_ERRCNT	00000072	D	VCB$L_VPBL	00000040	D		
UCB$W_FUNC	0000007E	D	VCB$L_VPFL	0000003C	D		
UCB$W_MB_SEED	00000000	D	VCB$L_WCB	00000038	D		
UCB$W_MSGCNT	00000016	D	VCB$T_QNAME	0000000C	D		
UCB$W_MSGMAX	00000014	D	VCB$T_VOLNAME	00000014	D		
UCB$W_NT_CHAN	0000007C	D	VCB$W_CLUSTER	0000003C	D		
UCB$W_OFFSET	0000008A	D	VCB$W_CUR_NUM	00000024	D		
UCB$W_REFC	00000050	D	VCB$W_CUR_SEQ	00000026	D		
UCB$W_SIZE	00000008	D	VCB$W_EXTEND	0000003E	D		
UCB$W_SRCADDR	0000001A	D	VCB$W_FILEPROT	0000004A	D		
UCB$W_STS	00000058	D	VCB$W_MCOUNT	0000004C	D		
UCB$W_TT_DESIZE	000000A5	D	VCB$W_MODE	0000002C	D		
UCB$W_UNIT	00000048	D	VCB$W_PENDERR	00000062	D		
UCB$W_VPROT	0000001A	D	VCB$W_QUOSIZE	00000060	D		
VA$S_BYTE	= 00000009	D	VCB$W_RECORDSZ	00000050	D		
VCB$B_BLOCKFACT	00000052	D	VCB$W_RVN	0000000E	D		
VCB$B_CUR_RVN	0000002F	D	VCB$W_SIZE	00000008	D		
VCB$B_EOFDELTA	0000004E	D	VCB$W_START_NUM	00000028	D		
VCB$B_IBMAPSIZE	00000038	D	VCB$W_START_SEQ	0000002A	D		
VCB$B_IBMAPVBN	0000003A	D	VCB$W_TRANS	0000000C	D		
VCB$B_LRU_LIM	00000049	D	WCB$B_ACCESS	0000000B	D		
VCB$B_QNAMECNT	0000000B	D	WCB$B_TYPE	0000000A	D		
VCB$B_RESFILES	0000004F	D	WCB$C_LENGTH	00000024	D		
VCB$B_SBMAPSIZE	00000039	D	WCB$C_MAP	00000024	D		
VCB$B_SBMAPVBN	0000003B	D	WCB$K_LENGTH	00000024	D		
VCB$B_STATUS	0000000B	D	WCB$K_MAP	00000024	D		
VCB$B_STATUS2	00000053	D	WCB$L_FCB	00000018	D		
VCB$B_TM	0000002E	D	WCB$L_LBN	00000002	D		
VCB$B_TYPE	0000000A	D	WCB$L_ORGUCB	00000010	D		
VCB$B_WINDOW	00000048	D	WCB$L_P1_LBN	00000026	D		
VCB$C_COMLEN	00000024	D	WCB$L_P2_LBN	0000002C	D		
VCB$C_LENGTH	00000064	D	WCB$L_PID	0000000C	D		
VCB$C_MRKLEN	0000000B	D	WCB$L_PREVLBN	FFFFFFFC	D		
VCB$K_COMLEN	00000024	D	WCB$L_RVT	0000001C	D		
VCB$K_LENGTH	00000064	D	WCB$L_STVBN	00000020	D		
VCB$K_MRKLEN	0000000B	D	WCB$L_WLBL	00000004	D		
VCB$L_AQB	00000010	D	WCB$L_WLFL	00000000	D		
VCB$L_BLOCKBL	00000004	D	WCB$W_ACON	00000014	D		
VCB$L_BLOCKFL	00000000	D	WCB$W_COUNT	00000000	D		
VCB$L_CACHE	00000058	D	WCB$W_NMAP	00000016	D		
VCB$L_CUR_FID	00000024	D	WCB$W_P1_COUNT	00000024	D		
VCB$L_FCBBL	00000004	D	WCB$W_P2_COUNT	0000002A	D		
VCB$L_FCBFL	00000000	D	WCB$W_PREVCOUNT	FFFFFFFA	D		
VCB$L_FREE	00000040	D	WCB$W_REFCNT	0000000E	D		
VCB$L_HOME2LBN	00000028	D	WCB$W_SIZE	00000008	D		
VCB$L_HOMELBN	00000024	D					
VCB$L_IBMAPLBN	00000030	D					
VCB$L_IXHDR2LBN	0000002C	D					
VCB$L_MAXFILES	00000044	D					
VCB$L_MVL	00000034	D					
VCB$L_QUOCACHE	0000005C	D					
VCB$L_QUOTAFCB	00000054	D					
VCB$L_RVT	00000020	D					
VCB$L_SBMAPLBN	00000034	D					
VCB$L_START_FID	00000028	D					
VCB$L_ST_RECORD	00000030	D					

```
                    +-----------------+
                    ! Psect synopsis !
                    +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
. ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
SEP                             000000FB  (  251.)  01 (  1.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD    WRT NOVEC LONG
$ABS$                           FFFFFFFC  (    0.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACP$ACCESS	00000000-R	129 (1)	
ACP$ACCESSNET	00000000-R	130 (1)	
ACP$DEACCESS	0000000B-R	161 (1)	
ACP$MODIFY	00000016-R	192 (1)	
ACP$MOUNT	00000021-R	223 (1)	
ACP$READBLK	0000002C-R	257 (1)	
ACP$WRITEBLK	00000049-R	269 (1)	
DEV$V_RCK	=0000001E		#-259 (1)
DEV$V_SQD	=00000005		#-276 (1) #-320 (1)
DEV$V_SWL	=00000019		#-270 (1)
DEV$V_WCK	=0000001F		#-272 (1)
EXE$ABORTIO	00000000-XR		132 (1) 163 (1) 194 (1) 225 (1) 308 (1)
EXE$FINISHIOC	00000000-XR		267 (1) 335 (1)
EXE$QIODRVPKT	00000000-XR		331 (1)
EXE$READLOCK	00000000-XR		258 (1)
EXE$WRITELOCK	00000000-XR		271 (1)
IO$M_DATACHECK	=00004000		#-260 (1)
IO$M_INHERLOG	=00000800		#-329 (1)
IO$M_INHSEEK	=00001000		#-329 (1)
IO$_LOGICAL	=0000002F		#-285 (1)
IO$_PHYSICAL	=0000001F		#-316 (1)
IO$_WRITELBLK	=00000020		#-318 (1)
IO$_WRITEPBLK	=0000000B		#-318 (1)
IOC$CVTLOGPHY	00000000-XR		#-328 (1)
IRP$L_MEDIA	00000034		#-310 (1)
IRP$S_FCODE	=00000006		#-284 (1) #-315 (1)
IRP$V_FCODE	=00000000		#-284 (1) #-315 (1)
IRP$W_ABCNT	0000003C		#-283 (1)
IRP$W_BCNT	00000032		#-281 (1)
IRP$W_FUNC	00000020		#-260 (1) 285 (1) 316 (1) #-319 (1) #-330 (1)
IRP$W_OBCNT	0000003E		#-282 (1)
MXDESCR	=00000016	98 (1)	
P1	=00000000	87 (1)	#-273 (1)
P2	=00000004	88 (1)	#-274 (1)
P3	=00000008	89 (1)	#-280 (1)
P4	=0000000C	90 (1)	
P5	=00000010	91 (1)	
P6	=00000014	92 (1)	
PCB$L_ARB	00000084		#-312 (1)
SS$_BADPARAM	00000000-XR		#-293 (1)
SS$_ENDOFFILE	00000000-XR		#-334 (1)
SS$_FILNOTACC	00000000-XR		#-131 (1) #-162 (1) #-193 (1) #-224 (1) #-300 (1)
SS$_ILLBLKNUM	00000000-XR		#-332 (1)
SS$_NOPRIV	00000000-XR		#-307 (1)
SS$_NORMAL	00000000-XR		#-266 (1)
UCB$L_DEVCHAR	00000034		259 (1) 270 (1) 272 (1) 276 (1) 320 (1)

```
UCB$L_MAXBLOCK                    00000084                  #-325    (1)
UCB$L_OWNUIC                      0000001C                  #-314    (1)
UCB$W_VPROT                       0000001A                  #-313    (1)
VA$S_BYTE                        =00000009                  #-322    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CCBDEF	1	66 (1)	66 (1)
$DEFINI	1	66 (1)	66 (1) 67 (1) 68 (1) 69 (1) 70 (1) 71 (1) 72 (1) 73 (1) 74 (1) 75 (1) 76 (1) 77 (1) 78 (1) 79 (1)
$DEVDEF	1	67 (1)	67 (1)
$DYNDEF	2	68 (1)	68 (1)
$IODEF	15	70 (1)	70 (1)
$IPLDEF	1	69 (1)	69 (1)
$IRPDEF	4	71 (1)	71 (1)
$JIBDEF	3	72 (1)	72 (1)
$PCBDEF	4	73 (1)	73 (1)
$PRDEF	5	74 (1)	74 (1)
$PRVDEF	4	75 (1)	75 (1)
$UCBDEF	10	76 (1)	76 (1)
$VADEF	1	77 (1)	77 (1)
$VCBDEF	6	78 (1)	78 (1)
$WCBDEF	3	79 (1)	79 (1)

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDL            00C0      323    (1)
ASHL            0078      322    (1)
BBC             00E1      259    (1)    276    (1)
BBS             00E0      270    (1)    272    (1)    320    (1)
BCS             001F      324    (1)
BEQL            0013      275    (1)
BGEQU           001E      326    (1)
BGTR            0014      278    (1)
BICW            00AA      329    (1)
BISW            00A8      260    (1)
BLEQ            0015      286    (1)    317    (1)
BRB             0011      261    (1)    287    (1)    294    (1)    301    (1)    333    (1)
BSBW            0030      328    (1)
CLRW            00B4      283    (1)
CMPL            00D1      277    (1)    325    (1)
CMPZV           00ED      284    (1)    315    (1)
DECL            00D7      321    (1)
JSB             0016      132    (1)    163    (1)    194    (1)    225    (1)    267    (1)    279    (1)    308    (1)
                          331    (1)    335    (1)
MOVAB           009E      258    (1)    271    (1)
MOVL            00D0      273    (1)    280    (1)    310    (1)    312    (1)    314    (1)    327    (1)
MOVQ            007D      311    (1)
MOVW            00B0      282    (1)
MOVZWL          003C      131    (1)    162    (1)    193    (1)    224    (1)    266    (1)    274    (1)    281    (1)
                          293    (1)    300    (1)    307    (1)    313    (1)    332    (1)    334    (1)
SUBW            00A2      318    (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.DSABL	226	(1)	336	(1)												
.ENABL	128	(1)	256	(1)												
.END	338	(1)														
.ENDM	66	(1)	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)	72	(1)	73	(1)
	74	(1)	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)				
.IDENT	8	(1)														
.NOCROSS	66	(1)	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)	72	(1)	73	(1)
	74	(1)	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)				
.PAGE	133	(1)	164	(1)	195	(1)	227	(1)	337	(1)	99	(1)				
.PSECT	9	(1)														
.RESTORE	66	(1)	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)	72	(1)	73	(1)
	74	(1)	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)				
.SAVE	66	(1)	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)	72	(1)	73	(1)
	74	(1)	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)				
.SBTTL	100	(1)	134	(1)	165	(1)	196	(1)	228	(1)						
.TITLE	7	(1)														

```
                                        +---------------------------+
                                        ! Register Cross Reference !
                                        +---------------------------+

REGISTER  NO. REFERENCES     REFERENCES...
--------  --------------     -------------
AP                 3         #-273    (1)    #-274    (1)    #-280    (1)
R0                16         #-131    (1)    #-162    (1)    #-193    (1)    #-224    (1)    #-266    (1)    #-273    (1)
                              #-280    (1)    #-293    (1)    #-300    (1)    #-307    (1)    #-310    (1)    #-311    (1)
                              #-312    (1)    #-327    (1)    #-332    (1)    #-334    (1)
R1                 5         #-274    (1)    #-277    (1)    #-281    (1)    #-282    (1)    #-313    (1)
R10                8         #-258    (1)    #-271    (1)      279    (1)    #-321    (1)    #-322    (1)    #-323    (1)
                              #-325    (1)
R2                 1         #-314    (1)
R3                 9         #-260    (1)    #-281    (1)    #-282    (1)    #-283    (1)      285    (1)    #-310    (1)
                                316    (1)    #-319    (1)    #-330    (1)
R4                 1         #-312    (1)
R5                 8           259    (1)      270    (1)      272    (1)      276    (1)    #-313    (1)    #-314    (1)
                                320    (1)    #-325    (1)
R9                 3         #-311    (1)    #-323    (1)    #-327    (1)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     25    00:00:00.01    00:00:00.19
Command processing                884    00:00:00.25    00:00:00.83
Pass 1                            728    00:00:03.32    00:00:04.50
Symbol table sort                   0    00:00:00.29    00:00:00.30
Pass 2                              6    00:00:00.82    00:00:01.40
Symbol table output                 0    00:00:00.07    00:00:00.21
Psect synopsis output               0    00:00:00.01    00:00:00.01
Cross-reference output              1    00:00:00.07    00:00:00.11
Assembler run totals             1644    00:00:04.84    00:00:07.55

The working set limit was 3512 pages.
198028 bytes (387 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1161 non-local and 15 local symbols.
338 source lines were read in Pass 1, producing 73 object records in Pass 2.
104 pages of virtual memory were used to define 23 macros.

                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                            Macros defined
------------------                            --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8                  9
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                    0
SYS$COMMON:[SYSLIB]LIB.MLB;2                               2
SYS$COMMON:[SYSLIB]STARLET.MLB;2                           8
TOTALS (all libraries)                                    19

1544 GETS were required to define 19 macros.
```

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 2 Frame E3          Sequence 236
ACPFDT                      *** ACPFDT ACP function dispatch     9-DEC-1987 11:48:53  VAX/VMS Macro V04-00        Page  31
VAX-11 Macro Run Statistics                                     12-MAY-1986 09:41:34  ACPFDT.MAR;1                      (1)

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ACPFDT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element ANSI.B32
;    0002  0      ! *3    12-MAY-1986 09:52:33 BARANSKI "Cleaning up Object Libraries."
;    0003  0      ! *2     7-MAY-1986 12:59:31 BARANSKI "Documentation Check."
;    0004  0      ! *1     6-FEB-1986 15:13:24 DS ""
;    0005  0      ! DEC/CMS REPLACEMENT HISTORY, Element ANSI.B32
;    0006  0      %title '*** ANSI magtape RMS support'
;    0007  0      module ansi (                                   ! Routines to handle Magtape
;    0008  0                   ident = '7.0-7'                                                  !G:3
;    0009  0                   ) =
;    0010  0      !
;    0011  0      ! COPYRIGHT (c) 1980, 1981, 1984
;    0012  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0013  0      !
;    0014  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0015  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0016  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0017  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0018  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0019  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0020  0      ! REMAIN IN DEC.
;    0021  0      !
;    0022  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0023  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0024  0      ! CORPORATION.
;    0025  0      !
;    0026  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0027  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0028  0
;    0029  0      !++
;    0030  0      ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0031  0      !
;    0032  0      ! ABSTRACT:
;    0033  0      !
;    0034  0      !       These modules simulate the function of the respective
;    0035  0      !       RMS functions.
;    0036  0      !
;    0037  0      ! AUTHOR:       Roger Riggs, CREATION DATE: 10-November-1979
;    0038  0      !
;    0039  0      ! MODIFIED BY:
;    0040  0      !
;    0041  0      !               Dave Butenhof, 07-oct-1980, version 6.1
;    0042  0      !       01      To allow support of magtape directory command which will
;    0043  0      !               function in standalone or user mode, support minimal level
;    0044  0      !               of wildcarding.  "*" means open next file on tape, return
;    0045  0      !               full file name in NAM block if any.  There is no wildcard
;    0046  0      !               context of any sort; the caller must remember the first
;    0047  0      !               file it found, and exit the wildcard loop when that name
;    0048  0      !               is repeated.  This is for Supervisor use only and will not
;    0049  0      !               be documented outside the program listings of ANSI and
;    0050  0      !               DIRECTORY.
;    0051  0      !                                                                        !G:2
;    0052  0      !       02      Support version numbers
;    0053  0      !               Roger Riggs, 22-oct-1980
;    0054  0      !                                                                        !G:2
;    0055  0      !       03      Added code to implement FAB$V_RWO rewind on open.
;    0056  0      !               The magtape is rewound before the search for the file begins.
;    0057  0      !               also added to code to issue IO$_drvclr before first operation.
```

```
!                                                                                    !G:2
!                - Dave Butenhof, 02-Jun-1981, version 6.4                          !G:2
!       04       Fix library specifications, use $DS, $DIAG logical names for
!                flexibility.
!
!       05       - Jack Stansbury, 28-Oct-1981, Version 6.-
!                Changed the PSECT declarations to take out the SEP PSECT.
!
!       06       Bob Bergazzi    May 14, 1984    Version 7.0
!                Fixed the compiler warning from BLISS released with VMS 4.0.
!                This occurred at the redeclaration of BOO$QIO with the CALL_BOO
!                linkage in routine ANSI$OPEN.
!
!       07       Bob Bergazzi    May 25, 1984    Version 7.0
!                Added support for VMS V4 long filenames using HDR4 labels.
!-

begin
!
! include files:
!

library '$diag';                !

library '$ds';                  !

library 'sys$library:lib';

linkage
    jsb = jsb,
    jsb_scan = jsb (register = 3, register = 4, register = 5) : nopreserve (2),
    get_block = call ( ;register = 2) : global (cblock = 9, fab = 11);  !

!
! table of contents:
!

forward routine
    ansi$open : call_rms addressing_mode (long_relative),       !Open file on magtape
    ansi$close : call_rms addressing_mode (long_relative),      !Close file on magtape
    ansi$read : call_rms addressing_mode (long_relative),       !Read block(s) from magtape
    ansi$get : call_rms addressing_mode (long_relative),        !Get record from magtape
    get_block_length : get_block addressing_mode (long_relative);!Do a BOO$QIO, returning block length

!
! macros
!


!
! external references:
!

external routine
    scan$numeric : jsb_scan addressing_mode (long_relative),
    kb_check : jsb addressing_mode (long_relative),
    rms$alloc_cache : jsb_cblock addressing_mode (long_relative);
```

```
;    0115  1
;    0116  1     !
;    0117  1     ! psect definitions:
;    0118  1     !
;    0119  1
;    0120  1     PSECT                                                   !                       [05]
;    0121  1         Plit    = Data (NoExecute, Share, NoWrite,          !                       [05]
;    0122  1                         Addressing_Mode (Long_Relative));   !                       [05]
;    0123  1
;    0124  1     PSECT                                                   !                       [05]
;    0125  1         Code    = Code (Execute, Share, NoWrite,            !                       [05]
;    0126  1                         Addressing_Mode (Long_Relative));   !                       [05]
;    0127  1
;    0128  1     PSECT                                                   !                       [05]
;    0129  1         Own     = Work (NoExecute, NoShare, Write,          !                       [05]
;    0130  1                         Addressing_Mode (Long_Relative));   !                       [05]
;    0131  1
;    0132  1     PSECT                                                   !                       [05]
;    0133  1         Global  = Work (NoExecute, NoShare, Write,          !                       [05]
;    0134  1                         Addressing_Mode (Long_Relative));   !                       [05]
```

```
;    0136  1      %sbttl 'ansi$open'
;    0137  1
;    0138  1      global routine ansi$open : call_rms =
;    0139  1
;    0140  1      !++
;    0141  1      ! Functional description:
;    0142  1      !
;    0143  1      !       This routine is responsible for locating the file specified
;    0144  1      !       by the FAB on the magtape specified.  It will search forward
;    0145  1      !       from the current tape position to locate the file.
;    0146  1      !       When a HDR1 record is found the file name is checked,
;    0147  1      !       if the filename matches the rest of the header are skiped.
;    0148  1      !       When a EOF1 record is found the file name is checked,
;    0149  1      !       if the filename matches the tape is backspaced to the start of the data
;    0150  1      !       If at the beginning of the wrong file the entire file is skiped.
;    0151  1      !       If the incorrect file is the same as the first file found when the
;    0152  1      !       search began RMS$_FNF is returned.
;    0153  1      !
;    0154  1      !       If double tape marks are found instead of a file header the
;    0155  1      !       tape is rewound and the search continued until end of tape is
;    0156  1      !       encountered again, when file not found is returned.
;    0157  1      !
;    0158  1      !       For the correct file the entire file is read to determine it length.
;    0159  1      !       The first two blocks read from the tape are saved in the cache
;    0160  1      !       to reduce the amount of positioning for small files.
;    0161  1      !       The current tape position within a file is maintained as TAPEPOS.
;    0162  1      !
;    0163  1      !       If the FAB$V_RWO bit is set in the FAB the tape is rewound before
;    0164  1      !       the search begins
;    0165  1      !
;    0166  1      ! Formal parameters:    none
;    0167  1      !
;    0168  1      ! Implicit inputs:
;    0169  1      !
;    0170  1      !       FAB     Address of related FAB
;    0171  1      !       cblock  Address of per-file context area
;    0172  1      !
;    0173  1      ! Implicit outputs:
;    0174  1      !
;    0175  1      !       FAB     Various fields updated to reflect the file found
;    0176  1      !       cblock  Various fileds updated to reflect the file found
;    0177  1      !
;    0178  1      ! Routine value:
;    0179  1      !
;    0180  1      !       Standard RMS completion codes
;    0181  1      !
;    0182  1      ! Side effects:
;    0183  1      !
;    0184  1      !       The device and associated channel are accessed
;    0185  1      !
;    0186  1      !--
;    0187  1
;    0188  2          BEGIN
;    0189  2
;    0190  2          literal
;    0191  2              true = 1,                                   !Truth
;    0192  2              false = 0,                                  !Falsity
```

```
;    0193  2            end_of_pass1 = 1,                        !Set if end of tape encountered once
;    0194  2            end_of_file = 2;                         !Set if end of file sensed on previous read
;    0195  2
;    0196  2        local
;    0197  2            version,                                 ! Version number in file name
;    0198  2            tapeversion,                             ! Version number in hdr1 block
;    0199  2            flag : bitvector [32],                   !Contains flag bits
;    0200  2            hdr1 : block [80, byte],                 !Contains HDR1 record
;    0201  2            hdr2 : block [80, byte],                 !Contains HDR2 record
;    0202  2            hdr4 : block [80, byte],                 !Contains HDR3 or HDR4 record                          [07]
;    0203  2            first1 : vector [80, byte],              !Contains first HDR1 record found
;    0204  2            filename : vector [132, byte];           !expected filename in HDR1 label
;    0205  2
;    0206  2        external register
;    0207  2            fab = 11 : ref $fab_decl,                !Base of FAB
;    0208  2            cblock = 9 : ref block [, byte];         !Base of common area
;    0209  2
;    0210  2        external routine                             !                                      [06]
;    0211  2            get_block_length : get_block addressing_mode (long_relative),   !BOO$QIO that returns block_length      [06]
;    0212  2            boo$qio : addressing_mode (long_relative);     !                               [06]
;    0213  2
;    0214  2        bind
;    0215  2            status = fab [fab$l_stv];                !Return value from boo$qio
;    0216  2
;    0217  2    !
;    0218  2    ! Initialize dispatch addresses
;    0219  2    !
;    0220  2        cblock [ctl$l_get] = ansi$get;
;    0221  2        cblock [ctl$l_read] = ansi$read;
;    0222  2        cblock [ctl$l_close] = ansi$close;
;    0223  2    !+
;    0224  2    ! convert the file name in the control region into the format
;    0225  2    ! expected for comparison with HDR1 labels.
;    0226  2    ! note that the version number is ignored.
;    0227  2    !-
;    0228  3        begin
;    0229  3
;    0230  3        local
;    0231  3            len,
;    0232  3            verlen,
;    0233  3            veradr,
;    0234  3            adr;                                     !Length and address of current segment
;    0235  3
;    0236  3        adr = ch$find_ch (.cblock [ctl$w_file_len], .cblock [ctl$l_file_ptr], %C']') + 1;
;    0237  3        veradr = ch$find_ch (.cblock [ctl$w_file_len], .cblock [ctl$l_file_ptr], %C';');
;    0238  3        len = .veradr - .adr;                        ! Calculate length of <name>.<type>
;    0239  3        veradr = .veradr + 1;                        ! Address of version (after ";")
;    0240  3        verlen = .cblock [ctl$w_file_len] - (.veradr - .cblock [ctl$l_file_ptr]);   ! Length of rest of name
;    0241  3        version = scan$numeric (10, .verlen, .veradr);      ! Convert ascii to binary
;    0242  3        ch$fill (' ', 132, filename);
;    0243  3        ch$move (.len, .adr, filename);
;    0244  2        end;
;    0245  2        flag = false;
;    0246  2        first1 [0] = 0;                              !Clear first character
;    0247  2
;    0248  3        if not (status = boo$qio (0, 0, 0, io$_drvclr, 0, .cblock [ctl$l_rpb]))
;    0249  2        then return .status;
```

```
;    0250  2
;    0251  2          if .fab [fab$v_rwo]
;    0252  2          then
;    0253  3              begin
;    0254  4              if not (status = boo$qio (0, 0, 0, io$_rewind, 0, .cblock [ctl$l_rpb]))
;    0255  3              then return .status;
;    0256  2              end;
;    0257  2
;    0258  2          until 0 do
;    0259  3              begin
;    0260  3              kb_check ();                                    !Check for ^C
;    0261  3              status = boo$qio (hdr1, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);
;    0262  3
;    0263  3              selectone .status of
;    0264  3                  set
;    0265  3
;    0266  3                  [ss$_normal] :
;    0267  4                      begin
;    0268  4                      flag [end_of_file] = false;
;    0269  4
;    0270  4                      if ((.hdr1 eql %ascii'EOF1') or (.hdr1 eql %ascii'HDR1')) and
;    0271  5                          begin
;    0272  5                          tapeversion = (scan$numeric (10, 4, hdr1 [hd1$t_genno]) - 1)*100 + scan$numeric (10, 2,
;    0273  5                              hdr1 [hd1$t_genver]) + 1;       ! Calculate version number
;    0274  7                          if ((ch$compare (17, filename, 17, hdr1 [hd1$t_fileid], 0) eql 0) or (.filename [0] eql %c
;    0275  6                              '*' and .hdr1 eql %ascii'HDR1')) and (.tapeversion eql .version or .version eql 0)
;    0276  6                              then begin
;    0277  6                                   status = boo$qio (hdr2, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);
;    0278  6                                   if not .status then return rms$_rer;
;    0279  6
;    0280  6                              status = boo$qio (hdr4, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);      ! Really
;    0281  6                              if not .status then return rms$_rer;                                       ! reads HDR3
;    0282  6
;    0283  6                              status = boo$qio (hdr4, 80, 0, io$_readlblk, 0, .cblock [ctl$l_rpb]);      ! Reads HDR4
;    0284  6
;    0285  7                              if ((.filename [0] eql %c'*') and (.version eql 0))                        ! If wildcar
;    0286  6                                 then true                                                               !
;    0287  6                                 else if .status eql ss$_endoffile                                       ! EOF means
;    0288  6                                         then true                                                       ! no HDR4
;    0289  6                                         else if not .status                                             ! Error whic
;    0290  6                                                 then return rms$_rer                                    ! is not EOF
;    0291  6                                                 else if .hdr4 [hd4$b_fileid_ext_size] eql 0     ! Filename fits in H
;    0292  6                                                         then true                                       !
;    0293  6                                                         else          ! Compare the rest of the filename in HDR4
;    0294  6                                                              if ch$compare (.hdr4 [hd4$b_fileid_ext_size],      !
;    0295  6                                                                             filename [17],                      !
;    0296  6                                                                             .hdr4 [hd4$b_fileid_ext_size],  !
;    0297  6                                                                             hdr4 [hd4$t_fileid_ext], 0) eql 0 !
;    0298  6                                                                 then true                                      !
;    0299  6                                                                 else false                                     !
;    0300  6                                                 end
;    0301  5                                     else false
;    0302  5                          end
;    0303  4                      then
;    0304  5                          begin
;    0305  5
;    0306  5                          local
```

```
;    0307  5                          block_length;
;    0308  5
;    0309  5      !
;    0310  5      ! KLUGE:
;    0311  5      !       This module does NOT actually support $NAM block.  However, if one
;    0312  5      !       is present, it will copy the "resultant name string" into the RSA
;    0313  5      !       field.  This supports the DS "Directory" command and has no other
;    0314  5      !       purpose.
;    0315  5      !
;    0316  5
;    0317  5                          if .fab [fab$l_nam] neq 0
;    0318  5                          then
;    0319  6                              begin
;    0320  6
;    0321  6                              bind
;    0322  6                                  phynam = .cblock [ctl$l_ptable] : bblock,
;    0323  6                                  nam = .fab [fab$l_nam] : bblock;
;    0324  6
;    0325  6                              local
;    0326  6                                  blank,
;    0327  6                                  dev : bblock [dsc$c_s_bln],
;    0328  6                                  file : bblock [dsc$c_s_bln],
;    0329  6                                  longfile : bblock [dsc$c_s_bln],                                    !       [07]
;    0330  6                                  a : bblock [dsc$c_s_bln];
;    0331  6
;    0332  6                              a [dsc$w_length] = .nam [nam$b_rss];
;    0333  6                              a [dsc$a_pointer] = .nam [nam$l_rsa];
;    0334  6                              dev [dsc$a_pointer] = .phynam [dsc$a_pointer] + 1;      ! Trim off "_"
;    0335  6                              dev [dsc$w_length] = .phynam [dsc$w_length] - 1;
;    0336  6                              file [dsc$a_pointer] = hdr1 [hd1$t_fileid];
;    0337  6                              blank = ch$find_ch (17, hdr1 [hd1$t_fileid], %c' ');
;    0338  6                              file [dsc$w_length] = (if .blank eql 0 then 17 else .blank - hdr1 [hd1$t_fileid]);
;    0339  6                              longfile [dsc$a_pointer] = hdr4 [hd4$t_fileid_ext];                     !       [07]
;    0340  7                              if ((.hdr4 eql %ascii'HDR4') or (.hdr4 eql %ascii'EOF4'))               !       [07]
;    0341  6                                 then longfile [dsc$w_length] = .hdr4 [hd4$b_fileid_ext_size]         !       [07]
;    0342  6                                 else longfile [dsc$w_length] = 0;                                    !       [07]
;    0343  6                              $fao ($ascid ('!AS:!AS!AS;!ZL'), a, a, dev, file, longfile, .tapeversion);      !       [07]
;    0344  6                                                  ! Convert version to ascii
;    0345  6                              nam [nam$b_rsl] = .a [dsc$w_length]
;    0346  5                              end;
;    0347  5
;    0348  5                          if .hdr1 eql %ascii'HDR1'
;    0349  6                          then begin                                                                  !       [07]
;    0350  6                               if .hdr4 eql %ascii'HDR4'      ! At start of file, skip leading EOF        [07]
;    0351  7                                  then begin                  ! Already there if pre-VMS V4 tape          [07]
;    0352  7                                       status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0353  7                                       if not .status then return rms$_rer
;    0354  7                                       end
;    0355  6                               end
;    0356  6                          else begin                          ! at end of file, backover trailers and data
;    0357  6                               if .hdr4 neq %ascii'EOF4'      ! Backup an extra tape mark                 [07]
;    0358  7                                  then begin                                                       !       [07]
;    0359  7                                       status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);     !       [07]
;    0360  7                                       if not .status then return rms$_rer                        !       [07]
;    0361  6                                       end;                                                       !       [07]
;    0362  6
;    0363  6                              status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
```

```
;    0364  6
;    0365  6                        if not .status then return rms$_rer;
;    0366  6
;    0367  6                        status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0368  6
;    0369  6                        if not .status then return rms$_rer;
;    0370  6
;    0371  6                        status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0372  6
;    0373  6                        if not .status then return rms$_rer;
;    0374  6
;    0375  5                        end;
;    0376  5
;    0377  5                    if .(hdr2 [hd2$t_recatr1]) neq %ascii'    '
;    0378  5                    then
;    0379  6                        begin
;    0380  6
;    0381  6                        bind
;    0382  6                            fat = hdr2 [hd2$t_recatr1] : block [fat$c_length, byte];
;    0383  6
;    0384  6                        fab [fab$b_rat] = .fat [fat$b_rattrib];
;    0385  6                        fab [fab$b_rfm] = .fat [fat$v_rtype];
;    0386  6                        fab [fab$b_fsz] = .fat [fat$b_vfcsize];
;    0387  6                        fab [fab$w_mrs] = .fat [fat$w_maxrec];
;    0388  6                        end
;    0389  5                    else
;    0390  6                        begin
;    0391  6
;    0392  6                        selectone .hdr2 [hd2$b_recformat] of
;    0393  6                            set
;    0394  6
;    0395  6                            [%c'F'] :
;    0396  7                                begin
;    0397  7                                fab [fab$b_rfm] = fab$c_fix;
;    0398  7                                fab [fab$w_mrs] = scan$numeric (10, 5, hdr2 [hd2$t_reclen]);
;    0399  6                                end;
;    0400  6
;    0401  6                            [%c'D'] :
;    0402  7                                begin
;    0403  7                                fab [fab$b_rfm] = fab$c_var;
;    0404  7                                fab [fab$w_mrs] = scan$numeric (10, 5, hdr2 [hd2$t_reclen]) - 4;
;    0405  6                                end;
;    0406  6
;    0407  6                            [otherwise] :
;    0408  6                                return rms$_org;
;    0409  6                            tes;
;    0410  6
;    0411  6                        selectone .hdr2 [hd2$b_formcntrl] of
;    0412  6                            set
;    0413  6
;    0414  6                            [%c'A'] :
;    0415  6                                fab [fab$b_rat] = fab$m_ftn;
;    0416  6
;    0417  6                            [%c' '] :
;    0418  6                                fab [fab$b_rat] = fab$m_cr;
;    0419  6
;    0420  6                            [otherwise] :
```

```
;    0421  6                                    fab [fab$b_rat] = 0;
;    0422  6                                tes;
;    0423  6
;    0424  5                                end;
;    0425  5
;    0426  6                            if not (status = rms$alloc_cache ((scan$numeric (10, 5, hdr2 [hd2$t_blocklen]) + 511)/512
;    0427  6                                ))
;    0428  5                            then
;    0429  5                                return rms$_dme;
;    0430  5
;    0431  5                            cblock [ctl$l_eofvbn] = 0;
;    0432  5                            cblock [ctl$w_offbyte] = 0;
;    0433  5                            cblock [ctl$l_lvbn] = cblock [ctl$l_hvbn] = 1;
;    0434  5                            cblock [ctl$l_tapepos] = 1;
;    0435  5
;    0436  5                            while get_block_length( ;block_length) do           !                                [06]
;    0437  6                                begin
;    0438  6
;    0439  6                                bind
;    0440  6                                    size = cblock [ctl$w_cache_size] : word;
;    0441  6
;    0442  6                                if .cblock [ctl$l_tapepos] eql 1
;    0443  6                                then
;    0444  7                                    begin
;    0445  7
;    0446  7                                    local
;    0447  7                                        tmp;
;    0448  7
;    0449  7                                    tmp = .cblock [ctl$l_cache3];
;    0450  7                                    cblock [ctl$l_cache3] = .cblock [ctl$l_cache1];
;    0451  7                                    cblock [ctl$l_cache1] = .tmp;
;    0452  7                                    cblock [ctl$l_hvbn] = .size + 1;
;    0453  6                                    end;
;    0454  6
;    0455  6                                if .cblock [ctl$l_tapepos] eql .size + 1
;    0456  6                                then
;    0457  7                                    begin
;    0458  7
;    0459  7                                    local
;    0460  7                                        tmp;
;    0461  7
;    0462  7                                    tmp = .cblock [ctl$l_cache3];
;    0463  7                                    cblock [ctl$l_cache3] = .cblock [ctl$l_cache2];
;    0464  7                                    cblock [ctl$l_cache2] = .tmp;
;    0465  7                                    cblock [ctl$l_hvbn] = .size^1 + 1;
;    0466  6                                    end;
;    0467  6
;    0468  6                                cblock [ctl$l_tapepos] = .cblock [ctl$l_tapepos] + .size;
;    0469  6                                cblock [ctl$l_eofvbn] = ((.cblock [ctl$l_eofvbn] + .size - 1) and not (.size - 1)) +
;    0470  6                                ((.block_length + 511)^-9);
;    0471  6                                cblock [ctl$w_offbyte] = .block_length and 511;
;    0472  5                                end;
;    0473  5
;    0474  5                            cblock [ctl$l_eofvbn] = .cblock [ctl$l_eofvbn] + 1;
;    0475  5                            status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0476  5
;    0477  5                            if not .status then return rms$_rer;
```

```
;    0478  5
;    0479  5                        Return RMS$_normal
;    0480  5                        end
;    0481  4                    else
;    0482  5                        begin                        ! Not correct file
;    0483  5
;    0484  5                        if .hdr1 eql 'HDR1'          ! If at start of file skip whole file
;    0485  5                        then
;    0486  6                            begin
;    0487  6
;    0488  6                            if ch$compare (80, hdr1, 80, first1, 0) eql 0
;    0489  6                            then
;    0490  7                                begin
;    0491  7                                if .hdr4 eql %ascii'HDR4'                                    !       [07]
;    0492  8                                    then begin                                               !       [07]
;    0493  8                                         status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]); !       [07]
;    0494  8                                         if not .status then return rms$_rer;                !       [07]
;    0495  8                                         end                                                 !       [07]
;    0496  8                                    else begin                          ! Must be pre-VMS V4 tape (no HDR4)      [07]
;    0497  8                                         status = boo$qio (-1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]); !       [07]
;    0498  8                                         if not .status then return rms$_rer;                !       [07]
;    0499  7                                         end;                                                !       [07]
;    0500  7                                status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);         !       [07]
;    0501  7                                if not .status then return rms$_rer;                         !       [07]
;    0502  7                                status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);         !       [07]
;    0503  7                                if not .status then return rms$_rer;                         !       [07]
;    0504  7                                status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);
;    0505  7                                if not .status then return rms$_rer;
;    0506  7                                return rms$_fnf;
;    0507  7                                end
;    0508  6                            else
;    0509  6
;    0510  6                                if .first1 [0] eql 0 then ch$move (80, hdr1, first1);
;    0511  6
;    0512  6                                if .hdr4 eql %ascii'HDR4'                                    !       [07]
;    0513  7                                   then begin                                                !       [07]
;    0514  7                                        status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0515  7                                        if not .status then return rms$_rer;
;    0516  6                                        end;
;    0517  6
;    0518  6                                status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0519  6
;    0520  6                                if not .status then return rms$_rer;
;    0521  6
;    0522  6                                status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0523  6
;    0524  6                                if not .status then return rms$_rer;
;    0525  6
;    0526  6                                flag [end_of_file] = true;
;    0527  5                                end;
;    0528  5
;    0529  4                            end;
;    0530  4
;    0531  3                        end;
;    0532  3
;    0533  3                    [ss$_dataoverun] :
;    0534  4                        begin
```

```
;   0535  4                 status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;   0536  4
;   0537  4                 if not .status then return rms$_rer;
;   0538  4
;   0539  4                 flag [end_of_file] = false;
;   0540  3                 end;
;   0541  3
;   0542  3             [ss$_endoffile] :
;   0543  4                 begin
;   0544  4
;   0545  4                 if .flag [end_of_file]
;   0546  4                 then
;   0547  5                     begin
;   0548  5                     status = boo$qio (0, 0, 0, io$_rewind, 0, .cblock [ctl$l_rpb]);
;   0549  5                     flag [end_of_file] = false;
;   0550  5
;   0551  5                     if .flag [end_of_pass1] then return rms$_fnf else flag [end_of_pass1] = true;
;   0552  5
;   0553  5                     end
;   0554  4                 else
;   0555  4                     flag [end_of_file] = true;
;   0556  4
;   0557  3                 end;
;   0558  3
;   0559  3             [otherwise] :
;   0560  3                 return rms$_rer;
;   0561  3             tes;
;   0562  3
;   0563  2         end;
;   0564  2
;   0565  2     ss$_normal
;   0566  1     END;


                                                                .TITLE  ANSI *** ANSI magtape RMS support
                                                                .IDENT  \7.0-7\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

4C  5A  21  3B  53  41  21  53  41  21  3A  53  41  21  00000 P.AAB:  .ASCII  \!AS:!AS!AS;!ZL\                          ;
                                                        0000E       .BLKB   2
                                                0000000E  00010 P.AAA:  .LONG   14                                        ;
                                                00000000' 00014       .ADDRESS P.AAB                                    ;

                                                                .EXTRN  SCAN$NUMERIC, KB_CHECK
                                                                .EXTRN  RMS$ALLOC_CACHE
                                                                .EXTRN  GET_BLOCK_LENGTH
                                                                .EXTRN  BOO$QIO, SYS$FAO

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                  05FC 00000    .ENTRY  ANSI$OPEN, Save R2,R3,R4,R5,R6,R7,R8,R10    ; 0138
                                5E      FE14   CE  9E 00002    MOVAB   -492(SP), SP                                ;
                                56        0C   AB  9E 00007    MOVAB   12(FAB), R6                                 ; 0215
                            0C  A9 00000000V   EF  9E 0000B    MOVAB   ANSI$GET, 12(CBLOCK)                        ; 0220
                            10  A9 00000000V   EF  9E 00013    MOVAB   ANSI$READ, 16(CBLOCK)                       ; 0221
                            14  A9 00000000V   EF  9E 0001B    MOVAB   ANSI$CLOSE, 20(CBLOCK)                      ; 0222
```

```
                4C   B9       48   A9       5D   8F  3A 00023          LOCC    #93, 72(CBLOCK), a76(CBLOCK)                ; 0236
                                                 02  12 0002A          BNEQ    1$                                          ;
                                                 51  D4 0002C          CLRL    R1                                          ;
                                   57       01   A1  9E 0002E 1$:      MOVAB   1(R1), ADR                                  ;
                4C   B9       48   A9            3B  3A 00032          LOCC    #59, 72(CBLOCK), a76(CBLOCK)                ; 0237
                                                 02  12 00038          BNEQ    2$                                          ;
                                                 51  D4 0003A          CLRL    R1                                          ;
                     58            51            57  C3 0003C 2$:      SUBL3   ADR, VERADR, LEN                            ; 0238
                                                 51  D6 00040          INCL    VERADR                                      ; 0239
                     50       4C   A9            51  C3 00042          SUBL3   VERADR, 76(CBLOCK), R0                      ; 0240
                                   54       48   A9  3C 00047          MOVZWL  72(CBLOCK), VERLEN                          ;
                                   54            50  C0 0004B          ADDL2   R0, VERLEN                                  ;
                                   55            51  D0 0004E          MOVL    VERADR, R5                                  ; 0241
                                   53            0A  D0 00051          MOVL    #10, R3                                     ;
                                      00000000G  EF  16 00054          JSB     SCAN$NUMERIC                                ;
                                   6E            50  D0 0005A          MOVL    R0, VERSION                                 ;
   0084   8F         20            6E            00  2C 0005D          MOVC5   #0, (SP), #32, #132, FILENAME               ; 0242
                                           28   AE     00064                                                              ;
                28   AE            67            58  28 00066          MOVC3   LEN, (ADR), FILENAME                        ; 0243
                                                 5A  D4 0006B          CLRL    FLAG                                        ; 0245
                                         00AC    CE  94 0006D          CLRB    FIRST1                                      ; 0246
                                           38   A9  DD 00071          PUSHL   56(CBLOCK)                                  ; 0248
                                   7E            04  7D 00074          MOVQ    #4, -(SP)                                   ;
                                                 7E  7C 00077          CLRQ    -(SP)                                       ;
                                                 7E  D4 00079          CLRL    -(SP)                                       ;
                      00000000G    EF            06  FB 0007B          CALLS   #6, BOO$QIO                                 ;
                                   66            50  D0 00082          MOVL    R0, (R6)                                    ;
                                   1C            50  E9 00085          BLBC    R0, 3$                                      ;
                                           04   AB  95 00088          TSTB    4(FAB)                                      ; 0251
                                                 1B  18 0008B          BGEQ    4$                                          ;
                                           38   A9  DD 0008D          PUSHL   56(CBLOCK)                                  ; 0254
                                   7E            24  7D 00090          MOVQ    #36, -(SP)                                  ;
                                                 7E  7C 00093          CLRQ    -(SP)                                       ;
                                                 7E  D4 00095          CLRL    -(SP)                                       ;
                      00000000G    EF            06  FB 00097          CALLS   #6, BOO$QIO                                 ;
                                   66            50  D0 0009E          MOVL    R0, (R6)                                    ;
                                   04            50  E8 000A1          BLBS    R0, 4$                                      ;
                                   50            66  D0 000A4 3$:      MOVL    (R6), R0                                    ; 0255
                                                     04 000A7          RET                                                 ;
                                      00000000G  EF  16 000A8 4$:      JSB     KB_CHECK                                    ; 0260
                                   58       38   A9  D0 000AE          MOVL    56(CBLOCK), R8                              ; 0261
                                                 58  DD 000B2          PUSHL   R8                                          ;
                                   7E            21  7D 000B4          MOVQ    #33, -(SP)                                  ;
                                                 7E  D4 000B7          CLRL    -(SP)                                       ;
                                   7E       50   8F  9A 000B9          MOVZBL  #80, -(SP)                                  ;
                                           B0   AD  9F 000BD          PUSHAB  HDR1                                        ;
                      00000000G    EF            06  FB 000C0          CALLS   #6, BOO$QIO                                 ;
                                   66            50  D0 000C7          MOVL    R0, (R6)                                    ;
                                   01            66  D1 000CA          CMPL    (R6), #1                                    ; 0266
                                                 03  13 000CD          BEQL    5$                                          ;
                                              0470  31 000CF          BRW     54$                                         ;
                                   5A            04  8A 000D2 5$:      BICB2   #4, FLAG                                    ; 0268
                      31464F45     8F       B0   AD  D1 000D5          CMPL    HDR1, #826691397                            ; 0270
                                                 0A  13 000DD          BEQL    6$                                          ;
                      31524448     8F       B0   AD  D1 000DF          CMPL    HDR1, #827475016                            ;
                                                 5A  12 000E7          BNEQ    9$                                          ;
                                   55       D3   AD  9E 000E9 6$:      MOVAB   HDR1+35, R5                                 ; 0272
```

```
                        54          04  D0 000ED          MOVL    #4, R4                                  ;
                        53          0A  D0 000F0          MOVL    #10, R3                                 ;
                          00000000G EF  16 000F3          JSB     SCAN$NUMERIC                            ;
                        57          50  D0 000F9          MOVL    R0, R7                                  ;
                        57 00000064 8F  C4 000FC          MULL2   #100, R7                                ;
                        55      D7  AD  9E 00103          MOVAB   HDR1+39, R5                             ; 0273
                        54          02  D0 00107          MOVL    #2, R4                                  ;
                        53          0A  D0 0010A          MOVL    #10, R3                                 ;
                          00000000G EF  16 0010D          JSB     SCAN$NUMERIC                            ;
                    04  AE      9D A047 9E 00113          MOVAB   -99(R0)[R7], TAPEVERSION                ;
                        54          01  D0 00119          MOVL    #1, R4                                  ; 0274
          B4   AD   28  AE          11  29 0011C          CMPC3   #17, FILENAME, HDR1+4                   ;
                                    03  1A 00122          BGTRU   7$                                      ;
                        54          01  D9 00124          SBWC    #1, R4                                  ;
                                    54  D5 00127 7$:      TSTL    R4                                      ;
                                    10  13 00129          BEQL    8$                                      ;
                        2A      28  AE  91 0012B          CMPB    FILENAME, #42                           ;
                                    12  12 0012F          BNEQ    9$                                      ;
                31524448 8F     B0  AD  D1 00131          CMPL    HDR1, #827475016                        ; 0275
                                    08  12 00139          BNEQ    9$                                      ;
                        6E      04  AE  D1 0013B 8$:      CMPL    TAPEVERSION, VERSION                    ;
                                    07  13 0013F          BEQL    10$                                     ;
                                    6E  D5 00141          TSTL    VERSION                                 ;
                                    03  13 00143 9$:      BEQL    10$                                     ;
                                  0084  31 00145          BRW     15$                                     ;
                                    58  DD 00148 10$:     PUSHL   R8                                      ; 0277
                        7E          21  7D 0014A          MOVQ    #33, -(SP)                              ;
                                    7E  D4 0014D          CLRL    -(SP)                                   ;
                        7E      50  8F  9A 0014F          MOVZBL  #80, -(SP)                              ;
                              FF60  CD  9F 00153          PUSHAB  HDR2                                    ;
                00000000G EF        06  FB 00157          CALLS   #6, BOO$QIO                             ;
                        66          50  D0 0015E          MOVL    R0, (R6)                                ;
                        48          66  E9 00161          BLBC    (R6), 12$                               ; 0278
                                    58  DD 00164          PUSHL   R8                                      ; 0280
                        7E          21  7D 00166          MOVQ    #33, -(SP)                              ;
                                    7E  D4 00169          CLRL    -(SP)                                   ;
                        7E      50  8F  9A 0016B          MOVZBL  #80, -(SP)                              ;
                              FF10  CD  9F 0016F          PUSHAB  HDR4                                    ;
                00000000G EF        06  FB 00173          CALLS   #6, BOO$QIO                             ;
                        66          50  D0 0017A          MOVL    R0, (R6)                                ;
                        2C          66  E9 0017D          BLBC    (R6), 12$                               ; 0281
                                    58  DD 00180          PUSHL   R8                                      ; 0283
                        7E          21  7D 00182          MOVQ    #33, -(SP)                              ;
                                    7E  D4 00185          CLRL    -(SP)                                   ;
                        7E      50  8F  9A 00187          MOVZBL  #80, -(SP)                              ;
                              FF10  CD  9F 0018B          PUSHAB  HDR4                                    ;
                00000000G EF        06  FB 0018F          CALLS   #6, BOO$QIO                             ;
                        66          50  D0 00196          MOVL    R0, (R6)                                ;
                        2A      28  AE  91 00199          CMPB    FILENAME, #42                           ; 0285
                                    04  12 0019D          BNEQ    11$                                     ;
                                    6E  D5 0019F          TSTL    VERSION                                 ;
                                    2C  13 001A1          BEQL    16$                                     ;
                00000870 8F         66  D1 001A3 11$:     CMPL    (R6), #2160                             ; 0287
                                    23  13 001AA          BEQL    16$                                     ;
                        03          66  E8 001AC 12$:     BLBS    (R6), 13$                               ; 0289
                                  03EE  31 001AF          BRW     61$                                     ;
                        50    FF14  CD  9A 001B2 13$:     MOVZBL  HDR4+4, R0                              ; 0291
```

```
                                    16  13 001B7          BEQL    16$                                  ;
                        54          01  D0 001B9          MOVL    #1, R4                               ; 0297
   FF15   CD       39   AE          50  29 001BC          CMPC3   R0, FILENAME+17, HDR4+5              ;
                                    03  1A 001C3          BGTRU   14$                                  ;
                        54          01  D9 001C5          SBWC    #1, R4                               ;
                                    54  D5 001C8 14$:     TSTL    R4                                   ;
                                    03  13 001CA          BEQL    16$                                  ;
                                  0282  31 001CC 15$:     BRW     42$                                  ;
                        52      28  AB  D0 001CF 16$:     MOVL    40(FAB), R2                          ; 0317
                                    03  12 001D3          BNEQ    17$                                  ;
                                  0086  31 001D5          BRW     24$                                  ;
                        50      3C  A9  D0 001D8 17$:     MOVL    60(CBLOCK), R0                       ; 0322
                    08   AE      02  A2  9B 001DC          MOVZBW  2(R2), A                             ; 0332
                    0C   AE      04  A2  D0 001E1          MOVL    4(R2), A+4                           ; 0333
     24   AE       04   A0          01  C1 001E6          ADDL3   #1, 4(R0), DEV+4                     ; 0334
     20   AE            60          01  A3 001EC          SUBW3   #1, (R0), DEV                        ; 0335
                    1C   AE      B4  AD  9E 001F1          MOVAB   HDR1+4, FILE+4                       ; 0336
     B4   AD            11          20  3A 001F6          LOCC    #32, #17, HDR1+4                     ; 0337
                                    02  12 001FB          BNEQ    18$                                  ;
                                    51  D4 001FD          CLRL    R1                                   ;
                                    51  D5 001FF 18$:     TSTL    BLANK                                ; 0338
                                    05  12 00201          BNEQ    19$                                  ;
                        51          11  D0 00203          MOVL    #17, R1                              ;
                                    07  11 00206          BRB     20$                                  ;
                        50      B4  AD  9E 00208 19$:     MOVAB   HDR1+4, R0                           ;
                        51          50  C2 0020C          SUBL2   R0, R1                               ;
                    18   AE          51  B0 0020F 20$:     MOVW    R1, FILE                             ;
                    14   AE    FF15  CD  9E 00213          MOVAB   HDR4+5, LONGFILE+4                   ; 0339
              34524448   8F    FF10  CD  D1 00219          CMPL    HDR4, #877806664                     ; 0340
                                    0B  13 00222          BEQL    21$                                  ;
              34464F45   8F    FF10  CD  D1 00224          CMPL    HDR4, #877023045                     ;
                                    08  12 0022D          BNEQ    22$                                  ;
                    10   AE    FF14  CD  9B 0022F 21$:     MOVZBW  HDR4+4, LONGFILE                     ; 0341
                                    03  11 00235          BRB     23$                                  ;
                            10      AE  B4 00237 22$:     CLRW    LONGFILE                             ; 0342
                            04      AE  DD 0023A 23$:     PUSHL   TAPEVERSION                          ; 0343
                            14      AE  9F 0023D          PUSHAB  LONGFILE                             ;
                            20      AE  9F 00240          PUSHAB  FILE                                 ;
                            2C      AE  9F 00243          PUSHAB  DEV                                  ;
                            18      AE  9F 00246          PUSHAB  A                                    ;
                            1C      AE  9F 00249          PUSHAB  A                                    ;
                      00000000'     EF  9F 0024C          PUSHAB  P.AAA                                ;
              00000000G   00          07  FB 00252          CALLS   #7, SYS$FAO                          ;
                    03   A2      08  AE  90 00259          MOVB    A, 3(R2)                             ; 0345
              31524448   8F      B0  AD  D1 0025E 24$:     CMPL    HDR1, #827475016                     ; 0348
                                    24  12 00266          BNEQ    27$                                  ;
              34524448   8F    FF10  CD  D1 00268          CMPL    HDR4, #877806664                     ; 0350
                                    6B  12 00271          BNEQ    29$                                  ;
                                    58  DD 00273 25$:     PUSHL   R8                                   ; 0352
                        7E          25  7D 00275          MOVQ    #37, -(SP)                           ;
                                    7E  7C 00278          CLRQ    -(SP)                                ;
                                    01  DD 0027A          PUSHL   #1                                   ;
              00000000G   EF          06  FB 0027C          CALLS   #6, BOO$QIO                          ;
                        66          50  D0 00283          MOVL    R0, (R6)                             ;
                        55          66  E8 00286          BLBS    (R6), 29$                            ; 0353
                                  0314  31 00289 26$:     BRW     61$                                  ;
              34464F45   8F    FF10  CD  D1 0028C 27$:     CMPL    HDR4, #877023045                     ; 0357
```

```
                                                17  13 00295          BEQL    28$                                  ;
                                                58  DD 00297          PUSHL   R8                                   ; 0359
                                 7E             25  7D 00299          MOVQ    #37, -(SP)                           ;
                                                7E  7C 0029C          CLRQ    -(SP)                                ;
                                 7E             01  CE 0029E          MNEGL   #1, -(SP)                            ;
                      00000000G  EF             06  FB 002A1          CALLS   #6, BOO$QIO                          ;
                                 66             50  D0 002A8          MOVL    R0, (R6)                             ;
                                 DB             66  E9 002AB          BLBC    (R6), 26$                            ; 0360
                                                58  DD 002AE 28$:     PUSHL   R8                                   ; 0363
                                 7E             25  7D 002B0          MOVQ    #37, -(SP)                           ;
                                                7E  7C 002B3          CLRQ    -(SP)                                ;
                                 7E             01  CE 002B5          MNEGL   #1, -(SP)                            ;
                      00000000G  EF             06  FB 002B8          CALLS   #6, BOO$QIO                          ;
                                 66             50  D0 002BF          MOVL    R0, (R6)                             ;
                                 C4             66  E9 002C2          BLBC    (R6), 26$                            ; 0365
                                                58  DD 002C5          PUSHL   R8                                   ; 0367
                                 7E             25  7D 002C7          MOVQ    #37, -(SP)                           ;
                                                7E  7C 002CA          CLRQ    -(SP)                                ;
                                 7E             01  CE 002CC          MNEGL   #1, -(SP)                            ;
                      00000000G  EF             06  FB 002CF          CALLS   #6, BOO$QIO                          ;
                                 66             50  D0 002D6          MOVL    R0, (R6)                             ;
                                 AD             66  E9 002D9          BLBC    (R6), 26$                            ; 0369
                                                95  11 002DC          BRB     25$                                  ; 0371
                      20202020   8F     FF6F    CD  D1 002DE 29$:     CMPL    HDR2+15, #538976288                  ; 0377
                                                1F  13 002E7          BEQL    30$                                  ;
                            1E   AB     FF70    CD  90 002E9          MOVB    FAT+1, 30(FAB)                       ; 0384
   50     FF6F   CD              04             00  EF 002EF          EXTZV   #0, #4, FAT, R0                      ; 0385
                            1F   AB             50  90 002F6          MOVB    R0, 31(FAB)                          ;
                            3F   AB     FF7E    CD  90 002FA          MOVB    FAT+15, 63(FAB)                      ; 0386
                            36   AB     FF7F    CD  B0 00300          MOVW    FAT+16, 54(FAB)                      ; 0387
                                                6E  11 00306          BRB     36$                                  ;
                                 50     FF64    CD  9A 00308 30$:     MOVZBL  HDR2+4, R0                           ; 0392
                            46   8F             50  91 0030D          CMPB    R0, #70                              ; 0395
                                                1B  12 00311          BNEQ    31$                                  ;
                            1F   AB             01  90 00313          MOVB    #1, 31(FAB)                          ; 0397
                                 55     FF6A    CD  9E 00317          MOVAB   HDR2+10, R5                          ; 0398
                                 54             05  D0 0031C          MOVL    #5, R4                               ;
                                 53             0A  D0 0031F          MOVL    #10, R3                              ;
                                    00000000G   EF  16 00322          JSB     SCAN$NUMERIC                         ;
                            36   AB             50  B0 00328          MOVW    R0, 54(FAB)                          ;
                                                2A  11 0032C          BRB     33$                                  ;
                            44   8F             50  91 0032E 31$:     CMPB    R0, #68                              ; 0401
                                                1C  12 00332          BNEQ    32$                                  ;
                            1F   AB             02  90 00334          MOVB    #2, 31(FAB)                          ; 0403
                                 55     FF6A    CD  9E 00338          MOVAB   HDR2+10, R5                          ; 0404
                                 54             05  D0 0033D          MOVL    #5, R4                               ;
                                 53             0A  D0 00340          MOVL    #10, R3                              ;
                                    00000000G   EF  16 00343          JSB     SCAN$NUMERIC                         ;
          36   AB                50             04  A3 00349          SUBW3   #4, R0, 54(FAB)                      ;
                                                08  11 0034E          BRB     33$                                  ;
                                 50 0001860C    8F  D0 00350 32$:     MOVL    #99852, R0                           ; 0408
                                                    04 00357          RET                                          ;
                                 50        84    AD  9A 00358 33$:     MOVZBL  HDR2+36, R0                          ; 0411
                            41   8F             50  91 0035C          CMPB    R0, #65                              ; 0414
                                                06  12 00360          BNEQ    34$                                  ;
                            1E   AB             01  90 00362          MOVB    #1, 30(FAB)                          ; 0415
                                                0E  11 00366          BRB     36$                                  ;
```

```
                        20          50  91 00368 34$:   CMPB    R0, #32                          ; 0417
                                    06  12 0036B        BNEQ    35$                              ;
                  1E    AB          02  90 0036D        MOVB    #2, 30(FAB)                      ; 0418
                                    03  11 00371        BRB     36$                              ;
                              1E    AB  94 00373 35$:   CLRB    30(FAB)                          ; 0421
                        55    FF65  CD  9E 00376 36$:   MOVAB   HDR2+5, R5                       ; 0426
                        54          05  D0 0037B        MOVL    #5, R4                           ;
                        53          0A  D0 0037E        MOVL    #10, R3                          ;
                        00000000G   EF  16 00381        JSB     SCAN$NUMERIC                     ;
                        50    01FF  C0  9E 00387        MOVAB   511(R0), R0                      ;
                        50 00000200 8F  C6 0038C        DIVL2   #512, R0                         ;
                        00000000G   EF  16 00393        JSB     RMS$ALLOC_CACHE                  ;
                        66          50  D0 00399        MOVL    R0, (R6)                         ;
                        08          50  E8 0039C        BLBS    R0, 37$                          ;
                        50 000184D4 8F  D0 0039F        MOVL    #99540, R0                       ; 0429
                                    04 003A6            RET                                      ;
                              44    A9  D4 003A7 37$:   CLRL    68(CBLOCK)                       ; 0431
                              42    A9  B4 003AA        CLRW    66(CBLOCK)                       ; 0432
                  24    A9          01  D0 003AD        MOVL    #1, 36(CBLOCK)                   ; 0433
                  28    A9          01  D0 003B1        MOVL    #1, 40(CBLOCK)                   ;
                  58    A9          01  D0 003B5        MOVL    #1, 88(CBLOCK)                   ; 0434
            00000000G   EF          00  FB 003B9 38$:   CALLS   #0, GET_BLOCK_LENGTH             ; 0436
                        53          52  D0 003C0        MOVL    R2, R3                           ;
                        68          50  E9 003C3        BLBC    R0, 41$                          ;
                        01    58    A9  D1 003C6        CMPL    88(CBLOCK), #1                   ; 0442
                                    15  12 003CA        BNEQ    39$                              ;
                        50    34    A9  D0 003CC        MOVL    52(CBLOCK), TMP                  ; 0449
                  34    A9    2C    A9  D0 003D0        MOVL    44(CBLOCK), 52(CBLOCK)           ; 0450
                  2C    A9          50  D0 003D5        MOVL    TMP, 44(CBLOCK)                  ; 0451
                  24    A9    0A    A9  3C 003D9        MOVZWL  10(CBLOCK), 36(CBLOCK)           ; 0452
                              24    A9  D6 003DE        INCL    36(CBLOCK)                       ;
                        50    0A    A9  3C 003E1 39$:   MOVZWL  10(CBLOCK), R0                   ; 0455
                        51    01    A0  9E 003E5        MOVAB   1(R0), R1                        ;
                        51    58    A9  D1 003E9        CMPL    88(CBLOCK), R1                   ;
                                    15  12 003ED        BNEQ    40$                              ;
                        51    34    A9  D0 003EF        MOVL    52(CBLOCK), TMP                  ; 0462
                  34    A9    30    A9  D0 003F3        MOVL    48(CBLOCK), 52(CBLOCK)           ; 0463
                  30    A9          51  D0 003F8        MOVL    TMP, 48(CBLOCK)                  ; 0464
        24    A9        50          01  78 003FC        ASHL    #1, R0, 36(CBLOCK)               ; 0465
                              24    A9  D6 00401        INCL    36(CBLOCK)                       ;
                  58    A9          50  C0 00404 40$:   ADDL2   R0, 88(CBLOCK)                   ; 0468
              51        50    44    A9  C1 00408        ADDL3   68(CBLOCK), R0, R1               ; 0469
                                    51  D7 0040D        DECL    R1                               ;
                                    50  D7 0040F        DECL    R0                               ;
                        51          50  CA 00411        BICL2   R0, R1                           ;
                        52    01FF  C3  9E 00414        MOVAB   511(R3), R2                      ; 0470
              52        52    F7    8F  78 00419        ASHL    #-9, R2, R2                      ;
        44    A9        51          52  C1 0041E        ADDL3   R2, R1, 68(CBLOCK)               ;
  50          53        09          00  EF 00423        EXTZV   #0, #9, BLOCK_LENGTH, R0         ; 0471
                  42    A9          50  B0 00428        MOVW    R0, 66(CBLOCK)                   ;
                                    8B  11 0042C        BRB     38$                              ;
                              44    A9  D6 0042E 41$:   INCL    68(CBLOCK)                       ; 0474
                              38    A9  DD 00431        PUSHL   56(CBLOCK)                       ; 0475
                        7E          25  7D 00434        MOVQ    #37, -(SP)                       ;
                                    7E  7C 00437        CLRQ    -(SP)                            ;
                        7E          01  CE 00439        MNEGL   #1, -(SP)                        ;
            00000000G   EF          06  FB 0043C        CALLS   #6, BOO$QIO                      ;
```

```
                                66          50  D0 00443          MOVL    R0, (R6)                                  ;
                                7E          66  E9 00446          BLBC    (R6), 47$                                 ; 0477
                                50 00010001 8F  D0 00449          MOVL    #65537, R0                                ; 0479
                                            04  00450             RET                                               ;
                    31524448    8F      B0  AD  D1 00451 42$:     CMPL    HDR1, #827475016                          ; 0484
                                            03  13 00459          BEQL    43$                                       ;
                                          FC4A  31 0045B          BRW     4$                                        ;
                                54          01  D0 0045E 43$:     MOVL    #1, R4                                    ; 0488
          00AC   CE       B0    AD    0050  8F  29 00461          CMPC3   #80, HDR1, FIRST1                         ;
                                            03  1A 0046A          BGTRU   44$                                       ;
                                54          01  D9 0046C          SBWC    #1, R4                                    ;
                                            54  D5 0046F 44$:     TSTL    R4                                        ;
                                            71  12 00471          BNEQ    48$                                       ;
                    34524448    8F    FF10  CD  D1 00473          CMPL    HDR4, #877806664                          ; 0491
                                            07  12 0047C          BNEQ    45$                                       ;
                                            58  DD 0047E          PUSHL   R8                                        ; 0493
                                7E          26  7D 00480          MOVQ    #38, -(SP)                                ;
                                            05  11 00483          BRB     46$                                       ;
                                            58  DD 00485 45$:     PUSHL   R8                                        ; 0497
                                7E          25  7D 00487          MOVQ    #37, -(SP)                                ;
                                            7E  7C 0048A 46$:     CLRQ    -(SP)                                     ;
                                7E          01  CE 0048C          MNEGL   #1, -(SP)                                 ;
                    00000000G   EF          06  FB 0048F          CALLS   #6, BOO$QIO                               ;
                                66          50  D0 00496          MOVL    R0, (R6)                                  ;
                                75          66  E9 00499          BLBC    (R6), 50$                                 ; 0498
                                            58  DD 0049C          PUSHL   R8                                        ; 0500
                                7E          26  7D 0049E          MOVQ    #38, -(SP)                                ;
                                            7E  7C 004A1          CLRQ    -(SP)                                     ;
                                7E          01  CE 004A3          MNEGL   #1, -(SP)                                 ;
                    00000000G   EF          06  FB 004A6          CALLS   #6, BOO$QIO                               ;
                                66          50  D0 004AD          MOVL    R0, (R6)                                  ;
                                74          66  E9 004B0          BLBC    (R6), 52$                                 ; 0501
                                            58  DD 004B3          PUSHL   R8                                        ; 0502
                                7E          26  7D 004B5          MOVQ    #38, -(SP)                                ;
                                            7E  7C 004B8          CLRQ    -(SP)                                     ;
                                7E          01  CE 004BA          MNEGL   #1, -(SP)                                 ;
                    00000000G   EF          06  FB 004BD          CALLS   #6, BOO$QIO                               ;
                                66          50  D0 004C4          MOVL    R0, (R6)                                  ;
                                76          66  E9 004C7 47$:     BLBC    (R6), 53$                                 ; 0503
                                            58  DD 004CA          PUSHL   R8                                        ; 0504
                                7E          26  7D 004CC          MOVQ    #38, -(SP)                                ;
                                            7E  7C 004CF          CLRQ    -(SP)                                     ;
                                7E          01  CE 004D1          MNEGL   #1, -(SP)                                 ;
                    00000000G   EF          06  FB 004D4          CALLS   #6, BOO$QIO                               ;
                                66          50  D0 004DB          MOVL    R0, (R6)                                  ;
                                7D          66  E9 004DE          BLBC    (R6), 55$                                 ; 0505
                                          00A9  31 004E1          BRW     57$                                       ;
                                      00AC  CE  95 004E4 48$:     TSTB    FIRST1                                    ; 0510
                                            09  12 004E8          BNEQ    49$                                       ;
          00AC   CE       B0    AD    0050  8F  28 004EA          MOVC3   #80, HDR1, FIRST1                         ;
                    34524448    8F    FF10  CD  D1 004F3 49$:     CMPL    HDR4, #877806664                          ; 0512
                                            16  12 004FC          BNEQ    51$                                       ;
                                            58  DD 004FE          PUSHL   R8                                        ; 0514
                                7E          25  7D 00500          MOVQ    #37, -(SP)                                ;
                                            7E  7C 00503          CLRQ    -(SP)                                     ;
                                            01  DD 00505          PUSHL   #1                                        ;
                    00000000G   EF          06  FB 00507          CALLS   #6, BOO$QIO                               ;
```

```
                        66          50  D0 0050E          MOVL    R0, (R6)                                  ;
                        4A          66  E9 00511 50$:     BLBC    (R6), 55$                                 ; 0515
                                    58  DD 00514 51$:     PUSHL   R8                                        ; 0518
                        7E          25  7D 00516          MOVQ    #37, -(SP)                                ;
                                    7E  7C 00519          CLRQ    -(SP)                                     ;
                                    01  DD 0051B          PUSHL   #1                                        ;
            00000000G   EF          06  FB 0051D          CALLS   #6, BOO$QIO                               ;
                        66          50  D0 00524          MOVL    R0, (R6)                                  ;
                        76          66  E9 00527 52$:     BLBC    (R6), 61$                                 ; 0520
                                    58  DD 0052A          PUSHL   R8                                        ; 0522
                        7E          25  7D 0052C          MOVQ    #37, -(SP)                                ;
                                    7E  7C 0052F          CLRQ    -(SP)                                     ;
                                    01  DD 00531          PUSHL   #1                                        ;
            00000000G   EF          06  FB 00533          CALLS   #6, BOO$QIO                               ;
                        66          50  D0 0053A          MOVL    R0, (R6)                                  ;
                        5A          66  E8 0053D          BLBS    (R6), 59$                                 ; 0524
                                    5E  11 00540 53$:     BRB     61$                                       ;
            00000838    8F          66  D1 00542 54$:     CMPL    (R6), #2104                               ; 0533
                                    1B  12 00549          BNEQ    56$                                       ;
                                    58  DD 0054B          PUSHL   R8                                        ; 0535
                        7E          25  7D 0054D          MOVQ    #37, -(SP)                                ;
                                    7E  7C 00550          CLRQ    -(SP)                                     ;
                                    01  DD 00552          PUSHL   #1                                        ;
            00000000G   EF          06  FB 00554          CALLS   #6, BOO$QIO                               ;
                        66          50  D0 0055B          MOVL    R0, (R6)                                  ;
                        3F          66  E9 0055E 55$:     BLBC    (R6), 61$                                 ; 0537
                        5A          04  8A 00561          BICB2   #4, FLAG                                  ; 0539
                                    37  11 00564          BRB     60$                                       ;
            00000870    8F          66  D1 00566 56$:     CMPL    (R6), #2160                               ; 0542
                                    31  12 0056D          BNEQ    61$                                       ;
        27              5A          02  E1 0056F          BBC     #2, FLAG, 59$                             ; 0545
                                    58  DD 00573          PUSHL   R8                                        ; 0548
                        7E          24  7D 00575          MOVQ    #36, -(SP)                                ;
                                    7E  7C 00578          CLRQ    -(SP)                                     ;
                                    7E  D4 0057A          CLRL    -(SP)                                     ;
            00000000G   EF          06  FB 0057C          CALLS   #6, BOO$QIO                               ;
                        66          50  D0 00583          MOVL    R0, (R6)                                  ;
                        5A          04  8A 00586          BICB2   #4, FLAG                                  ; 0549
        08              5A          01  E1 00589          BBC     #1, FLAG, 58$                             ; 0551
                        50 00018292 8F  D0 0058D 57$:     MOVL    #98962, R0                                ;
                                        04 00594          RET                                               ;
                        5A          02  88 00595 58$:     BISB2   #2, FLAG                                  ;
                                    03  11 00598          BRB     60$                                       ;
                        5A          04  88 0059A 59$:     BISB2   #4, FLAG                                  ; 0555
                                  FB08  31 0059D 60$:     BRW     4$                                        ; 0545
                        50 0001C0F4 8F  D0 005A0 61$:     MOVL    #114932, R0                               ; 0560
                                        04 005A7          RET                                               ; 0566

; Routine Size:  1448 bytes,    Routine Base:  CODE + 0000

;   0567  1
```

```
;    0569  1      %sbttl 'ansi$close'
;    0570  1
;    0571  1      global routine ansi$close : call_rms =
;    0572  1
;    0573  1      !++
;    0574  1      ! Functional description:
;    0575  1      !
;    0576  1      !       This routine is used to close a file. It positions the tape
;    0577  1      !       to the final end-of-file mark.
;    0578  1      !       This positioning is important since if the same file
;    0579  1      !       is opened again the tape will already be positioned to the EOF1
;    0580  1      !       record.
;    0581  1      !
;    0582  1      ! Formal parameters:    none
;    0583  1      !
;    0584  1      ! Implicit inputs:
;    0585  1      !
;    0586  1      !       FAB     Address of FAB
;    0587  1      !       cblock  Address of perfile data
;    0588  1      !
;    0589  1      ! Implicit outputs:
;    0590  1      !
;    0591  1      ! Routine value:
;    0592  1      !
;    0593  1      !       RMS completion codes
;    0594  1      !
;    0595  1      ! Side effects:
;    0596  1      !
;    0597  1      !       The tape may be moved
;    0598  1      !
;    0599  1      !--
;    0600  1
;    0601  2          begin
;    0602  2
;    0603  2          external register
;    0604  2              cblock = 9 : ref block [, byte],
;    0605  2              fab = 11 : ref $fab_decl;
;    0606  2
;    0607  2          external routine                                    !                   [06]
;    0608  2              boo$qio : addressing_mode (long_relative);      !                   [06]
;    0609  2
;    0610  2          bind
;    0611  2              status = fab [fab$l_stv];
;    0612  2
;    0613  2          kb_check ();
;    0614  2          status = boo$qio (1, 0, 0, io$_skipfile, 0, .cblock [ctl$l_rpb]);
;    0615  2
;    0616  2          if not .status then return rms$_ccf;
;    0617  2
;    0618  2          return rms$_normal;
;    0619  1          end;


                                05FC 00000          .ENTRY  ANSI$CLOSE, Save R2,R3,R4,R5,R6,R7,R8,R10    ; 0571
```

```
                                00000000G  EF  16 00002          JSB     KB_CHECK                                ; 0613
                                      38   A9  DD 00008          PUSHL   56(CBLOCK)                              ; 0614
                                7E         25  7D 0000B          MOVQ    #37, -(SP)                              ;
                                           7E  7C 0000E          CLRQ    -(SP)                                   ;
                                           01  DD 00010          PUSHL   #1                                      ;
                           00000000G  EF         06  FB 00012    CALLS   #6, BOO$QIO                             ;
                                 0C   AB         50  D0 00019    MOVL    R0, 12(FAB)                             ;
                                      08     0C   AB  E8 0001D    BLBS    12(FAB), 1$                             ; 0616
                                      50 0001C0DC 8F  D0 00021    MOVL    #114908, R0                             ;
                                                      04 00028    RET                                             ;
                                      50 00010001 8F  D0 00029 1$: MOVL    #65537, R0                              ; 0618
                                                      04 00030    RET                                             ; 0619

; Routine Size:  49 bytes,    Routine Base:  CODE + 05A8

;   0620  1
```

```
;    0622  1      %sbttl 'ansi$cachevbn'
;    0623  1
;    0624  1      global routine ansi$CACHEvbn (vbn) : call_rms =
;    0625  1
;    0626  1      !++
;    0627  1      ! Functional description:
;    0628  1      !
;    0629  1      !       This routine is used to place the desired block in the cache
;    0630  1      !       If necessary the tape is re-positioned to the desired block
;    0631  1      !       and the cache updated to include the desired data
;    0632  1      !
;    0633  1      ! Formal parameters:
;    0634  1      !
;    0635  1      !       VBN     VBN of block desired in the cache
;    0636  1      !
;    0637  1      ! Implicit inputs:
;    0638  1      !
;    0639  1      !       cblock  Address of per-file area
;    0640  1      !
;    0641  1      ! Implicit outputs:
;    0642  1      !
;    0643  1      !       cblock  The cache may be updated
;    0644  1      !
;    0645  1      ! Routine value:
;    0646  1      !
;    0647  1      !       Success or failure RMS codes and I/O failure codes
;    0648  1      !
;    0649  1      ! Side effects:
;    0650  1      !
;    0651  1      !       The device and controller may be accessed to read data from the tape.
;    0652  1      !
;    0653  1      !--
;    0654  1
;    0655  2          begin
;    0656  2
;    0657  2          external register
;    0658  2              cblock = 9 : ref block [, byte];        !control region
;    0659  2
;    0660  2          external routine                                !                       [06]
;    0661  2              boo$qio : addressing_mode (long_relative);  !                       [06]
;    0662  2
;    0663  2          local
;    0664  2              status;                                 !Status of boo$qio calls
;    0665  2
;    0666  2          if .vbn geq .cblock [ctl$l_eofvbn] then return ss$_endoffile;
;    0667  2
;    0668  3          if (.vbn lss .cblock [ctl$l_lvbn]) or (.vbn geq .cblock [ctl$l_hvbn])
;    0669  2          then
;    0670  2                                                  !This block is not in the cache
;    0671  3              begin
;    0672  3
;    0673  3              while .cblock [ctl$l_tapepos] gtr .vbn do
;    0674  4                  begin                           !Backup until the block found
;    0675  4                  kb_check ();
;    0676  4                  status = boo$qio (-1, 0, 0, io$_skiprecord, 0, .cblock [ctl$l_rpb]);
;    0677  4
;    0678  4                  if not .status then return .status;
```

```
;   0679  4
;   0680  4                 cblock [ctl$l_tapepos] = .cblock [ctl$l_tapepos] - .cblock [ctl$w_cache_size];
;   0681  4                 cblock [ctl$l_lvbn] = cblock [ctl$l_hvbn] = .cblock [ctl$l_tapepos];
;   0682  3                 end;
;   0683  3
;   0684  3             until (.vbn geq .cblock [ctl$l_lvbn]) and (.vbn lss .cblock [ctl$l_hvbn]) do
;   0685  4                 begin                                   !Read forward until desired data is in the cache
;   0686  4
;   0687  4                 local
;   0688  4                     cache;                              !Address of cache block
;   0689  4
;   0690  4                 kb_check ();
;   0691  4
;   0692  4                 case (.cblock [ctl$l_hvbn] - .cblock [ctl$l_lvbn])/.cblock [ctl$w_cache_size] from 0 to 3 of
;   0693  4                     set
;   0694  4
;   0695  4                     [0] :
;   0696  4                         cache = .cblock [ctl$l_cache1];
;   0697  4
;   0698  4                     [1] :
;   0699  4                         cache = .cblock [ctl$l_cache2];
;   0700  4
;   0701  4                     [2] :
;   0702  4                         cache = .cblock [ctl$l_cache3];
;   0703  4
;   0704  4                     [3] :
;   0705  5                         begin                           !Rotate cache in use
;   0706  5                         cache = .cblock [ctl$l_cache1];
;   0707  5                         cblock [ctl$l_cache1] = .cblock [ctl$l_cache2];
;   0708  5                         cblock [ctl$l_cache2] = .cblock [ctl$l_cache3];
;   0709  5                         cblock [ctl$l_cache3] = .cache;
;   0710  5                         cblock [ctl$l_lvbn] = .cblock [ctl$l_lvbn] + .cblock [ctl$w_cache_size];
;   0711  4                         end;
;   0712  4                     tes;
;   0713  4
;   0714  4                 ch$fill (%c'^', 12*.cblock [ctl$w_cache_size], .cache);
;   0715  4                 status = boo$qio (.cache, .cblock [ctl$w_cache_size]*512, 0, io$_readlblk, 0,
;   0716  4                     .cblock [ctl$l_rpb]);
;   0717  4
;   0718  4                 if not .status then return .status;
;   0719  4
;   0720  4                 cblock [ctl$l_tapepos] = .cblock [ctl$l_tapepos] + .cblock [ctl$w_cache_size];
;   0721  4                 cblock [ctl$l_hvbn] = min (.cblock [ctl$l_eofvbn] + 1,
;   0722  4                     .cblock [ctl$l_hvbn] + .cblock [ctl$w_cache_size]);
;   0723  3                 end;
;   0724  3
;   0725  2             end;
;   0726  2
;   0727  2         ss$_normal
;   0728  1         end;
```

```
                                                0DFC 00000          .ENTRY  ANSI$CACHEVBN, Save R2,R3,R4,R5,R6,R7,R8,-   ; 0624
                                                                            R10,R11                                      ;
```

```
                                 5E          04  C2 00002        SUBL2   #4, SP                                  ;
                                 5B      04  AC  D0 00005        MOVL    VBN, R11                                ; 0666
                            44   A9          5B  D1 00009        CMPL    R11, 68(CBLOCK)                         ;
                                             06  19 0000D        BLSS    1$                                      ;
                                 50    0870  8F  3C 0000F        MOVZWL  #2160, R0                               ;
                                             04  00014           RET                                             ;
                            28   A9          5B  D1 00015 1$:    CMPL    R11, 40(CBLOCK)                         ; 0668
                                             06  19 00019        BLSS    2$                                      ;
                            24   A9          5B  D1 0001B        CMPL    R11, 36(CBLOCK)                         ;
                                             4A  19 0001F        BLSS    6$                                      ;
                                 57      58  A9  9E 00021 2$:    MOVAB   88(CBLOCK), R7                          ; 0673
                                 58      0A  A9  9E 00025        MOVAB   10(CBLOCK), R8                          ; 0680
                                 5A      24  A9  9E 00029        MOVAB   36(CBLOCK), R10                         ; 0681
                                 5B          67  D1 0002D 3$:    CMPL    (R7), R11                               ; 0673
                                             30  15 00030        BLEQ    5$                                      ;
                                 00000000G   EF  16 00032        JSB     KB_CHECK                                ; 0675
                                         38  A9  DD 00038        PUSHL   56(CBLOCK)                              ; 0676
                                 7E          26  7D 0003B        MOVQ    #38, -(SP)                              ;
                                             7E  7C 0003E        CLRQ    -(SP)                                   ;
                                 7E          01  CE 00040        MNEGL   #1, -(SP)                               ;
                       00000000G EF          06  FB 00043        CALLS   #6, BOO$QIO                             ;
                                 6E          50  D0 0004A        MOVL    R0, STATUS                              ;
                                 03          6E  E8 0004D        BLBS    STATUS, 4$                              ; 0678
                                           008B  31 00050        BRW     14$                                     ;
                                 50          68  3C 00053 4$:    MOVZWL  (R8), R0                                ; 0680
                                 67          50  C2 00056        SUBL2   R0, (R7)                                ;
                                 6A          67  D0 00059        MOVL    (R7), (R10)                             ; 0681
                            28   A9          6A  D0 0005C        MOVL    (R10), 40(CBLOCK)                       ;
                                             CB  11 00060        BRB     3$                                      ;
                            28   A9          5B  D1 00062 5$:    CMPL    R11, 40(CBLOCK)                         ; 0684
                                             08  19 00066        BLSS    7$                                      ;
                                 6A          5B  D1 00068        CMPL    R11, (R10)                              ;
                                             03  18 0006B 6$:    BGEQ    7$                                      ;
                                           0091  31 0006D        BRW     17$                                     ;
                                 00000000G   EF  16 00070 7$:    JSB     KB_CHECK                                ; 0690
                     50          6A      28  A9  C3 00076        SUBL3   40(CBLOCK), (R10), R0                   ; 0692
                                 51          68  3C 0007B        MOVZWL  (R8), R1                                ;
                                 50          51  C6 0007E        DIVL2   R1, R0                                  ;
                     03          00          50  CF 00081        CASEL   R0, #0, #3                              ;
 001A           0014         000E         0008     00085 8$:    .WORD   9$-8$,-                                 ;
                                                                         10$-8$,-                                ;
                                                                         11$-8$,-                                ;
                                                                         12$-8$                                  ;
                                 56      2C  A9  D0 0008D 9$:    MOVL    44(CBLOCK), CACHE                       ; 0696
                                             20  11 00091        BRB     13$                                     ;
                                 56      30  A9  D0 00093 10$:   MOVL    48(CBLOCK), CACHE                       ; 0699
                                             1A  11 00097        BRB     13$                                     ;
                                 56      34  A9  D0 00099 11$:   MOVL    52(CBLOCK), CACHE                       ; 0702
                                             14  11 0009D        BRB     13$                                     ;
                                 56      2C  A9  D0 0009F 12$:   MOVL    44(CBLOCK), CACHE                       ; 0706
                            2C   A9      30  A9  7D 000A3        MOVQ    48(CBLOCK), 44(CBLOCK)                  ; 0707
                            34   A9          56  D0 000A8        MOVL    CACHE, 52(CBLOCK)                       ; 0709
                                 50          68  3C 000AC        MOVZWL  (R8), R0                                ; 0710
                            28   A9          50  C0 000AF        ADDL2   R0, 40(CBLOCK)                          ;
                                 50          68  3C 000B3 13$:   MOVZWL  (R8), R0                                ; 0714
                                 50          0C  C4 000B6        MULL2   #12, R0                                 ;
  50        5E    8F             6E          00  2C 000B9        MOVC5   #0, (SP), #94, R0, (CACHE)              ;
```

```
                                                    66     000BF                                                                ;
                                                38  A9  DD 000C0          PUSHL   56(CBLOCK)                                  ; 0716
                                                    21  7D 000C3          MOVQ    #33, -(SP)                                  ; 0715
                                                    7E  D4 000C6          CLRL    -(SP)                                       ;
                                                    68  3C 000C8          MOVZWL  (R8), R0                                    ;
                                                    09  78 000CB          ASHL    #9, R0, -(SP)                               ;
                                                    56  DD 000CF          PUSHL   CACHE                                       ;
                                                    06  FB 000D1          CALLS   #6, BOO$QIO                                 ;
                                                    6E  50 D0 000D8       MOVL    R0, STATUS                                  ;
                                                    04  6E E8 000DB       BLBS    STATUS, 15$                                 ; 0718
                                                    50  6E D0 000DE 14$:   MOVL    STATUS, R0                                  ;
                                                        04 000E1          RET                                                 ;
                                                    50  68 3C 000E2 15$:   MOVZWL  (R8), R0                                    ; 0720
                                                    67  50 C0 000E5       ADDL2   R0, (R7)                                    ;
                                                51  44  A9  01 C1 000E8    ADDL3   #1, 68(CBLOCK), R1                          ; 0721
                                                    50  68 3C 000ED       MOVZWL  (R8), R0                                    ; 0722
                                                    50  6A C0 000F0       ADDL2   (R10), R0                                   ;
                                                    50  51 D1 000F3       CMPL    R1, R0                                      ;
                                                        03 15 000F6       BLEQ    16$                                         ;
                                                    51  50 D0 000F8       MOVL    R0, R1                                      ;
                                                    6A  51 D0 000FB 16$:   MOVL    R1, (R10)                                   ; 0721
                                                FF61    31 000FE          BRW     5$                                          ;
                                                    50  01 D0 00101 17$:   MOVL    #1, R0                                      ; 0728
                                                        04 00104          RET                                                 ;

; Routine Size:  261 bytes,    Routine Base:  CODE + 05D9


;   0729  1
```

```
;    0731  1      %sbttl 'ansi$read'
;    0732  1
;    0733  1      global routine ansi$read : call_rms =
;    0734  1
;    0735  1      !++
;    0736  1      ! Functional description:
;    0737  1      !
;    0738  1      !       This routine is used to read a block from an open file.
;    0739  1      !       It uses the block number in the RAB to select the next block
;    0740  1      !       to be read.  Data blocks read from the tape are cached so
;    0741  1      !       this may not require a tape access.  Most of the work to fill
;    0742  1      !       and update the cache is done by ANSI$cacheVBN.
;    0743  1      !
;    0744  1      ! Formal parameters:    none
;    0745  1      !
;    0746  1      ! Implicit inputs:
;    0747  1      !
;    0748  1      !       FAB     Address of FAB for file
;    0749  1      !       RAB     Address of RAB from caller
;    0750  1      !       cblock  Address of per-file area
;    0751  1      !
;    0752  1      ! Implicit outputs:
;    0753  1      !
;    0754  1      !       FAB     Unaffected
;    0755  1      !       RAB     The current position is updated
;    0756  1      !       cblock  The cache may be updates
;    0757  1      !
;    0758  1      ! Routine value:
;    0759  1      !
;    0760  1      !       RMS completion codes
;    0761  1      !
;    0762  1      ! Side effects:
;    0763  1      !
;    0764  1      !       The device and controller may be accessed to read data from the tape.
;    0765  1      !
;    0766  1      !--
;    0767  1
;    0768  2          begin
;    0769  2
;    0770  2          external register
;    0771  2              cblock = 9 : ref block [, byte],
;    0772  2              fab = 11 : ref $fab_decl,
;    0773  2              rab = 10 : ref $rab_decl;
;    0774  2
;    0775  2          external routine                                !                       [06]
;    0776  2              boo$qio : addressing_mode (long_relative);      !                   [06]
;    0777  2
;    0778  2          local
;    0779  2              cache : ref vector [, byte],                !Address of current cache block
;    0780  2              user_size,                                  !Size to be transfered
;    0781  2              user_address,                               !Address to receive data
;    0782  2              vbn;                                        !Current desired VBN
;    0783  2
;    0784  2          bind
;    0785  2              status = rab [rab$l_stv];                   !Current status return value
;    0786  2
;    0787  2          vbn = (if .rab [rab$l_bkt] eql 0 then .cblock [ctl$l_nbp] else .rab [rab$l_bkt]);
```

```
;    0788  2          user_size = .rab [rab$w_usz];
;    0789  2          user_address = rab [rab$l_rbf] = .rab [rab$l_ubf];
;    0790  2          rab [rab$w_rsz] = 0;
;    0791  2          rab [rab$l_rfa0] = .vbn;
;    0792  2          rab [rab$w_rfa4] = 0;
;    0793  2          status = rms$_normal;
;    0794  2
;    0795  2          while .user_size gtr 0 do
;    0796  3              begin
;    0797  3
;    0798  3              local
;    0799  3                  cache_offset,                   !offset within cache
;    0800  3                  cache_section;                  !Index to current section of cache
;    0801  3
;    0802  3              status = ansi$cachevbn (.vbn);
;    0803  3
;    0804  3              if not .status then exitloop;
;    0805  3
;    0806  3              cache_offset = (.vbn - .cblock [ctl$l_lvbn]);
;    0807  3              cache_section = .cache_offset/.cblock [ctl$w_cache_size];
;    0808  3              cache_offset = (.cache_offset - .cache_section*.cblock [ctl$w_cache_size])*512;
;    0809  3              cache = .(cblock [ctl$l_cache1] + .cache_section*4);
;    0810  3              ch$move (min (512, .user_size), cache [.cache_offset], .user_address);
;    0811  3              user_address = .user_address + min (512, .user_size);
;    0812  3              rab [rab$w_rsz] = .rab [rab$w_rsz] + min (512, .user_size);
;    0813  3              user_size = .user_size - min (512, .user_size);
;    0814  3              vbn = .vbn + 1;
;    0815  2              end;
;    0816  2
;    0817  2          cblock [ctl$l_nbp] = .vbn;
;    0818  2
;    0819  2          if .status and %x'FFFF' neq 0
;    0820  2          then
;    0821  2              .status
;    0822  2          else
;    0823  2
;    0824  2              if .status eql ss$_endoffile then rms$_normal else rms$_rer
;    0825  2
;    0826  1          end;


                                    01FC 00000          .ENTRY  ANSI$READ, Save R2,R3,R4,R5,R6,R7,R8        ; 0733
                              5E        08 C2 00002       SUBL2   #8, SP                                      ;
                           0C AA  9F    00005          PUSHAB  12(RAB)                                     ; 0785
                           38 AA  D5    00008          TSTL    56(RAB)                                     ; 0787
                              07  12    0000B          BNEQ    1$                                          ;
                    04 AE     04 A9  D0  0000D          MOVL    4(CBLOCK), VBN                              ;
                              05  11    00012          BRB     2$                                          ;
                    04 AE     38 AA  D0  00014 1$:      MOVL    56(RAB), VBN                                ;
                       56     20 AA  3C  00019 2$:      MOVZWL  32(RAB), USER_SIZE                          ; 0788
                       50     24 AA  D0  0001D          MOVL    36(RAB), R0                                 ; 0789
                    28 AA        50  D0  00021          MOVL    R0, 40(RAB)                                 ;
                    08 AE        50  D0  00025          MOVL    R0, USER_ADDRESS                            ;
                           22 AA  B4    00029          CLRW    34(RAB)                                     ; 0790
```

```
                    10   AA        04   AE  D0 0002C          MOVL    VBN, 16(RAB)                                  ; 0791
                                   14   AA  B4 00031          CLRW    20(RAB)                                       ; 0792
                    00   BE 00010001   8F  D0 00034          MOVL    #65537, a0(SP)                                ; 0793
                                       56  D5 0003C 3$:      TSTL    USER_SIZE                                     ; 0795
                                       59  15 0003E          BLEQ    5$                                            ;
                                   04   AE  DD 00040          PUSHL   VBN                                           ; 0802
                  FEB3   CF            01  FB 00043          CALLS   #1, ANSI$CACHEVBN                             ;
                    00   BE            50  D0 00048          MOVL    R0, a0(SP)                                    ;
                         49        00   BE  E9 0004C          BLBC    a0(SP), 5$                                    ; 0804
          51        04   AE        28   A9  C3 00050          SUBL3   40(CBLOCK), VBN, CACHE_OFFSET                 ; 0806
                         50        0A   A9  3C 00056          MOVZWL  10(CBLOCK), CACHE_SECTION                     ; 0807
          50             51            50  C7 0005A          DIVL3   CACHE_SECTION, CACHE_OFFSET, CACHE_SECTION    ;
                         52        0A   A9  3C 0005E          MOVZWL  10(CBLOCK), R2                                ; 0808
                         52            50  C4 00062          MULL2   CACHE_SECTION, R2                             ;
          52             51            52  C3 00065          SUBL3   R2, CACHE_OFFSET, R2                          ;
          51             52            09  78 00069          ASHL    #9, R2, CACHE_OFFSET                          ;
                         58       2C A940  D0 0006D          MOVL    44(CBLOCK)[CACHE_SECTION], CACHE              ; 0809
                         57            56  D0 00072          MOVL    USER_SIZE, R7                                 ; 0810
             00000200    8F            57  D1 00075          CMPL    R7, #512                                      ;
                                       05  15 0007C          BLEQ    4$                                            ;
                         57      0200   8F  3C 0007E          MOVZWL  #512, R7                                      ;
     08   BE           6148            57  28 00083 4$:      MOVC3   R7, (CACHE_OFFSET)[CACHE], aUSER_ADDRESS      ;
                    08   AE            57  C0 00089          ADDL2   R7, USER_ADDRESS                              ; 0811
                    22   AA            57  A0 0008D          ADDW2   R7, 34(RAB)                                   ; 0812
                         56            57  C2 00091          SUBL2   R7, USER_SIZE                                 ; 0813
                                   04   AE  D6 00094          INCL    VBN                                           ; 0814
                                       A3  11 00097          BRB     3$                                            ;
                    04   A9        04   AE  D0 00099 5$:      MOVL    VBN, 4(CBLOCK)                                ; 0817
                         05        00   BE  E9 0009E          BLBC    a0(SP), 6$                                    ; 0819
                         50        00   BE  D0 000A2          MOVL    a0(SP), R0                                    ; 0821
                                           04 000A6          RET                                                   ;
             00000870    8F        00   BE  D1 000A7 6$:      CMPL    a0(SP), #2160                                 ; 0824
                                       08  12 000AF          BNEQ    7$                                            ;
                         50 00010001   8F  D0 000B1          MOVL    #65537, R0                                    ;
                                           04 000B8          RET                                                   ;
                         50 0001C0F4   8F  D0 000B9 7$:      MOVL    #114932, R0                                   ;
                                           04 000C0          RET                                                   ; 0826

; Routine Size:  193 bytes,    Routine Base:  CODE + 06DE

;   0827  1
```

```
;    0829  1      %sbttl 'ansi$get'
;    0830  1
;    0831  1      global routine ansi$get : call_rms =
;    0832  1
;    0833  1      !++
;    0834  1      ! Functional description:
;    0835  1      !
;    0836  1      !       This routine transfers the next record from the tape into
;    0837  1      !       the callers buffer. Only sequential mode is allowed.  ansi$cachevbn
;    0838  1      !       may be called to read more data from the tape.  The routine does
;    0839  1      !       the necessary deblocking of records from ANS X3.27 format tapes.
;    0840  1      !
;    0841  1      ! Formal parameters:    none
;    0842  1      !
;    0843  1      ! Implicit inputs:
;    0844  1      !
;    0845  1      !       FAB     Address of related open file block
;    0846  1      !       RAB     Address of RAB
;    0847  1      !       cblock  Address of per-file area
;    0848  1      !
;    0849  1      ! Implicit outputs:
;    0850  1      !
;    0851  1      !       FAB     unaffected
;    0852  1      !       RAB     the current position is updated
;    0853  1      !       cblock  Various fields may be updated
;    0854  1      !
;    0855  1      ! Routine value:
;    0856  1      !
;    0857  1      !       RMS completion codes
;    0858  1      !
;    0859  1      ! Side effects:
;    0860  1      !
;    0861  1      !       The device and associated channels may be accessed to read the record.
;    0862  1      !
;    0863  1      !--
;    0864  1
;    0865  2          begin
;    0866  2
;    0867  2          external register
;    0868  2              cblock = 9 : ref block [, byte],        !Base of control block
;    0869  2              rab = 10 : REF $RAB_DECL,               !Base of current rab
;    0870  2              fab = 11 : ref $fab_DECL;               !Base of current fab
;    0871  2
;    0872  2          external routine                            !                          [06]
;    0873  2              boo$qio : addressing_mode (long_relative);      !                  [06]
;    0874  2
;    0875  2          local
;    0876  2              boff,                                   !Current byte offset in page
;    0877  2              vbn;                                    !Current desired VBN
;    0878  2
;    0879  2          bind
;    0880  2              status = rab [rab$l_stv];               !Status returned by calls
;    0881  2
;    0882  2          vbn = .cblock [ctl$l_vbn];                  !Fetch current virtual block
;    0883  2          boff = .cblock [ctl$w_byte];                !Fetch byte offset
;    0884  2
;    0885  2          if .rab [rab$b_rac] eql rab$c_seq           !If sequential access
```

```
;    0886  2          then                                    !Update position based on RFA
;    0887  3              begin
;    0888  3              vbn = .vbn + (.cblock [ctl$w_rec_size] + .boff)/(512*.cblock [ctl$w_cache_size]);
;    0889  3              boff = (.cblock [ctl$w_rec_size] + .boff) mod (512*.cblock [ctl$w_cache_size]);
;    0890  2              end;
;    0891  2
;    0892  2          do
;    0893  3              begin
;    0894  3
;    0895  3              local
;    0896  3                  cache : ref vector [, byte],        !Address of current cache block
;    0897  3                  cache_offset,                       !offset within cache
;    0898  3                  cache_section;                      !Index to current section of cache
;    0899  3
;    0900  3              if not (status = ansi$cachevbn (.vbn)) then exitloop;
;    0901  3
;    0902  3              cache_offset = .vbn - .cblock [ctl$l_lvbn];
;    0903  3              cache_section = .cache_offset/.cblock [ctl$w_cache_size];
;    0904  3              cache_offset = .cache_offset - .cache_section*.cblock [ctl$w_cache_size]*512;
;    0905  3              cache = .(cblock [ctl$l_cache1] + .cache_section*4);
;    0906  3              rab [rab$l_rbf] = .rab [rab$l_ubf];
;    0907  3
;    0908  3              case .fab [fab$b_rfm] from fab$c_fix to fab$c_vfc of
;    0909  3                  set
;    0910  3
;    0911  3                  [fab$c_fix] :
;    0912  4                      begin
;    0913  4
;    0914  5                      if (.boff + .rab [fab$w_mrs]) geq (.cblock [ctl$w_cache_size]*512)
;    0915  4                      then
;    0916  5                          begin
;    0917  5                          vbn = .vbn + .cblock [ctl$w_cache_size];
;    0918  5                          boff = 0;
;    0919  5                          status = 0
;    0920  5                          end
;    0921  4                      else
;    0922  5                          begin
;    0923  5                          cblock [ctl$w_rec_size] = .fab [fab$w_mrs];
;    0924  5
;    0925  5                          if .rab [rab$w_usz] lss .cblock [ctl$w_rec_size] - 4
;    0926  5                          then
;    0927  6                              begin
;    0928  6                              status = rms$_rtb;
;    0929  6                              rab [rab$w_rsz] = .rab [rab$w_usz];
;    0930  6                              end
;    0931  5                          else
;    0932  6                              begin
;    0933  6                              status = rms$_normal;
;    0934  6                              rab [rab$w_rsz] = .cblock [ctl$w_rec_size];
;    0935  5                              end;
;    0936  5
;    0937  5                          ch$move (.rab [rab$w_rsz], cache [.boff], .rab [rab$l_rbf]);
;    0938  5                          status = rms$_normal;
;    0939  4                          end;
;    0940  4
;    0941  3                      end;
;    0942  3
```

```
;    0943  3                    [fab$c_var] :
;    0944  4                        begin
;    0945  4
;    0946  5                        if (.cache [.boff] lss %c'0') or (.cache [.boff] gtr %c'9')
;    0947  4                        then
;    0948  5                            begin                           !Not valid record go to next block
;    0949  5                            boff = 0;
;    0950  5                            vbn = .vbn + .cblock [ctl$w_cache_size];
;    0951  5                            status = 0;
;    0952  5                            end
;    0953  4                        else
;    0954  5                            begin                           !Valid record copy it
;    0955  5                            cblock [ctl$w_rec_size] = scan$numeric (10, 4, cache [.boff]);
;    0956  5
;    0957  5                            if .rab [rab$w_usz] lss .cblock [ctl$w_rec_size] - 4
;    0958  5                            then
;    0959  6                                begin
;    0960  6                                status = rms$_rtb;
;    0961  6                                rab [rab$w_rsz] = .rab [rab$w_usz];
;    0962  6                                end
;    0963  5                            else
;    0964  6                                begin
;    0965  6                                status = rms$_normal;
;    0966  6                                rab [rab$w_rsz] = .cblock [ctl$w_rec_size] - 4;
;    0967  5                                end;
;    0968  5
;    0969  5                            ch$move (.rab [rab$w_rsz], cache [.boff + 4], .rab [rab$l_rbf]);
;    0970  5                            status = rms$_normal;
;    0971  4                            end;
;    0972  4
;    0973  3                        end;
;    0974  3
;    0975  3                    [fab$c_vfc] :
;    0976  4                        begin
;    0977  4
;    0978  5                        if (.cache [.boff] lss %c'0') or (.cache [.boff] gtr %c'9')
;    0979  4                        then
;    0980  5                            begin                           !Not valid record go to next block
;    0981  5                            boff = 0;
;    0982  5                            vbn = .vbn + .cblock [ctl$w_cache_size];
;    0983  5                            status = 0;
;    0984  5                            end
;    0985  4                        else
;    0986  5                            begin                           !Valid record copy it
;    0987  5                            cblock [ctl$w_rec_size] = scan$numeric (10, 4, cache [.boff]);
;    0988  5
;    0989  5                            if .rab [rab$w_usz] lss .cblock [ctl$w_rec_size] - .fab [fab$b_fsz] - 4
;    0990  5                            then
;    0991  6                                begin
;    0992  6                                status = rms$_rtb;
;    0993  6                                rab [rab$w_rsz] = .rab [rab$w_usz] - .rab [fab$b_fsz];
;    0994  6                                end
;    0995  5                            else
;    0996  6                                begin
;    0997  6                                status = rms$_normal;
;    0998  6                                rab [rab$w_rsz] = .cblock [ctl$w_rec_size] - .fab [fab$b_fsz] - 4;
;    0999  5                                end;
```

```
;   1000  5
;   1001  5                         if .rab [rab$l_rhb] neq 0
;   1002  5                         then
;   1003  5                             ch$move (.fab [fab$b_fsz], cache [.boff + 4],
;   1004  5                                 .rab [rab$l_rhb]);
;   1005  5
;   1006  5                         ch$move (.rab [rab$w_rsz], cache [.boff + .fab [fab$b_fsz] + 4], .rab [rab$l_rbf]);
;   1007  5                         status = rms$_normal;
;   1008  4                         end;
;   1009  4
;   1010  3                     end;
;   1011  3
;   1012  3                 [outrange] :
;   1013  3                     return rms$_org;
;   1014  3                 tes;
;   1015  3
;   1016  3             end
;   1017  2         until .status;
;   1018  2
;   1019  2         cblock [ctl$l_vbn] = .vbn;
;   1020  2         cblock [ctl$w_byte] = .boff;
;   1021  2
;   1022  2         if .status and %x'FFFF' NEQ 0
;   1023  2         then
;   1024  2             rms$_normal
;   1025  2         else
;   1026  2
;   1027  2             if .status eql ss$_endoffile then rms$_eof else rms$_rer
;   1028  2
;   1029  1         end;


                                        01FC 00000          .ENTRY  ANSI$GET, Save R2,R3,R4,R5,R6,R7,R8          ; 0831
                          5E                04  C2 00002      SUBL2   #4, SP                                       ;
                          58          0C    AA  9E 00005      MOVAB   12(RAB), R8                                  ; 0880
                                      18    A9  DD 00009      PUSHL   24(CBLOCK)                                   ; 0882
                          56          1C    A9  3C 0000C      MOVZWL  28(CBLOCK), BOFF                             ; 0883
                                      1E    AA  95 00010      TSTB    30(RAB)                                      ; 0885
                                            20  12 00013      BNEQ    1$                                           ;
                          52          1E    A9  3C 00015      MOVZWL  30(CBLOCK), R2                               ; 0888
                          52                56  C0 00019      ADDL2   BOFF, R2                                     ;
                          50          0A    A9  3C 0001C      MOVZWL  10(CBLOCK), R0                               ;
                 50       50                09  78 00020      ASHL    #9, R0, R0                                   ;
                 51       52                50  C7 00024      DIVL3   R0, R2, R1                                   ;
                          6E                51  C0 00028      ADDL2   R1, VBN                                      ;
        7E       00       52                01  7A 0002B      EMUL    #1, R2, #0, -(SP)                            ; 0889
        56       56       8E                50  7B 00030      EDIV    R0, (SP)+, BOFF, BOFF                        ;
                                            6E  DD 00035 1$:  PUSHL   VBN                                          ; 0900
                     FDFE CF                01  FB 00037      CALLS   #1, ANSI$CACHEVBN                            ;
                          68                50  D0 0003C      MOVL    R0, (R8)                                     ;
                          03                50  E8 0003F      BLBS    R0, 2$                                       ;
                                          015C  31 00042      BRW     21$                                          ;
                 53       6E          28    A9  C3 00045 2$:  SUBL3   40(CBLOCK), VBN, CACHE_OFFSET                ; 0902
                          51          0A    A9  3C 0004A      MOVZWL  10(CBLOCK), R1                               ; 0903
```

```
                50          53          51  C7 0004E        DIVL3   R1, CACHE_OFFSET, CACHE_SECTION                 ;
                52          50          51  C5 00052        MULL3   R1, CACHE_SECTION, R2                           ; 0904
                52          52          09  78 00056        ASHL    #9, R2, R2                                      ;
                            53          52  C2 0005A        SUBL2   R2, CACHE_OFFSET                                ;
                            57      2C A940  D0 0005D        MOVL    44(CBLOCK)[CACHE_SECTION], CACHE                ; 0905
                      28    AA      24  AA  D0 00062        MOVL    36(RAB), 40(RAB)                                ; 0906
                02          01      1F  AB  8F 00067        CASEB   31(FAB), #1, #2                                 ; 0908
              00AF        005D        000E     0006C 3$:    .WORD   4$-3$,-                                         ;
                                                                     8$-3$,-                                         ;
                                                                     12$-3$                                          ;
                            50 0001860C  8F  D0 00072        MOVL    #99852, R0                                      ; 1013
                                         04 00079            RET                                                     ;
                            52      36  AA  3C 0007A 4$:    MOVZWL  54(RAB), R2                                     ; 0914
                            52          56  C0 0007E        ADDL2   BOFF, R2                                        ;
                50          51          09  78 00081        ASHL    #9, R1, R0                                      ;
                            50          52  D1 00085        CMPL    R2, R0                                          ;
                                        08  19 00088        BLSS    5$                                              ;
                            6E          51  C0 0008A        ADDL2   R1, VBN                                         ; 0917
                                        56  D4 0008D        CLRL    BOFF                                            ; 0918
                                      009A  31 0008F        BRW     14$                                             ; 0919
                      1E    A9      36  AB  B0 00092 5$:    MOVW    54(FAB), 30(CBLOCK)                             ; 0923
                            50      1E  A9  3C 00097        MOVZWL  30(CBLOCK), R0                                  ; 0925
                            50          04  C2 0009B        SUBL2   #4, R0                                          ;
      50      20    AA    10          00  ED 0009E        CMPZV   #0, #16, 32(RAB), R0                            ;
                                        0E  18 000A4        BGEQ    6$                                              ;
                            68 000181A8  8F  D0 000A6        MOVL    #98728, (R8)                                    ; 0928
                      22    AA      20  AA  B0 000AD        MOVW    32(RAB), 34(RAB)                                ; 0929
                                        0C  11 000B2        BRB     7$                                              ;
                            68 00010001  8F  D0 000B4 6$:    MOVL    #65537, (R8)                                    ; 0933
                      22    AA      1E  A9  B0 000BB        MOVW    30(CBLOCK), 34(RAB)                             ; 0934
          28    BA        6647      22  AA  28 000C0 7$:    MOVC3   34(RAB), (BOFF)[CACHE], @40(RAB)                ; 0937
                                        50  11 000C7        BRB     11$                                             ; 0938
                            30        6647  91 000C9 8$:    CMPB    (BOFF)[CACHE], #48                              ; 0946
                                        58  1F 000CD        BLSSU   13$                                             ;
                            39        6647  91 000CF        CMPB    (BOFF)[CACHE], #57                              ;
                                        52  1A 000D3        BGTRU   13$                                             ;
                55          57          56  C1 000D5        ADDL3   BOFF, CACHE, R5                                 ; 0955
                            54          04  D0 000D9        MOVL    #4, R4                                          ;
                            53          0A  D0 000DC        MOVL    #10, R3                                         ;
                              00000000G  EF  16 000DF        JSB     SCAN$NUMERIC                                    ;
                      1E    A9          50  B0 000E5        MOVW    R0, 30(CBLOCK)                                  ;
                            50      1E  A9  3C 000E9        MOVZWL  30(CBLOCK), R0                                  ; 0957
                            50          04  C2 000ED        SUBL2   #4, R0                                          ;
      50      20    AA    10          00  ED 000F0        CMPZV   #0, #16, 32(RAB), R0                            ;
                                        0E  18 000F6        BGEQ    9$                                              ;
                            68 000181A8  8F  D0 000F8        MOVL    #98728, (R8)                                    ; 0960
                      22    AA      20  AA  B0 000FF        MOVW    32(RAB), 34(RAB)                                ; 0961
                                        0B  11 00104        BRB     10$                                             ;
                            68 00010001  8F  D0 00106 9$:    MOVL    #65537, (R8)                                    ; 0965
                      22    AA          50  B0 0010D        MOVW    R0, 34(RAB)                                     ; 0966
          28    BA    04 A647      22  AA  28 00111 10$:   MOVC3   34(RAB), 4(BOFF)[CACHE], @40(RAB)               ; 0969
                                        79  11 00119 11$:   BRB     19$                                             ; 0970
                            30        6647  91 0011B 12$:   CMPB    (BOFF)[CACHE], #48                              ; 0978
                                        06  1F 0011F        BLSSU   13$                                             ;
                            39        6647  91 00121        CMPB    (BOFF)[CACHE], #57                              ;
                                        09  1B 00125        BLEQU   15$                                             ;
                                        56  D4 00127 13$:   CLRL    BOFF                                            ; 0981
```

```
                                 6E          51  C0 00129        ADDL2   R1, VBN                                  ; 0982
                                             68  D4 0012C 14$:   CLRL    (R8)                                     ; 0983
                                             6B  11 0012E        BRB     20$                                      ;
                 55              57          56  C1 00130 15$:   ADDL3   BOFF, CACHE, R5                          ; 0987
                                 54          04  D0 00134        MOVL    #4, R4                                   ;
                                 53          0A  D0 00137        MOVL    #10, R3                                  ;
                                  00000000G  EF  16 0013A        JSB     SCAN$NUMERIC                             ;
                          1E     A9          50  B0 00140        MOVW    R0, 30(CBLOCK)                           ;
                          04     AE      3F  AB  9A 00144        MOVZBL  63(FAB), 4(SP)                           ; 0989
                                 50      1E  A9  3C 00149        MOVZWL  30(CBLOCK), R0                           ;
                                 50      04  AE  C2 0014D        SUBL2   4(SP), R0                                ;
                                 50          04  C2 00151        SUBL2   #4, R0                                   ;
     50       20   AA            10          00  ED 00154        CMPZV   #0, #16, 32(RAB), R0                     ;
                                             13  18 0015A        BGEQ    16$                                      ;
                                 68 000181A8 8F  D0 0015C        MOVL    #98728, (R8)                             ; 0992
                                 50      3F  AA  9A 00163        MOVZBL  63(RAB), R0                              ; 0993
              22   AA       20     AA          50  A3 00167        SUBW3   R0, 32(RAB), 34(RAB)                     ;
                                             0B  11 0016D        BRB     17$                                      ;
                                 68 00010001 8F  D0 0016F 16$:   MOVL    #65537, (R8)                             ; 0997
                          22     AA          50  B0 00176        MOVW    R0, 34(RAB)                              ; 0998
                                         2C  AA  D5 0017A 17$:   TSTL    44(RAB)                                  ; 1001
                                             08  13 0017D        BEQL    18$                                      ;
              2C   BA       04 A647      04  AE  28 0017F        MOVC3   4(SP), 4(BOFF)[CACHE], a44(RAB)          ; 1004
                 50              56      04  AE  C1 00187 18$:   ADDL3   4(SP), BOFF, R0                          ; 1006
              28   BA       04 A047      22  AA  28 0018C        MOVC3   34(RAB), 4(R0)[CACHE], a40(RAB)          ;
                                 68 00010001 8F  D0 00194 19$:   MOVL    #65537, (R8)                             ; 1007
                                 03          68  E8 0019B 20$:   BLBS    (R8), 21$                                ; 1017
                                          FE94  31 0019E        BRW     1$                                       ;
                          18     A9          6E  D0 001A1 21$:   MOVL    VBN, 24(CBLOCK)                          ; 1019
                          1C     A9          56  B0 001A5        MOVW    BOFF, 28(CBLOCK)                         ; 1020
                                 08          68  E9 001A9        BLBC    (R8), 22$                                ; 1022
                                 50 00010001 8F  D0 001AC        MOVL    #65537, R0                               ;
                                                 04 001B3        RET                                              ;
                      00000870   8F          68  D1 001B4 22$:   CMPL    (R8), #2160                              ; 1027
                                             08  12 001BB        BNEQ    23$                                      ;
                                 50 0001827A 8F  D0 001BD        MOVL    #98938, R0                               ;
                                                 04 001C4        RET                                              ;
                                 50 0001C0F4 8F  D0 001C5 23$:   MOVL    #114932, R0                              ;
                                                 04 001CC        RET                                              ; 1029

; Routine Size:  461 bytes,    Routine Base:  CODE + 079F


;   1030  1
;   1031  1
;   1032  1     global routine get_block_length ( ;block_length) : get_block =  ! begin [06]
;   1033  1
;   1034  2         begin
;   1035  2
;   1036  2         linkage
;   1037  2             call_boo = call : global (r1 = 1);
;   1038  2
;   1039  2         external routine
;   1040  2             boo$qio : call_boo addressing_mode (long_relative);
;   1041  2
;   1042  2         global register
;   1043  2             r1 = 1;
```

```
;   1044  2
;   1045  2         external register
;   1046  2             cblock = 9 : ref block [, byte],                    !Base of common area
;   1047  2             fab = 11 : ref $fab_decl;                           !Base of FAB
;   1048  2
;   1049  2         bind
;   1050  2             status = fab [fab$l_stv];                           !Return value from boo$qio
;   1051  2
;   1052  2         kb_check ();
;   1053  2         status = boo$qio (.cblock [ctl$l_cache3], .cblock [ctl$w_cache_size]*512, 0, io$_readlblk, 0, .cblock [ctl$l_rpb
;   1054  2         block_length = .r1;
;   1055  2         .status
;   1056  1         end;                                                    ! end [06]



                                                  05F8 00000          .ENTRY  GET_BLOCK_LENGTH, Save R3,R4,R5,R6,R7,R8,-   ; 1032
                                                                              R10                                          ;
                                  00000000G  EF   16 00002          JSB     KB_CHECK                                     ; 1052
                                        38   A9   DD 00008          PUSHL   56(CBLOCK)                                   ; 1053
                                 7E          21   7D 0000B          MOVQ    #33, -(SP)                                   ;
                                             7E   D4 0000E          CLRL    -(SP)                                        ;
                                 50      0A   A9   3C 00010          MOVZWL  10(CBLOCK), R0                               ;
                     7E          50           09   78 00014          ASHL    #9, R0, -(SP)                                ;
                                        34   A9   DD 00018          PUSHL   52(CBLOCK)                                   ;
                        00000000G  EF           06   FB 0001B          CALLS   #6, BOO$QIO                                  ;
                              0C   AB           50   D0 00022          MOVL    R0, 12(FAB)                                  ;
                                   52           51   D0 00026          MOVL    R1, BLOCK_LENGTH                             ; 1054
                                   50      0C   AB   D0 00029          MOVL    12(FAB), R0                                  ; 1056
                                                    04 0002D          RET                                                  ;

; Routine Size:  46 bytes,    Routine Base:  CODE + 096C


;   1057  1
;   1058  1     end
;   1059  0     eludom




;                                   PSECT SUMMARY
;
;       Name                    Bytes                          Attributes
;
;  DATA                            24  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          2458  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...										
$ASCID	Macro	Lib01	343										
$FAB_DECL	Macro	Lib03	207	605	772	870	1047						
$FAO	Macro	Lib03	343										
$RAB_DECL	Macro	Lib03	773	869									
A	Local	330	332=	333=	343a	345.							
ADR	Local	234	236=	238.	243a.								
ANSI$CACHEVBN	GlobRout	624	802c	900c									
ANSI$CLOSE	GlobRout	571	97f	222a									
ANSI$GET	GlobRout	831	99f	220a									
ANSI$OPEN	GlobRout	138	96f										
ANSI$READ	GlobRout	733	98f	221a									
BBLOCK	Structur	Lib02	322	323	327	328	329	330					
BLANK	Local	326	337=	338.									
BLOCK_LENGTH	Local	307	436=	470.	471.								
	RoutForm	1032	1054=										
BOFF	Local	876	883=	888.	889=	889.	914.	918=	937.	946.	949=	955.	969.	978.
			981=	987.	1003.	1006.	1020.						
BOO$QIO	ExtRout	212	248c	254c	261c	277c	280c	283c	352c	359c	363c	367c	371c	475c
			493c	497c	500c	502c	504c	514c	518c	522c	535c	548c	608e	614c
			661e	676c	715c	776e	873e	1040e	1053c					
CACHE	Local	688	696=	699=	702=	706=	709.	714a=	715.					
	Local	779	809=	810a.										
	Local	896	905=	937a.	946a.	955aa	969a.	978a.	987aa	1003a.	1006a.			
CACHE_OFFSET	Local	799	806=	807.	808=	808.	810.							
	Local	897	902=	903.	904=	904.								
CACHE_SECTION	Local	800	807=	808.	809.									
	Local	898	903=	904.	905.									
CALL_BOO	Linkage	1037	1040										
CALL_RMS	Linkage	Lib02	96	97	98	99	138	571	624	733	831			
CBLOCK	ExtReg	208	220a=	221a=	222a=	236a.	237a.	240a.	248a.	254a.	261a.	277a.	280a.	283a.
			322a.	352a.	359a.	363a.	367a.	371a.	431a=	432a=	433a=	434a=	440aa	442a.
			449a.	450a=	450a.	451a=	452a=	455a.	462a.	463a=	463a.	464a=	465a=	468a=
			468a.	469a=	469a.	471a=	474a=	474a.	475a.	493a.	497a.	500a.	502a.	504a.
			514a.	518a.	522a.	535a.	548a.							
	ExtReg	604	614a.											
	ExtReg	658	666a.	668a.	673a.	676a.	680a=	680a.	681a=	681a.	684a.	692a.	696a.	699a.
			702a.	706a.	707a=	707a.	708a=	708a.	709a=	710a=	710a.	714a.	715a.	716a.
			720a=	720a.	721a=	721a.	722a.							
	ExtReg	771	787a.	806a.	807a.	808a.	809.	817a=						
	ExtReg	868	882a.	883a.	888a.	889a.	902a.	903a.	904a.	905.	914a.	917a.	923a=	925a.
			934a.	950a.	955a=	957a.	966a.	982a.	987a=	989a.	998a.	1019a=	1020a=	
	ExtReg	1046	1053a.											
CTL$L_CACHE1	Macro	Lib02	450	451	696	706	707	809	905					
CTL$L_CACHE2	Macro	Lib02	463	464	699	707	708							
CTL$L_CACHE3	Macro	Lib02	449	450	462	463	702	708	709	1053				
CTL$L_CLOSE	Macro	Lib02	222											
CTL$L_EOFVBN	Macro	Lib02	431	469	474	666	721							
CTL$L_FILE_PTR	Macro	Lib02	236	237	240									
CTL$L_GET	Macro	Lib02	220											
CTL$L_HVBN	Macro	Lib02	433	452	465	668	681	684	692	721	722			
CTL$L_LVBN	Macro	Lib02	433	668	681	684	692	710	806	902				
CTL$L_NBP	Macro	Lib02	787	817										
CTL$L_PTABLE	Macro	Lib02	322											
CTL$L_READ	Macro	Lib02	221											
CTL$L_RPB	Macro	Lib02	248	254	261	277	280	283	352	359	363	367	371	475

Symbol	Type	Defined	Referenced ...											
			493	497	500	502	504	514	518	522	535	548	614	676
			716	1053										
CTL$L_TAPEPOS	Macro	Lib02	434	442	455	468	673	680	681	720				
CTL$L_VBN	Macro	Lib02	882	1019										
CTL$W_BYTE	Macro	Lib02	883	1020										
CTL$W_CACHE_SIZE	Macro	Lib02	440	680	692	710	714	715	720	722	807	808	888	889
			903	904	914	917	950	982	1053					
CTL$W_FILE_LEN	Macro	Lib02	236	237	240									
CTL$W_OFFBYTE	Macro	Lib02	432	471										
CTL$W_REC_SIZE	Macro	Lib02	888	889	923	925	934	955	957	966	987	989	998	
DEV	Local	327	334=	335=	343a									
DSC$A_POINTER	Macro	Lib03	333	334	336	339								
DSC$C_S_BLN	Literal	Lib03	327	328	329	330								
DSC$W_LENGTH	Macro	Lib03	332	335	338	341	342	345						
END_OF_FILE	Literal	194	268	526	539	545	549	555						
END_OF_PASS1	Literal	193	551											
FAB	ExtReg	207	215aa	251a.	317a.	323a.	384a=	385a=	386a=	387a=	397a=	398a=	403a=	404a=
			415a=	418a=	421a=									
	ExtReg	605	611aa											
	ExtReg	772												
	ExtReg	870	908a.	923a.	989a.	998a.	1003a.	1006a.						
	ExtReg	1047	1050aa											
FAB$B_FSZ	Macro	Lib03	386	989	993	998	1003	1006						
FAB$B_RAT	Macro	Lib03	384	415	418	421								
FAB$B_RFM	Macro	Lib03	385	397	403	908								
FAB$C_BLN	Literal	Lib03	207	605	772	870	1047							
FAB$C_FIX	Literal	Lib03	397	908	911									
FAB$C_VAR	Literal	Lib03	403	943										
FAB$C_VFC	Literal	Lib03	908	975										
FAB$L_NAM	Macro	Lib03	317	323										
FAB$L_STV	Macro	Lib03	215	611	1050									
FAB$M_CR	Literal	Lib03	418											
FAB$M_FTN	Literal	Lib03	415											
FAB$V_RWO	Macro	Lib03	251											
FAB$W_MRS	Macro	Lib03	387	398	404	914	923							
FALSE	Literal	192	245	268	299	301	539	549						
FAT	Bind	382	384.	385.	386.	387.								
FAT$B_RATTRIB	Macro	Lib03	384											
FAT$B_VFCSIZE	Macro	Lib03	386											
FAT$C_LENGTH	Literal	Lib03	382											
FAT$V_RTYPE	Macro	Lib03	385											
FAT$W_MAXREC	Macro	Lib03	387											
FILE	Local	328	336=	338=	343a									
FILENAME	Local	204	242=	243=	274.	285.	295.							
FIRST1	Local	203	246=	488.	510.	510=								
FLAG	Local	199	245=	268=	526=	539=	545.	549=	551.	551=	555=			
GET_BLOCK	Linkage	89	100	211	1032									
GET_BLOCK_LENGTH	GlobRout	1032	100f	436c										
HD1$T_FILEID	Macro	Lib03	274	336	337	338								
HD1$T_GENNO	Macro	Lib03	272											
HD1$T_GENVER	Macro	Lib03	273											
HD2$B_FORMCNTRL	Macro	Lib03	411											
HD2$B_RECFORMAT	Macro	Lib03	392											
HD2$T_BLOCKLEN	Macro	Lib03	426											
HD2$T_RECATR1	Macro	Lib03	377	382										

Symbol	Type	Defined	Referenced ...										
HD2$T_RECLEN	Macro	Lib03	398	404									
HD4$B_FILEID_EXT_SIZE	Macro	Lib03	291	294	296	341							
HD4$T_FILEID_EXT	Macro	Lib03	297	339									
HDR1	Local	200	261a	270.	272a	273a	274.	275.	336a	337.	338a	348.	484.	488.
			510.										
HDR2	Local	201	277a	377.	382a	392.	398a	404a	411.	426a			
HDR4	Local	202	280a	283a	291.	294.	296.	297.	339a	340.	341.	350.	357.	491.
			512.										
IO$_DRVCLR	Literal	Lib03	248										
IO$_READLBLK	Literal	Lib03	261	277	280	283	715	1053					
IO$_REWIND	Literal	Lib03	254	548									
IO$_SKIPFILE	Literal	Lib03	352	359	363	367	371	475	497	514	518	522	535	614
IO$_SKIPRECORD	Literal	Lib03	493	500	502	504	676							
JSB	Linkage	87	113											
JSB_CBLOCK	Linkage	Lib02	114											
JSB_SCAN	Linkage	88	112											
KB_CHECK	ExtRout	113	260c	613c	675c	690c	1052c							
LEN	Local	231	238=	243.										
LONGFILE	Local	329	339=	341=	342=	343a								
NAM	Bind	323	332.	333.	345=									
NAM$B_RSL	Macro	Lib03	345											
NAM$B_RSS	Macro	Lib03	332											
NAM$L_RSA	Macro	Lib03	333											
PHYNAM	Bind	322	334.	335.										
R1	GlobReg	1043	1054.											
RAB	ExtReg	773	785aa	787a.	788a.	789a=	789a.	790a=	791a=	792a=	812a=	812a.		
	ExtReg	869	880aa	885a.	906a=	906a.	914a.	925a.	929a=	929a.	934a=	937a.	937a=	957a.
			961a=	961a.	966a=	969a.	969a=	989a.	993a=	993a.	998a=	1001a.	1004a=	1006a.
			1006a=											
RAB$B_RAC	Macro	Lib03	885											
RAB$C_BLN	Literal	Lib03	773	869										
RAB$C_SEQ	Literal	Lib03	885											
RAB$L_BKT	Macro	Lib03	787											
RAB$L_RBF	Macro	Lib03	789	906	937	969	1006							
RAB$L_RFA0	Macro	Lib03	791											
RAB$L_RHB	Macro	Lib03	1001	1004										
RAB$L_STV	Macro	Lib03	785	880										
RAB$L_UBF	Macro	Lib03	789	906										
RAB$W_RFA4	Macro	Lib03	792											
RAB$W_RSZ	Macro	Lib03	790	812	929	934	937	961	966	969	993	998	1006	
RAB$W_USZ	Macro	Lib03	788	925	929	957	961	989	993					
RMS$ALLOC_CACHE	ExtRout	114	426c											
RMS$_CCF	Literal	Lib03	616											
RMS$_DME	Literal	Lib03	429											
RMS$_EOF	Literal	Lib03	1027											
RMS$_FNF	Literal	Lib03	506	551										
RMS$_NORMAL	Literal	Lib03	479	618	793	824	933	938	965	970	997	1007	1024	
RMS$_ORG	Literal	Lib03	408	1013										
RMS$_RER	Literal	Lib03	278	281	290	353	360	365	369	373	477	494	498	501
			503	505	515	520	524	537	560	824	1027			
RMS$_RTB	Literal	Lib03	928	960	992									
SCAN$NUMERIC	ExtRout	112	241c	272c	398c	404c	426c	955c	987c					
SIZE	Bind	440	452.	455.	465.	468.	469.							
SS$_DATAOVERUN	Literal	Lib03	533											
SS$_ENDOFFILE	Literal	Lib03	287	542	666	824	1027							

Symbol	Type	Defined	Referenced ...										
SS$_NORMAL	Literal	Lib03	266	565	727								
STATUS	Bind	215	248=	249.	254=	255.	261=	263.	277=	278.	280=	281.	283=	287.
			289.	352=	353.	359=	360.	363=	365.	367=	369.	371=	373.	426=
			475=	477.	493=	494.	497=	498.	500=	501.	502=	503.	504=	505.
			514=	515.	518=	520.	522=	524.	535=	537.	548=			
	Bind	611	614=	616.										
	Local	664	676=	678.	715=	718.								
	Bind	785	793=	802=	804.	819.	821.	824.						
	Bind	880	900=	919=	928=	933=	938=	951=	960=	965=	970=	983=	992=	997=
			1007=	1017.	1022.	1027.								
	Bind	1050	1053=	1055.										
SYS$FAO	ExtRout	343	343c											
TAPEVERSION	Local	198	272=	275.	343.									
TMP	Local	447	449=	451.										
	Local	460	462=	464.										
TRUE	Literal	191	286	288	292	298	526	551	555					
USER_ADDRESS	Local	781	789=	810a=	811=	811.								
USER_SIZE	Local	780	788=	795.	810.	811.	812.	813=	813.					
VBN	RoutForm	624	666.	668.	673.	684.								
	Local	782	787=	791.	802.	806.	814=	814.	817.					
	Local	877	882=	888=	888.	900.	902.	917=	917.	950=	950.	982=	982.	1019.
VERADR	Local	233	237=	238.	239=	239.	240.	241.						
VERLEN	Local	232	240=	241.										
VERSION	Local	197	241=	275.	285.									

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]ANSI.B32;1
7	Module ANSI	
80	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;8
82	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;8
84	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
1059	Eludom ANSI	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
a	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

;   Library Statistics
;

```
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;8
;                                            940         1         0          96         00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;8
;                                            675        23         3          47         00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2             20450        65         0        1101         00:00.8



;                                   COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:ANSI.B32

; Size:          2458 code + 24 data bytes
; Run Time:         00:08.9
; Elapsed Time:     00:11.4
; Lines/CPU Min:     7179
; Lexemes/CPU-Min: 88637
; Memory Used:  429 pages
; Compilation Complete
```

Table of contents


```
    (1)       55        Libraries, Included Macros, Equated Symbols
    (1)       62        Data Psect Declarations
    (1)       76        Work Psect Declarations
    (1)       92        Return Error Message
    (1)      100        DSX$PPSCB                                    ;G:2
```

```
                     0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element APBOOT.MAR
                     0000     2 ; *5    12-MAY-1986 09:59:21 BARANSKI "Cleaning up Object Libraries."
                     0000     3 ; *4     7-MAY-1986 13:09:03 BARANSKI "Documentation Check."
                     0000     4 ; *3    25-APR-1986 16:42:47 SHELHAMER "<no reason given>"
                     0000     5 ; *2    25-APR-1986 13:34:16 SHELHAMER "<no reason given>"
                     0000     6 ; *1     6-FEB-1986 15:13:27 DS ""
                     0000     7 ; DEC/CMS REPLACEMENT HISTORY, Element APBOOT.MAR
                     0000     8         .Title  APBOOT       Diagnostic Supervisor AP Boot Command Module
                     0000     9         .Ident  "7.0-0"                                                    ;G:4
                     0000    10         .NoShow Conditionals
                     0000    11
                     0000    12 ;++
                     0000    13 ; COPYRIGHT (C) 1984, 1985
                     0000    14 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                     0000    15 ;
                     0000    16 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                     0000    17 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                     0000    18 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                     0000    19 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                     0000    20 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                     0000    21 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                     0000    22 ; REMAIN IN DEC.
                     0000    23 ;
                     0000    24 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                     0000    25 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                     0000    26 ; CORPORATION.
                     0000    27 ;
                     0000    28 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                     0000    29 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                     0000    30 ;--
```

```
                        0000     32 ;++
                        0000     33 ; Facility:
                        0000     34 ;
                        0000     35 ;       VAX Diagnostic Supervisor
                        0000     36 ;
                        0000     37 ; Abstract:
                        0000     38 ;
                        0000     39 ;       This module provides support for booting an attached processor on the BI.
                        0000     40 ;       For all supervisors except Scorpio, this module issues an error message and
                        0000     41 ;       returns.
                        0000     42 ;
                        0000     43 ; Environment:
                        0000     44 ;
                        0000     45 ; Author:
                        0000     46 ;
                        0000     47 ;       James Shelhamer         20-Nov-84       Version 7.
                        0000     48 ;
                        0000     49 ;
                        0000     50 ; Modifications:
                        0000     51 ;
                        0000     52 ;       None
                        0000     53 ;--
```

```
                        0000     55          .SubTitle       Libraries, Included Macros, Equated Symbols
                        0000     56
                        0000     57          .Library        /Sys$Library:Lib/       ; Use Lib last
                        0000     58          .Library        /$CRD/                  ; Use CRD.Mar third
                        0000     59          .Library        /$DS/                   ; Use DS.Mar second
                        0000     60          .Library        /$Diag/                 ; Use Diag.Mar first
```

```
                                    0000        .SubTitle       Data Psect Declarations
                                00000000        .Psect          Data, NoExe, Shr, NoWrt, Long
                                    0000                                                                    ;G:3
                                    0000        $DS_DSDEF                                                   ;G:3
                                    0000  ; The following return codes are for use with APR$CHECK_FOR_PROMPT. They
                                    0000  ; indicate what was received from the AP.
                        00000001    0000  AP$GK_PROMPT == 1                                  ; prompt only received
                        00000002    0000  AP$GK_PROMPT_WITH_STATUS == 2                      ; prompt and status revd
                        00000003    0000  AP$GK_NO_PROMPT == 3                               ; no prompt received,
                                    0000
                                    0000  NOT_AN_AP_SYSTEM:
6E 6E 61 63 20 75 6F 59 20 3F 3F 00'  0000        .ASCIC  "?? You cannot boot an attached processor on this system.!/"
61 20 6E 61 20 74 6F 6F 62 20 74 6F   000C
63 6F 72 70 20 64 65 68 63 61 74 74   0018
69 68 74 20 6E 6F 20 72 6F 73 73 65   0024
   2F 21 2E 6D 65 74 73 79 73 20 73   0030
                                 3A   0000
                                    003B
```

```
                        003B     76          .Subtitle       Work Psect Declarations
                    00000000     77          .Psect          Work, NoExe, NoShr, Wrt, Long
                        0000     78
                        0000     79 ;
                        0000     80 ; Request codes. Issued by the AP to the PP. They are the first longword
                        0000     81 ; in AP$GR_SHARED_DATA, the data block associated with requests.
                        0000     82 ;
                    00000001    0000     83 AP$GK_REQ_CONSRX == 1                           ; Request for console floppy I/O
                        0000     84 AP$GW_DATA::
                        0000     85          .WORD   0
                        0002     86 AP$GW_AP_NODES::
                        0002     87          .WORD   0
                        0004     88 AP$GR_SHARED_DATA::                             ; CPU-shared area. First
                    00000008    0004     89          .BLKL   1                      ; longword used to indicate
                        0008     90                                                 ; request type or return code
```

```
                            0008    92          .Subtitle       Return Error Message
                        00000000    93          .Psect          Code, Exe, Shr, NoWrt, Long
                            0000    94
                            0000    95 CMK$APBOOT::
                            0000    96          $DS_PRINTF_S    NOT_AN_AP_SYSTEM        ; Issue error message
                                                        PUSHAB  NOT_AN_AP_SYSTEM
                                                        CALLS   #$$N, @#DS$PRINTF
                            000D    97          REI                                     ; And return.
                            000E    98
```

```
                                000E          .Subtitle       DSX$PPSCB                                                    ;G:2
                                000E    ;++                                                                                ;G:2
                                000E    ;                                                                                  ;G:2
                                000E    ; Functional Description:                                                          ;G:2
                                000E    ;       Returns an error - there is no PP SCB on non-8200 systems                  ;G:2
                                000E    ;                                                                                  ;G:2
                                000E    ;--                                                                                ;G:2
                            0000  000E          .ENTRY  DSX$PPSCB,^M<>                                                     ;G:2
        50   006600B1 8F     D0   0010          MOVL    #DS$_NOSUPPORT,R0                ; ERROR                           ;G:2
                              04   0017          RET                                                                        ;G:2
                                   0018                                                                                     ;G:2
                                   0018          .End
```

```
$$N                             = 00000001    D
AP$GK_NO_PROMPT                 = 00000003  G D
AP$GK_PROMPT                    = 00000001  G D
AP$GK_PROMPT_WITH_STATUS        = 00000002  G D
AP$GK_REQ_CONSRX                = 00000001  G D
AP$GR_SHARED_DATA                 00000004 RG D  02
AP$GW_AP_NODES                    00000002 RG D  02
AP$GW_DATA                        00000000 RG D  02
BIT...                          = 00660160    D
CMK$APBOOT                        00000000 RG D  03
DS$K_ERROR                      = 00000002    D
DS$K_NORMAL                     = 00000001    D
DS$K_SEVERE                     = 00000004    D
DS$K_SUBSYS                     = 00000066    D
DS$K_WARNING                    = 00000000    D
DS$PRINTF                         ********  X   03
DS$_AP_NORMAL_BREAK             = 00660150    D
DS$_ARITH                       = 006600D0    D
DS$_ASBE                        = 00660118    D
DS$_BADLINK                     = 006600F0    D
DS$_BADTYPE                     = 006600E8    D
DS$_BIIC                        = 00660120    D
DS$_CHME                        = 006600A8    D
DS$_CHMK                        = 006600E0    D
DS$_DEVNAME                     = 00660108    D
DS$_ERROR                       = 00660002    D
DS$_FHWE                        = 00660068    D
DS$_FRAGBUF                     = 00660080    D
DS$_ICBUSY                      = 006600C8    D
DS$_ICERR                       = 006600C0    D
DS$_IHWE                        = 00660060    D
DS$_ILLCHAR                     = 00660018    D
DS$_ILLPAGCNT                   = 00660078    D
DS$_ILLUNIT                     = 00660100    D
DS$_INIT_FAIL                   = 00660140    D
DS$_INSFMEM                     = 00660050    D
DS$_INVCPU                      = 00660130    D
DS$_IPL2HI                      = 006600B8    D
DS$_IVADDR                      = 00660040    D
DS$_IVVECT                      = 00660038    D
DS$_KRNLSTK                     = 00660090    D
DS$_LOGIC                       = 00660070    D
DS$_MCHK                        = 00660088    D
DS$_MEM_ALLOC_ERR               = 00660138    D
DS$_MMOFF                       = 00660058    D
DS$_NEEDUNIT                    = 006600F8    D
DS$_NODE                        = 00660128    D
DS$_NOPCS                       = 00660110    D
DS$_NORMAL                      = 00660001    D
DS$_NOSUPPORT                   = 006600B1    D
DS$_NOTALLOWMP                  = 00660148    D
DS$_NOTDON                      = 00660030    D
DS$_NOTIMP                      = 006600B0    D
DS$_NULLSTR                     = 00660010    D
DS$_OVERFLOW                    = 00660008    D
DS$_POWER                       = 00660098    D
DS$_PROGERR                     = 00660020    D
DS$_SEVERE                      = 00660004    D
DS$_TRANSL                      = 006600A0    D
DS$_TRUNCATE                    = 00660028    D
DS$_UNEXPINT                    = 006600D8    D
DS$_UNSUPPDEV                   = 00660158    D
DS$_VASFULL                     = 00660048    D
DS$_WARNING                     = 00660000    D
DSX$PPSCB                         0000000E RG D  03
NOT_AN_AP_SYSTEM                  00000000 R  D  01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
DATA	0000003B (59.)	01 (1.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
WORK	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	00000018 (24.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000001	96 (1)	#-96 (1)
AP$GK_NO_PROMPT	=00000003	70 (1)	
AP$GK_PROMPT	=00000001	68 (1)	
AP$GK_PROMPT_WITH_STATUS	=00000002	69 (1)	
AP$GK_REQ_CONSRX	=00000001	83 (1)	
AP$GR_SHARED_DATA	00000004-R	88 (1)	
AP$GW_AP_NODES	00000002-R	86 (1)	
AP$GW_DATA	00000000-R	84 (1)	
BIT...	=00660160	65 (1)	65 (1)
CMK$APBOOT	00000000-R	95 (1)	
DS$K_ERROR	=00000002	65 (1)	
DS$K_NORMAL	=00000001	65 (1)	
DS$K_SEVERE	=00000004	65 (1)	
DS$K_SUBSYS	=00000066	65 (1)	65 (1)
DS$K_WARNING	=00000000	65 (1)	
DS$PRINTF	00000000-XR		96 (1)
DS$_AP_NORMAL_BREAK	=00660150	65 (1)	
DS$_ARITH	=006600D0	65 (1)	
DS$_ASBE	=00660118	65 (1)	
DS$_BADLINK	=006600F0	65 (1)	
DS$_BADTYPE	=006600E8	65 (1)	
DS$_BIIC	=00660120	65 (1)	
DS$_CHME	=006600A8	65 (1)	
DS$_CHMK	=006600E0	65 (1)	
DS$_DEVNAME	=00660108	65 (1)	
DS$_ERROR	=00660002	65 (1)	
DS$_FHWE	=00660068	65 (1)	
DS$_FRAGBUF	=00660080	65 (1)	
DS$_ICBUSY	=006600C8	65 (1)	
DS$_ICERR	=006600C0	65 (1)	
DS$_IHWE	=00660060	65 (1)	
DS$_ILLCHAR	=00660018	65 (1)	
DS$_ILLPAGCNT	=00660078	65 (1)	
DS$_ILLUNIT	=00660100	65 (1)	
DS$_INIT_FAIL	=00660140	65 (1)	
DS$_INSFMEM	=00660050	65 (1)	
DS$_INVCPU	=00660130	65 (1)	
DS$_IPL2HI	=006600B8	65 (1)	
DS$_IVADDR	=00660040	65 (1)	
DS$_IVVECT	=00660038	65 (1)	
DS$_KRNLSTK	=00660090	65 (1)	
DS$_LOGIC	=00660070	65 (1)	
DS$_MCHK	=00660088	65 (1)	
DS$_MEM_ALLOC_ERR	=00660138	65 (1)	
DS$_MMOFF	=00660058	65 (1)	
DS$_NEEDUNIT	=006600F8	65 (1)	
DS$_NODE	=00660128	65 (1)	
DS$_NOPCS	=00660110	65 (1)	
DS$_NORMAL	=00660001	65 (1)	
DS$_NOSUPPORT	=006600B1	65 (1)	#-108 (1)

```
DS$_NOTALLOWMP                  =00660148     65     (1)
DS$_NOTDON                      =00660030     65     (1)
DS$_NOTIMP                      =006600B0     65     (1)       65     (1)
DS$_NULLSTR                     =00660010     65     (1)
DS$_OVERFLOW                    =00660008     65     (1)
DS$_POWER                       =00660098     65     (1)
DS$_PROGERR                     =00660020     65     (1)
DS$_SEVERE                      =00660004     65     (1)
DS$_TRANSL                      =006600A0     65     (1)
DS$_TRUNCATE                    =00660028     65     (1)
DS$_UNEXPINT                    =006600D8     65     (1)
DS$_UNSUPPDEV                   =00660158     65     (1)
DS$_VASFULL                     =00660048     65     (1)
DS$_WARNING                     =00660000     65     (1)       65     (1)
DSX$PPSCB                        0000000E-R   107    (1)
NOT_AN_AP_SYSTEM                 00000000-R   72     (1)       96     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_PRINT_S	1	96 (1)	96 (1)
$DEF	1	65 (1)	
$DS_DSDEF	3	65 (1)	65 (1)
$DS_PRINTF_S	1	96 (1)	96 (1)
$EQU	1	65 (1)	65 (1)
$EQULS1	1	65 (1)	65 (1)
$EQULST	1	65 (1)	65 (1)
$GBLINI	2	65 (1)	65 (1)
$VIELD1	1	65 (1)	

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
CALLS           00FB      96      (1)
MOVL            00D0      108     (1)
PUSHAB          009F      96      (1)
REI             0002      97      (1)
RET             0004      109     (1)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ASCIC          73      (1)
.BLKL           89      (1)
.END            111     (1)
.ENDC           65      (1)     96      (1)
.ENDM           65      (1)     96      (1)
.ENDR           65      (1)     96      (1)
.ENTRY          107     (1)
.IDENT          9       (1)
.IF             65      (1)     96      (1)
.IFF            65      (1)
.IIF            65      (1)
.IRP            65      (1)     96      (1)
.LIBRARY        57      (1)     58      (1)     59      (1)     60      (1)
.MACRO          65      (1)
.MDELETE        65      (1)
.NOSHOW         10      (1)
.PAGE           31      (1)     54      (1)     61      (1)     75      (1)     91      (1)     99      (1)
.PSECT          63      (1)     77      (1)     93      (1)
.SUBTITLE       100     (1)     55      (1)     62      (1)     76      (1)     92      (1)
.TITLE          8       (1)
.WORD           85      (1)     87      (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
R0	1	#-108 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.03	00:00:00.19
Command processing	895	00:00:00.22	00:00:00.80
Pass 1	291	00:00:00.53	00:00:01.63
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	9	00:00:00.11	00:00:00.23
Symbol table output	0	00:00:00.02	00:00:00.02
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	3	00:00:00.07	00:00:00.13
Assembler run totals	1233	00:00:01.00	00:00:03.02

The working set limit was 2512 pages.
23823 bytes (47 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 66 non-local and 0 local symbols.
111 source lines were read in Pass 1, producing 22 object records in Pass 2.
21 pages of virtual memory were used to define 8 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
DISK$DS:[DS.LOCAL.SCRATCH]CRD.MLB;1	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	2
TOTALS (all libraries)	5

130 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:APBOOT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 91 DECLARATIONS
(2) 120 APT SUPPORT ROUTINES

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element APT.MAR
0000     2 ; *8     6-OCT-1987 13:53:34 BARANSKI ""
0000     3 ; *7    22-OCT-1986 10:17:50 MARTIN "FIX DS$GT_PROMPT"
0000     4 ; *6    23-JUN-1986 14:11:02 MARTIN "DONE"
0000     5 ; *5    23-JUN-1986 09:37:21 MARTIN "DONE"
0000     6 ; *4    23-JUN-1986 09:34:12 MARTIN "DONE"
0000     7 ; *3    12-MAY-1986 10:07:16 BARANSKI "Cleaning up Object Libraries."
0000     8 ; *2     7-MAY-1986 13:26:39 BARANSKI "Documentation Check."
0000     9 ; *1     6-FEB-1986 15:13:31 DS ""
0000    10 ; DEC/CMS REPLACEMENT HISTORY, Element APT.MAR
0000    11         .TITLE  APT     *** APT interface to APT control
0000    12         .IDENT  "10"                                                ;G:11
0000    13         .LIST   MEB
0000    14         .NLIST  CND
0000    15 ;
0000    16 ; COPYRIGHT (C) 1977, 1983, 1986                                    ;G:6
0000    17 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    18 ;
0000    19 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    20 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    21 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    22 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    23 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    24 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    25 ; REMAIN IN DEC.
0000    26 ;
0000    27 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    28 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    29 ; CORPORATION.
0000    30 ;
0000    31 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    32 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    33
0000    34 ;++
0000    35 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    36 ;
0000    37 ; ABSTRACT:
0000    38 ;
0000    39 ; ENVIRONMENT:
0000    40 ;
0000    41 ; AUTHOR:       N. HOWGATE      03-DEC-77       VERSION 01.
0000    42 ;                                                                   ;G:2
0000    43 ; MODIFIED BY:
0000    44 ;                                                                   ;G:2
0000    45 ;                N. HOWGATE      22-JUN-78       VERSION 02(ESSAA-4.03).
0000    46 ;       01       CORRECT DUMP-MODE PROBLEM
0000    47 ;                                                                   ;G:2
0000    48 ;                N. HOWGATE      07-DEC-78       VERSION 03 (ESSAA-5.01)
0000    49 ;       02       EMULATE CHARACTER STRING INSTRUCTION
0000    50 ;                                                                   ;G:2
0000    51 ;                R. RIGGS        12-JUL-79
0000    52 ;       03       ADD CODE TO EXECUTE APT PTEXT DECODER
0000    53 ;                                                                   ;G:2
0000    54 ;                Roger Riggs, 28-May-1980, Version 5.4
0000    55 ;       04       Added code to check that on APT PTEXT command
0000    56 ;                that there is at least one command to process.
0000    57 ;
```

```
                                0000     58 ;       05      - Jack Stansbury, 22-Oct-1981, Version 6.-
                                0000     59 ;               Changed SEP Psect to CODE.
                                0000     60 ;               Fixed truncation errors. Change to lower case on message.
                                0000     61 ;               Also added .LIBRARY statements for $DS and $DIAG.
                                0000     62 ;
                                0000     63 ;       06      - Jack Stansbury, 12-Oct-1981, Version 6.-
                                0000     64 ;               Changed the call to DS$PARSE to DSX$SUPERPARSE. See the CLI
                                0000     65 ;               module for an explanation of SUPERPARSE. It returns with the
                                0000     66 ;               low bit clear in R0 if anything went wrong.
                                0000     67 ;
                                0000     68 ;       07      - Jack Stansbury, 12-Oct-1981, Version 6.-
                                0000     69 ;               Took out references to HALTD and HALTI. Made them just HALT.
                                0000     70 ;
                                0000     71 ;       08      - Jack Stansbury, 3-Feb-1982, Version 6.6
                                0000     72 ;               Added a .LIBRARY statement for LIB. Note: This .LIBRARY
                                0000     73 ;               statement must be the first one since the libraries are
                                0000     74 ;               searched in LIFC (Last-In, First-Checked) order.
                                0000     75 ;
                                0000     76 ;       09      Bob Bergazzi    2-Feb-1983      Version 6.11
                                0000     77 ;               Added code to COM_PTEXT that clears out DS$GL_CLIBASE.
                                0000     78 ;               We should clear these flags out on every command,
                                0000     79 ;               otherwise subsequent commands may be affected by previous
                                0000     80 ;               commands. This coincides with a change to APT which allows
                                0000     81 ;               any commands as PTEXT. The bug prompting this was
                                0000     82 ;               SET EVENT 1, CLEAR EVENT 1, which caused a hung load.
                                0000     83 ;
                                0000     84 ;       10      Bob Bergazzi    11-Apr-1983     Version 6.11
                                0000     85 ;               Fixed the previous edit.
                                0000     86 ;                                                                       ;G:6
                                0000     87 ;       11      STEVE MEIER     23-jun-1986     Version 10.2           ;G:6
                                0000     88 ;               changed literal to symbolic string ( "DS> " ).         ;G:6
                                0000     89 ;--
```

```
                                0000     91          .SBTTL  DECLARATIONS
                                0000     92 ;
                                0000     93 ; INCLUDE FILES:
                                0000     94 ;
                                0000     95          .LIBRARY        /SYS$LIBRARY:LIB/                       ;               [08]
                                0000     96          .LIBRARY        /$DS/                                   ;               [05]
                                0000     97          .LIBRARY        /$DIAG/                                 ;               [05]
                                0000     98
                                0000     99 ;
                                0000    100 ; MACROS:
                                0000    101 ;
                                0000    102
                                0000    103 ;
                                0000    104 ; EQUATED SYMBOLS:
                                0000    105 ;
                                0000    106 $DS_DSDEF
                                0000    107 $DS_DSADEF                                          ; APT DEFINITIONS
                                0000    108 $DS_HDRDEF                                          ; HEADER DEFINITIONS
                                0000    109 CLIDEF                                              ; CLI DEFINITIONS
                                0000    110 DSFDEF
                                0000    111
                                0000    112 ;
                                0000    113 ; OWN STORAGE:
                                0000    114 ;
                                0000    115          .SAVE
                           00000000     116          .PSECT  Data, NoExe, Shr, NoWrt, Long                   ;               [05]
                                0000    117          MODNAM  APT
                     54 50 41 00' 0000      $MODULE:        .ASCIC  \APT\
                                  03 0000
64 65 76 72 65 73 65 52 20 3F 3F 00' 0004    118 T_PRGERR:       .ASCIC  "?? Reserved APT command received"      ;               [05]
64 6E 61 6D 6D 6F 63 20 54 50 41 20  0010
         64 65 76 69 65 63 65 72 20  001C
                                 20  0004
```

```
                     0025   120          .SBTTL  APT SUPPORT ROUTINES
                 00000000   121          .PSECT  Code, Exe, Shr, NoWrt, Long                    ;                [05]
                     0000   122 ;++
                     0000   123 ; FUNCTIONAL DESCRIPTION:
                     0000   124 ;
                     0000   125 ;
                     0000   126 ; CALLING SEQUENCE:
                     0000   127 ;
                     0000   128 ;       NONE
                     0000   129 ;
                     0000   130 ; INPUT PARAMETERS:
                     0000   131 ;
                     0000   132 ;       NONE
                     0000   133 ;
                     0000   134 ; IMPLICIT INPUTS:
                     0000   135 ;
                     0000   136 ;       NONE
                     0000   137 ;
                     0000   138 ; OUTPUT PARAMETERS:
                     0000   139 ;
                     0000   140 ;       NONE
                     0000   141 ;
                     0000   142 ; IMPLICIT OUTPUTS:
                     0000   143 ;
                     0000   144 ;       NONE
                     0000   145 ;
                     0000   146 ; COMPLETION CODES:
                     0000   147 ;
                     0000   148 ;       NONE
                     0000   149 ;
                     0000   150 ; SIDE EFFECTS:
                     0000   151 ;
                     0000   152 ;       NONE
                     0000   153 ;
                     0000   154 ;--
                     0000   155
                     0000   156 APT_COM::
                     0000   157          CASE    DSA$GL_APTCOM,LIMIT=#0,TYPE=L,<-
                     0000   158          COM_NOP,-                                              ; <000> NOP
                     0000   159          COM_RSRV,-                                             ; <001> RESERVED
                     0000   160          COM_PTEXT,-                                            ; <002> PROCESS PTEXT
                     0000   161          COM_DUMP,-                                             ; <003> DUMP
                     0000   162          COM_ZERO,-                                             ; <004> ZERO CORE
                     0000   163          COM_START,-                                            ; <005> START DIAGNOSTIC
                     0000   164          COM_ABORT,-                                            ; <006> ABORT DIAGNOSTIC
                     0000   165          COM_RSRV,-                                             ; <007> RESERVED
                     0000   166          COM_RSRV,-                                             ; <010> RESERVED
                     0000   167          COM_CONT,-                                             ; <011> CONTINUE
                     0000   168          >
09'  00   0000FE04'EF   CF   0000               CASEL   DSA$GL_APTCOM,#0,S^#<<30001$-30000$>/2>-1
                     0008       30000$:
                0036' 0008               .SIGNED_WORD    COM_NOP-30000$
                0014' 000A               .SIGNED_WORD    COM_RSRV-30000$
                00DD' 000C               .SIGNED_WORD    COM_PTEXT-30000$
                0043' 000E               .SIGNED_WORD    COM_DUMP-30000$
                007B' 0010               .SIGNED_WORD    COM_ZERO-30000$
                00AA' 0012               .SIGNED_WORD    COM_START-30000$
```

```
                        00BD' 0014                .SIGNED_WORD    COM_ABORT-30000$
                        0014' 0016                .SIGNED_WORD    COM_RSRV-30000$
                        0014' 0018                .SIGNED_WORD    COM_RSRV-30000$
                        00D0' 001A                .SIGNED_WORD    COM_CONT-30000$
                              001C          30001$:
                              001C     169
                              001C     170 COM_RSRV:
              0000FE04'EF   D4   001C     171          CLRL    DSA$GL_APTCOM                      ; CLEAR COMMAND
              0000FE40'EF   D4   0022     172          CLRL    DSA$GL_MSGTYP                      ; CLEAR MESSAGE TYPE
                              0028     173          ERRSUP_S        MSGADR=T_PRGERR ; RESERVED COMMAND CODE
              00000000'EF   DF   0028                  PUSHAL  $MODULE
              00000004'EF   DF   002E                  PUSHAL  T_PRGERR
                        01  DD   0034                  PUSHL   #$ER
           00000000'EF    03   FB   0036                  CALLS   #3, DS_ERRSUP
                              05   003D     174          RSB                                        ; RETURN
                              003E     175
                              003E     176 COM_NOP:
              0000FE04'EF   D4   003E     177          CLRL    DSA$GL_APTCOM                      ; CLEAR COMMAND
              0000FE40'EF   D4   0044     178          CLRL    DSA$GL_MSGTYP                      ; CLEAR MESSAGE TYPE
                              05   004A     179          RSB                                        ; RETURN
                              004B     180
                              004B     181 COM_DUMP:
              0000FE04'EF   D4   004B     182          CLRL    DSA$GL_APTCOM                      ; CLEAR COMMAND
              0000FE40'EF   D4   0051     183          CLRL    DSA$GL_MSGTYP                      ; CLEAR MESSAGE TYPE
    0000FE00'EF   88000000 8F   CA   0057     184          BICL    #DSA$M_APT!DSA$M_NORPT,DSA$GL_FLAGS     ; CLEAR APT AND NO REPORT
    0000FE00'EF   00001000 8F   C8   0062     185          BISL    #DSA$M_OPER,DSA$GL_FLAGS           ; INDICATE OPERATOR PRESENT
                        06  C8   006D     186          BISL    #DS$M_LODFLG!DS$M_STRFLG,-
              00000000'EF   006F     187                  DS$GL_FLAGS                        ; INDICATE PROGRAM LOADED AND STARTE
         04 A2  00000000'EF   9E   0074     188          MOVAB   VRRESTART,CLI$L_COMMAND(R2)        ; LOAD DUMMY RESTART COMMAND
              00000000'EF   16   007C     189          JSB     VRRESTART                          ; DO A RESTART                  [05]
                              05   0082     190          RSB                                        ; RETURN
```

```
                                0083   COM_ZERO:
      00000000'8F    00    DA   0083           MTPR    #0,#PR$_MAPEN                  ; TURN OFF MEMORY MANAGEMENT IF ENAB
           0000FE04'EF      D4   008A           CLRL    DSA$GL_APTCOM                  ; CLEAR COMMAND
           0000FE40'EF      D4   0090           CLRL    DSA$GL_MSGTYP                  ; CLEAR MESSAGE TYPE
      0000001C'EF    00    D2   0096           MCOML   #0,DS$GL_CLIBASE+CLI$L_DATA    ; SET UP AN "ALL" INDICATOR
           00000000'EF      16   009D           JSB     VRCLRBRK                       ;  AND GO CLEAR BREAKPOINTS      [05]
                      50    D4   00A3           CLRL    R0                             ; CLEAR INDEX
                                00A5
                0000'C0    7C   00A5   10$:    CLRQ    DS$A_PRGBGN(R0)                ; CLEAR
   F4 50    00000000'8F    F2   00A9           AOBLSS  #<DS$K_PRGSIZ/8>,R0,10$        ; ALL LOCATIONS
                           05   00B1           RSB                                    ; RETURN
                                00B2
                                00B2   COM_START:
           0000FE04'EF      D4   00B2           CLRL    DSA$GL_APTCOM                  ; CLEAR COMMAND
           0000FE40'EF      D4   00B8           CLRL    DSA$GL_MSGTYP                  ; CLEAR MESSAGE TYPE
           00000000'EF      16   00BE           JSB     VRSTART                        ; START                          [05]
                           05   00C4           RSB                                    ; RETURN
                                00C5
                                00C5   COM_ABORT:
           0000FE04'EF      D4   00C5           CLRL    DSA$GL_APTCOM                  ; CLEAR COMMAND
           0000FE40'EF      D4   00CB           CLRL    DSA$GL_MSGTYP                  ; CLEAR MESSAGE TYPE
           00000000'EF      16   00D1           JSB     VRABORT                        ; ABORT                          [05]
                           05   00D7           RSB                                    ; RETURN
                                00D8
                                00D8   COM_CONT:
           0000FE04'EF      D4   00D8           CLRL    DSA$GL_APTCOM                  ; CLEAR COMMAND
           0000FE40'EF      D4   00DE           CLRL    DSA$GL_MSGTYP                  ; CLEAR MESSAGE TYPE
                           04   00E4           RET                                    ; RETURN TO CLI'S CALLER
```

```
                                  00E5
                                  00E5   COM_PTEXT:
                       30   BB    00E5           PUSHR   #^M<R4,R5>                  ; Save a couple
55   0000FA00'EF       9E    00E7           MOVAB   DSA$AT_APTTXT,R5            ; Point to text area
                                  00EE
                       04   11    00EE           BRB     15$                         ; Assume must be some text
                       65   95    00F0   10$:    TSTB    (R5)                        ; Any text left?
                       26   13    00F2           BEQL    25$                         ; NO, exit
                                  00F4   15$:
50   0000FE00'EF       9E    00F4           MOVAB   DSA$AT_APTTXT+1024,R0       ; Point past two pages
          54   65       9E    00FB           MOVAB   (R5),R4                     ; Point to current position
     64   0A0D 8F       B1    00FE   20$:    CMPW    #^X0A0D,(R4)                ; CRLF here?
                       18   13    0103           BEQL    30$                         ; Branch if it is
       F5 54   50       F2    0105           AOBLSS  R0,R4,20$                   ; increment and try next
                                  0109           ERRSUP_S
     00000000'EF       DF    0109                       PUSHAL  $MODULE
                       00   DD    010F                       PUSHL   #0
                       02   DD    0111                       PUSHL   #$ER
00000000'EF            03   FB    0113                       CALLS   #3, DS_ERRSUP
                     0090   31    011A   25$:    BRW     100$                        ; Exit
                                  011D
          54   55       C2    011D   30$:    SUBL2   R5,R4                       ; Length of text just scanned
          54   04       C2    0120           SUBL    #4,R4                       ; Must be at least 4 characters
                       13   18    0123           BGEQ    40$                         ; Branch if enough
                                  0125           ERRSUP_S
     00000000'EF       DF    0125                       PUSHAL  $MODULE
                       00   DD    012B                       PUSHL   #0
                       03   DD    012D                       PUSHL   #$ER
00000000'EF            03   FB    012F                       CALLS   #3, DS_ERRSUP
                       75   11    0136           BRB     100$                        ; Exit
                                  0138
85   00000001'EF       D1    0138   40$:    CMPL    L^DS$GT_PROMPT+1,(R5)+      ; Prompt present?   "DS> "          [11]
                       13   13    013F           BEQL    45$                         ; Branch if present
                                  0141           ERRSUP_S                            ; no prompt present
     00000000'EF       DF    0141                       PUSHAL  $MODULE
                       00   DD    0147                       PUSHL   #0
                       04   DD    0149                       PUSHL   #$ER
00000000'EF            03   FB    014B                       CALLS   #3, DS_ERRSUP
                       59   11    0152           BRB     100$                        ; Exit
                                  0154
                       03   BB    0154   45$:    PUSHR   #^M<R0,R1>                  ; Better saved than sorry.          [09]
50   0000044C'EF       DE    0156           MOVAL   L^DS$GL_CLIBASE+CLI$K_SIZE,R0   ; Start at the end.          [09]
                       70   D4    015D           CLRL    -(R0)                       ; After this it is quad aligned     [10]
51   00000089 8F       D0    015F           MOVL    #CLI$K_SIZE/8,R1            ; Number of QUAD words to clear     [09]
                                  0166
                       70   7C    0166   50$:    CLRQ    -(R0)                       ; Loop til we reach beginning       [09]
            FB 51       F5    0168           SOBGTR  R1,50$                      ;                                   [09]
                       03   BA    016B           POPR    #^M<R0,R1>                  ; Restore                           [09]
                                  016D
      34 A2   54       7D    016D           MOVQ    R4,CLI$Q_BUFQWD(R2)         ; Set length and address of input
                                  0171
     00000000'EF       DF    0171           PUSHAL  L^CLI_ACTION                ; Push SUPERPARSE params            [06]
     00000000'EF       DF    0177           PUSHAL  L^COMMAND_TREE              ;                                   [06]
          34 A2       7F    017D           PUSHAQ  CLI$Q_BUFQWD(R2)            ;                                   [06]
00000000'9F            03   FB    0180           CALLS   #3, a#DSX$SUPERPARSE        ; Call the Supervisor Parser        [06]
                                  0187
          13 50       E8    0187           BLBS    R0,60$                      ; Branch if it worked               [06]
```

```
                          018A   274          ERRSUP_S                          ; OOPS, APT should not make mistakes
     00000000'EF    DF    018A                        PUSHAL  $MODULE
                00    DD    0190                        PUSHL   #0
                05    DD    0192                        PUSHL   #$ER
00000000'EF     03    FB    0194                        CALLS   #3, DS_ERRSUP
                10    11    019B   275          BRB     100$                      ; Exit
                          019D   276
                          019D   277 ;                                                                                [06]
                          019D   278 ; No longer need to check NOTNUF flag.                                           [06]
                          019D   279 ;                                                                                [06]
                          019D   280
      50    04 A2    D0    019D   281 60$:     MOVL    CLI$L_COMMAND(R2),R0      ; Address of processing routine
                02    13    01A1   282          BEQL    70$                       ; Branch if none
                60    16    01A3   283          JSB     (R0)                      ; Call it
                          01A5   284
     55    02 A544    9E    01A5   285 70$:     MOVAB   2(R5)[R4],R5              ; Point past current line
              FF43    31    01AA   286          BRW     10$                       ; Try again
                          01AD   287
                30    BA    01AD   288 100$:    POPR    #^M<R4,R5>                ; Restore
     0000FE04'EF    D4    01AF   289          CLRL    DSA$GL_APTCOM             ; Clear command
     0000FE40'EF    D4    01B5   290          CLRL    DSA$GL_MSGTYP             ; Clear error text
                    05    01BB   291          RSB                               ; Return
```

```
                          01BC   293
                          01BC   294 ;+
                          01BC   295 ; ROUTINE TO SYNCHRONIZE SUPERVISOR
                          01BC   296 ; WITH APT RECIEPT OF ERROR MESSAGES
                          01BC   297 ;-
                          01BC   298 APT_MSG::
0000FE00'EF   01   D3     01BC   299          BITL    #DSA$M_HALT, -          ; IF HALT ON ERROR                [07]
                          01C3   300                  DSA$GL_FLAGS            ;
              0E   12     01C3   301          BNEQ    APT_RTN                 ; IN-LINE BPT HAS ALREADY
                          01C5   302                                          ; BEEN SET
                          01C5   303 APT_DONE::
                          01C5   304 10$:
00000000'EF   16          01C5   305          JSB     KB_CHECK                ; Check for ^C at terminal        [05]
0000FE40'EF   D5          01CB   306          TSTL    DSA$GL_MSGTYP           ; WAIT FOR MESSAGE TO BE
                          01D1   307                                          ; ACKNOWLEDGED
              F2   12     01D1   308          BNEQ    10$                     ; LOOP
                          01D3   309 APT_RTN:
                    05    01D3   310          RSB                             ; RETURN
                          01D4   311 .END
```

```
$ER                             = 00000006     D
$MODULE                           00000000 R   D  02
APT_COM                           00000000 RG  D  03
APT_DONE                          000001C5 RG  D  03
APT_MSG                           000001BC RG  D  03
APT_RTN                           000001D3 R   D  03
BIT...                          = 0000001A     D
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_SIZE                        0000044C     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_FLAGS2                      00000444     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEMP_BASE                   00000448     D
CLI$L_TEST                        00000020     D
CLI$M_START_OR_RUN              = 00000008     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BASE_QUAL                 = 00000002     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NEXT                      = 00000001     D
CLI$V_NOALLOC                   = 00000000     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_START_OR_RUN              = 00000003     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
CLI_ACTION                        ********     X   03
COMMAND_TREE                      ********     X   03
COM_ABORT                         000000C5 R   D   03
COM_CONT                          000000D8 R   D   03
COM_DUMP                          0000004B R   D   03
COM_NOP                           0000003E R   D   03
COM_PTEXT                         000000E5 R   D   03
COM_RSRV                          0000001C R   D   03
COM_START                         000000B2 R   D   03
COM_ZERO                          00000083 R   D   03
DS$A_PRGBGN                       ********     X   03
DS$GL_CLIBASE                     ********     X   03
DS$GL_FLAGS                       ********     X   03
DS$GT_PROMPT                      ********     X   03
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRGSIZ                       ********     X   03
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_NI                         = 04000000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMED                      = 02000000     D
DS$M_TIMED_OUT                  = 08000000     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
```

```
DS$V_BATCH                      = 00000016    D       DS$_NODE                        = 00660128    D
DS$V_BRKCLR                     = 0000000C    D       DS$_NOPCS                       = 00660110    D
DS$V_BRKPT                      = 0000000B    D       DS$_NORMAL                      = 00660001    D
DS$V_CHARFLG                    = 00000008    D       DS$_NOSUPPORT                   = 006600B1    D
DS$V_CMDFLG                     = 00000007    D       DS$_NOTALLOWMP                  = 00660148    D
DS$V_CTRLC                      = 00000000    D       DS$_NOTDON                      = 00660030    D
DS$V_CTRLO                      = 00000010    D       DS$_NOTIMP                      = 006600B0    D
DS$V_DEVFLG                     = 00000009    D       DS$_NULLSTR                     = 00660010    D
DS$V_DISABLCC                   = 00000018    D       DS$_OVERFLOW                    = 00660008    D
DS$V_DONFLG                     = 0000000D    D       DS$_POWER                       = 00660098    D
DS$V_ERRFLG                     = 00000004    D       DS$_PROGERR                     = 00660020    D
DS$V_EXCEPT                     = 00000013    D       DS$_SEVERE                      = 00660004    D
DS$V_EXETST                     = 00000012    D       DS$_TRANSL                      = 006600A0    D
DS$V_HLTFLG                     = 00000003    D       DS$_TRUNCATE                    = 00660028    D
DS$V_LODFLG                     = 00000001    D       DS$_UNEXPINT                    = 006600D8    D
DS$V_MEMMGT                     = 0000000F    D       DS$_UNSUPPDEV                   = 00660158    D
DS$V_NI                         = 0000001A    D       DS$_VASFULL                     = 00660048    D
DS$V_OUTPUT                     = 00000017    D       DS$_WARNING                     = 00660000    D
DS$V_RUBFLG                     = 00000005    D       DSA$AL_APTMAIL                    0000FE00    D
DS$V_SCRIPT                     = 00000015    D       DSA$AT_APTTXT                     0000FA00    D
DS$V_SETIMR                     = 00000019    D       DSA$GL_APTCOM                     0000FE04    D
DS$V_STRFLG                     = 00000002    D       DSA$GL_DEVLEN                     0000FE58    D
DS$V_SUBT                       = 0000000E    D       DSA$GL_ERRNO                      0000FE44    D
DS$V_SYSFLG                     = 0000000A    D       DSA$GL_EVENT                      0000FE48    D
DS$V_TIMED                      = 00000019    D       DSA$GL_FLAGS                      0000FE00    D
DS$V_TIMED_OUT                  = 0000001B    D       DSA$GL_MSGTYP                     0000FE40    D
DS$V_TIMRON                     = 00000011    D       DSA$GL_PASSES                     0000FE08    D
DS$_AP_NORMAL_BREAK             = 00660150    D       DSA$GL_PASSNO                     0000FE54    D
DS$_ARITH                       = 006600D0    D       DSA$GL_SECTNO                     0000FE10    D
DS$_ASBE                        = 00660118    D       DSA$GL_SID                        0000FE14    D
DS$_BADLINK                     = 006600F0    D       DSA$GL_SUBTNO                     0000FE4C    D
DS$_BADTYPE                     = 006600E8    D       DSA$GL_TESTNO                     0000FE50    D
DS$_BIIC                        = 00660120    D       DSA$GL_UNITS                      0000FE0C    D
DS$_CHME                        = 006600A8    D       DSA$GQ_MSGPTR                     0000FE68    D
DS$_CHMK                        = 006600E0    D       DSA$GT_DEVNAM                     0000FE5C    D
DS$_DEVNAME                     = 00660108    D       DSA$M_APT                       = 80000000    D
DS$_ERROR                       = 00660002    D       DSA$M_HALT                      = 00000001    D
DS$_FHWE                        = 00660068    D       DSA$M_NORPT                     = 08000000    D
DS$_FRAGBUF                     = 00660080    D       DSA$M_OPER                      = 00001000    D
DS$_ICBUSY                      = 006600C8    D       DSX$SUPERPARSE                    ********    X    03
DS$_ICERR                       = 006600C0    D       DS_ERRSUP                         ********    X    03
DS$_IHWE                        = 00660060    D       KB_CHECK                          ********    X    03
DS$_ILLCHAR                     = 00660018    D       L$A_CCP                           00000240    D
DS$_ILLPAGCNT                   = 00660078    D       L$A_DEVP                          0000021C    D
DS$_ILLUNIT                     = 00660100    D       L$A_DREG                          00000224    D
DS$_INIT_FAIL                   = 00660140    D       L$A_DTP                           00000218    D
DS$_INSFMEM                     = 00660050    D       L$A_ICP                           0000023C    D
DS$_INVCPU                      = 00660130    D       L$A_LASTADR                       00000214    D
DS$_IPL2HI                      = 006600B8    D       L$A_NAME                          00000208    D
DS$_IVADDR                      = 00660040    D       L$A_REPP                          00000244    D
DS$_IVVECT                      = 00660038    D       L$A_SECNAM                        00000250    D
DS$_KRNLSTK                     = 00660090    D       L$A_STATAB                        00000248    D
DS$_LOGIC                       = 00660070    D       L$A_TSTCNT                        00000254    D
DS$_MCHK                        = 00660088    D       L$L_ENVIRON                       00000204    D
DS$_MEM_ALLOC_ERR               = 00660138    D       L$L_ERRTYP                        0000024C    D
DS$_MMOFF                       = 00660058    D       L$L_HEADLENGTH                    00000200    D
DS$_NEEDUNIT                    = 006600F8    D       L$L_REV                           0000020C    D
```

```
L$L_UNIT                    00000220    D
L$L_UNUSED                  00000228    D
L$L_UPDATE                  00000210    D
PR$_MAPEN                   ********   X   03
SIZ...                    = 00000001    D
T_PRGERR                    00000004 R  D  02
VRABORT                     ********   X   03
VRCLRBRK                    ********   X   03
VRRESTART                   ********   X   03
VRSTART                     ********   X   03

                                      +----------------+
                                      ! Psect synopsis !
                                      +----------------+

PSECT name                  Allocation          PSECT No.  Attributes
----------                  ----------          ---------  ----------
.  ABS  .                   00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                       0000FE70  (65136.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
DATA                        00000025  (   37.)  02 (  2.)  NOPIC   USR   CON   REL   LCL   SHR NOEXE   RD  NOWRT NOVEC LONG
CODE                        000001D4  (  468.)  03 (  3.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000006	274 (5)	#-173 (2) #-235 (5) #-241 (5) #-246 (5) #-274 (5)
$MODULE	00000000-R	117 (1)	173 (2) 235 (5) 241 (5) 246 (5) 274 (5)
APT_COM	00000000-R	156 (2)	
APT_DONE	000001C5-R	303 (7)	
APT_MSG	000001BC-R	298 (7)	
APT_RTN	000001D3-R	309 (7)	#-301 (7)
BIT...	=0000001A	110 (1)	106 (1) 110 (1)
CLI$K_BUFSIZ	=00000100	109 (1)	109 (1)
CLI$K_SIZE	0000044C	109 (1)	250 (5) 251 (5) 254 (5) 257 (5) #-260 (5)
CLI$L_ADDRESS	00000018	109 (1)	
CLI$L_COMMAND	00000004	109 (1)	#-188 (2) #-281 (5)
CLI$L_DATA	0000001C	109 (1)	#-196 (3)
CLI$L_FLAGS	00000000	109 (1)	
CLI$L_FLAGS2	00000444	109 (1)	
CLI$L_LAST	00000024	109 (1)	
CLI$L_NEXT	00000030	109 (1)	
CLI$L_PASS	0000002C	109 (1)	
CLI$L_SUBT	00000028	109 (1)	
CLI$L_TEMP_BASE	00000448	109 (1)	
CLI$L_TEST	00000020	109 (1)	
CLI$M_START_OR_RUN	=00000008	109 (1)	
CLI$Q_BUFQWD	00000034	109 (1)	#-266 (5) 270 (5)
CLI$Q_FILE	00000008	109 (1)	
CLI$Q_SECTION	00000010	109 (1)	
CLI$Q_TIME	0000043C	109 (1)	
CLI$T_BUFFER	0000003C	109 (1)	
CLI$V_ADAPTER	=00000018	109 (1)	
CLI$V_ADR	=0000000B	109 (1)	
CLI$V_ASCII	=00000013	109 (1)	
CLI$V_BASE_QUAL	=00000002	109 (1)	
CLI$V_BREAK	=0000000A	109 (1)	
CLI$V_BRIEF	=0000001B	109 (1)	
CLI$V_BYTE	=0000000D	109 (1)	
CLI$V_CLEAR	=00000002	109 (1)	
CLI$V_DEC	=00000010	109 (1)	
CLI$V_DEFAULT	=0000000C	109 (1)	
CLI$V_DEPOSIT	=00000019	109 (1)	
CLI$V_EVENT	=00000008	109 (1)	
CLI$V_EXAM	=00000005	109 (1)	
CLI$V_FLAGS	=00000009	109 (1)	
CLI$V_HEX	=00000012	109 (1)	
CLI$V_KERNEL	=00000017	109 (1)	
CLI$V_LOAD	=00000006	109 (1)	
CLI$V_LONG	=0000000F	109 (1)	
CLI$V_NEXT	=00000001	109 (1)	
CLI$V_NOALLOC	=00000000	109 (1)	
CLI$V_NOTNUF	=00000001	109 (1)	

```
CLI$V_OCT                       =00000011     109    (1)
CLI$V_PREG                      =0000001A     109    (1)
CLI$V_QA                        =00000007     109    (1)
CLI$V_QACKLOOPLOOPS             =0000001C     109    (1)
CLI$V_QAERRORPRINTS             =0000001B     109    (1)
CLI$V_QAMULTIPLEPASS            =0000001F     109    (1)
CLI$V_QASUBTESTLOOPS            =0000001E     109    (1)
CLI$V_QATESTLOOPS               =0000001D     109    (1)
CLI$V_REG                       =00000014     109    (1)
CLI$V_REQUIRED                  =00000000     109    (1)
CLI$V_RUN                       =00000015     109    (1)
CLI$V_SET                       =00000003     109    (1)
CLI$V_SHOW                      =00000004     109    (1)
CLI$V_START_OR_RUN              =00000003     109    (1)      109    (1)
CLI$V_VALSEC                    =00000016     109    (1)
CLI$V_WORD                      =0000000E     109    (1)
CLI_ACTION                       00000000-XR                 268    (5)
COMMAND_TREE                     00000000-XR                 269    (5)
COM_ABORT                        000000C5-R    210    (3)      168    (2)
COM_CONT                         000000D8-R    216    (3)      168    (2)
COM_DUMP                         0000004B-R    181    (2)      168    (2)
COM_NOP                          0000003E-R    176    (2)      168    (2)
COM_PTEXT                        000000E5-R    222    (5)      168    (2)
COM_RSRV                         0000001C-R    170    (2)      168    (2)
COM_START                        000000B2-R    204    (3)      168    (2)
COM_ZERO                         00000083-R    192    (3)      168    (2)
DS$A_PRGBGN                      00000000-XR                 #-200    (3)
DS$GL_CLIBASE                    00000000-XR                 #-196    (3)      250    (5)
DS$GL_FLAGS                      00000000-XR                 #-187    (2)
DS$GT_PROMPT                     00000000-XR                 #-244    (5)
DS$K_ERROR                      =00000002     106    (1)
DS$K_NORMAL                     =00000001     106    (1)
DS$K_PRGSIZ                      00000000-XR                 #-201    (3)
DS$K_SEVERE                     =00000004     106    (1)
DS$K_SUBSYS                     =00000066     106    (1)      106    (1)
DS$K_WARNING                    =00000000     106    (1)
DS$M_ABRTFLG                    =00000040     110    (1)
DS$M_BADTIME                    =00100000     110    (1)
DS$M_BATCH                      =00400000     110    (1)
DS$M_BRKCLR                     =00001000     110    (1)
DS$M_BRKPT                      =00000800     110    (1)
DS$M_CHARFLG                    =00000100     110    (1)
DS$M_CMDFLG                     =00000080     110    (1)
DS$M_CTRLC                      =00000001     110    (1)
DS$M_CTRLO                      =00010000     110    (1)
DS$M_DEVFLG                     =00000200     110    (1)
DS$M_DISABLCC                   =01000000     110    (1)
DS$M_DONFLG                     =00002000     110    (1)
DS$M_ERRFLG                     =00000010     110    (1)
DS$M_EXCEPT                     =00080000     110    (1)
DS$M_EXETST                     =00040000     110    (1)
DS$M_HLTFLG                     =00000008     110    (1)
DS$M_LODFLG                     =00000002     110    (1)      #-186    (2)
DS$M_MEMMGT                     =00008000     110    (1)
DS$M_NI                         =04000000     110    (1)
DS$M_OUTPUT                     =00800000     110    (1)
DS$M_RUBFLG                     =00000020     110    (1)
```

```
DS$M_SCRIPT                     =00200000     110     (1)
DS$M_SETIMR                     =02000000     110     (1)
DS$M_STRFLG                     =00000004     110     (1)     #-186     (2)
DS$M_SUBT                       =00004000     110     (1)
DS$M_SYSFLG                     =00000400     110     (1)
DS$M_TIMED                      =02000000     110     (1)
DS$M_TIMED_OUT                  =08000000     110     (1)
DS$M_TIMRON                     =00020000     110     (1)
DS$V_ABRTFLG                    =00000006     110     (1)
DS$V_BADTIME                    =00000014     110     (1)
DS$V_BATCH                      =00000016     110     (1)
DS$V_BRKCLR                     =0000000C     110     (1)
DS$V_BRKPT                      =0000000B     110     (1)
DS$V_CHARFLG                    =00000008     110     (1)
DS$V_CMDFLG                     =00000007     110     (1)
DS$V_CTRLC                      =00000000     110     (1)
DS$V_CTRLO                      =00000010     110     (1)
DS$V_DEVFLG                     =00000009     110     (1)
DS$V_DISABLCC                   =00000018     110     (1)
DS$V_DONFLG                     =0000000D     110     (1)
DS$V_ERRFLG                     =00000004     110     (1)
DS$V_EXCEPT                     =00000013     110     (1)
DS$V_EXETST                     =00000012     110     (1)
DS$V_HLTFLG                     =00000003     110     (1)
DS$V_LODFLG                     =00000001     110     (1)
DS$V_MEMMGT                     =0000000F     110     (1)
DS$V_NI                         =0000001A     110     (1)
DS$V_OUTPUT                     =00000017     110     (1)
DS$V_RUBFLG                     =00000005     110     (1)
DS$V_SCRIPT                     =00000015     110     (1)
DS$V_SETIMR                     =00000019     110     (1)
DS$V_STRFLG                     =00000002     110     (1)
DS$V_SUBT                       =0000000E     110     (1)
DS$V_SYSFLG                     =0000000A     110     (1)
DS$V_TIMED                      =00000019     110     (1)
DS$V_TIMED_OUT                  =0000001B     110     (1)
DS$V_TIMRON                     =00000011     110     (1)
DS$_AP_NORMAL_BREAK             =00660150     106     (1)
DS$_ARITH                       =006600D0     106     (1)
DS$_ASBE                        =00660118     106     (1)
DS$_BADLINK                     =006600F0     106     (1)
DS$_BADTYPE                     =006600E8     106     (1)
DS$_BIIC                        =00660120     106     (1)
DS$_CHME                        =006600A8     106     (1)
DS$_CHMK                        =006600E0     106     (1)
DS$_DEVNAME                     =00660108     106     (1)
DS$_ERROR                       =00660002     106     (1)
DS$_FHWE                        =00660068     106     (1)
DS$_FRAGBUF                     =00660080     106     (1)
DS$_ICBUSY                      =006600C8     106     (1)
DS$_ICERR                       =006600C0     106     (1)
DS$_IHWE                        =00660060     106     (1)
DS$_ILLCHAR                     =00660018     106     (1)
DS$_ILLPAGCNT                   =00660078     106     (1)
DS$_ILLUNIT                     =00660100     106     (1)
DS$_INIT_FAIL                   =00660140     106     (1)
DS$_INSFMEM                     =00660050     106     (1)
```

```
DS$_INVCPU                      =00660130     106     (1)
DS$_IPL2HI                      =006600B8     106     (1)
DS$_IVADDR                      =00660040     106     (1)
DS$_IVVECT                      =00660038     106     (1)
DS$_KRNLSTK                     =00660090     106     (1)
DS$_LOGIC                       =00660070     106     (1)
DS$_MCHK                        =00660088     106     (1)
DS$_MEM_ALLOC_ERR               =00660138     106     (1)
DS$_MMOFF                       =00660058     106     (1)
DS$_NEEDUNIT                    =006600F8     106     (1)
DS$_NODE                        =00660128     106     (1)
DS$_NOPCS                       =00660110     106     (1)
DS$_NORMAL                      =00660001     106     (1)
DS$_NOSUPPORT                   =006600B1     106     (1)
DS$_NOTALLOWMP                  =00660148     106     (1)
DS$_NOTDON                      =00660030     106     (1)
DS$_NOTIMP                      =006600B0     106     (1)       106     (1)
DS$_NULLSTR                     =00660010     106     (1)
DS$_OVERFLOW                    =00660008     106     (1)
DS$_POWER                       =00660098     106     (1)
DS$_PROGERR                     =00660020     106     (1)
DS$_SEVERE                      =00660004     106     (1)
DS$_TRANSL                      =006600A0     106     (1)
DS$_TRUNCATE                    =00660028     106     (1)
DS$_UNEXPINT                    =006600D8     106     (1)
DS$_UNSUPPDEV                   =00660158     106     (1)
DS$_VASFULL                     =00660048     106     (1)
DS$_WARNING                     =00660000     106     (1)       106     (1)
DSA$AT_APTTXT                    0000FA00                       224     (5)       230     (5)
DSA$GL_APTCOM                    0000FE04                     #-168     (2)     #-171     (2)     #-177     (2)     #-182     (2)
                                                              #-194     (3)     #-205     (3)     #-211     (3)     #-217     (3)
                                                              #-289     (5)
DSA$GL_FLAGS                     0000FE00                     #-184     (2)     #-185     (2)     #-300     (7)
DSA$GL_MSGTYP                    0000FE40                     #-172     (2)     #-178     (2)     #-183     (2)     #-195     (3)
                                                              #-206     (3)     #-212     (3)     #-218     (3)     #-290     (5)
                                                              #-306     (7)
DSA$M_APT                       =80000000                     #-184     (2)
DSA$M_HALT                      =00000001                     #-299     (7)
DSA$M_NORPT                     =08000000                     #-184     (2)
DSA$M_OPER                      =00001000                     #-185     (2)
DSX$SUPERPARSE                   00000000-XR                    271     (5)
DS_ERRSUP                        00000000-XR                    173     (2)       235     (5)       241     (5)       246     (5)
                                                                274     (5)
KB_CHECK                         00000000-XR                    305     (7)
PR$_MAPEN                        00000000-XR                  #-193     (3)
SIZ...                          =00000001     110     (1)       110     (1)
T_PRGERR                         00000004-R    118     (1)       173     (2)
VRABORT                          00000000-XR                    213     (3)
VRCLRBRK                         00000000-XR                    197     (3)
VRRESTART                        00000000-XR                    188     (2)       189     (2)
VRSTART                          00000000-XR                    207     (3)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEF            1           110    (1)
$DEFINI         1           107    (1)      107    (1)      108    (1)
$DS_DSADEF      5           107    (1)      107    (1)
$DS_DSDEF       3           106    (1)      106    (1)
$DS_HDRDEF      2           108    (1)      108    (1)
$EQU            1           110    (1)      106    (1)
$EQULS1         1           106    (1)      106    (1)
$EQULST         1           106    (1)      106    (1)
$GBLINI         2           106    (1)      106    (1)      110    (1)
$PUSHADR        1           173    (2)      173    (2)      235    (5)      241    (5)      246    (5)      274    (5)
$VIELD          1                           110    (1)
$VIELD1         1           110    (1)      110    (1)
CASE            1           157    (2)      157    (2)
CLIDEF          4           109    (1)      109    (1)
DSFDEF          3           110    (1)      110    (1)
ERRSUP_S        1           173    (2)      173    (2)      235    (5)      241    (5)      246    (5)      274    (5)
MODNAM          1           117    (1)      117    (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
AOBLSS	00F2	201	(3)	234	(5)										
BEQL	0013	228	(5)	233	(5)	245	(5)	282	(5)						
BGEQ	0018	240	(5)												
BICL	00CA	184	(2)												
BISL	00C8	185	(2)	186	(2)										
BITL	00D3	299	(7)												
BLBS	00E8	273	(5)												
BNEQ	0012	301	(7)	308	(7)										
BRB	0011	226	(5)	242	(5)	247	(5)	275	(5)						
BRW	0031	236	(5)	286	(5)										
CALLS	00FB	173	(2)	235	(5)	241	(5)	246	(5)	271	(5)	274	(5)		
CASEL	00CF	168	(2)												
CLRL	00D4	171	(2)	172	(2)	177	(2)	178	(2)	182	(2)	183	(2)	194	(3)
		195	(3)	198	(3)	205	(3)	206	(3)	211	(3)	212	(3)	217	(3)
		218	(3)	258	(5)	289	(5)	290	(5)						
CLRQ	007C	200	(3)	262	(5)										
CMPL	00D1	244	(5)												
CMPW	00B1	232	(5)												
JSB	0016	189	(2)	197	(3)	207	(3)	213	(3)	283	(5)	305	(7)		
MCOML	00D2	196	(3)												
MOVAB	009E	188	(2)	224	(5)	230	(5)	231	(5)	285	(5)				
MOVAL	00DE	250	(5)												
MOVL	00D0	260	(5)	281	(5)										
MOVQ	007D	266	(5)												
MTPR	00DA	193	(3)												
POPR	00BA	264	(5)	288	(5)										
PUSHAL	00DF	173	(2)	235	(5)	241	(5)	246	(5)	268	(5)	269	(5)	274	(5)
PUSHAQ	007F	270	(5)												
PUSHL	00DD	173	(2)	235	(5)	241	(5)	246	(5)	274	(5)				
PUSHR	00BB	223	(5)	249	(5)										
RET	0004	219	(3)												
RSB	0005	174	(2)	179	(2)	190	(2)	202	(3)	208	(3)	214	(3)	291	(5)
		310	(7)												
SOBGTR	00F5	263	(5)												
SUBL	00C2	239	(5)												
SUBL2	00C2	238	(5)												
TSTB	0095	227	(5)												
TSTL	00D5	306	(7)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	117	(1)	118	(1)												
.BLKL	109	(1)														
.END	311	(7)														
.ENDC	106	(1)	110	(1)	173	(2)	235	(5)	241	(5)	246	(5)	253	(5)	256	(5)
	259	(5)	274	(5)												
.ENDM	106	(1)	107	(1)	108	(1)	109	(1)	110	(1)	117	(1)	157	(2)	173	(2)
.ENDR	106	(1)	110	(1)	168	(2)										
.IDENT	12	(1)														
.IF	106	(1)	110	(1)	173	(2)	235	(5)	241	(5)	246	(5)	251	(5)	254	(5)
	257	(5)	274	(5)												
.IFF	106	(1)	110	(1)	173	(2)	235	(5)	241	(5)	246	(5)	274	(5)		
.IIF	106	(1)	110	(1)	173	(2)	235	(5)	241	(5)	246	(5)	274	(5)		
.IRP	106	(1)	110	(1)	168	(2)										
.LIBRARY	95	(1)	96	(1)	97	(1)										
.LIST	107	(1)	108	(1)	109	(1)	13	(1)								
.MACRO	106	(1)	110	(1)												
.MDELETE	106	(1)														
.NLIST	107	(1)	108	(1)	109	(1)	14	(1)								
.NOCROSS	107	(1)	108	(1)												
.PAGE	90	(1)														
.PSECT	109	(1)	116	(1)	121	(2)										
.RESTORE	107	(1)	108	(1)	109	(1)										
.SAVE	107	(1)	108	(1)	109	(1)	115	(1)								
.SBTTL	120	(2)	91	(1)												
.SIGNED_WORD	168	(2)														
.TITLE	11	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	13	#-198	(3)	#-200	(3)	#-201	(3)	#-230	(5)	#-234	(5)	#-249	(5)
		#-250	(5)	#-258	(5)	#-262	(5)	#-264	(5)	#-273	(5)	#-281	(5)
		283	(5)										
R1	4	#-249	(5)	#-260	(5)	#-263	(5)	#-264	(5)				
R2	4	#-188	(2)	#-266	(5)	270	(5)	#-281	(5)				
R4	9	#-223	(5)	#-231	(5)	#-232	(5)	#-234	(5)	#-238	(5)	#-239	(5)
		#-266	(5)	285	(5)	#-288	(5)						
R5	9	#-223	(5)	#-224	(5)	#-227	(5)	231	(5)	#-238	(5)	#-244	(5)
		285	(5)	#-288	(5)								

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.01	00:00:00.19
Command processing	876	00:00:00.18	00:00:00.80
Pass 1	465	00:00:01.31	00:00:02.67
Symbol table sort	0	00:00:00.09	00:00:00.08
Pass 2	8	00:00:00.32	00:00:00.46
Symbol table output	0	00:00:00.05	00:00:00.15
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	4	00:00:00.18	00:00:00.23
Assembler run totals	1377	00:00:02.16	00:00:04.59

The working set limit was 3012 pages.
73184 bytes (143 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 291 non-local and 15 local symbols.
311 source lines were read in Pass 1, producing 30 object records in Pass 2.
50 pages of virtual memory were used to define 18 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	4
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	14

357 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:APT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DIA
```

Table of contents

(1) 106 ASSIGN I/O CHANNEL
(1) 308 TEST IF MAILBOX SPECIFIED

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ASSIGN.MAR
0000     2 ; *3    12-MAY-1986 10:10:48 BARANSKI "Cleaning up Object Libraries."
0000     3 ; *2     7-MAY-1986 14:02:49 BARANSKI "Documentaion Check."
0000     4 ; *1     6-FEB-1986 15:13:33 DS ""
0000     5 ; DEC/CMS REPLACEMENT HISTORY, Element ASSIGN.MAR
0000     6         .TITLE  ASSIGN *** ASSIGN assign device
0000     7         .IDENT  "8.3-3"                                                ;G:2
0000     8
0000     9 ;
0000    10 ;*****************************************************************************
0000    11 ;*                                                                           *
0000    12 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980, 1985                               * ;G:2
0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    14 ;*                                                                           *
0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    20 ;*  TRANSFERRED.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    24 ;*  COPPORATION.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    28 ;*                                                                           *
0000    29 ;*****************************************************************************
0000    30 ;
0000    31 ; D. N. CUTLER 25-AUG-76
0000    32 ;
0000    33 ; SYSTEM SERVICE ASSIGN I/O CHANNEL
0000    34 ;
0000    35 ; MODIFICATION HISTORY:
0000    36 ;                                                                          ;G:2
0000    37 ;       03      M. Baggett      14-Aug-1985     Version 8.3
0000    38 ;               Trucation error.
0000    39 ;
0000    40 ;               Dave Butenhof   13-may-1980, Version 5.4
0000    41 ;       02      Modify VMS V2.0 source to run in supervisor environment
0000    42 ;
0000    43 ;               N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
0000    44 ;       01      DIAGNOSTIC SUPERVISOR INTEGRATION
0000    45 ;
0000    46 ;       V08     LMK0003         LEN KAWELL      20-JAN-1980
0000    47 ;               ADD NON-SHAREABLE DEVICE IMPLICIT ALLOCATION PRIVILEGE CHECK.
0000    48 ;
0000    49 ;       V07     RIH0033         R. HUSTVEDT     16-OCT-1979
0000    50 ;               CHANGE PCB$W_BYTCNT TO JIB$L_BYTCNT.
0000    51 ;
0000    52 ;       V06     LMK0002         LEN KAWELL      27-SEP-1979
0000    53 ;               SET CCB$M_AMB IF A MAILBOX WAS ASSOCIATED WITH DEVICE.
0000    54 ;
0000    55 ;       V05     LMK0001         LEN KAWELL      27-JUL-1979
0000    56 ;               RETAINED LOCK OF I/O DATABASE BEFORE CALL TO IOC$CREATE_UCB
0000    57 ;               AS THAT IS THE NEW INTERFACE TO IOC$CREATE_UCB.
```

```
                                0000     58 ;
                                0000     59 ;       V04     ADE0002         A.D.E           22-JUN-1979     13:00
                                0000     60 ;               Test UCB$V_TEMPLATE in UCB$W_STS to determine if a
                                0000     61 ;               UCB needs to be created rather than deadreckoning
                                0000     62 ;               on UCB$NET0
                                0000     63 ;
                                0000     64 ;       V03     SGD0001         S.G.D.          04-APR-1979     17:10
                                0000     65 ;               Use "_NET" instead of "NET".
                                0000     66 ;
                                0000     67 ;       V02     TGD0001         T.G. DOPIRAK    22-FEB-1979
                                0000     68 ;               RETURN SS$_DEVACTIVE IF MAILBOX ALREADY ASSOCIATED
                                0000     69 ;               WITH DEVICE.
                                0000     70 ;
                                0000     71 ;--
                                0000     72
                                0000     73 ;
                                0000     74 ; MACRO LIBRARY CALLS
                                0000     75 ;
                                0000     76
                                0000     77          $CCBDEF                             ;DEFINE CCB OFFSETS
                                0000                .SAVE   LOCAL_BLOCK
                                0000                .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
                    00000004    0000                        .BLKL   1
                                0004                .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
                    00000008    0004                        .BLKL   1
                                0008                .IIF    NB,CCB$B_STS,   CCB$B_STS:
                    00000009    0008                        .BLKB   1
                                0009                .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
                    0000000A    0009                        .BLKB   1
                                000A                .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
                    0000000C    000A                        .BLKW   1
                                000C                .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
                    00000010    000C                        .BLKL   1
                                0010                .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                                0010                .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                                0000                .RESTORE
                                0000     78         $IODEF                              ;DEFINE I/O FUNCTION CODES
                                0000                .SAVE   LOCAL_BLOCK
                                0000                .PSECT  $ABS$,ABS
                    00000000    0010                .=0
                                0000                .RESTORE
                                0000     79         $JIBDEF                             ;DEFINE JIB OFFSETS
                                0000                .SAVE   LOCAL_BLOCK
                                0000                .PSECT  $ABS$,ABS
                    00000000    0010                .=0
                                0000                .RESTORE
                                0000     80         $LOGDEF                             ;DEFINE LOG OFFSETS
                                0000                .SAVE   LOCAL_BLOCK
                                0000                .PSECT  $ABS$,ABS
                    00000000    0010                .=0
                                0000                .RESTORE
                                0000     81         $PCBDEF                             ;DEFINE PCB OFFSETS
                                0000                .SAVE   LOCAL_BLOCK
                                0000                .PSECT  $ABS$,ABS
                    00000000    0010                .=0
                                0000                .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
                    00000004    0000                        .BLKL   1
```

```
                        0004            .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008                0004                    .BLKL   1
                        0008            .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A                0008                    .BLKW   1
                        000A            .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B                000A                    .BLKB   1
                        000B            .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C                000B                    .BLKB   1
                        000C            .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D                000C                    .BLKB   1
                        000D            .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E                000D                    .BLKB   1
                        000E            .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010                000E                    .BLKW   1
                        0010            .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014                0010                    .BLKL   1
                        0014            .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018                0014                    .BLKL   1
                        0018            .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C                0018                    .BLKL   1
                        001C            .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020                001C                    .BLKL   1
                        0020            .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024                0020                    .BLKL   1
                        0024            .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028                0024                    .BLKL   1
                        0028            .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C                0028                    .BLKL   1
                        002C            .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E                002C                    .BLKW   1
                        002E            .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F                002E                    .BLKB   1
                        002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030                002F                    .BLKB   1
                        0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032                0030                    .BLKW   1
                        0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034                0032                    .BLKW   1
                        0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036                0034                    .BLKW   1
                        0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038                0036                    .BLKW   1
                        0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A                0038                    .BLKW   1
                        003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C                003A                    .BLKW   1
                        003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E                003C                    .BLKW   1
                        003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040                003E                    .BLKW   1
                        0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042                0040                    .BLKW   1
                        0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044                0042                    .BLKW   1
                        0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C                0044                    .BLKB   8
                        004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
```

```
                     004C                .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050             004C                        .BLKL   1
                     0050                .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054             0050                        .BLKL   1
                     0054                .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058             0054                        .BLKL   1
                     0058                .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C             0058                        .BLKL   1
                     005C                .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060             005C                        .BLKL   1
                     0060                .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064             0060                        .BLKL   1
                     0064                .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068             0064                        .BLKL   1
                     0068                .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078             0068                        .BLKB   16
                     0078                .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C             0078                        .BLKL   1
                     007C                .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084             007C                        .BLKQ   1
                     0084                .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088             0084                        .BLKL   1
                     0088                .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                     0088                .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A             0088                        .BLKW   1
                     008A                .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C             008A                        .BLKW   1
                     008C                .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                     008C                .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                     0000                .RESTORE
                     0000     82         $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000     83         $PRVDEF                         ;DEFINE PRIVILEGE BITS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000     84         $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000     85         $UCBDEF                         ;DEFINE UCB OFFSETS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                     0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004             0000                        .BLKL   1
                     0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                     0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008             0004                        .BLKL   1
                     0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
0000000A             0008                        .BLKW   1
```

```
                    000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
0000000B  000A                                      .BLKB   1
                    000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
0000000C  000B                                      .BLKB   1
                    000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
0000000D  000C                                      .BLKB   1
00000010  000D                                      .BLKB   3
                    0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014  0010                                      .BLKL   1
                    0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016  0014                                      .BLKW   1
                    0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018  0016                                      .BLKW   1
                    0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
0000001A  0018                                      .BLKW   1
                    001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
0000001C  001A                                      .BLKW   1
                    001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
00000020  001C                                      .BLKL   1
                    0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
00000024  0020                                      .BLKL   1
                    0024                .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
00000028  0024                                      .BLKL   1
                    0028                .IIF    NB,UCB$L_PID,   UCB$L_PID:
0000002C  0028                                      .BLKL   1
                    002C                .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030  002C                                      .BLKL   1
                    0030                .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034  0030                                      .BLKL   1
                    0034                .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038  0034                                      .BLKL   1
                    0038                .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039  0038                                      .BLKB   1
                    0039                .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A  0039                                      .BLKB   1
                    003A                .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C  003A                                      .BLKW   1
                    003C                .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C                .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C                .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D  003C                                      .BLKB   1
                    003D                .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                    003D                .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E  003D                                      .BLKB   1
                    003E                .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                    003E                .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F  003E                                      .BLKB   1
                    003F                .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040  003F                                      .BLKB   1
                    0040                .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
```

```
00000044  0040                          .BLKL   1
          0044          .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                          .BLKL   1
          0048          .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                          .BLKW   1
          004A          .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
          004A          .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                          .BLKB   1
          004B          .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                          .BLKB   1
          004C          .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                          .BLKL   1
          0050          .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                          .BLKW   1
          0052          .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
          0052          .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                          .BLKB   1
          0053          .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                          .BLKB   1
          0054          .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                          .BLKL   1
          0058          .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                          .BLKW   1
          005A          .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                          .BLKW   1
          005C          .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C          .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                          .BLKL   1
          0060          .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                          .BLKL   1
          0064          .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
          0064          .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                          .BLKL   1
          0068          .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                          .BLKL   1
          006C          .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                          .BLKW   1
          006E          .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                          .BLKW   1
          0070          .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                          .BLKB   1
          0071          .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                          .BLKB   1
          0072          .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                          .BLKW   1
          0074          .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
          0074          .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074          . = 0
          0000          .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                          .BLKW   1
00000074  0002          . = 116
          0074          .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                          .BLKL   1
          0078          .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                          .BLKL   1
```

```
                     0080          .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                             .BLKL   1
                     0084          .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                             .BLKL   1
                     0088          .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                             .BLKL   1
                     008C          .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                             .BLKL   1
                     0090          .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                     0090          .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                     . = 116
                     0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                             .BLKB   1
                     0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                             .BLKB   1
                     0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                             .BLKB   1
                     0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                             .BLKB   1
                     0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                             .BLKL   1
0000007E  007C                             .BLKW   1
                     007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                             .BLKW   1
                     0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                             .BLKL   1
00000080  0084                     . = 128
00000084  0080                             .BLKL   1
                     0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                             .BLKL   1
                     0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                             .BLKW   1
                     008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                             .BLKW   1
                     008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                     008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                             .BLKW   1
                     008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                             .BLKW   1
                     0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                             .BLKW   1
                     0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                             .BLKW   1
                     0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                             .BLKB   1
                     0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                             .BLKB   1
                     0096          .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                             .BLKW   1
00000096  0098                     . = 150
00000098  0096                             .BLKW   1
                     0098          .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                             .BLKL   1
                     009C          .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                             .BLKL   1
                     00A0          .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                             .BLKL   1
```

```
                    00A4                .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6            00A4                        .BLKW   1
                    00A6                .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7            00A6                        .BLKB   1
000000A8            00A7                        .BLKB   1
00000074            00A8                . = 116
00000078            0074                        .BLKL   1
0000007C            0078                        .BLKL   1
00000080            007C                        .BLKL   1
00000084            0080                        .BLKL   1
00000088            0084                        .BLKL   1
0000008C            0088                        .BLKL   1
                    008C                .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090            008C                        .BLKL   1
00000094            0090                        .BLKL   1
00000098            0094                        .BLKL   1
0000009C            0098                        .BLKL   1
                    009C                .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D            009C                        .BLKB   1
                    009D                .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E            009D                        .BLKB   1
                    009E                .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F            009E                        .BLKB   1
000000A0            009F                        .BLKB   1
                    00A0                .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1            00A0                        .BLKB   1
                    00A1                .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2            00A1                        .BLKB   1
                    00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3            00A2                        .BLKB   1
000000A4            00A3                        .BLKB   1
                    00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5            00A4                        .BLKB   1
                    00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7            00A5                        .BLKW   1
000000A8            00A7                        .BLKB   1
                    00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC            00A8                        .BLKL   1
000000B0            00AC                        .BLKL   1
000000B4            00B0                        .BLKL   1
000000B8            00B4                        .BLKL   1
                    00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC            00B8                        .BLKL   1
                    00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                    00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074            00BC                . = 116
                    0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078            0074                        .BLKL   1
                    0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C            0078                        .BLKL   1
                    007C                .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E            007C                        .BLKW   1
00000080            007E                        .BLKW   1
                    0000                .RESTORE
                    0000    86
                    0000    87 ;
                    0000    88 ; LOCAL SYMBOLS
```

```
                     0000     89 ;
                     0000     90 ; ARGUMENT LIST OFFSET DEFINITIONS
                     0000     91 ;
                     0000     92
            00000004 0000     93 DEVNAM=4                        ;ADDRESS OF DEVICE NAME STRING DESCRIPTOR
            00000008 0000     94 CHAN=8                          ;ADDRESS TO STORE ASSIGNED CHANNEL NUMBER
            0000000C 0000     95 ACMODE=12                       ;ACCESS MODE
            00000010 0000     96 MBXNAM=16                       ;ADDRESS OF MAILBOX NAME STRING DESCRIPTOR
                     0000     97
                     0000     98 ;
                     0000     99 ; LOCAL DATA
                     0000    100 ;
                     0000    101
                00000000    102          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
      54 45 4E 5F    0000    103 NETNAM:  .ASCII  /_NET/                 ;NETWORK DEVICE NAME
                     0004    104 NETEND:                                 ;REFERENCE LABEL
```

```
0004    106          .SBTTL  ASSIGN I/O CHANNEL
0004    107 ;+
0004    108 ; EXE$ASSIGN - ASSIGN I/O CHANNEL
0004    109 ;
0004    110 ; THIS SERVICE PROVIDES THE CAPABILITY TO ASSIGN A DEVICE TO AN I/O CHANNEL
0004    111 ; AND ESTABLISH NECESSARY DEVICE LINKAGE AND CONTROL INFORMATION IN THE
0004    112 ; ASSOCIATED CHANNEL CONTROL BLOCK. OPTIONALLY A MAILBOX CAN ALSO BE
0004    113 ; SPECIFIED WHICH WILL RECEIVE UNSOLICITED INPUT SENT TO THE ASSIGNED
0004    114 ; DEVICE.
0004    115 ;
0004    116 ; INPUTS:
0004    117 ;
0004    118 ;       DEVNAM(AP) = ADDRESS OF DEVICE NAME STRING DESCRIPTOR.
0004    119 ;       CHAN(AP) = ADDRESS TO STORE ASSIGNED CHANNEL NUMBER.
0004    120 ;       ACMODE(AP) = ACCESS MODE CHANNEL IS TO BE ASSIGNED TO.
0004    121 ;       MBXNAM(AP) = ADDRESS OF MAILBOX NAME STRING DESCRIPTOR (ZERO IMPLIES
0004    122 ;               NONE).
0004    123 ;
0004    124 ;       R4 = CURRENT PROCESS PCB ADDRESS.
0004    125 ;
0004    126 ; OUTPUTS:
0004    127 ;
0004    128 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO ASSIGN CHANNEL TO DEVICE.
0004    129 ;
0004    130 ;               R0 = SS$_ACCVIO - DEVICE NAME STRING, DEVICE NAME STRING
0004    131 ;                       DESCRIPTOR, MAILBOX NAME STRING, OR MAILBOX NAME
0004    132 ;                       STRING DESCRIPTOR CANNOT BE READ BY CALLING ACCESS
0004    133 ;                       MODE, OR CHANNEL NUMBER CANNOT BE WRITTEN BY CALLING
0004    134 ;                       ACCESS MODE.
0004    135 ;
0004    136 ;       R0 = SS$_DEVALLOC - DEVICE ALLOCATED TO ANOTHER PROCESS.
0004    137 ;
0004    138 ;       R0 = SS$_DEVNOTMBX - SPECIFIED MAILBOX DEVICE IS NOT A
0004    139 ;               MAILBOX.
0004    140 ;
0004    141 ;       R0 = SS$_EXQUOTA - PROCESS HAS INSUFFICIENT BUFFER QUOTA TO
0004    142 ;               ALLOCATE NETWORK UCB.
0004    143 ;
0004    144 ;       R0 = SS$_INSFMEM - SUFFICIENT SYSTEM DYNAMIC MEMORY DOES NOT
0004    145 ;               EXIST TO ALLOCATE NETWORK UCB.
0004    146 ;
0004    147 ;       R0 = SS$_IVDEVNAM - DEVICE OR MAILBOX NAME STRING CONTAINS
0004    148 ;               INVALID CHARACTERS, OR NO DEVICE NAME STRING
0004    149 ;               DESCRIPTOR SPECIFIED.
0004    150 ;
0004    151 ;       R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
0004    152 ;               DEVICE OR MAILBOX NAME STRING SPECIFIED.
0004    153 ;
0004    154 ;       R0 = SS$_NOIOCHAN - NO I/O CHANNEL IS AVAILABLE FOR ASSIGNMENT.
0004    155 ;
0004    156 ;       R0 = SS$_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO CREATE
0004    157 ;               NETWORK UCB OR DOES NOT HAVE PRIVILEGE TO ALLOCATE
0004    158 ;               THE DEVICE.
0004    159 ;
0004    160 ;       R0 = SS$_NOSUCHDEV - SPECIFIED DEVICE OR MAILBOX DOES NOT
0004    161 ;               EXIST ON HOST SYSTEM.
0004    162 ;
```

```
                                0004   163 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0004   164 ;
                                0004   165 ;               R0 = SS$_REMOTE - NORMAL COMPLETION, ASSIGNMENT COMPLETED
                                0004   166 ;                    ON REMOTE SYSTEM.
                                0004   167 ;
                                0004   168 ;               R0 = SS$_NORMAL - NORMAL COMPLETION, ASSIGNMENT COMPLETED
                                0004   169 ;                    ON HOST SYSTEM.
                                0004   170 ;
                                0004   171 ;               R0 = SS$_DEVACTIVE - MAILBOX ALREADY ASSOCIATED WITH DEVICE
                                0004   172 ;-
                                0004   173
                           0FFC 0004   174          .ENTRY  EXE$ASSIGN,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      54  00000000'EF    9E 0006   175          MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
        5B    08 AC    D0 000D   176          MOVL    CHAN(AP),R11            ;GET ADDRESS TO STORE CHANNEL NUMBER
                         11 0011   177          IFNOWRT #2,(R11),30$            ;CAN CHANNEL NUMBER BE WRITTEN?
    6B    02    00    0D 0011                       PROBEW  #0,#2,(R11)
                      3B   13 0015                       BEQL    30$
                      6B   B4 0017   178          CLRW    (R11)                   ;CLEAR CHANNEL NUMBER IN CASE OF ERROR
      5A    10 AC    D0 0019   179          MOVL    MBXNAM(AP),R10          ;GET ADDRESS OF MAILBOX NAME DESCRIPTOR
                      0C   13 001D   180          BEQL    10$                     ;IF EQL NO MAILBOX SPECIFIED
                         0C 001F   181          IFNORD  #8,(R10),30$            ;CAN MAILBOX DESCRIPTOR BE READ?
    6A    08    00    0C 001F                       PROBER  #0,#8,(R10)
                      2D   13 0023                       BEQL    30$
        7E    6A    7D 0025   182          MOVQ    (R10),-(SP)             ;COPY MAILBOX NAME DESCRIPTOR
        5A    5E    D0 0028   183          MOVL    SP,R10                  ;SET ADDRESS OF MAILBOX NAME DESCRIPTOR
    50  0144 8F    3C 002B   184 10$:     MOVZWL  #SS$_IVDEVNAM,R0        ;SET INVALID DEVICE NAME STATUS
      59    04 AC    D0 0030   185          MOVL    DEVNAM(AP),R9           ;GET ADDRESS OF DEVICE NAME DESCRIPTOR
                      1B   13 0034   186          BEQL    20$                     ;IF EQL NO DEVICE SPECIFIED
                         0C 0036   187          IFNORD  #8,(R9),30$             ;CAN DEVICE NAME DESCRIPTOR BE READ?
    69    08    00    0C 0036                       PROBER  #0,#8,(R9)
                      16   13 003A                       BEQL    30$
50  0C AC    02    00    EF 003C   188          EXTZV   #0,#2,ACMODE(AP),R0     ;GET SPECIFIED ACCESS MODE
      00000000'EF    16   0042   189          JSB     EXE$MAXACMODE           ;MAXIMIZE ACCESS MODE                   [03]
        58    50    D0 0048   190          MOVL    R0,R8                   ;SAVE MAXIMIZED ACCESS MODE
          FFB2'    30 004B   191          BSBW    IOC$FFCHAN              ;FIND FREE I/O CHANNEL
        05 50    E8 004E   192          BLBS    R0,FREECHAN             ;IF LBS FREE I/O CHANNEL FOUND
                      04   0051   193 20$:     RET                             ;
    50    0C    3C 0052   194 30$:     MOVZWL  #SS$_ACCVIO,R0          ;SET ACCESS VIOLATION STATUS
                      04   0055   195          RET                             ;
                           0056   196
                           0056   197 ;
                           0056   198 ; FREE CHANNEL FOUND
                           0056   199 ;
                           0056   200
                           0056   201          .ENABL  LSB
                           0056   202 FREECHAN:                              ;FREE CHANNEL FOUND
    57    51    D0 0056   203          MOVL    R1,R7                   ;SAVE CHANNEL INDEX
        00B2    30 0059   204          BSBW    TEST_MAILBOX            ;TEST IF MAILBOX SPECIFIED
      3D 50    E9 005C   205          BLBC    R0,30$                  ;IF LBC SEARCH FAILURE
    51    59    D0 005F   206          MOVL    R9,R1                   ;SET ADDRESS OF DEVICE NAME DESCRIPTOR
        FF9B'    30 0062   207          BSBW    IOC$SEARCHDEV           ;SEARCH FOR DEVICE
      36 50    E9 0065   208          BLBC    R0,40$                  ;IF LBC SEARCH FAILURE
                           0068   209
                           0068   210 ;
                           0068   211 ; DEVICE FOUND
                           0068   212 ;
                           0068   213
```

```
                55   51   D0  0068   214          MOVL    R1,R5                           ;SAVE ADDRESS OF DEVICE UCB
          6C 34 A5   00'  E0  006B   215          BBS     S^#DEV$V_SPL,UCB$L_DEVCHAR(R5),80$ ;IF SET, SPOOLED DEVICE
                     0D   E1  0070   216          BBC     S^#UCB$V_TEMPLATE,-             ;IF CLR, ASSIGNING TO A LOCAL DEVICE
                03 58 A5         0072   217                  UCB$W_STS(R5),LOCAL             ;
                     0084   31  0075   218          BRW     NETWORK                         ;IF SET, NETWORK ASSIGN
                                 0078   219
                                 0078   220 ;
                                 0078   221 ; LOCAL ASSIGNMENT
                                 0078   222 ;
                                 0078   223
                                 0078   224 LOCAL:                                        ;LOCAL ASSIGNMENT
             50    28 A5   D0  0078   225          MOVL    UCB$L_PID(R5),R0                ;GET PROCESS ID OF OWNER
                     2A   13  007C   226          BEQL    50$                             ;IF EQL DEVICE NOT ALLOCATED
                51   54   D0  007E   227          MOVL    R4,R1                           ;COPY PROCESS PCB ADDRESS
             60 A1   50   D1  0081   228 10$:     CMPL    R0,PCB$L_PID(R1)                ;PROCESS ID MATCH?
                     2B   13  0085   229          BEQL    70$                             ;IF EQL YES
             51    1C A1   3C  0087   230          MOVZWL  PCB$L_OWNER(R1),R1              ;GET CREATOR PROCESS INDEX
                     0A   13  008B   231          BEQL    20$                             ;IF EQL NO CREATOR
          51   00000000'FF41   D0  008D   232          MOVL    @L^SCH$GL_PCBVEC[R1],R1         ;GET ADDRESS OF CREATOR PCB
                     EA   11  0095   233          BRB     10$                             ;
           50    0840 8F   3C  0097   234 20$:     MOVZWL  #SS$_DEVALLOC,R0                ;SET DEVICE ALREADY ALLOCATED
                     5B   11  009C   235 30$:     BRB     90$                             ;
                                 009E   236
                                 009E   237 ;
                                 009E   238 ; DEVICE SEARCH FAILURE
                                 009E   239 ;
                                 009E   240
           50    08F0 8F   B1  009E   241 40$:     CMPW    #SS$_NONLOCAL,R0                ;REMOTE DEVICE?
                     54   12  00A3   242          BNEQ    90$                             ;IF NEQ NO
                  005D   31  00A5   243          BRW     REMOTE                          ;
                                 00A8   244
                                 00A8   245 ;
                                 00A8   246 ; DEVICE NOT SPOOLED OR ALLOCATED - IF IT'S ALSO NOT SHAREABLE, CHECK THAT
                                 00A8   247 ; PROCESS HAS PRIVILEGE TO ALLOCATE IT
                                 00A8   248 ;
                                 00A8   249
    05 34 A5   00'  E0  00A8   250 50$:     BBS     S^#DEV$V_SHR,UCB$L_DEVCHAR(R5),70$ ;IF SET, DEVICE SHAREABLE
                                 00AD   251 ;
                                 00AD   252 ; NONSHAREABLE DEVICE - PERFORM IMPLICIT ALLOCATION
                                 00AD   253 ;
                                 00AD   254
    28 A5   60 A4   D0  00AD   255 60$:     MOVL    PCB$L_PID(R4),UCB$L_PID(R5) ;SET CURRENT PROCESS AS OWNER
                                 00B2   256
                                 00B2   257 ;
                                 00B2   258 ; ASSOCIATE MAILBOX IF:
                                 00B2   259 ;
                                 00B2   260 ;       1. NOT FILE DEVICE
                                 00B2   261 ;       2. NOT SHAREABLE DEVICE
                                 00B2   262 ;       3. MAILBOX NOT ALREADY ASSOCIATED
                                 00B2   263 ;       4. MAILBOX IS SPECIFIED
                                 00B2   264 ;
                                 00B2   265
    25 34 A5   00'  E0  00B2   266 70$:     BBS     S^#DEV$V_FOD,UCB$L_DEVCHAR(R5),80$ ;IF SET, FILE DEVICE
    20 34 A5   00'  E0  00B7   267          BBS     S^#DEV$V_SHR,UCB$L_DEVCHAR(R5),80$ ;IF SET, SHARED DEVICE
                56   D5  00BC   268          TSTL    R6                              ;ARE WE ASSOCIATING A MBX
                     1C   13  00BE   269          BEQL    80$                             ;IF NOT JUST CONTINUE
                54 A5   D5  00C0   270          TSTL    UCB$L_AMB(R5)                   ;IS THERE ONE CURRENTLY ASSOC?
```

```
                  0D   13  00C3          BEQL    75$                     ;IF NOT ASSOC NEW ONE
          54 A5   56   D1  00C5          CMPL    R6,UCB$L_AMB(R5)        ;TRYING TO ASSOC DIFFERENT MBX?
                  11   13  00C9          BEQL    80$                     ;IF NOT JUST CONTINUE
       50   02C4 8F    3C  00CB          MOVZWL  #SS$_DEVACTIVE,R0       ;DON'T DO THE ASSIGN
                  27   11  00D0          BRB     90$                     ;RETURN THE ERROR
                           00D2  75$:
          54 A5   56   D0  00D2          MOVL    R6,UCB$L_AMB(R5)        ;SET ASSOCIATED MAILBOX UCB ADDRESS
             50 A6     B6  00D6          INCW    UCB$W_REFC(R6)          ;INCREMENT MAILBOX UCB REFERENCE COUNT
             56   01   9A  00D9          MOVZBL  #CCB$M_AMB,R6           ;SET ASSOCIATED MAILBOX FLAG
   50   00000000'FF47   9E  00DC  80$:    MOVAB   @CTL$GL_CCBBASE[R7],R0  ;COMPUTE BASE OF CCB
             60   55   D0  00E4          MOVL    R5,CCB$L_UCB(R0)        ;STORE UCB ADDRESS IN CCB
             50 A5     B6  00E7          INCW    UCB$W_REFC(R5)          ;INCREMENT UCB REFERENCE COUNT
      09 A0  58   01   81  00EA          ADDB3   #1,R8,CCB$B_AMOD(R0)    ;STORE ACCESS MODE OF CHANNEL
          08 A0   56   90  00EF          MOVB    R6,CCB$B_STS(R0)        ;SET CHANNEL STATUS FLAGS
             6B   57   AE  00F3          MNEGW   R7,(R11)                ;STORE ASSIGNED CHANNEL NUMBER
             50   01   3C  00F6          MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION STATUS
                FF04'  31  00F9  90$:    BRW     IOC$UNLOCK              ;UNLOCK I/O DATA BASE AND RETURN
                           00FC
                           00FC  ;
                           00FC  ; NETWORK ASSIGNMENT
                           00FC  ;
                           00FC
                           00FC  NETWORK:                                 ;
       50   08F0 8F    3C  00FC          MOVZWL  #SS$_NONLOCAL,R0        ; Set return to non-local
                FEFC'  30  0101          BSBW    IOC$UNLOCK              ; Unlock I/O database and return
                       04  0104          RET
                           0105          .DSABL  LSB
                           0105
                           0105  ;
                           0105  ; REMOTE DEVICE SPECIFIED
                           0105  ;
                           0105
                           0105  REMOTE:                                  ;
       50   08F0 8F    3C  0105          MOVZWL  #SS$_NONLOCAL,R0        ; Set return to non-local
                FEF3'  30  010A          BSBW    IOC$UNLOCK              ; unlock i/o data base
                       04  010D          RET
```

```
                                010E          .SBTTL  TEST IF MAILBOX SPECIFIED
                                010E  ;
                                010E  ; SUBROUTINE TO TEST IF A MAILBOX IS SPECIFIED
                                010E  ;
                                010E  ; INPUTS:
                                010E  ;
                                010E  ;       R10 = ADDRESS OF MAILBOX NAME DESCRIPTOR
                                010E  ;
                                010E  ; OUTPUTS:
                                010E  ;
                                010E  ;       R0 = SS$_NORMAL IF SPECIFIED MAILBOX EXISTS
                                010E  ;            SS$_NOSUCHDEV IF SPECIFIED MAILBOX DOES NOT EXIST
                                010E  ;            SS$_DEVNOTMBX IF SPECIFIED DEVICE IS NOT A MAILBOX
                                010E  ;       R6 = ADDRESS OF MAILBOX UCB
                                010E  ;            ZERO IF MAILBOX NOT SPECIFIED (USED AS CHANNEL STATUS FLAGS)
                                010E  ;
                                010E
                                010E  TEST_MAILBOX:                                  ;
          56    5A    D0        010E          MOVL    R10,R6                         ;SET ADDRESS OF MAILBOX NAME DESCRIPTOR
                1B    13        0111          BEQL    10$                            ;IF EQL NO NAME SPECIFIED
          51    56    D0        0113          MOVL    R6,R1                          ;COPY ADDRESS OF MAILBOX NAME DESCRIPTOR
             FEE7'    30        0116          BSBW    IOC$SEARCHDEV                  ;SEARCH FOR DEVICE
            15 50     E9        0119          BLBC    R0,20$                         ;IF LBC SEARCH ERROR
    50    0074 8F     3C        011C          MOVZWL  #SS$_DEVNOTMBX,R0              ;SET DEVICE NOT MAILBOX STATUS
  0B 34 A1    00'     E1        0121          BBC     S^#DEV$V_MBX,UCB$L_DEVCHAR(R1),20$ ;IF CLR, DEVICE NOT MAILBOX
  06 34 A1    00'     E0        0126          BBS     S^#DEV$V_NET,UCB$L_DEVCHAR(R1),20$ ;IF SET, NETWORK DEVICE
          56    51    D0        012B          MOVL    R1,R6                          ;SAVE ADDRESS OF MAILBOX UCB
          50    01    3C        012E  10$:    MOVZWL  #SS$_NORMAL,R0                 ;SET NORMAL COMPLETION STATUS
                      05        0131  20$:    RSB                                    ;
                                0132
                                0132          .END
```

```
ACMODE                          = 0000000C     D
CCB$B_AMOD                        00000009     D
CCB$B_STS                         00000008     D
CCB$C_LENGTH                      00000010     D
CCB$K_LENGTH                      00000010     D
CCB$L_DIRP                        0000000C     D
CCB$L_UCB                         00000000     D
CCB$L_WIND                        00000004     D
CCB$M_AMB                       = 00000001     D
CCB$W_IOC                         0000000A     D
CHAN                            = 00000008     D
CTL$GL_CCBBASE                    ********   X   02
DEV$V_FOD                         ********   X   02
DEV$V_MBX                         ********   X   02
DEV$V_NET                         ********   X   02
DEV$V_SHR                         ********   X   02
DEV$V_SPL                         ********   X   02
DEVNAM                          = 00000004     D
DS$AX_SOFTPCB                     ********   X   02
EXE$ASSIGN                        00000004 RG  D   02
EXE$MAXACMODE                     ********   X   02
FREECHAN                          00000056 R   D   02
IOC$FFCHAN                        ********   X   02
IOC$SEARCHDEV                     ********   X   02
IOC$UNLOCK                        ********   X   02
LOCAL                             00000078 R   D   02
MBXNAM                          = 00000010     D
NETEND                            00000004 R   D   02
NETNAM                            00000000 R   D   02
NETWORK                           000000FC R   D   02
PCB$B_ASTACT                      0000000C     D
PCB$B_ASTEN                       0000000D     D
PCB$B_PRI                         0000000B     D
PCB$B_PRIB                        0000002F     D
PCB$B_TYPE                        0000000A     D
PCB$B_WEFC                        0000002E     D
PCB$C_LENGTH                      0000008C     D
PCB$K_LENGTH                      0000008C     D
PCB$L_ARB                         00000084     D
PCB$L_ASTQBL                      00000014     D
PCB$L_ASTQFL                      00000010     D
PCB$L_EFC2P                       00000058     D
PCB$L_EFC3P                       0000005C     D
PCB$L_EFCS                        00000050     D
PCB$L_EFCU                        00000054     D
PCB$L_EFWM                        0000004C     D
PCB$L_JIB                         00000078     D
PCB$L_OWNER                       0000001C     D
PCB$L_PHD                         00000064     D
PCB$L_PHYPCB                      00000018     D
PCB$L_PID                         00000060     D
PCB$L_PQB                         0000004C     D
PCB$L_SQBL                        00000004     D
PCB$L_SQFL                        00000000     D
PCB$L_STS                         00000024     D
PCB$L_UIC                         00000088     D
PCB$L_WSSWP                       00000020     D
PCB$L_WTIME                       00000028     D
PCB$Q_PRIV                        0000007C     D
PCB$T_LNAME                       00000068     D
PCB$T_TERMINAL                    00000044     D
PCB$W_APTCNT                      00000030     D
PCB$W_ASTCNT                      00000038     D
PCB$W_BIOCNT                      0000003A     D
PCB$W_BIOLM                       0000003C     D
PCB$W_DIOCNT                      0000003E     D
PCB$W_DIOLM                       00000040     D
PCB$W_GPGCNT                      00000034     D
PCB$W_GRP                         0000008A     D
PCB$W_MEM                         00000088     D
PCB$W_MTXCNT                      0000000E     D
PCB$W_PPGCNT                      00000036     D
PCB$W_PRCCNT                      00000042     D
PCB$W_SIZE                        00000008     D
PCB$W_STATE                       0000002C     D
PCB$W_TMBU                        00000032     D
REMOTE                            00000105 R   D   02
SCH$GL_PCBVEC                     ********   X   02
SIZ...                          = 00000002     D
SS$_ACCVIO                      = 0000000C     D
SS$_DEVACTIVE                   = 000002C4     D
SS$_DEVALLOC                    = 00000840     D
SS$_DEVNOTMBX                   = 00000074     D
SS$_IVDEVNAM                    = 00000144     D
SS$_NONLOCAL                    = 000008F0     D
SS$_NORMAL                      = 00000001     D
TEST_MAILBOX                      0000010E R   D   02
UCB$B_AMOD                        00000053     D
UCB$B_CEX                         00000077     D
UCB$B_CM1                         0000004A     D
UCB$B_CM2                         0000004B     D
UCB$B_DEVCLASS                    00000038     D
UCB$B_DEVTYPE                     00000039     D
UCB$B_DIPL                        00000052     D
UCB$B_DX_SCTCNT                   000000A6     D
UCB$B_ERTCNT                      00000070     D
UCB$B_ERTMAX                      00000071     D
UCB$B_FEX                         00000076     D
UCB$B_FIPL                        0000000B     D
UCB$B_LOCSRV                      0000003C     D
UCB$B_OFFNDX                      00000094     D
UCB$B_OFFRTC                      00000095     D
UCB$B_REMSRV                      0000003D     D
UCB$B_SECTORS                     0000003C     D
UCB$B_SLAVE                       00000074     D
UCB$B_SPR                         00000075     D
UCB$B_STATE                       00000052     D
UCB$B_TRACKS                      0000003D     D
UCB$B_TT_CRFILL                   0000009D     D
UCB$B_TT_DECRF                    000000A1     D
UCB$B_TT_DELFF                    000000A2     D
UCB$B_TT_DESPEE                   000000A0     D
UCB$B_TT_DETYPE                   000000A4     D
UCB$B_TT_LFFILL                   0000009E     D
```

UCB$B_TT_SPEED	0000009C	D	UCB$V_TEMPLATE	=	0000000D	D
UCB$B_TYPE	0000000A	D	UCB$W_BCNT		0000006E	D
UCB$B_VERTSZ	0000003F	D	UCB$W_BCR		00000096	D
UCB$C_LENGTH	00000074	D	UCB$W_BOFF		0000006C	D
UCB$C_MB_LENGTH	00000090	D	UCB$W_BUFQUO		00000018	D
UCB$C_TT_LENGTH	000000BC	D	UCB$W_BYTESTOGO		0000003E	D
UCB$K_LENGTH	00000074	D	UCB$W_CHARGE		0000004A	D
UCB$K_MB_LENGTH	00000090	D	UCB$W_CYLINDERS		0000003E	D
UCB$K_TT_LENGTH	000000BC	D	UCB$W_DA		0000008C	D
UCB$L_AMB	00000054	D	UCB$W_DC		0000008E	D
UCB$L_ASTQBL	00000010	D	UCB$W_DEVBUFSIZ		0000003A	D
UCB$L_ASTQFL	0000000C	D	UCB$W_DEVSTS		0000005A	D
UCB$L_CPID	0000005C	D	UCB$W_DIRSEQ		00J00088	D
UCB$L_CRB	00000020	D	UCB$W_DSTADDR		00000018	D
UCB$L_DDB	00000024	D	UCB$W_DX_BCR		000000A4	D
UCB$L_DEVCHAR	00000034	D	UCB$W_EC1		00000090	D
UCB$L_DEVDEPEND	0000003C	D	UCB$W_EC2		00000092	D
UCB$L_DPC	00000080	D	UCB$W_ERRCNT		00000072	D
UCB$L_DUETIM	0000005C	D	UCB$W_FUNC		0000007E	D
UCB$L_DX_BFPNT	0000009C	D	UCB$W_MB_SEED		00000000	D
UCB$L_DX_BUF	00000098	D	UCB$W_MSGCNT		00000016	D
UCB$L_DX_RXDB	000000A0	D	UCB$W_MSGMAX		00000014	D
UCB$L_EMB	00000078	D	UCB$W_NT_CHAN		0000007C	D
UCB$L_FIRST	00000014	D	UCB$W_OFFSET		0000008A	D
UCB$L_FPC	0000000C	D	UCB$W_REFC		00000050	D
UCB$L_FQBL	00000004	D	UCB$W_SIZE		00000008	D
UCB$L_FQFL	00000000	D	UCB$W_SRCADDR		0000001A	D
UCB$L_FR3	00000010	D	UCB$W_STS		00000058	D
UCB$L_FR4	00000014	D	UCB$W_TT_DESIZE		000000A5	D
UCB$L_IOQBL	00000044	D	UCB$W_UNIT		00000048	D
UCB$L_IOQFL	00000040	D	UCB$W_VPROT		0000001A	D
UCB$L_IRP	0000004C	D				
UCB$L_LINK	0000002C	D				
UCB$L_LOGADR	00000064	D				
UCB$L_MAXBLOCK	00000084	D				
UCB$L_MB_MBX	0000007C	D				
UCB$L_MB_PORT	0000008C	D				
UCB$L_MB_RAST	00000078	D				
UCB$L_MB_SHB	00000080	D				
UCB$L_MB_WAST	00000074	D				
UCB$L_MB_WIOQBL	00000088	D				
UCB$L_MB_WIOQFL	00000084	D				
UCB$L_MEDIA	0000008C	D				
UCB$L_NT_DATSSB	00000074	D				
UCB$L_NT_INTSSB	00000078	D				
UCB$L_OPCNT	00000060	D				
UCB$L_OWNUIC	0000001C	D				
UCB$L_PID	00000028	D				
UCB$L_RQBL	00000004	D				
UCB$L_RQFL	00000000	D				
UCB$L_SVAPTE	00000068	D				
UCB$L_SVPN	00000064	D				
UCB$L_TT_DECHAR	000000A8	D				
UCB$L_TT_RDUE	0000008C	D				
UCB$L_TT_RTIMOU	000000B8	D				
UCB$L_VCB	00000030	D				
UCB$T_PARTNER	0000000C	D				

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000132 (306.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL                          VALUE        DEFINITION      REFERENCES...
------                          -----        ----------      -------------
ACMODE                         =0000000C     95     (1)       188     (1)
CCB$B_AMOD                      00000009                    #-283    (1)
CCB$B_STS                       00000008                    #-284    (1)
CCB$L_UCB                       00000000                    #-281    (1)
CCB$M_AMB                      =00000001                    #-279    (1)
CHAN                           =00000008     94     (1)     #-176    (1)
CTL$GL_CCBBASE                  00000000-XR                  280     (1)
DEV$V_FOD                       00000000-XR                 #-266    (1)
DEV$V_MBX                       00000000-XR                 #-332    (1)
DEV$V_NET                       00000000-XR                 #-333    (1)
DEV$V_SHR                       00000000-XR                 #-250    (1)     #-267    (1)
DEV$V_SPL                       00000000-XR                 #-215    (1)
DEVNAM                         =00000004     93     (1)     #-185    (1)
DS$AX_SOFTPCB                   00000000-XR                  175     (1)
EXE$ASSIGN                      00000004-R   174     (1)
EXE$MAXACMODE                   00000000-XR                  189     (1)
FREECHAN                        00000056-R   202     (1)     #-192    (1)
IOC$FFCHAN                      00000000-XR                 #-191    (1)
IOC$SEARCHDEV                   00000000-XR                 #-207    (1)     #-329    (1)
IOC$UNLOCK                      00000000-XR                 #-287    (1)     #-295    (1)     #-305    (1)
LOCAL                           00000078-R   224     (1)     #-217    (1)
MBXNAM                         =00000010     96     (1)     #-179    (1)
NETEND                          00000004-R   104     (1)
NETNAM                          00000000-R   103     (1)
NETWORK                         000000FC-R   293     (1)     #-218    (1)
PCB$L_OWNER                     0000001C                    #-230    (1)
PCB$L_PID                       00000060                    #-228    (1)     #-255    (1)
REMOTE                          00000105-R   303     (1)     #-243    (1)
SCH$GL_PCBVEC                   00000000-XR                 #-232    (1)
SS$_ACCVIO                     =0000000C                    #-194    (1)
SS$_DEVACTIVE                  =000002C4                    #-274    (1)
SS$_DEVALLOC                   =00000840                    #-234    (1)
SS$_DEVNOTMBX                  =00000074                    #-331    (1)
SS$_IVDEVNAM                   =00000144                    #-184    (1)
SS$_NONLOCAL                   =000008F0                    #-241    (1)     #-294    (1)     #-304    (1)
SS$_NORMAL                     =00000001                    #-286    (1)     #-335    (1)
TEST_MAILBOX                    0000010E-R   325     (1)     #-204    (1)
UCB$L_AMB                       00000054                    #-270    (1)     #-272    (1)     #-277    (1)
UCB$L_DEVCHAR                   00000034                      215     (1)       250    (1)       266    (1)       267    (1)
                                                              332     (1)       333    (1)
UCB$L_PID                       00000028                    #-225    (1)     #-255    (1)
UCB$V_TEMPLATE                 =0000000D                    #-216    (1)
UCB$W_REFC                      00000050                    #-278    (1)     #-282    (1)
UCB$W_STS                       00000058                      217     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CCBDEF	1	77 (1)	77 (1)
$DEFINI	1	77 (1)	77 (1) 78 (1) 79 (1) 80 (1) 81 (1) 82 (1) 83 (1) 84 (1) 85 (1)
$IODEF	15	78 (1)	78 (1)
$JIBDEF	3	79 (1)	79 (1)
$LOGDEF	1	80 (1)	80 (1)
$PCBDEF	4	81 (1)	81 (1)
$PRDEF	5	82 (1)	82 (1)
$PRVDEF	4	83 (1)	83 (1)
$SSDEF	21	84 (1)	84 (1)
$UCBDEF	10	85 (1)	85 (1)
IFNORD	1	181 (1)	181 (1) 187 (1)
IFNOWRT	1	177 (1)	177 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDB3	0081	283	(1)												
BBC	00E1	216	(1)	332	(1)										
BBS	00E0	215	(1)	250	(1)	266	(1)	267	(1)	333	(1)				
BEQL	0013	177	(1)	180	(1)	181	(1)	186	(1)	187	(1)	226	(1)	229	(1)
		231	(1)	269	(1)	271	(1)	273	(1)	327	(1)				
BLBC	00E9	205	(1)	208	(1)	330	(1)								
BLBS	00E8	192	(1)												
BNEQ	0012	242	(1)												
BRB	0011	233	(1)	235	(1)	275	(1)								
BRW	0031	218	(1)	243	(1)	287	(1)								
BSBW	0030	191	(1)	204	(1)	207	(1)	295	(1)	305	(1)	329	(1)		
CLRW	00B4	178	(1)												
CMPL	00D1	228	(1)	272	(1)										
CMPW	00B1	241	(1)												
EXTZV	00EF	188	(1)												
INCW	00B6	278	(1)	282	(1)										
JSB	0016	189	(1)												
MNEGW	00AE	285	(1)												
MOVAB	009E	175	(1)	280	(1)										
MOVB	0090	284	(1)												
MOVL	00D0	176	(1)	179	(1)	183	(1)	185	(1)	190	(1)	203	(1)	206	(1)
		214	(1)	225	(1)	227	(1)	232	(1)	255	(1)	277	(1)	281	(1)
		326	(1)	328	(1)	334	(1)								
MOVQ	007D	182	(1)												
MOVZBL	009A	279	(1)												
MOVZWL	003C	184	(1)	194	(1)	230	(1)	234	(1)	274	(1)	286	(1)	294	(1)
		304	(1)	331	(1)	335	(1)								
PROBER	000C	181	(1)	187	(1)										
PROBEW	000D	177	(1)												
RET	0004	193	(1)	195	(1)	296	(1)	306	(1)						
RSB	0005	336	(1)												
TSTL	00D5	268	(1)	270	(1)										

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ASCII          103     (1)
.DSABL          297     (1)
.ENABL          201     (1)
.END            338     (1)
.ENDM           177     (1)     181     (1)     77      (1)     78      (1)     79      (1)     80      (1)     81      (1)     82      (1)
                83      (1)     84      (1)     85      (1)
.ENTRY          174     (1)
.IDENT          7       (1)
.NOCROSS        77      (1)     78      (1)     79      (1)     80      (1)     81      (1)     82      (1)     83      (1)     84      (1)
                85      (1)
.PAGE           105     (1)     307     (1)
.PSECT          102     (1)
.RESTORE        77      (1)     78      (1)     79      (1)     80      (1)     81      (1)     82      (1)     83      (1)     84      (1)
                85      (1)
.SAVE           77      (1)     78      (1)     79      (1)     80      (1)     81      (1)     82      (1)     83      (1)     84      (1)
                85      (1)
.SBTTL          106     (1)     308     (1)
.TITLE          6       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	4	#-176 (1) #-179 (1) #-185 (1) 188 (1)
R0	22	#-184 (1) #-188 (1) #-190 (1) #-192 (1) #-194 (1) #-205 (1)
		#-208 (1) #-225 (1) #-228 (1) #-234 (1) #-241 (1) #-274 (1)
		#-280 (1) #-281 (1) #-283 (1) #-284 (1) #-286 (1) #-294 (1)
		#-304 (1) #-330 (1) #-331 (1) #-335 (1)
R1	13	#-203 (1) #-206 (1) #-214 (1) #-227 (1) #-228 (1) #-230 (1)
		#-232 (1) #-328 (1) 332 (1) 333 (1) #-334 (1)
R10	6	174 (1) #-179 (1) 181 (1) #-182 (1) #-183 (1) #-326 (1)
R11	5	174 (1) #-176 (1) 177 (1) #-178 (1) #-285 (1)
R2	1	174 (1)
R3	1	174 (1)
R4	4	174 (1) #-175 (1) #-227 (1) #-255 (1)
R5	14	174 (1) #-214 (1) 215 (1) 217 (1) #-225 (1) 250 (1)
		#-255 (1) 266 (1) 267 (1) #-270 (1) #-272 (1) #-277 (1)
		#-281 (1) #-282 (1)
R6	10	174 (1) #-268 (1) #-272 (1) #-277 (1) #-278 (1) #-279 (1)
		#-284 (1) #-326 (1) #-328 (1) #-334 (1)
R7	4	174 (1) #-203 (1) 280 (1) #-285 (1)
R8	3	174 (1) #-190 (1) #-283 (1)
R9	4	174 (1) #-185 (1) 187 (1) #-206 (1)
SP	2	#-182 (1) #-183 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.03	00:00:00.23
Command processing	885	00:00:00.24	00:00:00.85
Pass 1	597	00:00:02.66	00:00:03.96
Symbol table sort	0	00:00:00.36	00:00:00.36
Pass 2	10	00:00:00.77	00:00:01.19
Symbol table output	0	00:00:00.03	00:00:00.09
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	2	00:00:00.07	00:00:00.16
Assembler run totals	1523	00:00:04.17	00:00:06.84

The working set limit was 3512 pages.
155496 bytes (304 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1254 non-local and 15 local symbols.
338 source lines were read in Pass 1, producing 82 object records in Pass 2.
57 pages of virtual memory were used to define 20 macros.

```
                              +---------------------------+
                              ! Macro library statistics !
                              +---------------------------+

Macro library name                                 Macros defined
------------------                                 --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8                     3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                       0
SYS$COMMON:[SYSLIB]LIB.MLB;2                                  4
SYS$COMMON:[SYSLIB]STARLET.MLB;2                              9
TOTALS (all libraries)                                       16

1508 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ASSIGN.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 44 HISTORY ; DETAILED
(1) 60 DECLARATIONS
(1) 89 EXE$SETAST - SET AST ENABLES

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ASTCON.MAR
                        0000     2 ; *3    12-MAY-1986 10:13:00 BARANSKI "RESERVE ASTDEL"
                        0000     3 ; *2     7-MAY-1986 14:11:27 BARANSKI "Documentaion Check."
                        0000     4 ; *1     6-FEB-1986 15:13:35 DS ""
                        0000     5 ; DEC/CMS REPLACEMENT HISTORY, Element ASTCON.MAR
                        0000     6         .TITLE  ASTCON  *** ASTCON AST control service
                        0000     7         .IDENT  "7.0-2"                                                ;G:2
                        0000     8
                        0000     9 ;
                        0000    10 ;*****************************************************************************
                        0000    11 ;*                                                                           *
                        0000    12 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980, 1981                               * ;G:2
                        0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000    14 ;*                                                                           *
                        0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        0000    20 ;*  TRANSFERRED.                                                            *
                        0000    21 ;*                                                                           *
                        0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000    24 ;*  CORPORATION.                                                            *
                        0000    25 ;*                                                                           *
                        0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000    28 ;*                                                                           *
                        0000    29 ;*****************************************************************************
                        0000    30
                        0000    31 ;++
                        0000    32 ; FACILITY: EXECUTIVE, SYSTEM SERVICES
                        0000    33 ;
                        0000    34 ; ABSTRACT:
                        0000    35 ;       THIS MODULE CONTAINS THE SYSTEM SERVICES SETAST WHICH CONTROLS
                        0000    36 ;       THE STATE OF THE AST ENABLES AND DCLAST WHICH DECLARES AN
                        0000    37 ;       AST FOR A PARTICULAR ACCESS MODE.
                        0000    38 ;
                        0000    39 ; ENVIRONMENT:
                        0000    40 ;
                        0000    41 ;--
                        0000    42 ;
                        0000    43 ;       .PAGE
                        0000    44         .SBTTL  HISTORY                 ; DETAILED
                        0000    45 ;
                        0000    46 ; AUTHOR:       R. HUSTVEDT      CREATION DATE: 21-SEP-76
                        0000    47 ;
                        0000    48 ; MODIFIED BY:
                        0000    49 ;
                        0000    50 ;               Dave Butenhof   18-jun-1980,    version 5.5
                        0000    51 ;       01      Use VMS SYSASTCON module to implement supervisor SETAST
                        0000    52 ;               service.
                        0000    53 ;
                        0000    54 ;       02      - Jack Stansbury, 23-Oct-1981, Version 7.0              ;G:2
                        0000    55 ;               Fixed some truncation errors. Added .LIBRARY statements
                        0000    56 ;               for $DS and $DIAG. Removed previous .PSECT SEP statement,
                        0000    57 ;               changed it to .PSECT CODE.
```

```
                0000     58
```

```
                        0000     60          .SBTTL  DECLARATIONS
                        0000     61
                        0000     62 ;
                        0000     63 ; INCLUDE FILES:
                        0000     64 ;
                        0000     65          .LIBRARY        /$DS/           ;                                       [02]
                        0000     66          .LIBRARY        /$DIAG/         ;                                       [02]
                        0000     67                          ;                                                       [02]
                        0000     68          $ACBDEF                         ; DEFINE AST CONTROL BLOCK OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,ACB$L_ASTQFL,        ACB$L_ASTQFL:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,ACB$L_ASTQBL,        ACB$L_ASTQBL:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,ACB$W_SIZE,  ACB$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,ACB$B_TYPE,  ACB$B_TYPE:
                0000000B 000A                        .BLKB   1
                        000B                 .IIF    NB,ACB$B_RMOD,  ACB$B_RMOD:
                0000000C 000B                        .BLKB   1
                        000C                 .IIF    NB,ACB$L_PID,   ACB$L_PID:
                00000010 000C                        .BLKL   1
                        0010                 .IIF    NB,ACB$L_AST,   ACB$L_AST:
                00000014 0010                        .BLKL   1
                        0014                 .IIF    NB,ACB$L_ASTPRM,        ACB$L_ASTPRM:
                00000018 0014                        .BLKL   1
                        0018                 .IIF    NB,ACB$C_LENGTH,        ACB$C_LENGTH:
                        0018                 .IIF    NB,ACB$K_LENGTH,        ACB$K_LENGTH:
                        0018                 .IIF    NB,ACB$L_KAST,  ACB$L_KAST:
                0000001C 0018                        .BLKL   1
                        0000                 .RESTORE
                        0000     69          $IPLDEF                         ; DEFINE IPL VALUES
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 001C                .=0
                        0000                 .RESTORE
                        0000     70          $PCBDEF                         ; DEFINE PCB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 001C                .=0
                        0000                 .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
                0000000B 000A                        .BLKB   1
                        000B                 .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
                0000000C 000B                        .BLKB   1
                        000C                 .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
                0000000D 000C                        .BLKB   1
                        000D                 .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
                0000000E 000D                        .BLKB   1
                        000E                 .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
                00000010 000E                        .BLKW   1
                        0010                 .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
```

```
                        00000014  0010                    .BLKL   1
                                  0014            .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
                        00000018  0014                    .BLKL   1
                                  0018            .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
                        0000001C  0018                    .BLKL   1
                                  001C            .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
                        00000020  001C                    .BLKL   1
                                  0020            .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
                        00000024  0020                    .BLKL   1
                                  0024            .IIF    NB,PCB$L_STS,   PCB$L_STS:
                        00000028  0024                    .BLKL   1
                                  0028            .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
                        0000002C  0028                    .BLKL   1
                                  002C            .IIF    NB,PCB$W_STATE, PCB$W_STATE:
                        0000002E  002C                    .BLKW   1
                                  002E            .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
                        0000002F  002E                    .BLKB   1
                                  002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
                        00000030  002F                    .BLKB   1
                                  0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
                        00000032  0030                    .BLKW   1
                                  0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
                        00000034  0032                    .BLKW   1
                                  0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
                        00000036  0034                    .BLKW   1
                                  0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
                        00000038  0036                    .BLKW   1
                                  0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
                        0000003A  0038                    .BLKW   1
                                  003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
                        0000003C  003A                    .BLKW   1
                                  003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
                        0000003E  003C                    .BLKW   1
                                  003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
                        00000040  003E                    .BLKW   1
                                  0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
                        00000042  0040                    .BLKW   1
                                  0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
                        00000044  0042                    .BLKW   1
                                  0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
                        0000004C  0044                    .BLKB   8
                                  004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                                  004C            .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
                        00000050  004C                    .BLKL   1
                                  0050            .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
                        00000054  0050                    .BLKL   1
                                  0054            .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
                        00000058  0054                    .BLKL   1
                                  0058            .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
                        0000005C  0058                    .BLKL   1
                                  005C            .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
                        00000060  005C                    .BLKL   1
                                  0060            .IIF    NB,PCB$L_PID,   PCB$L_PID:
                        00000064  0060                    .BLKL   1
                                  0064            .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
                        00000068  0064                    .BLKL   1
                                  0068            .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
```

```
                00000078  0068                        .BLKB   16
                          0078              .IIF      NB,PCB$L_JIB,   PCB$L_JIB:
                0000007C  0078                        .BLKL   1
                          007C              .IIF      NB,PCB$Q_PRIV,  PCB$Q_PRIV:
                00000084  007C                        .BLKQ   1
                          0084              .IIF      NB,PCB$L_ARB,   PCB$L_ARB:
                00000088  0084                        .BLKL   1
                          0088              .IIF      NB,PCB$L_UIC,   PCB$L_UIC:
                          0088              .IIF      NB,PCB$W_MEM,   PCB$W_MEM:
                0000008A  0088                        .BLKW   1
                          008A              .IIF      NB,PCB$W_GRP,   PCB$W_GRP:
                0000008C  008A                        .BLKW   1
                          008C              .IIF      NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                          008C              .IIF      NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                          0000              .RESTORE
                          0000      71      $PRDEF                            ; DEFINE PROCESSOR REGISTER VALUES
                          0000              .SAVE     LOCAL_BLOCK
                          0000              .PSECT    $ABS$,ABS
                00000000  008C              .=0
                          0000              .RESTORE
                          0000      72      $PSLDEF                           ; DEFINE PSL FIELDS
                          0000              .SAVE     LOCAL_BLOCK
                          0000              .PSECT    $ABS$,ABS
                00000000  008C              .=0
                          0000              .RESTORE
                          0000      73      $RSNDEF                           ; DEFINE RESOURCE NUMBERS
                          0000              .SAVE     LOCAL_BLOCK
                          0000              .PSECT    $ABS$,ABS
                00000000  008C              .=0
                          0000              .RESTORE
                          0000      74      $SSDEF                            ; DEFINE STATUS CODE VALUES
                          0000              .SAVE     LOCAL_BLOCK
                          0000              .PSECT    $ABS$,ABS
                00000000  008C              .=0
                          0000              .RESTORE
                          0000      75
                          0000      76 ;
                          0000      77 ; EQUATES:
                          0000      78 ;
                          0000      79
                0000000C  0000      80 ACMODE=12                              ; DISPLACEMENT TO ACCESS MODE
                00000004  0000      81 AST=4                                  ; DISPLACEMENT TO AST ADDRESS
                00000008  0000      82 ASTPRM=8                               ; DISPLACEMENT TO AST PARAMETER
                00000004  0000      83 ENABLE=4                               ; DISPLACEMENT TO ENABLE ARGUMENT
                          0000      84 ;
                          0000      85 ; OWN STORAGE:
                          0000      86 ;
                      00000000      87          .PSECT    Y$EXEPAGED,BYTE     ; PAGED PSECT
```

```
                                  0000          .SBTTL  EXE$SETAST - SET AST ENABLES
                              00000000          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                [02]
                                  0000
                                  0000  ;++
                                  0000  ; FUNCTIONAL DESCRIPTION:
                                  0000  ;       EXE$SETAST ALLOWS A PROCESS TO ENABLE OR DISABLE THE DELIVERY
                                  0000  ;       OF ASTS FOR THE PROCESS AT THE ACCESS MODE ISSUING THE CALL TO
                                  0000  ;       EXE$SETAST.   THE PREVIOUS STATE OF THE AST ENABLE FOR THAT
                                  0000  ;       ACCESS MODE IS RETURNED IN BIT 1 OF REGISTER 0 AS PART OF
                                  0000  ;       THE SUCCESSFUL COMPLETION STATUS CODE.
                                  0000  ;
                                  0000  ; CALLING SEQUENCE:
                                  0000  ;       CALLG   ARGLIST,EXE$SETAST
                                  0000  ;
                                  0000  ; INPUT PARAMETERS:
                                  0000  ;       04(AP) - NEW SETTING FOR AST ENABLE
                                  0000  ;       R4 - PCB ADDRESS OF CURRENT PROCESS
                                  0000  ;
                                  0000  ; IMPLICIT INPUTS:
                                  0000  ;       SCH$GL_CURPCB - POINTER TO CURRENT PROCESS PCB
                                  0000  ;
                                  0000  ; OUTPUT PARAMETERS:
                                  0000  ;       R0 - COMPLETION STATUS CODE
                                  0000  ;
                                  0000  ; IMPLICIT OUTPUTS:
                                  0000  ;       AST ENABLE FOR PREVIOUS MODE IS SET ACCORDING TO ARGUMENT
                                  0000  ;
                                  0000  ;       THE NEW SETTING FOR THE ASTLVL REGISTER IS PLACED BOTH IN
                                  0000  ;       THE PROCESSOR REGISTER AND THE CURRENT HARDWARE PCB.
                                  0000  ;
                                  0000  ; COMPLETION CODES:
                                  0000  ;       SS$_WASCLR  -  AST ENABLE FOR ACCESS MODE WAS CLEAR
                                  0000  ;       SS$_WASSET  -  AST ENABLE FOR ACCESS MODE WAS SET
                                  0000  ;
                                  0000  ; SIDE EFFECTS:
                                  0000  ;       ANY PENDING ASTS FOR THIS PROCESS WHICH BECOME DELIVERABLE
                                  0000  ;       AS A RESULT OF SETTING AN AST ENABLE MAY BE DELIVERED UPON
                                  0000  ;       EXIT FROM THIS SERVICE.
                                  0000  ;
                                  0000  ;--
                                  0000
                            003C  0000          .ENTRY  EXE$SETAST,^M<R2,R3,R4,R5>
      54   00000000'EF       DE  0002          MOVAL   DS$AX_SOFTPCB, R4          ; Get PCB address
                   51       DC  0009          MOVPSL  R1                         ; GET CURRENT PSL
   51   51   02   18       EF  000B          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1 ; Extract current mode
             55   09       3C  0010          MOVZWL  #SS$_WASSET,R5             ; ASSUME SET
      03 0D A4   51       E0  0013          BBS     R1,PCB$B_ASTEN(R4),10$     ; TEST OLD ENABLE SETTING
             55   01       3C  0018          MOVZWL  #SS$_WASCLR,R5             ; IT WAS CLEAR
0D A4   01   51   04 AC    F0  001B  10$:    INSV    ENABLE(AP),R1,#1,PCB$B_ASTEN(R4)  ; SET NEW ENABLE VALUE
         00000000'EF       16  0022          JSB     SCH$NEWLVL                 ;;; COMPUTE NEW ASTLVL                    [02]
             50   55       D0  0028          MOVL    R5,R0                      ; SET COMPLETION STATUS
                          04  002B          RET                                ; AND RETURN TO CALLER
                                  002C          .END
```

```
ACB$B_RMOD                      0000000B    D
ACB$B_TYPE                      0000000A    D
ACB$C_LENGTH                    00000018    D
ACB$K_LENGTH                    00000018    D
ACB$L_AST                       00000010    D
ACB$L_ASTPRM                    00000014    D
ACB$L_ASTQBL                    00000004    D
ACB$L_ASTQFL                    00000000    D
ACB$L_KAST                      00000018    D
ACB$L_PID                       0000000C    D
ACB$W_SIZE                      00000008    D
ACMODE                        = 0000000C    D
AST                           = 00000004    D
ASTPRM                        = 00000008    D
DS$AX_SOFTPCB                   ********  X   03
ENABLE                        = 00000004    D
EXE$SETAST                      00000000 RG D   03
PCB$B_ASTACT                    0000000C    D
PCB$B_ASTEN                     0000000D    D
PCB$B_PRI                       0000000B    D
PCB$B_PRIB                      0000002F    D
PCB$B_TYPE                      0000000A    D
PCB$B_WEFC                      0000002E    D
PCB$C_LENGTH                    0000008C    D
PCB$K_LENGTH                    0000008C    D
PCB$L_ARB                       00000084    D
PCB$L_ASTQBL                    00000014    D
PCB$L_ASTQFL                    00000010    D
PCB$L_EFC2P                     00000058    D
PCB$L_EFC3P                     0000005C    D
PCB$L_EFCS                      00000050    D
PCB$L_EFCU                      00000054    D
PCB$L_EFWM                      0000004C    D
PCB$L_JIB                       00000078    D
PCB$L_OWNER                     0000001C    D
PCB$L_PHD                       00000064    D
PCB$L_PHYPCB                    00000018    D
PCB$L_PID                       00000060    D
PCB$L_PQB                       0000004C    D
PCB$L_SQBL                      00000004    D
PCB$L_SQFL                      00000000    D
PCB$L_STS                       00000024    D
PCB$L_UIC                       00000088    D
PCB$L_WSSWP                     00000020    D
PCB$L_WTIME                     00000028    D
PCB$Q_PRIV                      0000007C    D
PCB$T_LNAME                     00000068    D
PCB$T_TERMINAL                  00000044    D
PCB$W_APTCNT                    00000030    D
PCB$W_ASTCNT                    00000038    D
PCB$W_BIOCNT                    0000003A    D
PCB$W_BIOLM                     0000003C    D
PCB$W_DIOCNT                    0000003E    D
PCB$W_DIOLM                     00000040    D
PCB$W_GPGCNT                    00000034    D
PCB$W_GRP                       0000008A    D
PCB$W_MEM                       00000088    D
PCB$W_MTXCNT                    0000000E    D
PCB$W_PPGCNT                    00000036    D
PCB$W_PRCCNT                    00000042    D
PCB$W_SIZE                      00000008    D
PCB$W_STATE                     0000002C    D
PCB$W_TMBU                      00000032    D
PSL$S_CURMOD                  = 00000002    D
PSL$V_CURMOD                  = 00000018    D
SCH$NEWLVL                      ********  X   03
SIZ...                        = 00000001    D
SS$_WASCLR                    = 00000001    D
SS$_WASSET                    = 00000009    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000008C (140.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	00000000 (0.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	0000002C (44.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACMODE	=0000000C	80 (1)	
AST	=00000004	81 (1)	
ASTPRM	=00000008	82 (1)	
DS$AX_SOFTPCB	00000000-XR		131 (1)
ENABLE	=00000004	83 (1)	#-137 (1)
EXE$SETAST	00000000-R	130 (1)	
PCB$B_ASTEN	0000000D		135 (1) 137 (1)
PSL$S_CURMOD	=00000002		#-133 (1)
PSL$V_CURMOD	=00000018		#-133 (1)
SCH$NEWLVL	00000000-XR		138 (1)
SS$_WASCLR	=00000001		#-136 (1)
SS$_WASSET	=00000009		#-134 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	68 (1)	68 (1)
$DEFINI	1	68 (1)	68 (1) 69 (1) 70 (1) 71 (1) 72 (1) 73 (1) 74 (1)
$IPLDEF	1	69 (1)	69 (1)
$PCBDEF	4	70 (1)	70 (1)
$PRDEF	5	71 (1)	71 (1)
$PSLDEF	2	72 (1)	72 (1)
$RSNDEF	1	73 (1)	73 (1)
$SSDEF	21	74 (1)	74 (1)

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
BBS             00E0      135     (1)
EXTZV           00EF      133     (1)
INSV            00F0      137     (1)
JSB             0016      138     (1)
MOVAL           00DE      131     (1)
MOVL            00D0      139     (1)
MOVPSL          00DC      132     (1)
MOVZWL          003C      134     (1)     136     (1)
RET             0004      140     (1)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.END            141     (1)
.ENDM           68      (1)     69      (1)     70      (1)     71      (1)     72      (1)     73      (1)     74      (1)
.ENTRY          130     (1)
.IDENT          7       (1)
.LIBRARY        65      (1)     66      (1)
.NOCROSS        68      (1)     69      (1)     70      (1)     71      (1)     72      (1)     73      (1)     74      (1)
.PAGE           59      (1)     88      (1)
.PSECT          87      (1)     90      (1)
.RESTORE        68      (1)     69      (1)     70      (1)     71      (1)     72      (1)     73      (1)     74      (1)
.SAVE           68      (1)     69      (1)     70      (1)     71      (1)     72      (1)     73      (1)     74      (1)
.SBTTL          44      (1)     60      (1)     89      (1)
.TITLE          6       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...							
AP	1	#-137	(1)						
R0	1	#-139	(1)						
R1	5	#-132	(1)	133	(1)	#-135	(1)	#-137	(1)
R2	1	130	(1)						
R3	1	130	(1)						
R4	4	130	(1)	#-131	(1)	135	(1)	137	(1)
R5	4	130	(1)	#-134	(1)	#-136	(1)	#-139	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.04	00:00:00.19
Command processing	863	00:00:00.21	00:00:00.78
Pass 1	449	00:00:01.42	00:00:02.81
Symbol table sort	0	00:00:00.19	00:00:00.19
Pass 2	8	00:00:00.40	00:00:00.64
Symbol table output	0	00:00:00.00	00:00:00.01
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	1	00:00:00.03	00:00:00.07
Assembler run totals	1343	00:00:02.30	00:00:04.70

The working set limit was 3012 pages.
82723 bytes (162 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 674 non-local and 1 local symbols.
141 source lines were read in Pass 1, producing 53 object records in Pass 2.
40 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	1
SYS$COMMON:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	11

816 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ASTCON.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 45 HISTORY ; DETAILED
(1) 94 Librarys and Equated Symbols
(1) 118 Data PSect Declarations
(1) 130 SCH$ASTDEL - AST DELIVERY INTERRUPT HANDLER
(1) 366 SCH$QAST - ENQUEUE AST CONTROL BLOCK FOR PROCESS

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ASTDEL.MAR
0000     2 ; *3    12-MAY-1986 10:14:49 BARANSKI "Cleaning up Object Libraries."
0000     3 ; *2     7-MAY-1986 14:16:44 BARANSKI "Documentaion Check."
0000     4 ; *1     6-FEB-1986 15:13:37 DS ""
0000     5 ; DEC/CMS REPLACEMENT HISTORY, Element ASTDEL.MAR
0000     6         .TITLE  ASTDEL  *** ASTDEL AST delivery
0000     7         .IDENT  "6.7-7"                                                ;G:2
0000     8         .NoShow Conditionals                ;[07]                      ;G:2
0000     9
0000    10 ;
0000    11 ;*************************************************************************
0000    12 ;*                                                                       *
0000    13 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980, 1982                           * ;G:2
0000    14 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                     *
0000    15 ;*                                                                       *
0000    16 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED *
0000    17 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE *
0000    18 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER *
0000    19 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000    20 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY *
0000    21 ;*  TRANSFERRED.                                                         *
0000    22 ;*                                                                       *
0000    23 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE *
0000    24 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT *
0000    25 ;*  CORPORATION.                                                         *
0000    26 ;*                                                                       *
0000    27 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS *
0000    28 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.              *
0000    29 ;*                                                                       *
0000    30 ;*************************************************************************
0000    31
0000    32 ;++
0000    33 ; FACILITY: EXECUTIVE, SCHEDULER
0000    34 ;
0000    35 ; ABSTRACT:
0000    36 ;       ASTDEL CONTAINS THE AST DELIVERY INTERRUPT SERVICE ROUTINE AND THE
0000    37 ;       ASSOCIATED SUBROUTINES SCH$QAST AND SCH$NEWLVL.  THESE ROUTINES
0000    38 ;       IMPLEMENT THE PRIMITIVE AST QUEUEING AND DELIVERY MECHANISMS.
0000    39 ;
0000    40 ; ENVIRONMENT:
0000    41 ;       MODE = KERNEL
0000    42 ;--
0000    43 ;
0000    44 ;       .PAGE
0000    45         .SBTTL  HISTORY                 ; DETAILED
0000    46 ;
0000    47 ; AUTHOR:       R. HUSTVEDT      CREATION DATE: 1-SEP-76
0000    48 ;
0000    49 ; MODIFIED BY:
0000    50 ;                                                                       ;G:2
0000    51 ;       V2.04   RIH0082         RICHARD I. HUSTVEDT     04-APR-1980
0000    52 ;               REMOVE ASTPEN FLAG AND REPLACE WITH ACCESS MODE CHECK
0000    53 ;
0000    54 ;       V2.03   LMK0001         LEN KAWELL      23-JAN-1980
0000    55 ;               ADD CHECKS TO AVOID FILLING THE STACK WHEN AST'S ARE
0000    56 ;               CONTINUALLY QUEUED TO A MODE BEFORE IT CAN EXECUTE
0000    57 ;               THE (SECOND) AST DELIVERY REI.
```

```
0000     58 ;
0000     59 ;       V2.02   RIH0031         R. I. HUSTVEDT  07-JUL-1979     14:35
0000     60 ;               CHECK FOR AUTOMATIC STACK EXPANSION AND EXPAND USER STACK.
0000     61 ;
0000     62 ;       V2.01   RIH0030         R. I. HUSTVEDT  20-JUL-1979     10:40
0000     63 ;               ADD GLOBAL FOR RETURN ADDRESS FROM AST CALL
0000     64 ;
0000     65 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
0000     66 ;                                                                       ;G:2
0000     67 ;               Roger Riggs, 27-Jan-1980, Version 5.2
0000     68 ;       01      Fixed so ACB is requeued to head of queue if rejected
0000     69 ;               because ast is already active on this level.
0000     70 ;                                                                       ;G:2
0000     71 ;       02      David Butenhof   06-Feb-80
0000     72 ;               Add routine EXE$SETAST to emulate VMS function of
0000     73 ;               enabling/disabling AST delivery.
0000     74 ;
0000     75 ;               Dave Butenhof   13-may-1980, Version 5.4
0000     76 ;       03      Modify VMS V2.0 source for supervisor environment
0000     77 ;                                                                       ;G:2
0000     78 ;               Dave Butenhof   18-jun-1980, version 5.5
0000     79 ;       04      Remove EXE$SETAST routine from here, to enhance VMS
0000     80 ;               compatibility AST delivery enable/disable will be
0000     81 ;               implemented as VMS does it.
0000     82 ;                                                                       ;G:2
0000     83 ;       05      Dave Butenhof, 23-feb-1981, version 6.3
0000     84 ;               Fix truncation errors
0000     85 ;
0000     86 ;       06      - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     87 ;               Changed PSECT SEP to PSECT Data and PSECT Code.
0000     88 ;               Also added .LIBRARY statements for $DS and $DIAG.
0000     89 ;
0000     90 ;       07      Jack Stansbury, 1-June-1982, Version 6.7
0000     91 ;               Fixed three truncation errors.
0000     92 ;-
```

```
                        0000     94          .Subtitle       Librarys and Equated Symbols
                        0000     95
                        0000     96 ;
                        0000     97 ; INCLUDE FILES:
                        0000     98 ;
                        0000     99          .LIBRARY        /$DS/           ;                                       [06]
                        0000    100          .LIBRARY        /$DIAG/         ;                                       [06]
                        0000    101                                          ;                                       [06]
                        0000    102          CMKDEF                          ; CHMK HOOKS AND SYMBOLS
                        0000    103          $ACBDEF                         ; AST CONTROL BLOCK DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,ACB$L_ASTQFL,        ACB$L_ASTQFL:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,ACB$L_ASTQBL,        ACB$L_ASTQBL:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,ACB$W_SIZE,  ACB$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,ACB$B_TYPE,  ACB$B_TYPE:
                0000000B 000A                        .BLKB   1
                        000B                 .IIF    NB,ACB$B_RMOD,  ACB$B_RMOD:
                0000000C 000B                        .BLKB   1
                        000C                 .IIF    NB,ACB$L_PID,   ACB$L_PID:
                00000010 000C                        .BLKL   1
                        0010                 .IIF    NB,ACB$L_AST,   ACB$L_AST:
                00000014 0010                        .BLKL   1
                        0014                 .IIF    NB,ACB$L_ASTPRM,        ACB$L_ASTPRM:
                00000018 0014                        .BLKL   1
                        0018                 .IIF    NB,ACB$C_LENGTH,        ACB$C_LENGTH:
                        0018                 .IIF    NB,ACB$K_LENGTH,        ACB$K_LENGTH:
                        0018                 .IIF    NB,ACB$L_KAST,  ACB$L_KAST:
                0000001C 0018                        .BLKL   1
                        0000                 .RESTORE
                        0000    104          $IPLDEF                         ; IPL DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 001C                .=0
                        0000                 .RESTORE
                        0000    105          $PCBDEF                         ; PCB DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 001C                .=0
                        0000                 .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
                0000000B 000A                        .BLKB   1
                        000B                 .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
                0000000C 000B                        .BLKB   1
                        000C                 .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
                0000000D 000C                        .BLKB   1
                        000D                 .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
                0000000E 000D                        .BLKB   1
                        000E                 .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
                00000010 000E                        .BLKW   1
```

```
                     0010          .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014             0010                  .BLKL   1
                     0014          .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018             0014                  .BLKL   1
                     0018          .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C             0018                  .BLKL   1
                     001C          .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020             001C                  .BLKL   1
                     0020          .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024             0020                  .BLKL   1
                     0024          .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028             0024                  .BLKL   1
                     0028          .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C             0028                  .BLKL   1
                     002C          .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E             002C                  .BLKW   1
                     002E          .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F             002E                  .BLKB   1
                     002F          .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030             002F                  .BLKB   1
                     0030          .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032             0030                  .BLKW   1
                     0032          .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034             0032                  .BLKW   1
                     0034          .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036             0034                  .BLKW   1
                     0036          .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038             0036                  .BLKW   1
                     0038          .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A             0038                  .BLKW   1
                     003A          .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C             003A                  .BLKW   1
                     003C          .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E             003C                  .BLKW   1
                     003E          .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040             003E                  .BLKW   1
                     0040          .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042             0040                  .BLKW   1
                     0042          .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044             0042                  .BLKW   1
                     0044          .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C             0044                  .BLKB   8
                     004C          .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                     004C          .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050             004C                  .BLKL   1
                     0050          .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054             0050                  .BLKL   1
                     0054          .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058             0054                  .BLKL   1
                     0058          .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C             0058                  .BLKL   1
                     005C          .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060             005C                  .BLKL   1
                     0060          .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064             0060                  .BLKL   1
                     0064          .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068             0064                  .BLKL   1
```

```
                     0068                .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078             0068                        .BLKB   16
                     0078                .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C             0078                        .BLKL   1
                     007C                .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084             007C                        .BLKQ   1
                     0084                .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088             0084                        .BLKL   1
                     0088                .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                     0088                .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A             0088                        .BLKW   1
                     008A                .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C             008A                        .BLKW   1
                     008C                .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                     008C                .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                     0000                .RESTORE
                     0000    106         $PHDDEF                                 ; PHD DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000    107         $PRDEF                                  ; PROCESSOR REGISTER DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000    108         $PRIDEF                                 ; PRIORITY INCREMENT CLASS DEFS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000    109         $PSLDEF                                 ; PSL FIELD DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000    110         $SSDEF                                  ; STATUS CODE DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             008C                .=0
                     0000                .RESTORE
                     0000    111
                     0000    112 ;
                     0000    113 ; EQUATED SYMBOLS:
                     0000    114 ;
                     0000    115
00000000             0000    116 ASTEXIT=0                                       ; AST EXIT CHANGE MODE CODE
```

```
                                0000   118          .Subtitle       Data PSect Declarations
                           00000000   119          .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                    [06]
                                0000   120
                                0000   121 ;
                                0000   122 ; OWN STORAGE:
                                0000   123 ;
                                0000   124
                                0000   125          MODNAM  ASTDEL
                  4C 45 44 54 53 41 00' 0000      $MODULE:        .ASCIC  \ASTDEL\
                                06    0000
                                0007   126
                                0007   127 ASTACKERR:
52 4F 52 52 45 20 4B 43 41 54 53 00' 0007   128          .ASCIC  .STACK ERROR.
                                0B    0007
```

```
                     0013   130          .SBTTL  SCH$ASTDEL - AST DELIVERY INTERRUPT HANDLER
                 00000000   131          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                    [06]
                     0000   132
                     0000   133 ;++
                     0000   134 ; FUNCTIONAL DESCRIPTION:
                     0000   135 ;       SCH$ASTDEL RECEIVES THE AST DELIVERY INTERRUPT (IPL - 2) WHICH
                     0000   136 ;       IS INITIATED BY AN REI INSTRUCTION DETECTING ASTLVL LESS THAN
                     0000   137 ;       OR EQUAL TO PSL<CURRENT_MODE>.  THE HEAD OF THE AST QUEUE
                     0000   138 ;       FOR THE CURRENT PROCESS IS REMOVED AND PROCESSED.  SPECIAL
                     0000   139 ;       KERNEL MODE ASTS ARE PROCESSED WITH IPL REMAINING AT IPL 2.
                     0000   140 ;       NORMAL ASTS ARE DELIVERED BY PUSHING THE AST INFORMATION ON
                     0000   141 ;       THE STACK OF THE MODE RECEIVING THE AST AND THE AST ACTIVE
                     0000   142 ;       BIT FOR THAT MODE IS SET TO PREVENT SUBSEQUENT ASTS UNTIL THE
                     0000   143 ;       CURRENT ONE FOR THAT MODE HAS BEEN PROCESSED.
                     0000   144 ;       SPURIOUS AST INTERRUPTS WILL BE DETECTED AND IGNORED.
                     0000   145 ;
                     0000   146 ;
                     0000   147 ; CALLING SEQUENCE:
                     0000   148 ;       IPL - 2  INTERRUPT
                     0000   149 ;
                     0000   150 ;
                     0000   151 ; INPUT PARAMETERS:
                     0000   152 ;       00(SP) = PC AT AST DELIVERY INTERRUPT
                     0000   153 ;       04(SP) = PSL AT AST DELIVERY INTERRUPT
                     0000   154 ;
                     0000   155 ; IMPLICIT INPUTS:
                     0000   156 ;       PCB OF CURRENT PROCESS LOCATED VIA SCH$GL_CURPCB
                     0000   157 ;       AST CONTROL BLOCK AT HEAD OF AST QUEUE FOR PROCESS
                     0000   158 ;
                     0000   159 ;
                     0000   160 ; OUTPUT PARAMETERS:
                     0000   161 ;       NONE
                     0000   162 ;
                     0000   163 ; IMPLICIT OUTPUTS:
                     0000   164 ;       *** TBS ***
                     0000   165 ;
                     0000   166 ; COMPLETION CODES:
                     0000   167 ;       NONE
                     0000   168 ;
                     0000   169 ; SIDE EFFECTS:
                     0000   170 ;       *** TBS ***
                     0000   171 ;
                     0000   172 ;--
                     0000   173
                     0000   174          .ALIGN  LONG                        ; INTERRUPT ROUTINES ON LW BOUND
                     0000   175 SCH$ASTDEL::                                 ; AST DELIVERY INTERRUPT HANDLER
           3F   BB   0000   176          PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; SAVE R0-R5
54 00000000'EF   D0   0002   177          MOVL    L^SCH$GL_CURPCB, R4         ; Get pointer to current PCB
                     0009   178          SETIPL  #IPL$_SYNCH                 ; BLOCK SYSTEM EVENTS
        12   07   DA   0009                       MTPR    #IPL$_SYNCH,S^#PR$_IPL
     55   10 B4   0F   000C   179 10$:    REMQUE  @PCB$L_ASTQFL(R4),R5       ; AND REMOVE HEAD OF QUEUE
           18   1D   0010   180          BVS     ASTDEXIT                    ; EXIT IF QUEUE EMPTY
  4B 0B A5   07   E5   0012   181          BBCC    #ACB$V_KAST,ACB$B_RMOD(R5),NORM ; BR IF NORMAL AST
```

```
                        0017   183 ;
                        0017   184 ;       THE KERNEL AST ROUTINE IS ENTERED VIA A JSB TO THE SPECIFIED
                        0017   185 ;       ADDRESS WITH IPL=2 AND THE POINTER TO THE AST CONTROL BLOCK
                        0017   186 ;       IN R5.  IT IS THE RESPONSIBILITY OF THE KERNEL AST ROUTINE
                        0017   187 ;       TO PROPERLY RELEASE OR OTHERWISE DISPOSE OF THE AST CONTROL
                        0017   188 ;       BLOCK.  THE PCB BASE ADDRESS IS IN R4.
                        0017   189 ;
                        0017   190 ;       REGISTERS R0-R5 HAVE BEEN PRESERVED AND ARE AVAILABLE FOR
                        0017   191 ;       USE BY THE AST ROUTINE.
                        0017   192 ;
                        0017   193          SETIPL  #IPL$_ASTDEL                ; DROP IPL TO PERMIT SYSTEM EVENTS
            12   02  DA 0017                MTPR    #IPL$_ASTDEL,S^#PR$_IPL
          18 B5  16  001A   194          JSB     @ACB$L_KAST(R5)             ; DO KERNEL MODE AST
    54 00000000'EF  D0  001D   195          MOVL    L^SCH$GL_CURPCB, R4         ; Get pointer to current PCB
       0000002D'EF  16  0024   196          JSB     SCH$NEWLVL                  ; COMPUTE NEW ASTLVL                    [07]
                        002A   197
                        002A   198 ASTDEXIT:                                    ; AST DELIVERY EXIT
            3F  BA  002A   199          POPR    #^M<R0,R1,R2,R3,R4,R5>      ; RESTORE REGISTERS R0-R5
                02  002C   200          REI                                 ; AND RETURN
```

```
                                  002D   ;
                                  002D   ;       SCH$NEWLVL - COMPUTE NEW ASTLVL
                                  002D   ;
                                  002D   ; INPUT:
                                  002D   ;       R4 - ADDRESS OF CURRENT PCB
                                  002D   ;
                                  002D   ; OUTPUT:
                                  002D   ;       PR$_ASTLVL - NEW AST LEVEL
                                  002D   ;       PHD$V_ASTLVL - NEW AST LEVEL
                                  002D   ;
                                  002D
                                  002D   SCH$NEWLVL::                                  ; COMPUTE NEW AST LEVEL
          50    10 A4     DE      002D           MOVAL   PCB$L_ASTQFL(R4),R0           ; GET ADDRESS OF AST QUEUE
                                  0031           DSBINT  #IPL$_SYNCH                   ; DISABLE SYSTEM EVENTS
             7E    12     DB      0031                   MFPR    S^#PR$_IPL,-(SP)
             12    07     DA      0034                   MTPR    #IPL$_SYNCH,S^#PR$_IPL
             51    60     D0      0037           MOVL    (R0),R1                       ; GET FLINK
             51    50     D1      003A           CMPL    R0,R1                         ; AND CHECK FOR EMPTY QUEUE
                   19     13      003D           BEQL    20$                           ; YES, QUEUE IS EMPTY
  52   0B A1    02    00     EF      003F           EXTZV   #0,#2,ACB$B_RMOD(R1),R2       ; GET REQUEST MODE
       04 0B A1    07     E1      0045           BBC     #ACB$V_KAST,ACB$B_RMOD(R1),10$  ; NOT A KERNEL AST
                   52     D4      004A           CLRL    R2                            ; SET TO KERNEL MODE
                   0D     11      004C           BRB     30$                           ; FOR ASTLVL
                                  004E
   50   0D A4    0C A4     8B      004E   10$:    BICB3   PCB$B_ASTACT(R4),PCB$B_ASTEN(R4),R0 ; COMPUTE DELIVERABLE
          03 50    52     E0      0054           BBS     R2,R0,30$                     ; YES, CAN BE DELIVERED
                                  0058
             52    04     D0      0058   20$:    MOVL    #4,R2                         ; NONE DELIVERABLE, NULL ASTLVL
                                  005B
             13    52     DA      005B   30$:    MTPR    R2,#PR$_ASTLVL                ; SET ASTLVL REGISTER
                                  005E           ENBINT                                ; ENABLE SYSTEM EVENTS AGAIN
             12    8E     DA      005E                   MTPR    (SP)+,S^#PR$_IPL
                   05             0061           RSB                                   ; RETURN
```

```
                                0062   233 ;
                                0062   234 ;       AT THIS POINT THE KERNEL STACK IS:
                                0062   235 ;       00(SP) = SAVED R0
                                0062   236 ;       04(SP) = SAVED R1
                                0062   237 ;       08(SP) = SAVED R2
                                0062   238 ;       12(SP) = SAVED R3
                                0062   239 ;       16(SP) = SAVED R4
                                0062   240 ;       20(SP) = SAVED R5
                                0062   241 ;       24(SP) = SAVED PC
                                0062   242 ;       28(SP) = SAVED PSL
                                0062   243 ;
                                0062   244
                                0062   245 NORM:                                        ; NORMAL AST DELIVERY
   53   0B A5   02   00   EF    0062   246          EXTZV   #ACB$V_MODE, -              ; GET AST REQUEST MODE                  [07]
                                0068   247                    #ACB$S_MODE, -          ;                                       [07]
                                0068   248                    ACB$B_RMOD(R5), -       ;                                       [07]
                                0068   249                    R3                      ;                                       [07]
   53   1C AE   02   18   ED    0068   250          CMPZV   #PSL$V_CURMOD, -            ; IS CURRENT MODE LEGAL                 [07]
                                006E   251                    #PSL$S_CURMOD, -        ;                                       [07]
                                006E   252                    28(SP), -               ;                                       [07]
                                006E   253                    R3                      ;                                       [07]
                           06   18   006E   254          BGEQ    20$                     ; YES, NOT SPURIOUS
                                0070   255
              10 A4        65   0E   0070   256 10$:     INSQUE  (R5),PCB$L_ASTQFL(R4)   ; REQUEUE AT HEAD OF QUEUE
                           B4   11   0074   257          BRB     ASTDEXIT                ; AND EXIT
                                0076   258
F5 0D A4        53   E1   0076   259 20$:     BBC     R3,PCB$B_ASTEN(R4),10$  ; BR IF AST DISABLED
F0 0C A4        53   E2   007B   260          BBSS    R3,PCB$B_ASTACT(R4),10$ ; SET AST ACTIVE
03 0B A5        06   E5   0080   261          BBCC    #ACB$V_QUOTA,ACB$B_RMOD(R5),30$ ; SKIP IF NO QUOTA ACCOUNTING
           38 A4        B6   0085   262          INCW    PCB$W_ASTCNT(R4)        ; UPDATE OUTSTANDING COUNT
                                0088   263
                                0088   264 30$:                                       ; AND DELIVER AST
              A2 AF     16   0088   265          JSB     SCH$NEWLVL              ; COMPUTE NEW AST LEVEL                 [07]
                                008B   266          SETIPL  #IPL$_ASTDEL            ; NOW DROP IPL TO PERMIT SYSTEM EVENTS
                                008B                MTPR    #IPL$_ASTDEL,S^#PR$_IPL
           12   02   DA   008B
                                008E   267 ;
                                008E   268 ;       A NEW VALUE FOR ASTLVL HAS NOW BEEN COMPUTED AND SET.
                                008E   269 ;       THE AST REPRESENTED BY THE AST CONTROL BLOCK LOCATED VIA
                                008E   270 ;       R5 CAN NOW BE DELIVERED.
                                008E   271 ;
                53   D5   008E   272          TSTL    R3                      ; CHECK FOR DELIVERY TO KERNEL
                43   12   0090   273          BNEQ    NOTKMODE                ; NOT KERNEL MODE
```

```
                        0092   275 ;
                        0092   276 ;       DELIVER NORMAL AST TO KERNEL MODE
                        0092   277 ;
                 03 BA  0092   278         POPR    #^M<R0,R1>              ; RESTORE R0,R1
                        0094   279 ;
                        0094   280 ;       00(SP) = R2, 04(SP) = R3, 08(SP) = R4, 12(SP) = R5,
                        0094   281 ;       16(SP) = PC, 20(SP) = PSL
                        0094   282 ;
                        0094   283 40$:
              7E 08 AE  7D  0094   284         MOVQ    8(SP),-(SP)             ; SHUFFLE STACK
                        0098   285 ;
                        0098   286 ;       00(SP) = R4, 04(SP) = R5, 08(SP) = R2, 12(SP) = R3,
                        0098   287 ;       16(SP) = R4, 20(SP) = R5, 24(SP) = PC, 28(SP) = PSL
                        0098   288 ;
              7E 08 AE  7D  0098   289         MOVQ    8(SP),-(SP)             ; OPEN FOR AST ARG LIST
                        009C   290 ;
                        009C   291 ;       00(SP) = R2, 04(SP) = R3, 08(SP) = R4, 12(SP) = R5,
                        009C   292 ;       16(SP) = R2, 20(SP) = R3, 24(SP) = R4, 28(SP) = R5,
                        009C   293 ;       32(SP) = PC, 36(SP) = PSL
                        009C   294 ;
              18 AE 50  7D  009C   295         MOVQ    R0,24(SP)               ; SET R0,R1 IN ARG LIST
                        00A0   296
           14 AE 14 A5  D0  00A0   297 50$:    MOVL    ACB$L_ASTPRM(R5),20(SP) ; SET AST PARAMETER IN ARG LIST
              10 AE 05  D0  00A5   298         MOVL    #5,16(SP)               ; SET COUNT FOR ARGUMENT LIST
                 50 55  D0  00A9   299         MOVL    R5,R0                   ; RELEASE AST CONTROL BLOCK
                    10 A5  DD  00AC   300         PUSHL   ACB$L_AST(R5)           ; SAVE AST ROUTINE ADDRESS
           00000000'EF 16  00AF   301         JSB     EXE$DEANONPAGED         ; TO DYNAMIC POOL                       [07]
                    3E BA  00B5   302         POPR    #^M<R1,R2,R3,R4,R5>     ; RESTORE R1-R5
                           00B7   303         SETIPL  #0                      ; DROP IPL TO ZERO
                 12 00 DA  00B7                 MTPR    #0,S^#PR$_IPL
                           00BA   304 ;       BRB     EXE$ASTDEL              ; FALL THROUGH TO CALL AST ROUTINE
```

```
                        00BA    306 ;
                        00BA    307 ;       CALL AST ROUTINE WITH AST ARGUMENT LIST
                        00BA    308 ;
                        00BA    309 ;       THE CALL IS EXECUTED AT THE MODE WHICH RECEIVED THE AST WITH
                        00BA    310 ;       THE AST ARGUMENT LIST ON THE TOP OF THE STACK.  WHEN THE
                        00BA    311 ;       AST ROUTINE RETURNS FROM THE CALL, AN ASTEXIT CHANGE MODE
                        00BA    312 ;       TO KERNEL INSTRUCTION WILL BE ISSUED.  ASTEXIT WILL RESET
                        00BA    313 ;       THE AST ACTIVE BIT FOR THE CURRENT MODE AND MAY CAUSE DELIVERY
                        00BA    314 ;       OF ADDITIONAL ASTS.
                        00BA    315 ;
                        00BA    316 ;       AST ARGUMENT LIST:
                        00BA    317 ;       ------------------
                        00BA    318 ;
                        00BA    319 ;       00(SP) = NUMBER OF ARGUMENTS, =5
                        00BA    320 ;       04(SP) = AST PARAMETER
                        00BA    321 ;       08(SP) = SAVED R0
                        00BA    322 ;       12(SP) = SAVED R1
                        00BA    323 ;       16(SP) = SAVED PC
                        00BA    324 ;       20(SP) = SAVED PSL
                        00BA    325 ;
                        00BA    326 EXE$ASTDEL:                                 ; DELIVER AST CALL
          61    6E  FA  00BA    327         CALLG   (SP),(R1)                   ; CALL AST ROUTINE
          5E    08  C0  00BD    328         ADDL    #8,SP                       ; REMOVE ARG COUNT AND ASTPRM
                        00C0    329         CMK     ASTEXIT                     ; AND EXIT FROM AST ROUTINE
                7E  DC  00C0                MOVPSL  -(SP)
        06 6E   1A  E0  00C2                BBS     #PSL$V_IS, (SP), 30000$
                8E  D4  00C6                CLRL    (SP)+
                00  BC  00C8                CHMK    #CMK$_ASTEXIT
                06  11  00CA                BRB     30001$
    00000000'EF     16  00CC        30000$: JSB     CMK$ASTEXIT
                        00D2        30001$:
                03  BA  00D2  330           POPR    #^M<R0,R1>                  ; RESTORE R0,R1
                    02  00D4  331           REI                                 ; EXECUTE REI IN MODE OF AST
```

```
                        00D5    333 ;
                        00D5    334 ;       DELIVER NORMAL AST FOR EXEC, SUPER AND USER MODE
                        00D5    335 ;
                        00D5    336
                        00D5    337 NOTKMODE:                                   ; NOT AN AST FOR KERNEL MODE
                        00D5    338 ;
          51    53  DB  00D5    339         MFPR    R3,R1                       ; GET STACK POINTER
                        00D8    340         IFNOWRT #24,-24(R1),STACKERR,R3 ; ENOUGH STACK SPACE??
E8 A1     18    53  0D  00D8                PROBEW  R3,#24,-24(R1)
                2D  13  00DD                BEQL    STACKERR
       71   18 AE  7D  00DF    341         MOVQ    24(SP),-(R1)                ; MOVE PC,PSL TO PROPER STACK
          71   8E  7D  00E3    342         MOVQ    (SP)+,-(R1)                 ; AND R0,R1 FROM KERNEL STACK
                        00E6    343
       71   14 A5  D0  00E6    344 10$:    MOVL    ACB$L_ASTPRM(R5),-(R1)  ; SET AST PARAMETER IN ARG LIST
          71   05  D0  00EA    345         MOVL    #5,-(R1)                    ; AND FINALLY, ARGUMENT COUNT OF 5
          53   51  DA  00ED    346         MTPR    R1,R3                       ; SAVE UPDATED STACK POINTER
            10 A5  DD  00F0    347         PUSHL   ACB$L_AST(R5)               ; STACK AST ENTRY POINT
  14 AE     C4 AF  DE  00F3    348         MOVAL   EXE$ASTDEL,20(SP)           ; SET PC TO AST DELIVERY CALL
          50   55  D0  00F8    349         MOVL    R5,R0                       ; SET ADDRESS OF ACB FOR RELEASE
          53   05  C4  00FB    350         MULL    #1+<1@<PSL$V_CURMOD-PSL$V_PRVMOD>>, -; CURRENT MODE = PREVIOUS MODE
                        00FE    351                   R3                    ;                                       [07]
18 AE     53   16  78  00FE    352         ASHL    #PSL$V_PRVMOD,R3,24(SP) ; SYNTHESIZE PSL FOR PROPER MODE
    00000000'EF    16  0103    353         JSB     EXE$DEANONPAGED             ; RELEASE AST CONTROL BLOCK             [07]
          3E   BA  0109    354         POPR    #^M<R1,R2,R3,R4,R5>         ; RESTORE R1-R5
               02  010B    355         REI                                 ; AND ENTER AST MODE
                        010C    356                                         ; DROPS IPL TO ZERO
                        010C    357
                        010C    358 ;
                        010C    359 ;       REFLECT STACK ERROR
                        010C    360 ;
                        010C    361 STACKERR:                                   ; ERROR IN STACK MOVE
                        010C    362         ERRSUP_S  ,ASTACKERR
    00000000'EF    DF  010C                PUSHAL  $MODULE
    00000007'EF    DF  0112                PUSHAL  ASTACKERR
          01   DD  0118                PUSHL   #$ER
00000000'EF   03   FB  011A                CALLS   #3, DS_ERRSUP
               00  0121    363         HALT
          E8   11  0122    364         BRB     STACKERR
```

```
                                0124   366          .SBTTL  SCH$QAST - ENQUEUE AST CONTROL BLOCK FOR PROCESS
                                0124   367
                                0124   368 ;++
                                0124   369 ;
                                0124   370 ; FUNCTIONAL DESCRIPTION:
                                0124   371 ;       SCH$QAST INSERTS THE AST CONTROL BLOCK SUPPLIED IN THE PROPER
                                0124   372 ;       POSITION BY ACCESS MODE IN THE AST QUEUE OF THE PROCESS SPECIFIED
                                0124   373 ;       BYT THE PID FIELD OF THE AST CONTROL BLOCK.  AN AST ARRIVAL EVENT
                                0124   374 ;       IS THEN REPORTED FOR THE PROCESS TO REACTIVATE FROM A WAIT STATE
                                0124   375 ;       IF APPROPRIATE.  THE AST CONTROL BLOCK WILL BE RELEASED IMMEDIATELY
                                0124   376 ;       IF THE PID SPECIFIES A NON-EXISTENT PROCESS.
                                0124   377 ;
                                0124   378 ; CALLING SEQUENCE:
                                0124   379 ;       BSB/JSB SCH$QAST
                                0124   380 ;
                                0124   381 ; INPUT PARAMETERS:
                                0124   382 ;       R2 - PRIORITY INCREMENT CLASS
                                0124   383 ;       R5 - POINTER TO AST CONTROL BLOCK
                                0124   384 ;
                                0124   385 ; IMPLICIT INPUTS:
                                0124   386 ;       PCB OF PROCESS IDENTIFIED BY PID FIELD
                                0124   387 ;
                                0124   388 ; OUTPUT PARAMETERS:
                                0124   389 ;       R0 - COMPLETION STATUS CODE
                                0124   390 ;       R4 - PCB ADDRESS OF PROCESS FOR WHICH AST WAS QUEUED
                                0124   391 ;
                                0124   392 ; IMPLICIT OUTPUTS:
                                0124   393 ;       *** TBS ***
                                0124   394 ;
                                0124   395 ; SIDE EFFECTS:
                                0124   396 ;       THE PROCESS IDENTIFIED BY THE PID IN THE AST CONTROL BLOCK
                                0124   397 ;       WILL BE MADE EXECUTABLE IF NOT SUSPENDED.
                                0124   398 ;
                                0124   399 ; COMPLETION CODES:
                                0124   400 ;       SS$_NORMAL  -  NORMAL SUCCESSFUL COMPLETION STATUS
                                0124   401 ;       SS$_NONEXPR -  NON-EXISTENT PROCESS
                                0124   402 ;
                                0124   403 ;--
                                0124   404
                                0124   405 SCH$QAST::                                ; ENQUEUE AST FOR PROCESS
          50    0C A5    3C     0124   406          MOVZWL  ACB$L_PID(R5),R0         ; GET PROCESS INDEX FOR AST TARGET
                                0128   407          DSBINT  #IPL$_SYNCH              ; DISABLE SYSTEM EVENTS
             7E    12    DB     0128                MFPR    S^#PR$_IPL,-(SP)
             12    07    DA     012B                MTPR    #IPL$_SYNCH,S^#PR$_IPL
    54   00000000'FF40    D0     012E   408          MOVL    @L^SCH$GL_PCBVEC[R0],R4 ; Look up PCB address
         60 A4    0C A5    D1     0136   409          CMPL    ACB$L_PID(R5),PCB$L_PID(R4) ; CHECK FOR MATCH IN PID
                   12    13     013B   410          BEQL    20$                      ; OK, PID MATCHES
                                013D   411
                                013D   412 10$:                                     ; PROCESS NOT FOUND
             50    55    D0     013D   413          MOVL    R5,R0                    ; RELEASE AST CONTROL BLOCK
          00000000'EF    16     0140   414          JSB     EXE$DEANONPAGED          ; IF NO SUCH PROCESS                    [07]
                                0146   415
                                0146   416          ENBINT                           ; ENABLE INTERRUPTS
             12    8E    DA     0146                MTPR    (SP)+,S^#PR$_IPL
          50    08E8 8F    3C     0149   417          MOVZWL  #SS$_NONEXPR,R0          ; SET ERROR STATUS CODE
                   05    05     014E   418          RSB                              ; AND RETURN
                                014F   419
```

```
                                  014F   20$:                                        ; PROCESS EXISTS
   50   0B A5   02   00   EF      014F          EXTZV   #ACB$V_MODE, -               ; EXTRACT REQUEST ACCESS MODE          [07]
                                  0155                    #ACB$S_MODE, -             ;                                      [07]
                                  0155                    ACB$B_RMOD(R5), -          ;                                      [07]
                                  0155                    R0                         ;                                      [07]
            51   10 A4   DE      0155          MOVAL   PCB$L_ASTQFL(R4),R1          ; POINT TO QUEUE HEADER
                53   61   D0      0159          MOVL    (R1),R3                      ; AND COPY HEADER ADDRESS
                                  015C
                53   51   D1      015C   30$:   CMPL    R1,R3                        ; CHECK FOR END OF QUEUE
                     17   13      015F          BEQL    40$                          ; INSERT IF AT END
         0D 0B A5   07   E0      0161          BBS     #ACB$V_KAST,ACB$B_RMOD(R5),37$  ; BR IF SPECIAL KERNEL AST
   50   0B A3   02   00   ED      0166          CMPZV   #0,#2,ACB$B_RMOD(R3),R0      ; COMPARE ACCESS MODES
                     05   14      016C          BGTR    37$                          ; IF GTR AT RIGHT PLACE
                                  016E
                53   63   D0      016E   35$:   MOVL    (R3),R3                      ; FLINK ON TO NEXT ACB
                     E9   11      0171          BRB     30$                          ; AND TRY AGAIN
                                  0173
         F6 0B A3   07   E0      0173   37$:   BBS     #ACB$V_KAST,ACB$B_RMOD(R3),35$  ; BR TO QUEUE KAST FIFO
                                  0178
                                  0178   40$:                                        ; INSERT IN AST QUEUE
            04 B3   65   0E      0178          INSQUE  (R5),@ACB$L_ASTQBL(R3)       ; INSERT AFTER PREVIOUS
            51   10 A4   D0      017C          MOVL    PCB$L_ASTQFL(R4),R1          ; ADDRESS OF AST QUEUE HEAD
                     50   D4      0180          CLRL    R0                           ; ASSUME KERNEL AST
                  0B A1   95      0182          TSTB    ACB$B_RMOD(R1)               ; KERNAL AST REQUESTED?
                     10   19      0185          BLSS    50$                          ; AN INTERNAL AST
   50   0B A1   02   00   EF      0187          EXTZV   #ACB$V_MODE, -               ; GET MOST PRIVILEGED AST LEVEL        [07]
                                  018D                    #ACB$S_MODE, -             ;                                      [07]
                                  018D                    ACB$B_RMOD(R1), -          ;                                      [07]
                                  018D                    R0                         ;                                      [07]
   51   0D A4   0C A4   8B      018D          BICB3   PCB$B_ASTACT(R4), -          ; COMPUTE INACTIVE ENABLED             [07]
                                  0193                    PCB$B_ASTEN(R4), -         ;                                      [07]
                                  0193                    R1                         ;                                      [07]
            03 51   50   E1      0193          BBC     R0,R1,80$                    ; SKIP IF ACTIVE OR DISABLED
                                  0197
                13   50   DA      0197   50$:   MTPR    R0,#PR$_ASTLVL               ; ALSO SET ASTLVL REGISTER
                                  019A
                                  019A   80$:   ENBINT                               ; ENABLE INTERRUPTS
                12   8E   DA      019A                  MTPR    (SP)+,S^#PR$_IPL
                50   01   3C      019D          MOVZWL  #SS$_NORMAL,R0               ; SET SUCCESS STATUS CODE
                          05      01A0          RSB                                  ; AND RETURN
                                  01A1
                                  01A1          .END
```

```
$ER                             = 00000002    D
$MODULE                           00000000 R  D   02
ACB$B_RMOD                        0000000B    D
ACB$B_TYPE                        0000000A    D
ACB$C_LENGTH                      00000018    D
ACB$K_LENGTH                      00000018    D
ACB$L_AST                         00000010    D
ACB$L_ASTPRM                      00000014    D
ACB$L_ASTQBL                      00000004    D
ACB$L_ASTQFL                      00000000    D
ACB$L_KAST                        00000018    D
ACB$L_PID                         0000000C    D
ACB$S_MODE                      = 00000002    D
ACB$V_KAST                      = 00000007    D
ACB$V_MODE                      = 00000000    D
ACB$V_QUOTA                     = 00000006    D
ACB$W_SIZE                        00000008    D
ASTACKERR                         00000007 R  D   02
ASTDEXIT                          0000002A R  D   03
ASTEXIT                         = 00000000    D
CMK$ASTEXIT                       ********  X   03
CMK$_APBOOT                     = 00000004    D
CMK$_ASTEXIT                    = 00000000    D
CMK$_CANTIM                     = 0000000B    D
CMK$_HIBER                      = 00000007    D
CMK$_INITSCB                    = 00000008    D
CMK$_LAST                       = 0000000E    D
CMK$_MMENABLE                   = 00000003    D
CMK$_RCONSOLE                   = 00000001    D
CMK$_REFRESH                    = 00000005    D
CMK$_SCHDWK                     = 00000006    D
CMK$_SETIMR                     = 0000000C    D
CMK$_SETIPL                     = 00000009    D
CMK$_SETPRT                     = 0000000D    D
CMK$_SGIPR                      = 0000000A    D
CMK$_TCONSOLE                   = 00000002    D
DS_ERRSUP                         ********  X   03
EXE$ASTDEL                        000000BA R  D   03
EXE$DEANONPAGED                   ********  X   03
IPL$_ASTDEL                     = 00000002    D
IPL$_SYNCH                      = 00000007    D
NORM                              00000062 R  D   03
NOTKMODE                          000000D5 R  D   03
PCB$B_ASTACT                      0000000C    D
PCB$B_ASTEN                       0000000D    D
PCB$B_PRI                         0000000B    D
PCB$B_PRIB                        0000002F    D
PCB$B_TYPE                        0000000A    D
PCB$B_WEFC                        0000002E    D
PCB$C_LENGTH                      0000008C    D
PCB$K_LENGTH                      0000008C    D
PCB$L_ARB                         00000084    D
PCB$L_ASTQBL                      00000014    D
PCB$L_ASTQFL                      00000010    D
PCB$L_EFC2P                       00000058    D
PCB$L_EFC3P                       0000005C    D
PCB$L_EFCS                        00000050    D
PCB$L_EFCU                        00000054    D
PCB$L_EFWM                        0000004C    D
PCB$L_JIB                         00000078    D
PCB$L_OWNER                       0000001C    D
PCB$L_PHD                         00000064    D
PCB$L_PHYPCB                      00000018    D
PCB$L_PID                         00000060    D
PCB$L_PQB                         0000004C    D
PCB$L_SQBL                        00000004    D
PCB$L_SQFL                        00000000    D
PCB$L_STS                         00000024    D
PCB$L_UIC                         00000088    D
PCB$L_WSSWP                       00000020    D
PCB$L_WTIME                       00000028    D
PCB$Q_PRIV                        0000007C    D
PCB$T_LNAME                       00000068    D
PCB$T_TERMINAL                    00000044    D
PCB$W_APTCNT                      00000030    D
PCB$W_ASTCNT                      00000038    D
PCB$W_BIOCNT                      0000003A    D
PCB$W_BIOLM                       0000003C    D
PCB$W_DIOCNT                      0000003E    D
PCB$W_DIOLM                       00000040    D
PCB$W_GPGCNT                      00000034    D
PCB$W_GRP                         0000008A    D
PCB$W_MEM                         00000088    D
PCB$W_MTXCNT                      0000000E    D
PCB$W_PPGCNT                      00000036    D
PCB$W_PRCCNT                      00000042    D
PCB$W_SIZE                        00000008    D
PCB$W_STATE                       0000002C    D
PCB$W_TMBU                        00000032    D
PR$_ASTLVL                      = 00000013    D
PR$_IPL                         = 00000012    D
PSL$S_CURMOD                    = 00000002    D
PSL$V_CURMOD                    = 00000018    D
PSL$V_IS                        = 0000001A    D
PSL$V_PRVMOD                    = 00000016    D
SCH$ASTDEL                        00000000 RG D   03
SCH$GL_CURPCB                     ********    X   03
SCH$GL_PCBVEC                     ********    X   03
SCH$NEWLVL                        0000002D RG D   03
SCH$QAST                          00000124 RG D   03
SIZ...                          = 00000001    D
SS$_NONEXPR                     = 000008E8    D
SS$_NORMAL                      = 00000001    D
STACKERR                          0000010C R  D   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000008C (140.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000013 (19.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	000001A1 (417.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000002	362 (1)	#-362 (1)
$MODULE	00000000-R	125 (1)	362 (1)
ACB$B_RMOD	0000000B		181 (1) 219 (1) 220 (1) 248 (1)
			261 (1) 423 (1) 430 (1) 431 (1)
			437 (1) #-443 (1) 447 (1)
ACB$L_AST	00000010		#-300 (1) #-347 (1)
ACB$L_ASTPRM	00000014		#-297 (1) #-344 (1)
ACB$L_ASTQBL	00000004		440 (1)
ACB$L_KAST	00000018		194 (1)
ACB$L_PID	0000000C		#-406 (1) #-409 (1)
ACB$S_MODE	=00000002		#-247 (1) #-422 (1) #-446 (1)
ACB$V_KAST	=00000007		#-181 (1) #-220 (1) #-430 (1) #-437 (1)
ACB$V_MODE	=00000000		#-246 (1) #-421 (1) #-445 (1)
ACB$V_QUOTA	=00000006		#-261 (1)
ASTACKERR	00000007-R	127 (1)	362 (1)
ASTDEXIT	000002A-R	198 (1)	#-180 (1) #-257 (1)
ASTEXIT	=00000000	116 (1)	
BIT...	=00000018	102 (1)	102 (1)
CMK$ASTEXIT	00000000-XR		329 (1)
CMK$_APBOOT	=00000004	102 (1)	
CMK$_ASTEXIT	=00000000	102 (1)	#-329 (1)
CMK$_CANTIM	=0000000B	102 (1)	
CMK$_HIBER	=00000007	102 (1)	
CMK$_INITSCB	=00000008	102 (1)	
CMK$_LAST	=0000000E	102 (1)	
CMK$_MMENABLE	=00000003	102 (1)	
CMK$_RCONSOLE	=00000001	102 (1)	
CMK$_REFRESH	=00000005	102 (1)	
CMK$_SCHDWK	=00000006	102 (1)	
CMK$_SETIMR	=0000000C	102 (1)	
CMK$_SETIPL	=00000009	102 (1)	
CMK$_SETPRT	=0000000D	102 (1)	
CMK$_SGIPR	=0000000A	102 (1)	
CMK$_TCONSOLE	=00000002	102 (1)	
DS_ERRSUP	00000000-XR		362 (1)
EXE$ASTDEL	000000BA-R	326 (1)	348 (1)
EXE$DEANONPAGED	00000000-XR		301 (1) 353 (1) 414 (1)
IPL$_ASTDEL	=00000002		#-193 (1) #-266 (1)
IPL$_SYNCH	=00000007		#-178 (1) #-215 (1) #-407 (1)
NORM	00000062-R	245 (1)	#-181 (1)
NOTKMODE	000000D5-R	337 (1)	#-273 (1)
PCB$B_ASTACT	0000000C		#-224 (1) 260 (1) #-449 (1)
PCB$B_ASTEN	0000000D		#-224 (1) 259 (1) #-450 (1)
PCB$L_ASTQFL	00000010		179 (1) 214 (1) 256 (1) 425 (1)
			#-441 (1)
PCB$L_PID	00000060		#-409 (1)
PCB$W_ASTCNT	00000038		#-262 (1)
PR$_ASTLVL	=00000013		#-229 (1) #-454 (1)
PR$_IPL	=00000012		#-178 (1) #-193 (1) #-215 (1) #-230 (1)
			#-266 (1) #-303 (1) #-407 (1) #-416 (1)

```
                                                  #-456    (1)
PSL$S_CURMOD                =00000002             #-251    (1)
PSL$V_CURMOD                =00000018             #-250    (1)    #-350    (1)
PSL$V_IS                    =0000001A     102    (1)  #-329    (1)
PSL$V_PRVMOD                =00000016             #-350    (1)    #-352    (1)
SCH$ASTDEL                   00000000-R   175    (1)
SCH$GL_CURPCB                00000000-XR           #-177    (1)    #-195    (1)
SCH$GL_PCBVEC                00000000-XR           #-408    (1)
SCH$NEWLVL                   0000002D-R   213    (1)    196    (1)      265    (1)
SCH$QAST                     00000124-R   405    (1)
SS$_NONEXPR                 =000008E8             #-417    (1)
SS$_NORMAL                  =00000001             #-457    (1)
STACKERR                     0000010C-R   361    (1)  #-340    (1)    #-364    (1)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$ACBDEF         2           103    (1)      103    (1)
$DEF            1           102    (1)
$DEFINI         1           103    (1)      103    (1)      104    (1)      105    (1)      106    (1)      107    (1)
                                            108    (1)      109    (1)      110    (1)
$EQU            1           102    (1)      102    (1)
$EQULS1         1           102    (1)      102    (1)
$EQULST         1           102    (1)      102    (1)
$GBLINI         2           102    (1)      102    (1)
$IPLDEF         1           104    (1)      104    (1)
$PCBDEF         4           105    (1)      105    (1)
$PHDDEF         6           106    (1)      106    (1)
$PRDEF          5           107    (1)      107    (1)
$PRIDEF         1           108    (1)      108    (1)
$PSLDEF         2           109    (1)      109    (1)
$PUSHADR        1           362    (1)      362    (1)
$SSDEF          21          110    (1)      110    (1)
$VIELD1         1           102    (1)
CMK             1           329    (1)      329    (1)
CMKDEF          1           102    (1)      102    (1)
DSBINT          1           215    (1)      215    (1)      407    (1)
ENBINT          1           230    (1)      230    (1)      416    (1)      456    (1)
ERRSUP_S        1           362    (1)      362    (1)
IFNOWRT         1           340    (1)      340    (1)
MODNAM          1           125    (1)      125    (1)
SETIPL          1           178    (1)      178    (1)      193    (1)      266    (1)      303    (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	328 (1)
ASHL	0078	352 (1)
BBC	00E1	220 (1) 259 (1) 452 (1)
BBCC	00E5	181 (1) 261 (1)
BBS	00E0	225 (1) 329 (1) 430 (1) 437 (1)
BBSS	00E2	260 (1)
BEQL	0013	218 (1) 340 (1) 410 (1) 429 (1)
BGEQ	0018	254 (1)
BGTR	0014	432 (1)
BICB3	008B	224 (1) 449 (1)
BLSS	0019	444 (1)
BNEQ	0012	273 (1)
BRB	0011	222 (1) 257 (1) 329 (1) 364 (1) 435 (1)
BVS	001D	180 (1)
CALLG	00FA	327 (1)
CALLS	00FB	362 (1)
CHMK	00BC	329 (1)
CLRL	00D4	221 (1) 329 (1) 442 (1)
CMPL	00D1	217 (1) 409 (1) 428 (1)
CMPZV	00ED	250 (1) 431 (1)
EXTZV	00EF	219 (1) 246 (1) 421 (1) 445 (1)
HALT	0000	363 (1)
INCW	00B6	262 (1)
INSQUE	000E	256 (1) 440 (1)
JSB	0016	194 (1) 196 (1) 265 (1) 301 (1) 329 (1) 353 (1) 414 (1)
MFPR	00DB	215 (1) 339 (1) 407 (1)
MOVAL	00DE	214 (1) 348 (1) 425 (1)
MOVL	00D0	177 (1) 195 (1) 216 (1) 227 (1) 297 (1) 298 (1) 299 (1)
		344 (1) 345 (1) 349 (1) 408 (1) 413 (1) 426 (1) 434 (1)
		441 (1)
MOVPSL	00DC	329 (1)
MOVQ	007D	284 (1) 289 (1) 295 (1) 341 (1) 342 (1)
MOVZWL	003C	406 (1) 417 (1) 457 (1)
MTPR	00DA	178 (1) 193 (1) 215 (1) 229 (1) 230 (1) 266 (1) 303 (1)
		346 (1) 407 (1) 416 (1) 454 (1) 456 (1)
MULL	00C4	350 (1)
POPR	00BA	199 (1) 278 (1) 302 (1) 330 (1) 354 (1)
PROBEW	000D	340 (1)
PUSHAL	00DF	362 (1)
PUSHL	00DD	300 (1) 347 (1) 362 (1)
PUSHR	00BB	176 (1)
REI	0002	200 (1) 331 (1) 355 (1)
REMQUE	000F	179 (1)
RSB	0005	231 (1) 418 (1) 458 (1)
TSTB	0095	443 (1)
TSTL	00D5	272 (1)

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ALIGN          174     (1)
.ASCIC          125     (1)     128     (1)
.END            460     (1)
.ENDC           102     (1)     178     (1)     193     (1)     215     (1)     230     (1)     266     (1)     303     (1)     362     (1)
                407     (1)     416     (1)     456     (1)
.ENDM           102     (1)     103     (1)     104     (1)     105     (1)     106     (1)     107     (1)     108     (1)     109     (1)
                110     (1)     125     (1)     178     (1)     215     (1)     230     (1)     329     (1)     340     (1)     362     (1)
.ENDR           102     (1)
.IDENT          7       (1)
.IF             102     (1)     178     (1)     193     (1)     215     (1)     230     (1)     266     (1)     303     (1)     362     (1)
                407     (1)     416     (1)     456     (1)
.IFF            102     (1)     178     (1)     193     (1)     215     (1)     230     (1)     266     (1)     303     (1)     362     (1)
                407     (1)     416     (1)     456     (1)
.IIF            102     (1)     362     (1)
.IRP            102     (1)
.LIBRARY        100     (1)     99      (1)
.MACRO          102     (1)
.MDELETE        102     (1)
.NOCROSS        103     (1)     104     (1)     105     (1)     106     (1)     107     (1)     108     (1)     109     (1)     110     (1)
.NOSHOW         8       (1)
.PAGE           117     (1)     129     (1)     182     (1)     201     (1)     232     (1)     274     (1)     305     (1)     332     (1)
                365     (1)     93      (1)
.PSECT          119     (1)     131     (1)
.RESTORE        103     (1)     104     (1)     105     (1)     106     (1)     107     (1)     108     (1)     109     (1)     110     (1)
.SAVE           103     (1)     104     (1)     105     (1)     106     (1)     107     (1)     108     (1)     109     (1)     110     (1)
.SBTTL          130     (1)     366     (1)     45      (1)
.SUBTITLE       118     (1)     94      (1)
.TITLE          6       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	23	#-176	(1)	#-199	(1)	#-214	(1)	#-216	(1)	#-217	(1)	#-224	(1)
		225	(1)	#-278	(1)	#-295	(1)	#-299	(1)	#-330	(1)	#-349	(1)
		#-406	(1)	#-408	(1)	#-413	(1)	#-417	(1)	#-424	(1)	#-431	(1)
		#-442	(1)	#-448	(1)	#-452	(1)	#-454	(1)	#-457	(1)		
R1	26	#-176	(1)	#-199	(1)	#-216	(1)	#-217	(1)	219	(1)	220	(1)
		#-278	(1)	#-302	(1)	327	(1)	#-330	(1)	#-339	(1)	340	(1)
		#-341	(1)	#-342	(1)	#-344	(1)	#-345	(1)	#-346	(1)	#-354	(1)
		#-425	(1)	#-426	(1)	#-428	(1)	#-441	(1)	#-443	(1)	447	(1)
		#-451	(1)	452	(1)								
R2	9	#-176	(1)	#-199	(1)	#-219	(1)	#-221	(1)	#-225	(1)	#-227	(1)
		#-229	(1)	#-302	(1)	#-354	(1)						
R3	21	#-176	(1)	#-199	(1)	#-249	(1)	#-253	(1)	#-259	(1)	#-260	(1)
		#-272	(1)	#-302	(1)	#-339	(1)	#-340	(1)	#-346	(1)	#-351	(1)
		#-352	(1)	#-354	(1)	#-426	(1)	#-428	(1)	431	(1)	#-434	(1)
		437	(1)	440	(1)								
R4	20	#-176	(1)	#-177	(1)	179	(1)	#-195	(1)	#-199	(1)	214	(1)
		#-224	(1)	256	(1)	259	(1)	260	(1)	#-262	(1)	#-302	(1)
		#-354	(1)	#-408	(1)	#-409	(1)	425	(1)	#-441	(1)	#-449	(1)
		#-450	(1)										
R5	22	#-176	(1)	#-179	(1)	181	(1)	194	(1)	#-199	(1)	248	(1)
		256	(1)	261	(1)	#-297	(1)	#-299	(1)	#-300	(1)	#-302	(1)
		#-344	(1)	#-347	(1)	#-349	(1)	#-354	(1)	#-406	(1)	#-409	(1)
		#-413	(1)	423	(1)	430	(1)	440	(1)				
SP	22	#-215	(1)	#-230	(1)	252	(1)	#-284	(1)	#-289	(1)	#-295	(1)
		#-297	(1)	#-298	(1)	327	(1)	#-328	(1)	#-329	(1)	#-341	(1)
		#-342	(1)	#-348	(1)	#-352	(1)	#-407	(1)	#-416	(1)	#-456	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.21
Command processing	876	00:00:00.17	00:00:00.84
Pass 1	693	00:00:02.03	00:00:03.72
Symbol table sort	0	00:00:00.25	00:00:00.24
Pass 2	79	00:00:00.55	00:00:00.93
Symbol table output	2	00:00:00.02	00:00:00.04
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	5	00:00:00.12	00:00:00.18
Assembler run totals	1680	00:00:03.16	00:00:06.16

The working set limit was 2512 pages.
114580 bytes (224 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 813 non-local and 20 local symbols.
460 source lines were read in Pass 1, producing 58 object records in Pass 2.
62 pages of virtual memory were used to define 25 macros.

```
                              +---------------------------+
                              ! Macro library statistics !
                              +---------------------------+

Macro library name                                 Macros defined
------------------                                 --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8                     3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                       4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8                     0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                       0
SYS$COMMON:[SYSLIB]LIB.MLB;2                                  6
SYS$COMMON:[SYSLIB]STARLET.MLB;2                              9
TOTALS (all libraries)                                       22

1024 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ASTDEL.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1)	207	Libraries, Equated Symbols, Externals	
(1)	262	Data Psect Declarations	
(1)	344	Work Psect Declarations	
(1)	361	DSV$ATTACH	Attach command processor
(1)	462	DSX$ATTACH	Process ATTACH command
(1)	658	Process invalid generic name error	
(1)	667	Process invalid generic name error	
(1)	777	Get name processing	
(3)	849	Get device attributes	
(3)	942	parse device name	
(3)	1160	PT_INTERPRET	Decode PT-descriptor
(3)	1463	DP$EXTRACT	Make P-table printable
(4)	1658	GET_LINE	Condition getline
(4)	1708	FORMAT_PROMPT	Format and load prompt
(4)	1754	DSV$SHOWDEVICE	Display device information
(4)	1828	DSV$SHOWSELECT	Display information selects
(4)	1880	DSV$DEATTACH	Deattach device
(4)	1949	Deattach subtree	
(4)	1978	DSV$SELECT	Select device for testing
(4)	2171	DSV$DESELECT	Deselect device for testing
(5)	2292	SHOW_DEV Routine	
(6)	2385	INI$PTABLE	Initialize P-table
(6)	2424	Check ambiguous device	
(6)	2495	Locate /Adapter value	
(6)	2555	LOC$PTABLE Routine	
(6)	2623	LOC$PTDESC	Locate PT-descriptor
(6)	2695	INS$PTABLE	Insert P-table

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ATTACH.MAR
                        0000     2 ; *9    11-NOV-1987 08:34:14 GEARIN "ADDED SUPPORT FOR EXTENDED P-TABLE"
                        0000     3 ; *8    10-AUG-1987 09:50:52 GEARIN "FIXED IDENT FIELD"
                        0000     4 ;  7A1  10-AUG-1987 09:15:00 GEARIN "MODIFY MAX UNIT # FIX"
                        0000     5 ; *7    13-JUL-1987 09:10:42 GEARIN "FIXED MAX UNIT # BUG"
                        0000     6 ; *6    12-MAY-1987 09:42:30 MI "ADDED ALL AS A KEYWORD PARAMETER TO DEATTACH"
                        0000     7 ; *5    27-JAN-1987 09:22:46 GEARIN "REPLACED = IDENT WITH"
                        0000     8 ; *4    12-MAY-1986 10:18:26 BARANSKI "Cleaning up Object Libraries."
                        0000     9 ; *3     7-MAY-1986 14:29:16 BARANSKI "Documentaion Check."
                        0000    10 ; *2    12-FEB-1986 12:26:04 BARANSKI "<no reason given>"
                        0000    11 ; *1     6-FEB-1986 15:13:39 DS ""
                        0000    12 ; DEC/CMS REPLACEMENT HISTORY, Element ATTACH.MAR
                        0000    13         .Title  ATTACH  *** ATTACH Handle ATTACH command
                        0000    14         .Ident  "10.10-9"                                                    ;G:9
                        0000    15         .Dsabl  GBL
                        0000    16         .NoShow Conditionals
                        0000    17                                                                             ;G:9
                        0000    18 ;++
                        0000    19 ; Copyright (c) 1978, 1982, 1983, 1984,1985,1987                            ;G:6
                        0000    20 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000    21 ;
                        0000    22 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    23 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    24 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    25 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    26 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    27 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    28 ; REMAIN IN DEC.
                        0000    29 ;
                        0000    30 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    31 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    32 ; CORPORATION.
                        0000    33 ;
                        0000    34 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    35 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    36
                        0000    37 ;++
                        0000    38 ; Facility:
                        0000    39 ;
                        0000    40 ; Abstract:
                        0000    41 ;
                        0000    42 ; Environment:
                        0000    43 ;
                        0000    44 ; Author: Roger Riggs, Creation date:  10-OCT-1978
                        0000    45 ;
                        0000    46 ; Modified by:
                        0000    47 ;                                                                           ;G:3
                        0000    48 ;       [02]    Roger Riggs, 27-Mar-1980                                    ;G:3
                        0000    49 ;               Extended buffer for show devices.
                        0000    50 ;                                                                           ;G:3
                        0000    51 ;       [03]    Roger Riggs, 10-Apr-1980                                    ;G:3
                        0000    52 ;               Added $DS_$ADD and $DS_$COMPLEMENT operations to
                        0000    53 ;               PT-descriptor interpreter
                        0000    54 ;                                                                           ;G:3
                        0000    55 ;       [04]    Dave Butenhof, 24-sep-1980, Version 6.1                     ;G:3
                        0000    56 ;               Split DSV$ATTACH routine into DSV$ATTACH control routine
                        0000    57 ;               and DSX$ATTACH, which is callable by diagnostics and will
```

```
                        0000     58 ;               cause no typeout or console/script reads.  The primary
                        0000     59 ;               purpose is to support the AUTOSIZER.
                        0000     60 ;                                                                        ;G:3
                        0000     61 ;       [05]    Dave Butenhof, 19-dec-1980, Version 6.2                  ;G:3
                        0000     62 ;               Implement MCT as NEBULA 'central bus'.  Also generalize
                        0000     63 ;               'central bus' string compares, and generate error if
                        0000     64 ;               the wrong 'central bus' name is typed (e.g., 'CMI' for 780).
                        0000     65 ;                                                                        ;G:3
                        0000     66 ;       [06]    Dave Butenhof, 30-mar-1981, version 6.3                  ;G:3
                        0000     67 ;               Implement new 'hub' structures, cross-implementation-wise:
                        0000     68 ;               'HUB' and 'PHAD'.
                        0000     69 ;                                                                        ;G:3
                        0000     70 ;       [07]    Marion Baggett, 2-Oct-1981 Version 6.-                   ;G:3
                        0000     71 ;               Print only device and type fields for /BRIEF switch
                        0000     72 ;               on SHOW DEVICE, and SHOW SELECT commands.
                        0000     73 ;                                                                        ;G:3
                        0000     74 ;       [08]    Dave Butenhof, 20-Oct-1981, version 6.-                  ;G:3
                        0000     75 ;               Fix a bug in Ptable descriptor location routine.
                        0000     76 ;                                                                        ;G:3
                        0000     77 ;       [09]    Dave Butenhof, 22-Oct-1981, version 6.-                  ;G:3
                        0000     78 ;               Add ATTACH control C handler.  Also new Ptable
                        0000     79 ;               descriptor commands $DS_$LOGICAL and $DS_$CASE.
                        0000     80 ;                                                                        ;G:3
                        0000     81 ;       [10]    Jack Stansbury, 27-Oct-1981, Version 6.-                 ;G:3
                        0000     82 ;               Fixed truncation errors.
                        0000     83 ;                                                                        ;G:3
                        0000     84 ;       [11]    Dave Butenhof, 06-Nov-1981, Version 6.-                  ;G:3
                        0000     85 ;               Implemented SELECT/ADAPTER and DESELECT/ADAPTER.
                        0000     86 ;                                                                        ;G:3
                        0000     87 ;       [12]    Dave Butenhof, 18-Nov-1981, version 6.5                  ;G:3
                        0000     88 ;               Implement DEATTACH command.
                        0000     89 ;                                                                        ;G:3
                        0000     90 ;       [13]    Dave Butenhof, 20-Nov-1981, version 6.5                  ;G:3
                        0000     91 ;               Convert all PRINT's to use type codes, and delete unused
                        0000     92 ;               message 't_insfmem'.
                        0000     93 ;                                                                        ;G:3
                        0000     94 ;       [14]    Dave Butenhof, 24-Feb-1982, version 6.6                  ;G:3
                        0000     95 ;               Convert parsing of Device type field to allow use of
                        0000     96 ;               '_' in the name (e.g., 'CI_NODE').
                        0000     97 ;                                                                        ;G:3
                        0000     98 ;       [15]    Dave Butenhof, 02-Mar-1982, version 6.6                  ;G:3
                        0000     99 ;               Correct bug in SELECT/DESELECT: If /Adapter not specified,
                        0000    100 ;               and device is duplicated, only first found is acted on, and
                        0000    101 ;               no message is given.  Due to SELECT sorting, it may not even
                        0000    102 ;               be possible to DESELECT that device (without /Adapter).
                        0000    103 ;               Solution: Give error if no /Adapter on duplicate name.
                        0000    104 ;                                                                       ;G:3
                        0000    105 ;       [16]    Dave Butenhof, 10-Mar-1982, version 6.6                 ;G:3
                        0000    106 ;               Correct some truncation errors.
                        0000    107 ;                                                                       ;G:3
                        0000    108 ;       [17]    Dave Butenhof, 07-Apr-1982, version 6.7                 ;G:3
                        0000    109 ;               Add support for new Ptable descriptor directive, $DS_$NAME;
                        0000    110 ;               this will allow use of enforced generic naming conventions,
                        0000    111 ;               and inherited device names, while remaining compatible with
                        0000    112 ;               the current schema.  Note that this directive must IMMEDIATELY
                        0000    113 ;               follow the $DS_$INITIALIZE directive, and is treated as an
                        0000    114 ;               extension of same.
```

```
0000   115 ;                                                                    ;G:3
0000   116 ;       [18]    Jack Stansbury, 13-Apr-1982, Version 6.7             ;G:3
0000   117 ;               Fixed two truncation errors.
0000   118 ;                                                                    ;G:3
0000   119 ;       [19]    Dave Butenhof, 06-May-1982, Version 6.8              ;G:3
0000   120 ;               Make modifications (very minor) to support splitting
0000   121 ;               PTD$M_INHERITED into two separated bits, PTD$M_INHERIT_PRE
0000   122 ;               to inherit device prefix, and PTD$M_INHERIT_CON to inherit
0000   123 ;               controller designation.
0000   124 ;
0000   125 ;       [20]    Marion Baggett, 6-Oct-1982 Version 6.9               ;G:3
0000   126 ;               When inserting PTABLE for RB730, do not re-adjust the RB730
0000   127 ;               PTABLE for CSR values.  The RB730 is not on the the unibus.
0000   128 ;
0000   129 ;       [21]    Domenic Andella, 13-Apr-82 Version 6.11              ;G:3
0000   130 ;               Added central bus name "ABUS" to CENTRAL_BUS list, and
0000   131 ;               changed constant in DSX$ATTACH routine, GET_LINK label
0000   132 ;               from 3 to 4 to account for new central bus name.
0000   133 ;
0000   134 ;       [22]    John Ciukaj     14-Apr-1983     Version 6.11
0000   135 ;               - Inhibit device allocation in user mode during
0000   136 ;                 Select operation if 'Inhibit Allocate' bit is
0000   137 ;                 set in CRD Flags
0000   138 ;
0000   139 ;       [23]    Domenic Andella 14-Apr-83       Version 6.11
0000   140 ;               Corrected bug in DSX$ATTACH routine, GET_LINK label
0000   141 ;               to check all CPU link names.
0000   142 ;
0000   143 ;       [24]    M. Baggett      11-MAY-1983     VERSION 6.11
0000   144 ;               Changed ptd$m_name to mean alphanumeric.
0000   145 ;
0000   146 ;       [25]    Bob Bergazzi    May 16,1983     Version 6.11
0000   147 ;               Changed the order of the .LIB statements.
0000   148 ;
0000   149 ;       [26]    M. Baggett      10-OCT-1983     VERSION 6.13
0000   150 ;               Change to support HSC50 disk/tape names of the form
0000   151 ;               name$GGannn, where 'name' is a maximum
0000   152 ;               of 6 characters and 'GG' is the generic name and is
0000   153 ;               present with a controller letter and unit number.
0000   154 ;
0000   155 ;       [27]    Domenic Andella 16-FEB-1984     VERSION 6.13
0000   156 ;               Add code in GET_LINK to add offset of X200 to
0000   157 ;               ptable vector field if attached to SBIA 1 for
0000   158 ;               an 8600 CPU.
0000   159 ;
0000   160 ;       [28]    RE MUSE         11 MAY 1984     Version 7.0
0000   161 ;               added code to handle long device names (15 characters)
0000   162 ;               of the form 'node$device'.
0000   163 ;
0000   164 ;       [29]    Ted Salem       09 AUG 1984     Version 7.1          ;G:3
0000   165 ;               Added code to check if /noallocate is present in
0000   166 ;               the select command.
0000   167 ;
0000   168 ;       [30]    Ted Salem       29 AUG 1984     Version 7.1
0000   169 ;               Modified SHOW SELECT and SHOW DEVICE commands so that the returned I
0000   170 ;               will contain information on the devices' allocation status.
0000   171 ;
```

```
0000    172 ;       [31]    Amy Iorio       16 OCT 1984     Version 7.1
0000    173 ;               Added allocation class to possible device names
0000    174 ;               user may declare.
0000    175 ;
0000    176 ;       [32]    M. Baggett      14-Nov-1984     Version 7.1
0000    177 ;               Fixed bugs in [31] and clean up some coding.
0000    178 ;
0000    179 ;       [33]    Richard E. Muse 18-Jan-1985     VERSION 6.13
0000    180 ;               If adapter being attached is on a BI. Need to
0000    181 ;               add code to GET_LINK such that the  HP$W_VECTOR
0000    182 ;               field contains the correct BI offset.
0000    183 ;
0000    184 ;       [34]    Ted Salem       22-Jan-1985     Version 8.1
0000    185 ;               Fixed a /noallocate problem.
0000    186 ;
0000    187 ;       [35]    R.E. Muse       9-MAY-1985      Version 8.2                     ;G:3
0000    188 ;               removed cpu specific code that check for the vector
0000    189 ;               offset requirement - this code was placed in their
0000    190 ;               respective "ONLYnnn" modules for each different cpu type.
0000    191 ;
0000    192 ;               Jason Mi        12-May-1987     Version 10.8                    ;G:6
0000    193 ;               Added ALL as a keyword paramater to DEATTACH command.           ;G:6
0000    194 ;
0000    195 ;               David Gearin    07-Jul-1987     Version 10.9                    ;G:8
0000    196 ;               If max unit # is equal to 255 do not decrement the              ;G:9
0000    197 ;               the maximum unit # by one. [temp fix]                           ;G:9
0000    198 ;                                                                               ;G:8
0000    199 ;               David Gearin    10-Aug-1987     Version 10.9                    ;G:8
0000    200 ;               Fix ident field to 10.9-8.                                      ;G:8
0000    201 ;                                                                               ;G:9
0000    202 ;               David Gearin    02-Oct-1987     Version 10.10                   ;G:9
0000    203 ;               Add new extended ptable block format to ptables.                ;G:9
0000    204 ;-
0000    205
```

```
                                0000   207          .Subtitle       Libraries, Equated Symbols, Externals
                                0000   208 ;
                                0000   209 ; Libraries
                                0000   210 ;
                                0000   211
                                0000   212 .library         \SYS$LIBRARY:LIB\       ;                                [25]
                                0000   213 .library         \$DS\                   ;                                [25]
                                0000   214 .library         \$diag\                 ;                                [25]
                                0000   215
                                0000   216 ;
                                0000   217 ; Equated Symbols:
                                0000   218 ;
                                0000   219 ;       .NLIST
                                0000   220 ;       .NLIST  ME,MEB
                                0000   221 ;       .LIST
                                0000   222         $Ds_PtdDef                              ; Define $DS_$NAME flag codes   [17]
                                0000   223         $Ds_PdDef                               ; Define PD$_ byte codes        [17]
                                0000   224         $DS_HDRDEF
                                0000   225         $DS_HPODEF
                                0000               .SAVE   LOCAL_BLOCK
                                0000               .PSECT  $ABS$,ABS
                    00000000    0258               .=0
                                0000               .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
                    00000008    0000               .IIF    NB,,BLKQ,       .BLKQ   1
                                0008               .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
                    0000000A    0008               .IIF    NB,,BLKW,       .BLKW   1
                                000A               .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
                    0000000B    000A               .IIF    NB,,BLKB,       .BLKB   1
                                000B               .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
                    0000000C    000B               .IIF    NB,,BLKB,       .BLKB   1
                                000C               .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
                    00000018    000C               .IIF    NB,,BLKB,       .BLKB   12
                                0018               .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
                    0000001C    0018               .IIF    NB,,BLKL,       .BLKL   1
                                001C               .IIF    NB,HP$A_DVA,    HP$A_DVA:
                    00000020    001C               .IIF    NB,,BLKL,       .BLKL   1
                                0020               .IIF    NB,HP$A_LINK,   HP$A_LINK:
                    00000024    0020               .IIF    NB,,BLKL,       .BLKL   1
                                0024               .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
                    00000026    0024               .IIF    NB,,BLKW,       .BLKW   1
                                0026               .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
                    00000032    0026               .IIF    NB,,BLKB,       .BLKB   12
                                0032               .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                                0000               .RESTORE
                                0000   226         $DS_HPEODEF                                                     ;G:9
                                0000               .SAVE   LOCAL_BLOCK
                                0000               .PSECT  $ABS$,ABS
                    00000000    0258               .=0
                                0000               .IIF    NB,HPE$T_DEVICE,        HPE$T_DEVICE:
                    00000014    0000               .IIF    NB,,BLKL,       .BLKL   5
                                0014               .IIF    NB,HPE$W_EXT_DRIVE,     HPE$W_EXT_DRIVE:
                    00000016    0014               .IIF    NB,,BLKW,       .BLKW   1
                                0016               .IIF    NB,HPE$A_EPB,   HPE$A_EPB:
                    0000001A    0016               .IIF    NB,,BLKL,       .BLKL   1
                                0000               .RESTORE
                                0000   227         $DS_DSADEF
                                0000   228         $DS_DSDEF
```

```
                     0000   229          $PRDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 FE70                .=0
                     0000                .RESTORE
                     0000   230          $SSDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 FE70                .=0
                     0000                .RESTORE
                     0000   231          $IPLDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 FE70                .=0
                     0000                .RESTORE
                     0000   232          CLIDEF
                     0000   233          DSFDEF                                  ;                                       [07]
                     0000   234          $ds_dsadef                              ; APT mailbox defs                      [09]
                     0000   235          $ds_typedef                             ; Define type codes                     [12]
                     0000   236          CRDDef                                  ; Define bit definitions for
                     0000   237                                                  ; CRD$GL_CRD_Flags                      [22]
                     0000   238
                     0000   239 ;        .NLIST
                     0000   240 ;        .LIST   MEB
                     0000   241 ;        .LIST
                     0000   242
                     0000   243
                     0000   244          .EXTRN  DS$GA_SELECTS,  DS$GL_CLIBASE,  ONLY_ATTACH
                     0000   245          .EXTRN  DS$PRINTF,      DS_SUPERLINE,   DS_ERRSUP,      SCRIPT$FLUSH
                     0000   246          .EXTRN  EXE$ALONONPAGED,EXE$DEANONPAGED,DSX$PRINT       ;                       [12]
                     0000   247          .EXTRN  DS$GA_ALLOCATES                                 ;                       [30]
                     0000   248          .EXTRN  SCAN$ALPHANUM,  SCAN$DEVICE,    SCAN$NUMERIC
                     0000   249          .EXTRN  SCAN$SPACES,    SCAN$SYMBOL
                     0000   250          .EXTRN  SYS$FAO,        SYS$ALLOC,      SYS$DALLOC
                     0000   251          .EXTRN  DS$GA_PTABLE,   DS$GA_PTDESC
                     0000   252          .EXTRN  DSR$COMPLETION, DSR$IFLOADDEV
                     0000   253          .EXTRN  DS$GL_FLAGS, KB_CHECK   ;                                               [09]
                     0000   254          .extrn  scan$decimal,scan$alpha ;                                               [09]
                     0000   255          .extrn  ds$cntrlc, Dsx$Print, Ds$GB_Inhibit_Naming      ;                       [17]
                     0000   256          .extrn  DS$GL_CRD_FLAGS         ;                                               [22]
                     0000   257                                                                                         ;G:9
            0000001A 0000   258          HPE$C_EPB_SIZE  == 26           ; Size of extended p-table                      ;G:9
            00000014 0000   259          HP$C_NAMELEN    == 20           ;                                               [28]
                     0000   260                                                                                         ;G:9
```

```
                                        0000   262          .SBTTL  Data Psect Declarations
                                        0000   263
                                    00000000   264          .PSECT  Data, NoExe, Shr, NoWrt, Long          ;                [10]
                                        0000   265          MODNAM  ATTACH
                48 43 41 54 54 41 00'   0000       $MODULE:        .ASCIC  \ATTACH\
                                        06      0000
                                        0007   266 T_INVALID:
69 6C 61 76 6E 49 0000000F'010E0000'    0007   267          .ASCID  "Invalid or unknown name.!/!AC? "
6E 77 6F 6E 6B 6E 75 20 72 6F 20 64     0015
3F 43 41 21 2F 21 2E 65 6D 61 6E 20     0021
                                20      002D
                                        002E   268 T_Bad_Name:                                                 ;        [17]
72 72 45 20 3F 3F 00000036'010E0000'    002E   269          .Ascid  \?? Error in !AC!ACname, format should be "!AC"!/!AC? \ ;        [17]
43 41 21 43 41 21 20 6E 69 20 72 6F     003C
74 61 6D 72 6F 66 20 2C 65 6D 61 6E     0048
22 20 65 62 20 64 6C 75 6F 68 73 20     0054
   20 3F 43 41 21 2F 21 22 43 41 21     0060
                                        006B   270 T_BAD_RANGE:                                               ;        [31]
72 72 45 20 3F 3F 00000073'010E0000'    006B   271          .ASCID  \?? Error in !AC, range should be "!AC"!/!AC? \;
72 20 2C 43 41 21 20 6E 69 20 72 6F     0079
20 64 6C 75 6F 68 73 20 65 67 6E 61     0085
41 21 2F 21 22 43 41 21 22 20 65 62     0091
                           20 3F 43     009D
                                        00A0   272 T_ALLOC:                                                   ;        [31]
            6E 61 67 67 24 6D 24 00'    00A0   273          .Ascic  \$m$ggan\                               ;
                                    07  00A0
                                        00A8   274 T_RANGE:                                                   ;        [31]
61 20 31 20 6E 65 65 77 74 65 62 00'    00A8   275          .Ascic  \between 1 and 255\    ;
                  35 35 32 20 64 6E     00B4
                                    11  00A8
                                        00BA   276 T_UNRANGE:                                                 ;        [31]
6E 6F 69 74 61 63 6F 6C 6C 61 20 00'    00BA   277          .Ascic  \ allocation range \    ;
               20 65 67 6E 61 72 20     00C6
                                    12  00BA
                                        00CD   278 T_Part_Of:                                                 ;        [17]
      20 66 6F 20 74 72 61 70 20 00'    00CD   279          .Ascic  \ part of \             ;                [17]
                                    09  00CD
                                        00D7   280 T_UNIT_EXCESS:
6E 20 74 69 6E 55 000000DF'010E0000'    00D7   281          .ASCID  "Unit number too big or not allowed.!/!AC? "
69 62 20 6F 6F 74 20 72 65 62 6D 75     00E5
6C 6C 61 20 74 6F 6E 20 72 6F 20 67     00F1
20 3F 43 41 21 2F 21 2E 64 65 77 6F     00FD
                                        0109   282 T_MAIN_BUS:
20 65 72 65 68 54 00000111'010E0000'    0109   283          .ASCID  "There is no !AD on this type processor.!/!AC? "
6E 6F 20 44 41 21 20 6F 6E 20 73 69     0117
70 20 65 70 79 74 20 73 69 68 74 20     0123
21 2F 21 2E 72 6F 73 73 65 63 6F 72     012F
                        20 3F 43 41     013B
                                        013F   284 T_PROMPT:
   20 3F 43 41 21 00000147'010E0000'    013F   285          .ASCID  "!AC? "
65 6D 61 4E 20 65 63 69 76 65 44 00'    014C   286 T_NAME: .ASCIC  "Device Name"
                                    0B  014C
6B 6E 69 4C 20 65 63 69 76 65 44 00'    0158   287 T_LINK: .ASCIC  "Device Link"
                                    0B  0158
65 70 79 74 20 65 63 69 76 65 44 00'    0164   288 T_TYPE: .ASCIC  "Device type"
                                    0B  0164
                                        0170   289 T_NO_SUCH_DEV:
```

```
76 65 64 20 68 63 75 73 20 6F 4E 00' 0170   290          .ASCIC  "No such device !AD:!/"
      2F 21 3A 44 41 21 20 65 63 69  017C
                                 15  0170
                                     0186   291 t_no_such_adapter:                    ;                                  [11]
61 64 61 20 68 63 75 73 20 6F 4E 00' 0186   292          .ascic  "No such adapter !AD:!/"        ;                          [11]
   2F 21 3A 44 41 21 20 72 65 74 70  0192
                                 16  0186
                                     019D   293 t_no_such_on_adapter:                 ;                                  [11]
76 65 64 20 68 63 75 73 20 6F 4E 00' 019D   294          .ascic  "No such device !AD: on adapter !AD:!/" ;                  [11]
20 6E 6F 20 3A 44 41 21 20 65 63 69  01A9
3A 44 41 21 20 72 65 74 70 61 64 61  01B5
                              2F 21  01C1
                                 25  019D
                                     01C3   295 T_Ambig_Adap:                         ; Ambigous adapter name            [15]
74 61 63 69 6C 70 75 44 20 3F 3F 00' 01C3   296          .AsciC  "?? Duplicate adapter name is illegal (!AS)!/"  ;          [15]
61 6E 20 72 65 74 70 61 64 61 20 65  01CF
61 67 65 6C 6C 69 20 73 69 20 65 6D  01DB
         2F 21 29 53 41 21 28 20 6C  01E7
                                 2C  01C3
                                     01F0   297 T_Ambig_Dev:                          ; Ambiguous device name            [15]
65 73 75 20 74 73 75 4D 20 3F 3F 00' 01F0   298          .AsciC  "?? Must use /Adapter with duplicated device name !AS!/" ;  [15]
69 77 20 72 65 74 70 61 64 41 2F 20  01FC
65 74 61 63 69 6C 70 75 64 20 68 74  0208
6D 61 6E 20 65 63 69 76 65 64 20 64  0214
               2F 21 53 41 21 20 65  0220
                                 36  01F0
                                     0227   299 t_deattach:                           ;                                  [12]
65 65 62 20 73 61 68 20 53 41 21 00' 0227   300          .ascic  "!AS has been de-attached.!/"   ;                          [12]
65 68 63 61 74 74 61 2D 65 64 20 6E  0233
                        2F 21 2E 64  023F
                                 1B  0227
                                     0243   301 T_DEVALLOC:
61 63 6F 6C 6C 41 20 3A 44 41 21 00' 0243   302          .ASCIC  "!AD: Allocated to another user!/"
68 74 6F 6E 61 20 6F 74 20 64 65 74  024F
         2F 21 72 65 73 75 20 72 65  025B
                                 20  0243
                                     0264   303 T_DEV_MOUNTED:                                          ;                  [34]
65 63 69 76 65 44 20 3A 44 41 21 00' 0264   304          .ASCIC  "!AD: Device is currently mounted and cannot be allocated.!/" ; [34]
6C 74 6E 65 72 72 75 63 20 73 69 20  0270
6E 61 20 64 65 74 6E 75 6F 6D 20 79  027C
20 65 62 20 74 6F 6E 6E 61 63 20 64  0288
2F 21 2E 64 65 74 61 63 6F 6C 6C 61  0294
                                 3B  0264
                                     02A0   305 T_SHOWDEV_ALLOC:                                        ;                  [30]
21 5F 21 43 41 21 5F 21 53 41 21 00' 02A0   306          .ASCIC  "!AS!_!AC!_!AS!_!XL  !AS  ALLOCATED !/"         ;                  [30]
53 41 21 20 20 4C 58 21 5F 21 53 41  02AC
20 44 45 54 41 43 4F 4C 4C 41 20 20  02B8
                              2F 21  02C4
                                 25  02A0
                                     02C6   307 T_SHOWDEVB_ALLOC:                                ;                          [30]
41 20 20 43 41 21 5F 21 53 41 21 00' 02C6   308          .ASCIC  "!AS!_!AC  ALLOCATED !/"                ;                  [30]
   2F 21 20 44 45 54 41 43 4F 4C 4C  02D2
                                 16  02C6
                                     02DD   309 T_SHOWDEV:
21 5F 21 43 41 21 5F 21 53 41 21 00' 02DD   310          .ASCIC  "!AS!_!AC!_!AS!_!XL  !AS  !/"
53 41 21 20 20 4C 58 21 5F 21 53 41  02E9
                        2F 21 20 20  02F5
```

```
                                        1B  02DD
                                            02F9   311 T_SHOWDEVB:                                                ;                                  [07]
21 20 20 43 41 21 5F 21 53 41 21 00'        02F9   312          .ASCIC  "!AS!_!AC  !/"                             ;                                  [07]
                                        2F  0305
                                        0C  02F9
70 20 72 6F 66 20 65 75 6C 61 56 00'        0306   313 T_PNF:   .ASCIC  'Value for prompt "!AC" not found in script!/'  ;                       [09]
20 22 43 41 21 22 20 74 70 6D 6F 72         0312
6E 69 20 64 6E 75 6F 66 20 74 6F 6E         031E
          2F 21 74 70 69 72 63 73 20        032A
                                        2C  0306
68 74 20 67 6E 69 70 70 69 6B 53 00'        0333   314 T_SKIP: .ASCIC  'Skipping this line!/'  ;                                              [09]
          2F 21 65 6E 69 6C 20 73 69        033F
                                        14  0333
                                            0348   315 T_SHOULDBE:
      65 62 20 64 6C 75 6F 68 73 00'        0348   316          .ascic  \should be\                                ;                                  [09]
                                        09  0348
                                            0352   317 t_yesno:                                                  ;                                  [09]
73 65 59 2C 6F 4E 0000035A'010E0000'        0352   318          .ascid  \No,Yes\                                   ;                                  [09]
                                            C360   319 T_STRING:
6E 6F 20 43 41 21 00000368'010E0000'        0360   320          .ASCID  "!AC one of !AS!/!AC? "
41 21 2F 21 53 41 21 20 66 6F 20 65         036E
                        20 3F 43            037A
                                            037D   321 Q_HUB:                                                    ;                                  [11]
         42 55 48 00000385'010E0000'        037D   322          .ASCID  "HUB"                                     ; Name of HUB device               [11]
                                            0388   323
                                            0388   324 part_list:                                               ; List of generic name parts       [17]
                                        05' 0388   325          .byte   10$-part_list                             ; Offset to 'generic'              [17]
                                        05' 0389   326          .byte   10$-part_list                             ;   same                           [17]
                                        0D' 038A   327          .byte   20$-part_list                             ; Offset to 'controller'           [17]
                                        18' 038B   328          .byte   30$-part_list                             ; Offset to 'unit'                 [17]
                                        1D' 038C   329          .byte   40$-part_list                             ; Offset to ''                     [17]
            63 69 72 65 6E 65 67 00'        038D   330 10$:     .ascic  \generic\                                 ;                                  [17]
                                    07  038D
72 65 6C 6C 6F 72 74 6E 6F 63 00'           0395   331 20$:     .ascic  \controller\                              ;                                  [17]
                                    0A  0395
                  74 69 6E 75 00'           03A0   332 30$:     .ascic  \unit\                                    ;                                  [17]
                                    04  03A0
                                        00' 03A5   333 40$:     .ascic  \\                                        ;                                  [17]
                                        00  03A5
                                            03A6   334
                                            03A6   335 CENTRAL_BUS:                                                     ; List of 'central bus' names
                                        05  03A6   336          .BYTE   5                                                 ; Number of bus names      [21]
                     49 42 53 00'       03A7   337          .ASCIC  "SBI"                                             ; STAR - CPU type 1
                                    03  03A7
                     49 4D 43 00'       03AB   338          .ASCIC  "CMI"                                             ; COMET - CPU type 2
                                    03  03AB
                  44 41 48 50 00'       03AF   339          .ASCIC  "PHAD"                                            ; PHysical Address Decoder [06]
                                    04  03AF
                  53 55 42 41 00'       03B4   340          .ASCIC  "ABUS"                                            ; VENUS - CPU type 4       [21]
                                    04  03B4
                     42 55 48 00'       03B9   341          .ASCIC  "HUB"                                             ; system HUB device        [06]
                                    03  03B9
                                            03BD   342
```

```
                03BD          .SBTTL  Work Psect Declarations
                03BD
            00000000          .PSECT  Work, Noshr, Noexe, Wrt, Long   ;                                [07]
    00000000  0000          FLG$V_BRIEF     = 0                 ; Flag for brief form of commands        [07]
                0000  COMM_FLAGS:                               ;                                        [07]
    00000001  0000          .BLKB   1                           ; Storage area for /BRIEF flag           [07]
                0001  abort_attach:                             ;                                        [09]
    00000002  0001          .blkb   1                           ; Use to abort ATTACH cmd on ^C          [09]
                0002  alloc:                                    ;                                        [31]
          00  0002          .byte   0                           ; Allocation class flag                  [31]
                0003  range:                                    ;                                        [31]
          00  0003          .byte   0                           ; Range for allocation flag              [31]
                0004  q_adapter::                               ;                                        [11]
    0000000C  0004          .blkq   1                           ; Descriptor for link device name        [11]
                000C  temp_store:                                                                      ;G:9
    00000010  000C          .blkl   1                                                                  ;G:9
```

```
                     0010   361          .SBTTL  DSV$ATTACH      Attach command processor
                     0010   362
                 00000000   363          .PSECT  Code, Exe, Shr, NoWrt, Long              ;                    [10]
                     0000   364 ;++
                     0000   365 ;
                     0000   366 ; Functional Description:
                     0000   367 ;       This routine is called from CLI to finish processing an
                     0000   368 ;       ATTACH command
                     0000   369 ;
                     0000   370 ; Calling Sequence:
                     0000   371 ;       BSBW    DSV$ATTACH
                     0000   372 ;
                     0000   373 ; Input Parameters:
                     0000   374 ;       NONE
                     0000   375 ;
                     0000   376 ; Implicit Inputs:
                     0000   377 ;       R2      Base of CLI data base, DS$GL_CLIBASE
                     0000   378 ;       CLI$Q_FILE      Length,address of remainder of command line
                     0000   379 ;
                     0000   380 ; Output Parameters:
                     0000   381 ;       NONE
                     0000   382 ;
                     0000   383 ; Implicit Outputs:
                     0000   384 ;       New Device Parameter Block or modifications to previous one
                     0000   385 ;
                     0000   386 ; Side Effects:
                     0000   387 ;       NONE
                     0000   388 ;--
```

```
                                0000   390
                                0000   391 DSV$ATTACH::
        00000001'EF   94        0000   392         clrb    L^abort_attach          ; Clear abort flag                      [09]
                                0006   393         $ds_cntrlc_s    set_abort       ; Set up ^C handler                     [09]
                     00   DD    0006               PUSHL   #0
        000000FD'EF  DF    0008               PUSHAL  set_abort
  00000000'9F  02    FB    000E               CALLS   #2, a#DS$CNTRLC
        00000000'EF  16    0015   394         JSB     SCRIPT$FLUSH            ; Flush scripts if this console command [10]
            7E  53   7D    001B   395         MOVQ    R3,-(SP)                ; Save R3 and R4
                55   DD    001E   396         PUSHL   R5                      ; Save R5
            53  7E   DE    0020   397         MOVAL   -(SP),R3                ; Allocate location and set ptr
  5E    00000064 8F   C2    0023   398         SUBL2   #100,SP                 ; Allocate stack buffer
            54  6E   9E    002A   399         MOVAB   (SP),R4                 ;  and set pointer to it
  5E    0000006C 8F   C2    002D   400         SUBL2   #108,SP                 ; Allocate ATTACH prompt string
     04 AE    08 AE   9E    0034   401         MOVAB   8(SP),4(SP)             ; Set string descriptor address
            55  6E   7E    0039   402         MOVAQ   (SP),R5                 ;  and remember where it is
                3C   BB    003C   403         PUSHR   #^M<R2,R3,R4,R5>        ; Save regs destroyed by MOVC3
              08 A2   28    003E   404         MOVC3   CLI$Q_FILE(R2),-        ; Copy command line to local
            64   0C B2           0041   405          aCLI$Q_FILE+4(R2),(R4)  ;  buffer (for later appends)
                3C   BA    0044   406         POPR    #^M<R2,R3,R4,R5>        ; Restore registers
          0C A2   64   9E    0046   407         MOVAB   (R4),CLI$Q_FILE+4(R2)   ; Correct buffer pointer
        65   0064 8F   3C    004A   408 10$:     MOVZWL  #100,(R5)               ; Re-set length of buffer
                65   7F    004F   409         PUSHAQ  (R5)                    ; Address of return prompt
              08 A2   7F    0051   410         PUSHAQ  CLI$Q_FILE(R2)          ; Address of command descriptor
        010A'CF   02   FB    0054   411         CALLS   #2,W^DSX$ATTACH         ; Try to do the attach
              74 50   E8    0059   412         BLBS    R0,40$                  ; Exit loop if success
  0000FE00'EF   1F   E0    005C   413         bbs     #dsa$v_apt, -           ; If APT mode, abort processing         [09]
                30         0063   414          L^dsa$gl_flags, 20$    ; ...with error (fatal)                 [09]
                15   E1    0064   415         BBC     #DS$V_SCRIPT,-          ;                                       [07]
          2B 00000000'EF           0066   416          L^DS$GL_FLAGS,30$      ; Branch if not script running          [07]
                04   B5    9F    006C   417         PUSHAB  a4(R5)                  ; Address of prompt ASCIC string        [07]
             00000306'EF   9F    006F   418         PUSHAB  L^T_PNF                 ; Edit string for prompt not found      [07]
                01   DD    0075   419         pushl   #ds$k_printf            ; Use Printf                            [13]
                15   DD    0077   420         pushl   #ds$k_type_command_err  ; And command error type code          [13]
        00000000'EF   04   FB    0079   421         CALLS   #4, L^DSX$PRINT         ; Type it                               [13]
             00000333'EF   9F    0080   422         PUSHAB  L^T_SKIP                ; Edit string for skipping              [07]
                01   DD    0086   423         pushl   #ds$k_printf            ; Use Printf                            [13]
                15   DD    0088   424         pushl   #ds$k_type_command_err  ; Command error also                    [13]
        00000000'EF   03   FB    008A   425         CALLS   #3, L^DSX$PRINT         ; Type it.                              [13]
                  003C   31    0091   426         BRW     40$                     ; exit                                  [07]
                  0052   31    0094   427 20$:    brw     50$                     ; Waystation to 50$                     [13]
                         0097   428 30$:                                         ;                                       [07]
             00000000'EF   16    0097   429         jsb     L^kb_check              ; Check for ^C's                        [09]
          2C 00000001'EF   E8    009D   430         blbs    abort_attach, 40$       ; Return directly if so                 [09]
                04   B5    9F    00A4   431         PUSHAB  a4(R5)                  ; Address of prompt ASCIC string
7E  00000064 8F   08 A2   C3    00A7   432         SUBL3   CLI$Q_FILE(R2),#100,-(SP); Length of remaining buffer
                63   DF    00B0   433         PUSHAL  (R3)                    ; Where to return length read
            51   08 B244   9E    00B2   434         MOVAB   aCLI$Q_FILE(R2)[R4],R1  ; Point to end of current string
                81   20    90    00B7   435         MOVB    #^A" ",(R1)+            ; Separate new argument by space
              08 A2   B6    00BA   436         INCW    CLI$Q_FILE(R2)          ; Increase length
                61   9F    00BD   437         PUSHAB  (R1)                    ; Calculate address to read to
        00000000'EF   04   FB    00BF   438         CALLS   #4,L^DS_SUPERLINE       ; Read a line
              07 50   E9    00C6   439         BLBC    R0,40$                  ; Can't continue if failure
           08 A2   63   C0    00C9   440         ADDL2   (R3),CLI$Q_FILE(R2)     ; Set new input line size
                FF7A   31    00CD   441         BRW     10$                     ; Try again                             [13]
  5E    000000D4 8F   C0    00D0   442 40$:     ADDL2   #212,SP                 ; Clear off buffers
            55   8E   D0    00D7   443         MOVL    (SP)+,R5                ; Restore registers
```

```
                53   8E   7D  00DA          MOVQ    (SP)+,R3
                                00DD          $ds_cntrlc_s                    ; Clear out handler                    [09]
                     00   DD  00DD                  PUSHL   #0
                     00   DD  00DF                  PUSHL   #0
        00000000'9F  02   FB  00E1                  CALLS   #2, a#DS$CNTRLC
                          05  00E8          RSB                             ; Return to caller
                                00E9
                                00E9  50$:                                  ; ATTACH error in APT mode             [09]
                                00E9          errsup_s        msgadr=a4(r5)   ; ERRSUP w/ returned message           [09]
            00000000'EF   DF  00E9                  PUSHAL  $MODULE
                   04 B5  DF  00EF                  PUSHAL  a4(r5)
                     01   DD  00F2                  PUSHL   #$ER
        00000000'EF  03   FB  00F4                  CALLS   #3, DS_ERRSUP
                     D3   11  00FB          brb     40$                     ; Exit routine if continued            [09]
                                00FD
                                00FD  ;
                                00FD  ; Control C handler for DSV$ATTACH.  Intercepts ^C and causes quick return
                                00FD  ; to CLI.  Putting a KB_CHECK call in the loop would work as well: but this
                                00FD  ; way causes routine cleanup, so CONTINUE will not continue ATTACH cmd.
                                00FD  ;
                        0000    00FD  .entry  set_abort, ^M<>                ;                                      [09]
        00000001'EF  01   90  00FF          movb    #1, abort_attach        ; Set abort flag                       [09]
                50   01   D0  0106          movl    #1, r0                  ; Return success                       [09]
                          04  0109          ret                             ;                                      [09]
```

```
                        010A   462          .SBTTL  DSX$ATTACH      Process ATTACH command
                        010A   463
                        010A   464 ;++
                        010A   465 ;
                        010A   466 ; Functional description:
                        010A   467 ;       This routine is called from DSV$ATTACH to translate an attach
                        010A   468 ;       command line using a PTABLE descriptor, and create a PTABLE
                        010A   469 ;       from this data.  If any argument is missing or invalid, it will
                        010A   470 ;       return with the given prompt string buffer set to the correct
                        010A   471 ;       prompt to get more data from the operator, and the input line
                        010A   472 ;       descriptor re-set to exclude the remaining (if any) incorrect
                        010A   473 ;       text of the input line.
                        010A   474 ;
                        010A   475 ; Calling sequence:
                        010A   476 ;       DS$ATTACH (cmdline, [prompt])
                        010A   477 ;
                        010A   478 ; Input Parameters:
                        010A   479 ;       cmdline         pointer to command line descriptor
                        010A   480 ;       prompt          pointer to ASCIC-type buffer (with length of remainder
                        010A   481 ;                       of buffer in byte 1)
                        010A   482 ;
                        010A   483 ; Implicit inputs:
                        010A   484 ;       none
                        010A   485 ;
                        010A   486 ; Output Parameters
                        010A   487 ;       prompt          the buffer is filled with an ASCIC string suitable for
                        010A   488 ;                       prompting the operator for the next ATTACH parameter.
                        010A   489 ; Implicit outputs:
                        010A   490 ;       none
                        010A   491 ;
                        010A   492 ; Side Effects:
                        010A   493 ;       If successfull, creates a PTABLE in dynamic pool space
                        010A   494 ;
                        010A   495 ; Register usage:
                        010A   496 ;       R4,R5           Descriptor of remaining input string
                        010A   497 ;       R6              Pointer to prompt string descriptor
                        010A   498 ;       R8              Pointer to input descriptor
                        010A   499 ;       R10             PTABLE descriptor "PC" for decode
                        010A   500 ;       R11             Pointer to PTABLE
                        010A   501 ;
                        010A   502 ; Return codes:
                        010A   503 ;       DS$_BADTYPE     Device type argument is bad (no PTABLE descriptor)
                        010A   504 ;       DS$_BADLINK     Device given as link is not attached
                        010A   505 ;       DS$_ILLUNIT     Unit number not given or was illegal (too big)
                        010A   506 ;       DS$_DEVNAME     Invalid device name given
                        010A   507 ;       SS$_BADPARAM    Number was in bad radix or out of range
                        010A   508 ;       SS$_INSFARG     Too few arguments given to complete ATTACH
                        010A   509 ;--
```

```
                        010A   511
                        010A   512          $OFFSET 4,POSITIVE,<-
                        010A   513                  <ATC$_CMDLINE,4>,-
                        010A   514                  <ATC$_PROMPT,4>>
                        010A                .SAVE   LOCAL_BLOCK
                        010A                .PSECT $ABS$ ABS
                00000004 FE70               .=4
                        0004        ATC$_CMDLINE:
                00000008 0004               .BLKB   4
                        0008        ATC$_PROMPT:
                0000000C 0008               .BLKB   4
                       0000010A             .RESTORE
                        010A   515
                        010A   516          $OffSet 0, NEGATIVE, <-          ; Offsets for local data                  [17]
                        010A   517                  <Name_Flags, 1>, -       ; .. Name flags from $DS_$NAME            [17]
                        010A   518                  <Parent_Controller, 1>, - ; .. Controller letter of parent         [17]
                        010A   519                  <Max_Unit, 2>, -         ; .. Maximum unit number allowed          [17]
                        010A   520                  <PtDesc, 4>, -           ; .. Address of Ptable descriptor         [17]
                        010A   521                  <Parent_Ptable, 4>, -    ; .. Address of Parent Ptable             [17]
                        010A   522                  <Generic, 10>, -         ; .. Generic name string from $DS_$NAME   [17]
                        010A   523                  <Local_Block> -          ; Size of local block                     [17]
                        010A   524                  >                        ;                                         [17]
                        010A                .SAVE   LOCAL_BLOCK
                        010A                .PSECT $ABS$ ABS
                00000000 FE70               .=0
                FFFFFFFF 0000               .BLKB   -1
                        FFFF        Name_Flags:
                FFFFFFFE FFFF               .BLKB   -1
                        FFFE        Parent_Controller:
                FFFFFFFC FFFE               .BLKB   -2
                        FFFC        Max_Unit:
                FFFFFFF8 FFFC               .BLKB   -4
                        FFF8        PtDesc:
                FFFFFFF4 FFF8               .BLKB   -4
                        FFF4        Parent_Ptable:
                FFFFFFEA FFF4               .BLKB   -10
                        FFEA        Generic:
                FFFFFFE6 FFEA               .BLKB   -4
                        FFE6        Local_Block:
                       0000010A             .RESTORE
                        010A   525
                    0FFC 010A   526 .ENTRY  DSX$ATTACH, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        010C   527
5E   E6 AD   9E 010C    528          MovAB   Local_Block(Fp), Sp      ; Allocate local storage block            [17]
58   04 BC   7E 0110    529          MOVAQ   @ATC$_CMDLINE(AP),R8     ; Point to command line desc.
     02 A8   B4 0114    530          CLRW    2(R8)                    ; Clear length used so far
   54   68   7D 0117    531          MOVQ    (R8),R4                  ; Set up register descriptor
   02   6C   91 011A    532          CMPB    (AP),#2                  ; Both arguments given?
        06   19 011D    533          BLSS    10$                      ; No--fake prompt buffer
56   08 BC   7E 011F    534          MOVAQ   @ATC$_PROMPT(AP),R6      ; Set up prompt pointer
        07   12 0123    535          BNEQ    20$                      ; Branch if specified
        7E   3F 0125    536 10$:     PUSHAW  -(SP)                    ; Allocate fake buffer
        02   DD 0127    537          PUSHL   #2                       ; Finish descriptor
   56   6E   DE 0129    538          MOVAL   (SP),R6                  ; Save pointer to descriptor
                012C    539 20$:
        5B   D4 012C    540          CLRL    R11                      ; Clear PT address
        1D   11 012E    541          BRB     GET_TYPE                 ; Get hardware device type
```

```
                                0130
                                0130  BAD_TYPE:
        00000164'EF    9F       0130          PUSHAB  L^T_TYPE                  ; Set prompt to return                [10]
        00000007'EF    7F       0136          PUSHAQ  L^T_INVALID               ; Error message                       [10]
  00000B7E'EF    02    FB       013C          CALLS   #2,L^FORMAT_PROMPT        ; Format it for return                [10]
  50    006600E8 8F    D0       0143          MOVL    #DS$_BADTYPE,R0           ; Set return code
                032E    31       014A          BRW     ERROR_EXIT                ; Return
                                014D
                                014D  GET_TYPE:
  53    00000164'EF    9E       014D          MOVAB   L^T_TYPE,R3               ; Address of prompt string            [10]
        00000B56'EF    16       0154          Jsb     L^GET_LINE                ; Get something from operator         [18]
                36 50  E9       015A          BLBC    R0,10$                    ; Exit if error on input
                                015D
        00000000'EF    16       015D          Jsb     L^SCAN$SYMBOL             ; Scan the hardware type              [16]
            57    50   3C       0163          MOVZWL  R0,R7                     ; Save length of type string
                  C8   13       0166          BEQL    BAD_TYPE                  ; Branch if not there
                                0168
            7E    50   7D       0168          MOVQ    R0,-(SP)                  ; Save descriptor of type
        000011E6'EF    16       016B          Jsb     L^LOC$PTDESC              ; Locate descriptor R0/R1             [18]
                  BD   13       0171          BEQL    BAD_TYPE                  ; Branch if not found
                                0173
         02 A8    57   A0       0173          ADDW2   R7,2(R8)                  ; Add type string to length used
            5A    50   D0       0177          MOVL    R0,R10                    ; Save address of hardware type block
         F8 AD    50   D0       017A          MovL    R0, PtDesc(Fp)            ; Save it again                       [17]
            50    8A   9A       017E          MOVZBL  (R10)+,R0                 ; Get length of ASCIC name
            5A    50   C0       0181          ADDL    R0,R10                    ; Skip name
            51    8A   9A       0184          MOVZBL  (R10)+,R1                 ; Fetch size of PT
            51    1A   C0       0187          ADDL2   #HPE$C_EPB_SIZE,R1        ; make room for long device name      [28]
        00000000'EF    16       018A          Jsb     L^EXE$ALONONPAGED         ; Allocate a block for this PT        [18]
                03 50  E8       0190          BLBS    R0,20$                    ; Branch if enough memory
                                0193  10$:
                02E5   31       0193          BRW     ERROR_EXIT
                                0196  20$:
            5B    52   D0       0196          MOVL    R2,R11                    ; Copy address of block
  50    51    FD 8F    78       0199          ASHL    #-3,R1,R0                 ; Number of quadwords present
                  82   7C       019E  30$:    CLRQ    (R2)+                     ; Clear one
               FB 50   F5       01A0          SOBGTR  R0,30$                    ; Branch if not done yet
         08 AB    51   B0       01A3          MOVW    R1,HP$W_SIZE(R11)         ; Save true length of block
                                01A7
         52    26 AB   9E       01A7          MOVAB   HP$T_TYPE(R11),R2         ; Point to type storage location
            82    6E   90       01AB          MOVB    (SP),(R2)+                ; Store length
         57    04 AE   D0       01AE          MOVL    4(SP),R7                  ; Set address to copy from
            50    6E   3C       01B2          MOVZWL  (SP),R0                   ; Set length to copy
                  8E   7C       01B5          CLRQ    (SP)+                     ; Remove descriptor
                                01B7
            82    87   90       01B7  40$:    MOVB    (R7)+,(R2)+               ; Copy byte
               FA 50   F5       01BA          SOBGTR  R0,40$                    ; Copy whole string
                  1D   11       01BD          BRB     GET_LINK                  ; Get device linking this to the processor
                                01BF  BAD_LINK:
        00000158'EF    9F       01BF          PUSHAB  L^T_LINK                  ; Set prompt to return                [10]
        00000007'EF    7F       01C5          PUSHAQ  L^T_INVALID               ; Format string                       [10]
  00000B7E'EF    02    FB       01CB          CALLS   #2,L^FORMAT_PROMPT        ; Format prompt for output            [10]
  50    006600F0 8F    D0       01D2          MOVL    #DS$_BADLINK,R0           ; Set return code
                029F    31       01D9          BRW     ERROR_EXIT
                                01DC
                                01DC  GET_LINK:
  53    00000158'EF    9E       01DC          MOVAB   L^T_LINK,R3               ; Prompt line                         [10]
```

```
                00000B56'EF   16  01E3   599          Jsb     L^GET_LINE                      ; Get some input                        [18]
                      03 50   E8  01E9   600          BLBS    R0,10$                          ; Branch if it worked
                       028C   31  01EC   601          BRW     ERROR_EXIT                      ; Take error exit
                              01EF   602 10$:
                00000000'EF   16  01EF   603          Jsb     L^SCAN$DEVICE                   ; Get device name of connection         [16]
                    57   50   3C  01F5   604          MOVZWL  R0,R7                           ; Save length of link name
                         C5   13  01F8   605          BEQL    BAD_LINK                        ; Bad device
                              01FA   606
                              01FA   607 ;
                              01FA   608 ; See if the link name typed is one of the pre-defined 'central bus' names.
                              01FA   609 ; If so, make sure it is the correct one (e.g., 'SBI' for STAR, 'CMI' for
                              01FA   610 ; COMET).  If it is not the correct one, return an error.
                              01FA   611 ;
                              01FA   612
                    50   50   3C  01FA   613          MOVZWL  R0,R0                           ; isolate device name length
                    007C 8F   BB  01FD   614          PUSHR   #^M<R2,R3,R4,R5,R6>             ; Save registers
                         55   D4  0201   615          CLRL    R5                              ; name index
              53    000003A6'EF   9E  0203   616          MOVAB   L^CENTRAL_BUS, R3               ; Point to table of names               [10]
                    54   83   9A  020A   617          MOVZBL  (R3)+, R4                       ; Get number of names in table
                    52   83   9A  020D   618 20$:     MOVZBL  (R3)+, R2                       ; Get length of next name
                         55   D6  0210   619          INCL    R5                              ; Increment type code tried
                    56   51   D0  0212   620          MOVL    R1, R6                          ; Copy input string address
                    50   52   B1  0215   621          CMPW    R2, R0                          ; Same length as input?
                         05   13  0218   622          BEQL    40$                             ; If yes, compare strings
                    53   52   C0  021A   623 30$:     ADDL2   R2, R3                          ; Skip rest of string
                         4A   11  021D   624          BRB     60$                             ; Try next string
                    86   63   91  021F   625 40$:     CMPB    (R3), (R6)+                     ; Compare bytes of strings
                         F6   12  0222   626          BNEQ    30$                             ; If no match, try next string
                         53   D6  0224   627          INCL    R3                              ; Advance pointer
                      F6 52   F5  0226   628          SOBGTR  R2, 40$                         ; Loop through string
                04    55   91  0229   629          CMPB    R5, #4                          ; Is is implementation-specific?        [21]
                         2D   14  022C   630          BGTR    50$                             ; No, accept it unconditionally         [23]
          0000FE17'EF    55   91  022E   631          CMPB    R5, L^DSA$GL_SID+3              ; Match--is it correct one?
                         24   13  0235   632          BEQL    50$                             ; Yes, continue
                    007C 8F   BA  0237   633          POPR    #^M<R2,R3,R4,R5,R6>             ; Restore registers
                00000158'EF   9F  023B   634          PUSHAB  L^T_LINK                        ; Prompt string to return               [10]
                      7E   50   7D  0241   635          MOVQ    R0, -(SP)                       ; Descriptor of input name              [10]
                00000109'EF   7F  0244   636          PUSHAQ  L^T_MAIN_BUS                    ; 'There is no !AD on this type pro.'   [10]
          00000B7E'EF    04   FB  024A   637          CALLS   #4, L^FORMAT_PROMPT             ; Format it for return                  [10]
          006600F0 8F   D0  0251   638          MOVL    #DS$_BADLINK, R0                ; Set return code
                    0220   31  0258   639          BRW     ERROR_EXIT                      ; Error!!!
                    007C 8F   BA  025B   640 50$:     POPR    #^M<R2,R3,R4,R5,R6>             ; Good match.  Restore regs.
                 02 A8   57   A0  025F   641          ADDW2   R7, 2(R8)                       ; Increase string length used
                      F4 AD   D4  0263   642          ClrL    Parent_Ptable(Fp)               ; Device is connected to HUB            [17]
                    0163   31  0266   643          BRW     GET_NAME                        ; Get device name generic               [17]
                      A1 54   F5  0269   644 60$:     SOBGTR  R4, 20$                         ; Try next name string
                    007C 8F   BA  026C   645          POPR    #^M<R2,R3,R4,R5,R6>             ; Restore registers
                              0270   646                                                                                          ;G:9
                    52   01   CE  0270   647          mnegl   #1, r2                          ; Set wildcard link for loc$ptable      [09]
                00001197'EF   16  0273   648          Jsb     L^LOC$PTABLE                    ; Locate P-table for link               [18]
                   F4 AD   51   D0  0279   649          MovL    R1, Parent_Ptable(Fp)           ; Save address of parent                [17]
                         03   12  027D   650          BNEQ    70$                             ; Found it
                      FF3D   31  027F   651          BRW     BAD_LINK                        ; Can't find link P-table
                 02 A8   57   A0  0282   652 70$:     ADDW2   R7,2(R8)                        ; Increase length used so far
                 20 AB   51   D0  0286   653          MOVL    R1,HP$A_LINK(R11)               ; Store link address
              18 AB    1C A1   D0  028A   654          MOVL    HP$A_DVA(R1),HP$A_DEVICE(R11) ; Initialize device address
                              028F   655                                                                                           ;G:9
```

```
                    013A   31  028F   656          BrW     GET_NAME                        ; Get new device name           [27]
```

```
                                0292   658          .Subtitle       Process invalid generic name error
                                0292   659
                                0292   660 BAD_UNIT_EXC:
          0000014C'EF   9F      0292   661          PUSHAB  L^T_NAME                    ; Set prompt text                       [17]
          000000D7'EF   7F      0298   662          pushaq  t_unit_excess               ; Unit number too large                 [17]
    00000B7E'EF   02    FB      029E   663          CALLS   #2,L^FORMAT_PROMPT          ; Format and store                      [17]
50  00660100 8F         D0      02A5   664          movl    #ds$_illunit, r0            ; Illegal unit number                   [17]
                01CC    31      02AC   665          brw     error_exit                  ; Error                                 [17]
```

```
                                02AF  .sbttl  Process invalid generic name error
                                02AF  ;
                                02AF  ; BAD_NAME is branched to if parse_device routine fails.  The Ptable
                                02AF  ; descriptor is analysed to print out what the problem was, and what the
                                02AF  ; correct format of the name ought to be.  The format it uses is 'GGan'
                                02AF  ; where 'GG' is replaced by the Ptable descriptor's generic string (or
                                02AF  ; 'gg' if there is none), 'a' represents that the name must have a
                                02AF  ; controller letter, and 'n' represents that the name must have a unit
                                02AF  ; number.  If the PTD$V_NAME generic flag is set, and PTD$V_CONTROLLER and
                                02AF  ; PTD$V_UNIT flags are not, then the name is  an alphanumeric node name of
                                02AF  ; 6 or fewer characters.  This is represented  in the error message as
                                02AF  ; 'name' instead of the 'GGan' form. If PTD$V_NAME and PTD$V_CONTROLLER
                                02AF  ; and/or PTD$V_UNIT flag is set then the generic name is represented as
                                02AF  ; name$GGan in the error message. If the alloc flag is set then the
                                02AF  ; generic name is represented as '$(1-255)$GGan' in the error message.
                                02AF  ; If the range flag is set then the range is represented as decimal
                                02AF  ; 1 to 255.
                                02AF  ;
                                02AF  ;
                                02AF  ;
                                02AF  ; Inputs:
                                02AF  ;       R2 => return code from parse_device.
                                02AF  ;               0 = no generic found.
                                02AF  ;               1 = bad generic prefix
                                02AF  ;               2 = bad controller (none where should be, or present where
                                02AF  ;                       shouldn't be)
                                02AF  ;               3 = bad unit (none where should be, or present where
                                02AF  ;                       shouldn't be)
                                02AF  ;               4 = bad node name
                                02AF  ;               5 = bad allocation class name
                                02AF  ;
                                02AF  ; This code segment will cause loading of the return prompt string, and error
                                02AF  ; exit from DSX$ATTACH.
                                02AF  ;
                                02AF
                                02AF  bad_name:                                  ;                                        [17]
          04    52    D1        02AF          cmpl    r2, #4                     ; Validate status                        [17]
                03    15        02B2          bleq    10$                        ; branch if OK                           [17]
          52    04    D0        02B4          movl    #4, r2                     ; Knock value down to legal range        [17]
50   00000388'EF42    90        02B7  10$:    movb    part_list[r2], r0          ; Get byte offset of message             [17]
52   00000388'EF40    9E        02BF          movab   part_list[r0], r2          ; Get address of message                 [17]
       55    FF AD    9A        02C7          movzbl  Name_Flags(Fp), r5         ; Get generic flags                      [17]
          53    6E    9E        02CB          movab   (sp), r3                   ; Save stack pointer                     [17]
       1B 55    02    E1        02CE          bbc     #ptd$v_name, r5, 20$       ; Branch if normal name format           [17]
                                02D2  ;       BITL    #PTD$M_CONTROLLER!PTD$M_UNIT, R5 ; Anything else to check for ? [26]
                                02D2  ;       BNEQ    20$                        ; Branch if more to check.               [26]
  7E   656D616E 8F    D0        02D2          movl    #^A"name", -(sp)           ; Push string                            [17]
          7E    04    90        02D9          movb    #4, -(sp)                  ; ...make it ASCIC                       [17]
  51   0000038C'EF    9A        02DC          movzbl  part_list+4, r1            ; Get to the part_list null ASCIC        [17]
51   00000388'EF41    9E        02E3          movab   part_list[r1], r1          ; ...string                              [17]
                5A    11        02EB          brb     87$                        ; Go do formatting, etc.                 [17]
                                02ED  20$:                                       ; Normal name formatting                 [17]
                54    D4        02ED          clrl    r4                         ; Length of string created               [17]
       06 55    00    E1        02EF          bbc     #ptd$v_unit, r5, 30$       ; Branch if no unit needed               [17]
       7E    6E 8F    90        02F3          movb    #^A"n", -(sp)              ; Make string on stack                   [17]
                54    D6        02F7          incl    r4                         ; Count the byte                         [17]
          55    01    E1        02F9  30$:    bbc     #ptd$v_controller, -       ; Branch if                              [17]
```

```
                 12      02FC                  r5, 50$                  ; ...no controller needed                [17]
                 5E   D7 02FD          DECL    SP                       ; Allocate one byte                      [17]
                 54   D6 02FF          incl    r4                       ; And count it                           [17]
           55    04   E1 0301          bbc     #ptd$v_inherit_Con, -    ; Does device inherit parent's name?     [19]
                 06      0304                  r5, 40$                  ; ...if not, use 'a'                     [17]
        6E   FE AD   90 0305          MovB    Parent_Controller(Fp), -; Get parent's controller letter          [17]
                        0309                  (Sp)                     ; .. onto stack                          [17]
                 04   12 0309          bneq    50$                      ; If valid, use it                       [17]
        6E   61 8F   90 030B  40$:    movb    #^A"a", (sp)             ; Put controller format on stack         [17]
        51   EA AD   9E 030F  50$:    movab   Generic(Fp), r1          ; Get address of generic string          [17]
           50    61   9A 0313          MovZBL  (R1), R0                 ; Get length of generic string           [26]
                 0B   13 0316          beql    70$                      ; Branch if null (use 'gg')              [17]
         7E   6140   90 0318  60$:    movb    (r1)[r0], -(sp)          ; Push on bytes in right order           [26]
                 54   D6 031C          incl    r4                       ; Count 'em                              [17]
              F7 50   F5 031E          sobgtr  r0, 60$                  ; Do them all                            [17]
                 08   11 0321          brb     80$                      ; Finish off                             [17]
      7E   6767 8F   B0 0323  70$:    movw    #^A"gg", -(sp)           ; Push on 'gg' string                    [17]
           54    02   C0 0328          addl2   #2, r4                   ; And set length                         [17]
   0D 0A AB    02   E1 032B  80$:    BBC     #HP$V_NAME,HP$B_FLAGS(R11),85$  ; Check if name needed           [28]
           7E    24   90 0330          MOVB    #^A"$",-(SP)             ; Set delimiter                          [26]
7E   656D616E 8F   D0 0333          MOVL    #^A"name", -(SP)         ; Set string                             [26]
           54    05   C0 033A          ADDL2   #5, R4                   ; Adjust length                          [26]
           7E    54   90 033D  85$:    movb    r4, -(sp)                ; Make string on stack be ASCIC          [17]
51   000000CD'EF   9E 0340          movab   t_part_of, r1            ; say 'Error in ? part of name'          [17]
  54 00000003'EF   E8 0347  87$:    blbs    range,110$               ; See if error in allocation range       [31]
  25 00000002'EF   E8 034E          blbs    alloc,100$               ; See if error in allocation class       [31]
     0000014C'EF   9F 0355  90$:    pushab  L^t_name                 ; Prompt string                          [17]
              04 AE   9F 035B          pushab  4(sp)                    ; Correct name format string             [17]
                 61   9F 035E          pushab  (r1)                     ; ' part of ' or ''                      [17]
                 62   9F 0360          pushab  (r2)                     ; Failing part of the name               [17]
     0000002E'EF   7F 0362          pushaq  t_bad_name               ; Format string for the whole mess       [17]
   0B7E'CF    05   FB 0368          calls   #5, w^format_prompt      ; Format the output prompt               [17]
50   00660108 8F   D0 036D          movl    #ds$_devname, r0         ; Set failure reason                     [17]
           5E    53   D0 0374          movl    r3, sp                   ; Restore stack                          [17]
              0101   31 0377          brw     error_exit               ; return                                 [17]
     0000014C'EF   9F 037A  100$:   pushab  L^t_name                 ; Prompt string                          [31]
     000000A0'EF   9F 0380          pushab  t_alloc                  ; Correct name format string             [31]
                 61   9F 0386          pushab  (r1)                     ; ' part of ' or ''                      [31]
                 62   9F 0388          pushab  (r2)                     ; Failing part of the name               [31]
     0000002E'EF   7F 038A          pushaq  t_bad_name               ; Format string for the whole mess       [31]
   0B7E'CF    05   FB 0390          calls   #5, w^format_prompt      ; Format the output prompt               [31]
50   00660108 8F   D0 0395          movl    #ds$_devname, r0         ; Set failure reason                     [31]
           5E    53   D0 039C          movl    r3, sp                   ; Restore stack                          [31]
              00D9   31 039F          brw     error_exit               ; return                                 [31]
     0000014C'EF   9F 03A2  110$:   pushab  L^t_name                 ; Prompt string                          [31]
     000000A8'EF   9F 03A8          pushab  t_range                  ; Correct name format string             [31]
     000000BA'EF   9F 03AE          pushab  t_unrange                ; Failing part of the name               [31]
     0000006B'EF   7F 03B4          pushaq  t_bad_range              ; Format string for the whole mess       [31]
   0B7E'CF    04   FB 03BA          calls   #4, w^format_prompt      ; Format the output prompt               [31]
50   00660108 8F   D0 03BF          movl    #ds$_devname, r0         ; Set failure reason                     [31]
           5E    53   D0 03C6          movl    r3, sp                   ; Restore stack                          [31]
              00AF   31 03C9          brw     error_exit               ; return                                 [31]
```

```
                                    03CC   777          .sbttl  Get name processing
                                    03CC   778
                                    03CC   779 GET_NAME:
                          00C1   30 03CC   780          BsbW    Get_Attributes          ; Set up rest of local block          [17]
                 53 0000014C'EF   9E 03CF   781          MOVAB   L^T_NAME,R3             ; prompt string                       [10]
                      00000B56'EF   16 03D6   782          Jsb     L^GET_LINE              ; Get something                       [18]
                            03 50   E8 03DC   783          BLBS    R0,10$                  ; Branch if not error
                             0099   31 03DF   784          BRW     ERROR_EXIT              ; Take error exit
                                    03E2   785 10$:
                             0178   30 03E2   786          BsbW    Parse_Device            ; Get this new device name            [17]
                               03   12 03E5   787          BNeq    15$                     ; Skip if OK                          [17]
                             FEC5   31 03E7   788          Brw     Bad_Name                ; Branch if invalid                   [17]
                       57    08 AB   3C 03EA   789 15$:    MOVZWL  HP$W_SIZE(R11),R7       ;                                     ;G:9
                    59    5B   57   C1 03EE   790          ADDL3   R7,R11,R9               ; address of end of p-table           ;G:9
                    57    59   04   C3 03F2   791          SUBL3   #4,R9,R7                ; address to store hpe$a_epb          ;G:9
                    67    59   1A   C3 03F6   792          SUBL3   #HPE$C_EPB_SIZE,R9,(R7) ; beginning of extended ptable        ;G:9
                          59   67   D0 03FA   793          MOVL    (R7),R9                 ; address of extended p-table         ;G:9
                               52   D4 03FD   794          ClrL    R2                      ; Initial unit number                 [17]
                 11 FF AD      00   E1 03FF   795          BbC     #Ptd$V_Unit, -          ; Branch if no unit allowed           [17]
                                    0404   796                   Name_Flags(Fp), 25$     ;                                     [17]
                 52    50   10   10   EF 0404   797          EXTZV   #16,#16,R0,R2           ; Get unit number                     [26]
                    7E    FC AD   3C 0409   798          MOVZWL  MAX_UNIT(FP),-(SP)      ; Set for longword compare            [26]
                          8E   52   D1 040D   799          CMPL    R2,(SP)+                ; Check full unit number              [26]
                               03   1B 0410   800          BLeqU   25$                     ; Branch if OK                        [17]
                             FE7D   31 0412   801          BrW     Bad_Unit_Exc            ; Branch to error routine             [17]
                 00FF 8F       52   B1 0415   802 25$:    CMPW    R2, #^XFF               ; To small to fit in a byte?          ;G:9
                               04   14 041A   803          BGTR    30$                                                           ;G:9
                    0B AB      52   90 041C   804          MovB    R2, Hp$B_Drive(R11)     ; Move to Ptable                      [17]
                    14 A9      52   B0 0420   805 30$:    MOVW    R2,HPE$W_EXT_DRIVE(R9)  ; Move max unit # to extended ptable  ;G:9
                       50      50   9A 0424   806          MOVZBL  R0,R0                   ; Isolate length of device name
                    02 A8      50   A0 0427   807          ADDW2   R0,2(R8)                ; Increase length used so far
              05 0A AB         02   E1 042B   808          BBC     #HP$V_NAME,HP$B_FLAGS(R11),35$ ; if long device name?        [28]
                       52      69   DE 0430   809          MOVAL   HPE$T_DEVICE(R9),R2                                           ;G:9
                               04   11 0433   810          BRB     37$                     ; continue as before                  [28]
                                    0435   811 35$:
                    52    0C AB      9E 0435   812          MOVAB   HP$T_DEVICE(R11),R2     ; address to store device name in pt
                                    0439   813 37$:
                 6B    50      01   C1 0439   814          ADDL3   #1,R0,HP$Q_DEVICE(R11)  ; Store length of ascic name
                   04 AB      62   9E 043D   815          MOVAB   (R2),HP$Q_DEVICE+4(R11) ; Store address of "_" in device name
                   82    5F   8F   90 0441   816          MOVB    #^A"_",(R2)+            ; Insert "_" physical device indicator
                                    0445   817
                       82      81   90 0445   818 40$:    MOVB    (R1)+,(R2)+             ; Copy byte
                         FA    50   F5 0448   819          SOBGTR  R0,40$                  ; Copy whole string
                          5A   03   C0 044B   820          ADDL    #3,R10                  ; Point past max unit number and name
                00000000'EF      16 044E   821          JSB     ONLY_ATTACH             ; check need for vector offsets  [35]
```

```
                                0454   823
          7E    54    7D      0454   824        MOVQ    R4,-(SP)                ; length/address of remaining input
                66    7F      0457   825        PUSHAQ  (R6)                    ; Address of prompt descriptor
             04 AE    7F      0459   826        PUSHAQ  4(SP)                   ; Address of descriptor
                5B    DD      045C   827        PUSHL   R11                     ; Base of P-table
                5A    DD      045E   828        PUSHL   R10                     ; Start of PT-desc
     070C'CF    04    FB      0460   829        CALLS   #4,W^PT$INTERPRET       ; Do PT-descriptor interpretation
   02 A8    02 AE    A0      0465   830        ADDW2   2(SP),2(R8)             ; Include length used
```

```
                5E   08   C0  046A   832          ADDL    #8,SP                    ; Remove descriptor
                  0B 50   E9  046D   833          BLBC    R0,ERROR_EXIT            ; Take error exit if failed
        00001240'EF   6B   FA  0470   834          CALLG   (R11),L^INS$PTABLE       ; Insert it                    [10]
                  01 50   E9  0477   835          BLBC    R0,ERROR_EXIT            ; Failed take error exit
                          04  047A   836          RET
                              047B   837
                              047B   838 ERROR_EXIT:
                     50   DD  047B   839          PUSHL   R0                       ; Save error code
```

```
                68    02 A8    3C  047D   841          MOVZWL  2(R8),(R8)              ; Return length used
                   50    5B    D0  0481   842          MOVL    R11,R0                  ; Copy address of buffer
                         06    13  0484   843          BEQL    10$                     ; Skip if none
                00000000'EF    16  0486   844          Jsb     L^EXE$DEANONPAGED       ; Release it                    [18]
                                   048C   845 10$:
                         50 8ED0   048C   846          POPL    R0                      ; Restore error code            ;G:9
                                04 048F   847          RET
```

```
                                0490          .Subtitle       Get device attributes
                                0490
                                0490  ;++
                                0490  ; Functional Description:
                                0490  ;
                                0490  ;       This routine is called by DSX$ATTACH to set up the local
                                0490  ;       storage block with the information on generic name to
                                0490  ;       enforce, maximum number of units, etc.  If running from
                                0490  ;       a script, APT, or if there is no $DS_$NAME directive in
                                0490  ;       the PTABLE descriptor, it will cause behavior similar to
                                0490  ;       the previous ATTACH--i.e., there will be no enforced
                                0490  ;       generic names, and the PTD$V_UNIT bit will be set iff
                                0490  ;       MAX_UNIT is greater than 0.
                                0490  ;
                                0490  ; Inputs and Outputs for this routine are in the local storage
                                0490  ; block allocated at the beginning of DSX$ATTACH.  It also references
                                0490  ; both the parent Ptable (if any) and the current Ptable descriptor,
                                0490  ; through pointers stored in said local block.
                                0490  ;
                                0490  ;--
                                0490                                                                          ;G:9
                                0490  Get_Attributes:                          ;                                  [17]
                00FC 8F    BB   0490          PushR   #^M<R2,R3,R4,R5,R6,R7>  ; Save some registers                [17]
                  FF AD    94   0494          ClrB    Name_Flags(Fp)          ; No flags set                      [17]
                  EA AD    94   0497          ClrB    Generic(Fp)             ; No enforced generic yet           [17]
                  FE AD    94   049A          ClrB    Parent_Controller(Fp)   ; No enforced controller yet        [17]
            57    F8 AD    D0   049D          MovL    PtDesc(Fp), R7          ; Get Ptable descriptor start       [17]
                50    87    9A   04A1          MovZBL  (R7)+, R0               ; Get length of type string[17]   ;G:9
        FC AD   01 A740     9B   04A4          MovZBW  1(R7)[R0], Max_Unit(Fp) ; Fetch maximum unit number         [17]
                      04    13   04AA          BEqlU   10$                     ; Branch if old ATTACH said no unit[17] ;G:9
            FF AD   01      90   04AC          MovB    #Ptd$M_Unit, -          ; If there is no NAME statement,    [17]
                                04B0                   Name_Flags(Fp)          ; .. this device must have unit     [17]
          57    04 A740     9E   04B0  10$:    MovAB   4(R7)[R0], R7           ; Skip all of $DS_$INITIALIZE       [19]
            80 8F   87      91   04B5          CmpB    (R7)+, #PD$_Start       ; Next byte ought to be start code  [17]
                    76      12   04B9          BNeq    20$                     ; Skip out if something's wrong     [17]
            8E 8F   67      91   04BB          CMPB    (R7),#PD$_EPB           ; Extended P-table Block?         ;G:9
                    15      12   04BF          BNEQ    12$                     ; NO.  Branch.                    ;G:9
                    57      96   04C1          INCB    R7                      ; Skip PD$_EPB byte               ;G:9
            FC AD   87      B0   04C3          MOVW    (R7)+,MAX_UNIT(FP)                                        ;G:9
        FFFF 8F   FC AD     B1   04C7          CMPW    MAX_UNIT(FP),#^XFFFF    ; If # is 65535 do not decrement  ;G:9
                                04CD                                          ; because that is number to be used ;G:9
                                04CD                                          ; [65536 will not fit into a word   ;G:9
                                04CD                                          ; so 65535 is used].              ;G:9
                    04      13   04CD          BEQLU   11$                                                       ;G:9
            FF AD   01      90   04CF          MovB    #Ptd$M_Unit, -          ; If there is no NAME statement,    [17]
                                04D3                   Name_Flags(Fp)          ; .. this device must have unit     [17]
                                04D3                                                                          ;G:9
                  FC AD     B7   04D3  11$:    DECW    MAX_UNIT(FP)                                              ;G:9
                                04D6                                                                          ;G:9
                                04D6  12$:                                                                    ;G:9
            8D 8F   87      91   04D6          CmpB    (R7)+, #PD$_Name        ; Is the next byte start of NAME?   [17]
                    55      12   04DA          BNeq    20$                     ; Skip out if it's not              [17]
            FF AD   87      90   04DC          MovB    (R7)+, Name_Flags(Fp)   ; Set real name flags               [17]
                50    87    9A   04E0          MovZBL  (R7)+, R0               ; Get length of generic name        [17]
            EA AD   50      90   04E3          MovB    R0, Generic(Fp)         ; Save length away                  [17]
  EB AD   09   00   67      50   2C   04E7    MovC5   R0, (R7), -             ; Copy generic string               [17]
                                04EE                   #0, -                   ; .. zero filled                    [17]
```

```
                                  04EE                  #9, Generic+1(Fp)       ; .. into local storage                [17]
          50   F4 AD    D0       04EE          MovL    Parent_Ptable(Fp), R0    ; Get parent Ptable                    [17]
                    3D    13       04F2          BEql    20$                      ; Exit if attached to HUB              [17]
          FF AD    18    93       04F4          BitB    #Ptd$M_Inherit, -        ; Check if device should inherit either [19]
                                  04F8                  Name_Flags(Fp)          ; .. name or controller                [19]
                    37    13       04F8          BEql    20$                      ; Skip if niether bit is set           [19]
      55   04 A0    01    C1       04FA          AddL3   #1, Hp$Q_Device+4(R0), -; Get address of parent name (- '_')   [17]
                                  04FF                  R5                      ;                                      [17]
        54    60    01    A3       04FF          SubW3   #1, Hp$Q_Device(R0), R4 ; Get length of same                   [17]
              54    54    3C       0503          MovZWL  R4, R4                   ; Zero extend length                   [17]
        00000000'EF    16       0506          Jsb     L^Scan$Alpha             ; Scan over alphabetic                 [17]
                    23    13       050C          BEql    20$                      ; Exit if none there (??)              [17]
        52    50    01    C3       050E          SubL3   #1, R0, R2               ; Get length minus 1                   [17]
              02    52    D1       0512          CmpL    R2, #2                   ; Generic should be at least 2 chars   [17]
                    1A    1F       0515          BLssU   20$                      ; Don't use if not valid generic form  [17]
      05 FF AD    04    E1       0517          BbC     #Ptd$V_Inherit_Con, -    ; Don't use controller letter if       [19]
                                  051C                  Name_Flags(Fp), 15$     ; .. not supposed to                   [19]
        FE AD    6142    90       051C          MovB    (R1)[R2], -              ; Store parent's controller letter     [17]
                                  0521                  Parent_Controller(Fp)   ;                                      [17]
      0B FF AD    03    E1       0521  15$:    BbC     #Ptd$V_Inherit_Pre, -    ; Don't use prefix if shouldn't        [19]
                                  0526                  Name_Flags(Fp), 20$     ; .. inherit it                        [19]
          EA AD    52    90       0526          MovB    R2, Generic(Fp)          ; If valid, use parent name as generic [17]
EB AD   09   00    61    52    2C  052A          MovC5   R2, (R1), -              ; Copy to Generic, fill with 0         [17]
                                  0531                  #0, -                   ;                                      [17]
                                  0531                  #9, Generic+1(Fp)       ;                                      [17]
        00000000'EF    E9       0531  20$:    BlbC    L^Ds$GB_Inhibit_Naming,-; If should not enforce naming         [19]
                    0B            0537                  40$                     ;                                      [19]
              EA AD    94       0538  30$:    ClrB    Generic(Fp)              ; .. then no enforced generics         [17]
              FE AD    94       053B          ClrB    Parent_Controller(Fp)    ; .. or controller                     [17]
      FF AD   FA 8F    8A       053E          BicB2   #^C<Ptd$M_Unit ! -       ; .. and allow only unit and name      [17]
                                  0543                  Ptd$M_Name>, -          ; .... flags                           [17]
                                  0543                  Name_Flags(Fp)          ;                                      [17]
          00FC 8F    BA       0543  40$:    PopR    #^M<R2,R3,R4,R5,R6,R7>   ; Restore registers                    [17]
                    05       0547          Rsb                              ; Return                               [17]
                                  0548
```

```
                                0548  .sbttl  parse device name
                                0548  ;
                                0548  ; Small routine to save hassle in checking for alphas
                                0548  ;
                                0548
                                0548  if_alpha:                                  ;                                        [17]
                  50    D4      0548          clrl    r0                        ; Assume bad                             [17]
                  54    D5      054A          tstl    r4                        ; Any string left?                       [17]
                  0E    15      054C          bleq    10$                       ; Branch if not                          [17]
          41 8F   65    91      054E          cmpb    (r5), #^A"A"              ; Check for alpha                        [17]
                  08    19      0552          blss    10$                       ; Branch if not                          [17]
          5A 8F   65    91      0554          cmpb    (r5), #^A"Z"              ;                                        [17]
                  02    14      0558          bgtr    10$                       ; Branch if not                          [17]
                  50    D6      055A          incl    r0                        ; It's OK                                [17]
                        05      055C  10$:    rsb                               ; Return                                 [17]
                                055D
                                055D  ;
                                055D  ; This routine parses a device name.  It's output is the same as that for
                                055D  ; the SCAN$DEVICE routine in PARSE.  However, it takes into account the
                                055D  ; generic name and flags of the NAME statement in parsing. This routine
                                055D  ; can handle device names of the form 'node$ggan'.
                                055D  ;
                                055D  ; NOTE: This code assumes that if the PTD$V_INHERITED flag is set, the link
                                055D  ;       device will have a controller letter.  If there is ever a chain of
                                055D  ;       devices for which this might not be true, the code must be changed.
                                055D  ;
                                055D  ; Inputs:
                                055D  ;       R4 => remaining length of input string
                                055D  ;       R5 => address of remaining input string
                                055D  ;
                                055D  ; Outputs:
                                055D  ;       R0 => low word length of device name which was parsed (0 if none)
                                055D  ;               high word has device unit number (-1 if none)
                                055D  ;       R1 => address of device name (input R5)
                                055D  ;       R2 => Success code:
                                055D  ;               0 = OK (no errors found: may not be any name at all)
                                055D  ;               1 = error in generic prefix
                                055D  ;               2 = error in controller
                                055D  ;               3 = error in unit number
                                055D  ;               4 = error in node name format
                                055D  ;               5 = error in allocation class format
                                055D  ;       R4 => updated past device name (if any)
                                055D  ;       R5 => updated past device name (if any)
                                055D  ;
                                055D  parse_device::                            ;                                        [17]
          00FF 8F       BB      055D          pushr   #^M<r0,r1,r2,r3,r4,r5,r6,r7>    ; Save registers                 [17]
       57   FF AD       9A      0561          MovZBL  Name_Flags(Fp), R7        ; Get flags conveniently local           [17]
       04 AE    55      D0      0565          movl    r5, 4(sp)                 ; Set saved R1 to start of string        [17]
00000002'EF     00      90      0569          movb    #0,alloc                  ; Clear allocation flag                  [31]
00000003'EF     00      90      0570          movb    #0,range                  ; Clear range flag                       [31]
       03 57    02      E1      0577          bbc     #ptd$v_name, R7,1$        ; Handle if it's node name               [31]
             0134       31      057B          brw     120$                      ; long device name set up                [31]
                                057E  1$:                                       ; input for allocation class             [31]
    50   54     01      C3      057E          SUBL3   #1,R4,R0                  ; use for index                          [28]
         24     85      91      0582          CMPB    (R5)+,#^A"$"              ; check for allocation format            [31]
                15      12      0585          BNEQ    3$                        ; else check for cluster                 [31]
00000002'EF     01      90      0587          MOVB    #1,ALLOC                  ; set flag for allocation format         [31]
```

```
                30   85   91  058E   999          CMPB    (R5)+,#^A"0"                    ; if leading zeros then error                [31]
                     09   12  0591  1000          BNEQ     3$                             ; no error                                   [31]
                          0593  1001                                                      ; input for allocation class                 [31]
             08 AE   05   D0  0593  1002          movl    #5,8(sp)                        ; error flag for allocation format           [31]
                     52   D4  0597  1003          clrl    r2                              ; flag for un-enforced prefix devices        [31]
                   00FD   31  0599  1004          brw     100$                            ; clean up                                   [31]
                          059C  1005 3$:                                                  ; used to be 1$                              [31]
                24   85   91  059C  1006          CMPB    (R5)+,#^A"$"                    ; check for cluster device name format       [28]
                     0F   13  059F  1007          BEQL    4$                              ; if cluster format then 4$                  [28]
                  F8 50   F5  05A1  1008          SOBGTR  R0,3$                           ; do for all name length                     [28]
          00000002'EF   94  05A4  1009          CLRB    ALLOC                           ; not allocation only one $                  [31]
           55   04 AE   D0  05AA  1010          MOVL    4(SP),R5                        ; restore input string pointer               [28]
                     1A   11  05AE  1011          BRB     5$                              ; continue as before                         [28]
                          05B0  1012 4$:                                                  ; used to be 2$                              [31]
             0A AB   04   88  05B0  1013          BISB2   #HP$M_NAME,HP$B_FLAGS(R11) ; we have a long device name                      [28]
           55   04 AE   D0  05B4  1014          MOVL    4(SP),R5                        ; restore input string pointer               [28]
             08 AE   04   D0  05B8  1015          MOVL    #4,8(SP)                        ; condition error code                       [28]
                53   0A   D0  05BC  1016          MOVL    #10,R3                          ; max length for'node'                       [28]
          00000002'EF   95  05BF  1017          TSTB    ALLOC                           ; see if allocation flag set                 [32]
                     15   12  05C5  1018          BNEQ    15$                             ; check for digits between $                 [31]
                   00EF   31  05C7  1019          BRW     122$                            ; go to 122$                                 [28]
                          05CA  1020 5$:
             08 AE   01   D0  05CA  1021          movl    #1, 8(sp)                       ; Assume prefix error in saved R2            [17]
                     52   D4  05CE  1022          clrl    r2                              ; Flag for un-enforced prefix devices        [17]
           51   EA AD   9E  05D0  1023 10$:       MovAB   Generic(Fp), R1                 ; Get address of enforced generic            [17]
                50   81   9A  05D4  1024          MovZBL  (R1)+, R0                       ; .. and it's length                         [17]
                     2A   12  05D7  1025          bneq    20$                             ; Branch if OK to check                      [17]
                   00C4   31  05D9  1026          brw     110$                            ; No enforced generic for this device        [17]
                     55   D6  05DC  1027 15$:      incl    r5                              ; Get past '$'                               [31]
                     54   D7  05DE  1028          decl    r4                              ; Decrement length                           [31]
          00000000'EF   16  05E0  1029          jsb     L^scan$decimal                  ; Check for numbers                          [31]
                     07   12  05E6  1030          bneq    17$                             ; Branch around error jump                   [31]
             08 AE   05   D0  05E8  1031          movl    #5,8(sp)                        ; Error code for allocation class            [31]
                   00AA   31  05EC  1032          brw     100$                            ; Error found clean it up                    [31]
      0000FFFF 8F   50   D1  05EF  1033 17$:      cmpl    r0,#65535                       ; See if within range                        [32]
                     08   14  05F6  1034          bgtr    18$                             ; Intermediate step                          [31]
                01   50   D1  05F8  1035          cmpl    r0,#1                           ; Check range                                [32]
                     03   19  05FB  1036          blss    18$                             ; Intermediate step                          [31]
                   00C1   31  05FD  1037          brw     125$                            ; Do letter check for cluster name           [31]
                   0084   31  0600  1038 18$:      brw     95$                             ; Skip check for letters                     [31]
                     54   D5  0603  1039 20$:      tstl    r4                              ; Any string left?                           [17]
                     0C   15  0605  1040          bleq    25$                             ; Error if none                              [17]
                85   81   91  0607  1041          cmpb    (r1)+, (r5)+                    ; Compare a byte                             [17]
                     07   12  060A  1042          bneq    25$                             ; No good if mismatch                        [17]
                     54   D7  060C  1043          decl    r4                              ; Count the byte we checked                  [17]
                  F2 50   F5  060E  1044          sobgtr  r0, 20$                         ; And loop til we're done                    [17]
                     03   11  0611  1045          brb     30$                             ;                                            [31]
                   0083   31  0613  1046 25$:      BRW     100$                            ; Branches in 20$ out of range               [32]
                08 AE   D6  0616  1047 30$:      incl    8(sp)                           ; Increment saved R2                         [17]
                57   01   E0  0619  1048          bbs     #ptd$v_controller, -            ; Is there supposed to be a controller?      [17]
                     0F       061C  1049                   R7, 50$                         ; ...if so, check for it                     [17]
                  FF28   30  061D  1050          bsbw    if_alpha                        ; Is next character an alpha?                [17]
                1F 50   E9  0620  1051          blbc    r0, 60$                         ; If not, then we're OK--check unit          [17]
                73 52   E9  0623  1052          blbc    r2, 100$                        ; If enforced prefix, that's an error        [17]
                     54   D7  0626  1053          decl    r4                              ; Skip over the backtracked byte             [17]
                     55   D6  0628  1054          incl    r5                              ;                                            [17]
                     16   11  062A  1055          brb     60$                             ; Check unit number                          [17]
```

```
                        062C  1056 50$:                                    ; Need controller letter                    [17]
          FF19   30     062C  1057          bsbw    if_alpha              ; Check next character                      [17]
        67 50    E9     062F  1058          blbc    r0, 100$              ; Error if not a letter!                    [17]
           54    D7     0632  1059          decl    r4                    ; Want to lose this                         [17]
        50   85  9A     0634  1060          movzbl  (r5)+, r0             ; Get controller letter                     [17]
     56   FE AD  9A     0637  1061          MovZBL  Parent_Controller(Fp), -; Get correct letter                       [17]
                        063B  1062           R6                           ; .. into R6                               [17]
           05    13     063B  1063          beql    60$                   ; Not specified, check unit                 [17]
        56 50    D1     063D  1064          cmpl    r0, r6                ; Compare them                              [17]
           57    12     0640  1065          bneq    100$                  ; Error if not equal                        [17]
                        0642  1066 60$:                                    ; Check for unit                            [17]
        08 AE    D6     0642  1067          incl    8(sp)                 ; Increment assumed error code              [17]
00000000'EF      16     0645  1068          Jsb     L^scan$decimal        ; Is there a number here?                   [17]
           09    12     064B  1069          bneq    70$                   ; Branch if was                             [17]
     48 57  00  E0     064D  1070          bbs     #ptd$v_unit, R7, 100$ ; Error if wasn't, and needed one           [17]
        50  01  CE     0651  1071          mnegl   #1, r0                ; Set no unit number code in R0             [17]
           04    11     0654  1072          brb     80$                   ; Join common                               [17]
                        0656  1073 70$:                                    ; Found a unit number                       [17]
     3F 57  00  E1     0656  1074          bbc     #ptd$v_unit, R7, 100$ ; Error if was, and didn't want one         [17]
     02 AE  50  B0     065A  1075 80$:     movw    r0, 2(sp)             ; Set high word of saved R0                 [17]
  6E 55   14 AE  A3     065E  1076          subw3   20(sp), r5, (sp)      ; Calculate length of name found            [17]
           54    D5     0663  1077          tstl    r4                    ; Check for more characters                 [17]
           13    15     0665  1078          bleq    90$                   ; Really OK either way...                   [17]
           54    D7     0667  1079          decl    r4                    ; Assume we'll want next character          [17]
        3A 85    91     0669  1080          cmpb    (r5)+, #^A":"         ; Is it a colon?                            [17]
           0C    13     066C  1081          beql    90$                   ; That's cool, a colon feels just fine      [17]
           54    D6     066E  1082          incl    r4                    ; Don't want next character, elsewise       [17]
        20 75    91     0670  1083          cmpb    -(r5), #^A" "         ; Is next a space?                          [17]
           05    13     0673  1084          beql    90$                   ; Can be terminated by space                [17]
        09 65    91     0675  1085          cmpb    (r5), #^x9            ; Is it a tab?                              [17]
           1F    12     0678  1086          bneq    100$                  ; Otherwise, may be hex 'unit number'       [17]
                        067A  1087 90$:                                    ; Here's the successful return stuff        [17]
     10 AE  54    7D     067A  1088          movq    r4, 16(sp)            ; Update saved pointers                     [17]
        00FF 8F  BA     067E  1089          popr    #^M<r0,r1,r2,r3,r4,r5,r6,r7>    ; Restore the registers     [17]
           52    D4     0682  1090          clrl    r2                    ; No error found                            [17]
           50    B5     0684  1091          tstw    r0                    ; Check for anything found                  [17]
                 05     0686  1092          rsb                           ; And return happily                        [17]
                        0687  1093 95$:                                    ; Set codes for range error                 [31]
00000003'EF      01     0687  1094          movb    #1,range              ; Set range error flag                      [31]
00000002'EF      00     068E  1095          movb    #0,alloc              ; Clear allocation error flag               [31]
        08 AE    05     0695  1096          movzbl  #5,8(sp)              ; Sets code                                 [31]
                        0699  1097 100$:                                   ; Error return                              [17]
        00FF 8F  BA     0699  1098          popr    #^M<r0,r1,r2,r3,r4,r5,r6,r7>    ; Restore registers         [17]
           50    7C     069D  1099          clrq    r0                    ; Act as if nothing found                   [17]
                 05     069F  1100          rsb                           ; and return                                [17]
                        06A0  1101 110$:                                   ; Here if no enforced prefix                [17]
        53   02    D0     06A0  1102          movl    #2, r3                ; Set min length for generic                [17]
           004E  30     06A3  1103          bsbw    scan_alpha            ; Check string                              [17]
             F1  19     06A6  1104          blss    100$                  ; If too few, error                         [17]
             55  D7     06A8  1105          decl    r5                    ; Backup for controller check               [17]
             54  D6     06AA  1106          incl    r4                    ;                                           [17]
        52   01    D0     06AC  1107          movl    #1, r2                ; set flag                                  [17]
           FF64  31     06AF  1108          brw     30$                   ; Check controller                          [17]
                        06B2  1109 120$:                                   ; Node name check                           [17]
     08 AE  04    D0     06B2  1110          movl    #4, 8(sp)             ; Give back node name error code            [17]
        53  06    D0     06B6  1111          movl    #6, r3                ; Maximum length of node name               [17]
                        06B9  1112 122$:                                   ; Do letter check for cluster name          [31]
```

```
                45   10  06B9          bsbb    scan_alphanum           ; Check it                                  [24]
                DC   1A  06BB          BGtrU   100$                    ; Branch if error                           [17]
                50   B5  06BD          TstW    R0                      ; Check for zero length, too                [17]
                D8   15  06BF          BLeq    100$                    ; Error if nothing                          [17]
                         06C1  125$:                                   ; 122$ broken up to input allocation        [31]
            57  03   D3  06C1          BITL    #PTD$M_CONTROLLER!PTD$M_UNIT, R7 ; Check for more field proc'g     [26]
                27   13  06C4          BEQL    135$                    ; Branch if not                             [26]
                54   D5  06C6          TSTL    R4                      ; Anything left ?                           [26]
                1E   15  06C8          Bleq    129$                    ; Branch if not                             [26]
            24  65   91  06CA          CMPB    (R5),#^A"$"             ; should be '$'                             [28]
                1E   12  06CD          BNEQ    135$                    ; Branch if not.                            [26]
                55   D6  06CF          INCL    R5                      ; Skip '$'.                                 [26]
                54   D7  06D1          DECL    R4                      ; Skip character                            [26]
                34   BB  06D3          PUSHR   #^M<R2,R4,R5>           ; save registers                            [28]
        52  0C AB   9E  06D5          MOVAB   HP$T_DEVICE(R11),R2     ; pointer to device name                    [28]
        82  5F 8F   90  06D9          MOVB    #^A"_",(R2)+            ; insert '_' physical device indicator      [28]
                         06DD  128$:
            20  65   91  06DD          CMPB    (R5),#^A" "             ; are we finished ?                         [28]
                06   13  06E0          BEQL    129$                    ; then go to 129$                           [28]
            82  85   90  06E2          MOVB    (R5)+,(R2)+             ; copy byte                                 [28]
              F5 54   F5  06E5          SOBGTR  R4,128$                 ; do for whole string                       [28]
                         06E8  129$:
                34   BA  06E8          POPR    #^M<R2,R4,R5>           ; restore registers                         [28]
               FEDD  31  06EA          BRW     5$                      ;Check for other enforced functions.        [26]
        50  FF 8F   98  06ED  135$:    CvtBL   #-1, R0                 ; Set to return 'no unit number'            [17]
               FF66  31  06F1          BRW     80$                     ; OK if less than                           [17]
                         06F4
                         06F4  ;
                         06F4  ; Small routine to call scan$alpha and compare return length
                         06F4  ;
                         06F4
                         06F4  scan_alpha:                             ;                                           [17]
                08   BB  06F4          pushr   #^M<r3>                 ; Save the compare length                   [17]
        00000000'EF  16  06F6          Jsb     L^scan$alpha            ; Call PARSE scan routine                   [17]
            8E  50   D1  06FC          cmpl    r0, (sp)+               ; Do comparison                             [17]
                     05  06FF          rsb                             ; Return                                    [17]
                         0700  ;
                         0700  ; Small routine to call scan$alphanum and compare return length
                         0700  ;
                         0700
                         0700  scan_alphanum:                          ;                                           [24]
                08   BB  0700          pushr   #^M<r3>                 ; Save the compare length                   [24]
        00000000'EF  16  0702          Jsb     L^scan$alphanum         ; Call PARSE scan routine                   [24]
            8E  50   D1  0708          cmpl    r0, (sp)+               ; Do comparison                             [24]
                     05  070B          rsb                             ; Return                                    [24]
                         070C
```

```
                        1160          .SBTTL  PT_INTERPRET    Decode PT-descriptor
                        1161
                        1162 ;++
                        1163 ; FUNCTIONAL DESCRIPTION:
                        1164 ;
                        1165 ;       This routine is used to interpret the P-table descriptor
                        1166 ;       which builds the P-table for the current unit.
                        1167 ;       Each OP code is examined and executed.  If during the
                        1168 ;       execution input text is required, the input string is examined
                        1169 ;       if the string is empty the input routine is called to get input.
                        1170 ;
                        1171 ; CALLING SEQUENCE:
                        1172 ;
                        1173 ;       PT$INTERPRET (INITPC, BASE, TEXT, PROMPT)
                        1174 ;
                        1175 ; INPUT PARAMETERS:
                        1176 ;
                        1177 ;       INITPC  Initial PC for interpretation
                        1178 ;       BASE    Base for data storage
                        1179 ;       TEXT    Address of Quad descriptor of text
                        1180 ;       PROMPT  Address of buffer to receive ASCIC prompt text
                        1181 ;               if failure
                        1182 ;                                                              ;G:9
                        1183 ; IMPLICIT INPUTS:      NONE
                        1184 ;
                        1185 ; OUTPUT PARAMETERS:
                        1186 ;
                        1187 ; IMPLICIT OUTPUTS:
                        1188 ;
                        1189 ;       P-table effected by intpretation of PT-descriptor
                        1190 ;
                        1191 ; REGISTER USAGE:
                        1192 ;
                        1193 ;       R0      Length of compare string/scratch
                        1194 ;       R1      Address of compare string/scratch
                        1195 ;       R2      Current index in PD_STRING/scratch
                        1196 ;       R3      Remaining length in PD_STRING/scratch
                        1197 ;       R4      Length of input string
                        1198 ;       R5      Address of input string
                        1199 ;       R6      Address of prompt string descriptor
                        1200 ;       R8      Address of input descriptor
                        1201 ;       R9      Current VALUE
                        1202 ;       R10     Virtural PC for P-table intpretation
                        1203 ;       R11     Base of P-table
                        1204 ;
                        1205 ; COMPLETION CODES:
                        1206 ;
                        1207 ;       SS$_NORMAL
                        1208 ;       SS$_BADPARAM
                        1209 ;       SS$_INSFARG
                        1210 ;--
```

```
                        070C  1212
                        070C  1213          $OFFSET 4,POSITIVE,< -
                        070C  1214                  <IARG$_PC,4>,-
                        070C  1215                  <IARG$_BASE,4>,-
                        070C  1216                  <IARG$_TEXT,4>,-
                        070C  1217                  <IARG$_PROMPT,4>>
                        070C                .SAVE   LOCAL_BLOCK
                        070C                .PSECT $ABS$ ABS
00000004                FFFF                .=4
                        0004        IARG$_PC:
00000008                0004                .BLKB   4
                        0008        IARG$_BASE:
0000000C                0008                .BLKB   4
                        000C        IARG$_TEXT:
00000010                000C                .BLKB   4
                        0010        IARG$_PROMPT:
00000014                0010                .BLKB   4
                    0000070C                .RESTORE
                        070C  1218
                        070C  1219          $OFFSET 0,NEGATIVE,< -
                        070C  1220                  <OLDPC,4>,-
                        070C  1221                  <DESC,8>,-
                        070C  1222                  <BUFFER,132>,-
                        070C  1223                  <LOCAL,0>>
                        070C                .SAVE   LOCAL_BLOCK
                        070C                .PSECT $ABS$ ABS
00000000                FFFF                .=0
FFFFFFFC                0000                .BLKB   -4
                        FFFC        OLDPC:
FFFFFFF4                FFFC                .BLKB   -8
                        FFF4        DESC:
FFFFFF70                FFF4                .BLKB   -132
                        FF70        BUFFER:
                        FF70        LOCAL:
                    0000070C                .RESTORE
                        070C  1224
```

```
                                070C
                          0FFC  070C  .ENTRY  PT$INTERPRET,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                070E
      5E    FF70 CD     9E      070E          MOVAB   LOCAL(FP),SP                ; Allocate space for local storage
   F8 AD    FF70 CD     9E      0713          MOVAB   BUFFER(FP),DESC+4(FP)       ; Address of buffer to descriptor
                                0719
         58    0C BC     7E      0719          MOVAQ   @IARG$_TEXT(AP),R8          ; Get address of text descriptor
            54    68     7D      071D          MOVQ    (R8),R4                     ; Fetch descriptor
            54    54     3C      0720          MOVZWL  R4,R4                       ; Be sure we only use word of length
         56    10 BC     7E      0723          MOVAQ   @IARG$_PROMPT(AP),R6        ; Set address of prompt string
         5A    04 AC     D0      0727          MOVL    IARG$_PC(AP),R10            ; Get initial PC
         5B    08 AC     D0      072B          MOVL    IARG$_BASE(AP),R11          ; Base of P-table
         6A    80 8F     91      072F          CMPB    #PD$_START,(R10)            ; Must start with start OP
                  27     12      0733          BNEQ    BADPARAM                    ; Branch if not
                                0735  INTERPRET:
         FC AD    5A     D0      0735          MOVL    R10,OLDPC(FP)               ; Incase of backup
                                0739          CASE    SRC=(R10)+,TYPE=B,LIMIT=#PD$_START,DISPLIST=< -
                                0739                  PD_START, -
                                0739                  PD_FINISH, -
                                0739                  PD_DECIMAL, -
                                0739                  PD_OCTAL, -
                                0739                  PD_HEXADECIMAL, -
                                0739                  PD_STRING, -
                                0739                  PD_LITERAL, -
                                0739                  PD_FETCH, -
                                0739                  PD_STORE, -
                                0739                  PD_COMPLEMENT, -
                                0739                  pd_add, -                   ; Add statment                          [09]
                                0739                  pd_logical, -               ; Logical prompt statment               [09]
                                0739                  pd_case, -                  ; Case statement                        [09]
                                0739                  pd_name, -                  ; Name directive                       ;G:9
                                0739                  pd_epb>                     ; Extended P-table Block [17]          ;G:9
   0E'  80 8F    8A     8F      0739          CASEB   (R10)+,#PD$_START,S^#<<30001$-30000$>/2>-1
                                073E  30000$:
                          0022' 073E          .SIGNED_WORD    PD_START-30000$
                          0027' 0740          .SIGNED_WORD    PD_FINISH-30000$
                          0118' 0742          .SIGNED_WORD    PD_DECIMAL-30000$
                          0149' 0744          .SIGNED_WORD    PD_OCTAL-30000$
                          0179' 0746          .SIGNED_WORD    PD_HEXADECIMAL-30000$
                          0215' 0748          .SIGNED_WORD    PD_STRING-30000$
                          002B' 074A          .SIGNED_WORD    PD_LITERAL-30000$
                          0030' 074C          .SIGNED_WORD    PD_FETCH-30000$
                          0041' 074E          .SIGNED_WORD    PD_STORE-30000$
                          0052' 0750          .SIGNED_WORD    PD_COMPLEMENT-30000$
                          0057' 0752          .SIGNED_WORD    pd_add-30000$
                          0071' 0754          .SIGNED_WORD    pd_logical-30000$
                          00E9' 0756          .SIGNED_WORD    pd_case-30000$
                          0106' 0758          .SIGNED_WORD    pd_name-30000$
                          0112' 075A          .SIGNED_WORD    pd_epb-30000$
                                075C  30001$:
                                075C  BADPARAM:
            50    14     D0      075C          MOVL    S^#SS$_BADPARAM,R0          ; Bad PT-desc
                         04      075F          RET                                 ; Return
                                0760
                                0760  PD_START:
            59    01     CE      0760          MNEGL   #1,R9                       ; Initialize to minus 1
                  D0     11      0763          BRB     INTERPRET                   ; Next
```

```
                                0765
                                0765  PD_FINISH:
                50   01   D0    0765          MOVL    S^#SS$_NORMAL,R0          ; It worked
                          04    0768          RET
                                0769
                                0769  PD_LITERAL:
                59   8A   D0    0769          MOVL    (R10)+,R9                 ; Set current value
                     C7   11    076C          BRB     INTERPRET                 ; Next!
                                076E
                                076E  PD_FETCH:
                50   8A   3C    076E          MOVZWL  (R10)+,R0                 ; Get byte offset
                51   8A   9A    0771          MOVZBL  (R10)+,R1                 ; Bit offset
                52   8A   9A    0774          MOVZBL  (R10)+,R2                 ; Field size
      59   6B40 52   51   EF    0777          EXTZV   R1,R2,(R11)[R0],R9        ; Get field
                     B6   11    077D          BRB     INTERPRET                 ; Next!
                                077F
                                077F  PD_STORE:
                50   8A   3C    077F          MOVZWL  (R10)+,R0                 ; Get byte offset
                51   8A   9A    0782          MOVZBL  (R10)+,R1                 ; Bit offset
                52   8A   9A    0785          MOVZBL  (R10)+,R2                 ; Field size
      6B40   52 51   59   F0    0788          INSV    R9,R1,R2,(R11)[R0]        ; Insert the field
                     A5   11    078E          BRB     INTERPRET
                                0790
                                0790  PD_COMPLEMENT:
                59   59   D2    0790          MCOML   R9,R9                     ; Complement value
                     A0   11    0793          BRB     INTERPRET                 ; Next function
                                0795
                                0795  PD_ADD:
                50   8A   3C    0795          MOVZWL  (R10)+,R0                 ; Get byte offset
                51   8A   9A    0798          MOVZBL  (R10)+,R1                 ; Get bit offset
                52   8A   9A    079B          MOVZBL  (R10)+,R2                 ; Field size
      53   6B40 52   51   EE    079E          EXTV    R1,R2,(R11)[R0],R3        ; Get sign extended field value
                53   59   C0    07A4          ADDL    R9,R3                     ; Add in current value
      6B40   52 51   53   F0    07A7          INSV    R3,R1,R2,(R11)[R0]        ; Put field back
                     86   11    07AD          BRB     INTERPRET                 ; Next function
                                07AF
                                07AF  pd_logical:                               ; Process LOGICAL                    [09]
                52   5E   D0    07AF          movl    sp, r2                    ; Save stack pointer                 [09]
                53   5A   D0    07B2          movl    r10, r3                   ; Get address of prompt string       [09]
           00000B56'EF   16    07B5          Jsb     L^get_line                ; Get data                           [18]
                 68 50   E9    07BB          blbc    r0, 50$                   ; Exit immediately if error          [09]
           00000000'EF   16    07BE          Jsb     L^scan$alpha              ; Read alpha string                  [18]
                     41   13    07C4          beql    40$                       ; Error if none                      [09]
                53   50   D0    07C6          movl    r0, r3                    ; Save length read                   [09]
                     59   D4    07C9          clrl    r9                        ; Assume 'no'                        [09]
           7E   4F4E 8F   B0    07CB          movw    #^A"NO", -(sp)            ; Push on 'no' to check              [09]
                7E   02   90    07D0          movb    #2, -(sp)                 ; Make it ASCIC                      [09]
             4E 8F   61   91    07D3          cmpb    (r1), #^A"N"              ; Does it really start with an 'n'?  [09]
                     0B   13    07D7          beql    10$                       ; Yep--verify                        [09]
                5E   52   D0    07D9          movl    r2, sp                    ; Restore stack                      [09]
                     59   D6    07DC          incl    r9                        ; It's a 'yes' or nothing            [09]
           53455903 8F   DD    07DE          pushl   #3+<^A"YES"a8>            ; Push on ASCIC 'yes'                [09]
                50   8E   9A    07E4  10$:    movzbl  (sp)+, r0                 ; Get the length                     [09]
                50   53   D1    07E7          cmpl    r3, r0                    ; Is input too long?                 [09]
                     1B   14    07EA          bgtr    40$                       ; If so, error                       [09]
                50   53   D0    07EC          movl    r3, r0                    ; Copy to destructable it            [09]
                8E   81   91    07EF  20$:    cmpb    (r1)+, (sp)+              ; Compare bytes                      [09]
```

```
                    13   12  07F2          bneq    40$                     ; Error!                                [09]
                 F8 50   F5  07F4          sobgtr  r0, 20$                 ; Loop on the check                     [09]
              5E    52   D0  07F7  30$:    movl    r2, sp                  ; Restore stack                         [09]
              50    8A   9A  07FA          movzbl  (r10)+, r0              ; Get length of prompt                  [09]
              5A    50   C0  07FD          addl2   r0, r10                 ; Skip it                               [09]
           02 A8    53   A0  0800          addw    r3, 2(r8)               ; We've just accepted more of string    [09]
                 FF2E   31  0804          brw     interpret               ; Back for next statement               [09]
              5E    52   D0  0807  40$:    movl    r2, sp                  ; Restore stack                         [09]
                    6A   9F  080A          pushab  (r10)                   ; Address of prompt                     [09]
        00000352'EF    7F  080C          pushaq  t_yesno                 ; 'yes,no'                              [09]
        00000348'EF    9F  0812          pushab  t_shouldbe              ; 'Should be'                           [09]
        00000360'EF    7F  0818          pushaq  t_string                ; Use invalid string argument           [09]
       0B7E'CF    04   FB  081E          calls   #4, w^format_prompt     ; Format prompt return                  [09]
              50    14   D0  0823          movl    #ss$_badparam, r0       ; Invalid parameter                     [09]
                    04   0826  50$:    ret                             ;                                       [09]
                         0827
                         0827  pd_case:                                 ; Case operation                        [09]
              50    8A   9A  0827          movzbl  (r10)+, r0              ; Count of cases                        [09]
                    0B   13  082A          beql    20$                     ; Exit if none (it's OK, though)        [09]
              51    8A   7D  082C  10$:    movq    (r10)+, r1              ; Get next case pair                    [09]
              59    51   D1  082F          cmpl    r1, r9                  ; Does this one match?                  [09]
                    06   13  0832          beql    30$                     ; If it does, do replacement            [09]
                 F5 50   F5  0834          sobgtr  r0, 10$                 ; Loop through cases                    [09]
                 FEFB   31  0837  20$:    brw     interpret               ; OK, return                            [09]
              59    52   D0  083A  30$:    movl    r2, r9                  ; Do value replacement                  [09]
       5A   F8 AA40    7E  083D          movaq   -8(r10)[r0], r10        ; Skip rest of the values               [09]
                    F3   11  0842          brb     20$                     ; Mission accomplished, as ordered      [09]
                         0844
                         0844  Pd_Name:                                 ; Just skip over the name directive     [17]
           50    01 AA   9A  0844          MovZBL  1(R10), R0              ; Get length of enforced generic        [17]
       5A   02 AA40    9E  0848          MovAB   2(R10)[R0], R10         ; Skip over it all                      [17]
                 FEE5   31  084D          BrW     Interpret               ; Return                                [17]
                         0850  PD_EPB:                                                                          ;G:9
              5A    02   C0  0850          ADDL2   #2,R10                  ; Skip over word storage                ;G:9
                 FEDF   31  0853          BRW     INTERPRET                                                       ;G:9
                         0856  PD_DECIMAL:
              53    0A   D0  0856          MOVL    #10,R3                  ; Set radix
                 008F   30  0859          BSBW    PD_NUMERIC              ; Join common
41 21 20 43 41 21 00000864'010E0000'  085C          .ASCID  "!AC !AC decimal !UL thru !UL!/!AC? "                   ;G:
55 21 20 6C 61 6D 69 63 65 64 20 43  086A
2F 21 4C 55 21 20 75 72 68 74 20 4C  0876
                 20 3F 43 41 21  0882
                         0887  PD_OCTAL:
              53    08   D0  0887          MOVL    #8,R3                   ; Set radix
                    5F   10  088A          BSBB    PD_NUMERIC              ; Join common
41 21 20 43 41 21 00000894'010E0000'  088C          .ASCID  "!AC !AC octal !6OL thru !6OL!/!AC? "
4C 4F 36 21 20 6C 61 74 63 6F 20 43  089A
2F 21 4C 4F 36 21 20 75 72 68 74 20  08A6
                 20 3F 43 41 21  08B2
                         08B7  PD_HEXADECIMAL:
              53    10   D0  08B7          MOVL    #16,R3                  ; Set radix
                    2F   10  08BA          BSBB    PD_NUMERIC              ; join common
41 21 20 43 41 21 000008C4'010E0000'  08BC          .ASCID  "!AC !AC hexadecimal !XL thru !XL!/!AC? "
61 6D 69 63 65 64 61 78 65 68 20 43  08CA
21 20 75 72 68 74 20 4C 58 21 20 6C  08D6
          20 3F 43 41 21 2F 21 4C 58  08E2
                         08EB  PD_NUMERIC:
```

```
                53   DD  08EB  1370          PUSHL   R3                      ; Save radix
            53  5A   D0  08ED  1371          MOVL    R10,R3                  ; Address of ascic prompt
            52  5A   D0  08F0  1372          MOVL    R10,R2                  ; Save prompt string for later
 00000B56'EF    16   08F3  1373          Jsb     L^GET_LINE              ; Span spaces, new input if necessary   [18]
                53 8ED0  08F9  1374          POPL    R3                      ; restore radix
              53 50   E9  08FC  1375          BLBC    R0,30$                  ; Return if input errors
            51  8A   9A  08FF  1376          MOVZBL  (R10)+,R1               ; Length of prompt
            5A  51   C0  0902  1377          ADDL    R1,R10                  ; Skip over prompt
            57  55   D0  0905  1378          MOVL    R5,R7                   ; Save current string address
                04   BB  0908  1379          PUSHR   #^M<R2>                 ; Save pointer on stack now
 00000000'EF    16   090A  1380          Jsb     L^SCAN$NUMERIC          ; Scan and fetch input                  [18]
                04   BA  0910  1381          POPR    #^M<R2>                 ; Restore pointer
                22   13  0912  1382          BEQL    20$                     ; Branch on bad input
         57  55  57   C3  0914  1383          SUBL3   R7,R5,R7                ; Calculate length of number
                51   D5  0918  1384          TSTL    R1                      ; High order portion must be zero
                1A   12  091A  1385          BNEQ    20$                     ; Branch on out of range
            6A  50   D1  091C  1386          CMPL    R0,(R10)                ; Compare data with low limit
                15   1F  091F  1387          BLSSU   20$                     ; Branch if less than limit
             04 AA  50   D1  0921  1388          CMPL    R0,4(R10)               ; Compare data with high limit
                0F   1A  0925  1389          BGTRU   20$                     ; Branch if out of range
            59  50   D0  0927  1390          MOVL    R0,R9                   ; Set current value
            5A  08   C0  092A  1391          ADDL    #8,R10                  ; Skip over low/high
                8E   D5  092D  1392          TSTL    (SP)+                   ; Remove edit string address
             02 A8  57   A0  092F  1393          ADDW2   R7,2(R8)                ; Increase length used so far
              FDFF   31   0933  1394          BRW     INTERPRET               ; Next instruction
                         0936  1395 20$:
            50  8E   D0  0936  1396          MOVL    (SP)+,R0                ; Get address of error edit string
                62   9F  0939  1397          PUSHAB  (R2)                    ; Prompt address
              7E  6A   7D  093B  1398          MOVQ    (R10),-(SP)             ; Push lower/upper bounds
       00000348'EF    9F  093E  1399          PUSHAB  L^T_SHOULDBE            ; Descriptive text argument             [10]
                62   9F  0944  1400          PUSHAB  (R2)                    ; Prompt
                60   7F  0946  1401          PUSHAQ  (R0)                    ; Address of format descriptor
 00000B7E'EF    06   FB  0948  1402          CALLS   #6,L^FORMAT_PROMPT      ; Do formatting & copy                  [10]
            50  14   D0  094F  1403          MOVL    #SS$_BADPARAM,R0        ; Bad parameter return
                         0952  1404 30$:
                04   0952  1405          RET                             ; Return on error
```

```
                     0953  1407
                     0953  1408 PD_STRING:
      53   5A   D0   0953  1409          MOVL    R10,R3                  ; Copy address of ascic prompt
      57   53   D0   0956  1410          MOVL    R3,R7                   ; Save prompt address
00000B56'EF   16   0959  1411          Jsb     L^GET_LINE              ; Verify something there
      01   50   E8   095F  1412          BLBS    R0,10$                  ; Branch if it worked
                   04   0962  1413          RET                             ; Return if input spaz
                     0963  1414 10$:
      59   01   CE   0963  1415          MNEGL   #1,R9                   ; Initialize VALUE
      53   8A   9A   0966  1416          MOVZBL  (R10)+,R3               ; Get length of prompt
                     0969  1417
      5A   53   C0   0969  1418 20$:     ADDL    R3,R10                  ; Skip over something
      53   8A   9A   096C  1419          MOVZBL  (R10)+,R3               ; Get length of next something
                   2D   13   096F  1420          BEQL    70$                     ; Branch if end of list
                   59   D6   0971  1421          INCL    R9                      ; Update string index
                     0973  1422
   50   54   53   C3   0973  1423          SUBL3   R3,R4,R0                ; Length remaining after this match
                   F0   19   0977  1424          BLSS    20$                     ; Input shorter, try next string
      51   55   D0   0979  1425          MOVL    R5,R1                   ; Copy input address
                   05   11   097C  1426          BRB     40$                     ; Start with count
      8A   81   91   097E  1427 30$:     CMPB    (R1)+,(R10)+            ; Compare character
                   E6   12   0981  1428          BNEQ    20$                     ; Branch if mismatch
        F8 53   F4   0983  1429 40$:     SOBGEQ  R3,30$                  ; Count and try again if more
   53   51   55   C3   0986  1430          SUBL3   R5,R1,R3                ; Calc length of string
     02 A8   53   A0   098A  1431          ADDW2   R3,2(R8)                ; Increase length used so far
      54   50   7D   098E  1432          MOVQ    R0,R4                   ; Put updated input pointers back
                   53   D4   0991  1433          CLRL    R3                      ; Already in position, add zero
      5A   53   C0   0993  1434 50$:     ADDL    R3,R10                  ; Skip over this one
      53   8A   9A   0996  1435          MOVZBL  (R10)+,R3               ; Get length of next item
                   F8   12   0999  1436          BNEQ    50$                     ; Branch if not end
          FD97   31   099B  1437          BRW     INTERPRET               ; Next instruction
                     099E  1438
                     099E  1439 70$:
   52   FF70 CD   9E   099E  1440          MOVAB   BUFFER(FP),R2           ; Address buffer
                     09A3  1441
5A   FC AD   01   C1   09A3  1442          ADDL3   #1,OLDPC(FP),R10        ; Point to prompt
      50   8A   9A   09A8  1443          MOVZBL  (R10)+,R0               ; Length of prompt
      5A   50   C0   09AB  1444          ADDL    R0,R10                  ; Skip prompt
                     09AE  1445
      50   8A   9A   09AE  1446 80$:     MOVZBL  (R10)+,R0               ; Get length of possible string
                   0B   13   09B1  1447          BEQL    100$                    ; Branch if last string
      82   8A   90   09B3  1448 90$:     MOVB    (R10)+,(R2)+            ; Copy byte
        FA 50   F5   09B6  1449          SOBGTR  R0,90$                  ; Branch if any left
      82   2C   90   09B9  1450          MOVB    #^A',',(R2)+            ; Insert ','
                   F0   11   09BC  1451          BRB     80$                     ; Try again
                     09BE  1452
                   52   D7   09BE  1453 100$:    DECL    R2                      ; Remove ','
F4 AD   52   F8 AD   C3   09C0  1454          SUBL3   DESC+4(FP),R2,DESC(FP)  ; Length of prompt
                   67   9F   09C6  1455          PUSHAB  (R7)                    ; Address of prompt string
          F4 AD   7F   09C8  1456          PUSHAQ  DESC(FP)                ; Address of strings
00000348'EF   9F   09CB  1457          PUSHAB  L^T_SHOULDBE            ; Address of "should be" insert         [10]
00000360'EF   7F   09D1  1458          PUSHAQ  L^T_STRING              ; Address of format descriptor          [10]
00000B7E'EF   04   FB   09D7  1459          CALLS   #4,L^FORMAT_PROMPT      ; Do formatting                         [10]
      50   14   D0   09DE  1460          MOVL    #SS$_BADPARAM,R0        ; Bad parameter
                   04   09E1  1461          RET
```

```
                        09E2  1463          .SBTTL  DP$EXTRACT      Make P-table printable
                        09E2  1464
                        09E2  1465 ;++
                        09E2  1466 ; FUNCTIONAL DESCRIPTION:
                        09E2  1467 ;
                        09E2  1468 ;       This routine can be used to convert the data stored in                ;G:9
                        09E2  1469 ;       a P-table using the PT-descriptor to an ASCII string to be
                        09E2  1470 ;       printed.
                        09E2  1471 ;
                        09E2  1472 ; CALLING SEQUENCE:
                        09E2  1473 ;
                        09E2  1474 ;       PT$EXTRACT(PC,BASE,LENGTH,ADDRESS)
                        09E2  1475 ;
                        09E2  1476 ; INPUT PARAMETERS:
                        09E2  1477 ;
                        09E2  1478 ;       PC      Start of PT-descriptor
                        09E2  1479 ;       BASE    Base of P-table
                        09E2  1480 ;       TEXT    Address of QUAD descriptor of buffer
                        09E2  1481 ;
                        09E2  1482 ; IMPLICIT INPUTS:
                        09E2  1483 ;
                        09E2  1484 ; OUTPUT PARAMETERS:
                        09E2  1485 ;
                        09E2  1486 ;       TEXT    The length in the descriptor is updated
                        09E2  1487 ;
                        09E2  1488 ; IMPLICIT OUTPUTS:     NONE
                        09E2  1489 ;
                        09E2  1490 ; COMPLETION CODES:
                        09E2  1491 ;
                        09E2  1492 ;       SS$_BADPARAM    Incorrectly formed PT-descriptor
                        09E2  1493 ;       SS$_NORMAL      Completed successfully
                        09E2  1494 ;
                        09E2  1495 ; REGISTER USAGE:
                        09E2  1496 ;
                        09E2  1497 ;       R9      current VALUE
                        09E2  1498 ;       R10     Current PC in PT-descriptor
                        09E2  1499 ;       R11     Base address of P-table
                        09E2  1500 ;--
```

```
                                09E2  1502
                                09E2  1503          $OFFSET 4,POSITIVE,< -
                                09E2  1504          <EARG$_PC,4>, -
                                09E2  1505          <EARG$_BASE,4>, -
                                09E2  1506          <EARG$_TEXT,4>>
                                09E2                .SAVE   LOCAL_BLOCK
                                09E2                .PSECT $ABS$ ABS
                    00000004    FFFF                .=4
                                0004        EARG$_PC:
                    00000008    0004                .BLKB   4
                                0008        EARG$_BASE:
                    0000000C    0008                .BLKB   4
                                000C        EARG$_TEXT:
                    00000010    000C                .BLKB   4
                          000009E2                  .RESTORE
                                09E2  1507
                       0FFC     09E2  1508 .ENTRY  PT$EXTRACT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                09E4  1509
    5A    04 AC     7D         09E4  1510          MOVQ    EARG$_PC(AP),R10        ; Get PC and BASE
    57    0C BC     7D         09E8  1511          MOVQ    @EARG$_TEXT(AP),R7      ; Get length of buffer
             56     D4         09EC  1512          CLRL    R6                      ; Clear Last valid edit string
    6A    80 8F     91         09EE  1513          CMPB    #PD$_START,(R10)        ; Verify good format
             23     12         09F2  1514          BNEQ    EX_BADPARAM             ; WRONG!
                               09F4  1515 EXTRACT:
                               09F4  1516          CASE    SRC=(R10)+,TYPE=B,LIMIT=#PD$_START,DISPLIST=< -
                               09F4  1517                  EX_START, -
                               09F4  1518                  EX_FINISH, -
                               09F4  1519                  EX_DECIMAL, -
                               09F4  1520                  EX_OCTAL, -
                               09F4  1521                  EX_HEXADECIMAL, -
                               09F4  1522                  EX_STRING, -
                               09F4  1523                  EX_LITERAL, -
                               09F4  1524                  EX_FETCH, -
                               09F4  1525                  EX_STORE, -
                               09F4  1526                  EX_COMPLEMENT, -
                               09F4  1527                  ex_add, -               ; Add operation                 [09]
                               09F4  1528                  ex_logical, -           ; logical                       [09]
                               09F4  1529                  ex_case, -              ; Case                          [09]
                               09F4  1530                  ex_name,-                                             ;G:9
                               09F4  1531                  ex_epb>                 ; name                  [17]  ;G:9
0E'  80 8F    8A    8F         09F4                CASEB   (R10)+,#PD$_START,S^#<<30003$-30002$>/2>-1
                               09F9        30002$:
                    0022'      09F9                .SIGNED_WORD    EX_START-30002$
                    006C'      09FB                .SIGNED_WORD    EX_FINISH-30002$
                    0024'      09FD                .SIGNED_WORD    EX_DECIMAL-30002$
                    0024'      09FF                .SIGNED_WORD    EX_OCTAL-30002$
                    0024'      0A01                .SIGNED_WORD    EX_HEXADECIMAL-30002$
                    002F'      0A03                .SIGNED_WORD    EX_STRING-30002$
                    004F'      0A05                .SIGNED_WORD    EX_LITERAL-30002$
                    004F'      0A07                .SIGNED_WORD    EX_FETCH-30002$
                    0079'      0A09                .SIGNED_WORD    EX_STORE-30002$
                    0051'      0A0B                .SIGNED_WORD    EX_COMPLEMENT-30002$
                    004F'      0A0D                .SIGNED_WORD    ex_add-30002$
                    0041'      0A0F                .SIGNED_WORD    ex_logical-30002$
                    0053'      0A11                .SIGNED_WORD    ex_case-30002$
                    005C'      0A13                .SIGNED_WORD    ex_name-30002$
                    0067'      0A15                .SIGNED_WORD    ex_epb-30002$
```

```
                        0A17       30003$:
                        0A17  1532 EX_BADPARAM:
          50   14   D0   0A17  1533          MOVL    S^#SS$_BADPARAM,R0      ; bad format PT-descriptor
                    04   0A1A  1534          RET
                        0A1B  1535
                        0A1B  1536 EX_START:
                    D7  11   0A1B  1537          BRB     EXTRACT                 ; Nop
                        0A1D  1538 EX_DECIMAL:
                        0A1D  1539 EX_OCTAL:
                        0A1D  1540 EX_HEXADECIMAL:
          56   FF AA   9E   0A1D  1541          MOVAB   -1(R10),R6              ; Save PC of prompt type inst
                    1E  10   0A21  1542          BSBB    SKIP                    ; Skip prompt
             5A   08   C0   0A23  1543          ADDL    #8,R10                  ; Skip low and high limits
                    CC  11   0A26  1544          BRB     EXTRACT                 ; Next inst
                        0A28  1545 EX_STRING:
          56   FF AA   9E   0A28  1546          MOVAB   -1(R10),R6              ; Save address of prompt type inst
                    13  10   0A2C  1547          BSBB    SKIP
                    50  D4   0A2E  1548          CLRL    R0                      ; Clear advance length
                        0A30  1549 10$:
             5A   50   C0   0A30  1550          ADDL    R0,R10                  ; Skip something
             50   8A   9A   0A33  1551          MOVZBL  (R10)+,R0               ; Get length
                    F8  12   0A36  1552          BNEQ    10$                     ; Try again
                    BA  11   0A38  1553          BRB     EXTRACT                 ; Next instruction
                        0A3A  1554 ex_logical:                                  ;                                       [09]
          56   FF AA   9E   0A3A  1555          movab   -1(r10), r6             ; Save address of directive             [09]
               B3 AF   9F   0A3E  1556          pushab  extract                 ; Set up for SKIP to do RSB             [09]
                        0A41  1557 SKIP:
             50   8A   9A   0A41  1558          MOVZBL  (R10)+,R0               ; Get length
             5A   50   C0   0A44  1559          ADDL    R0,R10                  ; Skip string
                    05   0A47  1560          RSB
                        0A48  1561
                        0A48  1562 EX_LITERAL:
                        0A48  1563 EX_FETCH:
                        0A48  1564 EX_ADD:
                    8A  D5   0A48  1565          TSTL    (R10)+                  ; Skip literal or word+byte+byte
                        0A4A  1566 EX_COMPLEMENT:
                    A8  11   0A4A  1567          BRB     EXTRACT                 ; next!
                        0A4C  1568 ex_case:                                     ;                                       [09]
             50   8A   9A   0A4C  1569          movzbl  (r10)+, r0              ; Get number of cases                   [09]
          5A   6A40   7E   0A4F  1570          movaq   (r10)[r0], r10          ; Skip the cases                        [09]
                    9F  11   0A53  1571          brb     extract                 ; And continue interpretation           [09]
                        0A55  1572
                        0A55  1573 Ex_Name:                                     ; Skip over Name directive              [17]
          50   01 AA   9A   0A55  1574          MovZBL  1(R10), R0              ; Get length of enforced generic        [17]
       5A   02 AA40   9E   0A59  1575          MovAB   2(R10)[R0], R10         ; Skip over it all                      [17]
                    94  11   0A5E  1576          BrB     Extract                 ; Return                                [17]
                        0A60  1577
                        0A60  1578 ex_epb:                                                                              ;G:9
             5A   02   C0   0A60  1579          ADDL2   #2,R10                  ; Skip over word storage                ;G:9
                    8F  11   0A63  1580          BRb     extract                                                         ;G:9
                        0A65  1581                                                                                      ;G:9
                        0A65  1582 EX_FINISH:
          50   0C BC   7D   0A65  1583          MOVQ    @EARG$_TEXT(AP),R0      ; Get length and address again
    0C BC   58   51   C3   0A69  1584          SUBL3   R1,R8,@EARG$_TEXT(AP)   ; Store new length
             50   01   D0   0A6E  1585          MOVL    S^#SS$_NORMAL,R0        ; It worked
                    04   0A71  1586          RET
```

```
                                0A72  1588
                                0A72  1589 EX_STORE:
                50   8A   3C    0A72  1590         MOVZWL  (R10)+,R0                ; Get byte offset
                51   8A   9A    0A75  1591         MOVZBL  (R10)+,R1                ; Get bit offset
                52   8A   9A    0A78  1592         MOVZBL  (R10)+,R2                ; Get field size
      59   6B40  52   51   EF   0A7B  1593         EXTZV   R1,R2,(R11)[R0],R9       ; Set current value
                      56   D5   0A81  1594         TSTL    R6                       ; Valid?
                      03   12   0A83  1595         BNeq    5$                       ; Yes, continue                    [17]
                    FF6C   31   0A85  1596         BrW     EXTRACT                  ; No, next instruction             [17]
```

```
                                0A88  1598 5$:     CASE    SRC=(R6)+,TYPE=B,LIMIT=#PD$_DECIMAL,DISPLIST=< -
                                0A88  1599                   st_decimal,st_octal, -  ;                    [09]
                                0A88  1600                   st_hexadecimal, -       ;                    [09]
                                0A88  1601                   st_string, 10$, 10$, -  ;                    [09]
                                0A88  1602                   10$, 10$, 10$, -        ;                    [09]
                                0A88  1603                   st_logical>             ;                    [09]
            09' 82 8F   86   8F 0A88                 CASEB   (R6)+,#PD$_DECIMAL,S^#<<30005$-30004$>/2>-1
                                0A8D        30004$:
                           0017' 0A8D                .SIGNED_WORD    st_decimal-30004$
                           0027' 0A8F                .SIGNED_WORD    st_octal-30004$
                           0039' 0A91                .SIGNED_WORD    st_hexadecimal-30004$
                           004A' 0A93                .SIGNED_WORD    st_string-30004$
                           0014' 0A95                .SIGNED_WORD    10$-30004$
                           0014' 0A97                .SIGNED_WORD    10$-30004$
                           0014' 0A99                .SIGNED_WORD    10$-30004$
                           0014' 0A9B                .SIGNED_WORD    10$-30004$
                           0014' 0A9D                .SIGNED_WORD    10$-30004$
                           0075' 0A9F                .SIGNED_WORD    st_logical-30004$
                                0AA1        30005$:
                     FF50   31  0AA1  1604 10$:     brw     extract                 ; Nop, just return          [09]
                                0AA4  1605 ST_DECIMAL:
                  53   56   D0  0AA4  1606          MOVL    R6,R3                   ; Prompt address
                     0082   30  0AA7  1607          bsbw    ex_common               ; Do common                 [09]
    20 2E 4C 55 21 3D 43 41 21 00' 0AAA  1608          .ASCIC  "!AC=!UL. "
                           09   0AAA
                                0AB4  1609 ST_OCTAL:
                  53   56   D0  0AB4  1610          MOVL    R6,R3                   ; Prompt address
                        73   10  0AB7  1611          BSBB    EX_COMMON               ; Octal
29 4F 28 4C 4F 36 21 3D 43 41 21 00' 0AB9  1612          .ASCIC  "!AC=!6OL(O) "
                           20   0AC5
                           0C   0AB9
                                0AC6  1613 ST_HEXADECIMAL:
                  53   56   D0  0AC6  1614          MOVL    R6,R3                   ; Prompt address
                        61   10  0AC9  1615          BSBB    EX_COMMON               ; Hexadecimal
20 29 58 28 4C 58 21 3D 43 41 21 00' 0ACB  1616          .ASCIC  "!AC=!XL(X) "
                           0B   0ACB
                                0AD7  1617 ST_STRING:
                  53   56   D0  0AD7  1618          MOVL    R6,R3                   ; Save address of prompt
                  50   86   9A  0ADA  1619          MOVZBL  (R6)+,R0                ; Get length of prompt
                  56   50   C0  0ADD  1620 10$:     ADDL    R0,R6                   ; Skip previous string
                  50   86   9A  0AE0  1621          MOVZBL  (R6)+,R0                ; Get length of string
                       12   13  0AE3  1622          BEQL    30$                     ; Branch if end
                     F5 59   F4  0AE5  1623          SOBGEQ  R9,10$                  ; Count and branch
                59   FF A6   9E  0AE8  1624          MOVAB   -1(R6),R9               ; Address of prompt
                        3E   10  0AEC  1625          BSBB    EX_COMMON               ; Join common
        20 43 41 21 3D 43 41 21 00' 0AEE  1626          .ASCIC  "!AC=!AC "
                           08   0AEE
                        33   10  0AF7  1627 30$:     BSBB    EX_COMMON               ; Join common
        20 3F 3F 3F 3D 43 41 21 00' 0AF9  1628          .ASCIC  "!AC=??? "
                           08   0AF9
                                0B02  1629
                                0B02  1630 st_logical:                              ;                       [09]
                  53   56   D0  0B02  1631          movl    r6, r3                  ; Get last prompt address   [09]
            50   00000B23'EF   9E  0B05  1632          movab   20$, r0                 ; 'yes' string              [09]
                     07 59   E8  0B0C  1633          blbs    r9, 10$                 ; If stored value is 1 (yes), OK  [09]
            50   00000B28'EF   9E  0B0F  1634          movab   30$, r0                 ; Else set string to 'no'   [09]
                  59   50   D0  0B16  1635 10$:     movl    r0, r9                  ; Set R9 to string          [09]
```

```
                    11   10  0B19  1636          bsbb    ex_common               ; (SP) gives format string              [09]
 43 41 21 3D 43 41 21 00' 0B1B  1637          .ascic  "!AC=!AC"               ;                                       [09]
                    07   0B1B
          20 73 65 59 00' 0B23  1638 20$:      .ascic  "Yes "                  ; Yes response                          [09]
                    04   0B23
             20 6F 4E 00' 0B28  1639 30$:      .ascic  "No "                   ; No response                           [09]
                    03   0B28
                         0B2C  1640
                         0B2C  1641 EX_COMMON:
                 02   BA  0B2C  1642          POPR    #^M<R1>                 ; Address of edit string
              50 81   9A  0B2E  1643          MOVZBL  (R1)+,R0                ; Get length
            0183 8F   BB  0B31  1644          PUSHR   #^M<R0,R1,R7,R8>        ; Push R0/R1, buffer len,address
                         0B35  1645
            0208 8F   BB  0B35  1646          PUSHR   #^M<R3,R9>              ; Push field name, value
              10 AE   7F  0B39  1647          PUSHAQ  16(SP)                  ; Address of output descriptor
              14 AE   DF  0B3C  1648          PUSHAL  20(SP)                  ; Store length back in descriptor
              10 AE   7F  0B3F  1649          PUSHAQ  16(SP)                  ; Address of edit string
00000000'EF   05   FB  0B42  1650          CALLS   #5,L^SYS$FAO            ; Edit string
                         0B49  1651
                 0F   BA  0B49  1652          POPR    #^M<R0,R1,R2,R3>        ; Clean stack, get length in R2
              57 52   C2  0B4B  1653          SUBL    R2,R7                   ; Lessen available length
              58 52   C0  0B4E  1654          ADDL    R2,R8                   ; And skip over for next time           ;G:
                 56   D4  0B51  1655          CLRL    R6                      ; Don't do again
               FE9E   31  0B53  1656          BRW     EXTRACT                 ; Next!
```

```
                        .SBTTL  GET_LINE        Condition getline

;**
; FUNCTIONAL DESCRIPTION:
;
;       This routine conditionally prompts for input with the
;       prompt string provided, if any input is left from the last time
;       it is used before prompting is done.
;
; CALLING SEQUENCE:
;
;       BSBB    GET_LINE
;
; INPUT PARAMETERS:     NONE
;
; IMPLICIT INPUTS:
;
;       R3      Address of prompt string (ASCIC)
;       R4      Length remaining in last input string
;       R5      Address of remaining input string
;       R6      Address of prompt string output descriptor
;       R8      Address of input line descriptor
;
; OUTPUT PARAMETERS:    NONE
;
; IMPLICIT OUTPUTS:
;
;       R4      Length of input string
;       R5      Address of input string
;
; SIDE EFFECTS:         NONE
;
; COMPLETION CODES:     NONE
;--
```

```
                        0B56  1693
                        0B56  1694 GET_LINE:
              53   DD   0B56  1695          PUSHL   R3                      ; Save register
00000000'EF   16   0B58  1696          Jsb     L^SCAN$SPACES           ; Skip over leading spaces              [18]
 02 A8   50   A0   0B5E  1697          ADDW2   R0,2(R8)                ; Add to length used so far
              53 8ED0   0B62  1698          POPL    R3                      ; Restore register
    50   01   D0   0B65  1699          MOVL    S^#SS$_NORMAL,R0        ; Assume it worked
              54   D5   0B68  1700          TSTL    R4                      ; Any input left?
              11   12   0B6A  1701          BNEQ    10$                     ; Exit if some input left
              63   9F   0B6C  1702          PUSHAB  (R3)                    ; Prompt string
0000013F'EF   7F   0B6E  1703          PUSHAQ  L^T_PROMPT              ; Edit string (add " ?")                [10]
 7E'AF   02   FB   0B74  1704          CALLS   #2,B^FORMAT_PROMPT      ; Format it
50   0114 8F   3C   0B78  1705          MOVZWL  #SS$_INSFARG,R0         ; Not enough arguments
                        05   0B7D  1706 10$:    RSB
```

```
                        0B7E  1708          .SBTTL  FORMAT_PROMPT   Format and load prompt
                        0B7E  1709
                        0B7E  1710 ;++
                        0B7E  1711 ; Functional description:
                        0B7E  1712 ;       Call FAO with arguments and format string given, store resultant
                        0B7E  1713 ;       string in prompt output string
                        0B7E  1714 ;
                        0B7E  1715 ; Calling sequence:
                        0B7E  1716 ;       FORMAT_PROMPT (FORMAT, params)
                        0B7E  1717 ;
                        0B7E  1718 ; Input parameters:
                        0B7E  1719 ;       format          address of format descriptor
                        0B7E  1720 ;       params          list of arguments to FAO
                        0B7E  1721 ;
                        0B7E  1722 ; Implicit inputs:
                        0B7E  1723 ;       R6              contains pointer to output descriptor
                        0B7E  1724 ;
                        0B7E  1725 ; Outputs:
                        0B7E  1726 ;       none
                        0B7E  1727 ;
                        0B7E  1728 ; Implicit outputs:
                        0B7E  1729 ;       prompt string modified
                        0B7E  1730 ;
                        0B7E  1731 ; Side effects:
                        0B7E  1732 ;       none
                        0B7E  1733 ;
                        0B7E  1734 ; Return codes:
                        0B7E  1735 ;       FAO return codes (success, string truncated, or bad directive)
                        0B7E  1736 ;--
```

```
                                0B7E  1738
                                0B7E  1739          $OFFSET 4,POSITIVE,<-
                                0B7E  1740                  <FPT$_FORMAT,4>,-
                                0B7E  1741                  <FPT$_ARGS>>
                                0B7E                .SAVE   LOCAL_BLOCK
                                0B7E                .PSECT $ABS$ ABS
                    00000004    FFFF                .=4
                                0004        FPT$_FORMAT:
                    00000008    0004                .BLKB   4
                                0008        FPT$_ARGS:
                    0000000C    0008                .BLKB   4
                          00000B7E                  .RESTORE
                                0B7E  1742
                                0B7E  1743  FORMAT_PROMPT:
                          0008  0B7E  1744          .WORD   ^M<R3>
              53    7E    7E    0B80  1745          MOVAQ   -(SP),R3                ; Allocate output descriptor
        63    66    01    A3    0B83  1746          SUBW3   #1,(R6),(R3)            ; Set length
  04 A3    01    04 A6    C1    0B87  1747          ADDL3   4(R6),#1,4(R3)          ; Address to put text
                                0B8D  1748          $FAOL_S @FPT$_FORMAT(AP),(R3),- ; Do formatting
                                0B8D  1749                   (R3),FPT$_ARGS(AP)
                 08 AC    DF    0B8D                        PUSHAL  FPT$_ARGS(AP)
                    63    7F    0B90                        PUSHAQ  (R3)
                    63    3F    0B92                        PUSHAW  (R3)
                 04 BC    7F    0B94                        PUSHAQ  @FPT$_FORMAT(AP)
  00000000'GF    04    FB    0B97                        CALLS   #4,G^SYS$FAOL
      66    63    01    A1    0B9E  1750          ADDW3   #1,(R3),(R6)            ; Length including length byte
        04 B6    63    90    0BA2  1751          MOVB    (R3),@4(R6)             ; Store result length
                          04    0BA6  1752          RET
```

```
                        0BA7  1754          .SBTTL  DSV$SHOWDEVICE  Display device information
                        0BA7  1755
                        0BA7  1756 ;++
                        0BA7  1757 ; FUNCTIONAL DESCRIPTION:
                        0BA7  1758 ;
                        0BA7  1759 ;       This routine is called from CLI to display information about
                        0BA7  1760 ;       devices attached to the system.
                        0BA7  1761 ;
                        0BA7  1762 ; CALLING SEQUENCE:
                        0BA7  1763 ;
                        0BA7  1764 ;       BSBW    DSV$SHOWDEVICE
                        0BA7  1765 ;
                        0BA7  1766 ; INPUT PARAMETERS:
                        0BA7  1767 ;
                        0BA7  1768 ; IMPLICIT INPUTS:
                        0BA7  1769 ;
                        0BA7  1770 ;       CLI$Q_FILE(R2)  Has descriptor for rest of command line.
                        0BA7  1771 ;
                        0BA7  1772 ; OUTPUT PARAMETERS:    NONE
                        0BA7  1773 ;
                        0BA7  1774 ; IMPLICIT OUTPUTS:     NONE
                        0BA7  1775 ;
                        0BA7  1776 ; SIDE EFFECTS:         NONE
                        0BA7  1777 ;
                        0BA7  1778 ; COMPLETION CODES:     NONE
                        0BA7  1779 ;
                        0BA7  1780 ; REGISTER USAGE:
                        0BA7  1781 ;
                        0BA7  1782 ;       R0      Length of device name as typed
                        0BA7  1783 ;       R1      Address of device name as typed
                        0BA7  1784 ;       R2      Length of Device name in DPB
                        0BA7  1785 ;       R3      Address of device name in DPB
                        0BA7  1786 ;       R5      Index into DPBLIST
                        0BA7  1787 ;       R6      Address of link device for /Adapter
                        0BA7  1788 ;       R10     DPB address
                        0BA7  1789 ;--
```

```
                        0BA7  1791
                        0BA7  1792 DSV$SHOWDEVICEB::                                                    ;                                                [07]
00000000'EF   01   88   0BA7  1793          BISB2   #1@FLG$V_BRIEF,COMM_FLAGS  ; Set brief flag                                                [07]
                        0BAE  1794 DSV$SHOWDEVICE::
      50   08 A2   7D   0BAE  1795          MOVQ    CLI$Q_FILE(R2),R0         ; Get descriptor of input
         08FF 8F   BB   0BB2  1796          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R11>          ;                                               [11]
                        0BB6  1797
55    00000000'EF   DE   0BB6  1798          MOVAL   DS$GA_PTABLE,R5           ; Base PTABLE list
            0582    30   0BBD  1799          bsbw    loc$adapter               ; Find /Adapter value, if any                                   [11]
         39 50      E9   0BC0  1800          blbc    r0, 70$                   ; If not found, return                                          [11]
      54   65      D0   0BC3  1801 10$:      MOVL    (R5),R4                   ; Get count of PT entries                                       [11]
   5B   6544      D0   0BC6  1802 20$:      MOVL    (R5)[R4],R11              ; Get address of Associated PT
            2D    13   0BCA  1803          BEQL    60$                       ; Branch if none
            56    D5   0BCC  1804          tstl    r6                        ; Was /Adapter specified?                                       [11]
            06    19   0BCE  1805          blss    30$                       ; If not, continue                                              [11]
   20 AB    56    D1   0BD0  1806          cmpl    r6, hp$a_link(r11)        ; Compare with link of current device                           [11]
            23    12   0BD4  1807          bneq    60$                       ; If not, on to next device                                     [11]
      50    6E    D0   0BD6  1808 30$:      MOVL    (SP),R0                   ; Get length of input                                           [11]
            18    13   0BD9  1809          BEQL    50$                       ; Branch if Null input..equiv to all
   51   04 AE    D0   0BDB  1810          MOVL    4(SP),R1                  ; Get address of input
      52    6B    7D   0BDF  1811          MOVQ    HP$Q_DEVICE(R11),R2       ; Get length,address of device name
            52    D7   0BE2  1812          DECL    R2                        ; Remove "_"
            53    D6   0BE4  1813          INCL    R3                        ; skip "_"
      52    50    D1   0BE6  1814          CMPL    R0,R2                     ; Compare name lengths
            0E    14   0BE9  1815          BGTR    60$                       ; Skip compare if input longer than name
      83    81    91   0BEB  1816 40$:      CMPB    (R1)+,(R3)+               ; Compare characters
            09    12   0BEE  1817          BNEQ    60$                       ; Branch if mismatch
         F8 50    F5   0BF0  1818          SOBGTR  R0,40$                    ; Count and continue
                        0BF3  1819
00000FDA'EF   16   0BF3  1820 50$:      Jsb     L^SHOW_DEV                ; Display this DPB                                              [18]
                        0BF9  1821
      CA 54    F5   0BF9  1822 60$:      SOBGTR  R4,20$                    ; Count and branch if more
                        0BFC  1823
         08FF 8F   BA   0BFC  1824 70$:      POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R11>          ;
00000000'EF   01   8A   0C00  1825          BICB2   #1@FLG$V_BRIEF,COMM_FLAGS  ; Clear brief flag before exit                                 [07]
                   05   0C07  1826          RSB
```

```
                        0C08  1828          .SBTTL  DSV$SHOWSELECT  Display information selects
                        0C08  1829
                        0C08  1830 ;++
                        0C08  1831 ; FUNCTIONAL DESCRIPTION:
                        0C08  1832 ;
                        0C08  1833 ;       This routine is called from CLI to display information
                        0C08  1834 ;       about device selected for test
                        0C08  1835 ;
                        0C08  1836 ; CALLING SEQUENCE:
                        0C08  1837 ;
                        0C08  1838 ;       BSBW    DSV$SHOWSELECT
                        0C08  1839 ;
                        0C08  1840 ; INPUT PARAMETERS:     NONE
                        0C08  1841 ;
                        0C08  1842 ; IMPLICIT INPUTS:
                        0C08  1843 ;
                        0C08  1844 ;       CLI$Q_FILE(R2)  Descriptor of device to be shown
                        0C08  1845 ;
                        0C08  1846 ; OUTPUT PARAMETERS:    NONE
                        0C08  1847 ;
                        0C08  1848 ; IMPLICIT OUTPUTS:     NONE
                        0C08  1849 ;
                        0C08  1850 ; COMPLETION CODES:     NONE
                        0C08  1851 ;
                        0C08  1852 ; SIDE EFFECTS:         NONE
                        0C08  1853 ;
                        0C08  1854 ; REGISTER USAGE:
                        0C08  1855 ;
                        0C08  1856 ;       R0/R1    Scratch
                        0C08  1857 ;
                        0C08  1858 ;       R10      Address of DPB for device to be shown
                        0C08  1859 ;--
```

```
                                  0C08
                                  0C08  DSV$SHOWSELECTB::                                  ;                                    [07]
      00000000'EF    01    88     0C08          BISB2   #1@FLG$V_BRIEF,COMM_FLAGS          ; Set brief flag                     [07]
                                  0C0F  DSV$SHOWSELECT::
                0898 8F    BB     0C0F          PUSHR   #^M<R4,R3,R7,R11>
                                  0C13
      53   00000000'EF    DE     0C13          MOVAL   DS$GA_PTABLE,R3                  ; Base PTABLE LIST
                54   63    D0     0C1A          MOVL    (R3),R4                          ; Fetch count of PT entries
                                  0C1D  10$:
   0C 00000000'EF    54    E1     0C1D          BBC     R4,DS$GA_SELECTS,20$             ; Branch if not SELECTED
             5B   6344    D0     0C25          MOVL    (R3)[R4],R11                     ; Address of PT
                   06    13     0C29          BEQL    20$                              ; Branch if none
          00000FDA'EF    16     0C2B          Jsb     L^SHOW_DEV                       ; Display device information for this   [18]
                                  0C31  20$:
                E9 54    F5     0C31          SOBGTR  R4,10$                           ; Count and Branch if more
                0898 8F    BA     0C34          POPR    #^M<R4,R3,R7,R11>                        ; Restore
      00000000'EF    01    8A     0C38          BICB2   #1@FLG$V_BRIEF,COMM_FLAGS          ; Clear brief flag before exit  [07]
                           05     0C3F          RSB
```

```
                                0C40          .Subtitle       DSV$DEATTACH    Deattach device
                                0C40
                                0C40  ;++
                                0C40  ; FUNCTIONAL DESCRIPTION:
                                0C40  ;
                                0C40  ;       This routine is called from CLI to reverse the ATTACH command
                                0C40  ;       for a given device.  Since the attach data base is a tree
                                0C40  ;       structure, all subsidiary devices are also implicitly
                                0C40  ;       deattached.
                                0C40  ;
                                0C40  ; CALLING SEQUENCE:
                                0C40  ;
                                0C40  ;       BSBW    DSV$DEATTACH
                                0C40  ;
                                0C40  ; INPUT PARAMETERS:     NONE
                                0C40  ;
                                0C40  ; IMPLICIT INPUTS:
                                0C40  ;
                                0C40  ;       CLI$Q_FILE      descriptor of device to be added
                                0C40  ;       Q_ADAPTER       descriptor of adapter device is attached to
                                0C40  ;
                                0C40  ; OUTPUT PARAMETERS:    NONE
                                0C40  ;
                                0C40  ; IMPLICIT OUTPUTS:     NONE
                                0C40  ;
                                0C40  ; SIDE EFFECTS:         NONE
                                0C40  ;
                                0C40  ; COMPLETION CODES:     N/A
                                0C40  ;--
                                0C40
                                0C40  dsv$deattach::                                    ;                                [12]
          00E7 8F       BB      0C40          pushr   #^M<r0,r1,r2,r5,r6,r7>            ; Save registers                 [12]
      00000000'EF       16      0C44          jsb     script$flush                      ; Flush out scripts if should    [12]
55    00000000'EF       DE      0C4A          moval   ds$ga_ptable, r5                  ; Cache address of table         [12]
             04EE       30      0C51          bsbw    loc$adapter                       ; Find the specified adapter     [12]
            65 50       E9      0C54          blbc    r0, 20$                           ; If not found, return           [12]
      50    08 A2       7D      0C57          movq    cli$q_file(r2), r0                ; Get device to deattach         [12]
               03       12      0C5B          bneq    4$                                ; There is a device            ;G:6
             005C       31      0C5D          brw     20$                               ; Branch if no device          ;G:6
   52    61    08       9C      0C60  4$:     rotl    #8,(r1),r2                        ; Get first 3 characters of name ;G:
         52    50       90      0C64          movb    r0,r2                             ; Insert length                ;G:6
52    4C4C4103 8F       D1      0C67          cmpl    #<^A"@ALL"-61>,r2                 ; Compare with ascic "ALL"     ;G:6
               1A       12      0C6E          bneq    8$                                ; No. Branch for one device    ;G:6
         57    65       D0      0C70          movl    (r5),r7                           ; Get size of ptabel           ;G:6
      51    6547       D0      0C73  5$:     movl    (r5)[r7],r1                       ; Address of next ptable       ;G:6
               0B       13      0C77          beql    6$                                ; Skip if gone                 ;G:6
      56    20 A1       D1      0C79          cmpl    hp$a_link(r1),r6                  ; Check again adapter          ;G:6
               05       12      0C7D          bneq    6$                                ; If not linked to it , skip   ;G:6
         50    57       D0      0C7F          movl    r7,r0                             ; Copy the index               ;G:6
               3D       10      0C82          bsbb    30$                               ; Do deattach for this device  ;G:6
            EC 57       F5      0C84  6$:     sobgtr  r7,5$                             ; Loop through the table       ;G:6
             0032       31      0C87          brw     20$                               ; End of table, return         ;G:7
         52    56       D0      0C8A  8$:     movl    r6, r2                            ; Copy link address              [12]
             0507       30      0C8D          bsbw    loc$ptable                        ; Locate it                      [12]
               28       12      0C90          bneq    10$                               ; If found, then it's OK         [12]
      52    08 AE       D0      0C92          movl    8(sp), r2                         ; Restore CLI table pointer      [12]
7E    00000004'EF       7D      0C96          movq    q_adapter, -(sp)                  ; Get adapter name               [12]
```

```
                    6E    6E   3C   0C9D  1937          movzwl  (sp), (sp)                          ; Make sure high word is 0      [12]
                  7E   08 A2   7D   0CA0  1938          movq    cli$q_file(r2), -(sp)               ; Get device name               [12]
                    6E    6E   3C   0CA4  1939          movzwl  (sp), (sp)                          ; And zero extend length        [12]
                0000019D'EF    9F   0CA7  1940          pushab  L^t_no_such_on_adapter              ; No such device                [12]
                       01      DD   0CAD  1941          pushl   #ds$k_printf                        ; Use PRINTF                    [12]
                       15      DD   0CAF  1942          pushl   #ds$k_type_command_err              ; Message is command error      [12]
          00000000'GF  07      FB   0CB1  1943          calls   #7, G^dsx$print                     ; Print it                      [12]
                       02      11   0CB8  1944          brb     20$                                 ; Return                        [12]
                       05      10   0CBA  1945 10$:      bsbb    30$                                 ; Do the stuff, recursively     [12]
                  00E7 8F      BA   0CBC  1946 20$:      popr    #^M<r0,r1,r2,r5,r6,r7>              ; Restore the registers         [12]
                       05      0CC0  1947                rsb
```

```
                                 0CC1  1949          .Subtitle       Deattach subtree
                                 0CC1  1950 ;+
                                 0CC1  1951 ; This routine deallocates the Ptable for a device, clears the select bit
                                 0CC1  1952 ; and Ptable vector pointer for it; and if there were devices attached to
                                 0CC1  1953 ; it, does the same for each of them.
                                 0CC1  1954 ;-
                        1F   BB  0CC1  1955 30$:    pushr   #^M<r0,r1,r2,r3,r4>            ; Save some registers           [12]
                                 0CC3  1956 ;
                                 0CC3  1957 ; See if anything attached to this one (R1)
                                 0CC3  1958 ;
                     50   65     D0  0CC3  1959          movl    (r5), r0                    ; Size of table                [12]
                   51   6540     D0  0CC6  1960 40$:     movl    (r5)[r0], r1                ; Get address of next Ptable   [12]
                        09     13  0CCA  1961          beql    50$                         ; Skip if gone                 [12]
                04 AE   20 A1   D1  0CCC  1962          cmpl    hp$a_link(r1), 4(sp)        ; Check against input device   [12]
                        02     12  0CD1  1963          bneq    50$                         ; If not linked to it, skip    [12]
                        EC     10  0CD3  1964          bsbb    30$                         ; Do same for IT, recursively  [12]
                     EE 50     F5  0CD5  1965 50$:     sobgtr  r0, 40$                 ; Loop through the table       [12]
                                 0CD8  1966          $print  #ds$k_type_command_out, -   ; Print command info          [12]
                                 0CD8  1967                  #ds$k_printf, -             ; .. use PRINTF                [12]
                                 0CD8  1968                  L^t_deattach, -             ; .. Deattach message          [12]
                                 0CD8  1969                  4(sp)                       ; .. address of Ptable         [12]
                        04 AE    DD  0CD8                   PUSHL   4(sp)
                  00000227'EF    9F  0CDB                   PUSHAB  L^t_deattach
                        7E   01  9A  0CE1                   movzbl  #ds$k_printf, -(sp)
                        7E   16  98  0CE4                   cvtbl   #ds$k_type_command_out, -(sp)
              00000000'GF    04  FB  0CE7                   CALLS   #$$N+2, G^DSX$PRINT
                        50   6E  7D  0CEE  1970          movq    (sp), r0                    ; Get original r0 and r1       [12]
                           6540  D4  0CF1  1971          clrl    (r5)[r0]                    ; Clear table pointer          [12]
           00 00000000'EF    50  E5  0CF4  1972          bbcc    r0, L^ds$ga_selects, 60$    ; Clear select bit for it      [12]
                        50   51  D0  0CFC  1973 60$:     movl    r1, r0                      ; Copy address of Ptable       [12]
                  00000000'EF    16  0CFF  1974          Jsb     L^exe$deanonpaged           ; Deallocate it                [18]
                        1F   BA  0D05  1975          popr    #^M<r0,r1,r2,r3,r4>         ; Restore registers            [12]
                             05  0D07  1976          rsb                                 ; Return                       [12]
```

```
                        0D08  1978          .SBTTL  DSV$SELECT      Select device for testing
                        0D08  1979
                        0D08  1980 ;++
                        0D08  1981 ; FUNCTIONAL DESCRIPTION:
                        0D08  1982 ;
                        0D08  1983 ;       This routine is called from DS$CLI for each device to be
                        0D08  1984 ;       tested.  It sets the select bit and moves the P-table
                        0D08  1985 ;       entry to the end of the P-table list
                        0D08  1986 ;
                        0D08  1987 ; CALLING SEQUENCE:
                        0D08  1988 ;
                        0D08  1989 ;       BSBW    DSV$SELECT
                        0D08  1990 ;
                        0D08  1991 ; INPUT PARAMETERS:     NONE
                        0D08  1992 ;
                        0D08  1993 ; IMPLICIT INPUTS:
                        0D08  1994 ;
                        0D08  1995 ;       CLI$Q_FILE      descriptor of device to be added
                        0D08  1996 ;
                        0D08  1997 ; OUTPUT PARAMETERS:    NONE
                        0D08  1998 ;
                        0D08  1999 ; IMPLICIT OUTPUTS:
                        0D08  2000 ;
                        0D08  2001 ;       DS$GA_PTABLE    The indicated unit is moved to the head.
                        0D08  2002 ;
                        0D08  2003 ; SIDE EFFECTS:         NONE
                        0D08  2004 ;
                        0D08  2005 ; COMPLETION CODES:     N/A
                        0D08  2006 ;--
                        0D08  2007
                        0D08  2008 DSV$SELECT::
      00FF 8F    BB     0D08  2009          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7>     ; Preserve                          [11]
 00000000'EF     16     0D0C  2010          JSB     SCRIPT$FLUSH                    ; Flush scripts if console command  [10]
        03A6     30     0D12  2011          BsbW    CheckAmbiguous                  ; Check for ambiguous names         [15]
      1B 50      E9     0D15  2012          BlbC    R0, 10$                         ; If error, hopscotch on to exit    [15]
55 00000000'EF     DE     0D18  2013          MOVAL   DS$GA_PTABLE,R5                 ; Point to P-table
53 00000000'EF     DE     0D1F  2014          MOVAL   DS$GA_SELECTS,R3                ; Point to select bit mask
57 00000000'EF     DE     0D26  2015          MOVAL   DS$GA_ALLOCATES,R7              ; Point to allocated bit            [30]
        0412     30     0D2D  2016          bsbw    loc$adapter                     ; Find /Adapter, if any             [11]
      03 50      E8     0D30  2017          blbs    r0, 20$                         ; Continue if OK                    [11]
      0116       31     0D33  2018 10$:     brw     140$                            ; Return if not                     [11]
   50   08 A2     7D     0D36  2019 20$:     MOVQ    CLI$Q_FILE(R2),R0               ; Descriptor of device              [11]
        5E        13     0D3A  2020          BEQL    60$                             ; Exit if no device
52   61   08     9C     0D3C  2021          ROTL    #8,(R1),R2                      ; Get first 3 characters of name
     52   50     90     0D40  2022          MOVB    R0,R2                           ; Insert length
52 4C4C4103 8F     D1     0D43  2023          CMPL    #<^A"@ALL"-61>,R2               ; compare with ascic "all"
        51        12     0D4A  2024          BNEQ    70$                             ; Branch if not all
```

```
                        0D4C  2026
                        0D4C  2027 ;+
                        0D4C  2028 ; Process SELECT ALL command
                        0D4C  2029 ;-
          54    65  D0  0D4C  2030          MOVL    (R5),R4                 ; Get count of devices possible
       50   6544  D0  0D4F  2031 30$:       MOVL    (R5)[R4],R0             ; Anything here?
              42  13  0D53  2032            BEQL    50$                     ; No check next
              56  D5  0D55  2033            tstl    r6                      ; If no /Adapter                        [11]
              06  19  0D57  2034            blss    40$                     ; ...then branch around the check       [11]
       56   20 A0  D1  0D59  2035            cmpl    hp$a_link(r0), r6       ; Is this on the right link?            [11]
              38  12  0D5D  2036            bneq    50$                     ; If not, loop to next                  [11]
          50   60  7D  0D5F  2037 40$:       MOVQ    HP$Q_DEVICE(R0),R0      ; Get device name                       [11]
              50  D7  0D62  2038            DECL    R0                      ; remove "_" from length
              51  D6  0D64  2039            INCL    R1                      ; remove "_" from address
        00000000'EF  16  0D66  2040         Jsb     L^DSR$IFLOADDEV         ; Is this the load device?              [18]
            28 50  E8  0D6C  2041          BLBS    R0,50$                  ; Branch if it is
    63  01   54   01  F0  0D6F  2042          INSV    #1,R4,#1,(R3)           ; Set selected flag                     [30]
      1F 67   54  E0  0D74  2043          BBS     R4,(R7),50$             ; Branch if already set                 [30]
              00D6  30  0D78  2044          BSBW    150$                    ; Allocate the device if necessary      [11]
            04 50  E8  0D7B  2045          BLBS    R0,45$                  ; Branch if successful
      10 63   54  E5  0D7E  2046          BBCC    R4,(R3),47$             ; clear select bit if allocate failed
       52   6544  D0  0D82  2047 45$:       MOVL    (R5)[R4],R2             ; Get base of the P-table               [30]
    0C 0A A2   01  E1  0D86  2048          BBC     #HP$V_WASALL,   -       ; Branch if we did not allocate         [30]
                    0D8B  2049                  HP$B_FLAGS(R2),50$      ;                                       [30]
    67  01   54   01  F0  0D8B  2050          INSV    #1,R4,#1,(R7)           ; Set Allocated flag                    [30]
              05  11  0D90  2051          BRB     50$                     ; Continue                              [32]
    67  01   54   00  F0  0D92  2052 47$:       INSV    #0,R4,#1,(R7)           ; Clear allocated flag                  [30]
          B5 54  F5  0D97  2053 50$:       SOBGTR  R4,30$                  ; Do them all
            00AF  31  0D9A  2054 60$:       BRW     140$                    ; Branch to exit                        [11]
                    0D9D  2055
```

```
                                0D9D  2057
                                0D9D  2058 ;+
                                0D9D  2059 ; Process SELECT device command
                                0D9D  2060 ;-
            52    01    CE    0D9D  2061 70$:     mnegl   #1, r2                  ; Set wildcard link                          [09]
                  56    D5    0DA0  2062          tstl    r6                      ; If there's no /Adapter,                    [11]
                  03    19    0DA2  2063          blss    80$                     ; ...then don't check it                     [11]
            52    56    D0    0DA4  2064          movl    r6, r2                  ; If is, use address of link device          [11]
               03ED     30    0DA7  2065 80$:     bsbw    loc$ptable              ; Search for it                              [11]
                  48    12    0DAA  2066          BNEQ    110$                    ; Branch if found
         52    08 AE    D0    0DAC  2067          MOVL    8(SP),R2                ; Get base of CLI
                  56    D5    0DB0  2068          tstl    r6                      ; /ADAPTER?                                  [11]
                  25    19    0DB2  2069          blss    90$                     ; Branch if not                              [11]
      7E  00000004'EF    7D    0DB4  2070          movq    q_adapter, -(sp)        ; Push on adapter name                       [11]
            6E    6E    3C    0DBB  2071          movzwl  (sp), (sp)              ; Zero extend                                [11]
         7E    08 A2    7D    0DBE  2072          movq    cli$q_file(r2), -(sp)   ; Push on device name                        [11]
            6E    6E    3C    0DC2  2073          movzwl  (sp), (sp)              ; Zero extend                                [11]
         0000019D'EF    9F    0DC5  2074          pushab  L^at_no_such_on_adapter ; Format string                              [11]
                  01    DD    0DCB  2075          pushl   #ds$k_printf            ; Use printf                                 [13]
                  15    DD    0DCD  2076          pushl   #ds$k_type_command_err  ; Command error type code                   [13]
   00000000'GF    07    FB    0DCF  2077          calls   #7, G^dsx$print         ; Print it                                   [13]
               0073     31    0DD6  2078          brW     140$                    ; Exit                                       [11]
         7E    08 A2    7D    0DD9  2079 90$:     MOVQ    CLI$Q_FILE(R2),-(SP)    ; Stack file name                            [11]
            6E    6E    3C    0DDD  2080 100$:    movzwl  (sp), (sp)              ; Zero extend length                         [11]
         00000170'EF    9F    0DE0  2081          PUSHAB  L^AT_NO_SUCH_DEV        ; Edit string                                [11]
                  01    DD    0DE6  2082          pushl   #ds$k_printf            ; Use PRINTF                                 [13]
                  15    DD    0DE8  2083          pushl   #ds$k_type_command_err  ; Command error type code                   [13]
   00000000'GF    05    FB    0DEA  2084          CALLS   #5, G^DSX$PRINT         ; Type the error                             [13]
               0058     31    0DF1  2085          BRW     140$                    ; Branch to exit                             [11]
                                0DF4  2086 110$:
```

```
                                0DF4
                                0DF4  ;+
                                0DF4  ; Compress P-table list to high end of table
                                0DF4  ; R1 = P-table address
                                0DF4  ;-
                 54    50   D0  0DF4          MOVL    R0,R4                   ; Copy unit index of device
                       58   10  0DF7          BSBB    150$                    ; Allocate device if necessary          [11]
                    50 50   E9  0DF9          BLBC    R0,140$                 ; Branch if can't allocate
                     6544   DD  0DFC          PUSHL   (R5)[R4]                ; Save address of P-table
                     6544   D4  0DFF          CLRL    (R5)[R4]                ; Clear address of P-table
                 54    65   D0  0E02          MOVL    (R5),R4                 ; Get count of entries
                 52    54   D0  0E05          MOVL    R4,R2                   ; current position
                                0E08
              50     6542   D0  0E08  120$:   MOVL    (R5)[R2],R0             ; Copy address to first available
                       1D   13  0E0C          BEQL    130$                    ; Empty, next
                     6542   D4  0E0E          CLRL    (R5)[R2]                ; Clear it out
              6544    50   D0  0E11          MOVL    R0,(R5)[R4]             ; Put it in the first available slot
     50   63   01    52   EF  0E15          EXTZV   R2,#1,(R3),R0           ; Get select bit
     63   01   54    50   F0  0E1A          INSV    R0,R4,#1,(R3)           ; Insert new select bit
     50   67   01    52   EF  0E1F          EXTZV   R2,#1,(R7),R0           ; Get allocated bit                     [30]
     67   01   54    50   F0  0E24          INSV    R0,R4,#1,(R7)           ; Insert new allocated bit              [30]
                       54   D7  0E29          DECL    R4                      ; Use previous entry next time
                    DA 52   F5  0E2B  130$:   SOBGTR  R2,120$                 ; Decrement and continue
                                0E2E
                     6544 8ED0  0E2E          POPL    (R5)[R4]                ; Insert unit at available slot
     63   01   54    01   F0  0E32          INSV    #1,R4,#1,(R3)           ; Set the select bit
              52     6544   D0  0E37          MOVL    (R5)[R4],R2             ; Get address of P-table for base       [30]
        07 0A A2     01   E1  0E3B          BBC     #HP$V_WASALL, -         ; If device was allocated               [30]
                                0E40                  HP$B_FLAGS(R2),135$     ; then,                                 [30]
     67   01   54    01   F0  0E40          INSV    #1,R4,#1,(R7)           ; Set the allocated bit                 [30]
                       05   11  0E45          BRB     140$                    ; retrun                                [32]
     67   01   54    00   F0  0E47  135$:   INSV    #0,R4,#1,(R7)           ; Clear the allocated bit               [30]
                 00FF 8F   BA  0E4C  140$:   POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7>    ; Restore                       [11]
                           05  0E50          RSB
                                0E51
```

```
                                0E51  2124
                                0E51  2125 ;+
                                0E51  2126 ; Allocate device (R5)[R4] if usermode and should be allocated
                                0E51  2127 ; and not already allocated.  Produce error messages if it fails.
                                0E51  2128 ;-
              50    01    D0    0E51  2129 150$:   MOVL    S^#SS$_NORMAL,R0          ; Assume success. esp. if S/A
     0000FE00'EF    1C    E1    0E54  2130         BBC     #DSA$V_USER, -
                   38          0E5B  2131                 DSA$GL_FLAGS, 155$        ; Exit if not usermode
                               0E5C  2132         Br_If_Inhibit_Allocate  155$      ; If bit set in DS$GL_CRD_Flags,
     00000000'EF    03    E0    0E5C               BBS     #CRD$V_Inhibit_Allocate, -                ; If bit set, branch
                   30          0E63                         L^DS$GL_CRD_Flags, 155$        ;                                [36]
                               0E64  2133                                           ; inhibit device allocation                [22]
          52    0C AE    D0    0E64  2134         MOVL    12(SP),R2                 ; GET base of CLI                          [29]
  24 00000444 E2    00    E0    0E68  2135         BBS     #CLI$V_NOALLOC, -         ;                                          [29]
                               0E70  2136                 L^CLI$L_FLAGS2(R2),155$   ; BRANCH if /NOALLOCATE                    [29]
          52    6544    D0    0E70  2137         MOVL    (R5)[R4],R2               ; Get address of P-table
       1B 0A A2    00    E1    0E74  2138         BBC     #HP$V_ALLOC, -
                               0E79  2139                 HP$B_FLAGS(R2), 155$      ; Branch if should not be allocated
       16 0A A2    01    E2    0E79  2140         BBSS    #HP$V_WASALL, -
                               0E7E  2141                 HP$B_FLAGS(R2), 155$      ; Branch if already allocated
                               0E7E  2142         $ALLOC_S HP$Q_DEVICE(R2)          ; Address of device name
                   7E    7C    0E7E                       CLRQ    -(SP)
                   00    DD    0E80                       PUSHL   #0
                   00    DD    0E82                       PUSHL   #0
                   62    7F    0E84                       PUSHAQ  HP$Q_DEVICE(R2)
     00000000'GF    05    FB    0E86                       CALLS   #5,G^SYS$ALLOC
          50    0641 8F    B1    0E8D  2143         CMPW    #SS$_DEVALRALLOC,R0       ; Already allocated to us?
                   03    12    0E92  2144         BNEQ    160$                      ; Branch if FALSE                          [34]
                 0050    31    0E94  2145 155$:   BRW     180$                      ; Branch to exit
                4D 50    E8    0E97  2146 160$:   BLBS    R0,180$                   ; Branch to exit if successful
          0A A2    02    8A    0E9A  2147         BICB    #HP$M_WASALL, -
                               0E9E  2148                 HP$B_FLAGS(R2)            ; Clear allocated bit
                   50    DD    0E9E  2149         PUSHL   R0                        ; Save return code
              7E    62    7D    0EA0  2150         MOVQ    HP$Q_DEVICE(R2),-(SP)     ; device name arg for print
     00000170'EF    9F    0EA3  2151         PUSHAB  L^T_NO_SUCH_DEV           ; Assume no such device                    [10]
          50    0908 8F    B1    0EA9  2152         CMPW    #SS$_NOSUCHDEV,R0         ; No such device?
                   29    13    0EAE  2153         BEQL    170$                      ; Branch if truth
     6E   00000243'EF    9E    0EB0  2154         MOVAB   L^T_DEVALLOC,(SP)         ; Assume allocated to other user           [10]
          50    0840 8F    B1    0EB7  2155         CMPW    #SS$_DEVALLOC,R0          ; Already allocated to another?
                   1B    13    0EBC  2156         BEQL    170$                      ; Branch if truth
     6E   00000264'EF    9E    0EBE  2157         MOVAB   L^T_DEV_MOUNTED,(SP)      ; Assume device has been mounted           [34]
          50    006C 8F    B1    0EC5  2158         CMPW    #SS$_DEVMOUNT,R0          ; Already mounted ?                        [34]
                   0D    13    0ECA  2159         BEQL    170$                      ; Branch if truth                          [34]
              5E    0C    C0    0ECC  2160         ADDL    #12,SP                    ; Remove descriptor of name and text
                   50 8ED0    0ECF  2161         POPL    R0                        ; Restore completion code
     00000000'EF    16    0ED2  2162         Jsb     L^DSR$COMPLETION          ; Other errors                             [18]
                   05    0ED8  2163         RSB                                   ; Return
                               0ED9  2164
                   01    DD    0ED9  2165 170$:   pushl   #ds$k_printf              ; Use printf to print                      [13]
                   15    DD    0EDB  2166         pushl   #ds$k_type_command_err    ; Command error type code                  [13]
     00000000'GF    05    FB    0EDD  2167         CALLS   #5, G^DSX$PRINT           ; Print the error                          [13]
                   50 8ED0    0EE4  2168         POPL    R0                        ; Restore completion code
                   05    0EE7  2169 180$:   RSB
```

```
                        0EE8  2171          .SBTTL  DSV$DESELECT    Deselect device for testing
                        0EE8  2172
                        0EE8  2173 ;++
                        0EE8  2174 ; FUNCTIONAL DESCRIPTION:
                        0EE8  2175 ;
                        0EE8  2176 ;       This routine is called from DS$CLI for each device to be
                        0EE8  2177 ;       remved from testing.  It clears the select bit
                        0EE8  2178 ;
                        0EE8  2179 ; CALLING SEQUENCE:
                        0EE8  2180 ;
                        0EE8  2181 ;       BSBW    DSV$DESELECT
                        0EE8  2182 ;
                        0EE8  2183 ; INPUT PARAMETERS:     NONE
                        0EE8  2184 ;
                        0EE8  2185 ; IMPLICIT INPUTS:
                        0EE8  2186 ;
                        0EE8  2187 ;       CLI$Q_FILE      descriptor of device to be removed
                        0EE8  2188 ;
                        0EE8  2189 ; OUTPUT PARAMETERS:    NONE
                        0EE8  2190 ;
                        0EE8  2191 ; IMPLICIT OUTPUTS:     NONE
                        0EE8  2192 ;
                        0EE8  2193 ; SIDE EFFECTS:         NONE
                        0EE8  2194 ;
                        0EE8  2195 ; COMPLETION CODES:     N/A
                        0EE8  2196 ;
                        0EE8  2197 ; REGISTER USAGE:
                        0EE8  2198 ;--
```

```
                                0EE8  2200
                                0EE8  2201 DSV$DESELECT::
                    00FF 8F     BB  0EE8  2202          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7>    ; Preserve                          [11]
                00000000'EF     16  0EEC  2203          JSB     SCRIPT$FLUSH                    ; Flush scripts if console command  [10]
                        01C6    30  0EF2  2204          BsbW    CheckAmbiguous                  ; Check for ambiguous names         [15]
                      1B 50     E9  0EF5  2205          BlbC    R0, 10$                         ; If error, hopscotch on to exit    [15]
          55    00000000'EF     DE  0EF8  2206          MOVAL   DS$GA_PTABLE,R5                 ; Point to P-table
          57    00000000'EF     DE  0EFF  2207          MOVAL   DS$GA_ALLOCATES,R7              ; Point to allocated bit mask       [30]
          53    00000000'EF     DE  0F06  2208          MOVAL   DS$GA_SELECTS,R3                ; Point to select bit mask
                        0232    30  0F0D  2209          bsbw    loc$adapter                     ; Find /Adapter, if any             [11]
                      03 50     E8  0F10  2210          blbs    r0, 20$                         ; Continue if OK                    [11]
                        009D    31  0F13  2211 10$:     brw     120$                            ; Return elsewise                   [11]
               50     08 A2     7D  0F16  2212 20$:     MOVQ    CLI$Q_FILE(R2),R0               ; Descriptor of device              [11]
                        33      13  0F1A  2213          BEQL    60$                             ; Branch if no device               [11]
          52    61    08        9C  0F1C  2214          ROTL    #8,(R1),R2                      ; Get first 3 characters of name
               52     50        90  0F20  2215          MOVB    R0,R2                           ; Insert length
          52   4C4C4103 8F      D1  0F23  2216          CMPL    #<^A"@ALL"-61>,R2               ; compare with ascic "all"
                        25      12  0F2A  2217          BNEQ    70$                             ; Branch if not all
```

```
                        0F2C  2219
                        0F2C  2220 ;+
                        0F2C  2221 ; Process DESELECT ALL command
                        0F2C  2222 ;-
      54    65    D0    0F2C  2223          MOVL    (R5),R4                  ; Get count of devices possible
          6544    D5    0F2F  2224 30$:     TSTL    (R5)[R4]                 ; Anything here?
            18    13    0F32  2225          BEQL    50$                      ; No check next
            56    D5    0F34  2226          tstl    r6                       ; If no /Adapter,                    [11]
            0A    19    0F36  2227          blss    40$                      ; ...don't check it                  [11]
   50     6544    D0    0F38  2228          movl    (r5)[r4], r0             ; Get address of link Ptable         [11]
  56    20 A0    D1    0F3C  2229          cmpl    hp$a_link(r0), r6        ; Compare                            [11]
            0A    12    0F40  2230          bneq    50$                      ; If not equal, loop to next         [11]
  00 67     54    E5    0F42  2231 40$:     BBCC    R4,(R7),45$              ; Branch if already clear and clear  [30]
  02 63     54    E5    0F46  2232 45$:     BBCC    R4,(R3),50$              ; Branch if already clear and clear  [11]
            6C    10    0F4A  2233          BSBB    130$                     ; Deallocate the device if necessary
        E0 54    F5    0F4C  2234 50$:     SOBGTR  R4,30$                   ; Do them all
            62    11    0F4F  2235 60$:     BRB     120$                     ; Branch to exit                     [11]
                        0F51  2236
```

```
                        0F51  2238
                        0F51  2239 ;+
                        0F51  2240 ; Process DESELECT device command
                        0F51  2241 ;-
            52   01  CE  0F51  2242 70$:    mnegl   #1, r2                  ; Set wildcard link                 [09]
                 56  D5  0F54  2243         tstl    r6                      ; If no /Adapter,                   [11]
                 03  19  0F56  2244         blss    80$                     ; ...don't check for it             [11]
            52   56  D0  0F58  2245         movl    r6, r2                  ; Use /ADAPTER address              [11]
                0239 30  0F5B  2246 80$:    bsbw    loc$ptable              ; Search for it                     [09]
                 46  12  0F5E  2247         BNEQ    110$                    ; Branch if found
             52  08 AE  D0  0F60  2248         MOVL    8(SP),R2                ; Get base of CLI
                 56  D5  0F64  2249         tstl    r6                      ; /ADAPTER?                         [11]
                 24  19  0F66  2250         blss    90$                     ; Branch if not                     [11]
      7E  00000004'EF  7D  0F68  2251         movq    q_adapter, -(sp)        ; Push on adapter name              [11]
            6E   6E  3C  0F6F  2252         movzwl  (sp), (sp)              ; Zero extend                       [11]
          7E  08 A2  7D  0F72  2253         movq    cli$q_file(r2), -(sp)   ; Push on device name               [11]
            6E   6E  3C  0F76  2254         movzwl  (sp), (sp)              ; Zero extend                       [11]
         0000019D'EF  9F  0F79  2255         pushab  L^t_no_such_on_adapter  ; Format string                     [11]
                 01  DD  0F7F  2256         pushl   #ds$k_printf            ; Use PRINTF                        [13]
                 15  DD  0F81  2257         pushl   #ds$k_type_command_err  ; Command error type code           [13]
   00000000'GF   07  FB  0F83  2258         calls   #7, G^dsx$print         ; Print it                          [13]
                 27  11  0F8A  2259         brb     120$                    ; Exit                              [11]
          7E  08 A2  7D  0F8C  2260 90$:    MOVQ    CLI$Q_FILE(R2),-(SP)    ; Stack file name                   [11]
            6E   6E  3C  0F90  2261 100$:   movzwl  (sp), (sp)              ; Zero extend                       [11]
         00000170'EF  9F  0F93  2262         PUSHAB  L^T_NO_SUCH_DEV         ; Edit string                       [11]
                 01  DD  0F99  2263         pushl   #ds$k_printf            ; Use PRINTF                        [13]
                 15  DD  0F9B  2264         pushl   #ds$k_type_command_err  ; Command error type code           [13]
   00000000'GF   05  FB  0F9D  2265         CALLS   #5, G^DSX$PRINT         ; Type the error                    [13]
                 0D  11  0FA4  2266         BRB     120$                    ; Branch to exit
                     0FA6  2267 110$:
            54   50  D0  0FA6  2268         MOVL    R0,R4                   ; Copy index
         00 67   54  E5  0FA9  2269         BBCC    R4,(R7),115$            ; Branch if already clear and clear [30]
         02 63   54  E5  0FAD  2270 115$:   BBCC    R4,(R3),120$            ; Branch if not selected
                 05  10  0FB1  2271         bsbb    130$                    ; Deallocate the device if necessary [11]
             00FF 8F  BA  0FB3  2272 120$:   POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7>    ; Restore                   [11]
                 05  0FB7  2273         RSB                             ; Return
                     0FB8  2274
```

```
                                0FB8  2276
                                0FB8  2277 ;+
                                0FB8  2278 ; Deallocate the device (R5)[R4] if usermode and should be deallocated
                                0FB8  2279 ; and was allocated.
                                0FB8  2280 ;-
0000FE00'EF   1C   E1   0FB8  2281 130$:   BBC     #DSA$V_USER, -
                19     0FBF  2282                 DSA$GL_FLAGS, 140$      ; Exit if not usermode
      50   6544   D0   0FC0  2283          MOVL    (R5)[R4],R0             ; Get address of P-table
   10 0A A0   00   E1   0FC4  2284          BBC     #HP$V_ALLOC, -
                        0FC9  2285                 HP$B_FLAGS(R0), 140$    ; Branch if should not be allocated
   0B 0A A0   01   E5   0FC9  2286          BBCC    #HP$V_WASALL, -
                        0FCE  2287                 HP$B_FLAGS(R0), 140$    ; Branch if already allocated
                        0FCE  2288          $DALLOC_S HP$Q_DEVICE(R0)       ; Deallocate the device
                00   DD   0FCE                     PUSHL   #0
                60   7F   0FD0                     PUSHAQ  HP$Q_DEVICE(R0)
00000000'GF     02   FB   0FD2                     CALLS   #2,G^SYS$DALLOC
                05   0FD9  2289 140$:   RSB                                 ; Return
```

```
                        0FDA  2292          .SUBTITLE       SHOW_DEV Routine
                        0FDA  2293
```

```
0FDA  2295 ;++
0FDA  2296 ; FUNCTIONAL DESCRIPTION:
0FDA  2297 ;
0FDA  2298 ;       This routine types out a line describing a single device.
0FDA  2299 ;
0FDA  2300 ; CALLING SEQUENCE:
0FDA  2301 ;
0FDA  2302 ;       BSBW    SHOW_DEV
0FDA  2303 ;
0FDA  2304 ; INPUT PARAMETERS:     NONE
0FDA  2305 ;
0FDA  2306 ; IMPLICIT INPUTS:
0FDA  2307 ;
0FDA  2308 ;       R4      Index of the PTABLE entry                    [30]
0FDA  2309 ;       R10     Address of DPB for device
0FDA  2310 ;               Contents of DS$GA_TYPLIST
0FDA  2311 ;
0FDA  2312 ; OUTPUT PARAMETERS:    NONE
0FDA  2313 ;
0FDA  2314 ; IMPLICIT OUTPUTS:     NONE
0FDA  2315 ;
0FDA  2316 ; SIDE EFFECTS:         NONE
0FDA  2317 ;
0FDA  2318 ; REGISTER USAGE:
0FDA  2319 ;
0FDA  2320 ;       R7      Address of the string to be printed
0FDA  2321 ;       R11     Address of P-table for device to be displayed
0FDA  2322 ;--
```

```
                                0FDA  2324
                     00000100   0FDA  2325 SHOWBUF_SIZ     = 256                      ; Length of buffer
                                0FDA  2326 SHOW_DEV:
                                0FDA  2327
              5E    FF00 CE     9E   0FDA  2328          MOVAB   -SHOWBUF_SIZ(SP),SP     ; Allocate buffer space
                          6E    9F   0FDF  2329          PUSHAB  (SP)                    ; Address of buffer
              7E    0100 8F     3C   0FE1  2330          MOVZWL  #SHOWBUF_SIZ,-(SP)      ; Length of buffer
                                0FE6  2331
                51    26 AB     9E   0FE6  2332          MOVAB   HP$T_TYPE(R11),R1       ; Get address of ascic type string
                      50  81     9A   0FEA  2333          MOVZBL  (R1)+,R0                ; Get length
               000011E6'EF        16   0FED  2334          Jsb     L^LOC$PTDESC            ; Locate the descriptor                 [18]
                          0D      12   0FF3  2335          BNEQ    10$                     ; Branch if located
                                0FF5  2336
        08 AE    2E2E2E2E 8F      D0   0FF5  2337          MOVL    #^A"....",8(SP)         ; String is "...."
                      6E  03      D0   0FFD  2338          MOVL    #3,(SP)                 ; Set length of string
                          4C      11   1000  2339          BRB     20$                     ; Branch to print
                                1002  2340
                          6E    7F   1002  2341 10$:      PUSHAQ  (SP)                    ; Address of descriptor
                          5B    DD   1004  2342           PUSHL   R11                     ; Address of P-table
                      51  80      9A   1006  2343          MOVZBL  (R0)+,R1                ; Get length of name
                      04 A041     9F   1009  2344          PUSHAB  4(R0)[R1]               ; Push address of start code
           FFFFF9CE EF    03      FB   100D  2345          CALLS   #3,L^PT$EXTRACT         ; Convert to ascii                      [10]
                                1014  2346
                                1014  2347 ;+
                                1014  2348 ;       Put the address of the proper string to be printed in register 7        [30]
                                1014  2349 ;-
        1A 00000000'EF    00    E0   1014  2350          BBS     #FLG$V_BRIEF,COMM_FLAGS,16$ ; Branch if in brief form          [30]
        09 00000000'EF    54    E1   101C  2351          BBC     R4,DS$GA_ALLOCATES,15$  ; Branch if not ALLOCATED               [30]
           57   000002A0'EF    9E   1024  2352          MOVAB   L^T_SHOWDEV_ALLOC,R7    ; Use allocated message                 [30]
                          21    11   102B  2353          BRB     20$                     ; Continue                              [32]
           57   000002DD'EF    9E   102D  2354 15$:      MOVAB   L^T_SHOWDEV,R7          ; Use unallocated message               [30]
                          18    11   1034  2355          BRB     20$                     ; Continue                              [32]
                                1036  2356
        09 00000000'EF    54    E1   1036  2357 16$:      BBC     R4,DS$GA_ALLOCATES,17$  ; Branch if not ALLOCATED               [30]
           57   000002C6'EF    9E   103E  2358          MOVAB   L^T_SHOWDEVB_ALLOC,R7   ; Use allocated-brief message          [30]
                          07    11   1045  2359          BRB     20$                     ; Continue                              [32]
           57   000002F9'EF    9E   1047  2360 17$:      MOVAB   L^T_SHOWDEVB,R7         ; Use unallocated-brief message         [30]
                                104E  2361
        2B 00000000'EF    00    E0   104E  2362 20$:      BBS     #FLG$V_BRIEF,COMM_FLAGS,50$ ; Branch if brief form.            [07]
                          6E    7F   1056  2363          PUSHAQ  (SP)                    ; Address of descriptor                 [07]
                      18  AB    DD   1058  2364          PUSHL   HP$A_DEVICE(R11)        ; Stuff device address
                      20  BB    7F   105B  2365          PUSHAQ  @HP$A_LINK(R11)         ; Address of link name quad desc
                          07    12   105E  2366          BNEQ    30$                     ; Branch if descriptor
           6E   0000037D'EF    7E   1060  2367          MOVAQ   Q_HUB,(SP)              ; Use HUB device name                   [11]
                      26  AB    9F   1067  2368 30$:      PUSHAB  HP$T_TYPE(R11)          ; Address of ascic type
                          6B    7F   106A  2369          PUSHAQ  HP$Q_DEVICE(R11)        ; Address of ascid device name
                          57    DD   106C  2370          PUSHL   R7                      ; Address of edit string                [10]
                          01    DD   106E  2371          pushl   #ds$k_printf            ; Use PRINTF                            [13]
                          16    DD   1070  2372          pushl   #ds$k_type_command_out  ; Command output type code             [13]
           00000000'GF    08    FB   1072  2373          CALLS   #8, G^DSX$PRINT         ; Print the line                        [13]
                                1079  2374
           5E   00000108 8F     C0   1079  2375 40$:      ADDL    #SHOWBUF_SIZ+8,SP       ; Remove descriptor and buffer          [07]
                          05    05   1080  2376          RSB                             ; Return
                      26  AB    9F   1081  2377 50$:      PUSHAB  HP$T_TYPE(R11)          ; Address of ascic type                 [07]
                          6B    7F   1084  2378          PUSHAQ  HP$Q_DEVICE(R11)        ; Address of ascid device name          [07]
                          57    DD   1086  2379          PUSHL   R7                      ; Address of edit string                [07]
                          01    DD   1088  2380          pushl   #ds$k_printf            ; Use PRINTF                            [13]
```

```
                 16   DD  108A  2381        pushl   #ds$k_type_command_out  ; Command output type code              [13]
00000000'GF      05   FB  108C  2382        CALLS   #5, G^DSX$PRINT         ; Print the line                        [13]
                 E4   11  1093  2383        BRB     40$                     ;                                       [07]
```

```
                     1095  2385          .SBTTL  INI$PTABLE      Initialize P-table
                     1095  2386
                     1095  2387 ;++
                     1095  2388 ; FUNCTIONAL DESCRIPTION:
                     1095  2389 ;
                     1095  2390 ;       This routine is used to initialize DS$GA_PTABLE and
                     1095  2391 ;       DS$GA_SELECTS, DS$GA_SELECTED.
                     1095  2392 ;
                     1095  2393 ; CALLING SEQUENCE:
                     1095  2394 ;
                     1095  2395 ;       BSBW    INI$PTABLE
                     1095  2396 ;
                     1095  2397 ; INPUT PARAMETERS:     NONE
                     1095  2398 ;
                     1095  2399 ; IMPLICIT INPUTS:      NONE
                     1095  2400 ;
                     1095  2401 ; OUTPUT PARAMETERS:    NONE
                     1095  2402 ;
                     1095  2403 ; IMPLICIT OUTPUTS:
                     1095  2404 ;
                     1095  2405 ;       DS$GA_SELECTS, DS$GA_SELECTED, DS$GA_PTABLE.
                     1095  2406 ;
                     1095  2407 ; SIDE EFFECTS:         NONE
                     1095  2408 ;
                     1095  2409 ; COMPLETION CODES:     SS$_NORMAL
                     1095  2410 ;--
```

```
                                1095  2412
                                1095  2413 INI$PTABLE::
          51   00000000'EF   DE  1095  2414          MOVAL   DS$GA_PTABLE,R1          ; Point to table
                     50   61   D0  109C  2415          MOVL    (R1),R0                ; Get count
                        6140   D4  109F  2416 10$:     CLRL    (R1)[R0]
00000000'EF   01   50   00   F0  10A2  2417          INSV    #0,R0,#1,DS$GA_SELECTS  ; Clear select bit
00000000'EF   01   50   00   F0  10AB  2418          INSV    #0,R0,#1,DS$GA_ALLOCATES ; Clear the allocate bit       [30]
                     E8 50   F5  10B4  2419          SOBGTR  R0,10$                 ; Do next unit
                     50   01   D0  10B7  2420          MOVL    S^#SS$_NORMAL,R0       ; Success of course
                                05  10BA  2421          RSB
                                    10BB  2422
```

```
                        10BB  2424          .Subtitle       Check ambiguous device
                        10BB  2425
                        10BB  2426 ;++
                        10BB  2427 ; Functional Description:
                        10BB  2428 ;
                        10BB  2429 ;       This routines searches to make sure that the /Adapter
                        10BB  2430 ;       argument (if specified) is unique; and that if /Adapter
                        10BB  2431 ;       is NOT used, the device name is unique.
                        10BB  2432 ;
                        10BB  2433 ; Calling Sequence:
                        10BB  2434 ;
                        10BB  2435 ;       BsbW    CheckAmbiguous
                        10BB  2436 ;
                        10BB  2437 ; Input Parameters:     NONE
                        10BB  2438 ;
                        10BB  2439 ; Implicit Inputs:      NONE
                        10BB  2440 ;
                        10BB  2441 ; Output Parameters:    NONE
                        10BB  2442 ;
                        10BB  2443 ; Implicit Outputs:     NONE
                        10BB  2444 ;
                        10BB  2445 ; Status Return:        R0 (LBS if OK, LBC if error)
                        10BB  2446 ;
                        10BB  2447 ;--
```

```
                                10BB
                                10BB  CheckAmbiguous:                                  ; Check for ambiguous names                [15]
          007F 8F     BB        10BB          PushR   #^M<r0,r1,r2,r3,r4,r5,R6>        ; Save registers                           [15]
               56     D4        10BF          ClrL    R6                                ; Assume we'll skip ahead to device        [15]
       0B 62   18     E1        10C1          Bbc     #Cli$V_Adapter, -                 ; If no /Adapter, don't have               [15]
                                10C5                   Cli$L_Flags(R2), 10$            ; .. to check adapter name                 [15]
               56     D6        10C5          IncL    R6                                ; Have to check adapter first              [15]
  6E   00000004'EF    7D        10C7          MovQ    Q_Adapter, (Sp)                   ; Set save descriptor to adapter name      [15]
               0C     11        10CE          Brb     20$                               ; Join common                              [15]
       52   08 AE     D0        10D0  10$:    MovL    8(Sp), R2                         ; Make sure we have original R2            [15]
       41 62   18     E0        10D4          Bbs     #Cli$V_Adapter, -                 ; Duplicates are OK if                     [15]
                                10D8                   Cli$L_Flags(R2), 60$            ; .. /Adapter was given                    [15]
       6E   08 A2     7D        10D8          MovQ    Cli$Q_File(R2), (Sp)              ; Set descriptor to device name            [15]
          6E   6E     3C        10DC  20$:    MovZWL  (Sp), (Sp)                        ; Make sure length is word                 [15]
               53     D4        10DF          ClrL    R3                                ; No matches so far                        [15]
  55   00000000'EF    DE        10E1          MovAL   Ds$GA_PTable, R5                  ; Address of list                          [15]
          54   65     D0        10E8          MovL    (R5), R4                          ; Length of list                           [15]
       52   6544      D0        10EB  30$:    MovL    (R5)[R4], R2                      ; Get address of P-tables                  [15]
               22     13        10EF          BEql    50$                               ; Branch if no entry here                  [15]
     50   62   01     C3        10F1          SubL3   #1, Hp$Q_Device(R2), R0           ; Length of name                           [15]
          6E   50     D1        10F5          CmpL    R0, (Sp)                          ; Length match                             [15]
               19     12        10F8          BNeq    50$                               ; Branch if length mismatch                [15]
       51   04 AE     D0        10FA          MovL    4(Sp), R1                         ; Get address of desired                   [15]
  52   04 A2   01     C1        10FE          AddL3   #1, Hp$Q_Device+4(R2),R2          ; Point to device name                     [15]
          81   82     91        1103  40$:    CmpB    (R2)+, (R1)+                      ; Compare a byte                           [15]
               0B     12        1106          BNeq    50$                               ; Branch if mismatch                       [15]
            F8 50     F5        1108          SobGtr  R0, 40$                           ; Continue to compare                      [15]
               53     D5        110B          TstL    R3                                ; Found a match...is it the first?         [15]
               12     12        110D          BNeq    80$                               ; No, generate an error                    [15]
       53   6544      D0        110F          MovL    (R5)[R4], R3                      ; Get address of entry                     [15]
          D5 54       F5        1113  50$:    SobGtr  R4, 30$                           ; Next if any                              [15]
          B7 56       F4        1116          SobGeq  R6, 10$                           ; If just did adapter, do device           [15]
          6E   01     D0        1119  60$:    MovL    #1, (Sp)                          ; Success                                  [15]
          007F 8F     BA        111C  70$:    PopR    #^M<r0,r1,r2,r3,r4,r5,R6>        ; Restore all registers                    [15]
                      05        1120          Rsb                                       ; Return                                   [15]
                                1121
               6E     7F        1121  80$:    PushAQ  (Sp)                              ; Duplicate name descriptor on stack       [15]
     000001C3'EF      9F        1123          PushAB  L^T_Ambig_Adap                    ; Address of format string                 [15]
          07 56       E8        1129          BlbS    R6, 90$                           ; Branch if this is adapter error          [15]
  6E   000001F0'EF    9E        112C          MovAB   L^T_Ambig_Dev, (Sp)               ; Replace format string with device        [15]
               01     DD        1133  90$:    PushL   #Ds$K_PrintF                      ; Use PrintF typeout                       [15]
               15     DD        1135          PushL   #Ds$K_Type_Command_Err            ; Command error typecode                   [15]
  00000000'GF  04     FB        1137          CallS   #4, G^Dsx$Print                   ; Print error                              [15]
               6E     D4        113E          ClrL    (Sp)                              ; Clear saved R0                           [15]
               DA     11        1140          Brb     70$                               ; Return                                   [15]
```

```
                        1142  2495          .SBTTL  Locate /Adapter value
                        1142  2496
                        1142  2497 ;++
                        1142  2498 ; Functional Description:
                        1142  2499 ;
                        1142  2500 ;       This routine uses the module-global Q_ADAPTER as the
                        1142  2501 ;       name of an adapter device, if CLI$V_ADAPTER is set.
                        1142  2502 ;       It will look up this device and return the address
                        1142  2503 ;       of the Ptable.  If the device is the special name
                        1142  2504 ;       'HUB', it will return 0.  Otherwise it will return
                        1142  2505 ;       -1.
                        1142  2506 ;
                        1142  2507 ; Calling Sequence:
                        1142  2508 ;       bsbw    loc$adapter
                        1142  2509 ;
                        1142  2510 ; Parameters: none
                        1142  2511 ;
                        1142  2512 ; Implicit inputs:
                        1142  2513 ;       q_adapter       Descriptor of adapter name
                        1142  2514 ;       cli$l_flags(r2) Checks CLI$V_ADAPTER bit.
                        1142  2515 ;
                        1142  2516 ; Outputs:
                        1142  2517 ;       R6              Address of Ptable, or -1 if not found.
                        1142  2518 ;
                        1142  2519 ; Implicit inputs: none
                        1142  2520 ;
                        1142  2521 ; Return status:
                        1142  2522 ;       R0      1 if succeeded, 0 if failed.  (Success includes no /Adapter).
                        1142  2523 ;--
```

```
                                  1142  2525
                                  1142  2526 loc$adapter::                                  ;                                        [11]
                  56   01   CE   1142  2527          mnegl   #1, r6                         ; Set to no /Adapter                     [11]
               4A 62   18   E1   1145  2528          bbc     #cli$v_adapter, -              ; If not /Adapter                        [11]
                                  1149  2529                   cli$l_flags(r2), 20$          ; ...then branch                         [11]
          50   00000004'EF   7D   1149  2530          movq    q_adapter, r0                  ; Get descriptor of adapter device       [11]
                  03   50   B1   1150  2531          cmpw    r0, #3                         ; Is length same as 'HUB'?               [11]
                       0F   12   1153  2532          bneq    10$                            ; If not, try loc$ptable lookup          [11]
  00425548 8F   61   18   00   ED   1155  2533          cmpzv   #0, #24, (r1), #^A"HUB"        ; Compare with 'HUB' keyword             [11]
                       04   12   115E  2534          bneq    10$                            ; If not, try loc$ptable                 [11]
                       56   D4   1160  2535          clrl    r6                             ; OK, set return and go home             [11]
                       2F   11   1162  2536          brb     20$                            ; Return success                         [11]
                       04   BB   1164  2537 10$:     pushr   #^M<r2>                        ; Don't let R2 get wiped out             [11]
                  52   01   CE   1166  2538          mnegl   #1, r2                         ; Find it on any link                    [11]
                       2C   10   1169  2539          bsbb    loc$ptable                     ; Find the Ptable for it                 [11]
                       04   BA   116B  2540          popr    #^M<r2>                        ; And restore it                         [11]
                  56   51   D0   116D  2541          movl    r1, r6                         ; Store address away                     [11]
                       21   12   1170  2542          bneq    20$                            ; Branch if found                        [11]
                     04 A2   D4   1172  2543          clrl    cli$l_command(r2)              ; Otherwise, clear command address       [11]
          7E   00000004'EF   7D   1175  2544          movq    q_adapter, -(sp)               ; Push adapter name desc on stack        [11]
                  6E   6E   3C   117C  2545          movzwl  (sp), (sp)                     ; Zero extend length, just in case       [11]
               00000186'EF   9F   117F  2546          pushab  L^t_no_such_adapter            ; Message to type                        [11]
                       01   DD   1185  2547          pushl   #ds$k_printf                   ; Use PRINTF                             [13]
                       15   DD   1187  2548          pushl   #ds$k_type_command_err         ; Command error type code                [13]
          00000000'GF   05   FB   1189  2549          calls   #5, G^dsx$print                ; Print it out                           [13]
                       50   D4   1190  2550          clrl    r0                             ; Return failure                         [11]
                       05   1192  2551          rsb                                    ; Return                                 [11]
                  50   01   D0   1193  2552 20$:     movl    #1, r0                         ; Set success                            [11]
                       05   1196  2553          rsb                                    ; Return                                 [11]
```

```
1197  2555          .SUBTITLE       LOC$PTABLE Routine
1197  2556
1197  2557 ;++
1197  2558 ; FUNCTIONAL DESCRIPTION:
1197  2559 ;
1197  2560 ;       This routine locates a P-table given the device name.
1197  2561 ;
1197  2562 ; CALLING SEQUENCE:
1197  2563 ;
1197  2564 ;       BSBW    LOC$PTABLE
1197  2565 ;
1197  2566 ; INPUT PARAMETERS:     NONE
1197  2567 ;
1197  2568 ; IMPLICIT INPUTS:
1197  2569 ;
1197  2570 ;       R0      Length of name to be located
1197  2571 ;       R1      Address of name to be located
1197  2572 ;       R2      Link field of Ptable to locate
1197  2573 ;
1197  2574 ; OUTPUT PARAMETERS:    NONE
1197  2575 ;
1197  2576 ; IMPLICIT OUTPUTS:
1197  2577 ;
1197  2578 ;       R0      Index into DS$GA_PTABLE of located entry
1197  2579 ;       R1      Address of P-table
1197  2580 ;
1197  2581 ; CONDITION CODES:
1197  2582 ;
1197  2583 ;       Z-bit   Clear if entry found, else set
1197  2584 ;
1197  2585 ; REGISTER USAGE:
1197  2586 ;
1197  2587 ;       R5      Address of DS$GA_PTABLE
1197  2588 ;       R4      Current index into table
1197  2589 ;--
```

```
                          1197  2591
                          1197  2592 LOC$PTABLE::
                    3F    BB  1197  2593          pushr   #^M<r0,r1,r2,r3,r4,r5>  ; Save registers                          [09]
                 6E 6E    3C  1199  2594          movzwl  (sp), (sp)             ; Make sure length is word               [11]
                 53 52    D0  119C  2595          movl    r2, r3                 ; Do something with link address         [09]
                          119F  2596
              55 00000000'EF    DE  119F  2597          MOVAL   DS$GA_PTABLE,R5        ; Address of list
                    54 65    D0  11A6  2598          MOVL    (R5),R4                ; Length of list
                 52 6544     D0  11A9  2599 10$:     MOVL    (R5)[R4],R2            ; Get address of P-tables
                       2D    13  11AD  2600          BEQL    40$                    ; Branch if no entry here
                       53    D5  11AF  2601          tstl    r3                     ; Is R2 valid, or wildcard?              [09]
                       06    19  11B1  2602          blss    20$                    ; If wildcard, continue compare          [09]
                 20 A2 53    D1  11B3  2603          cmpl    r3, hp$a_link(r2)      ; Compare link fields                    [09]
                       23    12  11B7  2604          bneq    40$                    ; Check next                             [09]
                          11B9  2605 20$:                                          ;                                        [09]
              50 62    01    C3  11B9  2606          SUBL3   #1,HP$Q_DEVICE(R2),R0  ; Length of name
                 6E    50    D1  11BD  2607          CMPL    R0,(SP)                ; Length match
                       1A    12  11C0  2608          BNEQ    40$                    ; Branch if length mismatch
                 51 04 AE    D0  11C2  2609          MOVL    4(SP),R1               ; Get address of desired
              52 04 A2       01  C1  11C6  2610          ADDL3   #1,HP$Q_DEVICE+4(R2),R2 ; Point to device name
                 81    82    91  11CB  2611 30$:     CMPB    (R2)+,(R1)+            ; Compare a byte
                       0C    12  11CE  2612          BNEQ    40$                    ; Branch if mismatch
                    F8 50    F5  11D0  2613          SOBGTR  R0,30$                 ; Continue to compare
                 55 6544     D0  11D3  2614          MOVL    (R5)[R4],R5            ; Get address of entry                   ;G:
                 6E 54       7D  11D7  2615          MOVQ    R4,(SP)                ; Superceed saved R0/R1
                       05    11  11DA  2616          BRB     50$                    ; exit
                    CA 54    F5  11DC  2617 40$:     SOBGTR  R4,10$                 ; Next if any
                       6E    7C  11DF  2618          CLRQ    (SP)                   ; Clear R0/R1
                       3F    BA  11E1  2619 50$:     popr    #^M<r0,r1,r2,r3,r4,r5> ; Restore all registers                  [09]
                       51    D5  11E3  2620          TSTL    R1                     ; Set condition codes from PT address
                             05  11E5  2621          RSB
```

```
                        11E6  2623          .SBTTL  LOC$PTDESC      Locate PT-descriptor
                        11E6  2624
                        11E6  2625 ;++
                        11E6  2626 ; FUNCTIONAL DESCRIPTION:
                        11E6  2627 ;
                        11E6  2628 ;       This routine is used to locate the PT-descriptor
                        11E6  2629 ;       given the hardware type name.
                        11E6  2630 ;
                        11E6  2631 ; CALLING SEQUENCE:
                        11E6  2632 ;
                        11E6  2633 ;       BSBW    LOC$PTDESC
                        11E6  2634 ;
                        11E6  2635 ; INPUT PARAMETERS:     NONE
                        11E6  2636 ;
                        11E6  2637 ; IMPLICIT INPUTS:
                        11E6  2638 ;
                        11E6  2639 ;       R0      Length of device type
                        11E6  2640 ;       R1      Address of device type
                        11E6  2641 ;
                        11E6  2642 ; OUTPUT PARAMETERS:    NONE
                        11E6  2643 ;
                        11E6  2644 ; IMPLICIT OUTPUTS:
                        11E6  2645 ;
                        11E6  2646 ;       R0      Address of PT-descriptor
                        11E6  2647 ;
                        11E6  2648 ; CONDITION CODES:
                        11E6  2649 ;
                        11E6  2650 ;         Z-bit   Clear if found else set
                        11E6  2651 ;
                        11E6  2652 ; SIDE EFFECTS:         NONE
                        11E6  2653 ;--
```

```
                                 11E6  2655
                                 11E6  2656 LOC$PTDESC::
                           3F    BB  11E6  2657          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save a couple
                                 11E8  2658
                           52    D4  11E8  2659          CLRL    R2                      ; Base Page 0/page 1
00000058 8F    0200      C2    D1  11EA  2660          CMPL    L$L_HEADLENGTH(R2),#^X58; Check length of Header
                           0B    12  11F3  2661          BNEQ    10$                     ; Branch if bad header
               55  021C   C2    D0  11F5  2662          MOVL    L$A_DEVP(R2),R5         ; Address of DEVTYP list
                           04    13  11FA  2663          BEQL    10$                     ; Branch if null
                           13    10  11FC  2664          BSBB    PT_LOCATE               ; Find PT-descriptor here
                           09    12  11FE  2665          BNEQ    20$                     ; Exit if found
                                 1200  2666 10$:
           55  00000000'EF    DE  1200  2667          MOVAL   DS$GA_PTDESC,R5         ; Point to descriptor list
                           08    10  1207  2668          BSBB    PT_LOCATE               ; Search this list
                                 1209  2669 20$:
                     5E    08    C0  1209  2670          ADDL    #8,SP                   ; Remove saved R0/R1
                           3C    BA  120C  2671          POPR    #^M<R2,R3,R4,R5>
                           50    D5  120E  2672          TSTL    R0                      ; Set condition codes from return
                           05    10  1210  2673          RSB                             ; Return
                                 1211  2674
                                 1211  2675 PT_LOCATE:
                     54    65    D0  1211  2676          MOVL    (R5),R4                 ; Get length of table
                           27    13  1214  2677          beql    40$                     ; Branch if table is empty              [08]
                  53    6544    D0  1216  2678 10$:     MOVL    (R5)[R4],R3             ; Get address of PT-descriptor
               50    04  AE    7D  121A  2679          MOVQ    4(SP),R0                ; Get length,address
                  50    50    3C  121E  2680          movzwl  r0, r0                  ; Zero extend word length               [11]
                  83    50    91  1221  2681          CMPB    R0,(R3)+                ; Length match?
                           14    12  1224  2682          BNEQ    30$                     ; No match, next
                     83    81    91  1226  2683 20$:     CMPB    (R1)+,(R3)+             ; Compare byte
                           0F    12  1229  2684          BNEQ    30$                     ; Branch if not found
                        F8 50    F5  122B  2685          SOBGTR  R0,20$                  ; Count and branch
            04 A3       80 8F    91  122E  2686          CMPB    #PD$_START,4(R3)        ; Verify PT-descriptor follows
                           05    12  1233  2687          BNEQ    30$                     ; Branch if not
                  50    6544    D0  1235  2688          MOVL    (R5)[R4],R0             ; Address of PT-descriptor
                           05    10  1239  2689          RSB                             ; Return with Z-bit clear
                                 123A  2690
                        D9 54    F5  123A  2691 30$:     SOBGTR  R4,10$                  ; Do next entry
                           50    D4  123D  2692 40$:     clrl    r0                      ; Not found                             [08]
                           05    10  123F  2693          RSB                             ; Return Z-bit Set
```

```
                        1240  2695          .SBTTL  INS$PTABLE      Insert P-table
                        1240  2696
                        1240  2697 ;++
                        1240  2698 ; FUNCTIONAL DESCRIPTION:
                        1240  2699 ;
                        1240  2700 ;       This routine can be called with the address of the P-table
                        1240  2701 ;       to be entered in DS$GA_PTABLE. It takes care of superceding
                        1240  2702 ;       existing Ptables for the same device name and link.
                        1240  2703 ;
                        1240  2704 ; CALLING SEQUENCE:
                        1240  2705 ;
                        1240  2706 ;       INS$PTABLE(PT)
                        1240  2707 ;
                        1240  2708 ; INPUT PARAMETERS:
                        1240  2709 ;
                        1240  2710 ;       AP      Address of P-table to insert
                        1240  2711 ;
                        1240  2712 ; IMPLICIT INPUTS:
                        1240  2713 ;
                        1240  2714 ;       DS$GA_PTABLE
                        1240  2715 ;
                        1240  2716 ; OUTPUT PARAMETERS:    NONE
                        1240  2717 ;
                        1240  2718 ; IMPLICIT OUTPUTS:
                        1240  2719 ;
                        1240  2720 ;       DS$GA_PTABLE
                        1240  2721 ;
                        1240  2722 ; COMPLETION CODES:     NONE
                        1240  2723 ;--
```

```
                                 1240  2725
                          003C   1240  2726 .ENTRY  INS$PTABLE,^M<R2,R3,R4,R5>
                    50    6C     7D  1242  2727          MOVQ    HP$Q_DEVICE(AP),R0      ; Get length/address of device name
                                 51  D6  1245  2728          INCL    R1                      ; Skip "."
                                 50  D7  1247  2729          DECL    R0                      ; Skip "-"
                      52    20   AC  D0  1249  2730          movl    hp$a_link(ap), r2       ; Link of device                        [09]
                  FFFFFF44  EF     16  124D  2731          Jsb     L^LOC$PTABLE            ; See if already exists                 [18]
                                 48  13  1253  2732          BEQL    30$                     ; Branch if not found
                                     1255  2733
                55  00000000'EF     DE  1255  2734          MOVAL   DS$GA_PTABLE,R5         ; Get address of possibles
                      54     65     D0  125C  2735          MOVL    (R5),R4                 ; Get number of possibles
                        53   6544   D0  125F  2736 10$:     MOVL    (R5)[R4],R3             ; Point to this P-table
                                 26  13  1263  2737          BEQL    20$                     ; Branch if none
                     20 A3       51  D1  1265  2738          CMPL    R1,HP$A_LINK(R3)        ; This P-table point to OLD PT for new
                                 20  12  1269  2739          BNEQ    20$                     ; Branch if not
                  26 A3   5205   8F  B1  126B  2740          CMPW    #5!<^A"R"*8>,HP$T_TYPE(R3); Check for RB730. Special case.     [20]
                                 0A  12  1271  2741          BNEQ    12$                     ; Branch if no match on first part.     [20]
              28 A3   30333742   8F  D1  1273  2742          CMPL    #^A"B730",2+HP$T_TYPE(R3) ; Finish checking                     [20]
                                 0A  13  127B  2743          BEQL    15$                     ; Skip if match. Do not change PTABLE   [20]
                 18 A3     1C   A1  CA  127D  2744 12$:     BICL    HP$A_DVA(R1),HP$A_DEVICE(R3) ; Remove old DVA bits
                 18 A3     1C   AC  C8  1282  2745          BISL    HP$A_DVA(AP),HP$A_DEVICE(R3); Insert new DVA bits
                    20 A3        5C  D0  1287  2746 15$:     MOVL    AP,HP$A_LINK(R3)        ; Point to new P-table
                          D1     54  F5  128B  2747 20$:     SOBGTR  R4,10$                  ; Branch if more to process
                        6540     5C  D0  128E  2748          MOVL    AP,(R5)[R0]             ; Use this index again
                          50     51  D0  1292  2749          MOVL    R1,R0                   ; Copy address of dynamic memory
                    00000000'EF  16  1295  2750          Jsb     L^EXE$DEANONPAGED       ; deallocate old PT                     [18]
                                 2F  11  129B  2751          BRB     60$                     ; Return normal
                                     129D  2752 30$:
                55  00000000'EF     DE  129D  2753          MOVAL   DS$GA_PTABLE,R5         ; Point to P-table list
                      54     65     D0  12A4  2754          MOVL    (R5),R4                 ; Get length of list
                           6544     D5  12A7  2755 40$:     TSTL    (R5)[R4]                ; Occupied?
                             1C     13  12AA  2756          BEQL    50$                     ; No, use it
                          F8 54     F5  12AC  2757          SOBGTR  R4,40$                  ; Search whole list
                                     12AF  2758          ERRSUP_S
                    00000000'EF     DF  12AF                        PUSHAL  $MODULE
                                 00  DD  12B5                        PUSHL   #0
                                 02  DD  12B7                        PUSHL   #$ER
                00000000'EF      03  FB  12B9                        CALLS   #3, DS_ERRSUP
                50  00660002     8F  D0  12C0  2759          MOVL    #DS$_ERROR,R0           ; Flag error
                                 04  12C7  2760          RET                             ; Return
                                     12C8  2761 50$:
                        6544     5C  D0  12C8  2762          MOVL    AP,(R5)[R4]             ; Insert pointer to PT
            00 00000000'EF       54  E5  12CC  2763 60$:     BBCC    R4,DS$GA_SELECTS,70$    ; Clear select bit
                          50     01  D0  12D4  2764 70$:     MOVL    S^#SS$_NORMAL,R0        ; Success
                                 04  12D7  2765          RET
                                     12D8  2766
                                     12D8  2767          .END
```

```
$$N                             = 00000002     D
$$T1                            = 00000001     D
$ER                             = 00000003     D
$MODULE                           00000000 R   D   02
ABORT_ATTACH                      00000001 R   D   03
ALLOC                             00000002 R   D   03
ATC$_CMDLINE                      00000004     D
ATC$_PROMPT                       00000008     D
BADPARAM                          0000075C R   D   04
BAD_LINK                          000001BF R   D   04
BAD_NAME                          000002AF R   D   04
BAD_TYPE                          00000130 R   D   04
BAD_UNIT_EXC                      00000292 R   D   04
BIT...                          = 00000004     D
BUFFER                            FFFFFF70     D
CENTRAL_BUS                       000003A6 R   D   02
CHECKAMBIGUOUS                    000010BB R   D   04
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_SIZE                        0000044C     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_FLAGS2                      00000444     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEMP_BASE                   00000448     D
CLI$L_TEST                        00000020     D
CLI$M_START_OR_RUN              = 00000008     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BASE_QUAL                 = 00000002     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NEXT                      = 00000001     D
CLI$V_NOALLOC                   = 00000000     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_START_OR_RUN              = 00000003     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
COMM_FLAGS                        00000000 R   D   03
CRD$M_CRD_DEBUG                 = 00000001     D
CRD$M_CTRL_C_NO_ECHO            = 00000002     D
CRD$M_FLUSH_CTRL_C              = 00000004     D
CRD$M_INHIBIT_ALLOCATE          = 00000008     D
CRD$V_CRD_DEBUG                 = 00000000     D
CRD$V_CTRL_C_NO_ECHO            = 00000001     D
CRD$V_FLUSH_CTRL_C              = 00000002     D
CRD$V_INHIBIT_ALLOCATE          = 00000003     D
DESC                              FFFFFFF4     D
DIR...                          = 00000001     D
DS$CNTRLC                         ********   X   00
DS$GA_ALLOCATES                   ********   X   00
DS$GA_PTABLE                      ********   X   00
DS$GA_PTDESC                      ********   X   00
DS$GA_SELECTS                     ********   X   00
DS$GB_INHIBIT_NAMING              ********   X   00
DS$GL_CLIBASE                     ********   X   00
DS$GL_CRD_FLAGS                   ********   X   00
DS$GL_FLAGS                       ********   X   00
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
DS$K_TYPE_ERRSOFT               = 00000007     D
```

```
DS$K_TYPE_ERRSUP                = 00000004    D      DS$V_BADTIME                    = 00000014    D
DS$K_TYPE_ERRSYS                = 00000005    D      DS$V_BATCH                      = 00000016    D
DS$K_TYPE_ERR_HALT              = 0000000D    D      DS$V_BRKCLR                     = 0000000C    D
DS$K_TYPE_EXCEPTION             = 0000000C    D      DS$V_BRKPT                      = 0000000B    D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D      DS$V_CHARFLG                    = 00000008    D
DS$K_TYPE_FIRST_PASS            = 00000011    D      DS$V_CMDFLG                     = 00000007    D
DS$K_TYPE_GENERAL               = 00000000    D      DS$V_CTRLC                      = 00000000    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D      DS$V_CTRLO                      = 00000010    D
DS$K_TYPE_NO_TESTS              = 00000012    D      DS$V_DEVFLG                     = 00000009    D
DS$K_TYPE_PARAM_ERROR           = 0000001C    D      DS$V_DISABLCC                   = 00000018    D
DS$K_TYPE_PROGRAM_END           = 00000010    D      DS$V_DONFLG                     = 0000000D    D
DS$K_TYPE_PROGRAM_INFO          = 00000017    D      DS$V_ERRFLG                     = 00000004    D
DS$K_TYPE_PROGRAM_START         = 0000000F    D      DS$V_EXCEPT                     = 00000013    D
DS$K_TYPE_QIO_INVADP            = 00000024    D      DS$V_EXETST                     = 00000012    D
DS$K_TYPE_QIO_NODRIVER          = 00000022    D      DS$V_HLTFLG                     = 00000003    D
DS$K_TYPE_QIO_WRONGVER          = 00000023    D      DS$V_LODFLG                     = 00000001    D
DS$K_TYPE_SCRIPT_ECHO           = 00000021    D      DS$V_MEMMGT                     = 0000000F    D
DS$K_TYPE_SCRIPT_PNF            = 0000001E    D      DS$V_NI                         = 0000001A    D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020    D      DS$V_OUTPUT                     = 00000017    D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F    D      DS$V_RUBFLG                     = 00000005    D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D      DS$V_SCRIPT                     = 00000015    D
DS$K_TYPE_START_ERR             = 00000018    D      DS$V_SETIMR                     = 00000019    D
DS$K_TYPE_START_LIST            = 00000025    D      DS$V_STRFLG                     = 00000002    D
DS$K_TYPE_SUMMARY               = 0000000E    D      DS$V_SUBT                       = 0000000E    D
DS$K_TYPE_USER_PROMPT           = 00000002    D      DS$V_SYSFLG                     = 0000000A    D
DS$K_WARNING                    = 00000000    D      DS$V_TIMED                      = 00000019    D
DS$M_ABRTFLG                    = 00000040    D      DS$V_TIMED_OUT                  = 0000001B    D
DS$M_BADTIME                    = 00100000    D      DS$V_TIMRON                     = 00000011    D
DS$M_BATCH                      = 00400000    D      DS$_AP_NORMAL_BREAK             = 00660150    D
DS$M_BRKCLR                     = 00001000    D      DS$_ARITH                       = 006600D0    D
DS$M_BRKPT                      = 00000800    D      DS$_ASBE                        = 00660118    D
DS$M_CHARFLG                    = 00000100    D      DS$_BADLINK                     = 006600F0    D
DS$M_CMDFLG                     = 00000080    D      DS$_BADTYPE                     = 006600E8    D
DS$M_CTRLC                      = 00000001    D      DS$_BIIC                        = 00660120    D
DS$M_CTRLO                      = 00010000    D      DS$_CHME                        = 006600A8    D
DS$M_DEVFLG                     = 00000200    D      DS$_CHMK                        = 006600E0    D
DS$M_DISABLCC                   = 01000000    D      DS$_DEVNAME                     = 00660108    D
DS$M_DONFLG                     = 00002000    D      DS$_ERROR                       = 00660002    D
DS$M_ERRFLG                     = 00000010    D      DS$_FHWE                        = 00660068    D
DS$M_EXCEPT                     = 00080000    D      DS$_FRAGBUF                     = 00660080    D
DS$M_EXETST                     = 00040000    D      DS$_ICBUSY                      = 006600C8    D
DS$M_HLTFLG                     = 00000008    D      DS$_ICERR                       = 006600C0    D
DS$M_LODFLG                     = 00000002    D      DS$_IHWE                        = 00660060    D
DS$M_MEMMGT                     = 00008000    D      DS$_ILLCHAR                     = 00660018    D
DS$M_NI                         = 04000000    D      DS$_ILLPAGCNT                   = 00660078    D
DS$M_OUTPUT                     = 00800000    D      DS$_ILLUNIT                     = 00660100    D
DS$M_RUBFLG                     = 00000020    D      DS$_INIT_FAIL                   = 00660140    D
DS$M_SCRIPT                     = 00200000    D      DS$_INSFMEM                     = 00660050    D
DS$M_SETIMR                     = 02000000    D      DS$_INVCPU                      = 00660130    D
DS$M_STRFLG                     = 00000004    D      DS$_IPL2HI                      = 006600B8    D
DS$M_SUBT                       = 00004000    D      DS$_IVADDR                      = 00660040    D
DS$M_SYSFLG                     = 00000400    D      DS$_IVVECT                      = 00660038    D
DS$M_TIMED                      = 02000000    D      DS$_KRNLSTK                     = 00660090    D
DS$M_TIMED_OUT                  = 08000000    D      DS$_LOGIC                       = 00660070    D
DS$M_TIMRON                     = 00020000    D      DS$_MCHK                        = 00660088    D
DS$PRINTF                         ********    X  00  DS$_MEM_ALLOC_ERR               = 00660138    D
DS$V_ABRTFLG                    = 00000006    D      DS$_MMOFF                       = 00660058    D
```

```
DS$_NEEDUNIT                    = 006600F8    D
DS$_NODE                        = 00660128    D
DS$_NOPCS                       = 00660110    D
DS$_NORMAL                      = 00660001    D
DS$_NOSUPPORT                   = 006600B1    D
DS$_NOTALLOWMP                  = 00660148    D
DS$_NOTDON                      = 00660030    D
DS$_NOTIMP                      = 006600B0    D
DS$_NULLSTR                     = 00660010    D
DS$_OVERFLOW                    = 00660008    D
DS$_POWER                       = 00660098    D
DS$_PROGERR                     = 00660020    D
DS$_SEVERE                      = 00660004    D
DS$_TRANSL                      = 006600A0    D
DS$_TRUNCATE                    = 00660028    D
DS$_UNEXPINT                    = 006600D8    D
DS$_UNSUPPDEV                   = 00660158    D
DS$_VASFULL                     = 00660048    D
DS$_WARNING                     = 00660000    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSA$V_APT                       = 0000001F    D
DSA$V_USER                      = 0000001C    D
DSR$COMPLETION                    ********    X   00
DSR$IFLOADDEV                     ********    X   00
DSV$ATTACH                        00000000 RG  D   04
DSV$DEATTACH                      00000C40 RG  D   04
DSV$DESELECT                      00000EE8 RG  D   04
DSV$SELECT                        00000D08 RG  D   04
DSV$SHOWDEVICE                    00000BAE RG  D   04
DSV$SHOWDEVICEB                   00000BA7 RG  D   04
DSV$SHOWSELECT                    00000C0F RG  D   04
DSV$SHOWSELECTB                   00000C08 RG  D   04
DSX$ATTACH                        0000010A RG  D   04
DSX$PRINT                         ********    X   00
DS_ERRSUP                         ********    X   00
DS_SUPERLINE                      ********    X   00
EARG$_BASE                        00000008      D
EARG$_PC                          00000004      D
EARG$_TEXT                        0000000C      D
ERROR_EXIT                        0000047B R    D   04
EXE$ALONONPAGED                   ********    X   00
EXE$DEANONPAGED                   ********    X   00
EXTRACT                           000009F4 R    D   04
EX_ADD                            00000A48 R    D   04
EX_BADPARAM                       00000A17 R    D   04
EX_CASE                           00000A4C R    D   04
EX_COMMON                         00000B2C R    D   04
EX_COMPLEMENT                     00000A4A R    D   04
EX_DECIMAL                        00000A1D R    D   04
EX_EPB                            00000A60 R    D   04
EX_FETCH                          00000A48 R    D   04
EX_FINISH                         00000A65 R    D   04
EX_HEXADECIMAL                    00000A1D R    D   04
EX_LITERAL                        00000A48 R    D   04
EX_LOGICAL                        00000A3A R    D   04
EX_NAME                           00000A55 R    D   04
EX_OCTAL                          00000A1D R    D   04
EX_START                          00000A1B R    D   04
EX_STORE                          00000A72 R    D   04
EX_STRING                         00000A28 R    D   04
FLG$V_BRIEF                     = 00000000      D
FORMAT_PROMPT                     00000B7E R    D   04
FPT$_ARGS                         00000008      D
FPT$_FORMAT                       00000004      D
GENERIC                           FFFFFFEA      D
GET_ATTRIBUTES                    00000490 R    D   04
GET_LINE                          00000B56 R    D   04
GET_LINK                          000001DC R    D   04
GET_NAME                          000003CC R    D   04
GET_TYPE                          0000014D R    D   04
HP$A_DEPENDENT                    00000032      D
HP$A_DEVICE                       00000018      D
HP$A_DVA                          0000001C      D
HP$A_LINK                         00000020      D
HP$B_DRIVE                        0000000B      D
HP$B_FLAGS                        0000000A      D
HP$C_NAMELEN                    = 00000014    G D
HP$M_NAME                       = 00000004      D
HP$M_WASALL                     = 00000002      D
HP$Q_DEVICE                       00000000      D
HP$T_DEVICE                       0000000C      D
HP$T_TYPE                         00000026      D
HP$V_ALLOC                      = 00000000      D
HP$V_NAME                       = 00000002      D
HP$V_WASALL                     = 00000001      D
HP$W_SIZE                         00000008      D
HP$W_VECTOR                       00000024      D
HPE$A_EPB                         00000016      D
HPE$C_EPB_SIZE                  = 0000001A    G D
HPE$T_DEVICE                      00000000      D
HPE$W_EXT_DRIVE                   00000014      D
IARG$_BASE                        00000008      D
IARG$_PC                          00000004      D
IARG$_PROMPT                      00000010      D
IARG$_TEXT                        0000000C      D
IF_ALPHA                          00000548 R    D   04
INI$PTABLE                        00001095 RG   D   04
INS$PTABLE                        00001240 RG   D   04
```

```
INTERPRET                       00000735 R  D  04
KB_CHECK                        ******** X  00
L$A_CCP                         00000240    D
L$A_DEVP                        0000021C    D
L$A_DREG                        00000224    D
L$A_DTP                         00000218    D
L$A_ICP                         0000023C    D
L$A_LASTADR                     00000214    D
L$A_NAME                        00000208    D
L$A_REPP                        00000244    D
L$A_SECNAM                      00000250    D
L$A_STATAB                      00000248    D
L$A_TSTCNT                      00000254    D
L$L_ENVIRON                     00000204    D
L$L_ERRTYP                      0000024C    D
L$L_HEADLENGTH                  00000200    D
L$L_REV                         0000020C    D
L$L_UNIT                        00000220    D
L$L_UNUSED                      00000228    D
L$L_UPDATE                      00000210    D
LOC$ADAPTER                     00001142 RG D  04
LOC$PTABLE                      00001197 RG D  04
LOC$PTDESC                      000011E6 RG D  04
LOCAL                           FFFFFF70    D
LOCAL_BLOCK                     FFFFFFE6    D
MAX_UNIT                        FFFFFFFC    D
NAME_FLAGS                      FFFFFFFF    D
OLDPC                           FFFFFFFC    D
ONLY_ATTACH                     ******** X  00
PARENT_CONTROLLER               FFFFFFFE    D
PARENT_PTABLE                   FFFFFFF4    D
PARSE_DEVICE                    0000055D RG D  04
PART_LIST                       00000388 R  D  02
PD$_ADD                       = 0000008A    D
PD$_CASE                      = 0000008C    D
PD$_COMPLEMENT                = 00000089    D
PD$_DECIMAL                   = 00000082    D
PD$_END                       = 00000081    D
PD$_EPB                       = 0000008E    D
PD$_FETCH                     = 00000087    D
PD$_HEXADECIMAL               = 00000084    D
PD$_LITERAL                   = 00000086    D
PD$_LOGICAL                   = 0000008B    D
PD$_NAME                      = 0000008D    D
PD$_OCTAL                     = 00000083    D
PD$_START                     = 00000080    D
PD$_STORE                     = 00000088    D
PD$_STRING                    = 00000085    D
PD_ADD                          00000795 R  D  04
PD_CASE                         00000827 R  D  04
PD_COMPLEMENT                   00000790 R  D  04
PD_DECIMAL                      00000856 R  D  04
PD_EPB                          00000850 R  D  04
PD_FETCH                        0000076E R  D  04
PD_FINISH                       00000765 R  D  04
PD_HEXADECIMAL                  000008B7 R  D  04
PD_LITERAL                      00000769 R  D  04
PD_LOGICAL                      000007AF R  D  04
PD_NAME                         00000844 R  D  04
PD_NUMERIC                      000008EB R  D  04
PD_OCTAL                        00000887 R  D  04
PD_START                        00000760 R  D  04
PD_STORE                        0000077F R  D  04
PD_STRING                       00000953 R  D  04
PT$EXTRACT                      000009E2 RG D  04
PT$INTERPRET                    0000070C RG D  04
PTD$M_CONTROLLER              = 00000002    D
PTD$M_DEVICE                  = 00000003    D
PTD$M_ENDDEVICE               = 0000001B    D
PTD$M_INHERIT                 = 00000018    D
PTD$M_INHERITED               = 00000008    D
PTD$M_INHERIT_CON             = 00000010    D
PTD$M_INHERIT_PRE             = 00000008    D
PTD$M_NAME                    = 00000004    D
PTD$M_UNIT                    = 00000001    D
PTD$V_CONTROLLER              = 00000001    D
PTD$V_INHERIT_CON             = 00000004    D
PTD$V_INHERIT_PRE             = 00000003    D
PTD$V_NAME                    = 00000002    D
PTD$V_UNIT                    = 00000000    D
PTDESC                          FFFFFFF8    D
PT_LOCATE                       00001211 R  D  04
Q_ADAPTER                       00000004 RG D  03
Q_HUB                           0000037D R  D  02
RANGE                           00000003 R  D  03
SAVABS...                     = 0000000C    D
SCAN$ALPHA                      ******** X  00
SCAN$ALPHANUM                   ******** X  00
SCAN$DECIMAL                    ******** X  00
SCAN$DEVICE                     ******** X  00
SCAN$NUMERIC                    ******** X  00
SCAN$SPACES                     ******** X  00
SCAN$SYMBOL                     ******** X  00
SCAN_ALPHA                      000006F4 R  D  04
SCAN_ALPHANUM                   00000700 R  D  04
SCRIPT$FLUSH                    ******** X  00
SET_ABORT                       000000FD RG D  04
SHOWBUF_SIZ                   = 00000100    D
SHOW_DEV                        00000FDA R  D  04
SIZ...                        = 00000001    D
SKIP                            00000A41 R  D  04
SS$_BADPARAM                  = 00000014    D
SS$_DEVALLOC                  = 00000840    D
SS$_DEVALRALLOC               = 00000641    D
SS$_DEVMOUNT                  = 0000006C    D
SS$_INSFARG                   = 00000114    D
SS$_NORMAL                    = 00000001    D
SS$_NOSUCHDEV                 = 00000908    D
ST_DECIMAL                      00000AA4 R  D  04
ST_HEXADECIMAL                  00000AC6 R  D  04
ST_LOGICAL                      00000B02 R  D  04
ST_OCTAL                        00000AB4 R  D  04
ST_STRING                       00000AD7 R  D  04
SYS$ALLOC                       ******** GX 00
```

```
SYS$DALLOC                      ********  GX    00
SYS$FAO                         ********   X    00
SYS$FAOL                        ********  G     04
TEMP_STORE                      0000000C R  D   03
T_ALLOC                         000000A0 R  D   02
T_AMBIG_ADAP                    000001C3 R  D   02
T_AMBIG_DEV                     000001F0 R  D   02
T_BAD_NAME                      0000002E R  D   02
T_BAD_RANGE                     0000006B R  D   02
T_DEATTACH                      00000227 R  D   02
T_DEVALLOC                      00000243 R  D   02
T_DEV_MOUNTED                   00000264 R  D   02
T_INVALID                       00000007 R  D   02
T_LINK                          00000158 R  D   02
T_MAIN_BUS                      00000109 R  D   02
T_NAME                          0000014C R  D   02
T_NO_SUCH_ADAPTER               00000186 R  D   02
T_NO_SUCH_DEV                   00000170 R  D   02
T_NO_SUCH_ON_ADAPTER            0000019D R  D   02
T_PART_OF                       000000CD R  D   02
T_PNF                           00000306 R  D   02
T_PROMPT                        0000013F R  D   02
T_RANGE                         000000A8 R  D   02
T_SHOULDBE                      00000348 R  D   02
T_SHOWDEV                       000002DD R  D   02
T_SHOWDEVB                      000002F9 R  D   02
T_SHOWDEVB_ALLOC                000002C6 R  D   02
T_SHOWDEV_ALLOC                 000002A0 R  D   02
T_SKIP                          00000333 R  D   02
T_STRING                        00000360 R  D   02
T_TYPE                          00000164 R  D   02
T_UNIT_EXCESS                   000000D7 R  D   02
T_UNRANGE                       000000BA R  D   02
T_YESNO                         00000352 R  D   02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFF (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	000003BD (957.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
WORK	00000010 (16.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	000012D8 (4824.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000002	1969 (4)	1969 (4)
$$T1	=00000001	2142 (4)	2142 (4)
$ER	=00000003	2758 (6)	#-2758 (6) #-449 (1)
$MODULE	00000000-R	265 (1)	2758 (6) 449 (1)
ABORT_ATTACH	00000001-R	350 (1)	#-392 (1) #-430 (1) #-458 (1)
ALLOC	00000002-R	352 (1)	#-1009 (3) #-1017 (3) #-1095 (3) #-749 (1)
			#-990 (3) #-998 (3)
ATC$_CMDLINE	00000004	514 (1)	529 (1)
ATC$_PROMPT	00000008	514 (1)	534 (1)
BADPARAM	0000075C-R	1258 (3)	#-1239 (3)
BAD_LINK	000001BF-R	590 (1)	#-605 (1) #-651 (1)
BAD_NAME	000002AF-R	702 (1)	#-788 (1)
BAD_TYPE	00000130-R	543 (1)	#-557 (1) #-561 (1)
BAD_UNIT_EXC	00000292-R	660 (1)	#-801 (1)
BIT...	=00000004	236 (1)	222 (1) 223 (1) 228 (1) 233 (1)
			235 (1) 236 (1)
BUFFER	FFFFFF70	1223 (3)	1230 (3) 1440 (3)
CENTRAL_BUS	000003A6-R	335 (1)	616 (1)
CHECKAMBIGUOUS	000010BB-R	2450 (6)	#-2011 (4) #-2204 (4)
CLI$K_BUFSIZ	=00000100	232 (1)	232 (1)
CLI$K_SIZE	0000044C	232 (1)	
CLI$L_ADDRESS	00000018	232 (1)	
CLI$L_COMMAND	00000004	232 (1)	#-2543 (6)
CLI$L_DATA	0000001C	232 (1)	
CLI$L_FLAGS	00000000	232 (1)	2454 (6) 2460 (6) 2529 (6)
CLI$L_FLAGS2	00000444	232 (1)	2136 (4)
CLI$L_LAST	00000024	232 (1)	
CLI$L_NEXT	00000030	232 (1)	
CLI$L_PASS	0000002C	232 (1)	
CLI$L_SUBT	00000028	232 (1)	
CLI$L_TEMP_BASE	00000448	232 (1)	
CLI$L_TEST	00000020	232 (1)	
CLI$M_START_OR_RUN	=00000008	232 (1)	
CLI$Q_BUFQWD	00000034	232 (1)	
CLI$Q_FILE	00000008	232 (1)	#-1795 (4) #-1916 (4) #-1938 (4) #-2019 (4)
			#-2072 (4) #-2079 (4) #-2212 (4) #-2253 (4)
			#-2260 (4) #-2461 (6) #-404 (1) 405 (1)
			#-407 (1) 410 (1) #-432 (1) 434 (1)
			#-436 (1) #-440 (1)
CLI$Q_SECTION	00000010	232 (1)	
CLI$Q_TIME	0000043C	232 (1)	
CLI$T_BUFFER	0000003C	232 (1)	
CLI$V_ADAPTER	=00000018	232 (1)	#-2453 (6) #-2459 (6) #-2528 (6)
CLI$V_ADR	=0000000B	232 (1)	
CLI$V_ASCII	=00000013	232 (1)	
CLI$V_BASE_QUAL	=00000002	232 (1)	
CLI$V_BREAK	=0000000A	232 (1)	
CLI$V_BRIEF	=0000001B	232 (1)	
CLI$V_BYTE	=0000000D	232 (1)	
CLI$V_CLEAR	=00000002	232 (1)	

```
CLI$V_DEC                       =00000010     232     (1)
CLI$V_DEFAULT                   =0000000C     232     (1)
CLI$V_DEPOSIT                   =00000019     232     (1)
CLI$V_EVENT                     =00000008     232     (1)
CLI$V_EXAM                      =00000005     232     (1)
CLI$V_FLAGS                     =00000009     232     (1)
CLI$V_HEX                       =00000012     232     (1)
CLI$V_KERNEL                    =00000017     232     (1)
CLI$V_LOAD                      =00000006     232     (1)
CLI$V_LONG                      =0000000F     232     (1)
CLI$V_NEXT                      =00000001     232     (1)
CLI$V_NOALLOC                   =00000000     232     (1)     #-2135   (4)
CLI$V_NOTNUF                    =00000001     232     (1)
CLI$V_OCT                       =00000011     232     (1)
CLI$V_PREG                      =0000001A     232     (1)
CLI$V_QA                        =00000007     232     (1)
CLI$V_QACKLOOPLOOPS             =0000001C     232     (1)
CLI$V_QAERRORPRINTS             =0000001B     232     (1)
CLI$V_QAMULTIPLEPASS            =0000001F     232     (1)
CLI$V_QASUBTESTLOOPS            =0000001E     232     (1)
CLI$V_QATESTLOOPS               =0000001D     232     (1)
CLI$V_REG                       =00000014     232     (1)
CLI$V_REQUIRED                  =00000000     232     (1)
CLI$V_RUN                       =00000015     232     (1)
CLI$V_SET                       =00000003     232     (1)
CLI$V_SHOW                      =00000004     232     (1)
CLI$V_START_OR_RUN              =00000003     232     (1)      232     (1)
CLI$V_VALSEC                    =00000016     232     (1)
CLI$V_WORD                      =0000000E     232     (1)
COMM_FLAGS                       00000000-R   348     (1)     #-1793   (4)     #-1825   (4)     #-1863   (4)     #-1877   (4)
                                                                  2350   (6)       2362   (6)

CRD$M_CRD_DEBUG                 =00000001     236     (1)
CRD$M_CTRL_C_NO_ECHO            =00000002     236     (1)
CRD$M_FLUSH_CTRL_C              =00000004     236     (1)
CRD$M_INHIBIT_ALLOCATE          =00000008     236     (1)
CRD$V_CRD_DEBUG                 =00000000     236     (1)
CRD$V_CTRL_C_NO_ECHO            =00000001     236     (1)
CRD$V_FLUSH_CTRL_C              =00000002     236     (1)
CRD$V_INHIBIT_ALLOCATE          =00000003     236     (1)     #-2132   (4)
DESC                             FFFFFFF4     1223    (3)     #-1230   (3)     #-1454   (3)       1456   (3)
DIR...                          =00000001     1741    (4)       1217   (3)       1223   (3)       1506   (3)       1741   (4)
                                                                  514   (1)        524   (1)
DS$CNTRLC                        00000000-XR                      255   (1)        393   (1)       445   (1)
DS$GA_ALLOCATES                  00000000-XR                     2015   (4)       2207   (4)      2351   (6)      2357   (6)
                                                                 2418   (6)        247   (1)
DS$GA_PTABLE                     00000000-XR                     1798   (4)       1867   (4)      1913   (4)      2013   (4)
                                                                 2206   (4)       2414   (6)      2464   (6)       251   (1)
                                                                 2597   (6)       2734   (6)      2753   (6)
DS$GA_PTDESC                     00000000-XR                      251   (1)       2667   (6)
DS$GA_SELECTS                    00000000-XR                     1870   (4)       1972   (4)      2014   (4)      2208   (4)
                                                                 2417   (6)        244   (1)      2763   (6)
DS$GB_INHIBIT_NAMING             00000000-XR                      255   (1)      #-931   (3)
DS$GL_CLIBASE                    00000000-XR                      244   (1)
DS$GL_CRD_FLAGS                  00000000-XR                     2132   (4)        256   (1)
DS$GL_FLAGS                      00000000-XR                      253   (1)        416   (1)
DS$K_ERROR                      =00000002     228     (1)
DS$K_NORMAL                     =00000001     228     (1)
```

```
DS$K_PRINTB                     =00000002     235     (1)
DS$K_PRINTF                     =00000001     235     (1)     #-1941  (4)     #-1969  (4)     #-2075  (4)     #-2082  (4)
                                                              #-2165  (4)     #-2256  (4)     #-2263  (4)     #-2371  (6)
                                                              #-2380  (6)     #-2489  (6)     #-2547  (6)     #-419   (1)
                                                              #-423   (1)
DS$K_PRINTI                     =00000000     235     (1)
DS$K_PRINTX                     =00000003     235     (1)
DS$K_SEVERE                     =00000004     228     (1)
DS$K_SUBSYS                     =00000066     228     (1)       228   (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     235     (1)
DS$K_TYPE_ABORT_TEST            =00000013     235     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     235     (1)     #-1942  (4)     #-2076  (4)     #-2083  (4)     #-2166  (4)
                                                              #-2257  (4)     #-2264  (4)     #-2490  (6)     #-2548  (6)
                                                              #-420   (1)     #-424   (1)
DS$K_TYPE_COMMAND_OUT           =00000016     235     (1)     #-1969  (4)     #-2372  (6)     #-2381  (6)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     235     (1)
DS$K_TYPE_DS_PROMPT             =00000001     235     (1)
DS$K_TYPE_DS_START              =0000001D     235     (1)
DS$K_TYPE_ERRDEV                =00000008     235     (1)
DS$K_TYPE_ERRHARD               =00000006     235     (1)
DS$K_TYPE_ERROR_BODY            =00000009     235     (1)
DS$K_TYPE_ERROR_END             =0000000A     235     (1)
DS$K_TYPE_ERRPREP               =0000001B     235     (1)
DS$K_TYPE_ERRSOFT               =00000007     235     (1)
DS$K_TYPE_ERRSUP                =00000004     235     (1)
DS$K_TYPE_ERRSYS                =00000005     235     (1)
DS$K_TYPE_ERR_HALT              =0000000D     235     (1)
DS$K_TYPE_EXCEPTION             =0000000C     235     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     235     (1)
DS$K_TYPE_FIRST_PASS            =00000011     235     (1)
DS$K_TYPE_GENERAL               =00000000     235     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     235     (1)
DS$K_TYPE_NO_TESTS              =00000012     235     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     235     (1)
DS$K_TYPE_PROGRAM_END           =00000010     235     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     235     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     235     (1)
DS$K_TYPE_QIO_INVADP            =00000024     235     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     235     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     235     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     235     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     235     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     235     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     235     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     235     (1)
DS$K_TYPE_START_ERR             =00000018     235     (1)
DS$K_TYPE_START_LIST            =00000025     235     (1)
DS$K_TYPE_SUMMARY               =0000000E     235     (1)
DS$K_TYPE_USER_PROMPT           =00000002     235     (1)
DS$K_WARNING                    =00000000     228     (1)
DS$M_ABRTFLG                    =00000040     233     (1)
DS$M_BADTIME                    =00100000     233     (1)
DS$M_BATCH                      =00400000     233     (1)
DS$M_BRKCLR                     =00001000     233     (1)
DS$M_BRKPT                      =00000800     233     (1)
DS$M_CHARFLG                    =00000100     233     (1)
DS$M_CMDFLG                     =00000080     233     (1)
```

```
DS$M_CTRLC                      =00000001     233     (1)
DS$M_CTRLO                      =00010000     233     (1)
DS$M_DEVFLG                     =00000200     233     (1)
DS$M_DISABLCC                   =01000000     233     (1)
DS$M_DONFLG                     =00002000     233     (1)
DS$M_ERRFLG                     =00000010     233     (1)
DS$M_EXCEPT                     =00080000     233     (1)
DS$M_EXETST                     =00040000     233     (1)
DS$M_HLTFLG                     =00000008     233     (1)
DS$M_LODFLG                     =00000002     233     (1)
DS$M_MEMMGT                     =00008000     233     (1)
DS$M_NI                         =04000000     233     (1)
DS$M_OUTPUT                     =00800000     233     (1)
DS$M_RUBFLG                     =00000020     233     (1)
DS$M_SCRIPT                     =00200000     233     (1)
DS$M_SETIMR                     =02000000     233     (1)
DS$M_STRFLG                     =00000004     233     (1)
DS$M_SUBT                       =00004000     233     (1)
DS$M_SYSFLG                     =00000400     233     (1)
DS$M_TIMED                      =02000000     233     (1)
DS$M_TIMED_OUT                  =08000000     233     (1)
DS$M_TIMRON                     =00020000     233     (1)
DS$PRINTF                        00000000-XR                  245     (1)
DS$V_ABRTFLG                    =00000006     233     (1)
DS$V_BADTIME                    =00000014     233     (1)
DS$V_BATCH                      =00000016     233     (1)
DS$V_BRKCLR                     =0000000C     233     (1)
DS$V_BRKPT                      =0000000B     233     (1)
DS$V_CHARFLG                    =00000008     233     (1)
DS$V_CMDFLG                     =00000007     233     (1)
DS$V_CTRLC                      =00000000     233     (1)
DS$V_CTRLO                      =00000010     233     (1)
DS$V_DEVFLG                     =00000009     233     (1)
DS$V_DISABLCC                   =00000018     233     (1)
DS$V_DONFLG                     =0000000D     233     (1)
DS$V_ERRFLG                     =00000004     233     (1)
DS$V_EXCEPT                     =00000013     233     (1)
DS$V_EXETST                     =00000012     233     (1)
DS$V_HLTFLG                     =00000003     233     (1)
DS$V_LODFLG                     =00000001     233     (1)
DS$V_MEMMGT                     =0000000F     233     (1)
DS$V_NI                         =0000001A     233     (1)
DS$V_OUTPUT                     =00000017     233     (1)
DS$V_RUBFLG                     =00000005     233     (1)
DS$V_SCRIPT                     =00000015     233     (1)     #-415     (1)
DS$V_SETIMR                     =00000019     233     (1)
DS$V_STRFLG                     =00000002     233     (1)
DS$V_SUBT                       =0000000E     233     (1)
DS$V_SYSFLG                     =0000000A     233     (1)
DS$V_TIMED                      =00000019     233     (1)
DS$V_TIMED_OUT                  =0000001B     233     (1)
DS$V_TIMRON                     =00000011     233     (1)
DS$_AP_NORMAL_BREAK             =00660150     228     (1)
DS$_ARITH                       =006600D0     228     (1)
DS$_ASBE                        =00660118     228     (1)
DS$_BADLINK                     =006600F0     228     (1)     #-594     (1)     #-638     (1)
DS$_BADTYPE                     =006600E8     228     (1)     #-547     (1)
```

```
DS$_BIIC                        =00660120     228    (1)
DS$_CHME                        =006600A8     228    (1)
DS$_CHMK                        =006600E0     228    (1)
DS$_DEVNAME                     =00660108     228    (1)    #-756    (1)    #-765    (1)    #-773    (1)
DS$_ERROR                       =00660002     228    (1)    #-2759   (6)
DS$_FHWE                        =00660068     228    (1)
DS$_FRAGBUF                     =00660080     228    (1)
DS$_ICBUSY                      =006600C8     228    (1)
DS$_ICERR                       =006600C0     228    (1)
DS$_IHWE                        =00660060     228    (1)
DS$_ILLCHAR                     =00660018     228    (1)
DS$_ILLPAGCNT                   =00660078     228    (1)
DS$_ILLUNIT                     =00660100     228    (1)    #-664    (1)
DS$_INIT_FAIL                   =00660140     228    (1)
DS$_INSFMEM                     =00660050     228    (1)
DS$_INVCPU                      =00660130     228    (1)
DS$_IPL2HI                      =006600B8     228    (1)
DS$_IVADDR                      =00660040     228    (1)
DS$_IVVECT                      =00660038     228    (1)
DS$_KRNLSTK                     =00660090     228    (1)
DS$_LOGIC                       =00660070     228    (1)
DS$_MCHK                        =00660088     228    (1)
DS$_MEM_ALLOC_ERR               =00660138     228    (1)
DS$_MMOFF                       =00660058     228    (1)
DS$_NEEDUNIT                    =006600F8     228    (1)
DS$_NODE                        =00660128     228    (1)
DS$_NOPCS                       =00660110     228    (1)
DS$_NORMAL                      =00660001     228    (1)
DS$_NOSUPPORT                   =006600B1     228    (1)
DS$_NOTALLOWMP                  =00660148     228    (1)
DS$_NOTDON                      =00660030     228    (1)
DS$_NOTIMP                      =006600B0     228    (1)      228    (1)
DS$_NULLSTR                     =00660010     228    (1)
DS$_OVERFLOW                    =00660008     228    (1)
DS$_POWER                       =00660098     228    (1)
DS$_PROGERR                     =00660020     228    (1)
DS$_SEVERE                      =00660004     228    (1)
DS$_TRANSL                      =006600A0     228    (1)
DS$_TRUNCATE                    =00660028     228    (1)
DS$_UNEXPINT                    =006600D8     228    (1)
DS$_UNSUPPDEV                   =00660158     228    (1)
DS$_VASFULL                     =00660048     228    (1)
DS$_WARNING                     =00660000     228    (1)      228    (1)
DSA$GL_FLAGS                     0000FE00                     2131   (4)      2282   (4)      414    (1)
DSA$GL_SID                       0000FE14                  #-631    (1)
DSA$V_APT                       =0000001F                  #-413    (1)
DSA$V_USER                      =0000001C                  #-2130   (4)    #-2281   (4)
DSR$COMPLETION                   00000000-XR                 2162   (4)      252    (1)
DSR$IFLOADDEV                    00000000-XR                 2040   (4)      252    (1)
DSV$ATTACH                       00000000-R    391    (1)
DSV$DEATTACH                     00000C40-R    1910   (4)
DSV$DESELECT                     00000EE8-R    2201   (4)
DSV$SELECT                       00000D08-R    2008   (4)
DSV$SHOWDEVICE                   00000BAE-R    1794   (4)
DSV$SHOWDEVICEB                  00000BA7-R    1792   (4)
DSV$SHOWSELECT                   00000C0F-R    1864   (4)
DSV$SHOWSELECTB                  00000C08-R    1862   (4)
```

```
DSX$ATTACH                      0000010A-R   526   (1)      411   (1)
DSX$PRINT                       00000000-XR              1943   (4)     1969   (4)     2077   (4)     2084   (4)
                                                         2167   (4)     2258   (4)     2265   (4)     2373   (6)
                                                         2382   (6)      246   (1)     2491   (6)     2549   (6)
                                                          255   (1)      421   (1)      425   (1)
DS_ERRSUP                       00000000-XR              245   (1)     2758   (6)      449   (1)
DS_SUPERLINE                    00000000-XR              245   (1)      438   (1)
EARG$_BASE                      00000008     1506   (3)
EARG$_PC                        00000004     1506   (3)    #-1510   (3)
EARG$_TEXT                      0000000C     1506   (3)    #-1511   (3)    #-1583   (3)    #-1584   (3)
ERROR_EXIT                      0000047B-R   838    (2)     #-548   (1)    #-573   (1)    #-595   (1)    #-601   (1)
                                                          #-639   (1)    #-665   (1)    #-758   (1)    #-767   (1)
                                                          #-775   (1)    #-784   (1)    #-833   (2)    #-835   (2)
EXE$ALONONPAGED                 00000000-XR               246   (1)      570   (1)
EXE$DEANONPAGED                 00000000-XR              1974   (4)      246   (1)     2750   (6)      844   (3)
EXTRACT                         000009F4-R   1515   (3)    #-1537   (3)    #-1544   (3)    #-1553   (3)     1556   (3)
                                                          #-1567   (3)    #-1571   (3)    #-1576   (3)    #-1580   (3)
                                                          #-1596   (3)    #-1604   (4)    #-1656   (4)
EX_ADD                          00000A48-R   1564   (3)     1531   (3)
EX_BADPARAM                     00000A17-R   1532   (3)    #-1514   (3)
EX_CASE                         00000A4C-R   1568   (3)     1531   (3)
EX_COMMON                       00000B2C-R   1641   (4)    #-1607   (4)    #-1611   (4)    #-1615   (4)    #-1625   (4)
                                                          #-1627   (4)    #-1636   (4)
EX_COMPLEMENT                   00000A4A-R   1566   (3)     1531   (3)
EX_DECIMAL                      00000A1D-R   1538   (3)     1531   (3)
EX_EPB                          00000A60-R   1578   (3)     1531   (3)
EX_FETCH                        00000A48-R   1563   (3)     1531   (3)
EX_FINISH                       00000A65-R   1582   (3)     1531   (3)
EX_HEXADECIMAL                  00000A1D-R   1540   (3)     1531   (3)
EX_LITERAL                      00000A48-R   1562   (3)     1531   (3)
EX_LOGICAL                      00000A3A-R   1554   (3)     1531   (3)
EX_NAME                         00000A55-R   1573   (3)     1531   (3)
EX_OCTAL                        00000A1D-R   1539   (3)     1531   (3)
EX_START                        00000A1B-R   1536   (3)     1531   (3)
EX_STORE                        00000A72-R   1589   (3)     1531   (3)
EX_STRING                       00000A28-R   1545   (3)     1531   (3)
FLG$V_BRIEF                     =00000000    347    (1)    #-1793   (4)    #-1825   (4)    #-1863   (4)    #-1877   (4)
                                                          #-2350   (6)    #-2362   (6)
FORMAT_PROMPT                   00000B7E-R   1743   (4)     1334   (3)     1402   (3)     1459   (3)     1704   (4)
                                                          546   (1)      593   (1)      637   (1)      663   (1)
                                                          755   (1)      764   (1)      772   (1)
FPT$_ARGS                       00000008     1741   (4)     1749   (4)
FPT$_FORMAT                     00000004     1741   (4)     1749   (4)
GENERIC                         FFFFFFEA     524    (1)     1023   (3)      733   (1)    #-873   (3)    #-903   (3)
                                                          906   (3)    #-927   (3)      930   (3)    #-933   (3)
GET_ATTRIBUTES                  00000490-R   870    (3)    #-780   (1)
GET_LINE                        00000B56-R   1694   (4)     1304   (3)     1373   (3)     1411   (3)      552   (1)
                                                          599   (1)      782   (1)
GET_LINK                        000001DC-R   597    (1)    #-589   (1)
GET_NAME                        000003CC-R   779    (1)    #-643   (1)    #-656   (1)
GET_TYPE                        0000014D-R   550    (1)    #-541   (1)
HP$A_DEVICE                     00000018                   #-2364   (6)    #-2744   (6)    #-2745   (6)    #-654   (1)
HP$A_DVA                        0000001C                   #-2744   (6)    #-2745   (6)    #-654   (1)
HP$A_LINK                       00000020                   #-1806   (4)    #-1926   (4)    #-1962   (4)    #-2035   (4)
                                                          #-2229   (4)     2365   (6)    #-2603   (6)    #-2730   (6)
                                                          #-2738   (6)    #-2746   (6)    #-653   (1)
HP$B_DRIVE                      0000000B                   #-804   (1)
```

```
HP$B_FLAGS                      0000000A                        #-1013  (3)     2049  (4)     2116  (4)     2139  (4)
                                                                  2141  (4)  #-2148  (4)     2285  (4)     2287  (4)
                                                                   742  (1)     808  (1)
HP$C_NAMELEN                    =00000014       259     (1)
HP$M_NAME                       =00000004                       #-1013  (3)
HP$M_WASALL                     =00000002                       #-2147  (4)
HP$Q_DEVICE                      00000000                       #-1811  (4)  #-2037  (4)     2142  (4)  #-2150  (4)
                                                                  2288  (4)     2369  (6)     2378  (6)  #-2468  (6)
                                                                #-2472  (6)  #-2606  (6)  #-2610  (6)  #-2727  (6)
                                                                 #-814  (1)  #-815   (1)  #-912   (3)  #-914   (3)
HP$T_DEVICE                      0000000C                        1127  (3)     812  (1)
HP$T_TYPE                        00000026                        2332  (6)     2368  (6)     2377  (6)  #-2740  (6)
                                                                #-2742  (6)     581  (1)
HP$V_ALLOC                      =00000000                       #-2138  (4)  #-2284  (4)
HP$V_NAME                       =00000002                        #-742  (1)  #-808   (1)
HP$V_WASALL                     =00000001                       #-2048  (4)  #-2115  (4)  #-2140  (4)  #-2286  (4)
HP$W_SIZE                        00000008                        #-579  (1)  #-789   (1)
HPE$C_EPB_SIZE                  =0000001A       258     (1)     #-569  (1)  #-792   (1)
HPE$T_DEVICE                     00000000                        809  (1)
HPE$W_EXT_DRIVE                  00000014                       #-805  (1)
IARG$_BASE                       00000008       1217     (3)  #-1237  (3)
IARG$_PC                         00000004       1217     (3)  #-1236  (3)
IARG$_PROMPT                     00000010       1217     (3)  1235  (3)
IARG$_TEXT                       0000000C       1217     (3)  1232  (3)
IF_ALPHA                         00000548-R      947     (3)  #-1050  (3)  #-1057  (3)
INI$PTABLE                       00001095-R     2413     (6)
INS$PTABLE                       00001240-R     2726     (6)   834  (2)
INTERPRET                        00000735-R     1240     (3)  #-1264  (3)  #-1272  (3)  #-1279  (3)  #-1286  (3)
                                                              #-1290  (3)  #-1299  (3)  #-1328  (3)  #-1345  (3)
                                                              #-1353  (3)  #-1356  (3)  #-1394  (3)  #-1437  (3)
KB_CHECK                         00000000-XR                    253  (1)     429  (1)
L$A_DEVP                         0000021C                      #-2662  (6)
L$L_HEADLENGTH                   00000200                      #-2660  (6)
LOC$ADAPTER                      00001142-R     2526     (6)  #-1799  (4)  #-1914  (4)  #-2016  (4)  #-2209  (4)
LOC$PTABLE                       00001197-R     2592     (6)  #-1933  (4)  #-2065  (4)  #-2246  (4)  #-2539  (6)
                                                               2731  (6)     648  (1)
LOC$PTDESC                       000011E6-R     2656     (6)  2334  (6)     560  (1)
LOCAL                            FFFFFF70       1223     (3)  1229  (3)
LOCAL_BLOCK                      FFFFFFE6        524     (1)   528  (1)
MAX_UNIT                         FFFFFFFC        524     (1)  #-798  (1)  #-877  (3)  #-887  (3)  #-888  (3)
                                                              #-896  (3)
NAME_FLAGS                       FFFFFFFF        524     (1)  #-708  (1)     796  (1)  #-872  (3)  #-880  (3)
                                                              #-894  (3)  #-901  (3)  #-910  (3)     922  (3)
                                                               926  (3)  #-937  (3)  #-988  (3)
OLDPC                            FFFFFFFC       1223     (3)  #-1241  (3)  #-1442  (3)
ONLY_ATTACH                      00000000-XR                    244  (1)     821  (1)
PARENT_CONTROLLER                FFFFFFFE        524     (1)  #-1061  (3)  #-729   (1)  #-874   (3)  #-924   (3)
                                                              #-934   (3)
PARENT_PTABLE                    FFFFFFF4        524     (1)  #-642   (1)  #-649   (1)  #-907   (3)
PARSE_DEVICE                     0000055D-R      986     (3)  #-786   (1)
PART_LIST                        00000388-R      324     (1)   325  (1)     326  (1)     327  (1)     328  (1)
                                                               329  (1)  #-706  (1)     707  (1)  #-715  (1)
                                                               716  (1)
PD$_ADD                         =0000008A       223     (1)
PD$_CASE                        =0000008C       223     (1)
PD$_COMPLEMENT                  =00000089       223     (1)
PD$_DECIMAL                     =00000082       223     (1)  #-1603  (4)
```

```
PD$_END                         =00000081      223     (1)
PD$_EPB                         =0000008E      223     (1)     #-884    (3)
PD$_FETCH                       =00000087      223     (1)
PD$_HEXADECIMAL                 =00000084      223     (1)
PD$_LITERAL                     =00000086      223     (1)
PD$_LOGICAL                     =0000008B      223     (1)
PD$_NAME                        =0000008D      223     (1)     #-899    (3)
PD$_OCTAL                       =00000083      223     (1)
PD$_START                       =00000080      223     (1)     #-1238   (3)    #-1257   (3)    #-1513   (3)    #-1531   (3)
                                                               #-2686   (6)    #-882    (3)
PD$_STORE                       =00000088      223     (1)
PD$_STRING                      =00000085      223     (1)
PD_ADD                           00000795-R    1292    (3)       1257   (3)
PD_CASE                          00000827-R    1338    (3)       1257   (3)
PD_COMPLEMENT                    00000790-R    1288    (3)       1257   (3)
PD_DECIMAL                       00000856-R    1357    (3)       1257   (3)
PD_EPB                           00000850-R    1354    (3)       1257   (3)
PD_FETCH                         0000076E-R    1274    (3)       1257   (3)
PD_FINISH                        00000765-R    1266    (3)       1257   (3)
PD_HEXADECIMAL                   000008B7-R    1365    (3)       1257   (3)
PD_LITERAL                       00000769-R    1270    (3)       1257   (3)
PD_LOGICAL                       000007AF-R    1301    (3)       1257   (3)
PD_NAME                          00000844-R    1350    (3)       1257   (3)
PD_NUMERIC                       000008EB-R    1369    (3)     #-1359   (3)    #-1363   (3)    #-1367   (3)
PD_OCTAL                         00000887-R    1361    (3)       1257   (3)
PD_START                         00000760-R    1262    (3)       1257   (3)
PD_STORE                         0000077F-R    1281    (3)       1257   (3)
PD_STRING                        00000953-R    1408    (3)       1257   (3)
PT$EXTRACT                       000009E2-R    1508    (3)       2345   (6)
PT$INTERPRET                     0000070C-R    1227    (3)       829    (1)
PTD$M_CONTROLLER                =00000002      222     (1)     #-1118   (3)      222    (1)
PTD$M_DEVICE                    =00000003      222     (1)
PTD$M_ENDDEVICE                 =0000001B      222     (1)
PTD$M_INHERIT                   =00000018      222     (1)       222    (1)    #-909    (3)
PTD$M_INHERITED                 =00000008      222     (1)
PTD$M_INHERIT_CON               =00000010      222     (1)       222    (1)
PTD$M_INHERIT_PRE               =00000008      222     (1)       222    (1)
PTD$M_NAME                      =00000004      222     (1)     #-936    (3)
PTD$M_UNIT                      =00000001      222     (1)     #-1118   (3)      222    (1)    #-879    (3)    #-893    (3)
                                                               #-935    (3)
PTD$V_CONTROLLER                =00000001      222     (1)     #-1048   (3)    #-723    (1)
PTD$V_INHERIT_CON               =00000004      222     (1)     #-727    (1)    #-921    (3)
PTD$V_INHERIT_PRE               =00000003      222     (1)     #-925    (3)
PTD$V_NAME                      =00000002      222     (1)     #-710    (1)    #-992    (3)
PTD$V_UNIT                      =00000000      222     (1)     #-1070   (3)    #-1074   (3)    #-720    (1)    #-795    (1)
PTDESC                           FFFFFFF8      524     (1)     #-565    (1)    #-875    (3)
PT_LOCATE                        00001211-R    2675    (6)     #-2664   (6)    #-2668   (6)
Q_ADAPTER                        00000004-R    356     (1)     #-1936   (4)    #-2070   (4)    #-2251   (4)    #-2456   (6)
                                                               #-2530   (6)    #-2544   (6)
Q_HUB                            0000037D-R    321     (1)       2367   (6)
RANGE                            00000003-R    354     (1)     #-1094   (3)    #-748    (1)    #-991    (3)
SAVABS...                       =0000000C      1741    (4)
SCAN$ALPHA                       00000000-XR                     1146   (3)      1306   (3)      254    (1)      916    (3)
SCAN$ALPHANUM                    00000000-XR                     1155   (3)      248    (1)
SCAN$DECIMAL                     00000000-XR                     1029   (3)      1068   (3)      254    (1)
SCAN$DEVICE                      00000000-XR                     248    (1)      603    (1)
SCAN$NUMERIC                     00000000-XR                     1380   (3)      248    (1)
```

```
SCAN$SPACES                 00000000-XR                 1696   (4)     249    (1)
SCAN$SYMBOL                 00000000-XR                 249    (1)     555    (1)
SCAN_ALPHA                  000006F4-R   1144   (3)   #-1103   (3)
SCAN_ALPHANUM               00000700-R   1153   (3)   #-1113   (3)
SCRIPT$FLUSH                00000000-XR                 1912   (4)     2010   (4)     2203   (4)     245    (1)
                                                        394    (1)
SET_ABORT                   000000FD-R   457    (1)     393    (1)
SHOWBUF_SIZ                =00000100     2325   (6)     2328   (6)   #-2330   (6)   #-2375   (6)
SHOW_DEV                    00000FDA-R   2326   (6)     1820   (4)     1873   (4)
SIZ...                     =00000001     236    (1)     222    (1)     233    (1)     236    (1)
SKIP                        00000A41-R   1557   (3)   #-1542   (3)   #-1547   (3)
SS$_BADPARAM               =00000014                  #-1259   (3)   #-1335   (3)   #-1403   (3)   #-1460   (3)
                                                      #-1533   (3)
SS$_DEVALLOC               =00000840                  #-2155   (4)
SS$_DEVALRALLOC            =00000641                  #-2143   (4)
SS$_DEVMOUNT               =0000006C                  #-2158   (4)
SS$_INSFARG                =00000114                  #-1705   (4)
SS$_NORMAL                 =00000001                  #-1267   (3)   #-1585   (3)   #-1699   (4)   #-2129   (4)
                                                      #-2420   (6)   #-2764   (6)
SS$_NOSUCHDEV              =00000908                  #-2152   (4)
ST_DECIMAL                  00000AA4-R   1605   (4)     1603   (4)
ST_HEXADECIMAL              00000AC6-R   1613   (4)     1603   (4)
ST_LOGICAL                  00000B02-R   1630   (4)     1603   (4)
ST_OCTAL                    00000AB4-R   1609   (4)     1603   (4)
ST_STRING                   00000AD7-R   1617   (4)     1603   (4)
SYS$ALLOC                   00000000-XR                 2142   (4)     250    (1)
SYS$DALLOC                  00000000-XR                 2288   (4)     250    (1)
SYS$FAO                     00000000-XR                 1650   (4)     250    (1)
SYS$FAOL                    00000000-XR                 1749   (4)
TEMP_STORE                  0000000C-R   358    (1)
T_ALLOC                     000000A0-R   272    (1)     760    (1)
T_AMBIG_ADAP                000001C3-R   295    (1)     2486   (6)
T_AMBIG_DEV                 000001F0-R   297    (1)     2488   (6)
T_BAD_NAME                  0000002E-R   268    (1)     754    (1)     763    (1)
T_BAD_RANGE                 0000006B-R   270    (1)     771    (1)
T_DEATTACH                  00000227-R   299    (1)     1969   (4)
T_DEVALLOC                  00000243-R   301    (1)     2154   (4)
T_DEV_MOUNTED               00000264-R   303    (1)     2157   (4)
T_INVALID                   00000007-R   266    (1)     545    (1)     592    (1)
T_LINK                      00000158-R   287    (1)     591    (1)     598    (1)     634    (1)
T_MAIN_BUS                  00000109-R   282    (1)     636    (1)
T_NAME                      0000014C-R   286    (1)     661    (1)     750    (1)     759    (1)     768    (1)
                                                        781    (1)
T_NO_SUCH_ADAPTER           00000186-R   291    (1)     2546   (6)
T_NO_SUCH_DEV               00000170-R   289    (1)     2081   (4)     2151   (4)     2262   (4)
T_NO_SUCH_ON_ADAPTER        0000019D-R   293    (1)     1940   (4)     2074   (4)     2255   (4)
T_PART_OF                   000000CD-R   278    (1)     747    (1)
T_PNF                       00000306-R   313    (1)     418    (1)
T_PROMPT                    0000013F-R   284    (1)     1703   (4)
T_RANGE                     000000A8-R   274    (1)     769    (1)
T_SHOULDBE                  00000348-R   315    (1)     1332   (3)     1399   (3)     1457   (3)
T_SHOWDEV                   000002DD-R   309    (1)     2354   (6)
T_SHOWDEVB                  000002F9-R   311    (1)     2360   (6)
T_SHOWDEVB_ALLOC            000002C6-R   307    (1)     2358   (6)
T_SHOWDEV_ALLOC             000002A0-R   305    (1)     2352   (6)
T_SKIP                      00000333-R   314    (1)     422    (1)
T_STRING                    00000360-R   319    (1)     1333   (3)     1458   (3)
```

```
T_TYPE                          00000164-R   288    (1)      544    (1)      551    (1)
T_UNIT_EXCESS                   000000D7-R   280    (1)      662    (1)
T_UNRANGE                       000000BA-R   276    (1)      770    (1)
T_YESNO                         00000352-R   317    (1)     1331    (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION		REFERENCES...							
$ALLOC_S	1	2142	(4)	2142	(4)						
$D1_PRINT_S	1	1969	(4)	1969	(4)						
$DALLOC_S	1	2288	(4)	2288	(4)						
$DEF	1	236	(1)								
$DEFINI	1	224	(1)	224	(1)	225	(1)	226	(1)	227	(1)
				229	(1)	230	(1)	231	(1)		
$DS_CNTRLC_S	1	393	(1)	393	(1)	445	(1)				
$DS_DSADEF	1	227	(1)	227	(1)	234	(1)				
$DS_DSDEF	3	228	(1)	228	(1)						
$DS_HDRDEF	2	224	(1)	224	(1)						
$DS_HPEODEF	1	226	(1)	226	(1)						
$DS_HPODEF	2	225	(1)	225	(1)						
$DS_PDDEF	1	223	(1)	223	(1)						
$DS_PTDDEF	1	222	(1)	222	(1)						
$DS_TYPEDEF	4	235	(1)	235	(1)						
$EQU	1	236	(1)	222	(1)	223	(1)	228	(1)	235	(1)
$EQULS1	1	235	(1)	223	(1)	228	(1)	235	(1)		
$EQULST	1	223	(1)	223	(1)	228	(1)	235	(1)		
$FAOL_S	1	1748	(4)	1748	(4)						
$GBLINI	2	222	(1)	222	(1)	223	(1)	228	(1)	233	(1)
				235	(1)	236	(1)				
$IPLDEF	1	231	(1)	231	(1)						
$OFFSET	2	512	(1)	1213	(3)	1219	(3)	1503	(3)	1739	(4)
				512	(1)	516	(1)				
$OFFST1	1	514	(1)	1217	(3)	1223	(3)	1506	(3)	1741	(4)
				514	(1)	524	(1)				
$PRDEF	5	229	(1)	229	(1)						
$PRINT	2	1966	(4)	1966	(4)						
$PUSHADR	1	393	(1)	1749	(4)	2142	(4)	2288	(4)	2758	(6)
				393	(1)	445	(1)	449	(1)		
$PUSHTWO	1	2142	(4)	2142	(4)						
$SSDEF	21	230	(1)	230	(1)						
$VIELD	1	222	(1)	222	(1)	233	(1)	236	(1)		
$VIELD1	1	236	(1)	222	(1)	233	(1)	236	(1)		
BR_IF_INHIBIT_ALLOCATE	1	2132	(4)	2132	(4)						
CASE	1	1242	(3)	1242	(3)	1516	(3)	1598	(4)		
CLIDEF	4	232	(1)	232	(1)						
CRDDEF	1	236	(1)	236	(1)						
DSFDEF	3	233	(1)	233	(1)						
ERRSUP_S	1	449	(1)	2758	(6)	449	(1)				
MODNAM	1	265	(1)	265	(1)						

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	1297	(3)	1377	(3)	1391	(3)	1418	(3)	1434	(3)	1444	(3)	1543	(3)
		1550	(3)	1559	(3)	1620	(4)	1654	(4)	2160	(4)	2375	(6)	2670	(6)
		567	(1)	820	(1)	832	(2)								
ADDL2	00C0	1326	(3)	1355	(3)	1579	(3)	440	(1)	442	(1)	569	(1)	623	(1)
		741	(1)	745	(1)										
ADDL3	00C1	1442	(3)	1747	(4)	2472	(6)	2610	(6)	790	(1)	814	(1)	912	(3)
ADDW	00A0	1327	(3)												
ADDW2	00A0	1393	(3)	1431	(3)	1697	(4)	563	(1)	641	(1)	652	(1)	807	(1)
		830	(1)												
ADDW3	00A1	1750	(4)												
ASHL	0078	576	(1)												
BBC	00E1	1074	(3)	1870	(4)	2048	(4)	2115	(4)	2130	(4)	2138	(4)	2281	(4)
		2284	(4)	2351	(6)	2357	(6)	2453	(6)	2528	(6)	415	(1)	710	(1)
		720	(1)	723	(1)	727	(1)	742	(1)	795	(1)	808	(1)	921	(3)
		925	(3)	992	(3)										
BBCC	00E5	1972	(4)	2046	(4)	2231	(4)	2232	(4)	2269	(4)	2270	(4)	2286	(4)
		2763	(6)												
BBS	00E0	1048	(3)	1070	(3)	2043	(4)	2132	(4)	2135	(4)	2350	(6)	2362	(6)
		2459	(6)	413	(1)										
BBSS	00E2	2140	(4)												
BEQL	0013	1007	(3)	1063	(3)	1081	(3)	1084	(3)	1119	(3)	1131	(3)	1307	(3)
		1313	(3)	1340	(3)	1343	(3)	1382	(3)	1420	(3)	1447	(3)	1622	(4)
		1803	(4)	1809	(4)	1872	(4)	1925	(4)	1961	(4)	2020	(4)	2032	(4)
		2102	(4)	2153	(4)	2156	(4)	2159	(4)	2213	(4)	2225	(4)	2467	(6)
		2600	(6)	2663	(6)	2677	(6)	2732	(6)	2737	(6)	2743	(6)	2756	(6)
		557	(1)	561	(1)	605	(1)	622	(1)	632	(1)	735	(1)	843	(3)
		908	(3)	911	(3)	917	(3)								
BEQLU	0013	878	(3)	892	(3)										
BGTR	0014	1034	(3)	1319	(3)	1815	(4)	630	(1)	803	(1)	954	(3)		
BGTRU	001A	1114	(3)	1389	(3)										
BICB	008A	2147	(4)												
BICB2	008A	1825	(4)	1877	(4)	935	(3)								
BICL	00CA	2744	(6)												
BISB2	0088	1013	(3)	1793	(4)	1863	(4)								
BISL	00C8	2745	(6)												
BITB	0093	909	(3)												
BITL	00D3	1118	(3)												
BLBC	00E9	1051	(3)	1052	(3)	1058	(3)	1305	(3)	1375	(3)	1800	(4)	1915	(4)
		2012	(4)	2095	(4)	2205	(4)	439	(1)	553	(1)	833	(2)	835	(2)
		931	(3)												
BLBS	00E8	1412	(3)	1633	(4)	2017	(4)	2041	(4)	2045	(4)	2146	(4)	2210	(4)
		2487	(6)	412	(1)	430	(1)	571	(1)	600	(1)	748	(1)	749	(1)
		783	(1)												
BLEQ	0015	1040	(3)	1078	(3)	1116	(3)	1121	(3)	704	(1)	950	(3)		
BLEQU	001B	800	(1)												
BLSS	0019	1036	(3)	1104	(3)	1424	(3)	1805	(4)	2034	(4)	2063	(4)	2069	(4)
		2227	(4)	2244	(4)	2250	(4)	2602	(6)	533	(1)	952	(3)		
BLSSU	001F	1387	(3)	920	(3)										
BNEQ	0012	1000	(3)	1018	(3)	1025	(3)	1030	(3)	1042	(3)	1065	(3)	1069	(3)
		1086	(3)	1123	(3)	1239	(3)	1322	(3)	1385	(3)	1428	(3)	1436	(3)

```
                    1514  (3)   1552  (3)   1595  (3)   1701  (4)   1807  (4)   1817  (4)   1917  (4)
                    1922  (4)   1927  (4)   1934  (4)   1963  (4)   2024  (4)   2036  (4)   2066  (4)
                    2144  (4)   2217  (4)   2230  (4)   2247  (4)   2335  (6)   2366  (6)   2470  (6)
                    2474  (6)   2477  (6)   2532  (6)   2534  (6)   2542  (6)   2604  (6)   2608  (6)
                    2612  (6)   2661  (6)   2665  (6)   2682  (6)   2684  (6)   2687  (6)   2739  (6)
                    2741  (6)   535   (1)   626   (1)   650   (1)   731   (1)   787   (1)   883   (3)
                    885   (3)   900   (3)   997   (3)
BRB         0011    1011  (3)   1045  (3)   1055  (3)   1072  (3)   1264  (3)   1272  (3)   1279  (3)
                    1286  (3)   1290  (3)   1299  (3)   1348  (3)   1426  (3)   1451  (3)   1537  (3)
                    1544  (3)   1553  (3)   1567  (3)   1571  (3)   1576  (3)   1580  (3)   1944  (4)
                    2051  (4)   2118  (4)   2235  (4)   2259  (4)   2266  (4)   2339  (6)   2353  (6)
                    2355  (6)   2359  (6)   2383  (6)   2457  (6)   2493  (6)   2536  (6)   2616  (6)
                    2751  (6)   450   (1)   541   (1)   589   (1)   624   (1)   717   (1)   739   (1)
                    810   (1)
BRW         0031    1004  (3)   1019  (3)   1026  (3)   1032  (3)   1037  (3)   1038  (3)   1046  (3)
                    1108  (3)   1136  (3)   1138  (3)   1328  (3)   1345  (3)   1353  (3)   1356  (3)
                    1394  (3)   1437  (3)   1596  (3)   1604  (4)   1656  (4)   1918  (4)   1931  (4)
                    2018  (4)   2054  (4)   2078  (4)   2085  (4)   2145  (4)   2211  (4)   426   (1)
                    427   (1)   441   (1)   548   (1)   573   (1)   595   (1)   601   (1)   639   (1)
                    643   (1)   651   (1)   656   (1)   665   (1)   758   (1)   767   (1)   775   (1)
                    784   (1)   788   (1)   801   (1)   993   (3)
BSBB        0010    1113  (3)   1363  (3)   1367  (3)   1542  (3)   1547  (3)   1611  (4)   1615  (4)
                    1625  (4)   1627  (4)   1636  (4)   1929  (4)   1945  (4)   1964  (4)   2094  (4)
                    2233  (4)   2271  (4)   2539  (6)   2664  (6)   2668  (6)
BSBW        0030    1050  (3)   1057  (3)   1103  (3)   1359  (3)   1607  (4)   1799  (4)   1914  (4)
                    1933  (4)   2011  (4)   2016  (4)   2044  (4)   2065  (4)   2204  (4)   2209  (4)
                    2246  (4)   780   (1)   786   (1)
CALLG       00FA    834   (2)
CALLS       00FB    1334  (3)   1402  (3)   1459  (3)   1650  (4)   1704  (4)   1749  (4)   1943  (4)
                    1969  (4)   2077  (4)   2084  (4)   2142  (4)   2167  (4)   2258  (4)   2265  (4)
                    2288  (4)   2345  (6)   2373  (6)   2382  (6)   2491  (6)   2549  (6)   2758  (6)
                    393   (1)   411   (1)   421   (1)   425   (1)   438   (1)   445   (1)   449   (1)
                    546   (1)   593   (1)   637   (1)   663   (1)   755   (1)   764   (1)   772   (1)
                    829   (1)
CASEB       008F    1257  (3)   1531  (3)   1603  (4)
CLRB        0094    1009  (3)   392   (1)   872   (3)   873   (3)   874   (3)   933   (3)   934   (3)
CLRL        00D4    1003  (3)   1022  (3)   1090  (3)   1309  (3)   1433  (3)   1512  (3)   1548  (3)
                    1655  (4)   1971  (4)   2097  (4)   2103  (4)   2416  (6)   2452  (6)   2463  (6)
                    2492  (6)   2535  (6)   2543  (6)   2550  (6)   2659  (6)   2692  (6)   540   (1)
                    615   (1)   642   (1)   719   (1)   794   (1)   948   (3)
CLRQ        007C    1099  (3)   2142  (4)   2618  (6)   577   (1)   585   (1)
CLRW        00B4    530   (1)
CMPB        0091    1006  (3)   1041  (3)   1080  (3)   1083  (3)   1085  (3)   1122  (3)   1130  (3)
                    1238  (3)   1312  (3)   1321  (3)   1427  (3)   1513  (3)   1816  (4)   2473  (6)
                    2611  (6)   2681  (6)   2683  (6)   2686  (6)   532   (1)   625   (1)   629   (1)
                    631   (1)   882   (3)   884   (3)   899   (3)   951   (3)   953   (3)   996   (3)
                    999   (3)
CMPL        00D1    1033  (3)   1035  (3)   1064  (3)   1147  (3)   1156  (3)   1318  (3)   1342  (3)
                    1386  (3)   1388  (3)   1806  (4)   1814  (4)   1921  (4)   1926  (4)   1962  (4)
                    2023  (4)   2035  (4)   2216  (4)   2229  (4)   2469  (6)   2603  (6)   2607  (6)
                    2660  (6)   2738  (6)   2742  (6)   703   (1)   799   (1)   919   (3)
CMPW        00B1    2143  (4)   2152  (4)   2155  (4)   2158  (4)   2531  (6)   2740  (6)   621   (1)
                    802   (1)   888   (3)
CMPZV       00ED    2533  (6)
CVTBL       0098    1137  (3)   1969  (4)
DECL        00D7    1028  (3)   1043  (3)   1053  (3)   1059  (3)   1079  (3)   1105  (3)   1125  (3)
                    1453  (3)   1812  (4)   2038  (4)   2109  (4)   2729  (6)   725   (1)
```

DECW	00B7	896	(3)												
EXTV	00EE	1296	(3)												
EXTZV	00EF	1278	(3)	1593	(3)	2105	(4)	2107	(4)	797	(1)				
INCB	0096	886	(3)												
INCL	00D6	1027	(3)	1047	(3)	1054	(3)	1067	(3)	1082	(3)	1106	(3)	1124	(3)
		1315	(3)	1421	(3)	1813	(4)	2039	(4)	2455	(6)	2728	(6)	619	(1)
		627	(1)	722	(1)	726	(1)	737	(1)	955	(3)				
INCW	00B6	436	(1)												
INSV	00F0	1285	(3)	1298	(3)	2042	(4)	2050	(4)	2052	(4)	2106	(4)	2108	(4)
		2113	(4)	2117	(4)	2119	(4)	2417	(6)	2418	(6)				
JSB	0016	1029	(3)	1068	(3)	1146	(3)	1155	(3)	1304	(3)	1306	(3)	1373	(3)
		1380	(3)	1411	(3)	1696	(4)	1820	(4)	1873	(4)	1912	(4)	1974	(4)
		2010	(4)	2040	(4)	2162	(4)	2203	(4)	2334	(6)	2731	(6)	2750	(6)
		394	(1)	429	(1)	552	(1)	555	(1)	560	(1)	570	(1)	599	(1)
		603	(1)	648	(1)	782	(1)	821	(1)	844	(3)	916	(3)		
MCOML	00D2	1289	(3)												
MNEGL	00CE	1071	(3)	1263	(3)	1415	(3)	2061	(4)	2242	(4)	2527	(6)	2538	(6)
		647	(1)												
MOVAB	009E	1023	(3)	1127	(3)	1229	(3)	1230	(3)	1352	(3)	1440	(3)	1541	(3)
		1546	(3)	1555	(3)	1575	(3)	1624	(4)	1632	(4)	1634	(4)	2154	(4)
		2157	(4)	2328	(6)	2332	(6)	2352	(6)	2354	(6)	2358	(6)	2360	(6)
		2488	(6)	399	(1)	401	(1)	407	(1)	434	(1)	528	(1)	551	(1)
		581	(1)	598	(1)	616	(1)	707	(1)	709	(1)	716	(1)	733	(1)
		747	(1)	781	(1)	812	(1)	815	(1)	881	(3)				
MOVAL	00DE	1798	(4)	1867	(4)	1913	(4)	2013	(4)	2014	(4)	2015	(4)	2206	(4)
		2207	(4)	2208	(4)	2414	(6)	2464	(6)	2597	(6)	2667	(6)	2734	(6)
		2753	(6)	397	(1)	538	(1)	809	(1)						
MOVAQ	007E	1232	(3)	1235	(3)	1347	(3)	1570	(3)	1745	(4)	2367	(6)	402	(1)
		529	(1)	534	(1)										
MOVB	0090	1094	(3)	1095	(3)	1128	(3)	1132	(3)	1311	(3)	1448	(3)	1450	(3)
		1751	(4)	1920	(4)	2022	(4)	2215	(4)	435	(1)	458	(1)	582	(1)
		587	(1)	706	(1)	714	(1)	721	(1)	729	(1)	732	(1)	736	(1)
		743	(1)	746	(1)	804	(1)	816	(1)	818	(1)	879	(3)	893	(3)
		901	(3)	903	(3)	923	(3)	927	(3)	990	(3)	991	(3)	998	(3)
MOVC3	0028	404	(1)												
MOVC5	002C	904	(3)	928	(3)										
MOVL	00D0	1002	(3)	1010	(3)	1014	(3)	1015	(3)	1016	(3)	1021	(3)	1031	(3)
		1102	(3)	1107	(3)	1110	(3)	1111	(3)	1236	(3)	1237	(3)	1241	(3)
		1259	(3)	1267	(3)	1271	(3)	1302	(3)	1303	(3)	1308	(3)	1314	(3)
		1320	(3)	1324	(3)	1329	(3)	1335	(3)	1346	(3)	1358	(3)	1362	(3)
		1366	(3)	1371	(3)	1372	(3)	1378	(3)	1390	(3)	1396	(3)	1403	(3)
		1409	(3)	1410	(3)	1425	(3)	1460	(3)	1533	(3)	1585	(3)	1606	(4)
		1610	(4)	1614	(4)	1618	(4)	1631	(4)	1635	(4)	1699	(4)	1801	(4)
		1802	(4)	1808	(4)	1810	(4)	1868	(4)	1871	(4)	1923	(4)	1924	(4)
		1928	(4)	1932	(4)	1935	(4)	1959	(4)	1960	(4)	1973	(4)	2030	(4)
		2031	(4)	2047	(4)	2064	(4)	2067	(4)	2093	(4)	2098	(4)	2099	(4)
		2101	(4)	2104	(4)	2114	(4)	2129	(4)	2134	(4)	2137	(4)	2223	(4)
		2228	(4)	2245	(4)	2248	(4)	2268	(4)	2283	(4)	2337	(6)	2338	(6)
		2415	(6)	2420	(6)	2458	(6)	2465	(6)	2466	(6)	2471	(6)	2478	(6)
		2481	(6)	2541	(6)	2552	(6)	2595	(6)	2598	(6)	2599	(6)	2609	(6)
		2614	(6)	2662	(6)	2676	(6)	2678	(6)	2688	(6)	2730	(6)	2735	(6)
		2736	(6)	2746	(6)	2748	(6)	2749	(6)	2754	(6)	2759	(6)	2762	(6)
		2764	(6)	443	(1)	459	(1)	547	(1)	564	(1)	565	(1)	575	(1)
		583	(1)	594	(1)	620	(1)	638	(1)	649	(1)	653	(1)	654	(1)
		664	(1)	705	(1)	713	(1)	744	(1)	756	(1)	757	(1)	765	(1)
		766	(1)	773	(1)	774	(1)	793	(1)	842	(3)	875	(3)	907	(3)
		989	(3)												

MOVQ	007D	1088	(3)	1233	(3)	1341	(3)	1398	(3)	1432	(3)	1510	(3)	1511	(3)
		1583	(3)	1795	(4)	1811	(4)	1916	(4)	1936	(4)	1938	(4)	1970	(4)
		2019	(4)	2037	(4)	2070	(4)	2072	(4)	2079	(4)	2150	(4)	2212	(4)
		2251	(4)	2253	(4)	2260	(4)	2456	(6)	2461	(6)	2530	(6)	2544	(6)
		2615	(6)	2679	(6)	2727	(6)	395	(1)	444	(1)	531	(1)	559	(1)
		635	(1)	824	(1)										
MOVW	00B0	1075	(3)	1310	(3)	579	(1)	740	(1)	805	(1)	887	(3)		
MOVZBL	009A	1024	(3)	1060	(3)	1061	(3)	1096	(3)	1276	(3)	1277	(3)	1283	(3)
		1284	(3)	1294	(3)	1295	(3)	1317	(3)	1325	(3)	1339	(3)	1351	(3)
		1376	(3)	1416	(3)	1419	(3)	1435	(3)	1443	(3)	1446	(3)	1551	(3)
		1558	(3)	1569	(3)	1574	(3)	1591	(3)	1592	(3)	1619	(4)	1621	(4)
		1643	(4)	1969	(4)	2333	(6)	2343	(6)	566	(1)	568	(1)	617	(1)
		618	(1)	708	(1)	715	(1)	734	(1)	806	(1)	876	(3)	902	(3)
		988	(3)												
MOVZBW	009B	877	(3)												
MOVZWL	003C	1234	(3)	1275	(3)	1282	(3)	1293	(3)	1590	(3)	1705	(4)	1937	(4)
		1939	(4)	2071	(4)	2073	(4)	2080	(4)	2252	(4)	2254	(4)	2261	(4)
		2330	(6)	2462	(6)	2545	(6)	2594	(6)	2680	(6)	408	(1)	556	(1)
		584	(1)	604	(1)	613	(1)	789	(1)	798	(1)	841	(3)	915	(3)
POPL	8ED0	1374	(3)	1698	(4)	2112	(4)	2161	(4)	2168	(4)	846	(3)		
POPR	00BA	1089	(3)	1098	(3)	1135	(3)	1381	(3)	1642	(4)	1652	(4)	1824	(4)
		1876	(4)	1946	(4)	1975	(4)	2120	(4)	2272	(4)	2482	(6)	2540	(6)
		2619	(6)	2671	(6)	406	(1)	633	(1)	640	(1)	645	(1)	938	(3)
PUSHAB	009F	1330	(3)	1332	(3)	1397	(3)	1399	(3)	1400	(3)	1455	(3)	1457	(3)
		1556	(3)	1702	(4)	1940	(4)	1969	(4)	2074	(4)	2081	(4)	2151	(4)
		2255	(4)	2262	(4)	2329	(6)	2344	(6)	2368	(6)	2377	(6)	2486	(6)
		2546	(6)	417	(1)	418	(1)	422	(1)	431	(1)	437	(1)	544	(1)
		591	(1)	634	(1)	661	(1)	750	(1)	751	(1)	752	(1)	753	(1)
		759	(1)	760	(1)	761	(1)	762	(1)	768	(1)	769	(1)	770	(1)
PUSHAL	00DF	1648	(4)	1749	(4)	2758	(6)	393	(1)	433	(1)	449	(1)		
PUSHAQ	007F	1331	(3)	1333	(3)	1401	(3)	1456	(3)	1458	(3)	1647	(4)	1649	(4)
		1703	(4)	1749	(4)	2142	(4)	2288	(4)	2341	(6)	2363	(6)	2365	(6)
		2369	(6)	2378	(6)	2485	(6)	409	(1)	410	(1)	545	(1)	592	(1)
		636	(1)	662	(1)	754	(1)	763	(1)	771	(1)	825	(1)	826	(1)
PUSHAW	003F	1749	(4)	536	(1)										
PUSHL	00DD	1316	(3)	1370	(3)	1695	(4)	1941	(4)	1942	(4)	1969	(4)	2075	(4)
		2076	(4)	2082	(4)	2083	(4)	2096	(4)	2142	(4)	2149	(4)	2165	(4)
		2166	(4)	2256	(4)	2257	(4)	2263	(4)	2264	(4)	2288	(4)	2342	(6)
		2364	(6)	2370	(6)	2371	(6)	2372	(6)	2379	(6)	2380	(6)	2381	(6)
		2489	(6)	2490	(6)	2547	(6)	2548	(6)	2758	(6)	393	(1)	396	(1)
		419	(1)	420	(1)	423	(1)	424	(1)	445	(1)	449	(1)	537	(1)
		827	(1)	828	(1)	839	(2)								
PUSHR	00BB	1126	(3)	1145	(3)	1154	(3)	1379	(3)	1644	(4)	1646	(4)	1796	(4)
		1865	(4)	1911	(4)	1955	(4)	2009	(4)	2202	(4)	2451	(6)	2537	(6)
		2593	(6)	2657	(6)	403	(1)	614	(1)	871	(3)	987	(3)		
RET	0004	1260	(3)	1268	(3)	1336	(3)	1405	(3)	1413	(3)	1461	(3)	1534	(3)
		1586	(3)	1752	(4)	2760	(6)	2765	(6)	460	(1)	836	(2)	847	(3)
ROTL	009C	1919	(4)	2021	(4)	2214	(4)								
RSB	0005	1092	(3)	1100	(3)	1148	(3)	1157	(3)	1560	(3)	1706	(4)	1826	(4)
		1878	(4)	1947	(4)	1976	(4)	2121	(4)	2163	(4)	2169	(4)	2273	(4)
		2289	(4)	2376	(6)	2421	(6)	2483	(6)	2551	(6)	2553	(6)	2621	(6)
		2673	(6)	2689	(6)	2693	(6)	446	(1)	939	(3)	956	(3)		
SOBGEQ	00F4	1429	(3)	1623	(4)	2480	(6)								
SOBGTR	00F5	1008	(3)	1044	(3)	1133	(3)	1323	(3)	1344	(3)	1449	(3)	1818	(4)
		1822	(4)	1875	(4)	1930	(4)	1965	(4)	2053	(4)	2110	(4)	2234	(4)
		2419	(6)	2475	(6)	2479	(6)	2613	(6)	2617	(6)	2685	(6)	2691	(6)
		2747	(6)	2757	(6)	578	(1)	588	(1)	628	(1)	644	(1)	738	(1)

		819	(1)												
SUBL	00C2	1653	(4)												
SUBL2	00C2	398	(1)	400	(1)										
SUBL3	00C3	1383	(3)	1423	(3)	1430	(3)	1454	(3)	1584	(3)	2468	(6)	2606	(6)
		432	(1)	791	(1)	792	(1)	918	(3)	995	(3)				
SUBW3	00A3	1076	(3)	1746	(4)	914	(3)								
TSTB	0095	1017	(3)												
TSTL	00D5	1039	(3)	1077	(3)	1120	(3)	1384	(3)	1392	(3)	1565	(3)	1594	(3)
		1700	(4)	1804	(4)	2033	(4)	2062	(4)	2068	(4)	2224	(4)	2226	(4)
		2243	(4)	2249	(4)	2476	(6)	2601	(6)	2620	(6)	2672	(6)	2755	(6)
		949	(3)												
TSTW	00B5	1091	(3)	1115	(3)										

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	1608	(4)	1612	(4)	1616	(4)	1626	(4)	1628	(4)	1637	(4)	1638	(4)	1639	(4)
	265	(1)	273	(1)	275	(1)	277	(1)	279	(1)	286	(1)	287	(1)	288	(1)
	290	(1)	292	(1)	294	(1)	296	(1)	298	(1)	300	(1)	302	(1)	304	(1)
	306	(1)	308	(1)	310	(1)	312	(1)	313	(1)	314	(1)	316	(1)	330	(1)
	331	(1)	332	(1)	333	(1)	337	(1)	338	(1)	339	(1)	340	(1)	341	(1)
.ASCID	1360	(3)	1364	(3)	1368	(3)	267	(1)	269	(1)	271	(1)	281	(1)	283	(1)
	285	(1)	318	(1)	320	(1)	322	(1)								
.BLKB	1217	(3)	1223	(3)	1506	(3)	1741	(4)	349	(1)	351	(1)	514	(1)	524	(1)
.BLKL	232	(1)	359	(1)												
.BLKQ	357	(1)														
.BYTE	325	(1)	326	(1)	327	(1)	328	(1)	329	(1)	336	(1)	353	(1)	355	(1)
.DSABL	15	(1)														
.END	2767	(6)														
.ENDC	1217	(3)	1223	(3)	1506	(3)	1741	(4)	1749	(4)	1969	(4)	2142	(4)	222	(1)
	223	(1)	228	(1)	2288	(4)	233	(1)	235	(1)	236	(1)	2758	(6)	393	(1)
	445	(1)	449	(1)	514	(1)	524	(1)								
.ENDM	1242	(3)	1748	(4)	1966	(4)	1969	(4)	2132	(4)	2142	(4)	222	(1)	223	(1)
	224	(1)	225	(1)	226	(1)	227	(1)	228	(1)	2288	(4)	229	(1)	230	(1)
	231	(1)	232	(1)	233	(1)	235	(1)	236	(1)	265	(1)	393	(1)	449	(1)
	512	(1)	514	(1)												
.ENDR	1217	(3)	1223	(3)	1257	(3)	1506	(3)	1531	(3)	1603	(4)	1741	(4)	1969	(4)
	222	(1)	223	(1)	228	(1)	233	(1)	235	(1)	236	(1)	514	(1)	524	(1)
.ENTRY	1227	(3)	1508	(3)	2726	(6)	457	(1)	526	(1)						
.EXTRN	244	(1)	245	(1)	246	(1)	247	(1)	248	(1)	249	(1)	250	(1)	251	(1)
	252	(1)	253	(1)	254	(1)	255	(1)	256	(1)						
.GLOBL	1749	(4)	2142	(4)	2288	(4)										
.IDENT	14	(1)														
.IF	1217	(3)	1223	(3)	1506	(3)	1741	(4)	1749	(4)	1969	(4)	2142	(4)	222	(1)
	223	(1)	228	(1)	2288	(4)	233	(1)	235	(1)	236	(1)	2758	(6)	393	(1)
	445	(1)	449	(1)	514	(1)	524	(1)								
.IFF	1217	(3)	1223	(3)	1506	(3)	1741	(4)	1749	(4)	2142	(4)	222	(1)	223	(1)
	228	(1)	2288	(4)	233	(1)	235	(1)	236	(1)	2758	(6)	393	(1)	445	(1)
	449	(1)	514	(1)	524	(1)										
.IIF	222	(1)	223	(1)	228	(1)	233	(1)	235	(1)	236	(1)	2758	(6)	449	(1)
.IRP	1217	(3)	1223	(3)	1257	(3)	1506	(3)	1531	(3)	1603	(4)	1741	(4)	1969	(4)
	222	(1)	223	(1)	228	(1)	233	(1)	235	(1)	236	(1)	514	(1)	524	(1)
.LIBRARY	212	(1)	213	(1)	214	(1)										
.LIST	1217	(3)	1223	(3)	1506	(3)	1741	(4)	224	(1)	227	(1)	232	(1)	514	(1)
	524	(1)														
.MACRO	222	(1)	223	(1)	228	(1)	233	(1)	235	(1)	236	(1)				
.MDELETE	228	(1)														
.NLIST	1217	(3)	1223	(3)	1506	(3)	1741	(4)	224	(1)	227	(1)	232	(1)	514	(1)
	524	(1)														
.NOCROSS	224	(1)	225	(1)	226	(1)	227	(1)	229	(1)	230	(1)	231	(1)		
.NOSHOW	16	(1)														
.PAGE	1159	(3)	1211	(3)	1225	(3)	1406	(3)	1462	(3)	1501	(3)	1587	(3)	1657	(4)
	1692	(4)	1707	(4)	1737	(4)	1753	(4)	1790	(4)	1827	(4)	1860	(4)	1879	(4)
	1948	(4)	1977	(4)	2025	(4)	2056	(4)	206	(1)	2087	(4)	2123	(4)	2170	(4)
	2199	(4)	2218	(4)	2237	(4)	2275	(4)	2290	(4)	2323	(6)	2384	(6)	2411	(6)
	2423	(6)	2448	(6)	2494	(6)	2524	(6)	2554	(6)	2590	(6)	261	(1)	2622	(6)

```
                2654  (6)     2694  (6)     2724  (6)      343  (1)      360  (1)      389  (1)      461  (1)      510  (1)
                 657  (1)      666  (1)      776  (1)      822  (1)      848  (3)      941  (3)
.PSECT          1217  (3)     1223  (3)     1506  (3)     1741  (4)      232  (1)      264  (1)      346  (1)      363  (1)
                 514  (1)      524  (1)
.RESTORE        1217  (3)     1223  (3)     1506  (3)     1741  (4)      224  (1)      225  (1)      226  (1)      227  (1)
                 229  (1)      230  (1)      231  (1)      232  (1)      514  (1)      524  (1)
.SAVE           1217  (3)     1223  (3)     1506  (3)     1741  (4)      224  (1)      225  (1)      226  (1)      227  (1)
                 229  (1)      230  (1)      231  (1)      232  (1)      514  (1)      524  (1)
.SBTTL          1160  (3)     1463  (3)     1658  (4)     1708  (4)     1754  (4)     1828  (4)     1978  (4)     2171  (4)
                2385  (6)     2495  (6)      262  (1)     2623  (6)     2695  (6)      344  (1)      361  (1)      462  (1)
                 667  (1)      777  (1)      942  (3)
.SIGNED_WORD    1257  (3)     1531  (3)     1603  (4)
.SUBTITLE       1880  (4)     1949  (4)      207  (1)     2292  (5)     2424  (6)     2555  (6)      658  (1)      849  (3)
.TITLE            13  (1)
.WORD           1744  (4)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	19	1232 (3) 1235 (3) #-1236 (3) #-1237 (3) #-1510 (3) #-1511 (3)
		#-1583 (3) #-1584 (3) 1749 (4) #-2727 (6) #-2730 (6) #-2745 (6)
		#-2746 (6) #-2748 (6) #-2762 (6) 529 (1) #-532 (1) 534 (1)
FP	44	1023 (3) #-1061 (3) 1229 (3) 1230 (3) #-1241 (3) 1440 (3)
		#-1442 (3) #-1454 (3) 1456 (3) 528 (1) #-565 (1) #-642 (1)
		#-649 (1) #-708 (1) #-729 (1) 733 (1) 796 (1) #-798 (1)
		#-872 (3) #-873 (3) #-874 (3) #-875 (3) #-877 (3) #-880 (3)
		#-887 (3) #-888 (3) #-894 (3) #-896 (3) #-901 (3) #-903 (3)
		906 (3) #-907 (3) #-910 (3) 922 (3) #-924 (3) 926 (3)
		#-927 (3) 930 (3) #-933 (3) #-934 (3) #-937 (3) #-988 (3)
R0	245	#-1008 (3) #-1024 (3) #-1033 (3) #-1035 (3) #-1044 (3) #-1051 (3)
		#-1058 (3) #-1060 (3) #-1064 (3) #-1071 (3) #-1075 (3) #-1089 (3)
		#-1091 (3) #-1098 (3) #-1099 (3) #-1115 (3) #-1137 (3) #-1147 (3)
		#-1156 (3) #-1259 (3) #-1267 (3) #-1275 (3) 1278 (3) #-1282 (3)
		1285 (3) #-1293 (3) 1296 (3) 1298 (3) #-1305 (3) #-1308 (3)
		#-1317 (3) #-1318 (3) #-1320 (3) #-1323 (3) #-1325 (3) #-1326 (3)
		#-1335 (3) #-1339 (3) #-1344 (3) 1347 (3) #-1351 (3) 1352 (3)
		#-1375 (3) #-1386 (3) #-1388 (3) #-1390 (3) #-1396 (3) 1401 (3)
		#-1403 (3) #-1412 (3) #-1423 (3) #-1432 (3) #-1443 (3) #-1444 (3)
		#-1446 (3) #-1449 (3) #-1460 (3) #-1533 (3) #-1548 (3) #-1550 (3)
		#-1551 (3) #-1558 (3) #-1559 (3) #-1569 (3) 1570 (3) #-1574 (3)
		1575 (3) #-1583 (3) #-1585 (3) #-1590 (3) 1593 (3) #-1619 (4)
		#-1620 (4) #-1621 (4) #-1632 (4) #-1634 (4) #-1635 (4) #-1643 (4)
		#-1644 (4) #-1652 (4) #-1697 (4) #-1699 (4) #-1705 (4) #-1795 (4)
		#-1796 (4) #-1800 (4) #-1808 (4) #-1814 (4) #-1818 (4) #-1824 (4)
		#-1911 (4) #-1915 (4) #-1916 (4) #-1920 (4) #-1928 (4) #-1946 (4)
		#-1955 (4) #-1959 (4) #-1960 (4) #-1965 (4) #-1970 (4) #-1971 (4)
		#-1972 (4) #-1973 (4) #-1975 (4) #-2009 (4) #-2012 (4) #-2017 (4)
		#-2019 (4) #-2022 (4) #-2031 (4) #-2035 (4) #-2037 (4) #-2038 (4)
		#-2041 (4) #-2045 (4) #-2093 (4) #-2095 (4) #-2101 (4) #-2104 (4)
		#-2105 (4) #-2106 (4) #-2107 (4) #-2108 (4) #-2120 (4) #-2129 (4)
		#-2143 (4) #-2146 (4) #-2149 (4) #-2152 (4) #-2155 (4) #-2158 (4)
		#-2161 (4) #-2168 (4) #-2202 (4) #-2205 (4) #-2210 (4) #-2212 (4)
		#-2215 (4) #-2228 (4) #-2229 (4) #-2268 (4) #-2272 (4) #-2283 (4)
		2285 (4) 2287 (4) 2288 (4) #-2333 (6) #-2343 (6) 2344 (6)
		#-2415 (6) #-2416 (6) #-2417 (6) #-2418 (6) #-2419 (6) #-2420 (6)
		#-2451 (6) #-2468 (6) #-2469 (6) #-2475 (6) #-2482 (6) #-2530 (6)
		#-2531 (6) #-2550 (6) #-2552 (6) #-2593 (6) #-2606 (6) #-2607 (6)
		#-2613 (6) #-2619 (6) #-2657 (6) #-2672 (6) #-2679 (6) #-2680 (6)
		#-2681 (6) #-2685 (6) #-2688 (6) #-2692 (6) #-2727 (6) #-2729 (6)
		#-2748 (6) #-2749 (6) #-2759 (6) #-2764 (6) #-412 (1) #-439 (1)
		#-459 (1) #-547 (1) #-553 (1) #-556 (1) #-559 (1) #-564 (1)
		#-565 (1) #-566 (1) #-567 (1) #-571 (1) #-576 (1) #-578 (1)
		#-584 (1) #-588 (1) #-594 (1) #-600 (1) #-604 (1) #-613 (1)
		#-621 (1) #-635 (1) #-638 (1) #-664 (1) #-706 (1) 707 (1)
		#-734 (1) #-736 (1) #-738 (1) #-756 (1) #-765 (1) #-773 (1)
		#-783 (1) 797 (1) #-806 (1) #-807 (1) #-814 (1) #-819 (1)
		#-833 (2) #-835 (2) #-839 (2) #-842 (3) #-846 (3) #-876 (3)
		#-877 (3) 881 (3) #-902 (3) #-903 (3) #-904 (3) #-907 (3)
		#-912 (3) #-914 (3) #-918 (3) #-948 (3) #-955 (3) #-987 (3)

```
                #-995   (3)
R1          98  #-1023  (3)  #-1024  (3)  #-1041  (3)  #-1089  (3)  #-1098  (3)  #-1276  (3)
                #-1278  (3)  #-1283  (3)  #-1285  (3)  #-1294  (3)  #-1296  (3)  #-1298  (3)
                #-1312  (3)  #-1321  (3)  #-1341  (3)  #-1342  (3)  #-1376  (3)  #-1377  (3)
                #-1384  (3)  #-1425  (3)  #-1427  (3)  #-1430  (3)  #-1584  (3)  #-1591  (3)
                #-1593  (3)  #-1642  (4)  #-1643  (4)  #-1644  (4)  #-1652  (4)  #-1796  (4)
                #-1810  (4)  #-1816  (4)  #-1824  (4)  #-1911  (4)  #-1919  (4)  #-1924  (4)
                #-1926  (4)  #-1946  (4)  #-1955  (4)  #-1960  (4)  #-1962  (4)  #-1973  (4)
                #-1975  (4)  #-2009  (4)  #-2021  (4)  #-2039  (4)  #-2120  (4)  #-2202  (4)
                #-2214  (4)  #-2272  (4)  #-2332  (6)  #-2333  (6)  #-2343  (6)   2344   (6)
                #-2414  (6)  #-2415  (6)  #-2416  (6)  #-2451  (6)  #-2471  (6)  #-2473  (6)
                #-2482  (6)   2533   (6)  #-2541  (6)  #-2593  (6)  #-2609  (6)  #-2611  (6)
                #-2619  (6)  #-2620  (6)  #-2657  (6)  #-2683  (6)  #-2728  (6)  #-2738  (6)
                #-2744  (6)  #-2749  (6)  #-434   (1)  #-435   (1)   437   (1)  #-568   (1)
                #-569   (1)  #-576   (1)  #-579   (1)  #-620   (1)  #-649   (1)  #-653   (1)
                #-654   (1)  #-715   (1)   716   (1)  #-733   (1)  #-734   (1)  #-736   (1)
                #-747   (1)   752    (1)   761   (1)  #-818   (1)  #-923   (3)   928   (3)
                #-987   (3)
R10         77   1227   (3)  #-1236  (3)  #-1238  (3)  #-1241  (3)  #-1257  (3)  #-1271  (3)
                #-1275  (3)  #-1276  (3)  #-1277  (3)  #-1282  (3)  #-1283  (3)  #-1284  (3)
                #-1293  (3)  #-1294  (3)  #-1295  (3)  #-1303  (3)  #-1325  (3)  #-1326  (3)
                 1330   (3)  #-1339  (3)  #-1341  (3)   1347   (3)  #-1351  (3)   1352   (3)
                #-1355  (3)  #-1371  (3)  #-1372  (3)  #-1376  (3)  #-1377  (3)  #-1386  (3)
                #-1388  (3)  #-1391  (3)  #-1398  (3)  #-1409  (3)  #-1416  (3)  #-1418  (3)
                #-1419  (3)  #-1427  (3)  #-1434  (3)  #-1435  (3)  #-1442  (3)  #-1443  (3)
                #-1444  (3)  #-1446  (3)  #-1448  (3)   1508   (3)  #-1510  (3)  #-1513  (3)
                #-1531  (3)   1541   (3)  #-1543  (3)   1546   (3)  #-1550  (3)  #-1551  (3)
                 1555   (3)  #-1558  (3)  #-1559  (3)  #-1565  (3)  #-1569  (3)   1570   (3)
                #-1574  (3)   1575   (3)  #-1579  (3)  #-1590  (3)  #-1591  (3)  #-1592  (3)
                 526    (1)  #-564   (1)  #-566   (1)  #-567   (1)  #-568   (1)  #-820   (1)
                #-828   (1)
R11         44  #-1013  (3)   1127   (3)   1227   (3)  #-1237  (3)   1278   (3)   1285   (3)
                 1296   (3)   1298   (3)   1508   (3)   1593   (3)  #-1796  (4)  #-1802  (4)
                #-1806  (4)  #-1811  (4)  #-1824  (4)  #-1865  (4)  #-1871  (4)  #-1876  (4)
                 2332   (6)  #-2342  (6)  #-2364  (6)   2365   (6)   2368   (6)   2369   (6)
                 2377   (6)   2378   (6)   526    (1)  #-540   (1)  #-575   (1)  #-579   (1)
                 581    (1)  #-653   (1)  #-654   (1)   742    (1)  #-789   (1)  #-790   (1)
                #-804   (1)   808    (1)   812    (1)  #-814   (1)  #-815   (1)  #-827   (1)
                 834    (2)  #-842   (3)
R2         180  #-1003  (3)  #-1022  (3)  #-1052  (3)  #-1089  (3)  #-1090  (3)  #-1098  (3)
                #-1107  (3)  #-1126  (3)  #-1127  (3)  #-1128  (3)  #-1132  (3)  #-1135  (3)
                 1227   (3)  #-1277  (3)  #-1278  (3)  #-1284  (3)  #-1285  (3)  #-1295  (3)
                #-1296  (3)  #-1298  (3)  #-1302  (3)  #-1314  (3)  #-1324  (3)  #-1329  (3)
                #-1346  (3)  #-1372  (3)  #-1379  (3)  #-1381  (3)   1397   (3)   1400   (3)
                #-1440  (3)  #-1448  (3)  #-1450  (3)  #-1453  (3)  #-1454  (3)   1508   (3)
                #-1592  (3)  #-1593  (3)  #-1652  (4)  #-1653  (4)  #-1654  (4)  #-1795  (4)
                #-1796  (4)  #-1811  (4)  #-1812  (4)  #-1814  (4)  #-1824  (4)  #-1911  (4)
                #-1916  (4)  #-1919  (4)  #-1920  (4)  #-1921  (4)  #-1932  (4)  #-1935  (4)
                #-1938  (4)  #-1946  (4)  #-1955  (4)  #-1975  (4)  #-2009  (4)  #-2019  (4)
                #-2021  (4)  #-2022  (4)  #-2023  (4)  #-2047  (4)   2049   (4)  #-2061  (4)
                #-2064  (4)  #-2067  (4)  #-2072  (4)  #-2079  (4)  #-2099  (4)  #-2101  (4)
                #-2103  (4)  #-2105  (4)  #-2107  (4)  #-2110  (4)  #-2114  (4)   2116   (4)
                #-2120  (4)  #-2134  (4)   2136   (4)  #-2137  (4)   2139   (4)   2141   (4)
                 2142   (4)  #-2148  (4)  #-2150  (4)  #-2202  (4)  #-2212  (4)  #-2214  (4)
                #-2215  (4)  #-2216  (4)  #-2242  (4)  #-2245  (4)  #-2248  (4)  #-2253  (4)
                #-2260  (4)  #-2272  (4)  #-2451  (6)   2454   (6)  #-2458  (6)   2460   (6)
                #-2461  (6)  #-2466  (6)  #-2468  (6)  #-2472  (6)  #-2473  (6)  #-2482  (6)
```

```
                2529  (6)     #-2537  (6)     #-2538  (6)     #-2540  (6)     #-2543  (6)     #-2593  (6)
             #-2595  (6)     #-2599  (6)     #-2603  (6)     #-2606  (6)     #-2610  (6)     #-2611  (6)
             #-2619  (6)     #-2657  (6)     #-2659  (6)     #-2660  (6)     #-2662  (6)     #-2671  (6)
                2726  (6)     #-2730  (6)     #-403   (1)     #-404   (1)       405   (1)     #-406   (1)
             #-407   (1)       410   (1)     #-432   (1)       434   (1)     #-436   (1)     #-440   (1)
                526   (1)     #-575   (1)     #-577   (1)     #-581   (1)     #-582   (1)     #-587   (1)
             #-614   (1)     #-618   (1)     #-621   (1)     #-623   (1)     #-628   (1)     #-633   (1)
             #-640   (1)     #-645   (1)     #-647   (1)     #-703   (1)     #-705   (1)     #-706   (1)
             #-707   (1)       753   (1)       762   (1)     #-794   (1)     #-797   (1)     #-799   (1)
             #-802   (1)     #-804   (1)     #-805   (1)     #-809   (1)     #-812   (1)       815   (1)
             #-816   (1)     #-818   (1)     #-871   (3)     #-918   (3)     #-919   (3)     #-923   (3)
             #-927   (3)     #-928   (3)     #-938   (3)     #-987   (3)
R3      129  #-1016  (3)     #-1089  (3)     #-1098  (3)     #-1102  (3)     #-1111  (3)     #-1145  (3)
             #-1154  (3)       1227  (3)     #-1296  (3)     #-1297  (3)     #-1298  (3)     #-1303  (3)
             #-1308  (3)     #-1318  (3)     #-1320  (3)     #-1327  (3)     #-1358  (3)     #-1362  (3)
             #-1366  (3)     #-1370  (3)     #-1371  (3)     #-1374  (3)     #-1409  (3)     #-1410  (3)
             #-1416  (3)     #-1418  (3)     #-1419  (3)     #-1423  (3)     #-1429  (3)     #-1430  (3)
             #-1431  (3)     #-1433  (3)     #-1434  (3)     #-1435  (3)       1508  (3)     #-1606  (4)
             #-1610  (4)     #-1614  (4)     #-1618  (4)     #-1631  (4)     #-1646  (4)     #-1652  (4)
             #-1695  (4)     #-1698  (4)       1702  (4)       1744  (4)     #-1745  (4)     #-1746  (4)
             #-1747  (4)       1749  (4)     #-1750  (4)     #-1751  (4)     #-1796  (4)     #-1813  (4)
             #-1816  (4)     #-1824  (4)     #-1865  (4)     #-1867  (4)     #-1868  (4)     #-1871  (4)
             #-1876  (4)     #-1955  (4)     #-1975  (4)     #-2009  (4)     #-2014  (4)       2042  (4)
                2046  (4)       2105  (4)       2106  (4)       2113  (4)     #-2120  (4)     #-2202  (4)
             #-2208  (4)       2232  (4)       2270  (4)     #-2272  (4)     #-2451  (6)     #-2463  (6)
             #-2476  (6)     #-2478  (6)     #-2482  (6)     #-2593  (6)     #-2595  (6)     #-2601  (6)
             #-2603  (6)     #-2619  (6)     #-2657  (6)     #-2671  (6)     #-2678  (6)     #-2681  (6)
             #-2683  (6)     #-2686  (6)       2726  (6)     #-2736  (6)     #-2738  (6)     #-2740  (6)
             #-2742  (6)     #-2744  (6)     #-2745  (6)     #-2746  (6)     #-395   (1)     #-397   (1)
             #-403   (1)     #-406   (1)       433   (1)     #-440   (1)     #-444   (1)       526   (1)
             #-551   (1)     #-598   (1)     #-614   (1)     #-616   (1)     #-617   (1)     #-618   (1)
             #-623   (1)     #-625   (1)     #-627   (1)     #-633   (1)     #-640   (1)     #-645   (1)
             #-709   (1)     #-757   (1)     #-766   (1)     #-774   (1)     #-781   (1)     #-871   (3)
             #-938   (3)     #-987   (3)
R4      136  #-1028  (3)     #-1039  (3)     #-1043  (3)     #-1053  (3)     #-1059  (3)     #-1077  (3)
             #-1079  (3)     #-1082  (3)     #-1088  (3)     #-1089  (3)     #-1098  (3)     #-1106  (3)
             #-1120  (3)     #-1125  (3)     #-1126  (3)     #-1133  (3)     #-1135  (3)       1227  (3)
             #-1233  (3)     #-1234  (3)     #-1423  (3)     #-1432  (3)       1508  (3)     #-1700  (4)
             #-1796  (4)     #-1801  (4)     #-1802  (4)     #-1822  (4)     #-1824  (4)     #-1865  (4)
             #-1868  (4)     #-1870  (4)     #-1871  (4)     #-1875  (4)     #-1876  (4)     #-1955  (4)
             #-1975  (4)     #-2009  (4)     #-2030  (4)     #-2031  (4)     #-2042  (4)     #-2043  (4)
             #-2046  (4)     #-2047  (4)     #-2050  (4)     #-2052  (4)     #-2053  (4)     #-2093  (4)
             #-2096  (4)     #-2097  (4)     #-2098  (4)     #-2099  (4)     #-2104  (4)     #-2106  (4)
             #-2108  (4)     #-2109  (4)     #-2112  (4)     #-2113  (4)     #-2114  (4)     #-2117  (4)
             #-2119  (4)     #-2120  (4)     #-2137  (4)     #-2202  (4)     #-2223  (4)     #-2224  (4)
             #-2228  (4)     #-2231  (4)     #-2232  (4)     #-2234  (4)     #-2268  (4)     #-2269  (4)
             #-2270  (4)     #-2272  (4)     #-2283  (4)     #-2351  (6)     #-2357  (6)     #-2451  (6)
             #-2465  (6)     #-2466  (6)     #-2478  (6)     #-2479  (6)     #-2482  (6)     #-2593  (6)
             #-2598  (6)     #-2599  (6)     #-2614  (6)     #-2615  (6)     #-2617  (6)     #-2619  (6)
             #-2657  (6)     #-2671  (6)     #-2676  (6)     #-2678  (6)     #-2688  (6)     #-2691  (6)
                2726  (6)     #-2735  (6)     #-2736  (6)     #-2747  (6)     #-2754  (6)     #-2755  (6)
             #-2757  (6)     #-2762  (6)     #-2763  (6)     #-399   (1)     #-403   (1)       405   (1)
             #-406   (1)       407   (1)       434   (1)       526   (1)     #-531   (1)     #-614   (1)
             #-617   (1)     #-633   (1)     #-640   (1)     #-644   (1)     #-645   (1)     #-719   (1)
             #-722   (1)     #-726   (1)     #-737   (1)     #-741   (1)     #-745   (1)     #-746   (1)
             #-824   (1)     #-871   (3)     #-914   (3)     #-915   (3)     #-938   (3)     #-949   (3)
             #-987   (3)     #-995   (3)
```

R5	123	#-1006	(3)	#-1010	(3)	#-1014	(3)	#-1027	(3)	#-1041	(3)	#-1054	(3)
		#-1060	(3)	#-1076	(3)	#-1080	(3)	#-1083	(3)	#-1085	(3)	#-1089	(3)
		#-1098	(3)	#-1105	(3)	#-1122	(3)	#-1124	(3)	#-1126	(3)	#-1130	(3)
		#-1132	(3)	#-1135	(3)	1227	(3)	#-1378	(3)	#-1383	(3)	#-1425	(3)
		#-1430	(3)	1508	(3)	#-1796	(4)	#-1798	(4)	#-1801	(4)	#-1802	(4)
		#-1824	(4)	#-1911	(4)	#-1913	(4)	#-1923	(4)	#-1924	(4)	#-1946	(4)
		#-1959	(4)	#-1960	(4)	#-1971	(4)	#-2009	(4)	#-2013	(4)	#-2030	(4)
		#-2031	(4)	#-2047	(4)	#-2096	(4)	#-2097	(4)	#-2098	(4)	#-2101	(4)
		#-2103	(4)	#-2104	(4)	#-2112	(4)	#-2114	(4)	#-2120	(4)	#-2137	(4)
		#-2202	(4)	#-2206	(4)	#-2223	(4)	#-2224	(4)	#-2228	(4)	#-2272	(4)
		#-2283	(4)	#-2451	(6)	#-2464	(6)	#-2465	(6)	#-2466	(6)	#-2478	(6)
		#-2482	(6)	#-2593	(6)	#-2597	(6)	#-2598	(6)	#-2599	(6)	#-2614	(6)
		#-2619	(6)	#-2657	(6)	#-2662	(6)	#-2667	(6)	#-2671	(6)	#-2676	(6)
		#-2678	(6)	#-2688	(6)	2726	(6)	#-2734	(6)	#-2735	(6)	#-2736	(6)
		#-2748	(6)	#-2753	(6)	#-2754	(6)	#-2755	(6)	#-2762	(6)	#-396	(1)
		#-402	(1)	#-403	(1)	#-406	(1)	#-408	(1)	409	(1)	417	(1)
		431	(1)	#-443	(1)	449	(1)	526	(1)	#-614	(1)	#-615	(1)
		#-619	(1)	#-629	(1)	#-631	(1)	#-633	(1)	#-640	(1)	#-645	(1)
		#-708	(1)	710	(1)	720	(1)	724	(1)	728	(1)	#-871	(3)
		#-913	(3)	#-938	(3)	#-951	(3)	#-953	(3)	#-987	(3)	#-989	(3)
		#-996	(3)	#-999	(3)								
R6	71	#-1062	(3)	#-1064	(3)	#-1089	(3)	#-1098	(3)	1227	(3)	#-1235	(3)
		1508	(3)	#-1512	(3)	#-1541	(3)	#-1546	(3)	#-1555	(3)	#-1594	(3)
		#-1603	(4)	#-1606	(4)	#-1610	(4)	#-1614	(4)	#-1618	(4)	#-1619	(4)
		#-1620	(4)	#-1621	(4)	1624	(4)	#-1631	(4)	#-1655	(4)	#-1746	(4)
		#-1747	(4)	#-1750	(4)	#-1751	(4)	#-1796	(4)	#-1804	(4)	#-1806	(4)
		#-1824	(4)	#-1911	(4)	#-1926	(4)	#-1932	(4)	#-1946	(4)	#-2009	(4)
		#-2033	(4)	#-2035	(4)	#-2062	(4)	#-2064	(4)	#-2068	(4)	#-2120	(4)
		#-2202	(4)	#-2226	(4)	#-2229	(4)	#-2243	(4)	#-2245	(4)	#-2249	(4)
		#-2272	(4)	#-2451	(6)	#-2452	(6)	#-2455	(6)	#-2480	(6)	#-2482	(6)
		#-2487	(6)	#-2527	(6)	#-2535	(6)	#-2541	(6)	526	(1)	#-534	(1)
		#-538	(1)	#-614	(1)	#-620	(1)	#-625	(1)	#-633	(1)	#-640	(1)
		#-645	(1)	825	(1)	#-871	(3)	#-938	(3)	#-987	(3)		
R7	79	1049	(3)	1070	(3)	1074	(3)	#-1089	(3)	#-1098	(3)	#-1118	(3)
		1227	(3)	#-1378	(3)	#-1383	(3)	#-1393	(3)	#-1410	(3)	1455	(3)
		1508	(3)	#-1511	(3)	#-1644	(4)	#-1653	(4)	#-1796	(4)	#-1824	(4)
		#-1865	(4)	#-1876	(4)	#-1911	(4)	#-1923	(4)	#-1924	(4)	#-1928	(4)
		#-1930	(4)	#-1946	(4)	#-2009	(4)	#-2015	(4)	2043	(4)	2050	(4)
		2052	(4)	2107	(4)	2108	(4)	2117	(4)	2119	(4)	#-2120	(4)
		#-2202	(4)	#-2207	(4)	2231	(4)	2269	(4)	#-2272	(4)	#-2352	(6)
		#-2354	(6)	#-2358	(6)	#-2360	(6)	#-2370	(6)	#-2379	(6)	526	(1)
		#-556	(1)	#-563	(1)	#-583	(1)	#-587	(1)	#-604	(1)	#-641	(1)
		#-652	(1)	#-789	(1)	#-790	(1)	#-791	(1)	#-792	(1)	#-793	(1)
		#-871	(3)	#-875	(3)	#-876	(3)	#-877	(3)	881	(3)	#-882	(3)
		#-884	(3)	#-886	(3)	#-887	(3)	#-899	(3)	#-901	(3)	#-902	(3)
		904	(3)	#-938	(3)	#-987	(3)	#-988	(3)	992	(3)		
R8	22	1227	(3)	#-1232	(3)	#-1233	(3)	#-1327	(3)	#-1393	(3)	#-1431	(3)
		1508	(3)	#-1584	(3)	#-1644	(4)	#-1654	(4)	#-1697	(4)	526	(1)
		#-529	(1)	#-530	(1)	#-531	(1)	#-563	(1)	#-641	(1)	#-652	(1)
		#-807	(1)	#-830	(1)	#-841	(3)						
R9	29	1227	(3)	#-1263	(3)	#-1271	(3)	#-1278	(3)	#-1285	(3)	#-1289	(3)
		#-1297	(3)	#-1309	(3)	#-1315	(3)	#-1342	(3)	#-1346	(3)	#-1390	(3)
		#-1415	(3)	#-1421	(3)	1508	(3)	#-1593	(3)	#-1623	(4)	#-1624	(4)
		#-1633	(4)	#-1635	(4)	#-1646	(4)	526	(1)	#-790	(1)	#-791	(1)
		#-792	(1)	#-793	(1)	#-805	(1)	809	(1)				
SP	148	#-1002	(3)	#-1010	(3)	#-1014	(3)	#-1015	(3)	#-1021	(3)	#-1031	(3)
		#-1047	(3)	#-1067	(3)	#-1075	(3)	#-1076	(3)	#-1088	(3)	#-1096	(3)

#-1110	(3)	#-1147	(3)	#-1156	(3)	#-1229	(3)	#-1302	(3)	#-1310	(3)
#-1311	(3)	#-1314	(3)	#-1317	(3)	#-1321	(3)	#-1324	(3)	#-1329	(3)
#-1392	(3)	#-1396	(3)	#-1398	(3)	1647	(4)	1648	(4)	1649	(4)
1745	(4)	#-1808	(4)	#-1810	(4)	#-1935	(4)	#-1936	(4)	#-1937	(4)
#-1938	(4)	#-1939	(4)	#-1962	(4)	#-1969	(4)	#-1970	(4)	#-2067	(4)
#-2070	(4)	#-2071	(4)	#-2072	(4)	#-2073	(4)	#-2079	(4)	#-2080	(4)
#-2134	(4)	#-2142	(4)	#-2150	(4)	#-2154	(4)	#-2157	(4)	#-2160	(4)
#-2248	(4)	#-2251	(4)	#-2252	(4)	#-2253	(4)	#-2254	(4)	#-2260	(4)
#-2261	(4)	2328	(6)	2329	(6)	#-2330	(6)	#-2337	(6)	#-2338	(6)
2341	(6)	2363	(6)	#-2367	(6)	#-2375	(6)	#-2456	(6)	#-2458	(6)
#-2461	(6)	#-2462	(6)	#-2469	(6)	#-2471	(6)	#-2481	(6)	2485	(6)
#-2488	(6)	#-2492	(6)	#-2544	(6)	#-2545	(6)	#-2594	(6)	#-2607	(6)
#-2609	(6)	#-2615	(6)	#-2618	(6)	#-2670	(6)	#-2679	(6)	#-395	(1)
397	(1)	#-398	(1)	399	(1)	#-400	(1)	401	(1)	402	(1)
#-432	(1)	#-442	(1)	#-443	(1)	#-444	(1)	#-528	(1)	536	(1)
538	(1)	#-559	(1)	#-582	(1)	#-583	(1)	#-584	(1)	#-585	(1)
#-635	(1)	709	(1)	#-713	(1)	#-714	(1)	#-721	(1)	#-725	(1)
#-730	(1)	#-732	(1)	#-736	(1)	#-740	(1)	#-743	(1)	#-744	(1)
#-746	(1)	751	(1)	#-757	(1)	#-766	(1)	#-774	(1)	#-798	(1)
#-799	(1)	#-824	(1)	826	(1)	#-830	(1)	#-832	(2)	#-989	(3)

```
+-------------------------+
! Performance indicators !
+-------------------------+
```

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.02	00:00:00.19
Command processing	872	00:00:00.20	00:00:00.84
Pass 1	1576	00:00:05.45	00:00:07.48
Symbol table sort	20	00:00:00.36	00:00:00.36
Pass 2	424	00:00:01.64	00:00:03.26
Symbol table output	2	00:00:00.08	00:00:00.17
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	84	00:00:00.96	00:00:01.46
Assembler run totals	3009	00:00:08.71	00:00:13.76

The working set limit was 3012 pages.
240316 bytes (470 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1064 non-local and 207 local symbols.
2767 source lines were read in Pass 1, producing 102 object records in Pass 2.
113 pages of virtual memory were used to define 39 macros.

```
                              +---------------------------+
                              ! Macro library statistics !
                              +---------------------------+

Macro library name                                 Macros defined
------------------                                 --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8                      12
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                         7
SYS$COMMON:[SYSLIB]LIB.MLB;2                                    0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8                       0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                         0
SYS$COMMON:[SYSLIB]LIB.MLB;2                                    0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                               14
TOTALS (all libraries)                                         33

1129 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ATTACH.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(2)   198   Declarations
(3)   275   DRIVER FIXED DATA AREA
(4)   378   BOO$QIO - BOOTSTRAP QIO ROUTINE
(5)   675   BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QIO
(6)   736   BOO$PURDPR - Purge UBA Buffered Datapath
(8)   840   BOO$SELECT - Select boot driver
(9)   924   BOO$MOVE - Select and move boot driver

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element BOOTDRIVR.MAR
0000     2 ; *7     4-DEC-1986 13:28:59 BAGGETT "ADDED CHANGES FOR RT UVAX"
0000     3 ; *6    12-MAY-1986 10:29:22 BARANSKI "Cleaning up Object Libraries."
0000     4 ; *5     7-MAY-1986 16:51:32 BARANSKI "Documentation Check."
0000     5 ; *4     5-MAY-1986 16:47:41 SALEM "DONE"
0000     6 ; *3    25-APR-1986 16:03:39 MUSE "<no reason given>"
0000     7 ; *2    14-FEB-1986 09:59:45 BAGGETT "PUT BACK CHANGE DELETED BY CMS ERROR"
0000     8 ; *1     6-FEB-1986 15:13:49 DS ""
0000     9 ; DEC/CMS REPLACEMENT HISTORY, Element BOOTDRIVR.MAR
0000    10         .TITLE  BOOTDRIVR       *** BOOTDRIVR non-interrupt I/O
0000    11         .IDENT  "10.0-28"                                              ;G:7
0000    12
0000    13 ;
0000    14 ;*****************************************************************************
0000    15 ;*                                                                           *
0000    16 ;*  COPYRIGHT (c) 1979, 1980, 1983, 1984, 1985, 1986                         *
0000    17 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    18 ;*                                                                           *
0000    19 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    20 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    21 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    22 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    23 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    24 ;*  TRANSFERRED.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    27 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    28 ;*  CORPORATION.                                                             *
0000    29 ;*                                                                           *
0000    30 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    31 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    32 ;*                                                                           *
0000    33 ;*****************************************************************************
0000    34 ;
0000    35 ;++
0000    36 ;
0000    37 ;  FACILITY:
0000    38 ;
0000    39 ;       Minimal bootstrap driver for all VMS system disks.
0000    40 ;
0000    41 ; ENVIRONMENT:
0000    42 ;
0000    43 ;       Runs at IPL 31, kernel mode, memory management may be on or off,
0000    44 ;       IS=1 (running on interrupt stack), code must be PIC.
0000    45 ;
0000    46 ; ABSTRACT:
0000    47 ;
0000    48 ;       This module contains a routine called BOO$QIO that handles I/O
0000    49 ;       transfers to and from the VMS system disks.
0000    50 ;
0000    51 ; AUTHOR:
0000    52 ;
0000    53 ;       The VMS group
0000    54 ;
0000    55 ; MODIFIED BY:
0000    56 ;       28      M. Baggett      18-Nov-1986     VDS 10.3                ;G:7
0000    57 ;               Added changes RT uVAX to match AURORA.                  ;G:7
```

```
0000     58 ;                                                                    ;G:5
0000     59 ;       27      T. Salem        5-May-1986      VDS 10.0             ;G:4
0000     60 ;               Changed Processor SID to Microvax defined in $prdef  ;G:4
0000     61 ;                                                                    ;G:4
0000     62 ;       26      M. Baggett      17-Jan-1886     VDS 9.1              ;G:2
0000     63 ;               Change error code from DS$_NOSUPPORT to DS$_UNSUPPDEV ;G:2
0000     64 ;                                                                    ;G:2
0000     65 ;       25      M. Baggett      16-Jan-1986     VDS 9.1
0000     66 ;               Removed HALT and added error status for unsupported
0000     67 ;               load devices.
0000     68 ;
0000     69 ;       24      M. Baggett      20-Sep-1985     VDS 9.0
0000     70 ;               Added back changes for AURORA.
0000     71 ;
0000     72 ;       23      Jim Shelhamer   12-Sept-1985    Version 8.3
0000     73 ;               Reset BOO$GL_UMR_TMPL. It gets set up for UNIBUS
0000     74 ;               devices, but fails to be reset for MASSBUS devices.
0000     75 ;               Also, went to better revision numbering, ie [23] not
0000     76 ;               [0223], which runs off the listing.
0000     77 ;
0000     78 ;       0222    Bob Bergazzi    July 25,1985    Version  8.3
0000     79 ;               Add code from BDABTDRIVR to set up pseudo map register
0000     80 ;               address in EXE$GL_UMR_PHY and EXE$GL_UMR_VIR.
0000     81 ;
0000     82 ;       0221    Bob Bergazzi    July 24,1985    Version 8.3
0000     83 ;               Made compatible with VMB version 13.
0000     84 ;
0000     85 ;       0220    Bob Bergazzi    June 3, 1985    Version 8.3
0000     86 ;               Added fixed data areas onto BOO$AL_VECTOR in order to be
0000     87 ;               compatible with VMS bootdrivrs.
0000     88 ;
0000     89 ;       0219    Bob Bergazzi    Jan. 16, 1985   Version 8.1
0000     90 ;               Added real BOO$PURDPR for 11/8SS.
0000     91 ;
0000     92 ;       0218    Bob Bergazzi    Nov. 28,1984
0000     93 ;               Added code for the 11/8SS cpu to BOO$PURDPR. Also, moved
0000     94 ;               edit which makes BOO$SELECT global into the .ENH from the
0000     95 ;               .UPD due to an edit clash.
0000     96 ;
0000     97 ;       0217    Bob Bergazzi    27-May-1983
0000     98 ;               Added another dispatch for CPU XXX in the CPUDISP MACRO.
0000     99 ;               This edit was put in by the supervisor group in the .ENH
0000    100 ;               file because the edit applies to a change in the .ENH file.
0000    101 ;               Eventually a new BOOTDRIVR module may be taken from VMS
0000    102 ;               which has the correct code in it.
0000    103 ;
0000    104 ;       DS02    Marion Baggett, 4-March-1983 Version 6.11
0000    105 ;               Added support for HSCCI booting.
0000    106 ;
0000    107 ;       DS01    Roger Riggs, 24-oct-1980, version 6.1
0000    108 ;               Added changes to version 4 VMB/BOOTDRIVR to make it usable
0000    109 ;               as bootdrivers for the Diagnostic Supervisor.
0000    110 ;               Note that some changes are here an others are in
0000    111 ;               BOOTDRIVR.ENH, since there are conflicts between VMS
0000    112 ;               changes and DS changes.
0000    113 ;
0000    114 ;       0216    CAS0001         C.A. Samuelson          30-Apr-1980
```

```
                                0000   115 ;               Add Purge of Buffered Datapath for UBA boot drivers
                                0000   116 ;
                                0000   117 ;       0215    TMH0001         Tim Halvorsen           10-Mar-1980
                                0000   118 ;               Mask off upper bits of PTE when constructing map register.
                                0000   119 ;
                                0000   120 ;       0214    KDM0092         Kathleen D. Morse       23-Jan-1980     16:40
                                0000   121 ;               Change VMB version number from 2 to 3.
                                0000   122 ;
                                0000   123 ;       00213   SRB0005         Steve Beckhardt         27-Nov-1979     16:10
                                0000   124 ;               Added code to support passing driver name from
                                0000   125 ;               boot driver to SYSBOOT.  Also added version # vector.
                                0000   126 ;                                                                       ;G:5
                                0000   127 ;       0212    SRB0004         Steve Beckhardt         31-Oct-1979     10:45
                                0000   128 ;               Moved all drivers into separate modules.  Added code
                                0000   129 ;               for connecting only the necessary driver.
                                0000   130 ;                                                                       ;G:5
                                0000   131 ;       0211    SRB0003         Steve Beckhardt         25-Oct-1979     16:40
                                0000   132 ;               Added console TU58 driver.
                                0000   133 ;                                                                       ;G:5
                                0000   134 ;       0210    CHF0001         Charlie Franks          23-Sep-1979     15:50
                                0000   135 ;               Added code and definitions to support RM80 bootable
                                0000   136 ;               system, includes: SSE and SSEI bit definitions, additions
                                0000   137 ;               to the MBAGEOM table and the MBA_DISK_DRUVER routine, and
                                0000   138 ;               the SSERROR skip sector routine.
                                0000   139 ;               Corrected byte count remaining update after encountering
                                0000   140 ;               a transfer error in MBA_DISK_DRIVER (convert MBA$L_BCR from
                                0000   141 ;               word to longword).
                                0000   142 ;                                                                       ;G:5
                                0000   143 ;       0209    SRB0002         Steve Beckhardt         13-Sep-1979     16:50
                                0000   144 ;               Added symbol EXE$GB_CPUTYPE (removed from module VMB)
                                0000   145 ;               so that it gets loaded into non-paged pool with the
                                0000   146 ;               boot driver.
                                0000   147 ;                                                                       ;G:5
                                0000   148 ;       0208    SRB0001         Steve Beckhardt         27-Aug-1979     10:30
                                0000   149 ;               Added boot device definitions ($BTDDEF).  Fixed CPUDISP
                                0000   150 ;               macro.  Added support for booting from console block
                                0000   151 ;               storage device.  Added 11/780 console floppy driver.
                                0000   152 ;               Changed boot value of R0 when booting from RL01/2 from
                                0000   153 ;               0 to 2.
                                0000   154 ;                                                                       ;G:5
                                0000   155 ;       0207    CHP0006         C. Peters               23-Jul-1979     16:16
                                0000   156 ;               Correct calculation of number of map registers needed.
                                0000   157 ;               Always invalidate one past the number required.
                                0000   158 ;                                                                       ;G:5
                                0000   159 ;       0206    CHP0005         C. Peters               20-Jul-1979     12:14
                                0000   160 ;               Add geometry of RM05 to MBAGEOM table.
                                0000   161 ;                                                                       ;G:5
                                0000   162 ;       0205    CHP0004         C. Peters               06-Jul-1979     14:06
                                0000   163 ;               Recode RK06/7 drive initialization code to correct bug
                                0000   164 ;               in RK07 handling.
                                0000   165 ;                                                                       ;G:5
                                0000   166 ;       0204    CHP0003         C. Peters               24-May-1979     14:24
                                0000   167 ;               Only invalidate the last UBA map register if the
                                0000   168 ;               transfer was page-aligned. Remove all code relating
                                0000   169 ;               to COMET-specific UBA map register format. All UBA
                                0000   170 ;               map registers are now alike. Delete definition of
                                0000   171 ;               storage for CPU_TYPE. Use instead a global symbol
```

```
                0000   172 ;               EXE$GB_CPUTYPE initialized by VMB to hold the CPU
                0000   173 ;               identification code.
                0000   174 ;                                                                   ;G:5
                0000   175 ;       0203    CHP0002         C. Peters                 11-May-1979     13:20
                0000   176 ;               Add code to invalidate the last UBA map register to stop
                0000   177 ;               a wild transfer. This code is CPU-specific.
                0000   178 ;                                                                   ;G:5
                0000   179 ;       0202    CHP0001         C. Peters                 23-Mar-1979     8:48
                0000   180 ;               Fill out module header. Add CPUDISP macro. Add local
                0000   181 ;               symbol CPU_TYPE for CPUDISP's use. Add many comments.
                0000   182 ;               Extract processor type from PR$_SID and store in local
                0000   183 ;               CPU_TYPE. Find adapter type from CONFREG, and store in
                0000   184 ;               general register. Initialize map registers in a CPU-
                0000   185 ;               and adapter-dependent way. Determine which driver to
                0000   186 ;               use based on adapter and device type. Add routine
                0000   187 ;               header comments to MBA and RK06/7 driver code. Derive
                0000   188 ;               device CSR address in RK06/7 driver from PR$_MAPEN
                0000   189 ;               value used as index into RPB. Add routine header
                0000   190 ;               comments to ECC routine. Replace .PAGEs with form feeds.
                0000   191 ;               Add code and definitions to support RL02 bootable
                0000   192 ;               system: includes $RLDEF, a test for the RL02 device
                0000   193 ;               type, and the minimal RL02 driver code (written by C.
                0000   194 ;               Franks).
                0000   195 ;
                0000   196 ;--
```

```
                        0000    198          .SBTTL  Declarations
                        0000    199
                        0000    200 ;
                        0000    201 ; INCLUDE FILES;
                        0000    202 ;
                        0000    203          .LIBRARY        /SYS$LIBRARY:LIB/       ;
                        0000    204          .LIBRARY        /$DS/                   ;
                        0000    205          .LIBRARY        /$DIAG/                 ;
                        0000    206
                        0000    207 ;
                        0000    208 ; MACRO LIBRARY CALLS
                        0000    209 ;
                        0000    210          $DS_DSDEF                              ; DS error codes                        [25]
                        0000    211          $BQODEF                                ; Define boot qio offsets               [022
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    212          $BTDDEF                                ; Define boot device types
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                00000021 0000   213                  BTD$K_BDA       = 33       ;                                       [022
                        0000    214          $BUADEF                                ; Define BUA adapter registers          [218
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    215          $IODEF                                 ; DEFINE I/O FUNCTION CODES
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    216          $MBADEF                                ; DEFINE MASSBUS ADAPTER REGISTERS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    217          $NDTDEF                                ; NEXUS device types
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    218          $PRDEF                                 ; DEFINE PROCESSOR REGISTERS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    219          $PTEDEF                                ; DEFINE PAGE TABLE ENTRY FIELDS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    220          $RPBDEF                                ; DEFINE RESTART PARAMETER BLOCK
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    221          $SSDEF                                 ; DEFINE STATUS CODES
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    222          $UBADEF                                ; UNIBUS ADAPTER REGISTER DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    223          $UBIDEF                                ; 11/750 UNIBUS adapter regs.
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    224          $VADEF                                 ; DEFINE VIRTUAL ADDRESS FIELDS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    225
                        0000    226 ;
                        0000    227 ; MACROS
                        0000    228 ;
```

```
                    0000   229
                    0000   230 ;
                    0000   231 ; CPU type dispatch macro:
                    0000   232 ;
                    0000   233 ;       The addresses in the address list are:
                    0000   234 ;               address of code for CPU type=1 (11/780)
                    0000   235 ;               address of code for CPU type=2 (11/750)
                    0000   236 ;               address of code for CPU type=3 (?)
                    0000   237 ;
                    0000   238 ;       CPUDISP is invoked to handle CPU differences in line, e.g.,
                    0000   239 ;
                    0000   240 ;               CPUDISP <10$,20$>       ;*DISPATCH ON CPU TYPE*
                    0000   241 ;       10$:    <11/780 specific code>
                    0000   242 ;       20$:    <11/750 specific code>
                    0000   243 ;                                       ;*END OF CPU-DEPENDENT CODE*
                    0000   244 ;
                    0000   245 ; When the next CPU is added, all occurrences of CPUDISP must be
                    0000   246 ; expanded to handle three CPU specific paths.
                    0000   247 ;
                    0000   248
                    0000   249         .MACRO  CPUDISP,ADDRLIST
                    0000   250         CASE    W^EXE$GB_CPUTYPE,<ADDRLIST>,LIMIT=#PR$_SID_TYP780,TYPE=B
                    0000   251         .ENDM   CPUDISP
                    0000   252
                    0000   253 ;
                    0000   254 ; LOCAL SYMBOLS
                    0000   255 ;
                    0000   256
                    0000   257         $DEFINI BDT                     ; Define Boot Driver Table offsets
                    0000               .SAVE   LOCAL_BLOCK
                    0000   258
                    0000   259 $DEF     BDT$L_CPUTYPE   .BLKW   1       ;  CPU type
                    0000               .IIF    NB,BDT$L_CPUTYPE,       BDT$L_CPUTYPE:
00000002            0000               .IIF    NB,.BLKW,       .BLKW   1
                    0002   260 $DEF     BDT$L_DEVTYPE   .BLKW   1       ;  Boot R0 device type
                    0002               .IIF    NB,BDT$L_DEVTYPE,       BDT$L_DEVTYPE:
00000004            0002               .IIF    NB,.BLKW,       .BLKW   1
                    0004   261 $DEF     BDT$L_ACTION    .BLKL   1       ;  Action routine
                    0004               .IIF    NB,BDT$L_ACTION,        BDT$L_ACTION:
00000008            0004               .IIF    NB,.BLKL,       .BLKL   1
                    0008   262 $DEF     BDT$L_SIZE      .BLKL   1       ;  Driver size
                    0008               .IIF    NB,BDT$L_SIZE,  BDT$L_SIZE:
0000000C            0008               .IIF    NB,.BLKL,       .BLKL   1
                    000C   263 $DEF     BDT$L_ADDR      .BLKL   1       ;  Driver address (offset)
                    000C               .IIF    NB,BDT$L_ADDR,  BDT$L_ADDR:
00000010            000C               .IIF    NB,.BLKL,       .BLKL   1
                    0010   264 $DEF     BDT$L_ENTRY     .BLKL   1       ;  Driver entry (offset from address)
                    0010               .IIF    NB,BDT$L_ENTRY, BDT$L_ENTRY:
00000014            0010               .IIF    NB,.BLKL,       .BLKL   1
                    0014   265 $DEF     BDT$L_DRIVRNAME .BLKL   1       ;  Driver name (offset from address)
                    0014               .IIF    NB,BDT$L_DRIVRNAME,     BDT$L_DRIVRNAME:
00000018            0014               .IIF    NB,.BLKL,       .BLKL   1
                    0018   266
00000018            0018   267 BDT$K_LENGTH=.                         ;  Length of entry
                    0018   268
                    0018   269         $DEFEND BDT                     ; End of Boot Driver Table definitions
                    0000               .RESTORE
```

```
                        0000   270 ;
                        0000   271 ; WEAK SYMBOLS
                        0000   272 ;
                        0000   273 .WEAK   BDA_MAPREGS                  ; KDB50 pseudo map registers          [022
```

```
                                        .SBTTL  DRIVER FIXED DATA AREA

                                ;
                                ;       FIXED DATA CELLS FOR BOOTSTRAP DRIVER
                                ;

                        00000000        .PSECT  BOOTDRIVR_1,LONG        ; CERTAIN DRIVERS REQUIRE ALIGNMENT!    [022

                                BOO$AL_VECTOR::                         ; VECTOR TO BOOT DRIVER ENTRY POINTS
                00000050'       .LONG   BOO$QIO-BOO$AL_VECTOR   ; OFFSET TO BOOTSTRAP QIO ROUTINE
                000001A8'       .LONG   BOO$MAP-BOO$AL_VECTOR   ; OFFSET TO MAPPING ROUTINE
                00000000'       .LONG   BOO$SELECT-BOO$AL_VECTOR; OFFSET TO BOOTSTRAP I/O DRIVER
                                                                ; INITIALLY SET TO ROUTINE WHICH
                                                                ; SELECTS DRIVER
                00000000        .LONG   0                       ; OFFSET TO SYSTEM DISK DRIVER NAME
                                                                ; (ASCIC STRING). SET UP BY BOOT DRIVER.
                                ;
                                ; The next two words are the version number and the version number check fields.
                                ; (The second word is the ones complement of the first word.)  The version
                                ; number should be incremented whenever the interface between VMB and the
                                ; rest of the system changes.  Release 1.0 VMB did not contain these fields.
                                ;
                                ;       Version 2 - Boot driver passes system disk driver name to SYSBOOT
                                ;       Version 3 - VMB build memory description vector into RPB
                                ;       Version 4 - VMB BOOTDRIVR purges UBA buffered datapath, all drivers
                                ;                   return to BOOTDRIVR with success/failure status
                                ;       Version 5 - VMB passes an argument list to the secondary boot
                                ;                   in AP.  FILEREAD cacheing is present.
                                ;       Version 6 - VMB passes nexus device type of boot adapter in
                                ;                   RPB$B_BOOTNDT.
                                ;       Version 7 - BOO$AL_VECTOR now has new entry points for RESELECTing
                                ;                   a driver and UNIT_INIT for a driver.  Also new info
                                ;                   passed in the argument list.
                                ;       Version 8 - BOO$AL_VECTOR now has a new cell: BQO$L_UCODE.
                                ;       Version 9 - VMB passes number of bad memory pages found during
                                ;                   bootstrap scan in RPB$L_BADPGS.
                                ;       Version 10- BOO$AL_VECTOR has two new cells: UNIT_DISC and DEVNAME
                                ;
                                ;       Version 11- BOO$AL_VECTOR has two new cells:  TENUSEC and UBDELAY
                                ;
                                ;       Version 12- RPB$W_BOOTNDT is defined, high byte of this word must
                                ;                   be cleared in SYSBOOT for versions of VMB less than 12.
                                ;
                                ;       Version 13- RPB$B_CTRLLTR is defined; SYSBOOT must clear this field
                                ;                   for older versions of VMB.
                                ;

                0000000D        VMB_VERSION = 13

                                        ASSUME  <.-BOO$AL_VECTOR> EQ BQO$W_VERSION
               FFF2 000D        .WORD   VMB_VERSION,^C<VMB_VERSION> ; VERSION # AND VERSION # CHECK FIELD.
                00000000'       .LONG   BOO$RESELECT-BOO$AL_VECTOR ; Offset to set new driver
                00000070'       .LONG   BOO$MOVE-BOO$AL_VECTOR  ; Offset to routine to select and move
                                ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UNIT_INIT
                00000000        .LONG   0                       ; Offset to UNIT_INIT
                                ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_AUXDRNAME
                00000000        .LONG   0                       ; Offset to auxiliary driver name
```

```
                     0024   332                                    ;  second driver
                     0024   333          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_DIS
                     0024   334 BOO$GL_UMR_DIS::                    ; Number of map registers disabled
            00000000 0024   335          .LONG   0
                     0028   336          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UCODE
                     0028   337 BOO$GL_UCODE::                     ; Address of microcode in memory
            00000000 0028   338          .LONG   0
                     002C   339          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UNIT_DISC
            00000000 002C   340          .LONG   0                  ; Offset to UNIT_DISC
                     0030   341          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_DEVNAME
            00000000 0030   342          .LONG   0                  ; Offset to boot device name
                     0034   343          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_TMPL
                     0034   344 BOO$GL_UMR_TMPL::                   ; UNIBUS map register template
            80000000 0034   345          .LONG   UBA$M_MAP_VALID    ;  Default is valid, no buff data path
                     0038   346          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_UMR_DP
                     0038   347 BOO$GB_UMR_DP::                     ; UNIBUS map register data path
                  01 0038   348          .BYTE   1                  ;  Default is Buffered #1
                     0039   349          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_CPUTYPE
                     0039   350 EXE$GB_CPUTYPE::                    ; Location to hold processor
                  01 0039   351          .BYTE   1                  ;  identification code
                     003A   352          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_CPUDATA
                     003A   353 EXE$GB_CPUDATA::                    ; Location to hold contents of SID.
            00000001 003A   354          .LONG   1
                     003E   355          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_TENUSEC
                     003E   356 EXE$GL_TENUSEC::                    ; Location to hold TIMEDWAIT delay count
            00000001 003E   357          .LONG   1
                     0042   358          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UBDELAY
                     0042   359 EXE$GL_UBDELAY::                    ; Location to hold TIMEDWAIT delay count
            00000001 0042   360          .LONG   1
                     0046   361 ;
                     0046   362 ; The addition of two longwords here, EXE$GL_UMR_PHY, and EXE$GL_UMR_VIR, which
                     0046   363 ; point to the pseudo map registers, allows BOO$QIO to locate these longwords
                     0046   364 ; in memory.
                0000 0046   365          .WORD   0                  ; Reserved, puts structure back on
                     0048   366                                     ;  longword boundaries.
                     0048   367 ;        ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_PHY
            00000048 0048   368 BQO$L_UMR_PHY = .-BOO$AL_VECTOR     ; Remove this line, restore previous with ne
                     0048   369 EXE$GL_UMR_PHY::                    ; Location to hold physical address of
            00000000 0048   370          .LONG   0                  ;  pseudo map registers.
                     004C   371
                     004C   372 ;        ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_VIR
            0000004C 004C   373 BQO$L_UMR_VIR = .-BOO$AL_VECTOR     ; Remove this line, restore previous with ne
                     004C   374 EXE$GL_UMR_VIR::                    ; Location to hold virtual address of
            00000000 004C   375          .LONG   0                  ;  pseudo map registers.
                     0050   376 ;                                                                        end
```

```
                                0050   378          .SBTTL  BOO$QIO - BOOTSTRAP QIO ROUTINE
                                0050   379
                                0050   380 ;++
                                0050   381 ; FUNCTIONAL DESCRIPTION:
                                0050   382 ;
                                0050   383 ;       BOO$QIO PROVIDES THE DEVICE INDEPENDENT I/O INTERFACE FOR BOTH
                                0050   384 ;       READING AND WRITING THE BOOTSTRAP DEVICE.
                                0050   385 ;
                                0050   386 ; CALLING SEQUENCE:
                                0050   387 ;
                                0050   388 ;       CALLG   ARGLIST,BOO$QIO
                                0050   389 ;
                                0050   390 ; INPUT PARAMETERS:
                                0050   391 ;
                                0050   392 ;       BUF(AP)  - BUFFER ADDRESS
                                0050   393 ;       SIZE(AP) - SIZE OF BUFFER IN BYTES
                                0050   394 ;       LBN(AP)  - LOGICAL BLOCK NUMBER
                                0050   395 ;       FUNC(AP) - FUNCTION CODE
                                0050   396 ;                  ACCEPTS IO$_READLBLK AND IO$_WRITELBLK
                                0050   397 ;       MODE(AP) - ADDRESS INTERPRETATION MODE
                                0050   398 ;                  0 => PHYSICAL, 1 => VIRTUAL
                                0050   399 ;       RPB(AP) - ADDRESS OF RESTART PARAMETER BLOCK
                                0050   400 ;
                                0050   401 ; OUTPUT PARAMETERS:
                                0050   402 ;
                                0050   403 ;       R0 - COMPLETION STATUS CODE
                                0050   404 ;       R1 - TOTAL BYTES TRANSFERRED
                                0050   405 ;
                                0050   406 ;--
                                0050   407
                                0050   408 ;
                                0050   409 ; Offsets from AP to input arguments:
                                0050   410 ;
                                0050   411
                    00000004    0050   412          BUF     = 4
                    00000008    0050   413          SIZE    = 8
                    0000000C    0050   414          LBN     = 12
                    00000010    0050   415          FUNC    = 16
                    00000014    0050   416          MODE    = 20
                    00000018    0050   417          RPB     = 24
                                0050   418
                                0050   419 BOO$QIO::
                        0FFC    0050   420          .WORD   ^M<R2,R3,R4,R5,R6,R7,-  ; PRESERVE REGISTERS
                                0052   421                  R8,R9,R10,R11>
                                0052   422
DA AF   80000000 8F     D0      0052   423          MOVL    #UBA$M_MAP_VALID, -     ; Reset template                [23]
                                005A   424                  BOO$GL_UMR_TMPL         ;                               [23]
                                005A   425 ;
                                005A   426 ; If mapping is enabled, the processor register RP$_MAPEN contains a 1.
                                005A   427 ; Otherwise, the register contains a 0. Use this value as an index to
                                005A   428 ; choose the appropriate address of the adapter's register space.
                                005A   429 ;
                                005A   430
          59    18 AC     D0      005A   431          MOVL    RPB(AP),R9              ; GET BASE ADDRESS OF RESTART PARAMETER BLK
             51    38     DB      005E   432          MFPR    #PR$_MAPEN,R1           ; CHECK FOR MAPPING ENABLED
                                0061   433          ASSUME  RPB$L_ADPVIR EQ RPB$L_ADPPHY+4
       53    5C A941     D0      0061   434          MOVL    RPB$L_ADPPHY(R9)[R1],R3 ; GET CORRECT POINTER TO CONFIG REG
```

```
                                0066   435
                                0066   436 ;
                                0066   437 ; Using the argument list as input, calculate the transfer size, number
                                0066   438 ; of map registers, starting LBN, starting VPN, and base of a page table
                                0066   439 ; to use in mapping.
                                0066   440 ;
                                0066   441
          5A   04 AC    D0      0066   442          MOVL    BUF(AP),R10             ; Get buffer address
          58   08 AC    3C      006A   443          MOVZWL  SIZE(AP),R8             ; GET TRANSFER SIZE IN BYTES
                 04    12      006E   444          BNEQ    10$                     ; CONTINUE IF LEGAL SIZE
       58    01    10    9C    0070   445          ROTL    #16,#1,R8               ; ELSE FORCE TO 64K SIZE
    57 5A    09    00    EF    0074   446 10$:     EXTZV   #VA$V_BYTE,#VA$S_BYTE,R10,R7    ; Get byte offset into page
    57  03FF C847    9E      0079   447          MOVAB   ^X3FF(R8)[R7],R7        ; Calculate highest address plus
                                007F   448                                          ; an overflow page.
  57 57   F7 8F    78      007F   449          ASHL    #-9,R7,R7               ; Reduce to number of pages
                                0084   450                                          ; (= number of map registers).
          5B   0C AC    D0      0084   451          MOVL    LBN(AP),R11             ; AND BLOCK NUMBER FOR RETRY
          51   50 A9    D0      0088   452          MOVL    RPB$L_SVASPT(R9),R1     ; ASSUME SYSTEM SPACE
          03 5A    1F    E0     008C   453          BBS     #VA$V_SYSTEM,R10,20$    ; Branch if system address
             51    08    DB     0090   454          MFPR    #PR$_P0BR,R1            ; OTHERWISE GET P0 PT BASE
    52 5A    15    09    EF     0093   455 20$:     EXTZV   #VA$V_VPN,#VA$S_VPN,R10,R2      ; Get base VPN for transfer
                                0098   456 ;                                                                   begin   [022
                                0098   457 ; Now to determine if map registers are used, a check is made to see what
                                0098   458 ; device we are booting from.  Device types are set up as follows:
                                0098   459 ;        0 = a massbus device
                                0098   460 ;     1-31 = reserved for unibus devices
                                0098   461 ;       32 = HSC on a CI
                                0098   462 ;       33 = BDA
                                0098   463 ;       64 = Console
                                0098   464 ;
                                0098   465 ; All devices use map registers except the console device and the HSC.
                                0098   466 ;                                                                   end     [022
             66 A9    91      0098   467          CMPB    RPB$B_DEVTYP(R9),-      ; Booting from BDA?                     [022
                21            009B   468                  #BTD$K_BDA              ;                               [022
                  4F    12    009C   469          BNEQ    30$                     ; Branch if not a BDA                   [022
       08    98 AF    91      009E   470          cmpb    exe$gb_cputype,#pr$_sid_typuv2 ; Branch if MicroVax.    [24][27] ;G:
                  07    13    00A2   471          BEQL    22$                     ;                                       [24]
    00'8F    92 AF    91      00A4   472          cmpb    exe$gb_cputype,#pr$_sid_typrtuv2 ; Branch if RT MicroVax.       [24]
                  03    12    00A9   473          BNEQ    23$                     ;                                       [24]
                 00CC   31     00AB   474 22$:     BRW     PUSH_RETRY              ;                                       [24]
                                00AE   475 23$:                                                                 ;G:7
                                00AE   476 ;                                                                   begin   [022
                                00AE   477 ; Now set up the map registers for the BDA. There are 2 pages of map registers
                                00AE   478 ; allocated but we will only use 1 page.  The reason for allocating 2 pages
                                00AE   479 ; is to ensure that the page of map registers that we use are physically con-
                                00AE   480 ; tiguous.  So first we must find the page boundary of the map registers.
                                00AE   481 ; This will be the starting address of the map registers.
                                00AE   482 ;
 50   00000000'EF    9E      00AE   483          MOVAB   BDA_MAPREGS,R0          ; Get the start of the map registers    [022
 50   000001FF 8F    C0      00B5   484          ADDL    #^X1FF,R0               ; Determine the start of the map reg.   [022
 50   000001FF 8F    CA      00BC   485          BICL    #^X1FF,R0               ; on a page boundary                    [022
       85 AF    50    D0     00C3   486          MOVL    R0,EXE$GL_UMR_VIR       ; Save address of page aligned m.r.     [022
                                00C7   487
          54    38    DB      00C7   488          MFPR    #PR$_MAPEN,R4           ; Get the mapping status                [022
            17 54    E9      00CA   489          BLBC    R4,25$                  ; if map not enabled, branch around.    [022
          FF77 CF    D4      00CD   490          CLRL    EXE$GL_UMR_PHY          ; Initialize address of page aligned m.r.[02
50   50   15    09    EF     00D1   491          EXTZV   #VA$V_VPN,#VA$S_VPN,R0,R0 ; Convert VA to virtual page number   [022
```

```
                50    50 B940    D0  00D6    492          MOVL    @RPB$L_SVASPT(R9)[R0],R0 ; Get the physical page number          [022
                15    09   50    F0  00DB    493          INSV    R0,#VA$V_VPN,#VA$S_VPN,-;                                        [022
                      FF66 CF        00DF    494                  EXE$GL_UMR_PHY           ; Set physical address of map reg.      [022
                           07    11  00E2    495          BRB     27$                      ; Branch around                         [022
                                     00E4    496
                      FF64 CF    D0  00E4    497 25$:     MOVL    EXE$GL_UMR_VIR,-         ;                                       [022
                      FF5D CF        00E8    498                  EXE$GL_UMR_PHY           ; Save the physical addr. of m.r.       [022
                           09    11  00EB    499 27$:     BRB     INIT_MAPREGS             ; Go set up map registers               [022
                         66 A9    91  00ED    500 30$:     CMPB    RPB$B_DEVTYP(R9),-       ; If booting from console block         [022
                           20        00F0    501                  #BTD$K_HSCCI             ; storage device or CI,                 [DS0
                           03    19  00F1    502          BLSS    INIT_MAPREGS             ; don't load map registers              [022
                         0084    31  00F3    503          BRW     PUSH_RETRY               ;                                       [022
                                     00F6    504
                                     00F6    505 ;
                                     00F6    506 ; Register usage right now is as follows:
                                     00F6    507 ;
                                     00F6    508 ;       R1      - address of page table for virtual-->physical mapping
                                     00F6    509 ;       R2      - base VPN for the transfer
                                     00F6    510 ;       R3      - address of the adapter's configuration register
                                     00F6    511 ;       R7      - number of map registers needed (plus one extra)
                                     00F6    512 ;       R8      - transfer size in bytes
                                     00F6    513 ;       R9      - address of the RPB
                                     00F6    514 ;       R10     - buffer address
                                     00F6    515 ;       R11     - starting LBN of the transfer
                                     00F6    516 ;
                                     00F6    517 ; In an adapter-dependent fashion, initialize the required number of
                                     00F6    518 ; adapter map registers. First calculate the address of the starting map
                                     00F6    519 ; register number. Right now, map registers for all UNIBUS and MASSBUS
                                     00F6    520 ; adapters for all processors start at the same offset from the base of
                                     00F6    521 ; the adapter's register space.
                                     00F6    522 ;
                                     00F6    523 ; During map register initialization, the following registers change
                                     00F6    524 ; for each page mapped:
                                     00F6    525 ;
                                     00F6    526 ;       R2      - address of the next VPN to map
                                     00F6    527 ;       R4      - address of the next map register to load
                                     00F6    528 ;       R5      - PFN of the page being mapped
                                     00F6    529 ;
                                     00F6    530
                                     00F6    531 INIT_MAPREGS:                             ; Initialize the map registers.
                                     00F6    532 ; This assumes that UNIT_INIT routine of BDA initializes EXE$GL_UMR_PHY field    [022
                                     00F6    533 ; and it also initializes EXE$GL_UMR_VIR.                                        [022
                         66 A9    91  00F6    534          CMPB    RPB$B_DEVTYP(R9),-       ; Booting from BDA?                     [022
                           21        00F9    535                  #BTD$K_BDA               ;                                       [022
                           0B    12  00FA    536          BNEQ    10$                      ; Branch if not a BDA                   [022
                       54  38    DB  00FC    537          MFPR    #PR$_MAPEN,R4            ; Check for mapping enabled             [022
                                     00FF    538          ASSUME  EXE$GL_UMR_VIR  EQ  EXE$GL_UMR_PHY+4                 ;       [022
                54   FF44 CF44    D0  00FF    539          MOVL    EXE$GL_UMR_PHY[R4],R4    ; Get either physical or virtual address[022
                                     0105    540                                           ; pseudo map registers                  [022
                           0B    11  0105    541          BRB     COMPUTE_PFN              ;                                       [022
                                     0107    542 10$:     ASSUME  MBA$L_MAP EQ UBA$L_MAP
                                     0107    543          ASSUME  MBA$L_MAP EQ UBI$L_MAP
                  54   FF19 CF    D0  0107    544          MOVL    W^BOO$GL_UMR_DIS,R4      ; Pick up number of disable UMR's       [022
                54   0800 C344    DE  010C    545          MOVAL   MBA$L_MAP(R3)[R4],R4     ; Point to first useable map register   [022
                                     0112    546
                                     0112    547 COMPUTE_PFN:                              ; Loop once per page.
                         55   82    9E  0112    548          MOVAB   (R2)+,R5                 ; Get a virtual page number.
```

```
                 0B 14 AC    E9  0115   549          BLBC    MODE(AP),10$            ; If physical page #, branch.
                    55  6145    D0  0119   550          MOVL    (R1)[R5],R5             ; Get page table entry
              55  FFE00000 8F    CA  011D   551          BICL    #^C<PTE$M_PFN>,R5       ; and extract PFN from entry
                                    0124   552 ;                                                                    begin   [022
                                    0124   553 ; Derive the boot device adapter's type (UNIBUS adapter or MASSBUS
                                    0124   554 ; adapter) from the RPB, and save a flag indicating the adapter type
                                    0124   555 ; in a register.  The seemingly complicated fooling around with both
                                    0124   556 ; a UMR_DP and UMR_TMPL is to allow flexibility in what devices desire
                                    0124   557 ; in the way of data paths: Only UNIBUS devices will ever even pick up
                                    0124   558 ; the UMR_DP bit.  Thus all non-UNIBUS boot devices will never purge
                                    0124   559 ; a data path.  UNIBUS devices have a choice: by clearing UMR_DP in
                                    0124   560 ; their UNIT_INIT routines, the boot drivers can elect to not use
                                    0124   561 ; the buffered data path.
                                    0124   562 ;                                                                    end     [022
                                    0124   563
                       50    D4   0124   564 10$:    CLRL    R0                      ; Assume not UNIBUS
                    56  63    D0   0126   565          MOVL    (R3),R6                 ; Fetch nexus type of adapter
                 56  0102 8F    B1   0129   566          CMPW    #<NDT$_BUA&^xFFFF>,R6   ; See if a BAckplane-UNIBUS adapter. NOTE
                                    012E   567                                          ;  anding with ^xFFFF is to avoid
                                    012E   568                                          ;  truncation warning from assembler.
                       0F    13   012E   569          BEQL    15$                     ; EQL implies BAckplane-UNIBUS adapter.
                 56  0103 8F    B1   0130   570          CMPW    #<NDT$_BLA&^xFFFF>,R6   ; See if a BAckplane-LESI adapter. NOTE
                                    0135   571                                          ; anding with ^xFFFF is to avoid
                                    0135   572                                          ; truncation warning from assembler.
                       11    13   0135   573          BEQL    16$                     ; EQL implies BAckplane-LESI adapter.
                    56  03    CA   0137   574          BICL    #3,R6                   ; Make canonical adapter type
                    28  56    B1   013A   575          CMPW    R6, #NDT$_UBO           ; See if boot adapter is a UNIBUS,
                       11    12   013D   576          BNEQ    20$                     ; NEQ implies NOT.
                                    013F   577 15$:                                   ; If yes, UNIBUS adapter then
                 FEF5 CF    F0   013F   578          INSV    BOO$GB_UMR_DP,-         ;   Pick up the data path
                 02  15          0143   579                  #UBA$V_MAP_DPD,#2,-
                 FEEC CF        0145   580                  BOO$GL_UMR_TMPL         ;   and put it in the template
                       50    D6   0148   581 16$:    INCL    R0                      ;   Set a flag for later user           end
                                    014A   582 ;
                                    014A   583 ; This is a UNIBUS adapter.
                                    014A   584 ;
                                    014A   585 ; Map registers for the UNIBUS adapter look like the following:
                                    014A   586 ;
                                    014A   587 ;       +-----------------------------------------------+
                                    014A   588 ;       | V |   | BO | DP # |  |  page frame number  |
                                    014A   589 ;       +-----------------------------------------------+
                                    014A   590 ;
                                    014A   591 ; The code sets the byte offset bit if relevant, sets the valid bit,
                                    014A   592 ; sets the low bit in the 4-bit data path field to indicate that the
                                    014A   593 ; first buffered data path is to be used, and loads the page frame
                                    014A   594 ; number into the low bits.
                                    014A   595 ;
                                    014A   596
                    19  04 AC    F0   014A   597          INSV    BUF(AP),#UBA$V_MAP_BO,- ; Set UBA byte offset bit if
                    55  01          014E   598                  #1,R5                   ; necessary.
              84  FEDF CF    55   C9   0150   599 20$:    BISL3   R5,BOO$GL_UMR_TMPL,(R4)+; Set PFN and byte offset, valid      [022
                                    0156   600                                          ; bit, and buffered DP number
                                    0156   601                                          ; into map.
                                    0156   602
                                    0156   603 ; ****** NOTE *****  Always uses Datapath #1, IOC$PURDPR depends on this!!
                                    0156   604
                                    0156   605 ;
```

```
                     0156   606 ; MASSBUS adapter's map registers look like the following:
                     0156   607 ;
                     0156   608 ;       +-----------------------------------------+
                     0156   609 ;       | V |                 | page frame number |
                     0156   610 ;       +-----------------------------------------+
                     0156   611 ;
                     0156   612 ;
                     0156   613 ;*****
                     0156   614 ;       BISL3   R5,#MBA$M_MAP_VALID,-   ; Set the PFN and the valid bit
                     0156   615 ;               (R4)+                   ; in the map register.
                     0156   616 ;*****
                     0156   617 ;
                     0156   618
                     0156   619                                         ; Is there another page to do?          [022
          57   D7    0156   620         DECL    R7                      ; Decrement # of map registers.
          B8   14    0158   621         BGTR    COMPUTE_PFN             ; Loop to fill next map register
                     015A   622                                         ; if byte count not exhausted.
                     015A   623
                     015A   624 ;
                     015A   625 ; All map registers are set up. Set up 2 more inputs to driver code.
                     015A   626 ; Since the loaded MBA map registers are registers #0-n, the starting
                     015A   627 ; address of the transfer to be loaded into a device register by the
                     015A   628 ; device driver, is now simply the offset into the first page of the
                     015A   629 ; transfer buffer.  However, for the UBA, there may be some disabled
                     015A   630 ; map registers as a result of UNIBUS memory on the UBA.  Therefor,
                     015A   631 ; the starting address of the transfer must include the lowest enabled
                     015A   632 ; UMR in bits 9-17.
                     015A   633
          09   00   EF  015A   634         EXTZV   #VA$V_BYTE,#VA$S_BYTE,- ; Get byte offset into page
          5A   5A       015D   635                 R10,R10                 ; in R10
                     015F   636
                     015F   637 ;
                     015F   638 ; Invalidate the last UBA map register so that a wild transfer will stop
                     015F   639 ; at the end of the last valid block.
                     015F   640 ;
                     015F   641
        10 50   E9   015F   642         BLBC    R0,30$                  ; Skip invalidation for MBA.
 74 80000000 8F   CA  0162   643         BICL    #UBA$M_MAP_VALID,-(R4)  ; Invalidate last map register.
50 FEB6 CF   09   9C  0169   644         ROTL    #9,W^BOO$GL_UMR_DIS,R0  ; Form UNIBUS address of UMR            [022
       5A   50   C8  016F   645         BISL    R0,R10                  ; Set into address register             [022
                     0172   646 ;
                     0172   647 ; If it is a UNIBUS boot device, derive the address of the device's CSR.
                     0172   648 ;
                     0172   649
                     0172   650         ASSUME  RPB$L_CSRVIR  EQ  RPB$L_CSRPHY+4
                     0172   651
       50   38   DB  0172   652 30$:    MFPR    #PR$_MAPEN,R0           ; Check for mapping enabled
    57 54 A940   D0  0175   653         MOVL    RPB$L_CSRPHY(R9)[R0],R7 ; Get address of device's CSR
                     017A   654
                     017A   655 PUSH_RETRY:
            0A   DD  017A   656         PUSHL   #10                     ; Push retry count on stack
                     017C   657
       55   5B   D0  017C   658 10$:    MOVL    R11,R5                  ; Get a working copy of the block number
     50   34 A9   D0  017F   659         MOVL    RPB$L_IOVEC(R9),R0      ; Get address of boot vectors
        08 B040   16  0183   660         JSB     @8(R0)[R0]              ; Call driver thru self-relative vector
              15   E1  0187   661         BBC     #UBA$V_MAP_DPD,-        ; Branch if not using the Buffered      [022
  14 FEA8 CF        0189   662                 BOO$GL_UMR_TMPL,100$    ; data path                             [022
```

```
                      50    DD   018D   663          PUSHL   R0                      ; Save driver status
                      02    BB   018F   664          PUSHR   #^M<R1>                 ; Save R1 from driver
                    006C    30   0191   665          BSBW    BOO$PURDPR              ; Purge Buffered Datapath for UBA
                      02    BA   0194   666          POPR    #^M<R1>                 ; Restore R1
                   05 50    E8   0196   667          BLBS    R0,80$                  ; Branch if success
                5E    04    C0   0199   668          ADDL    #4,SP                   ; Clear previous status from stack
                      06    11   019C   669          BRB     150$                    ; Retry
                      50  8ED0   019E   670 80$:      POPL    R0                      ; Get driver status back
                   03 50    E8   01A1   671 100$:     BLBS    R0,200$                 ; Branch if success
                   D5 6E    F5   01A4   672 150$:     SOBGTR  (SP),10$                ; Retry if count > 0
                             04   01A7   673 200$:     RET                             ; Return with final status in R0
```

```
                                01A8   675          .SBTTL  BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QIO
                                01A8   676
                                01A8   677 ;++
                                01A8   678 ; FUNCTIONAL DESCRIPTION:
                                01A8   679 ;       BOO$MAP IS CALLED TO INITIALIZE THE DATA BASE FOR BOO$QIO TO PERMIT
                                01A8   680 ;       IT TO FUNCTION WITH MEMORY MANAGEMENT ENABLED.  AN AREA OF SYSTEM
                                01A8   681 ;       PAGE TABLE MUST BE PROVIDED SO THAT THE CONFIGURATION REGISTERS AND
                                01A8   682 ;       UNIBUS I/O PAGE CAN BE MAPPED.
                                01A8   683 ;
                                01A8   684 ; CALLING SEQUENCE:
                                01A8   685 ;       CALLG   ARGLIST,BOO$MAP
                                01A8   686 ;
                                01A8   687 ; INPUT PARAMETERS:
                                01A8   688 ;       SVASPT(AP) - SYSTEM VIRTUAL ADDRESS OF THE SYSTEM PAGE TABLE
                      00000004  01A8   689          SVASPT  =    4
                                01A8   690 ;       VABASE(AP) - BASE VIRTUAL ADDRESS OF A 24 PAGE WINDOW TO MAP
                                01A8   691 ;                    THE ADAPTER CONFIGURATION REGISTERS AND UNIBUS
                      00000008  01A8   692          VABASE  =    8
                                01A8   693 ;                    I/O PAGE.
                                01A8   694 ;       RPB(AP) -    ADDRESS OF RESTART PARAMETER BLOCK (RPB) CONTAINING
                                01A8   695 ;                    BOOTSTRAP DEVICE DESCRIPTION.
                      0000000C  01A8   696          RPB     =    12
                                01A8   697 ;
                                01A8   698 ; OUTPUT PARAMETERS:
                                01A8   699 ;       NONE
                                01A8   700 ;
                                01A8   701 ;--
                                01A8   702
                          00FC  01A8   703 BOO$MAP::      .WORD   ^M<R2,R3,R4,R5,R6,R7>   ;
              57    0C AC     D0  01AA   704          MOVL    RPB(AP),R7              ; GET BASE ADDRESS FOR RPB
              52    04 AC     D0  01AE   705          MOVL    SVASPT(AP),R2           ; GET BASE OF SPT
              50 A7    52     D0  01B2   706          MOVL    R2,RPB$L_SVASPT(R7)     ; AND SAVE IN DATA BASE
              53    08 AC     D0  01B6   707          MOVL    VABASE(AP),R3           ; GET VIRTUAL ADDRESS OF WINDOW
              60 A7    53     D0  01BA   708          MOVL    R3,RPB$L_ADPVIR(R7)     ; SET AS ADAPTER VIRTUAL ADDRESS
     54   5C A7    15    09     EF  01BE   709          EXTZV   #VA$V_VPN,#VA$S_VPN,RPB$L_ADPPHY(R7),R4 ; GET BASE PFN
                   55    08     D0  01C4   710          MOVL    #8,R5                   ; SET TO MAP 8 PAGES
        50   53    15    09     EF  01C7   711          EXTZV   #VA$V_VPN,#VA$S_VPN,R3,R0       ; GET BASE VIRTUAL PAGE
                  51    6240    DE  01CC   712          MOVAL   (R2)[R0],R1             ; COMPUTE WORKING SPT POINTER
                        20    10  01D0   713          BSBB    FILLSPT                 ; FILL SPT TO MAP CONFIGURATION REGS
                   55    10     D0  01D2   714          MOVL    #16,R5                  ; SET FOR 16 PAGES
     54   54 A7    00001FFF 8F    CB  01D5   715          BICL3   #^X1FFF,RPB$L_CSRPHY(R7),R4     ; GET PHY ADDR OF I/O PAGE BASE
              54    54    17     9C  01DE   716          ROTL    #<32-9>,R4,R4           ; AND CONVERT TO PAGE NUMBER
                        0E    10  01E2   717          BSBB    FILLSPT                 ; STORE PTES INTO SPT
                50    54 A7    3C  01E4   718          MOVZWL  RPB$L_CSRPHY(R7),R0     ; GET I/O PAGE OFFSET
     58 A7    FFFF3000 E043    9E  01E8   719          MOVAB   <^X1000-^XE000>(R0)[R3],RPB$L_CSRVIR(R7) ; SET VIRTUAL CSR ADDR
                        04    01F1   720          RET                             ;
                              01F2   721
                              01F2   722 ;++
                              01F2   723 ; FILLSPT
                              01F2   724 ;
                              01F2   725 ; INPUTS:
                              01F2   726 ;       R1 - POINTER TO CURRENT SPT ENTRY (UPDATED)
                              01F2   727 ;       R4 - PFN (UPDATED)
                              01F2   728 ;       R5 - COUNT OF PAGES TO FILL (UPDATED)
                              01F2   729 ;
                              01F2   730 FILLSPT:                             ;
    81    54    90000000 8F    C9  01F2   731          BISL3   #<PTE$M_VALID!PTE$C_KW>,R4,(R1)+ ; STORE A PTE
```

```
                54   D6  01FA   732          INCL    R4                      ; ADVANCE TO NEXT PFN
             F3 55   F5  01FC   733          SOBGTR  R5,FILLSPT              ; STORE THEM ALL
                     05  01FF   734          RSB
```

```
                        0200    736          .SBTTL  BOO$PURDPR - Purge UBA Buffered Datapath
                        0200    737
                        0200    738 ;++
                        0200    739 ; FUNCTIONAL DESCRIPTION:
                        0200    740 ;
                        0200    741 ;       This routine is called by BOOTDRIVR at the end of each boot device
                        0200    742 ;       transfer if the boot device is on the Unibus.  It purges the buffered
                        0200    743 ;       datapath and/or performs other Unibus adapter specific end-action.
                        0200    744 ;
                        0200    745 ;   NOTE: This routine contains processor specific code.
                        0200    746 ;
                        0200    747 ; CALLING SEQUENCE:
                        0200    748 ;
                        0200    749 ;       JSB     BOO$PURDPR
                        0200    750 ;
                        0200    751 ; INPUT PARAMETERS:
                        0200    752 ;
                        0200    753 ;       R3 - Address of UBA adapter configuration register
                        0200    754 ;       EXE$GB_CPUTYPE - Index specifying what CPU we are executing on
                        0200    755 ;       ** Assumes all drivers use DATAPATH 1 **
                        0200    756 ;
                        0200    757 ; OUTPUT PARAMETERS:
                        0200    758 ;
                        0200    759 ;       R0 - LBS -> Success
                        0200    760 ;            LBC -> Failure
                        0200    761 ;
                        0200    762 ;       R1,R2,R4 - Destroyed
                        0200    763 ;       All other registers preserved
                        0200    764 ;
                        0200    765 ;--
                        0200    766
                        0200    767 BOO$PURDPR:
                        0200    768
            50   01  3C 0200    769          MOVZWL  #SS$_NORMAL,R0              ; Assume success
                        0203    770          CPUDISP <100$,200$,300$,100$,400$,400$,100$,170$,-                  ;G:7
                        0203    771                  170$,-                      ; Not assigned                  [28]    ;G:
                        0203    772                  170$,-                      ; Not assigned                  [28]    ;G:
                        0203    773                  170$,-                      ; Not assigned                  [28]    ;G:
                        0203    774                  170$,-                      ; Not assigned                  [28]    ;G:
                        0203    775                  170$,-                      ; Not assigned                  [28]    ;G:
                        0203    776                  170$,-                      ; Not assigned                  [28]    ;G:
                        0203    777                  170$,-                      ; Not assigned                  [28]    ;G:
                        0203    778                  170$>                       ; Dispatch on EXE$GB_CPUTYPE    [218][24][28
0F'  01  FE32 CF   8F   0203          CASEB   W^EXE$GB_CPUTYPE,#PR$_SID_TYP780,S^#<<30001$-30000$>/2>-1
                        0209      30000$:
                0020'   0209          .SIGNED_WORD    100$-30000$
                0039'   020B          .SIGNED_WORD    200$-30000$
                0059'   020D          .SIGNED_WORD    300$-30000$
                0020'   020F          .SIGNED_WORD    100$-30000$
                0068'   0211          .SIGNED_WORD    400$-30000$
                0068'   0213          .SIGNED_WORD    400$-30000$
                0020'   0215          .SIGNED_WORD    100$-30000$
                0038'   0217          .SIGNED_WORD    170$-30000$
                0038'   0219          .SIGNED_WORD    170$-30000$
                0038'   021B          .SIGNED_WORD    170$-30000$
                0038'   021D          .SIGNED_WORD    170$-30000$
                0038'   021F          .SIGNED_WORD    170$-30000$
```

```
                    0038' 0221                .SIGNED_WORD    170$-30000$
                    0038' 0223                .SIGNED_WORD    170$-30000$
                    0038' 0225                .SIGNED_WORD    170$-30000$
                    0038' 0227                .SIGNED_WORD    170$-30000$
                          0229        30001$:
                          0229   779
      52    44 A3    DE   0229   780 100$:   MOVAL   UBA$L_DPR+4(R3),R2        ; CPU type 11/780, Datapth Register
   62    01    1F    78   022D   781         ASHL    #UBA$V_DPR_BNE,#1,(R2)    ; Purge datapath
         51    62    D0   0231   782         MOVL    (R2),R1                   ; Get Datapth register contents
      09 51    1E    E1   0234   783         BBC     #UBA$V_DPR_XMTER,R1,170$  ; Branch if no error
   62    01    1E    78   0238   784         ASHL    #UBA$V_DPR_XMTER,#1,(R2)  ; Clear error in datapath
   50    01F4 8F    3C   023C   785 150$:   MOVZWL  #SS$_PARITY,R0            ; Set failure status
                    05   0241   786 170$:   RSB                               ; Return to caller
                         0242   787
      52    04 A3    DE   0242   788 200$:   MOVAL   UBI$L_DPR+4(R3),R2        ; CPU type 11/750, Datapath Register
   62    01    00    78   0246   789         ASHL    #UBI$V_DPR_PUR,#1,(R2)    ; Purge Datapath
         54    0A    D0   024A   790         MOVL    #UBI$C_PURCNT,R4          ; Get max # of tries for
                         024D   791                                           ; purge done test
         51    62    D0   024D   792 230$:   MOVL    (R2),R1                   ; Get datapath register contents
      05 51    00    E1   0250   793         BBC     #UBI$V_DPR_PUR,R1,250$    ; Branch if purge done
            F6 54    F5   0254   794         SOBGTR  R4,230$                   ; Branch if more tries allowed
               04    11   0257   795         BRB     270$                      ; Return failure status
      E4 51    1F    E1   0259   796 250$:   BBC     #UBI$V_DPR_ERROR,R1,170$  ; Branch if no purge error
         62    00    D2   025D   797 270$:   MCOML   #0,(R2)                   ; Clear datapath error(s)
               DA    11   0260   798         BRB     150$                      ; Return with failure status
                         0262   799
      51    10 A3    D0   0262   800 300$:   MOVL    UBI$L_SR(R3),R1           ; Get Unibus Error Summary Register
                         0266   801                                           ; Nebula
51    8001C000 8F    D3   0266   802         BITL    #<UBI$M_SR_UWE!-          ; Any UB errors? (write error,
                         026D   803                 UBI$M_SR_MRPE!-           ;       map parity error,
                         026D   804                 UBI$M_SR_NXM!-            ;       non-existent memory,
                         026D   805                 UBI$M_SR_UCE>,R1          ; or uncorrected read error.)
               D2    13   026D   806         BEQL    170$                      ; Branch if no errors
                         026F   807 ; ****** QUESTION - Is there anything to do to clear the error status?
               CB    11   026F   808         BRB     150$                      ; Return failure status
                         0271   809
   0754 C3    01    D0   0271   810 400$:   MOVL    #BUA$M_PURGE, -           ; Purge datapath BDP #1                [219
                         0276   811                 BUA$L_DPCSR1(R3)
   51    0720 C3    D0   0276   812         MOVL    BUA$L_CSR(R3),R1          ; Copy BUA CSR register to R1          [219
      C2 51    1F    E1   027B   813         BBC     #BUA$V_ERR,R1,170$        ; If no errors, branch to RSB          [219
                    CA   027F   814         BICL    #<BUA$M_BIF!-             ; Clear error bits, BAckplane failure bit an
                         0280   815                 BUA$M_USSTO!-             ; UNIBUS SSYN Time Out bit and         [219
                         0280   816                 BUA$M_UIE!-               ; UNIBUS Interlock Error bit and       [219
                         0280   817                 BUA$M_IMR!-               ; Invalid Map Register and             [219
                         0280   818                 BUA$M_BADBDP>,-           ; Bad Buffered Data Path in            [219
51    1F000000 8F        0280   819                 R1                        ; BUA CSR copy                         [219
   0720 C3    51    D0   0286   820         MOVL    R1,BUA$L_CSR(R3)          ; Update BUA CSR register.             [219
               AF    11   028B   821         BRB     150$                      ;                                      [218
```

```
                    0000028D  028D    823 BOO$QIOSIZ=.-BOO$AL_VECTOR              ; Size of boot QIO routine
                              028D    824
                    0000028D  028D    825 BOO$DRIVER==.                           ; Start of boot driver (after
                              028D    826                                         ; it's been moved)
                              028D    827                                         ; NOTE:  Boot drivers must be in
                              028D    828                                         ; psect BOOTDRIVR_2
                              028D    829
                          00000000    830         .PSECT  BOOTDRIVR_3
                              0000    831
                    00000000  0000    832 BOO$DRIVER_TBL=.                        ; Boot driver table
                              0000    833
                          00000000    834         .PSECT  BOOTDRIVR_5
                              0000    835
                    00000000  0000    836         .LONG   0                       ; End of boot driver table
                              0004    837
                          00000000    838         .PSECT  BOOTDRIVR_6
```

```
                        0000    840          .SBTTL  BOO$SELECT - Select boot driver
                        0000    841
                        0000    842 ;++
                        0000    843 ; FUNCTIONAL DESCRIPTION:
                        0000    844 ;
                        0000    845 ;       This routine is called the first time BOO$QIO calls a driver.
                        0000    846 ;       It searches the boot driver table to locate the proper driver.
                        0000    847 ;       This driver is then moved so that it is adjacent to BOO$QIO
                        0000    848 ;       and the correct linkage is made in BOO$AL_VECTOR.
                        0000    849 ;       RPB$L_IOVECSZ is also stored with the size of BOO$QIO plus
                        0000    850 ;       the size of the driver.  The driver is then jumped to.
                        0000    851 ;
                        0000    852 ; CALLING SEQUENCE:
                        0000    853 ;
                        0000    854 ;       JSB     BOO$SELECT      (Actually called through self-relative
                        0000    855 ;                                vector in BOO$AL_VECTOR+8)
                        0000    856 ;
                        0000    857 ; INPUT PARAMETERS:
                        0000    858 ;
                        0000    859 ;       R9      Address of the RPB
                        0000    860 ;
                        0000    861 ; OUTPUT PARAMETERS:
                        0000    862 ;
                        0000    863 ;       None
                        0000    864 ;
                        0000    865 ;--
                        0000    866
                        0000    867 BOO$RESELECT::                              ;                               [0220]
                        0000    868 BOO$SELECT::                                ; Global so DS can re-init BOO$AL_VECTOR
            3E   BB     0000    869          PUSHR   #^M<R1,R2,R3,R4,R5>        ; Save registers
     55 0000'CF   DE     0002    870          MOVAL   W^BOO$DRIVER_TBL,R5        ; Get address of boot driver table
       53   66 A9   9A 0007    871          MOVZBL  RPB$B_DEVTYP(R9),R3        ; Get value of boot device type
     54 0039'CF   9A     000B    872          MOVZBL  W^EXE$GB_CPUTYPE,R4        ; Get cpu type
                        0010    873
                        0010    874 10$:     ;
                        0010    875          ; Determine if next driver in table is the correct one.
                        0010    876          ;
                        0010    877
        50   65   32     0010    878          CVTWL   BDT$L_CPUTYPE(R5),R0       ; Get cpu type from table
             53   13     0013    879          BEQL    80$                        ; End of table
             05   19     0015    880          BLSS    20$                        ; Driver doesn't care about cpu type
        54   50   D1     0017    881          CMPL    R0,R4                      ; Cpu types match?
             17   12     001A    882          BNEQ    40$                        ; No, try next driver
                        001C    883
     50   02 A5   32     001C    884 20$:     CVTWL   BDT$L_DEVTYPE(R5),R0       ; Get boot device type from table
             05   19     0020    885          BLSS    30$                        ; Driver doesn't care about device type
        53   50   D1     0022    886          CMPL    R0,R3                      ; Device types match?
             0C   12     0025    887          BNEQ    40$                        ; No, try next driver
                        0027    888
     50   04 A5   D0     0027    889 30$:     MOVL    BDT$L_ACTION(R5),R0        ; Get action routine offset from table
             0B   13     002B    890          BEQL    60$                        ; No action routine, this is the driver
           6540   16     002D    891          JSB     (R5)[R0]                   ; Call action routine
          05 50   E8     0030    892          BLBS    R0,60$                     ; Branch if this is the driver
        55   18   C0     0033    893 40$:     ADDL    #BDT$K_LENGTH,R5           ; Point to next driver entry
             D8   11     0036    894          BRB     10$                        ; Try next driver
                        0038    895
                        0038    896 60$:     ;
```

```
                                0038   897          ; Have the right driver.  R5 points to driver table entry.
                                0038   898          ; Set up size in RPB, driver vector, and then move driver.
                                0038   899          ; Finally, jump to driver.
                                0038   900
          0000028D 8F   C1      0038   901          ADDL3   #BOO$QIOSIZ,-                       ; Add boot QIO size to
        38 A9   08 A5           003E   902                  BDT$L_SIZE(R5),RPB$L_IOVECSZ(R9); driver size and store in RPB
    50   55   00000000'8F   C3  0042   903          SUBL3   #BOO$AL_VECTOR, R5, R0  ; Offset from vector to header
     0008'CF   10 A5   50   C1  004A   904          ADDL3   R0,-
                                0051   905                  BDT$L_ENTRY(R5),W^BOO$AL_VECTOR+8 ; entry and store in vector
         0008'CF   0C A5   C0  0051   906          ADDL2   BDT$L_ADDR(R5),W^BOO$AL_VECTOR+8 ; Plus offset from header
     000C'CF   14 A5   50   C1  0057   907          ADDL3   R0,-                    ; Compute offset to driver
                                005E   908                  BDT$L_DRIVRNAME(R5),W^BOO$AL_VECTOR+12 ; name and store in vector
                                005E   909 ; Do not move the driver in the supervisor
                                005E   910 ;         MOVC3   BDT$L_SIZE(R5),-                ; Move driver
                                005E   911 ;                 @BDT$L_ADDR(R5)[R5],W^BOO$DRIVER
                      3E   BA   005E   912          POPR    #^M<R1,R2,R3,R4,R5>             ; Restore registers
                50   34 A9   D0  0060   913          MOVL    RPB$L_IOVEC(R9),R0              ; Get address of vectors
                   08 B040   17  0064   914          JMP     @8(R0)[R0]                      ; Jump to driver
                                0068   915
                                0068   916
                                0068   917 80$:      ;
                                0068   918          ; No driver in the driver table accepted this QIO.
                                0068   919          ;
                                0068   920
          50   00660158 8F   D0  0068   921          MOVL    #DS$_UNSUPPDEV,R0               ; The load device is not supported i
                                05  006F   922          RSB
```

```
                0070          .SBTTL  BOO$MOVE - Select and move boot driver
                0070
                0070    ;++
                0070    ; FUNCTIONAL DESCRIPTION:
                0070    ;
                0070    ;       Not implemented for diagnostic supervisor use.
                0070    ;--
                0070
                0070    BOO$MOVE:                                       ;                       [202]
          00    0070          HALT                                    ;                       [202]
                0071          .END
```

```
BDA_MAPREGS                     ********W GX   00
BDT$K_LENGTH                  = 00000018     D
BDT$L_ACTION                    00000004     D
BDT$L_ADDR                      0000000C     D
BDT$L_CPUTYPE                   00000000     D
BDT$L_DEVTYPE                   00000002     D
BDT$L_DRIVRNAME                 00000014     D
BDT$L_ENTRY                     00000010     D
BDT$L_SIZE                      00000008     D
BIT...                        = 00660160     D
BOO$AL_VECTOR                   00000000 RG D  02
BOO$DRIVER                    = 0000028D RG D  02
BOO$DRIVER_TBL                = 00000000 R  D  03
BOO$GB_UMR_DP                   00000038 RG D  02
BOO$GL_UCODE                    00000028 RG D  02
BOO$GL_UMR_DIS                  00000024 RG D  02
BOO$GL_UMR_TMPL                 00000034 RG D  02
BOO$MAP                         000001A8 RG D  02
BOO$MOVE                        00000070 R  D  05
BOO$PURDPR                      00000200 R  D  02
BOO$QIO                         00000050 RG D  02
BOO$QIOSIZ                    = 0000028D     D
BOO$RESELECT                    00000000 RG D  05
BOO$SELECT                      00000000 RG D  05
BQO$B_CPUTYPE                 = 00000039     D
BQO$B_UMR_DP                  = 00000038     D
BQO$L_AUXDRNAME               = 00000020     D
BQO$L_CPUDATA                 = 0000003A     D
BQO$L_DEVNAME                 = 00000030     D
BQO$L_TENUSEC                 = 0000003E     D
BQO$L_UBDELAY                 = 00000042     D
BQO$L_UCODE                   = 00000028     D
BQO$L_UMR_DIS                 = 00000024     D
BQO$L_UMR_TMPL                = 00000034     D
BQO$L_UNIT_DISC               = 0000002C     D
BQO$L_UNIT_INIT               = 0000001C     D
BQO$W_VERSION                 = 00000010     D
BTD$K_BDA                     = 00000021     D
BTD$K_HSCCI                   = 00000020     D
BUA$L_CSR                     = 00000720     D
BUA$L_DPCSR1                  = 00000754     D
BUA$M_BADBDP                  = 01000000     D
BUA$M_BIF                     = 10000000     D
BUA$M_IMR                     = 02000000     D
BUA$M_PURGE                   = 00000001     D
BUA$M_UIE                     = 04000000     D
BUA$M_USSTO                   = 08000000     D
BUA$V_ERR                     = 0000001F     D
BUF                           = 00000004     D
COMPUTE_PFN                     00000112 R   D  02
DS$K_ERROR                    = 00000002     D
DS$K_NORMAL                   = 00000001     D
DS$K_SEVERE                   = 00000004     D
DS$K_SUBSYS                   = 00000066     D
DS$K_WARNING                  = 00000000     D
DS$_AP_NORMAL_BREAK           = 00660150     D
DS$_ARITH                     = 006600D0     D
DS$_ASBE                      = 00660118     D
DS$_BADLINK                   = 006600F0     D
DS$_BADTYPE                   = 006600E8     D
DS$_BIIC                      = 00660120     D
DS$_CHME                      = 006600A8     D
DS$_CHMK                      = 006600E0     D
DS$_DEVNAME                   = 00660108     D
DS$_ERROR                     = 00660002     D
DS$_FHWE                      = 00660068     D
DS$_FRAGBUF                   = 00660080     D
DS$_ICBUSY                    = 006600C8     D
DS$_ICERR                     = 006600C0     D
DS$_IHWE                      = 00660060     D
DS$_ILLCHAR                   = 00660018     D
DS$_ILLPAGCNT                 = 00660078     D
DS$_ILLUNIT                   = 00660100     D
DS$_INIT_FAIL                 = 00660140     D
DS$_INSFMEM                   = 00660050     D
DS$_INVCPU                    = 00660130     D
DS$_IPL2HI                    = 006600B8     D
DS$_IVADDR                    = 00660040     D
DS$_IVVECT                    = 00660038     D
DS$_KRNLSTK                   = 00660090     D
DS$_LOGIC                     = 00660070     D
DS$_MCHK                      = 00660088     D
DS$_MEM_ALLOC_ERR             = 00660138     D
DS$_MMOFF                     = 00660058     D
DS$_NEEDUNIT                  = 006600F8     D
DS$_NODE                      = 00660128     D
DS$_NOPCS                     = 00660110     D
DS$_NORMAL                    = 00660001     D
DS$_NOSUPPORT                 = 006600B1     D
DS$_NOTALLOWMP                = 00660148     D
DS$_NOTDON                    = 00660030     D
DS$_NOTIMP                    = 006600B0     D
DS$_NULLSTR                   = 00660010     D
DS$_OVERFLOW                  = 00660008     D
DS$_POWER                     = 00660098     D
DS$_PROGERR                   = 00660020     D
DS$_SEVERE                    = 00660004     D
DS$_TRANSL                    = 006600A0     D
DS$_TRUNCATE                  = 00660028     D
DS$_UNEXPINT                  = 006600D8     D
DS$_UNSUPPDEV                 = 00660158     D
DS$_VASFULL                   = 00660048     D
DS$_WARNING                   = 00660000     D
EXE$GB_CPUDATA                  0000003A RG D  02
EXE$GB_CPUTYPE                  00000039 RG D  02
EXE$GL_TENUSEC                  0000003E RG D  02
EXE$GL_UBDELAY                  00000042 RG D  02
EXE$GL_UMR_PHY                  00000048 RG D  02
EXE$GL_UMR_VIR                  0000004C RG D  02
FILLSPT                         000001F2 R   D  02
FUNC                          = 00000010     D
INIT_MAPREGS                    000000F6 R   D  02
LBN                           = 0000000C     D
MBA$L_MAP                     = 00000800     D
```

```
MODE                            = 00000014    D
NDT$_BLA                        = 80000103    D
NDT$_BUA                        = 80000102    D
NDT$_UBO                        = 00000028    D
PR$_MAPEN                       = 00000038    D
PR$_P0BR                        = 00000008    D
PR$_SID_TYP780                  = 00000001    D
PR$_SID_TYPRTUV2                  ********    X   02
PR$_SID_TYPUV2                  = 00000008    D
PTE$C_KW                        = 10000000    D
PTE$M_PFN                       = 001FFFFF    D
PTE$M_VALID                     = 80000000    D
PUSH_RETRY                        0000017A R  D   02
RPB                             = 0000000C    D
RPB$B_DEVTYP                    = 00000066    D
RPB$L_ADPPHY                    = 0000005C    D
RPB$L_ADPVIR                    = 00000060    D
RPB$L_CSRPHY                    = 00000054    D
RPB$L_CSRVIR                    = 00000058    D
RPB$L_IOVEC                     = 00000034    D
RPB$L_IOVECSZ                   = 00000038    D
RPB$L_SVASPT                    = 00000050    D
SIZE                            = 00000008    D
SS$_NORMAL                      = 00000001    D
SS$_PARITY                      = 000001F4    D
SVASPT                          = 00000004    D
UBA$L_DPR                       = 00000040    D
UBA$L_MAP                       = 00000800    D
UBA$M_MAP_VALID                 = 80000000    D
UBA$V_DPR_BNE                   = 0000001F    D
UBA$V_DPR_XMTER                 = 0000001E    D
UBA$V_MAP_BO                    = 00000019    D
UBA$V_MAP_DPD                   = 00000015    D
UBI$C_PURCNT                    = 0000000A    D
UBI$L_DPR                       = 00000000    D
UBI$L_MAP                       = 00000800    D
UBI$L_SR                        = 00000010    D
UBI$M_SR_MRPE                   = 00008000    D
UBI$M_SR_NXM                    = 00010000    D
UBI$M_SR_UCE                    = 80000000    D
UBI$M_SR_UWE                    = 00004000    D
UBI$V_DPR_ERROR                 = 0000001F    D
UBI$V_DPR_PUR                   = 00000000    D
VA$S_BYTE                       = 00000009    D
VA$S_VPN                        = 00000015    D
VA$V_BYTE                       = 00000000    D
VA$V_SYSTEM                     = 0000001F    D
VA$V_VPN                        = 00000009    D
VABASE                          = 00000008    D
VMB_VERSION                     = 0000000D    D
```

```
                                    +----------------+
                                    ! Psect synopsis !
                                    +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000018  (   24.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
BOOTDRIVR_1                     0000028D  (  653.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
BOOTDRIVR_3                     00000000  (    0.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
BOOTDRIVR_5                     00000004  (    4.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
BOOTDRIVR_6                     00000071  (  113.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BDA_MAPREGS	00000000-XR		273 (2) 483 (4)
BDT$K_LENGTH	=00000018		#-893 (8)
BDT$L_ACTION	00000004		#-889 (8)
BDT$L_ADDR	0000000C		#-906 (8)
BDT$L_CPUTYPE	00000000		#-878 (8)
BDT$L_DEVTYPE	00000002		#-884 (8)
BDT$L_DRIVRNAME	00000014		#-908 (8)
BDT$L_ENTRY	00000010		#-905 (8)
BDT$L_SIZE	00000008		#-902 (8)
BIT...	=00660160	210 (2)	210 (2)
BOO$AL_VECTOR	00000000-R	283 (3)	284 (3) 285 (3) 286 (3) 324 (3) 326 (3) 327 (3) 328 (3) 330 (3) 333 (3) 336 (3) 339 (3) 341 (3) 343 (3) 346 (3) 349 (3) 352 (3) 355 (3) 358 (3) 368 (3) 373 (3) 823 (7) #-903 (8) #-905 (8) #-906 (8) #-908 (8)
BOO$DRIVER	=0000028D-R	825 (7)	
BOO$DRIVER_TBL	=00000000-R	832 (7)	870 (8)
BOO$GB_UMR_DP	00000038-R	347 (3)	#-578 (4)
BOO$GL_UCODE	00000028-R	337 (3)	
BOO$GL_UMR_DIS	00000024-R	334 (3)	#-544 (4) #-644 (4)
BOO$GL_UMR_TMPL	00000034-R	344 (3)	#-424 (4) 580 (4) #-599 (4) 662 (4)
BOO$MAP	000001A8-R	703 (5)	285 (3)
BOO$MOVE	00000070-R	932 (9)	327 (3)
BOO$PURDPR	00000200-R	767 (6)	#-665 (4)
BOO$QIO	00000050-R	419 (4)	284 (3)
BOO$QIOSIZ	=0000028D	823 (7)	#-901 (8)
BOO$RESELECT	00000000-R	867 (8)	326 (3)
BOO$SELECT	00000000-R	868 (8)	286 (3)
BQO$B_CPUTYPE	=00000039		349 (3)
BQO$B_UMR_DP	=00000038		346 (3)
BQO$L_AUXDRNAME	=00000020		330 (3)
BQO$L_CPUDATA	=0000003A		352 (3)
BQO$L_DEVNAME	=00000030		341 (3)
BQO$L_TENUSEC	=0000003E		355 (3)
BQO$L_UBDELAY	=00000042		358 (3)
BQO$L_UCODE	=00000028		336 (3)
BQO$L_UMR_DIS	=00000024		333 (3)
BQO$L_UMR_PHY	=00000048	368 (3)	
BQO$L_UMR_TMPL	=00000034		343 (3)
BQO$L_UMR_VIR	=0000004C	373 (3)	
BQO$L_UNIT_DISC	=0000002C		339 (3)
BQO$L_UNIT_INIT	=0000001C		328 (3)
BQO$W_VERSION	=00000010		324 (3)
BTD$K_BDA	=00000021	213 (2)	#-468 (4) #-535 (4)
BTD$K_HSCCI	=00000020		#-501 (4)
BUA$L_CSR	=00000720		#-812 (6) #-820 (6)
BUA$L_DPCSR1	=00000754		#-811 (6)
BUA$M_BADBDP	=01000000		#-818 (6)

```
BUA$M_BIF                       =10000000                       #-814    (6)
BUA$M_IMR                       =02000000                       #-817    (6)
BUA$M_PURGE                     =00000001                       #-810    (6)
BUA$M_UIE                       =04000000                       #-816    (6)
BUA$M_USSTO                     =08000000                       #-815    (6)
BUA$V_ERR                       =0000001F                       #-813    (6)
BUF                             =00000004       412     (4)     #-442    (4)     #-597    (4)
COMPUTE_PFN                      00000112-R     547     (4)     #-541    (4)     #-621    (4)
DS$K_ERROR                      =00000002       210     (2)
DS$K_NORMAL                     =00000001       210     (2)
DS$K_SEVERE                     =00000004       210     (2)
DS$K_SUBSYS                     =00000066       210     (2)       210    (2)
DS$K_WARNING                    =00000000       210     (2)
DS$_AP_NORMAL_BREAK             =00660150       210     (2)
DS$_ARITH                       =006600D0       210     (2)
DS$_ASBE                        =00660118       210     (2)
DS$_BADLINK                     =006600F0       210     (2)
DS$_BADTYPE                     =006600E8       210     (2)
DS$_BIIC                        =00660120       210     (2)
DS$_CHME                        =006600A8       210     (2)
DS$_CHMK                        =006600E0       210     (2)
DS$_DEVNAME                     =00660108       210     (2)
DS$_ERROR                       =00660002       210     (2)
DS$_FHWE                        =00660068       210     (2)
DS$_FRAGBUF                     =00660080       210     (2)
DS$_ICBUSY                      =006600C8       210     (2)
DS$_ICERR                       =006600C0       210     (2)
DS$_IHWE                        =00660060       210     (2)
DS$_ILLCHAR                     =00660018       210     (2)
DS$_ILLPAGCNT                   =00660078       210     (2)
DS$_ILLUNIT                     =00660100       210     (2)
DS$_INIT_FAIL                   =00660140       210     (2)
DS$_INSFMEM                     =00660050       210     (2)
DS$_INVCPU                      =00660130       210     (2)
DS$_IPL2HI                      =006600B8       210     (2)
DS$_IVADDR                      =00660040       210     (2)
DS$_IVVECT                      =00660038       210     (2)
DS$_KRNLSTK                     =00660090       210     (2)
DS$_LOGIC                       =00660070       210     (2)
DS$_MCHK                        =00660088       210     (2)
DS$_MEM_ALLOC_ERR               =00660138       210     (2)
DS$_MMOFF                       =00660058       210     (2)
DS$_NEEDUNIT                    =006600F8       210     (2)
DS$_NODE                        =00660128       210     (2)
DS$_NOPCS                       =00660110       210     (2)
DS$_NORMAL                      =00660001       210     (2)
DS$_NOSUPPORT                   =006600B1       210     (2)
DS$_NOTALLOWMP                  =00660148       210     (2)
DS$_NOTDON                      =00660030       210     (2)
DS$_NOTIMP                      =006600B0       210     (2)       210    (2)
DS$_NULLSTR                     =00660010       210     (2)
DS$_OVERFLOW                    =00660008       210     (2)
DS$_POWER                       =00660098       210     (2)
DS$_PROGERR                     =00660020       210     (2)
DS$_SEVERE                      =00660004       210     (2)
DS$_TRANSL                      =006600A0       210     (2)
DS$_TRUNCATE                    =00660028       210     (2)
```

```
DS$_UNEXPINT                    =006600D8      210     (2)
DS$_UNSUPPDEV                   =00660158      210     (2)     #-921     (8)
DS$_VASFULL                     =00660048      210     (2)
DS$_WARNING                     =00660000      210     (2)       210     (2)
EXE$GB_CPUDATA                   0000003A-R    353     (3)
EXE$GB_CPUTYPE                   00000039-R    350     (3)     #-470     (4)     #-472     (4)     #-778     (6)     #-872     (8)
EXE$GL_TENUSEC                   0000003E-R    356     (3)
EXE$GL_UBDELAY                   00000042-R    359     (3)
EXE$GL_UMR_PHY                   00000048-R    369     (3)     #-490     (4)       494     (4)     #-498     (4)       538     (4)
                                                               #-539     (4)
EXE$GL_UMR_VIR                   0000004C-R    374     (3)     #-486     (4)     #-497     (4)       538     (4)
FILLSPT                          000001F2-R    730     (5)     #-713     (5)     #-717     (5)     #-733     (5)
FUNC                            =00000010      415     (4)
INIT_MAPREGS                     000000F6-R    531     (4)     #-499     (4)     #-502     (4)
LBN                             =0000000C      414     (4)     #-451     (4)
MBA$L_MAP                       =00000800                        542     (4)       543     (4)       545     (4)
MODE                            =00000014      416     (4)     #-549     (4)
NDT$_BLA                        =80000103                       #-570     (4)
NDT$_BUA                        =80000102                       #-566     (4)
NDT$_UBO                        =00000028                       #-575     (4)
PR$_MAPEN                       =00000038                       #-432     (4)     #-488     (4)     #-537     (4)     #-652     (4)
PR$_PQBR                        =00000008                       #-454     (4)
PR$_SID_TYP780                  =00000001                       #-778     (6)
PR$_SID_TYPRTUV2                 00000000-XR                    #-472     (4)
PR$_SID_TYPUV2                  =00000008                       #-470     (4)
PTE$C_KW                        =10000000                       #-731     (5)
PTE$M_PFN                       =001FFFFF                       #-551     (4)
PTE$M_VALID                     =80000000                       #-731     (5)
PUSH_RETRY                       0000017A-R    655     (4)     #-474     (4)     #-503     (4)
RPB                             =0000000C      696     (5)     #-431     (4)     #-704     (5)
RPB$B_DEVTYP                    =00000066                       #-467     (4)     #-500     (4)     #-534     (4)     #-871     (8)
RPB$L_ADPPHY                    =0000005C                         433     (4)     #-434     (4)       709     (5)
RPB$L_ADPVIR                    =00000060                         433     (4)     #-708     (5)
RPB$L_CSRPHY                    =00000054                         650     (4)     #-653     (4)     #-715     (5)     #-718     (5)
RPB$L_CSRVIR                    =00000058                         650     (4)     #-719     (5)
RPB$L_IOVEC                     =00000034                       #-659     (4)     #-913     (8)
RPB$L_IOVECSZ                   =00000038                       #-902     (8)
RPB$L_SVASPT                    =00000050                       #-452     (4)     #-492     (4)     #-706     (5)
SIZE                            =00000008      413     (4)     #-443     (4)
SS$_NORMAL                      =00000001                       #-769     (6)
SS$_PARITY                      =000001F4                       #-785     (6)
SVASPT                          =00000004      689     (5)     #-705     (5)
UBA$L_DPR                       =00000040                         780     (6)
UBA$L_MAP                       =00000800                         542     (4)
UBA$M_MAP_VALID                 =80000000                         345     (3)     #-423     (4)     #-643     (4)
UBA$V_DPR_BNE                   =0000001F                       #-781     (6)
UBA$V_DPR_XMTER                 =0000001E                       #-783     (6)     #-784     (6)
UBA$V_MAP_BO                    =00000019                       #-597     (4)
UBA$V_MAP_DPD                   =00000015                       #-579     (4)     #-661     (4)
UBI$C_PURCNT                    =0000000A                       #-790     (6)
UBI$L_DPR                       =00000000                         788     (6)
UBI$L_MAP                       =00000800                         543     (4)
UBI$L_SR                        =00000010                       #-800     (6)
UBI$M_SR_MRPE                   =00008000                       #-803     (6)
UBI$M_SR_NXM                    =00010000                       #-804     (6)
UBI$M_SR_UCE                    =80000000                       #-805     (6)
UBI$M_SR_UWE                    =00004000                       #-802     (6)
```

```
UBI$V_DPR_ERROR                 =0000001F                       #-796    (6)
UBI$V_DPR_PUR                   =00000000                       #-789    (6)     #-793    (6)
VA$S_BYTE                       =00000009                       #-446    (4)     #-634    (4)
VA$S_VPN                        =00000015                       #-455    (4)     #-491    (4)     #-493    (4)     #-709    (5)
                                                                #-711    (5)
VA$V_BYTE                       =00000000                       #-446    (4)     #-634    (4)
VA$V_SYSTEM                     =0000001F                       #-453    (4)
VA$V_VPN                        =00000009                       #-455    (4)     #-491    (4)     #-493    (4)     #-709    (5)
                                                                #-711    (5)
VABASE                          =00000008     692    (5)        #-707    (5)
VMB_VERSION                     =0000000D     322    (3)         325     (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BQODEF	2	211 (2)	211 (2)
$BTDDEF	2	212 (2)	212 (2)
$BUADEF	4	214 (2)	214 (2)
$DEF	1	210 (2)	
$DEFINI	1	211 (2)	211 (2) 212 (2) 214 (2) 215 (2) 216 (2) 217 (2) 218 (2) 219 (2) 220 (2) 221 (2) 222 (2) 223 (2) 224 (2) 257 (2)
$DS_DSDEF	3	210 (2)	210 (2)
$EQU	1	210 (2)	210 (2)
$EQULS1	1	210 (2)	210 (2)
$EQULST	1	210 (2)	210 (2)
$GBLINI	2	210 (2)	210 (2)
$IODEF	15	215 (2)	215 (2)
$MBADEF	5	216 (2)	216 (2)
$NDTDEF	3	217 (2)	217 (2)
$PRDEF	5	218 (2)	218 (2)
$PTEDEF	3	219 (2)	219 (2)
$RPBDEF	5	220 (2)	220 (2)
$SSDEF	21	221 (2)	221 (2)
$UBADEF	6	222 (2)	222 (2)
$UBIDEF	2	223 (2)	223 (2)
$VADEF	1	224 (2)	224 (2)
$VIELD1	1	210 (2)	
ASSUME	1	324 (3)	324 (3) 328 (3) 330 (3) 333 (3) 336 (3) 339 (3) 341 (3) 343 (3) 346 (3) 349 (3) 352 (3) 355 (3) 358 (3) 433 (4) 538 (4) 542 (4) 543 (4) 650 (4)
CASE	1	778 (6)	778 (6)
CPUDISP	1	249 (2)	770 (6)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	484 (4) 668 (4) 893 (8)
ADDL2	00C0	906 (8)
ADDL3	00C1	901 (8) 904 (8) 907 (8)
ASHL	0078	449 (4) 781 (6) 784 (6) 789 (6)
BBC	00E1	661 (4) 783 (6) 793 (6) 796 (6) 813 (6)
BBS	00E0	453 (4)
BEQL	0013	471 (4) 569 (4) 573 (4) 806 (6) 879 (8) 890 (8)
BGTR	0014	621 (4)
BICL	00CA	485 (4) 551 (4) 574 (4) 643 (4) 814 (6)
BICL3	00CB	715 (5)
BISL	00C8	645 (4)
BISL3	00C9	599 (4) 731 (5)
BITL	00D3	802 (6)
BLBC	00E9	489 (4) 549 (4) 642 (4)
BLBS	00E8	667 (4) 671 (4) 892 (8)
BLSS	0019	502 (4) 880 (8) 885 (8)
BNEQ	0012	444 (4) 469 (4) 473 (4) 536 (4) 576 (4) 882 (8) 887 (8)
BRB	0011	495 (4) 499 (4) 541 (4) 669 (4) 795 (6) 798 (6) 808 (6)
		821 (6) 894 (8)
BRW	0031	474 (4) 503 (4)
BSBB	0010	713 (5) 717 (5)
BSBW	0030	665 (4)
CASEB	008F	778 (6)
CLRL	00D4	490 (4) 564 (4)
CMPB	0091	467 (4) 470 (4) 472 (4) 500 (4) 534 (4)
CMPL	00D1	881 (8) 886 (8)
CMPW	00B1	566 (4) 570 (4) 575 (4)
CVTWL	0032	878 (8) 884 (8)
DECL	00D7	620 (4)
EXTZV	00EF	446 (4) 455 (4) 491 (4) 634 (4) 709 (5) 711 (5)
HALT	0000	933 (9)
INCL	00D6	581 (4) 732 (5)
INSV	00F0	493 (4) 578 (4) 597 (4)
JMP	0017	914 (8)
JSB	0016	660 (4) 891 (8)
MCOML	00D2	797 (6)
MFPR	00DB	432 (4) 454 (4) 488 (4) 537 (4) 652 (4)
MOVAB	009E	447 (4) 483 (4) 548 (4) 719 (5)
MOVAL	00DE	545 (4) 712 (5) 780 (6) 788 (6) 870 (8)
MOVL	00D0	423 (4) 431 (4) 434 (4) 442 (4) 451 (4) 452 (4) 486 (4)
		492 (4) 497 (4) 539 (4) 544 (4) 550 (4) 565 (4) 653 (4)
		658 (4) 659 (4) 704 (5) 705 (5) 706 (5) 707 (5) 708 (5)
		710 (5) 714 (5) 782 (6) 790 (6) 792 (6) 800 (6) 810 (6)
		812 (6) 820 (6) 889 (8) 913 (8) 921 (8)
MOVZBL	009A	871 (8) 872 (8)
MOVZWL	003C	443 (4) 718 (5) 769 (6) 785 (6)
POPL	8ED0	670 (4)
POPR	00BA	666 (4) 912 (8)
PUSHL	00DD	656 (4) 663 (4)
PUSHR	00BB	664 (4) 869 (8)

RET	0004	673	(4)	720	(5)		
ROTL	009C	445	(4)	644	(4)	716	(5)
RSB	0005	734	(5)	786	(6)	922	(8)
SOBGTR	00F5	672	(4)	733	(5)	794	(6)
SUBL3	00C3	903	(8)				

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.BYTE	348	(3)	351	(3)												
.END	934	(9)														
.ENDC	210	(2)	324	(3)	328	(3)	330	(3)	333	(3)	336	(3)	339	(3)	341	(3)
	343	(3)	346	(3)	349	(3)	352	(3)	355	(3)	358	(3)	433	(4)	538	(4)
	542	(4)	543	(4)	650	(4)										
.ENDM	210	(2)	211	(2)	212	(2)	214	(2)	215	(2)	216	(2)	217	(2)	218	(2)
	219	(2)	220	(2)	221	(2)	222	(2)	223	(2)	224	(2)	251	(2)	324	(3)
	778	(6)														
.ENDR	210	(2)	778	(6)												
.IDENT	11	(1)														
.IF	210	(2)	324	(3)	328	(3)	330	(3)	333	(3)	336	(3)	339	(3)	341	(3)
	343	(3)	346	(3)	349	(3)	352	(3)	355	(3)	358	(3)	433	(4)	538	(4)
	542	(4)	543	(4)	650	(4)										
.IFF	210	(2)	324	(3)	328	(3)	330	(3)	333	(3)	336	(3)	339	(3)	341	(3)
	343	(3)	346	(3)	349	(3)	352	(3)	355	(3)	358	(3)	433	(4)	538	(4)
	542	(4)	543	(4)	650	(4)										
.IIF	210	(2)														
.IRP	210	(2)	778	(6)												
.LIBRARY	203	(2)	204	(2)	205	(2)										
.LONG	284	(3)	285	(3)	286	(3)	289	(3)	326	(3)	327	(3)	329	(3)	331	(3)
	335	(3)	338	(3)	340	(3)	342	(3)	345	(3)	354	(3)	357	(3)	360	(3)
	370	(3)	375	(3)	836	(7)										
.MACRO	210	(2)	249	(2)												
.MDELETE	210	(2)														
.NOCROSS	211	(2)	212	(2)	214	(2)	215	(2)	216	(2)	217	(2)	218	(2)	219	(2)
	220	(2)	221	(2)	222	(2)	223	(2)	224	(2)	257	(2)				
.PSECT	281	(3)	830	(7)	834	(7)	838	(7)								
.RESTORE	211	(2)	212	(2)	214	(2)	215	(2)	216	(2)	217	(2)	218	(2)	219	(2)
	220	(2)	221	(2)	222	(2)	223	(2)	224	(2)	269	(2)				
.SAVE	211	(2)	212	(2)	214	(2)	215	(2)	216	(2)	217	(2)	218	(2)	219	(2)
	220	(2)	221	(2)	222	(2)	223	(2)	224	(2)	257	(2)				
.SBTTL	198	(2)	275	(3)	378	(4)	675	(5)	736	(6)	840	(8)	924	(9)		
.SIGNED_WORD	778	(6)														
.TITLE	10	(1)														
.WEAK	273	(2)														
.WORD	325	(3)	365	(3)	420	(4)	703	(5)								

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	9	#-431	(4)	#-442	(4)	#-443	(4)	#-451	(4)	#-549	(4)	#-597	(4)
		#-704	(5)	#-705	(5)	#-707	(5)						
R0	43	#-483	(4)	#-484	(4)	#-485	(4)	#-486	(4)	491	(4)	#-492	(4)
		#-493	(4)	#-564	(4)	#-581	(4)	#-642	(4)	#-644	(4)	#-645	(4)
		#-652	(4)	#-653	(4)	#-659	(4)	660	(4)	#-663	(4)	#-667	(4)
		#-670	(4)	#-671	(4)	#-711	(5)	712	(5)	#-718	(5)	719	(5)
		#-769	(6)	#-785	(6)	#-878	(8)	#-881	(8)	#-884	(8)	#-886	(8)
		#-889	(8)	891	(8)	#-892	(8)	#-903	(8)	#-904	(8)	#-907	(8)
		#-913	(8)	914	(8)	#-921	(8)						
R1	22	#-432	(4)	#-434	(4)	#-452	(4)	#-454	(4)	#-550	(4)	#-664	(4)
		#-666	(4)	#-712	(5)	#-731	(5)	#-782	(6)	783	(6)	#-792	(6)
		793	(6)	796	(6)	#-800	(6)	#-805	(6)	#-812	(6)	813	(6)
		#-819	(6)	#-820	(6)	#-869	(8)	#-912	(8)				
R10	8	421	(4)	#-442	(4)	446	(4)	453	(4)	455	(4)	635	(4)
		#-645	(4)										
R11	3	421	(4)	#-451	(4)	#-658	(4)						
R2	17	420	(4)	#-455	(4)	548	(4)	703	(5)	#-705	(5)	#-706	(5)
		712	(5)	#-780	(6)	#-781	(6)	#-782	(6)	#-784	(6)	#-788	(6)
		#-789	(6)	#-792	(6)	#-797	(6)	#-869	(8)	#-912	(8)		
R3	19	420	(4)	#-434	(4)	545	(4)	#-565	(4)	703	(5)	#-707	(5)
		#-708	(5)	711	(5)	719	(5)	780	(6)	788	(6)	#-800	(6)
		#-811	(6)	#-812	(6)	#-820	(6)	#-869	(8)	#-871	(8)	#-886	(8)
		#-912	(8)										
R4	24	420	(4)	#-488	(4)	#-489	(4)	#-537	(4)	#-539	(4)	#-544	(4)
		545	(4)	#-599	(4)	#-643	(4)	703	(5)	#-709	(5)	#-715	(5)
		#-716	(5)	#-731	(5)	#-732	(5)	#-790	(6)	#-794	(6)	#-869	(8)
		#-872	(8)	#-881	(8)	#-912	(8)						
R5	25	420	(4)	#-548	(4)	#-550	(4)	#-551	(4)	598	(4)	#-599	(4)
		#-658	(4)	703	(5)	#-710	(5)	#-714	(5)	#-733	(5)	#-869	(8)
		#-870	(8)	#-878	(8)	#-884	(8)	#-889	(8)	891	(8)	#-893	(8)
		#-902	(8)	#-903	(8)	#-905	(8)	#-906	(8)	#-908	(8)	#-912	(8)
R6	7	420	(4)	#-565	(4)	#-566	(4)	#-570	(4)	#-574	(4)	#-575	(4)
		703	(5)										
R7	16	420	(4)	#-446	(4)	447	(4)	#-449	(4)	#-620	(4)	#-653	(4)
		703	(5)	#-704	(5)	#-706	(5)	#-708	(5)	709	(5)	#-715	(5)
		#-718	(5)	#-719	(5)								
R8	4	421	(4)	#-443	(4)	#-445	(4)	447	(4)				
R9	13	421	(4)	#-431	(4)	#-434	(4)	#-452	(4)	#-467	(4)	#-492	(4)
		#-500	(4)	#-534	(4)	#-653	(4)	#-659	(4)	#-871	(8)	#-902	(8)
		#-913	(8)										
SP	2	#-668	(4)	#-672	(4)								

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.02	00:00:00.20
Command processing	871	00:00:00.21	00:00:00.79

Pass 1	814	00:00:03.26	00:00:04.91
Symbol table sort	0	00:00:00.42	00:00:00.42
Pass 2	7	00:00:00.96	00:00:01.85
Symbol table output	0	00:00:00.04	00:00:00.09
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	4	00:00:00.23	00:00:00.36
Assembler run totals	1719	00:00:05.15	00:00:08.63

The working set limit was 3512 pages.
182865 bytes (358 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1509 non-local and 35 local symbols.
934 source lines were read in Pass 1, producing 99 object records in Pass 2.
93 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	2
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	10
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	20

1609 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:BOOTDRIVR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

Table of contents

(1) 98 RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMBER
(1) 162 BOO$CACHE_INIT - INIT FILEREAD CACHE
(1) 295 BOO$IMAGE_ATT - Get image attributes from image header
(1) 341 SYS$ASSIGN, Dummy assign device system service

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element BOOTIO.MAR
0000     2 ; *3    12-MAY-1986 10:34:59 BARANSKI "Cleaning up Object Libraries."
0000     3 ; *2     7-MAY-1986 16:58:08 BARANSKI "Documentation Check."
0000     4 ; *1     6-FEB-1986 15:13:52 DS ""
0000     5 ; DEC/CMS REPLACEMENT HISTORY, Element BOOTIO.MAR
0000     6         .TITLE  BOOTIO - BOOTSTRAP FILEREAD IO MODULE
0000     7         .IDENT  "6.12-2"                                                ;G:2
0000     8 ;
0000     9 ;*******************************************************************************
0000    10 ;*                                                                             *
0000    11 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983 BY                                   * ;G:2
0000    12 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                     *
0000    13 ;*  ALL RIGHTS RESERVED.                                                       *
0000    14 ;*                                                                             *
0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED      *
0000    16 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE     *
0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER      *
0000    18 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    20 ;*  TRANSFERRED.                                                               *
0000    21 ;*                                                                             *
0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE      *
0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
0000    24 ;*  CORPORATION.                                                               *
0000    25 ;*                                                                             *
0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS      *
0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
0000    28 ;*                                                                             *
0000    29 ;*                                                                             *
0000    30 ;*******************************************************************************
0000    31
0000    32 ;++
0000    33 ; FACILITY:     SYSTEM BOOTSTRAPPING
0000    34 ;
0000    35 ; ABSTRACT:
0000    36 ;
0000    37 ;       THIS MODULE PERFORMS LOGICAL BLOCK I/O FOR FILEREAD
0000    38 ;
0000    39 ; ENVIRONMENT:  KERNEL MODE, UNMAPPED, IPL=31
0000    40 ;
0000    41 ; AUTHOR: RICHARD I. HUSTVEDT    , CREATION DATE: 14-APR-78
0000    42 ;
0000    43 ; MODIFIED BY:
0000    44 ;
0000    45 ;       02      Bob Bergazzi    June 29,1983    Version 6.12
0000    46 ;               Adapted to run for supervisor.
0000    47 ;
0000    48 ;       V03-001 KTA0091         Kerbey T. Altmann       30-Mar-1982
0000    49 ;               Redo some entries in the file cache table.  Make the
0000    50 ;               page allocation part of BOO$CACHE_INIT a subroutine.
0000    51 ;               Divide it into two pieces.
0000    52 ;
0000    53 ;       V02-003 PHL0012         Peter H. Lipman         02-Aug-1981
0000    54 ;               Explicitly initialize the FILEREAD cache pointer.
0000    55 ;       Must not allow assembly time initialization since VMB can restart.
0000    56 ;               Fix PSECT so that VMB can overlay most of itself with
0000    57 ;       the secondary boot file.  Remove BOO$GL_RPBBASE.
```

```
0000     58 ;
0000     59 ;       V02-002 PHL0011         Peter H. Lipman         31-Jul-1981
0000     60 ;               The page at which the FILEREAD cache is located
0000     61 ;       should be relative to the RPB not absolute.
0000     62 ;
0000     63 ;       V02-001 PHL0007         Peter H. Lipman         14-Mar-1981
0000     64 ;               Extend parmeter list to FIL$RDWRTLBN to make it
0000     65 ;       possible to read more than one block at a time.  This
0000     66 ;       enhancement was made in conjunction with the cacheing
0000     67 ;       feature in FILEREAD.
0000     68 ;               Add new BOO$CACHE_INIT routine for VMB and SYSBOOT to
0000     69 ;       init the FILEREAD cache.
0000     70 ;               Move other common FILEREAD pieces into this module thus
0000     71 ;       avoiding them being in two places.
0000     72 ;               Add new BOO$IMAGE_ATT routine to fetch certain image
0000     73 ;       attributes from an image header.
0000     74 ;
0000     75 ;--
```

```
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000            $IHDDEF                         ; IMAGE HEADER DEFINITIONS
0000            .SAVE   LOCAL_BLOCK
0000            .RESTORE
0000            $IHSDEF                         ; IMAGE HEADER SYMBOL TABLE DEFS
0000            .SAVE   LOCAL_BLOCK
0000            .RESTORE
0000            $IHPDEF                         ; IMAGE HEADER PATCH CONTROL DEFS
0000            .SAVE   LOCAL_BLOCK
0000            .RESTORE
0000            $RPBDEF                         ; DEFINE RESTART PARAMETER BLOCK
0000            .SAVE   LOCAL_BLOCK
0000            .RESTORE
0000    ;
0000    ; MACROS:
0000    ;
0000    ; Define Memory Size to File Cache Parameter table entry
0000    ;
0000            .MACRO  MEM_FILE_CACHE MEM_PAGE_CNT,CACHE_PAGE_NUM,CACHE_PAGE_CNT,MAX_PAGE
0000            .LONG   MEM_PAGE_CNT-<MEM_PAGE_CNT/10>
0000            .WORD   CACHE_PAGE_NUM
0000            .WORD   <<CACHE_PAGE_CNT+3>&^C<3>>
0000            .LONG   MAX_PAGE
0000            .ENDM   MEM_FILE_CACHE
```

```
                     00000000    96          .PSECT  code, exe, nowrt, shr, long     ;                                   [02]
                         0000    97
                         0000    98          .SBTTL  RDWRTLBN - READ/WRITE LOGICAL BLOCK NUMBER
                         0000    99 ;++
                         0000   100 ; FUNCTIONAL DESCRIPTION:
                         0000   101 ;
                         0000   102 ;       THIS ROUTINE READS/WRITES N BYTES FROM/TO THE SPECIFIED
                         0000   103 ; LOGICAL BLOCK NUMBER OF THE VOLUME ASSIGNED TO THE SPECIFIED CHANNEL
                         0000   104 ;
                         0000   105 ; CALLING SEQUENCE:
                         0000   106 ;
                         0000   107 ;       CALLG   ARGLIST,FIL$RDWRITLBN
                         0000   108 ;
                         0000   109 ; INPUT PARAMETERS:
                         0000   110 ;
                         0000   111 ;       CHAN(AP)        =               ;CHANNEL ASSIGNED TO THE VOLUME TO READ
                         0000   112 ;       LBN(AP)         =               ;LOGICAL BLOCK NUMBER TO READ
                         0000   113 ;       BUFADR(AP)      =               ;ADDRESS OF BUFFER TO READ INTO
                         0000   114 ;       IOFUNC(AP)      =               ;I/O FUNCTION CODE
                         0000   115 ;       BYTCNT(AP)      =               ;NUMBER OF BYTES TO TRANSFER
                         0000   116 ;
                         0000   117 ; IMPLICIT INPUTS:
                         0000   118 ;
                         0000   119 ;       NONE
                         0000   120 ;
                         0000   121 ; OUTPUT PARAMETERS:
                         0000   122 ;
                         0000   123 ;       R0 = SYSTEM STATUS CODE
                         0000   124 ;
                         0000   125 ; IMPLICIT OUTPUTS:
                         0000   126 ;
                         0000   127 ;       NONE
                         0000   128 ;
                         0000   129 ; COMPLETION CODES:
                         0000   130 ;
                         0000   131 ;       NONE
                         0000   132 ;
                         0000   133 ; SIDE EFFECTS:
                         0000   134 ;
                         0000   135 ;       NONE
                         0000   136 ;
                         0000   137 ; EQUATED SYMBOLS:
                         0000   138 ;
                         0000   139 ;    OFFSETS FROM AP
                         0000   140 ;
                     00000004   0000   141         CHAN            =       4       ;CHANNEL TO WHICH VOLUME IS ASSIGNED
                     00000008   0000   142         LBN             =       8       ;LOGICAL BLOCK NUMBER
                     0000000C   0000   143         BUFADR          =       12      ;BUFFER ADDRESS TO READ INTO
                     00000010   0000   144         IOFUNC          =       16      ;FUNCTION CODE FOR THE QIO
                     00000014   0000   145         BYTCNT          =       20      ;NUMBER OF BYTES TO TRANSFER
                         0000   0000   146 ;
                         0000   0000   147 ;--
                         0000   0000   148
                         0000   0000   149 FIL$RDWRTLBN::
                         0000   0000   150         .WORD   0
                   04 AC    DD  0002   151         PUSHL   CHAN(AP)                ; ADDRESS OF RPB
                50    6E    D0  0005   152         MOVL    (SP),R0                 ; GET ADDRESS OF RPB
```

```
                50    34 A0    D0  0008    153         MOVL    RPB$L_IOVEC(R0),R0      ; GET POINTER TO I/O ROUTINE VECTOR
                         00    DD  000C    154         PUSHL   #0                      ; SET MODE TO PHYSICAL ADDRESS
                      10 AC    DD  000E    155         PUSHL   IOFUNC(AP)              ; SET FUNCTION
                      08 AC    DD  0011    156         PUSHL   LBN(AP)                 ; LOGICAL BLOCK NUMBER
                      14 AC    DD  0014    157         PUSHL   BYTCNT(AP)              ; SET NUMBER OF BYTES
                      0C BC    DF  0017    158         PUSHAL  @BUFADR(AP)             ; SET BUFFER ADDRESS
              00 B040    06    FB  001A    159         CALLS   #6,@(R0)[R0]            ; CALL BOOTSTRAP DRIVER
                               04  001F    160         RET
```

```
                     0020   162          .SBTTL  BOO$CACHE_INIT - INIT FILEREAD CACHE
                     0020   163 ;++
                     0020   164 ;
                     0020   165 ; Functional description:
                     0020   166 ;
                     0020   167 ;       This routine establishes a desired FILEREAD cache size and
                     0020   168 ;       base address according to the size of memory.  It finds
                     0020   169 ;       good contiguous pages at or near the desired place and
                     0020   170 ;       calls the FIL$CACHE_INIT routine to initialize the cache.
                     0020   171 ;       The routine is further divided into two pieces: one to do
                     0020   172 ;       cache allocation, and one to do the actual mount and open.
                     0020   173 ;       This is necessary for VMB needs to allocate the cache long
                     0020   174 ;       befor it is ready to accept IO to the device.
                     0020   175 ;
                     0020   176 ; Calling Sequence:
                     0020   177 ;
                     0020   178 ;       BSBW    BOO$CACHE_INIT
                     0020   179 ;
                     0020   180 ; Inputs:
                     0020   181 ;
                     0020   182 ;       R11                         - RPB base address
                     0020   183 ;       RPB$L_PFNCNT(R11)           - actual number of good pages in memory
                     0020   184 ;       RPB$Q_PFNMAP+4(R11)         - base address of PFN bitmap
                     0020   185 ;
                     0020   186 ; Implicit inputs:
                     0020   187 ;
                     0020   188 ;       none
                     0020   189 ;
                     0020   190 ; Outputs:
                     0020   191 ;
                     0020   192 ;       R0-R4 altered
                     0020   193 ;       FIL$GQ_CACHE set up with size and address of cache
                     0020   194 ;
                     0020   195 ; Implicit outputs:
                     0020   196 ;
                     0020   197 ;--
                     0020   198 ;
                     0020   199 ; Table of memory sizes to file cache parameters
                     0020   200 ;
                     0020   201 ; NOTE: If this table is modified, a corresponding table in VMB around
                     0020   202 ;       label MEM_TAB should be checked for consitency.
                     0020   203 ;
                     0020   204 ;MEM_CACHE_TABLE:                                                    begin   [2]
                     0020   205 ;       MEM_FILE_CACHE 16384,2048,64,4096       ; More than 8 megabyte
                     0020   206 ;       MEM_FILE_CACHE  8192,1024,64,2048       ; More than 4 megabyte
                     0020   207 ;       MEM_FILE_CACHE  4096, 640,64,1024       ; More than 2 megabyte
                     0020   208 ;       MEM_FILE_CACHE  2048, 512,64, 768       ; More than 1 megabyte
                     0020   209 ;       MEM_FILE_CACHE  1024, 256,32, 512       ; More than 512k bytes
                     0020   210 ;       MEM_FILE_CACHE   512, 256,16, 256       ; More than 256k bytes
                     0020   211 ;       MEM_FILE_CACHE   384, 256, 8, 192       ; More than 192k bytes
                     0020   212 ;       MEM_FILE_CACHE   256, 128, 4, 128       ; More than 128k bytes
                     0020   213 ;       MEM_FILE_CACHE     0,   0, 0,   0       ;
```

```
0020   215 ;
0020   216 ; BOO$CACHE_ALLOC - The piece that does the allocation.
0020   217 ;
0020   218 ; Outputs:
0020   219 ;       FIL$GQ_CACHE filled in with size/address in blocks
0020   220 ;
0020   221 ;BOO$CACHE_ALLOC::
0020   222 ;       PUSHL   R5                              ; Save a register
0020   223 ;       CLRQ    W^FIL$GQ_CACHE                  ; Assume no cache available
0020   224 ;       MOVAL   B^MEM_CACHE_TABLE,R0            ; Adr of memory size to cache params tbl
0020   225 ;10$:   MOVQ    (R0)+,R1                        ; Get the next table entry
0020   226 ;       BEQL    20$                             ; Branch if memory too small for cache
0020   227 ;       MOVL    (R0)+,R5                        ; Max page
0020   228 ;       CMPL    RPB$L_PFNCNT(R11),R1            ; More memory than this entry?
0020   229 ;       BLSS    10$                             ; Branch if not, get next one
0020   230 ;       MOVZWL  R2,R0                           ; Starting relative bit (page) in PFNMAP
0020   231 ;       ASHL    #-16,R2,R1                      ; Count of bits (pages) to look for
0020   232 ;       ASHL    #-1,R1,R4                       ; Settle for half if can't find all
0020   233 ;       BSBB    BOO$ALLOC_PAGES                 ; Go get the pages
0020   234 ;       BLSS    20$                             ; Failed
0020   235 ;       MOVQ    R2,W^FIL$GQ_CACHE               ; Success, record the values
0020   236 ;20$:   POPL    R5                              ; Restore a register
0020   237 ;       RSB
0020   238 ;
0020   239 ;
0020   240 ; BOO$CACHE_INIT - Full routine to both allocate and open the cache
0020   241 ;
0020   242 ;BOO$CACHE_INIT::
0020   243 ;       BSBB    BOO$CACHE_ALLOC                 ; Allocate the cache
0020   244 ;                                               ; Fall thru to finish
0020   245 ;
0020   246 ;
0020   247 ; BOO$CACHE_OPEN - Actually mount the device and fill the cache
0020   248 ;
0020   249 ;BOO$CACHE_OPEN::
0020   250 ;       MOVL    W^FIL$GQ_CACHE,R2               ; Pick up size
0020   251 ;       BEQL    10$                             ; Zero length implies none
0020   252 ;       SUBL    #4,SP                           ; Location to store channel
0020   253 ;       MOVL    SP,R0                           ; Address to store channel
0020   254 ;       SUBL3   #2,R2,-(SP)                     ; Blocks in directory LBN cache
0020   255 ;       PUSHL   S^#<<1024-FIL$C_SIZE>/FIL$C_DIR_SIZE> ; No. of dir cache entries
0020   256 ;       ASHL    #9,W^FIL$GQ_CACHE+4,-(SP)       ; Byte address from page number
0020   257 ;       ASHL    #9,R2,-(SP)                     ; Size of cache in bytes
0020   258 ;       CLRL    -(SP)                           ; Null device name string descriptor
0020   259 ;       PUSHL   R0                              ; Address to store channel
0020   260 ;       CALLS   #6,W^FIL$CACHE_INIT             ; Init the FIL$OPENFILE cache
0020   261 ;                                               ; descriptor returned in FIL$GQ_CACHE
0020   262 ;       ADDL    #4,SP                           ; Clean off channel
0020   263 ;10$:   RSB
0020   264
0020   265 ;
0020   266 ; BOO$ALLOC_PAGES - Find a run of contiguous, good pages
0020   267 ;
0020   268 ; Inputs:
0020   269 ;       R0 - Page to start at
0020   270 ;       R1 - Number of pages needed
0020   271 ;       R4 - Number willing to settle for
```

```
0020   272 ;       R5 - Maximum page
0020   273 ; Outputs:
0020   274 ;       CC - Status (BLSS to an error routine)
0020   275 ;       R2 - Number found
0020   276 ;       R3 - Starting page number
0020   277 ;
0020   278 ;BOO$ALLOC_PAGES::
0020   279 ;       ROTL    #<32-9>,R11,R2          ; PFN of the RPB
0020   280 ;       ADDL    R2,R0                   ; Convert relative PFN to absolute
0020   281 ;       MOVL    R0,R3                   ; Make a copy of starting bit
0020   282 ;30$:   CMPL    R5,R0                   ; Less than max page
0020   283 ;       BLSS    50$                     ; No, failure
0020   284 ;       BBS     R0,@RPB$Q_PFNMAP+4(R11),40$ ; Branch if this is a good page
0020   285 ;       SUBL3   R3,R0,R2                ; Count of bits (pages) found
0020   286 ;       CMPL    R2,R4                   ; Found enough?
0020   287 ;       BGEQ    50$                     ; Branch if yes
0020   288 ;       ADDL3   #1,R0,R3                ; No, reset starting base
0020   289 ;40$:   INCL    R0                      ; Next bit (page)
0020   290 ;       SOBGTR  R1,30$                  ; Branch if more to check
0020   291 ;       SUBL3   R3,R0,R2                ; Count of bits (pages) found
0020   292 ;       CMPL    R2,R4                   ; Found enough?
0020   293 ;50$:   RSB                             ; Return (Status in CC)         end     [2]
```

```
                        0020   295          .SBTTL  BOO$IMAGE_ATT - Get image attributes from image header
                        0020   296 ;++
                        0020   297 ; Functional Description:
                        0020   298 ;
                        0020   299 ;       BOO$IMAGE_ATT returns to the caller some attributes of the image
                        0020   300 ;
                        0020   301 ; Calling Sequence:
                        0020   302 ;
                        0020   303 ;       BSBW    BOO$IMAGE_ATT
                        0020   304 ;
                        0020   305 ; Inputs:
                        0020   306 ;
                        0020   307 ;       R2 = Size of file in blocks
                        0020   308 ;       R3 = Address of image header block (first one only)
                        0020   309 ;
                        0020   310 ; Outputs:
                        0020   311 ;
                        0020   312 ;       R1 = Number of image header blocks at the front of the image
                        0020   313 ;       R2 = Size of image in blocks excluding the blocks at the end
                        0020   314 ;            containing local symbols, global symbols, or patch text
                        0020   315 ;
                        0020   316 ;--
                        0020   317
                        0020   318 BOO$IMAGE_ATT::
              50  04 A3  3C  0020   319          MOVZWL  IHD$W_SYMDBGOFF(R3),R0  ; ANY SYMBOL TABLE INFORMATION?
                    0D  13  0024   320          BEQL    20$                     ; BRANCH IF NOT
               51  6043  9E  0026   321          MOVAB   IHS$L_DSTVBN(R0)[R3],R1 ; ADR OF 1ST VBN IN DEBUG SYMBOL TABLE
                    19  10  002A   322          BSBB    40$                     ; PROCESS IT
            51  04 A043  9E  002C   323          MOVAB   IHS$L_GSTVBN(R0)[R3],R1 ; ADR OF 1ST VBN IN GLOBAL SYMBOL TABLE
                    12  10  0031   324          BSBB    40$                     ; PROCESS IT
              50  08 A3  3C  0033   325 20$:     MOVZWL  IHD$W_PATCHOFF(R3),R0   ; ANY PATCH CONTROL INFORMATION?
                    07  13  0037   326          BEQL    30$                     ; BRANCH IF NOT
            51  20 A043  9E  0039   327          MOVAB   IHP$L_PATCOMTXT(R0)[R3],R1 ; ADR OF 1ST VBN OF PATCH COMMAND TEXT
                    05  10  003E   328          BSBB    40$                     ; PROCESS IT
              51  10 A3  9A  0040   329 30$:     MOVZBL  IHD$B_HDRBLKCNT(R3),R1  ; GET IMAGE HEADER BLOCK COUNT
                        05  0044   330          RSB
                        0045   331 ;
                        0045   332 ; SEE IF VBN IS NON ZERO AND THEN IF IT IS SMALLER THAN THE CURRENT SMALLEST
                        0045   333 ;
            51  61  01  C3  0045   334 40$:     SUBL3   #1,(R1),R1              ; FETCH VBN - 1
                    08  19  0049   335          BLSS    50$                     ; BRANCH IF NO VBN IS PRESENT
                51  52  D1  004B   336          CMPL    R2,R1                   ; IS IT SMALLER THAN THE CURRENT ONE
                    03  15  004E   337          BLEQ    50$                     ; BRANCH IF NOT
                52  51  D0  0050   338          MOVL    R1,R2                   ; YES, USE IT
                        05  0053   339 50$:     RSB
```

```
                        0054   341          .SBTTL  SYS$ASSIGN, Dummy assign device system service
                        0054   342
                        0054   343 ;++
                        0054   344 ;
                        0054   345 ; Functional description:
                        0054   346 ;
                        0054   347 ;       SYS$ASSIGN is a dummy routine to satisfy the requirements of
                        0054   348 ;       FIL$OPENFILE.
                        0054   349 ;
                        0054   350 ; Inputs:
                        0054   351 ;
                        0054   352 ;       CHAN(AP)        - address at which to return channel
                        0054   353 ;
                        0054   354 ; Implicit inputs:
                        0054   355 ;
                        0054   356 ;       The label BOO$GL_RPBBASE contains the physical address of the RPB.
                        0054   357 ;
                        0054   358 ; Outputs:
                        0054   359 ;
                        0054   360 ;       R0      - success status code
                        0054   361 ;
                        0054   362 ; Implicit outputs:
                        0054   363 ;
                        0054   364 ;       The channel returned is not a channel. It is instead the base
                        0054   365 ;       address of the RPB.
                        0054   366 ;
                        0054   367 ;--
                        0054   368 ;                                                               begin   [2]
                        0054   369 ;CHAN   = 8
                        0054   370 ;
                        0054   371 ;SYS$ASSIGN::                                  ; Dummy system service.
                        0054   372 ;       .WORD   0
                        0054   373 ;
                        0054   374 ;       MOVL    W^BOO$GL_RPBBASE,@CHAN(AP) ; Store RPB address as channel.
                        0054   375 ;       MOVL    S^#SS$_NORMAL,R0            ; Return success status.
                        0054   376 ;       RET                                 ; Return to caller.
                        0054   377 ;       .PAGE
                        0054   378 ;       .SBTTL  Common Globals for VMB and SYSBOOT
                        0054   379 ;
                        0054   380 ; The following globals are common to VMB and SYSBOOT and are
                        0054   381 ; defined here to avoid replicate definitions.
                        0054   382 ;
                        0054   383 ;FIL$GT_DDSTRING::                             ; Default directory string.
                        0054   384 ;       .ASCIC  /[SYSEXE]/
                        0054   385 ;FIL$GT_DDDEV::                                ; Default device name
                        0054   386 ;       .BYTE   0                           ; Null ASCIC string
                        0054   387
                        0054   388          .END
```

```
BOO$IMAGE_ATT                     00000020 RG D  02
BUFADR                          = 0000000C     D
BYTCNT                          = 00000014     D
CHAN                            = 00000004     D
FIL$RDWRTLBN                      00000000 RG D  02
IHD$B_HDRBLKCNT                 = 00000010     D
IHD$W_PATCHOFF                  = 00000008     D
IHD$W_SYMDBGOFF                 = 00000004     D
IHP$L_PATCOMTXT                 = 00000020     D
IHS$L_DSTVBN                    = 00000000     D
IHS$L_GSTVBN                    = 00000004     D
IOFUNC                          = 00000010     D
LBN                             = 00000008     D
RPB$L_IOVEC                     = 00000034     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	00000054 (84.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL                          VALUE        DEFINITION      REFERENCES...
------                          -----        ----------      -------------
BOO$IMAGE_ATT                   00000020-R   318    (1)
BUFADR                         =0000000C     143    (1)       158    (1)
BYTCNT                         =00000014     145    (1)     #-157    (1)
CHAN                           =00000004     141    (1)     #-151    (1)
FIL$RDWRTLBN                    00000000-R   149    (1)
IHD$B_HDRBLKCNT                =00000010                    #-329    (1)
IHD$W_PATCHOFF                 =00000008                    #-325    (1)
IHD$W_SYMDBGOFF                =00000004                    #-319    (1)
IHP$L_PATCOMTXT                =00000020                      327    (1)
IHS$L_DSTVBN                   =00000000                      321    (1)
IHS$L_GSTVBN                   =00000004                      323    (1)
IOFUNC                         =00000010     144    (1)     #-155    (1)
LBN                            =00000008     142    (1)     #-156    (1)
RPB$L_IOVEC                    =00000034                    #-153    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	80 (1)	80 (1) 81 (1) 82 (1) 83 (1)
$IHDDEF	3	80 (1)	80 (1)
$IHPDEF	1	82 (1)	82 (1)
$IHSDEF	1	81 (1)	81 (1)
$RPBDEF	5	83 (1)	83 (1)
MEM_FILE_CACHE	1	89 (1)	

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
BEQL            0013      320    (1)    326    (1)
BLEQ            0015      337    (1)
BLSS            0019      335    (1)
BSBB            0010      322    (1)    324    (1)    328    (1)
CALLS           00FB      159    (1)
CMPL            00D1      336    (1)
MOVAB           009E      321    (1)    323    (1)    327    (1)
MOVL            00D0      152    (1)    153    (1)    338    (1)
MOVZBL          009A      329    (1)
MOVZWL          003C      319    (1)    325    (1)
PUSHAL          00DF      158    (1)
PUSHL           00DD      151    (1)    154    (1)    155    (1)    156    (1)    157    (1)
RET             0004      160    (1)
RSB             0005      330    (1)    339    (1)
SUBL3           00C3      334    (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...											
.END	388	(1)										
.ENDM	80	(1)	81	(1)	82	(1)	83	(1)	94	(1)		
.IDENT	7	(1)										
.MACRO	89	(1)										
.NOCROSS	80	(1)	81	(1)	82	(1)	83	(1)				
.PAGE	161	(1)	214	(1)	294	(1)	340	(1)	76	(1)	95	(1)
.PSECT	96	(1)										
.RESTORE	80	(1)	81	(1)	82	(1)	83	(1)				
.SAVE	80	(1)	81	(1)	82	(1)	83	(1)				
.SBTTL	162	(1)	295	(1)	341	(1)	98	(1)				
.TITLE	6	(1)										
.WORD	150	(1)										

```
                                        +---------------------------+
                                        ! Register Cross Reference !
                                        +---------------------------+

REGISTER  NO. REFERENCES     REFERENCES...
--------  --------------     -------------
AP                 5         #-151    (1)    #-155    (1)    #-156    (1)    #-157    (1)      158    (1)
R0                10         #-152    (1)    #-153    (1)      159    (1)    #-319    (1)      321    (1)      323    (1)
                             #-325    (1)      327    (1)
R1                 8         #-321    (1)    #-323    (1)    #-327    (1)    #-329    (1)    #-334    (1)    #-336    (1)
                             #-338    (1)
R2                 2         #-336    (1)    #-338    (1)
R3                 6         #-319    (1)      321    (1)      323    (1)    #-325    (1)      327    (1)    #-329    (1)
SP                 1         #-152    (1)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     26    00:00:00.05    00:00:00.21
Command processing                885    00:00:00.23    00:00:00.80
Pass 1                            260    00:00:00.55    00:00:01.24
Symbol table sort                   0    00:00:00.05    00:00:00.04
Pass 2                              8    00:00:00.22    00:00:00.44
Symbol table output                 0    00:00:00.01    00:00:00.01
Psect synopsis output               0    00:00:00.00    00:00:00.00
Cross-reference output              1    00:00:00.04    00:00:00.07
Assembler run totals             1180    00:00:01.15    00:00:02.81

The working set limit was 2512 pages.
26769 bytes (53 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 197 non-local and 4 local symbols.
388 source lines were read in Pass 1, producing 24 object records in Pass 2.
25 pages of virtual memory were used to define 11 macros.

                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           4
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       3
TOTALS (all libraries)                                 7

261 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:BOOTIO.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 72 GET ONE BYTE OF DATA FROM USER BUFFER
(1) 108 PUT ONE BYTE OF DATA INTO USER'S BUFFER
(1) 142 INITIALIZE FOR SINGLE BYTE TRANSFERS
(1) 164 MOVE FROM USER BUFFER
(1) 197 MOVE TO USER BUFFER
(1) 230 FILL SYSTEM PTE WITH TRANSFER PTE

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element BUFCTL.MAR
0000     2 ; *3    12-MAY-1986 10:40:47 BARANSKI "Cleaning up Object Libraries."
0000     3 ; *2     7-MAY-1986 17:12:04 BARANSKI "Documentation Check."
0000     4 ; *1     6-FEB-1986 15:13:55 DS ""
0000     5 ; DEC/CMS REPLACEMENT HISTORY, Element BUFCTL.MAR
0000     6         .TITLE  BUFCTL  *** BUFCTL QIO buffer control
0000     7         .IDENT  "5.4-3"                                                ;G:2
0000     8
0000     9 ;
0000    10 ;******************************************************************************
0000    11 ;*                                                                            *
0000    12 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                      *
0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
0000    14 ;*                                                                            *
0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED    *
0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY    *
0000    20 ;*  TRANSFERRED.                                                              *
0000    21 ;*                                                                            *
0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE    *
0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    24 ;*  CORPORATION.                                                              *
0000    25 ;*                                                                            *
0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS    *
0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
0000    28 ;*                                                                            *
0000    29 ;******************************************************************************
0000    30 ;
0000    31 ; D. N. CUTLER 9-AUG-76
0000    32 ;
0000    33 ; MODIFIED BY:
0000    34 ;
0000    35 ;                   NICK HOWGATE    28-MAR-78       VERSION 02 (ESSAA-4.01).
0000    36 ;       01          INCLUDED IN DIAGNOSTIC SUPERVISOR FOR QIO SUPPORT
0000    37 ;                                                                          ;G:2
0000    38 ;       02          CAM001          C. MONIA                  15-FEB-1979 ;G:2
0000    39 ;                   ADD IOC$PUTBYTE AND IOC$GETBYTE ROUTINES FOR TU-58 SUPPORT ;G:2
0000    40 ;                                                                          ;G:2
0000    41 ;                   Roger Riggs     9 April 1979    Version 5.0
0000    42 ;       02          Changed MOVFRUSER and MOVTOUSER not to use SVPN,
0000    43 ;                   instead just address it directly.
0000    44 ;                                                                          ;G:2
0000    45 ;       03          STJ0001         S. JEFFREYS               26-SEP-1979 ;G:2
0000    46 ;                   MODIFY IOC$PUTBYTE AND IOC$GETBYTE TO WORK CORRECTLY FOR   ;G:2
0000    47 ;                   BUFFERS THAT ARE PAGE ALIGNED.                         ;G:2
0000    48 ;                                                                          ;G:2
0000    49 ;       04          STJ0002         S. JEFFREYS               29-FEB-1980
0000    50 ;                   ADD ALTERNATE ENTRY POINTS FOR IOC$MOVTOUSER AND IOC$MOVFRUSER. ;G:2
0000    51 ;
0000    52 ;                   Dave Butenhof   13-may-1980, Version 5.4             ;G:2
0000    53 ;       03          Upgrade to VMS V2.0 level: add routines IOC$GETBYTE and   ;G:2
0000    54 ;                   IOC$PUTBYTE, dummy IOC$INITBUFWIND and IOC$FILSPT for     ;G:2
0000    55 ;                   code centralization, and extra entry points to            ;G:2
0000    56 ;                   IOC$MOVTOUSER and IOC$MOVFRUSER.                          ;G:2
0000    57 ;
```

```
                                0000     58 ;
                                0000     59 ; I/O BUFFER CONTROL ROUTINES
                                0000     60 ;
                                0000     61 ; MACRO LIBRARY CALLS
                                0000     62 ;
                                0000     63
                                0000     64          $PRDEF                              ;DEFINE PROCESSOR REGISTERS
                                0000                 .SAVE   LOCAL_BLOCK
                                0000                 .RESTORE
                                0000     65          $PTEDEF                             ;PAGE TABLE ENTRY DEFINITIONS
                                0000                 .SAVE   LOCAL_BLOCK
                                0000                 .RESTORE
                                0000     66          $UCBDEF                             ;DEFINE UCB OFFSETS
                                0000                 .SAVE   LOCAL_BLOCK
                                0000                 .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                                0000                 .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                    00000004    0000                         .BLKL   1
                                0004                 .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                                0004                 .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                    00000008    0004                         .BLKL   1
                                0008                 .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                    0000000A    0008                         .BLKW   1
                                000A                 .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                    0000000B    000A                         .BLKB   1
                                000B                 .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                    0000000C    000B                         .BLKB   1
                                000C                 .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                                000C                 .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                                000C                 .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                    0000000D    000C                         .BLKB   1
                    00000010    000D                         .BLKB   3
                                0010             .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                                0010             .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
                    00000014    0010                         .BLKL   1
                                0014             .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                                0014             .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                                0014             .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
                    00000016    0014                     .BLKW   1
                                0016             .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
                    00000018    0016                     .BLKW   1
                                0018             .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                                0018             .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
                    0000001A    0018                     .BLKW   1
                                001A             .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                                001A             .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                    0000001C    001A                     .BLKW   1
                                001C             .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                    00000020    001C                     .BLKL   1
                                0020             .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                    00000024    0020                     .BLKL   1
                                0024             .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                    00000028    0024                     .BLKL   1
                                0028             .IIF    NB,UCB$L_PID,   UCB$L_PID:
                    0000002C    0028                     .BLKL   1
                                002C             .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
                    00000030    002C                     .BLKL   1
                                0030             .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
```

```
00000034  0030                                  .BLKL   1
          0034                  .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039  0038                                  .BLKB   1
          0039                  .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A  0039                                  .BLKB   1
          003A                  .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C  003A                                  .BLKW   1
          003C                  .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
          003C                  .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
          003C                  .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D  003C                                  .BLKB   1
          003D                  .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
          003D                  .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E  003D                                  .BLKB   1
          003E                  .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
          003E                  .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F  003E                                  .BLKB   1
          003F                  .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040  003F                                  .BLKB   1
          0040                  .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                                  .BLKW   1
          004A                  .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
          004A                  .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                                  .BLKB   1
          004B                  .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                                  .BLKB   1
          004C                  .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                  .BLKW   1
          0052                  .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
          0052                  .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                  .BLKB   1
          0053                  .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                  .BLKB   1
          0054                  .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
          005A                  .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
          005C                  .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C                  .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                  .BLKL   1
          0060                  .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                                  .BLKL   1
          0064                  .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
          0064                  .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                                  .BLKL   1
          0068                  .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                                  .BLKL   1
```

```
                    006C            .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                              .BLKW   1
                    006E            .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                              .BLKW   1
                    0070            .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                              .BLKB   1
                    0071            .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                              .BLKB   1
                    0072            .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                              .BLKW   1
                    0074            .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                    0074            .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074                    . = 0
                    0000            .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                              .BLKW   1
00000074  0002                    . = 116
                    0074            .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                              .BLKL   1
                    0078            .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                              .BLKL   1
                    007C            .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                              .BLKL   1
                    0080            .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                              .BLKL   1
                    0084            .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                              .BLKL   1
                    0088            .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                              .BLKL   1
                    008C            .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                              .BLKL   1
                    0090            .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                    0090            .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                    . = 116
                    0074            .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                              .BLKB   1
                    0075            .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                              .BLKB   1
                    0076            .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                              .BLKB   1
                    0077            .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                              .BLKB   1
                    0078            .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                              .BLKL   1
0000007E  007C                              .BLKW   1
                    007E            .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                              .BLKW   1
                    0080            .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                              .BLKL   1
00000080  0084                    . = 128
00000084  0080                              .BLKL   1
                    0084            .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                              .BLKL   1
                    0088            .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                              .BLKW   1
                    008A            .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                              .BLKW   1
                    008C            .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
```

```
                        008C                .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                      .BLKW   1
                        008E                .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                      .BLKW   1
                        0090                .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                      .BLKW   1
                        0092                .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                      .BLKW   1
                        0094                .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                      .BLKB   1
                        0095                .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                      .BLKB   1
                        0096                .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                      .BLKW   1
00000096  0098                              . = 150
00000098  0096                                      .BLKW   1
                        0098                .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                      .BLKL   1
                        009C                .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                      .BLKL   1
                        00A0                .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                                      .BLKL   1
                        00A4                .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                                      .BLKW   1
                        00A6                .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                                      .BLKB   1
000000A8  00A7                                      .BLKB   1
00000074  00A8                              . = 116
00000078  0074                                      .BLKL   1
0000007C  0078                                      .BLKL   1
00000080  007C                                      .BLKL   1
00000084  0080                                      .BLKL   1
00000088  0084                                      .BLKL   1
0000008C  0088                                      .BLKL   1
                        008C                .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                      .BLKL   1
00000094  0090                                      .BLKL   1
00000098  0094                                      .BLKL   1
0000009C  0098                                      .BLKL   1
                        009C                .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                      .BLKB   1
                        009D                .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                      .BLKB   1
                        009E                .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                      .BLKB   1
000000A0  009F                                      .BLKB   1
                        00A0                .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                      .BLKB   1
                        00A1                .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                      .BLKB   1
                        00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                      .BLKB   1
000000A4  00A3                                      .BLKB   1
                        00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                      .BLKB   1
                        00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                      .BLKW   1
```

```
                    000000A8  00A7                          .BLKB   1
                              00A8          .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
                    000000AC  00A8                          .BLKL   1
                    000000B0  00AC                          .BLKL   1
                    000000B4  00B0                          .BLKL   1
                    000000B8  00B4                          .BLKL   1
                              00B8          .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
                    000000BC  00B8                          .BLKL   1
                              00BC          .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                              00BC          .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
                    00000074  00BC          . = 116
                              0074          .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
                    00000078  0074                          .BLKL   1
                              0078          .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
                    0000007C  0078                          .BLKL   1
                              007C          .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
                    0000007E  007C                          .BLKW   1
                    00000080  007E                          .BLKW   1
                              0000          .RESTORE
                              0000      67 ;
                              0000      68 ; OWN STORAGE
                              0000      69 ;
                         00000000       70          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
```

```
                     0000     72          .SBTTL  GET ONE BYTE OF DATA FROM USER BUFFER
                     0000     73 ;+
                     0000     74 ; IOC$GETBYTE - GET ONE BYTE OF DATA FROM USER'S BUFFER
                     0000     75 ;
                     0000     76 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO GET A SINGLE BYTE  FROM THE
                     0000     77 ; USER'S BUFFER.
                     0000     78 ;
                     0000     79 ; PRIOR TO CALLING THIS ROUTINE, A CALL TO IOC$INITBUFWIND MUST BE MADE TO
                     0000     80 ; MAP THE SYSTEM PAGE TABLE ENTRY INTO THE USER'S BUFFER
                     0000     81 ;
                     0000     82 ; INPUTS:
                     0000     83 ;
                     0000     84 ;       R0 = SYSTEM VIRTUAL ADDRESS OF ONE-PAGE WINDOW INTO USER'S BUFFER.
                     0000     85 ;       R5 = ADDRESS OF UCB.
                     0000     86 ;
                     0000     87 ; OUTPUTS:
                     0000     88 ;
                     0000     89 ;       R0 = UPDATED SYSTEM VIRTUAL ADDRESS
                     0000     90 ;       R1 = ONE BYTE OF DATA (ZERO EXTENDED)
                     0000     91 ;
                     0000     92 ;       UCB$L_SVAPTE IS UPDATED WHENEVER A PAGE BOUNDARY IS CROSSED
                     0000     93 ;
                     0000     94 ; THE DRIVER IS EXPECTED TO SAVE THE VALUE OF R0 FOR SUBSEQUENT CALLS.
                     0000     95 ;
                     0000     96 ;-
                     0000     97
                     0000     98
                     0000     99
                     0000    100 IOC$GETBYTE::                           ;
     51   80   90   0000    101          MOVB    (R0)+,R1                ;GET BYTE FROM USER'S BUFFER
50   01FF 8F   B3   0003    102          BITW    #^X01FF,R0              ;OVERFLOW PAGE BOUNDARY?
          06   12   0008    103          BNEQ    10$                     ;IF NEQ NO
  68 A5   04   C0   000A    104          ADDL    #4,UCB$L_SVAPTE(R5)     ;UPDATE ADDRESS OF PROCESS PTE
          49   10   000E    105          BSBB    IOC$FILSPT              ;FILL SPT, COMPUTE SYSTEM ADDRESS OF PAGE
               05   0010    106 10$:     RSB                             ;RETURN
```

```
                                0011   108          .SBTTL  PUT ONE BYTE OF DATA INTO USER'S BUFFER
                                0011   109 ;+
                                0011   110 ; IOC$PUTBYTE - PUT ONE BYTE OF DATA IN USER'S BUFFER
                                0011   111 ;
                                0011   112 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO PUT A SINGLE BYTE OF DATA
                                0011   113 ; INTO THE USER'S BUFFER.
                                0011   114 ;
                                0011   115 ; PRIOR TO CALLING THIS ROUTINE, A CALL TO IOC$INITBUFWIND MUST BE MADE TO
                                0011   116 ; MAP THE SYSTEM PAGE TABLE ENTRY INTO THE USER'S BUFFER.
                                0011   117 ;
                                0011   118 ; INPUTS:
                                0011   119 ;
                                0011   120 ;       R0 = SYSTEM VIRTUAL ADDRESS OF ONE-PAGE WINDOW INTO USER'S BUFFER
                                0011   121 ;       R1 LOW BYTE = DATA TO BE TRANSFERRED TO USER
                                0011   122 ;       R5 = ADDRESS OF UCB
                                0011   123 ;
                                0011   124 ; OUTPUTS:
                                0011   125 ;
                                0011   126 ;       R0 = UPDATED SYSTEM VIRTUAL ADDRESS OF BUFFER WINDOW
                                0011   127 ;
                                0011   128 ;       UCB$L_SVAPTE IS UPDATED WHENEVER A PAGE BOUNDARY IS CROSSED
                                0011   129 ;
                                0011   130 ; THE DRIVER IS EXPECTED TO SAVE THE VALUE OF R0 FOR SUBSEQUENT CALLS.
                                0011   131 ;
                                0011   132 ;-
                                0011   133
                                0011   134 IOC$PUTBYTE::                                ;
          80   51   90          0011   135          MOVB    R1,(R0)+                    ;PUT BYTE INTO USERS'S BUFFER
     50   01FF 8F   B3          0014   136          BITW    #^X01FF,R0                  ;OVERFLOW PAGE BOUNDARY?
               06   12          0019   137          BNEQ    10$                         ;IF NEQ NO
     68 A5     04   C0          001B   138          ADDL    #4,UCB$L_SVAPTE(R5)         ;UPDATE ADDRESS OF PROCESS PTE
               38   10          001F   139          BSBB    IOC$FILSPT                  ;FILL SPT, COMPUTE SYSTEM ADDRESS OF PAGE
               05   0021        140 10$: RSB                                            ;RETURN
```

```
                                0022          .SBTTL  INITIALIZE FOR SINGLE BYTE TRANSFERS
                                0022  ;+
                                0022  ; IOC$INITBUFWIND - INITIALIZE ONE-PAGE WINDOW INTO USER'S BUFFER
                                0022  ;
                                0022  ; THIS ROUTINE MUST BE CALLED BY A DRIVER TO SETUP THE INITIAL ONE-
                                0022  ; PAGE WINDOW INTO A USER'S BUFFER BEFORE CALLING IOC$GETBYTE OR
                                0022  ; IOC$PUTBYTE.
                                0022  ;
                                0022  ; INPUTS:
                                0022  ;
                                0022  ;       R5 = ADDRESS OF UCB
                                0022  ;
                                0022  ; OUTPUTS:
                                0022  ;
                                0022  ;       R0 = SYSTEM VIRTUAL ADDRESS OF WINDOW INTO USER'S BUFFER
                                0022  ;-
                                0022
                                0022  IOC$INITBUFWIND::                                ;
                     35   10    0022          BSBB    IOC$FILSPT                       ;FILL SPT, COMPUTE VIRTUAL ADDRESS OF PAGE
          50   6C A5   A8    0024          BISW    UCB$W_BOFF(R5),R0                ;MERGE BYTE OFFSET INTO ADDRESS
                       05    0028          RSB                                      ;
```

```
                                0029   164          .SBTTL  MOVE FROM USER BUFFER
                                0029   165 ;+
                                0029   166 ; IOC$MOVFRUSER - MOVE FROM USER BUFFER
                                0029   167 ;
                                0029   168 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO MOVE A STRING FROM A USER
                                0029   169 ; BUFFER TO AN INTERNAL BUFFER.
                                0029   170 ;
                                0029   171 ; INPUTS:
                                0029   172 ;
                                0029   173 ;       R1 = ADDRESS OF INTERNAL BUFFER.
                                0029   174 ;       R2 = NUMBER OF BYTES TO BE MOVED.
                                0029   175 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0029   176 ;
                                0029   177 ; OUTPUTS:
                                0029   178 ;
                                0029   179 ;       ***TBS***
                                0029   180 ;-
                                0029   181
                                0029   182          .ENABLE LSB
                                0029   183 IOC$MOVFRUSER::                          ;MOVE FROM USER BUFFER
                      F7   10   0029   184          BSBB    IOC$INITBUFWIND        ;SETUP WINDOW INTO BUFFER
                      0D   11   002B   185          BRB     20$                    ;
                                002D   186 IOC$MOVFRUSER2::
              50  01FF   8F   B3   002D   187 10$:    BITW    #^X01FF,R0             ;OVERFLOW PAGE BOUNDRY?
                      06   12   0032   188          BNEQ    20$                    ;IF NEQ NO
                68 A5   04   C0   0034   189          ADDL    #4,UCB$L_SVAPTE(R5)    ;UPDATE ADDRESS OF USER PTE
                      1F   10   0038   190          BSBB    IOC$FILSPT             ;FILL SYSTEM PTE WITH PROPER RELOCATION
                                003A   191 IOC$MOVFRUSER1::
                   81   80   90   003A   192 20$:    MOVB    (R0)+,(R1)+            ;MOVE BYTE TO INTERNAL BUFFER
                     ED 52   F5   003D   193          SOBGTR  R2,10$                 ;ANY MORE BYTES TO MOVE?
                      05   0040   194          RSB                            ;
                           0041   195          .DSABL  LSB
```

```
                        0041   197          .SBTTL  MOVE TO USER BUFFER
                        0041   198 ;+
                        0041   199 ; IOC$MOVTOUSER - MOVE TO USER BUFFER
                        0041   200 ;
                        0041   201 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO MOVE A STRING FROM AN INTERNAL
                        0041   202 ; BUFFER TO A USER BUFFER.
                        0041   203 ;
                        0041   204 ; INPUTS:
                        0041   205 ;
                        0041   206 ;       R1 = ADDRESS OF INTERNAL BUFFER.
                        0041   207 ;       R2 = NUMBER OF BYTES TO BE MOVED.
                        0041   208 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                        0041   209 ;
                        0041   210 ; OUTPUTS:
                        0041   211 ;
                        0041   212 ; ***TBS***
                        0041   213 ;-
                        0041   214
                        0041   215          .ENABLE LSB
                        0041   216 IOC$MOVTOUSER::                              ;MOVE TO USER BUFFER
              DF   10   0041   217          BSBB    IOC$INITBUFWIND             ;INITIALIZE WINDOW INTO BUFFER
              0D   11   0043   218          BRB     20$                         ;
                        0045   219 IOC$MOVTOUSER2::
        50  01FF 8F   B3  0045   220 10$:    BITW    #^X01FF,R0                  ;OVERFLOW PAGE BOUNDRY?
              06   12   004A   221          BNEQ    20$                         ;IF NEQ NO
          68 A5   04   C0  004C   222          ADDL    #4,UCB$L_SVAPTE(R5)         ;UPDATE ADDRESS OF USER PTE
              07   10   0050   223          BSBB    IOC$FILSPT                  ;FILL SYSTEM PTE WITH PROPER RELOCATION
                   0052   224 IOC$MOVTOUSER1::
        80    81   90   0052   225 20$:     MOVB    (R1)+,(R0)+                 ;MOVE BYTE TO USER BUFFER
          ED 52   F5   0055   226          SOBGTR  R2,10$                      ;ANY MORE BYTES TO MOVE?
              05   0058   227          RSB                                 ;
                   0059   228          .DSABL  LSB
```

```
                                0059   230          .SBTTL  FILL SYSTEM PTE WITH TRANSFER PTE
                                0059   231 ;+
                                0059   232 ; IOC$FILSPT - FILL SYSTEM PTE WITH TRANSFER PTE
                                0059   233 ;
                                0059   234 ; This routine extracts the PFN from current PTE and creates a virtual
                                0059   235 ; address from it.
                                0059   236 ;
                                0059   237 ; INPUTS:
                                0059   238 ;
                                0059   239 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0059   240 ;
                                0059   241 ; OUTPUTS:
                                0059   242 ;
                                0059   243 ;       R0 = SYSTEM VIRTUAL ADDRESS OF START OF PAGE CONTAINING THE BUFFER.
                                0059   244 ;
                                0059   245 ;       REGISTERS R1, R2, AND R3 ARE PRESERVED ACROSS CALL.
                                0059   246 ;-
                                0059   247
                                0059   248 IOC$FILSPT::                                ;FILL SYSTEM PTE WITH TRANSFER PTE
     50    68 B5    15    00    EF   0059   249          EXTZV   #PTE$V_PFN, #PTE$S_PFN, -
                                005F   250                   @UCB$L_SVAPTE(R5), R0   ; Get PFN of user buffer
          50    50    09    78   005F   251          ASHL    #9, R0, R0              ; Re-make virtual address
                                05   0063   252          RSB                             ;
                                0064   253
                                0064   254          .END
```

IOC$FILSPT	00000059	RG	D	02	UCB$L_DPC	00000080	D	UCB$W_FUNC	0000007E	D
IOC$GETBYTE	00000000	RG	D	02	UCB$L_DUETIM	0000005C	D	UCB$W_MB_SEED	00000000	D
IOC$INITBUFWIND	00000022	RG	D	02	UCB$L_DX_BFPNT	0000009C	D	UCB$W_MSGCNT	00000016	D
IOC$MOVFRUSER	00000029	RG	D	02	UCB$L_DX_BUF	00000098	D	UCB$W_MSGMAX	00000014	D
IOC$MOVFRUSER1	0000003A	RG	D	02	UCB$L_DX_RXDB	000000A0	D	UCB$W_NT_CHAN	0000007C	D
IOC$MOVFRUSER2	0000002D	RG	D	02	UCB$L_EMB	00000078	D	UCB$W_OFFSET	0000008A	D
IOC$MOVTOUSER	00000041	RG	D	02	UCB$L_FIRST	00000014	D	UCB$W_REFC	00000050	D
IOC$MOVTOUSER1	00000052	RG	D	02	UCB$L_FPC	0000000C	D	UCB$W_SIZE	00000008	D
IOC$MOVTOUSER2	00000045	RG	D	02	UCB$L_FQBL	00000004	D	UCB$W_SRCADDR	0000001A	D
IOC$PUTBYTE	00000011	RG	D	02	UCB$L_FQFL	00000000	D	UCB$W_STS	00000058	D
PTE$S_PFN	= 00000015		D		UCB$L_FR3	00000010	D	UCB$W_TT_DESIZE	000000A5	D
PTE$V_PFN	= 00000000		D		UCB$L_FR4	00000014	D	UCB$W_UNIT	00000048	D
SIZ...	= 00000002		D		UCB$L_IOQBL	00000044	D	UCB$W_VPROT	0000001A	D
UCB$B_AMOD	00000053		D		UCB$L_IOQFL	00000040	D			
UCB$B_CEX	00000077		D		UCB$L_IRP	0000004C	D			
UCB$B_CM1	0000004A		D		UCB$L_LINK	0000002C	D			
UCB$B_CM2	0000004B		D		UCB$L_LOGADR	00000064	D			
UCB$B_DEVCLASS	00000038		D		UCB$L_MAXBLOCK	00000084	D			
UCB$B_DEVTYPE	00000039		D		UCB$L_MB_MBX	0000007C	D			
UCB$B_DIPL	00000052		D		UCB$L_MB_PORT	0000008C	D			
UCB$B_DX_SCTCNT	000000A6		D		UCB$L_MB_RAST	00000078	D			
UCB$B_ERTCNT	00000070		D		UCB$L_MB_SHB	00000080	D			
UCB$B_ERTMAX	00000071		D		UCB$L_MB_WAST	00000074	D			
UCB$B_FEX	00000076		D		UCB$L_MB_WIOQBL	00000088	D			
UCB$B_FIPL	0000000B		D		UCB$L_MB_WIOQFL	00000084	D			
UCB$B_LOCSRV	0000003C		D		UCB$L_MEDIA	0000008C	D			
UCB$B_OFFNDX	00000094		D		UCB$L_NT_DATSSB	00000074	D			
UCB$B_OFFRTC	00000095		D		UCB$L_NT_INTSSB	00000078	D			
UCB$B_REMSRV	0000003D		D		UCB$L_OPCNT	00000060	D			
UCB$B_SECTORS	0000003C		D		UCB$L_OWNUIC	0000001C	D			
UCB$B_SLAVE	00000074		D		UCB$L_PID	00000028	D			
UCB$B_SPR	00000075		D		UCB$L_RQBL	00000004	D			
UCB$B_STATE	00000052		D		UCB$L_RQFL	00000000	D			
UCB$B_TRACKS	0000003D		D		UCB$L_SVAPTE	00000068	D			
UCB$B_TT_CRFILL	0000009D		D		UCB$L_SVPN	00000064	D			
UCB$B_TT_DECRF	000000A1		D		UCB$L_TT_DECHAR	000000A8	D			
UCB$B_TT_DELFF	000000A2		D		UCB$L_TT_RDUE	0000008C	D			
UCB$B_TT_DESPEE	000000A0		D		UCB$L_TT_RTIMOU	000000B8	D			
UCB$B_TT_DETYPE	000000A4		D		UCB$L_VCB	00000030	D			
UCB$B_TT_LFFILL	0000009E		D		UCB$T_PARTNER	0000000C	D			
UCB$B_TT_SPEED	0000009C		D		UCB$W_BCNT	0000006E	D			
UCB$B_TYPE	0000000A		D		UCB$W_BCR	00000096	D			
UCB$B_VERTSZ	0000003F		D		UCB$W_BOFF	0000006C	D			
UCB$C_LENGTH	00000074		D		UCB$W_BUFQUO	00000018	D			
UCB$C_MB_LENGTH	00000090		D		UCB$W_BYTESTOGO	0000003E	D			
UCB$C_TT_LENGTH	000000BC		D		UCB$W_CHARGE	0000004A	D			
UCB$K_LENGTH	00000074		D		UCB$W_CYLINDERS	0000003E	D			
UCB$K_MB_LENGTH	00000090		D		UCB$W_DA	0000008C	D			
UCB$K_TT_LENGTH	000000BC		D		UCB$W_DC	0000008E	D			
UCB$L_AMB	00000054		D		UCB$W_DEVBUFSIZ	0000003A	D			
UCB$L_ASTQBL	00000010		D		UCB$W_DEVSTS	0000005A	D			
UCB$L_ASTQFL	0000000C		D		UCB$W_DIRSEQ	00000088	D			
UCB$L_CPID	0000005C		D		UCB$W_DSTADDR	00000018	D			
UCB$L_CRB	00000020		D		UCB$W_DX_BCR	000000A4	D			
UCB$L_DDB	00000024		D		UCB$W_EC1	00000090	D			
UCB$L_DEVCHAR	00000034		D		UCB$W_EC2	00000092	D			
UCB$L_DEVDEPEND	0000003C		D		UCB$W_ERRCNT	00000072	D			

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000064 (100.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
IOC$FILSPT	00000059-R	248 (1)	#-105 (1) #-139 (1) #-160 (1) #-190 (1) #-223 (1)
IOC$GETBYTE	00000000-R	100 (1)	
IOC$INITBUFWIND	00000022-R	159 (1)	#-184 (1) #-217 (1)
IOC$MOVFRUSER	00000029-R	183 (1)	
IOC$MOVFRUSER1	0000003A-R	191 (1)	
IOC$MOVFRUSER2	0000002D-R	186 (1)	
IOC$MOVTOUSER	00000041-R	216 (1)	
IOC$MOVTOUSER1	00000052-R	224 (1)	
IOC$MOVTOUSER2	00000045-R	219 (1)	
IOC$PUTBYTE	00000011-R	134 (1)	
PTE$S_PFN	=00000015		#-249 (1)
PTE$V_PFN	=00000000		#-249 (1)
UCB$L_SVAPTE	00000068		#-104 (1) #-138 (1) #-189 (1) #-222 (1) 250 (1)
UCB$W_BOFF	0000006C		#-161 (1)

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEFINI         1           64     (1)      64     (1)      65     (1)      66     (1)
$PRDEF          5           64     (1)      64     (1)
$PTEDEF         3           65     (1)      65     (1)
$UCBDEF         10          66     (1)      66     (1)
```

```
                                    +-------------------------+
                                    ! Opcode Cross Reference !
                                    +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDL            00C0      104    (1)    138    (1)    189    (1)    222    (1)
ASHL            0078      251    (1)
BISW            00A8      161    (1)
BITW            00B3      102    (1)    136    (1)    187    (1)    220    (1)
BNEQ            0012      103    (1)    137    (1)    188    (1)    221    (1)
BRB             0011      185    (1)    218    (1)
BSBB            0010      105    (1)    139    (1)    160    (1)    184    (1)    190    (1)    217    (1)    223    (1)
EXTZV           00EF      249    (1)
MOVB            0090      101    (1)    135    (1)    192    (1)    225    (1)
RSB             0005      106    (1)    140    (1)    162    (1)    194    (1)    227    (1)    252    (1)
SOBGTR          00F5      193    (1)    226    (1)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE      REFERENCES...
---------      -------------
.DSABL         195    (1)    228    (1)
.ENABLE        182    (1)    215    (1)
.END           254    (1)
.ENDM          64     (1)    65     (1)    66     (1)
.IDENT         7      (1)
.NOCROSS       64     (1)    65     (1)    66     (1)
.PAGE          107    (1)    141    (1)    163    (1)    196    (1)    229    (1)    71     (1)
.PSECT         70     (1)
.RESTORE       64     (1)    65     (1)    66     (1)
.SAVE          64     (1)    65     (1)    66     (1)
.SBTTL         108    (1)    142    (1)    164    (1)    197    (1)    230    (1)    72     (1)
.TITLE         6      (1)
```

```
                                        +---------------------------+
                                        ! Register Cross Reference !
                                        +---------------------------+

REGISTER  NO. REFERENCES     REFERENCES...
--------  --------------     -------------
R0                12         #-101    (1)    #-102    (1)    #-135    (1)    #-136    (1)    #-161    (1)    #-187    (1)
                             #-192    (1)    #-220    (1)    #-225    (1)    #-250    (1)    #-251    (1)
R1                 4         #-101    (1)    #-135    (1)    #-192    (1)    #-225    (1)
R2                 2         #-193    (1)    #-226    (1)
R5                 6         #-104    (1)    #-138    (1)    #-161    (1)    #-189    (1)    #-222    (1)      250    (1)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     23    00:00:00.04    00:00:00.21
Command processing                889    00:00:00.23    00:00:00.81
Pass 1                            438    00:00:01.02    00:00:01.92
Symbol table sort                   0    00:00:00.10    00:00:00.09
Pass 2                             34    00:00:00.30    00:00:00.50
Symbol table output                 2    00:00:00.02    00:00:00.11
Psect synopsis output               2    00:00:00.01    00:00:00.01
Cross-reference output              4    00:00:00.03    00:00:00.09
Assembler run totals             1393    00:00:01.75    00:00:03.74

The working set limit was 2012 pages.
54476 bytes (107 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 352 non-local and 6 local symbols.
254 source lines were read in Pass 1, producing 31 object records in Pass 2.
25 pages of virtual memory were used to define 12 macros.

                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8              1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           1
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       6
TOTALS (all libraries)                                 8

529 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:BUFCTL.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 81 DECLARATIONS
(2) 115 GET A VERTUAL MEMORY BUFFER.
(4) 216 RELEASE A MEMORY BUFFER.

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element BUFFER.MAR
0000     2 ; *6    17-JUL-1986 11:06:39 MARTIN "FIXED"
0000     3 ; *5     9-JUL-1986 10:47:15 MARTIN "<no reason given>"
0000     4 ; *4     8-JUL-1986 15:51:33 MARTIN "IT WORKS."
0000     5 ; *3     2-JUN-1986 07:29:21 ANDELLA "<no reason given>"
0000     6 ; *2     7-MAY-1986 17:14:11 BARANSKI "Documentation Check."
0000     7 ; *1     6-FEB-1986 15:13:57 DS ""
0000     8 ; DEC/CMS REPLACEMENT HISTORY, Element BUFFER.MAR
0000     9         .TITLE  BUFFER  *** BUFFER GETBUF/RELBUF routines
0000    10         .IDENT  "10.2-10"                                          ;G:6
0000    11         .LIST   MEB
0000    12         .NLIST  CND
0000    13
0000    14 ;
0000    15 ; COPYRIGHT (C) 1977, 1980, 1983, 1985, 1986                       ;G:3
0000    16 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    17 ;
0000    18 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    19 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    20 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    21 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    22 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    23 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    24 ; REMAIN IN DEC.
0000    25 ;
0000    26 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    27 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    28 ; CORPORATION.
0000    29 ;
0000    30 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    31 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    32
0000    33 ;++
0000    34 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    35 ;
0000    36 ; ABSTRACT:
0000    37 ;
0000    38 ; ENVIRONMENT:
0000    39 ;
0000    40 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    41 ;
0000    42 ; MODIFIED BY:
0000    43 ;               KEN CHAPMAN     24-MAR-78       VERSION 02 (ESSAA-4.01).
0000    44 ;       01      ADDED 'REGION' ARGUMENT TO "GETBUF" AND "RELBUF".
0000    45 ;                                                                  ;G:2
0000    46 ;       02      PUT PAGE COUNT CHECK IN "RELBUF".
0000    47 ;
0000    48 ;       03      - Jack Stansbury, 21-Oct-1981, Version 6.-
0000    49 ;               Added .LIBRARY statements for $DIAG and $DS.
0000    50 ;
0000    51 ;       04      Bob Bergazzi    May 16,1983     Version 6.11
0000    52 ;               Changed the order of .LIB statements.
0000    53 ;
0000    54 ;       05      Richard Brown   12-August-83    Version 6.13
0000    55 ;               Changed $DS_RELBUF so that it will ignore DS$_FRAGBUF
0000    56 ;               errors returned from $CNTREG.  Eliminates problem of
0000    57 ;               RELBUF returning an error if part of the buffer space
```

```
0000     58 ;                  being deallocated is above the VDS and part is below.
0000     59 ;
0000     60 ;          06      Bob Bergazzi     21-Oct-1983     Version 6.13
0000     61 ;                  Added # sign to SS$_NORMAL in edit number 05.
0000     62 ;
0000     63 ;          07      Domenic Andella 29-Jan-1985     Version 8.1
0000     64 ;                  Add two entry points, DSX$GETSYSBUF and DSX$RELSYSBUF
0000     65 ;                  to support new services.
0000     66 ;
0000     67 ;          08      M. Baggett       23-May-1985     Version 8.2
0000     68 ;                  Fixed trucation errors caused by adding module ULTRIX.
0000     69 ;
0000     70 ;          09      Domenic Andella 29-May-1986     Version 10.2                ;G:3
0000     71 ;                  Modify routines DSX$GETBUF and DSX$RELBUF to check that     ;G:3
0000     72 ;                  the _PAGCNT is GTR zero, else return _ILLPAGCNT.            ;G:3
0000     73 ;                                                                              ;G:3
0000     74 ;          10      I. Martin/S. Meier  09-Jul-1986 Version 10.2                ;G:6
0000     75 ;                  Modify Rel_Common to handle case when net amount of pages   ;G:6
0000     76 ;                  to be deallocated, using DSX$RELBUF, is equal to zero.      ;G:6
0000     77 ;                  If it is zero, then return _ILLPAGCNT.                      ;G:6
0000     78 ;                                                                              ;G:3
0000     79 ;--
```

```
                     0000     81          .SBTTL  DECLARATIONS
                     0000     82 ;
                     0000     83 ; INCLUDE FILES:
                     0000     84 ;
                     0000     85          .LIBRARY        /$DS/                   ;                               [4]
                     0000     86          .LIBRARY        /$DIAG/                 ;                               [4]
                     0000     87
                     0000     88 ;
                     0000     89 ; MACROS:
                     0000     90 ;
                     0000     91
                     0000     92 ;
                     0000     93 ; EQUATED SYMBOLS:
                     0000     94 ;
                     0000     95          $DS_DSADEF
                     0000     96          $DS_DSDEF
                     0000     97          $DS_GETBUF_DEF
                     0000     98          $DS_RELBUF_DEF
                     0000     99
                     0000    100 ;
                     0000    101 ; CONSTANTS:
                     0000    102 ;
                     0000    103
00200000             0000    104          PTE$K_INTERNAL_VDS = ^X200000   ; Sets PTE<21> for $DS_GETSYSBUF   [7]
00000000             0000    105          PTE$K_EXTERNAL_USER = 0         ; Clears PTE<21> for $DS_GETBUF    [7]
00000010             0000    106          RELSYSBUF$_START_VA = ^X10      ; AP offset for starting VA used in DS$_RELS
                     0000    107
                     0000    108 ; OWN STORAGE:
                     0000    109 ;
                 00000000    110          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
00000008             0000    111 DS$GQ_PHYADR::  .BLKL   2               ; TEMPORARY PHYADR.
00000018             0008    112 DS$GL_BUFCNT::  .BLKL   4               ; KEEPS TRACK OF BUFFER ALLOCATIONS.
00000020             0018    113 Q_RETADR:       .BLKL   2               ; TEMPORARY RETADR.
```

```
                     0020   115          .SBTTL  GET A VERTUAL MEMORY BUFFER.
                 00000000   116          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000   117 ;++
                     0000   118 ; FUNCTIONAL DESCRIPTION:
                     0000   119 ;
                     0000   120 ;       GET A MEMORY BUFFER IS FUNCTIONALLY SIMILAR TO STARTLET $EXPREG.
                     0000   121 ;       IT ALLOCATES MEMORY FOR THE PROGRAM TO USE AS BUFFER.
                     0000   122 ;       IN THE REPAIR ENVIRONMENT, THE PHYSICAL MEMORY IS GUARANTEED TO
                     0000   123 ;       BE CONTIGUOUS.
                     0000   124 ;
                     0000   125 ;
                     0000   126 ; SPECIAL USAGE:
                     0000   127 ;       If this routine is called via "$DS_GETSYSBUF" from internal
                     0000   128 ;       VDS code, a PTE code argument is pushed which results in
                     0000   129 ;       each allocated page's PTE<21> being set.
                     0000   130 ;
                     0000   131 ; CALLING SEQUENCE:
                     0000   132 ;
                     0000   133 ;       STANDARD PROCEDURE CALL.  ($DS_GETBUF or $DS_GETSYSBUF)
                     0000   134 ;
                     0000   135 ; INPUT PARAMETERS:
                     0000   136 ;
                     0000   137 ;       GETBUF$_PAGCNT(AP) =    NUMBER OF PAGES TO BE ALLOCATED.
                     0000   138 ;       GETBUF$_RETADR(AP) =    ADDRESS OF TWO LONGWORD ARRAY TO RECEIVE
                     0000   139 ;                               VIRTUAL LIMITS.
                     0000   140 ;       GETBUF$_PHYADR(AP) =    ADDRESS OF TWO LONGWORD ARRAY TO RECEIVE
                     0000   141 ;                               PHYSICAL LIMITS.
                     0000   142 ;       GETBUF$_REGION(AP) =    REGION INDICATOR:
                     0000   143 ;                                       0 = PROGRAM (P0) REGION
                     0000   144 ;                                       1 = CONTROL (P1) REGION
                     0000   145 ;                                       2 = SYSTEM REGION.
                     0000   146 ;
                     0000   147 ; IMPLICIT INPUTS:
                     0000   148 ;
                     0000   149 ;       NONE
                     0000   150 ;
                     0000   151 ; OUTPUT PARAMETERS:
                     0000   152 ;
                     0000   153 ;       NONE
                     0000   154 ;
                     0000   155 ; IMPLICIT OUTPUTS:
                     0000   156 ;
                     0000   157 ;       NONE
                     0000   158 ;
                     0000   159 ; COMPLETION CODES:
                     0000   160 ;
                     0000   161 ;       SEE $EXPREG (IN MEMMGT MODULE).
                     0000   162 ;
                     0000   163 ; SIDE EFFECTS:
                     0000   164 ;
                     0000   165 ;       NONE
                     0000   166 ;
                     0000   167 ;--
```

```
                          001C  0000   169 .ENTRY  DSX$GETSYSBUF,^M<R2,R3,R4>      ; Registers to save.
          54   00200000 8F    D0  0002   170        MOVL    #PTE$K_INTERNAL_VDS,R4  ; Code to mark PTE<21>=1 for $DS_GETSYSBUF
                       05    11  0009   171        BRB     GET_COMMON              ; Branch to common
                                 000B   172
                          001C  000B   173 .ENTRY  DSX$GETBUF,^M<R2,R3,R4>         ; Registers to save
                    54    00    D0  000D   174        MOVL    #PTE$K_EXTERNAL_USER,R4 ; Set PTE<21> =0 for $DS_GETBUF
                                 0010   175
                                 0010   176 GET_COMMON:                                    ;[7]                    ;G:3
                                 0010   177                                                                      ;G:3
                       04 AC     D5  0010   178        TSTL    GETBUF$_PAGCNT(AP)      ; Check the page count.     [9]  ;G:3
                          08     14  0013   179        BGTR    10$                     ; Branch if count GTR 0.    [9]  ;G:3
                50    0000'8F   3C  0015   180        MOVZWL  #SS$_ILLPAGCNT,R0       ; Bad page count, rtn error. [9]  ;G:3
                        0052    31  001A   181        BRW     50$                     ; Exit with error status    [9]  ;G:3
                                 001D   182                                                                      ;G:3
          53    10 AC   02    00  EF  001D   183 10$:    EXTZV   #0, #2, GETBUF$_REGION(AP), -;                  [7][9] ;G:3
                                 0023   184                 R3                      ; REGION AVAILABLE.
                    50    7E    7E  0023   185        MOVAQ   -(SP),R0                ; Point to QUADWORD for start,end
                             54    DD  0026   186        PUSHL   R4                      ; Pass code to mark PTE<21>
                             53    DD  0028   187        PUSHL   R3                      ; Pass region code
                             00    DD  002A   188        PUSHL   #0                      ;
                             60    7F  002C   189        PUSHAQ  (R0)                    ; Pass locations to store VA limits
                          04 AC    DD  002E   190        PUSHL   GETBUF$_PAGCNT(AP)      ; Pass page count
              00000000'GF    05    FB  0031   191        CALLS   #5,G^SYS$EXPREG         ; Go get a block of virtual memory
                          0C AC    D5  0038   192        TSTL    GETBUF$_PHYADR(AP)      ; CHECK PHYSICAL ADDRESS ARG.
                             08    13  003B   193        BEQL    20$
           0C BC    00000000'EF   7D  003D   194        MOVQ    L^DS$GQ_PHYADR, -       ; COPY ADDRESS ARRAY.
                                 0045   195                  aGETBUF$_PHYADR(AP)
                                 0045   196
                    51    8E    7D  0045   197 20$:    MOVQ    (SP)+,R1                ; Get first and last address
                    52    51    D1  0048   198        CMPL    R1,R2                   ; First really smaller?
                             07    15  004B   199        BLEQ    30$                     ; Branch if so
                             52    DD  004D   200        PUSHL   R2                      ; Save smaller
                    52    51    D0  004F   201        MOVL    R1,R2                   ; copy larger to higher register
                             02    BA  0052   202        POPR    #^M<R1>                 ; Put smaller in lower register
                                 0054   203
                          08 AC    D5  0054   204 30$:    TSTL    GETBUF$_RETADR(AP)      ; CHECK FOR RETURN WANTED.
                             04    13  0057   205        BEQL    40$                     ; SKIP IF NOT.
                08 BC    51    7D  0059   206        MOVQ    R1, aGETBUF$_RETADR(AP) ; STORE BUFFER ADDRESSES.
                                 005D   207
                    52    51    C2  005D   208 40$:    SUBL    R1,R2                   ; Compute buffer size
                             52    D6  0060   209        INCL    R2                      ; make it a count [10]          ;G:6
          52    52    F7    8F    78  0062   210        ASHL    #-9, R2, R2             ; IN PAGES. [10]                ;G:6
00000008'EF43    52    C0  0067   211        ADDL2   R2,L^DS$GL_BUFCNT[R3]   ; KEEP TRACK.
                                 006F   212
                       FF8E'     30  006F   213 50$:    BSBW    KB_CHECK                ; ALLOW ^C.     [9]             ;G:3
                                 04  0072   214        RET
```

```
                        0073   216          .SBTTL  RELEASE A MEMORY BUFFER.
                        0073   217 ;++
                        0073   218 ; FUNCTIONAL DESCRIPTION:
                        0073   219 ;
                        0073   220 ;       This routine releases memory allocated via a $DS_GETBUF or $DS_GETSYSBUF
                        0073   221 ;
                        0073   222 ; CALLING SEQUENCE:
                        0073   223 ;
                        0073   224 ;       STANDARD PROCEDURE CALL ($DS_RELBUF or $DS_RELSYSBUF)
                        0073   225 ;
                        0073   226 ; INPUT PARAMETERS:
                        0073   227 ;
                        0073   228 ;       RELBUF$_PAGCNT(AP) =    NUMBER OF PAGES TO BE ALLOCATED.
                        0073   229 ;       RELBUF$_RETADR(AP) =    ADDRESS OF TWO LONGWORD ARRAY TO RECEIVE
                        0073   230 ;                               VIRTUAL LIMITS.
                        0073   231 ;       RELBUF$_REGION(AP) =    REGION INDICATOR:
                        0073   232 ;                                       0 = PROGRAM (P0) REGION
                        0073   233 ;                                       1 = CONTROL (P1) REGION
                        0073   234 ;                                       2 = SYSTEM REGION.
                        0073   235 ;
                        0073   236 ;       RELSYSBUF$_START_VA(AP) = STARTING VIRTUAL ADDRESS OF FIRST PAGE TO BE DEALL
                        0073   237 ;
                        0073   238 ; IMPLICIT INPUTS:
                        0073   239 ;
                        0073   240 ;       NONE
                        0073   241 ;
                        0073   242 ; OUTPUT PARAMETERS:
                        0073   243 ;
                        0073   244 ;       NONE
                        0073   245 ;
                        0073   246 ; IMPLICIT OUTPUTS:
                        0073   247 ;
                        0073   248 ;       NONE
                        0073   249 ;
                        0073   250 ; COMPLETION CODES:
                        0073   251 ;
                        0073   252 ;       SEE $CNTREG (IN MEMMGT MODULE).
                        0073   253 ;
                        0073   254 ; SIDE EFFECTS:
                        0073   255 ;
                        0073   256 ;       NONE
                        0073   257 ;
                        0073   258 ;--
```

```
                        000C  0073  .ENTRY  DSX$RELSYSBUF,^M<R2,R3>                  ; ENTRY MASK              [7]
          53   10 AC     D0  0075          MOVL    RELSYSBUF$_START_VA(AP),R3        ; PASS STARTING VA        [7]
                  04     11  0079          BRB     REL_COMMON                        ; BRANCH TO COMMON        [7]
                             007B
                        000C  007B  .ENTRY  DSX$RELBUF,^M<R2,R3>                     ; ENTRY MASK.             [7]
                  53     D4  007D          CLRL    R3                                ; INDICATES $DS_RELBUF    [7]
                             007F
                             007F  REL_COMMON:                                                                  ;[7]
                             007F                                                                                  ;G:3
  52   0C AC   02   00   EF  007F          EXTZV   #0,#2,RELBUF$_REGION(AP), R2      ; REGION AVAILABLE. [7][9][10] ;G:6
          51   04 AC     D0  0085          MOVL    RELBUF$_PAGCNT(AP), R1            ; USE CALLER PAGE COUNT.[10]  ;G:6
    51   00000008'EF42   D1  0089          CMPL    L^DS$GL_BUFCNT[R2], R1            ; CHECK FOR TOO LARGE PAGCNT.[10] ;G
                  08     18  0091          BGEQ    10$                               ; SKIP IF NOT TOO LARGE.[10]  ;G:6
    51   00000008'EF42   D0  0093          MOVL    L^DS$GL_BUFCNT[R2], R1            ; GET MAXIMUM PAGE COUNT.[10] ;G:6
                             009B                                                                                  ;G:3
                  51     D5  009B  10$:    TSTL    R1                                ; zero pages to be deallocated? [10]
                  08     14  009D          BGTR    20$                               ; yes, exit with error status [10] ;
        50   0000'8F     3C  009F          MOVZWL  #SS$_ILLPAGCNT,R0                 ; Bad page count, rtn error.    [9][
                0053     31  00A4          BRW     40$                               ; Exit with error status        [9][
                             00A7
            50   7E      7E  00A7  20$:    MOVAQ   -(SP),R0                          ; Address of desc
                  53     DD  00AA          PUSHL   R3                                ; PASS STARTING VA IF DS$_RELSYSBUF
                  52     DD  00AC          PUSHL   R2                                ; PASS REGION
                  00     DD  00AE          PUSHL   #0                                ;
                  60     7F  00B0          PUSHAQ  (R0)                              ; PASS ADDRESS TO STORE VA
                  51     DD  00B2          PUSHL   R1                                ; PASS NUMBER OF PAGES
    00000000'GF   05     FB  00B4          CALLS   #5,G^SYS$CNTREG                   ; GO RELEASE A BLOCK OF VIRTUAL MEMO
    00660080 8F   50     D1  00BB          CMPL    R0,#DS$_FRAGBUF                   ; IF FRAGBUF error code [5]
                  07     12  00C2          BNEQ    25$                               ; THEN                  [5]
    50   00000000'8F     D0  00C4          MOVL    #SS$_NORMAL,R0                    ;    ignore it.         [06]
                             00CB  25$:                                              ;                       [5]
              08 AC      D5  00CB          TSTL    RELBUF$_RETADR(AP)                ; CHECK FOR RETURN WANTED.
                  04     13  00CE          BEQL    30$                               ; SKIP IF NOT.
        08 BC   6E       7D  00D0          MOVQ    (SP), @RELBUF$_RETADR(AP)         ; STORE BUFFER ADDRESSES.
                             00D4
        51   8E   8E     C3  00D4  30$:    SUBL3   (SP)+,(SP)+, R1                   ; COMPUTE BUFFER SIZE.
                  03     18  00D8          BGEQ    35$                               ; Branch if positive
            51   51      CE  00DA          MNEGL   R1,R1                             ; Make it positive
                             00DD
                  51     D6  00DD  35$:    INCL    R1                                ; MAKE INTO COUNT.[11]        ;G:6
    51   51   F7 8F      78  00DF          ASHL    #-9, R1, R1                       ; IN PAGES.[11]               ;G:6
  00000008'EF42   51     C2  00E4          SUBL2   R1, L^DS$GL_BUFCNT[R2]            ; KEEP TRACK.
              0B 50      E9  00EC          BLBC    R0, 40$                           ; SKIP IF ALREADY ERROR.
        04 AC   51       D1  00EF          CMPL    R1, RELBUF$_PAGCNT(AP)            ; CHECK IF TOTAL SUCCESS.
                  05     13  00F3          BEQL    40$                               ; SKIP IF OK.
        50   0000'8F     3C  00F5          MOVZWL  #SS$_PAGOWNVIO, R0                ; ERROR CODE.
                             00FA
                FF03'    30  00FA  40$:    BSBW    KB_CHECK                          ; ALLOW ^C.
                  04     00FD              RET
                             00FE
                             00FE          .END
```

```
$$ARGS                          = 00000003     D
$$T1                            = 00000010     D
BIT...                          = 00660160     D
DS$GL_BUFCNT                      00000008 RG D  02
DS$GQ_PHYADR                      00000000 RG D  02
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$_AP_NORMAL_BREAK             = 00660150     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INIT_FAIL                   = 00660140     D
DS$_INSFMEM                     = 00660050     D
DS$_INVCPU                      = 00660130     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MEM_ALLOC_ERR               = 00660138     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTALLOWMP                  = 00660148     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_UNSUPPDEV                   = 00660158     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSX$GETBUF                        0000000B RG D  03
DSX$GETSYSBUF                     00000000 RG D  03
DSX$RELBUF                        0000007B RG D  03
DSX$RELSYSBUF                     00000073 RG D  03
GETBUF$_NARGS                   = 00000004     D
GETBUF$_PAGCNT                  = 00000004     D
GETBUF$_PHYADR                  = 0000000C     D
GETBUF$_REGION                  = 00000010     D
GETBUF$_RETADR                  = 00000008     D
GET_COMMON                        00000010 R   D  03
KB_CHECK                          ********     X  03
PTE$K_EXTERNAL_USER             = 00000000     D
PTE$K_INTERNAL_VDS              = 00200000     D
Q_RETADR                          00000018 R   D  02
RELBUF$_NARGS                   = 00000003     D
RELBUF$_PAGCNT                  = 00000004     D
RELBUF$_REGION                  = 0000000C     D
RELBUF$_RETADR                  = 00000008     D
RELSYSBUF$_START_VA             = 00000010     D
REL_COMMON                        0000007F R   D  03
SIZ...                          = 00000001     D
SS$_ILLPAGCNT                     ********     X  03
SS$_NORMAL                        ********     X  03
SS$_PAGOWNVIO                     ********     X  03
SYS$CNTREG                        ********     X  03
SYS$EXPREG                        ********     X  03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000020 (32.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	000000FE (254.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                      VALUE       DEFINITION      REFERENCES...
------                      -----       ----------      -------------
$$ARGS                     =00000003    98     (1)      97     (1)      98     (1)
$$T1                       =00000010    98     (1)      97     (1)      98     (1)
BIT...                     =00660160    96     (1)      96     (1)
DS$GL_BUFCNT                00000008-R  112    (1)    #-211    (3)    #-271    (5)    #-273    (5)    #-301    (5)
DS$GQ_PHYADR                00000000-R  111    (1)    #-194    (3)
DS$K_ERROR                 =00000002    96     (1)
DS$K_NORMAL                =00000001    96     (1)
DS$K_SEVERE                =00000004    96     (1)
DS$K_SUBSYS                =00000066    96     (1)      96     (1)
DS$K_WARNING               =00000000    96     (1)
DS$_AP_NORMAL_BREAK        =00660150    96     (1)
DS$_ARITH                  =006600D0    96     (1)
DS$_ASBE                   =00660118    96     (1)
DS$_BADLINK                =006600F0    96     (1)
DS$_BADTYPE                =006600E8    96     (1)
DS$_BIIC                   =00660120    96     (1)
DS$_CHME                   =006600A8    96     (1)
DS$_CHMK                   =006600E0    96     (1)
DS$_DEVNAME                =00660108    96     (1)
DS$_ERROR                  =00660002    96     (1)
DS$_FHWE                   =00660068    96     (1)
DS$_FRAGBUF                =00660080    96     (1)    #-287    (5)
DS$_ICBUSY                 =006600C8    96     (1)
DS$_ICERR                  =006600C0    96     (1)
DS$_IHWE                   =00660060    96     (1)
DS$_ILLCHAR                =00660018    96     (1)
DS$_ILLPAGCNT              =00660078    96     (1)
DS$_ILLUNIT                =00660100    96     (1)
DS$_INIT_FAIL              =00660140    96     (1)
DS$_INSFMEM                =00660050    96     (1)
DS$_INVCPU                 =00660130    96     (1)
DS$_IPL2HI                 =006600B8    96     (1)
DS$_IVADDR                 =00660040    96     (1)
DS$_IVVECT                 =00660038    96     (1)
DS$_KRNLSTK                =00660090    96     (1)
DS$_LOGIC                  =00660070    96     (1)
DS$_MCHK                   =00660088    96     (1)
DS$_MEM_ALLOC_ERR          =00660138    96     (1)
DS$_MMOFF                  =00660058    96     (1)
DS$_NEEDUNIT               =006600F8    96     (1)
DS$_NODE                   =00660128    96     (1)
DS$_NOPCS                  =00660110    96     (1)
DS$_NORMAL                 =00660001    96     (1)
DS$_NOSUPPORT              =006600B1    96     (1)
DS$_NOTALLOWMP             =00660148    96     (1)
DS$_NOTDON                 =00660030    96     (1)
DS$_NOTIMP                 =006600B0    96     (1)      96     (1)
DS$_NULLSTR                =00660010    96     (1)
DS$_OVERFLOW               =00660008    96     (1)
DS$_POWER                  =00660098    96     (1)
```

```
DS$_PROGERR                     =00660020      96      (1)
DS$_SEVERE                      =00660004      96      (1)
DS$_TRANSL                      =006600A0      96      (1)
DS$_TRUNCATE                    =00660028      96      (1)
DS$_UNEXPINT                    =006600D8      96      (1)
DS$_UNSUPPDEV                   =00660158      96      (1)
DS$_VASFULL                     =00660048      96      (1)
DS$_WARNING                     =00660000      96      (1)     96      (1)
DSX$GETBUF                       0000000B-R    173     (3)
DSX$GETSYSBUF                    00000000-R    169     (3)
DSX$RELBUF                       0000007B-R    264     (5)
DSX$RELSYSBUF                    00000073-R    260     (5)
GETBUF$_NARGS                   =00000004      97      (1)
GETBUF$_PAGCNT                  =00000004      97      (1)     #-178   (3)     #-190   (3)
GETBUF$_PHYADR                  =0000000C      97      (1)     #-192   (3)     #-195   (3)
GETBUF$_REGION                  =00000010      97      (1)      183    (3)
GETBUF$_RETADR                  =00000008      97      (1)     #-204   (3)     #-206   (3)
GET_COMMON                       00000010-R    176     (3)     #-171   (3)
KB_CHECK                         00000000-XR                   #-213   (3)     #-307   (5)
PTE$K_EXTERNAL_USER             =00000000      105     (1)     #-174   (3)
PTE$K_INTERNAL_VDS              =00200000      104     (1)     #-170   (3)
Q_RETADR                         00000018-R    113     (1)
RELBUF$_NARGS                   =00000003      98      (1)
RELBUF$_PAGCNT                  =00000004      98      (1)     #-270   (5)     #-303   (5)
RELBUF$_REGION                  =0000000C      98      (1)      269    (5)
RELBUF$_RETADR                  =00000008      98      (1)     #-291   (5)     #-293   (5)
RELSYSBUF$_START_VA             =00000010      106     (1)     #-261   (5)
REL_COMMON                       0000007F-R    267     (5)     #-262   (5)
SS$_ILLPAGCNT                    00000000-XR                   #-180   (3)     #-277   (5)
SS$_NORMAL                       00000000-XR                   #-289   (5)
SS$_PAGOWNVIO                    00000000-XR                   #-305   (5)
SYS$CNTREG                       00000000-XR                    286    (5)
SYS$EXPREG                       00000000-XR                    191    (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEF	1	96 (1)	
$DEFINI	1	95 (1)	95 (1)
$DS_DSADEF	5	95 (1)	95 (1)
$DS_DSDEF	3	96 (1)	96 (1)
$DS_GETBUF_DEF	1	97 (1)	97 (1)
$DS_RELBUF_DEF	1	98 (1)	98 (1)
$EQU	1	96 (1)	96 (1)
$EQULS1	1	96 (1)	96 (1)
$EQULST	1	96 (1)	96 (1)
$GBLINI	2		96 (1)
$OFFDEF	1	97 (1)	97 (1) 98 (1)
$VIELD1	1	96 (1)	

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL2	00C0	211 (3)
ASHL	0078	210 (3) 300 (5)
BEQL	0013	193 (3) 205 (3) 292 (5) 304 (5)
BGEQ	0018	272 (5) 296 (5)
BGTR	0014	179 (3) 276 (5)
BLBC	00E9	302 (5)
BLEQ	0015	199 (3)
BNEQ	0012	288 (5)
BRB	0011	171 (3) 262 (5)
BRW	0031	181 (3) 278 (5)
BSBW	0030	213 (3) 307 (5)
CALLS	00FB	191 (3) 286 (5)
CLRL	00D4	265 (5)
CMPL	00D1	198 (3) 271 (5) 287 (5) 303 (5)
EXTZV	00EF	183 (3) 269 (5)
INCL	00D6	209 (3) 299 (5)
MNEGL	00CE	297 (5)
MOVAQ	007E	185 (3) 280 (5)
MOVL	00D0	170 (3) 174 (3) 201 (3) 261 (5) 270 (5) 273 (5) 289 (5)
MOVQ	007D	194 (3) 197 (3) 206 (3) 293 (5)
MOVZWL	003C	180 (3) 277 (5) 305 (5)
POPR	00BA	202 (3)
PUSHAQ	007F	189 (3) 284 (5)
PUSHL	00DD	186 (3) 187 (3) 188 (3) 190 (3) 200 (3) 281 (5) 282 (5)
		283 (5) 285 (5)
RET	0004	214 (3) 308 (5)
SUBL	00C2	208 (3)
SUBL2	00C2	301 (5)
SUBL3	00C3	295 (5)
TSTL	00D5	178 (3) 192 (3) 204 (3) 275 (5) 291 (5)

```
                                        +-----------------------------+
                                        ! Directives Cross Reference !
                                        +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.BLKL           111     (1)     112     (1)     113     (1)
.END            310     (5)
.ENDC           96      (1)
.ENDM           95      (1)     96      (1)     97      (1)     98      (1)
.ENDR           96      (1)
.ENTRY          169     (3)     173     (3)     260     (5)     264     (5)
.IDENT          10      (1)
.IF             96      (1)
.IFF            96      (1)
.IIF            96      (1)
.IRP            96      (1)     97      (1)     98      (1)
.LIBRARY        85      (1)     86      (1)
.LIST           11      (1)     95      (1)     97      (1)     98      (1)
.MACRO          96      (1)
.MDELETE        96      (1)     97      (1)     98      (1)
.NLIST          12      (1)     95      (1)     97      (1)     98      (1)
.NOCROSS        95      (1)
.PAGE           80      (1)
.PSECT          110     (1)     116     (2)
.RESTORE        95      (1)
.SAVE           95      (1)
.SBTTL          115     (2)     216     (4)     81      (1)
.TITLE          9       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	13	#-178 (3), 183 (3), #-190 (3), #-192 (3), #-195 (3), #-204 (3), #-206 (3), #-261 (5), 269 (5), #-270 (5), #-291 (5), #-293 (5), #-303 (5)
R0	10	#-180 (3), #-185 (3), 189 (3), #-277 (5), #-280 (5), 284 (5), #-287 (5), #-289 (5), #-302 (5), #-305 (5)
R1	19	#-197 (3), #-198 (3), #-201 (3), #-202 (3), #-206 (3), #-208 (3), #-270 (5), #-271 (5), #-273 (5), #-275 (5), #-285 (5), #-295 (5), #-297 (5), #-299 (5), #-300 (5), #-301 (5), #-303 (5)
R2	17	169 (3), 173 (3), #-198 (3), #-200 (3), #-201 (3), #-208 (3), #-209 (3), #-210 (3), #-211 (3), 260 (5), 264 (5), #-269 (5), #-271 (5), #-273 (5), #-282 (5), #-301 (5)
R3	10	169 (3), 173 (3), #-184 (3), #-187 (3), #-211 (3), 260 (5), #-261 (5), 264 (5), #-265 (5), #-281 (5)
R4	5	169 (3), #-170 (3), 173 (3), #-174 (3), #-186 (3)
SP	6	185 (3), #-197 (3), 280 (5), #-293 (5), #-295 (5)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.19
Command processing	883	00:00:00.22	00:00:00.83
Pass 1	474	00:00:00.92	00:00:02.19
Symbol table sort	0	00:00:00.03	00:00:00.03
Pass 2	40	00:00:00.21	00:00:00.38
Symbol table output	2	00:00:00.02	00:00:00.05
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	4	00:00:00.12	00:00:00.21
Assembler run totals	1428	00:00:01.54	00:00:03.88

The working set limit was 2012 pages.
47658 bytes (94 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 158 non-local and 11 local symbols.
310 source lines were read in Pass 1, producing 35 object records in Pass 2.
34 pages of virtual memory were used to define 12 macros.

```
                                    +----------------------------+
                                    ! Macro library statistics !
                                    +----------------------------+

Macro library name                   Macros defined
------------------                   --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8              4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       6
TOTALS (all libraries)                                10

222 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:BUFFER.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element CALLFRAME.B32
;    0002  0      ! *2    12-MAY-1986 10:42:45 BARANSKI "Cleaning up Object Libraries."
;    0003  0      ! *1     6-FEB-1986 15:13:59 DS ""
;    0004  0      ! DEC/CMS REPLACEMENT HISTORY, Element CALLFRAME.B32
;    0005  0      %TITLE '*** CallFrame Module'
;    0006  0      MODULE CallFrame (                      ! Routines to access the call frames
;    0007  0                        IDENT = '01-01'
;    0008  0                        ) =
;    0009  0      !++
;    0010  0      ! COPYRIGHT (c) 1983
;    0011  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0012  0      !
;    0013  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0014  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0015  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0016  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0017  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0018  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0019  0      ! REMAIN IN DEC.
;    0020  0      !
;    0021  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0022  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0023  0      ! CORPORATION.
;    0024  0      !
;    0025  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0026  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0027  0      !-
```

```
;    0029  0      !++
;    0030  0      ! Facility:
;    0031  0      !
;    0032  0      !       Diagnostic Supervisor
;    0033  0      !
;    0034  0      ! Abstract:
;    0035  0      !
;    0036  0      !       This module contains rutines to access the call frames on the stack.
;    0037  0      !
;    0038  0      ! Author:
;    0039  0      !
;    0040  0      !       Jack Stansbury
;    0041  0      !
;    0042  0      ! Creation Date:
;    0043  0      !
;    0044  0      !       March 8, 1983
;    0045  0      !
;    0046  0      ! Version:
;    0047  0      !
;    0048  0      !       6.11
;    0049  0      !
;    0050  0      ! Modified By:
;    0051  0      !
;    0052  0      !       01      Bob Bergazzi    May 17, 1983    Version 6.11
;    0053  0      !               Changed the order of searching libraries
;    0054  0      !--
```

```
;    0056  0     %SBTTL  'Libraries'
;    0057  1     BEGIN                           ! Begin the CallFrame module
;    0058  1
;    0059  1     LIBRARY
;    0060  1             '$Diag';                ! Use Diag.L32 first
;    0061  1
;    0062  1     LIBRARY
;    0063  1             '$DS';                  ! Use Ds.L32 second
;    0064  1
;    0065  1     LIBRARY
;    0066  1             '$CRD';                 ! Use CRD.L32 third
;    0067  1
;    0068  1     LIBRARY
;    0069  1             'Sys$Library:Lib';      ! Use Lib as last resort
```

```
;    0071  1     %SBTTL  'Table of Contents'
;    0072  1
;    0073  1     FORWARD ROUTINE
;    0074  1             VRShow_Calls                    : NOVALUE JSB_Cli ADDRESSING_MODE (LONG_RELATIVE),
;    0075  1                                             ! Show the last 'n' call frames and PCs.
;    0076  1
;    0077  1             DSR$Find_User_PC                : ADDRESSING_MODE (LONG_RELATIVE);
;    0078  1                                             ! Find the first user PC on the call frame stack.
```

```
;    0080  1     %SBTTL  'External Routines'
;    0081  1
;    0082  1     EXTERNAL ROUTINE
;    0083  1              DSR$CheckLoad                  : JSB_None ADDRESSING_MODE (LONG_RELATIVE);
```

```
;   0085  1     %SBTTL  'External Own Storage'
;   0086  1
;   0087  1     EXTERNAL
;   0088  1             DS$GL_Flags                     : ADDRESSING_MODE (LONG_RELATIVE);
```

```
;   0090  1     %SBTTL  'Included Macros and Literals'
;   0091  1
;   0092  1             $DS_TypeDef;                    ! Typecodes for the $Print statements
```

```
;    0094  1      xSBTTL  'Psect Definitions'
;    0095  1
;    0096  1      PSECT
;    0097  1          PLIT    = Data (NOEXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0098  1
;    0099  1      PSECT
;    0100  1          CODE    = Code (  EXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0101  1
;    0102  1      PSECT
;    0103  1          OWN     = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0104  1
;    0105  1      PSECT
;    0106  1          GLOBAL  = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
```

```
;   0108  1     %SBTTL  'Module-Global Static Storage'
;   0109  1
;   0110  1             ModNam ('CallFrame');           ! Module name for ErrSup macro
```

```
;    0112  1     %SBTTL  'VRShow_Calls Routine'
;    0113  1
;    0114  1     GLOBAL ROUTINE VRShow_Calls (CLI_Buffer) : NOVALUE JSB_Cli =
;    0115  1
;    0116  1     !++
;    0117  1     ! Functional Description:
;    0118  1     !
;    0119  1     !       Print 'Number_Of_Frames' from the call frame stack. The PC from each of the frames is printed.
;    0120  1     !
;    0121  1     ! Formal Parameters:
;    0122  1     !
;    0123  1     !       CLI_Buffer:       The address of the start of the CLI buffer. This buffer contains the following data
;    0124  1     !                         at the CLI$K_Data location:
;    0125  1     !       Number_Of_Frames: The number of frames to print. If this is 0 (zero), print all the frames.
;    0126  1     !
;    0127  1     ! Implicit Inputs:
;    0128  1     !
;    0129  1     !       None
;    0130  1     !
;    0131  1     ! Side Effects:
;    0132  1     !
;    0133  1     !       None
;    0134  1     !
;    0135  1     ! Completion Codes:
;    0136  1     !
;    0137  1     !       None
;    0138  1     !--
```

```
;   0140  2     BEGIN                                           ! Start of routine VRShow_Calls
;   0141  2             MAP
;   0142  2                     CLI_Buffer : REF BLOCK [, BYTE];        ! The CLI_Buffer is a block of bytes
;   0143  2
;   0144  2             BUILTIN
;   0145  2                     PROBER,                         ! Probe read-accessability
;   0146  2                     PROBEW,                         ! Probe write-accessability
;   0147  2                     MOVPSL,                         ! Move from PSL
;   0148  2                     AP,                             ! Argument pointer
;   0149  2                     FP;                             ! Frame pointer
;   0150  2
;   0151  2             LOCAL
;   0152  2                     Psw,                            ! Processor Status Word (low word)
;   0153  2                     Number_Of_Frames,               ! The number of frames to print
;   0154  2                     Program_Counter,                ! PC from the current frame
;   0155  2                     Argument_Pointer,               ! Pointer to the argument array
;   0156  2                     Frame_Ptr;                      ! Points to current call frame
;   0157  2
;   0158  2             Number_Of_Frames = .CLI_Buffer [CLI$L_Data];    ! Get the data from the CLI buffer
;   0159  2
;   0160  2             IF (.Number_Of_Frames LEQ 0) THEN
;   0161  2                     Number_Of_Frames = 30;          ! Assume 30 is the most we will print
;   0162  2
;   0163  2             MOVPSL (Psw);                           ! Get the Psl (use the current PSL for the first line - belo
;   0164  2             Program_Counter = VRShow_Calls + 2;     ! Get the PC for this routine (skip over register save mask)
;   0165  2
; P 0166  2             $Print (DS$K_Type_General, DS$K_PrintF,
;   0167  2                     $ASCIC ('!/!/Call Frame Stack Dump (starting with the current frame):!/'));
; P 0168  2             $Print (DS$K_Type_General, DS$K_PrintF,
; P 0169  2                     $ASCIC ('  Frame: Current        PC: !XL       AP: !XL       PSL: !XL!/'),
;   0170  2                     .Program_Counter, .AP, .Psw);
;   0171  2
;   0172  2             Frame_Ptr = .FP;                        ! Initialize to current frame
;   0173  2
;   0174  2             INCR Frame FROM 0 TO (.Number_Of_Frames - 1) DO
;   0175  3                     BEGIN
;   0176  3                     Program_Counter  = .(.Frame_Ptr + 16);  ! Pick out PC from next frame
;   0177  3                     Argument_Pointer = .(.Frame_Ptr + 8);   ! Pick out AP from next frame
;   0178  3                     Psw = .(.Frame_Ptr + 4) <5, 11, 0>;     ! Get the saved Psw from next frame
;   0179  3
; P 0180  3                     $Print (DS$K_Type_General, DS$K_PrintF,
; P 0181  3                             $ASCIC ('  Frame: !XL Saved PC: !XL Saved AP: !XL Saved PSW: !3XL!/'),
;   0182  3                             .Frame_Ptr, .Program_Counter, .Argument_Pointer, .Psw);
;   0183  3
;   0184  3                     Frame_Ptr = .(.Frame_Ptr + 12);         ! Get next frame pointer
;   0185  3
;   0186  3                     IF (.Frame_Ptr EQL 0) THEN              ! If at the end of the linked list of frames,
;   0187  3                             EXITLOOP;                       ! ... exit this loop
;   0188  3
;   0189  3                     IF (NOT PROBER (%REF (0), %REF (20), .Frame_Ptr)) THEN
;   0190  3                             EXITLOOP;                       ! If we can't read it, exit this loop
;   0191  3
;   0192  3                     IF (NOT PROBEW (%REF (0), %REF (20), .Frame_Ptr)) THEN
;   0193  3                             EXITLOOP;                       ! If we can't write to it, exit this loop
;   0194  2                     END;
;   0195  2
;   0196  2             IF ((DSR$CheckLoad()) AND (.DS$GL_Flags <DS$V_StrFlg, 0>)) THEN
```

```
;   0197  3                     BEGIN
;   0198  3                     !+
;   0199  3                     ! A program is loaded and has been started. Check to see if there is a user PC on the
;   0200  3                     ! stack from the registers pushed on for the DS$Cli routine.
;   0201  3                     !-
;   0202  3                     BIND
;   0203  3                             Argument_Array = (.AP) : VECTOR [, LONG];
;   0204  3
;   0205  3                     LOCAL
;   0206  3                             User_PC;
;   0207  3
;   0208  3                     !+
;   0209  3                     ! The 15th longword in the argument array is the user's PC. See the KB_Check routine in the
;   0210  3                     ! Kernel.Mar module to see what gets pushed onto the stack for the DS$Cli routine.
;   0211  3                     !-
;   0212  3
;   0213  3                     User_PC = .Argument_Array [15];         ! Get the 15th longword
;   0214  3                     If (.User_PC GTR %X'00000200') AND (.User_PC LSS %X'0000FA00') THEN
;   0215  4                             BEGIN
;   0216  4                             !+
;   0217  4                             ! The user's PC is within the diagnostic load region (from 200 to FA00 hex).
;   0218  4                             !-
;   0219  4                             $Print (DS$K_Type_General, DS$K_PrintF, $ASCIC ('!/  User return PC = !XL(X)!/'), .User_PC);
;   0220  3                             END;
;   0221  2                     END;
;   0222  1     END;                                                    ! End of routine VRShow_Calls


                                                                        .TITLE  CALLFRAME *** CallFrame Module
                                                                        .IDENT  \01-01\

                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                    65  6D  61  72  46  6C  6C  61  43  09  00000 P.AAA:  .ASCII  <9>\CallFrame\                                  ;
65  6D  61  72  46  20  6C  6C  61  43  2F  21  2F  21  3E  0000A P.AAB:  .ASCII  \>!/!/Call Frame Stack Dump (starting wit\      ;
74  73  28  20  70  6D  75  44  20  6B  63  61  74  53  20  00019                                                                ;
                    74  69  77  20  67  6E  69  74  72  61  00028                                                                ;
66  20  74  6E  65  72  72  75  63  20  65  68  74  20  68  00032         .ASCII  \h the current frame):!/\                        ;
                            2F  21  3A  29  65  6D  61  72  00041                                                                ;
65  72  72  75  43  20  3A  65  6D  61  72  46  20  20  3E  00049 P.AAC:  .ASCII  \>  Frame: Current        PC: !XL       A\      ;
21  20  3A  43  50  20  20  20  20  20  20  20  20  74  6E  00058                                                                ;
                    41  20  20  20  20  20  20  20  4C  58  00067                                                                ;
53  50  20  20  20  20  20  20  20  4C  58  21  20  3A  50  00071         .ASCII  \P: !XL       PSL: !XL!/\                        ;
                            2F  21  4C  58  21  20  3A  4C  00080                                                                ;
53  20  4C  58  21  20  3A  65  6D  61  72  46  20  20  3A  00088 P.AAD:  .ASCII  \:  Frame: !XL Saved PC: !XL Saved AP: !X\      ;
61  53  20  4C  58  21  20  3A  43  50  20  64  65  76  61  00097                                                                ;
                    58  21  20  3A  50  41  20  64  65  76  000A6                                                                ;
33  21  20  3A  57  53  50  20  64  65  76  61  53  20  4C  000B0         .ASCII  \L Saved PSW: !3XL!/\                            ;
                                            2F  21  4C  58  000BF                                                                ;
72  75  74  65  72  20  72  65  73  55  20  20  2F  21  1D  000C3 P.AAE:  .ASCII  <29>\!/  User return PC = !XL(X)!/\              ;
2F  21  29  58  28  4C  58  21  20  3D  20  43  50  20  6E  000D2                                                                ;

                                                                $MODULE=            P.AAA
                                                                        .EXTRN  DSR$CHECKLOAD, DS$GL_FLAGS
                                                                        .EXTRN  DSX$PRINT

                                                                        .PSECT  CODE,NOWRT,  SHR,2
```

```
                                  00FC   8F  BB 00000 VRSHOW_CALLS::
                                                      PUSHR   #^M<R2,R3,R4,R5,R6,R7>                      ; 0114
                          57        1C   A2  D0 00004       MOVL    28(CLI_BUFFER), NUMBER_OF_FRAMES            ; 0158
                                         57  D5 00008       TSTL    NUMBER_OF_FRAMES                            ; 0160
                                         03  14 0000A       BGTR    1$                                          ;
                          57             1E  D0 0000C       MOVL    #30, NUMBER_OF_FRAMES                       ; 0161
                                         54  DC 0000F 1$:   MOVPSL  PSW                                         ; 0163
                          53        EE   AF  9E 00011       MOVAB   VRSHOW_CALLS+2, PROGRAM_COUNTER             ; 0164
                        00000000'        EF  9F 00015       PUSHAB  P.AAB                                       ; 0167
                                         01  DD 0001B       PUSHL   #1                                          ;
                                         7E  D4 0001D       CLRL    -(SP)                                       ;
                00000000G 9F             03  FB 0001F       CALLS   #3, @#DSX$PRINT                             ;
                                         54  DD 00026       PUSHL   PSW                                         ; 0170
                                  1008   8F  BB 00028       PUSHR   #^M<R3,AP>                                  ;
                        00000000'        EF  9F 0002C       PUSHAB  P.AAC                                       ;
                                         01  DD 00032       PUSHL   #1                                          ;
                                         7E  D4 00034       CLRL    -(SP)                                       ;
                00000000G 9F             06  FB 00036       CALLS   #6, @#DSX$PRINT                             ;
                          52             5D  D0 0003D       MOVL    FP, FRAME_PTR                               ; 0172
                          55             01  CE 00040       MNEGL   #1, FRAME                                   ; 0174
                                         37  11 00043       BRB     3$                                          ;
                          53        10   A2  D0 00045 2$:   MOVL    16(FRAME_PTR), PROGRAM_COUNTER              ; 0176
                          56        08   A2  D0 00049       MOVL    8(FRAME_PTR), ARGUMENT_POINTER              ; 0177
     54       04   A2     0B             05  EF 0004D       EXTZV   #5, #11, 4(FRAME_PTR), PSW                  ; 0178
                                         54  DD 00053       PUSHL   PSW                                         ; 0182
                                  004C   8F  BB 00055       PUSHR   #^M<R2,R3,R6>                               ;
                        00000000'        EF  9F 00059       PUSHAB  P.AAD                                       ;
                                         01  DD 0005F       PUSHL   #1                                          ;
                                         7E  D4 00061       CLRL    -(SP)                                       ;
                00000000G 9F             07  FB 00063       CALLS   #7, @#DSX$PRINT                             ;
                          52        0C   A2  D0 0006A       MOVL    12(FRAME_PTR), FRAME_PTR                    ; 0184
                                         10  13 0006E       BEQL    4$                                          ; 0186
                  62      14             00  0C 00070       PROBER  #0, #20, (FRAME_PTR)                        ; 0189
                                         0A  13 00074       BEQL    4$                                          ;
                  62      14             00  0D 00076       PROBEW  #0, #20, (FRAME_PTR)                        ; 0192
                                         04  13 0007A       BEQL    4$                                          ;
                  C5      55             57  F2 0007C 3$:   AOBLSS  NUMBER_OF_FRAMES, FRAME, 2$                 ; 0193
                        00000000G        EF  16 00080 4$:   JSB     DSR$CHECKLOAD                               ; 0196
                          31             50  E9 00086       BLBC    R0, 5$                                      ;
                  29 00000000G EF        02  E1 00089       BBC     #2, DS$GL_FLAGS, 5$                         ;
                          50        3C   AC  D0 00091       MOVL    60(AP), USER_PC                             ; 0213
                00000200  8F             50  D1 00095       CMPL    USER_PC, #512                               ; 0214
                                         1C  15 0009C       BLEQ    5$                                          ;
                0000FA00  8F             50  D1 0009E       CMPL    USER_PC, #64000                             ;
                                         13  18 000A5       BGEQ    5$                                          ;
                                         50  DD 000A7       PUSHL   USER_PC                                     ; 0219
                        00000000'        EF  9F 000A9       PUSHAB  P.AAE                                       ;
                                         01  DD 000AF       PUSHL   #1                                          ;
                                         7E  D4 000B1       CLRL    -(SP)                                       ;
                00000000G 9F             04  FB 000B3       CALLS   #4, @#DSX$PRINT                             ;
                                  00FC   8F  BA 000BA 5$:   POPR    #^M<R2,R3,R4,R5,R6,R7>                      ; 0222
                                         05  000BE          RSB                                                 ;

; Routine Size:  191 bytes,    Routine Base:  CODE + 0000
```

```
;    0224  1      %SBTTL  'DSR$Find_User_PC Routine'
;    0225  1
;    0226  1      GLOBAL ROUTINE DSR$Find_User_PC =
;    0227  1
;    0228  1      !++
;    0229  1      ! Functional description:
;    0230  1      !
;    0231  1      !       This routine is used to find a user program PC for typing.
;    0232  1      !
;    0233  1      ! Calling sequence:
;    0234  1      !
;    0235  1      !       CALLS   #0, DSR$Find_User_PC
;    0236  1      !
;    0237  1      ! Input Parameters:
;    0238  1      !
;    0239  1      !       None
;    0240  1      !
;    0241  1      ! Implicit Inputs:
;    0242  1      !
;    0243  1      !       None
;    0244  1      !
;    0245  1      ! Output Parameters:
;    0246  1      !
;    0247  1      !       None
;    0248  1      !
;    0249  1      ! Implicit Outputs:
;    0250  1      !
;    0251  1      !       None
;    0252  1      !
;    0253  1      ! Completion Codes:
;    0254  1      !
;    0255  1      !       The user PC is returned in R0. If no user PC can be found, a 0 is returned.
;    0256  1      !
;    0257  1      ! Side Effects:
;    0258  1      !
;    0259  1      !       None
;    0260  1      !--
```

```
;    0262  2     BEGIN                                   ! Routine DSR$Find_User_PC
;    0263  2             LOCAL
;    0264  2                     Frame_Pointer,
;    0265  2                     Program_Counter;
;    0266  2
;    0267  2             BUILTIN
;    0268  2                     PROBER,
;    0269  2                     PROBEW,
;    0270  2                     FP;
;    0271  2
;    0272  2             Frame_Pointer   = .FP;
;    0273  2             Program_Counter = .(.Frame_Pointer + 16);       ! Pick out the PC from the current frame
;    0274  2
;    0275  2             WHILE (.Program_Counter GEQ %X'00010000') DO
;    0276  3                     BEGIN
;    0277  3                     Frame_Pointer = .(.Frame_Pointer + 12); ! Get next frame pointer
;    0278  3
;    0279  3                     IF (.Frame_Pointer EQLA 0) THEN         ! If at the end of the linked list of frames,
;    0280  3                             EXITLOOP;                       ! ... exit this loop
;    0281  3
;    0282  3                     IF (NOT PROBER (%REF (0), %REF (20), .Frame_Pointer)) THEN
;    0283  3                             EXITLOOP;                       ! If we can't read it, exit this loop
;    0284  3
;    0285  3                     IF (NOT PROBEW (%REF (0), %REF (20), .Frame_Pointer)) THEN
;    0286  3                             EXITLOOP;                       ! If we can't write to it, exit this loop
;    0287  3
;    0288  3                     Program_Counter = .(.Frame_Pointer + 16);
;    0289  3                                                             ! Pick out PC from next frame
;    0290  2                     END;
;    0291  2
;    0292  2             IF (.Program_Counter LSSA %X'00010000') THEN
;    0293  3                     RETURN (.Program_Counter)
;    0294  2             ELSE
;    0295  2                     RETURN (0);
;    0296  1     END;                                    ! Routine DSR$Find_User_PC


                                                0000 00000          .ENTRY  DSR$FIND_USER_PC, Save nothing                  ; 0226
                                  51        5D  D0 00002          MOVL    FP, FRAME_POINTER                               ; 0272
                                  50   10   A1  D0 00005 1$:       MOVL    16(FRAME_POINTER), PROGRAM_COUNTER              ; 0273
                      00010000    8F        50  D1 00009          CMPL    PROGRAM_COUNTER, #65536                         ; 0275
                                            12  19 00010          BLSS    2$                                              ;
                                  51   0C   A1  D0 00012          MOVL    12(FRAME_POINTER), FRAME_POINTER                ; 0277
                                            0C  13 00016          BEQL    2$                                              ; 0279
                      61                    00  0C 00018          PROBER  #0, #20, (FRAME_POINTER)                        ; 0282
                                            06  13 0001C          BEQL    2$                                              ;
                      61                    00  0D 0001E          PROBEW  #0, #20, (FRAME_POINTER)                        ; 0285
                                            E1  12 00022          BNEQ    1$                                              ;
                      00010000    8F        50  D1 00024 2$:      CMPL    PROGRAM_COUNTER, #65536                         ; 0292
                                            02  1F 0002B          BLSSU   3$                                              ;
                                            50  D4 0002D          CLRL    R0                                              ; 0295
                                                04 0002F 3$:      RET                                                     ; 0296

; Routine Size:  48 bytes,    Routine Base:  CODE + 00BF
```

```
;   0298  1     END                                    ! End of CallFrame module
;   0299  0     ELUDOM



;                                  PSECT SUMMARY
;
;       Name                    Bytes                          Attributes
;
;  DATA                            225  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                            239  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



Symbol                      Type  Defined  Referenced ...
------------------------ -------- ----- ----------------
$ASCIC                    Macro    Lib01   167     169     181     219
$DS_TYPEDEF               Macro    Lib01    92
$EQULST                   Macro    Lib04    92
$ER                       Comptime   110
$MODULE                   Bind       110
$PRINT                    Macro    Lib02   166     168     180     219
AP                        Builtin    148   170.    203.
ARGUMENT_ARRAY            Bind       203   213.
ARGUMENT_POINTER          Local      155   177=    182.
CLI$L_DATA                Macro    Lib02   158
CLI_BUFFER                RoutForm   114   142m    158a.
DS$GL_FLAGS               External    88   196.
DS$K_PRINTB               Literal     92
DS$K_PRINTF               Literal     92   167     170     182     219
DS$K_PRINTI               Literal     92
DS$K_PRINTX               Literal     92
DS$K_TYPE_ABORT_PROGRAM   Literal     92
DS$K_TYPE_ABORT_TEST      Literal     92
DS$K_TYPE_COMMAND_ERR     Literal     92
DS$K_TYPE_COMMAND_OUT     Literal     92
DS$K_TYPE_CRD_AUTOTEST    Literal     92
DS$K_TYPE_DS_PROMPT       Literal     92
DS$K_TYPE_DS_START        Literal     92
DS$K_TYPE_ERRDEV          Literal     92
DS$K_TYPE_ERRHARD         Literal     92
DS$K_TYPE_ERROR_BODY      Literal     92
DS$K_TYPE_ERROR_END       Literal     92
DS$K_TYPE_ERRPREP         Literal     92
DS$K_TYPE_ERRSOFT         Literal     92
DS$K_TYPE_ERRSUP          Literal     92
DS$K_TYPE_ERRSYS          Literal     92
DS$K_TYPE_ERR_HALT        Literal     92
DS$K_TYPE_EXCEPTION       Literal     92
DS$K_TYPE_EXCEPTION_HEAD  Literal     92
DS$K_TYPE_FIRST_PASS      Literal     92
DS$K_TYPE_GENERAL         Literal     92   167     170     182     219
```

Symbol	Type	Defined	Referenced ...
DS$K_TYPE_GENERAL_ERROR	Literal	92	
DS$K_TYPE_NO_TESTS	Literal	92	
DS$K_TYPE_PARAM_ERROR	Literal	92	
DS$K_TYPE_PROGRAM_END	Literal	92	
DS$K_TYPE_PROGRAM_INFO	Literal	92	
DS$K_TYPE_PROGRAM_START	Literal	92	
DS$K_TYPE_QIO_INVADP	Literal	92	
DS$K_TYPE_QIO_NODRIVER	Literal	92	
DS$K_TYPE_QIO_WRONGVER	Literal	92	
DS$K_TYPE_SCRIPT_ECHO	Literal	92	
DS$K_TYPE_SCRIPT_PNF	Literal	92	
DS$K_TYPE_SCRIPT_PROMPT	Literal	92	
DS$K_TYPE_SCRIPT_SKIP	Literal	92	
DS$K_TYPE_SEQUENCE_ERROR	Literal	92	
DS$K_TYPE_START_ERR	Literal	92	
DS$K_TYPE_START_LIST	Literal	92	
DS$K_TYPE_SUMMARY	Literal	92	
DS$K_TYPE_USER_PROMPT	Literal	92	
DS$V_STRFLG	Macro Lib02	196	
DSR$CHECKLOAD	ExtRout	83	196c
DSR$FIND_USER_PC	GlobRout	226	77f
DSX$PRINT	ExtRout	167	167c 170e 170c 182e 182c 219e 219c
FP	Builtin	149	172.
	Builtin	270	272.
FRAME	Local	174	
FRAME_POINTER	Local	264	272= 273. 277= 277. 279. 282. 285. 288.
FRAME_PTR	Local	156	172= 176. 177. 178. 182. 184= 184. 186. 189. 192.
GET1ST_	Macro Lib04	92	
GET2ND_	Unbound	92	
JSB_CLI	Linkage Lib02	74	114
JSB_NONE	Linkage Lib02	83	
MODNAM	Macro Lib02	110	
MOVPSL	Builtin	147	163
NUL2ND_	Macro Lib04	92	
NUMBER_OF_FRAMES	Local	153	158= 160. 161= 174.
PROBER	Builtin	145	189
	Builtin	268	282
PROBEW	Builtin	146	192
	Builtin	269	285
PROGRAM_COUNTER	Local	154	164= 170. 176= 182.
	Local	265	273= 275. 288= 292. 293.
PSW	Local	152	163= 170. 178= 182.
USER_PC	Local	206	213= 214. 219.
VRSHOW_CALLS	GlobRout	114	74f 164a

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]CALLFRAME.B32;1
6	Module CALLFRAME	
60	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;8
63	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;8

```
  14  Library #3       DISK$DS:[DS.LOCAL.SCRATCH]CRD.L32;1
  69  Library #4       SYS$COMMON:[SYSLIB]LIB.L32;2
 299  Eludom CALLFRAME


       KEY TO REFERENCE TYPE FLAGS

            .   Fetch
            =   Store
            c   Routine call
            a   Address use
            @   Indirect use
            e   External, external routine, or external literal declaration
            f   Forward or forward routine declaration
            h   Condition handler enabling
            m   Map declaration
            u   Undeclare declaration


;                              Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;8
;                                            940         2         0         96         00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;8
;                                            675         6         0         47         00:00.1
;  DISK$DS:[DS.LOCAL.SCRATCH]CRD.L32;1        486         0         0         62         00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2             20450         3         0       1101         00:01.0




;                                   COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:CALLFRAME.B32

; Size:          239 code + 225 data bytes
; Run Time:         00:02.6
; Elapsed Time:     00:04.8
; Lines/CPU Min:     6795
; Lexemes/CPU-Min: 84613
; Memory Used:  112 pages
; Compilation Complete
```

Table of contents

(1) 81 CANCEL I/O ON CHANNEL

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CANCEL.MAR
0000     2 ; *2    12-MAY-1986 10:45:42 BARANSKI "Cleaning up Object Libraries."
0000     3 ; *1     6-FEB-1986 15:14:01 DS ""
0000     4 ; DEC/CMS REPLACEMENT HISTORY, Element CANCEL.MAR
0000     5         .TITLE  CANCEL  *** CANCEL cancel QIO
0000     6         .IDENT  /05-04/
0000     7
0000     8 ;
0000     9 ;*****************************************************************************
0000    10 ;*                                                                           *
0000    11 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
0000    12 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    13 ;*                                                                           *
0000    14 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    15 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    16 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    17 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    18 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    19 ;*  TRANSFERRED.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    22 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    23 ;*  CORPORATION.                                                             *
0000    24 ;*                                                                           *
0000    25 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    26 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    27 ;*                                                                           *
0000    28 ;*****************************************************************************
0000    29 ;
0000    30 ; D. N. CUTLER 4-AUG-77
0000    31 ;
0000    32 ; SYSTEM SERVICE CANCEL I/O ON CHANNEL
0000    33 ;
0000    34 ; MODIFIED BY:
0000    35 ;
0000    36 ;                   N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
0000    37 ;         01        DIAGNOSTIC SUPERVISOR INTEGRATION
0000    38 ;
0000    39 ;                   Dave Butenhof   13-may-1980, Version 5.4
0000    40 ;         02        Modify VMS V2.0 source for Supervisor environment
0000    41 ;                   Roger Riggs, 23-July-1980, Version 5.5
0000    42 ;         03        Changed word offset to long
0000    43 ;
0000    44 ;         04        M. Baggett      13-May-1983     Version 6.11
0000    45 ;                   Fixed trucation error.
0000    46 ;
0000    47 ;         V0002     ACG0047         Andrew C. Goldstein,    16-Aug-1979  18:37
0000    48 ;                   Add access rights block, protection interface changes
0000    49 ;
0000    50 ;**
0000    51 ;
0000    52 ; MACRO LIBRARY CALLS
0000    53 ;
0000    54
0000    55         $CADEF                          ;DEFINE CONDIATIONAL ASSEMBLY PARAMETERS
0000               .SAVE   LOCAL_BLOCK
0000               .RESTORE
```

```
                            0000     56          $CCBDEF                         ;DEFINE CCB OFFSETS
                            0000                 .SAVE   LOCAL_BLOCK
                            0000                 .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
                 00000004   0000                         .BLKL   1
                            0004                 .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
                 00000008   0004                         .BLKL   1
                            0008                 .IIF    NB,CCB$B_STS,   CCB$B_STS:
                 00000009   0008                         .BLKB   1
                            0009                 .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
                 0000000A   0009                         .BLKB   1
                            000A                 .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
                 0000000C   000A                         .BLKW   1
                            000C                 .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
                 00000010   000C                         .BLKL   1
                            0010                 .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                            0010                 .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                            0000                 .RESTORE
                            0000     57          $DDBDEF                         ;DEFINE DDB OFFSETS
                            0000                 .SAVE   LOCAL_BLOCK
                            0000                 .PSECT  $ABS$,ABS
                 00000000   0010                 .=0
                            0000                 .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
                 00000004   0000                         .BLKL   1
                            0004                 .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
                 00000008   0004                         .BLKL   1
                            0008                 .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
                 0000000A   0008                         .BLKW   1
                            000A                 .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
                 0000000B   000A                         .BLKB   1
                 0000000C   000B                         .BLKB   1
                            000C                 .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
                 00000010   000C                         .BLKL   1
                            0010                 .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
                 00000013   0010                         .BLKB   3
                            0013                 .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
                 00000014   0013                         .BLKB   1
                            0014                 .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
                 00000015   0014                         .BLKB   1
                 00000024   0015                         .BLKB   15
                            0024                 .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
                 00000025   0024                         .BLKB   1
                 00000034   0025                         .BLKB   15
                            0034                 .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                            0034                 .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                            0000                 .RESTORE
                            0000     58          $DDTDEF                         ;DEFINE DDT OFFSETS
                            0000                 .SAVE   LOCAL_BLOCK
                            0000                 .PSECT  $ABS$,ABS
                 00000000   0034                 .=0
                            0000                 .IIF    NB,DDT$L_START, DDT$L_START:
                 00000004   0000                         .BLKL   1
                            0004                 .IIF    NB,DDT$L_UNSOLINT,      DDT$L_UNSOLINT:
                 00000008   0004                         .BLKL   1
                            0008                 .IIF    NB,DDT$L_FDT,   DDT$L_FDT:
                 0000000C   0008                         .BLKL   1
                            000C                 .IIF    NB,DDT$L_CANCEL,        DDT$L_CANCEL:
                 00000010   000C                         .BLKL   1
```

```
                     0010              .IIF    NB,DDT$L_REGDUMP,       DDT$L_REGDUMP:
00000014             0010                              .BLKL   1
                     0014              .IIF    NB,DDT$W_DIAGBUF,       DDT$W_DIAGBUF:
00000016             0014                              .BLKW   1
                     0016              .IIF    NB,DDT$W_ERRORBUF,      DDT$W_ERRORBUF:
00000018             0016                              .BLKW   1
                     0018              .IIF    NB,DDT$L_UNITINIT,      DDT$L_UNITINIT:
0000001C             0018                              .BLKL   1
                     001C              .IIF    NB,DDT$L_ALTSTART,      DDT$L_ALTSTART:
00000020             001C                              .BLKL   1
                     0000              .RESTORE
                     0000      59      $DEVDEF                         ;DEFINE DEVICE CHARACTERISTIC BITS
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             0034              .=0
                     0000              .RESTORE
                     0000      60      $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             0034              .=0
                     0000              .RESTORE
                     0000      61      $IODEF                          ;DEFINE I/O FUNCTION CODES
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             0034              .=0
                     0000              .RESTORE
                     0000      62      $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             0034              .=0
                     0000              .RESTORE
                     0000      63      $IRPDEF                         ;DEFINE IRP OFFSETS
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             0034              .=0
                     0000              .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
00000004             0000                              .BLKL   1
                     0004              .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
00000008             0004                              .BLKL   1
                     0008              .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
0000000A             0008                              .BLKW   1
                     000A              .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
0000000B             000A                              .BLKB   1
                     000B              .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
0000000C             000B                              .BLKB   1
                     000C              .IIF    NB,IRP$L_PID,   IRP$L_PID:
00000010             000C                              .BLKL   1
                     0010              .IIF    NB,IRP$L_AST,   IRP$L_AST:
00000014             0010                              .BLKL   1
                     0014              .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
00000018             0014                              .BLKL   1
                     0018              .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
0000001C             0018                              .BLKL   1
                     001C              .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
00000020             001C                              .BLKL   1
                     0020              .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
00000022             0020                              .BLKW   1
```

```
                        0022            .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
00000023  0022                                  .BLKB   1
                        0023            .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
00000024  0023                                  .BLKB   1
                        0024            .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
00000028  0024                                  .BLKL   1
                        0028            .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
0000002A  0028                                  .BLKW   1
                        002A            .IIF    NB,IRP$W_STS,   IRP$W_STS:
0000002C  002A                                  .BLKW   1
                        002C            .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
00000030  002C                                  .BLKL   1
                        0030            .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
00000032  0030                                  .BLKW   1
                        0032            .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
00000034  0032                                  .BLKW   1
                        0034            .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
                        0034            .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
00000038  0034                                  .BLKL   1
                        0038            .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
                        0038            .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
                        0038            .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
00000039  0038                                  .BLKB   1
0000003C  0039                                  .BLKB   3
                        003C            .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
                        003C            .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
                        003C            .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                                  .BLKW   1
                        003E            .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                                  .BLKW   1
                        0040            .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                                  .BLKL   1
                        0044            .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                                  .BLKL   1
                        0048            .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                                  .BLKL   1
                        004C            .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                                  .BLKL   1
                        0050            .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                                  .BLKL   1
00000058  0054                                  .BLKL   1
0000005C  0058                                  .BLKL   1
                        005C            .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
                        005C            .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
                        0000            .RESTORE
                        0000     64     $PCBDEF                         ;DEFINE PCB OFFSETS
                        0000            .SAVE   LOCAL_BLOCK
                        0000            .PSECT  $ABS$,ABS
00000000  005C                          .=0
                        0000            .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004  0000                                  .BLKL   1
                        0004            .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008  0004                                  .BLKL   1
                        0008            .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A  0008                                  .BLKW   1
                        000A            .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B  000A                                  .BLKB   1
```

```
                        000B            .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C  000B                                  .BLKB   1
                        000C            .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D  000C                                  .BLKB   1
                        000D            .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E  000D                                  .BLKB   1
                        000E            .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010  000E                                  .BLKW   1
                        0010            .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014  0010                                  .BLKL   1
                        0014            .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018  0014                                  .BLKL   1
                        0018            .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C  0018                                  .BLKL   1
                        001C            .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020  001C                                  .BLKL   1
                        0020            .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024  0020                                  .BLKL   1
                        0024            .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028  0024                                  .BLKL   1
                        0028            .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C  0028                                  .BLKL   1
                        002C            .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E  002C                                  .BLKW   1
                        002E            .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                                  .BLKB   1
                        002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                                  .BLKB   1
                        0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                                  .BLKW   1
                        0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                                  .BLKW   1
                        0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                                  .BLKW   1
                        0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                                  .BLKW   1
                        0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                                  .BLKW   1
                        003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                                  .BLKW   1
                        003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                                  .BLKW   1
                        003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                                  .BLKW   1
                        0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                                  .BLKW   1
                        0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                                  .BLKW   1
                        0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                                  .BLKB   8
                        004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                        004C            .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                                  .BLKL   1
                        0050            .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                                  .BLKL   1
                        0054            .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                                  .BLKL   1
```

```
                        0058                .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
                        0058                        .BLKL   1
                        005C                .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
                        005C                        .BLKL   1
                        0060                .IIF    NB,PCB$L_PID,   PCB$L_PID:
                        0060                        .BLKL   1
                        0064                .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
                        0064                        .BLKL   1
                        0068                .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
                        0068                        .BLKB   16
                        0078                .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
                        0078                        .BLKL   1
                        007C                .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
                        007C                        .BLKQ   1
                        0084                .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
                        0084                        .BLKL   1
                        0088                .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                        0088                .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
                        0088                        .BLKW   1
                        008A                .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
                        008A                        .BLKW   1
                        008C                .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                        008C                .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                        0000                .RESTORE
                        0000    65          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                        008C                .=0
                        0000                .RESTORE
                        0000    66          $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                        008C                .=0
                        0000                .RESTORE
                        0000    67          $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                        008C                .=0
                        0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                        0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                        0000                        .BLKL   1
                        0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                        0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                        0004                        .BLKL   1
                        0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                        0008                        .BLKW   1
                        000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                        000A                        .BLKB   1
                        000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                        000B                        .BLKB   1
                        000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                        000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                        000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                        000C                        .BLKB   1
                        000D                        .BLKB   3
                        0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                        0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
```

```
                    00000014  0010                    .BLKL   1
                              0014          .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                              0014          .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                              0014          .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
                    00000016  0014                    .BLKW   1
                              0016          .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
                    00000018  0016                    .BLKW   1
                              0018          .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                              0018          .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
                    0000001A  0018                    .BLKW   1
                              001A          .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                              001A          .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                    0000001C  001A                    .BLKW   1
                              001C          .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                    00000020  001C                    .BLKL   1
                              0020          .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                    00000024  0020                    .BLKL   1
                              0024          .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                    00000028  0024                    .BLKL   1
                              0028          .IIF    NB,UCB$L_PID,   UCB$L_PID:
                    0000002C  0028                    .BLKL   1
                              002C          .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
                    00000030  002C                    .BLKL   1
                              0030          .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
                    00000034  0030                    .BLKL   1
                              0034          .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
                    00000038  0034                    .BLKL   1
                              0038          .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
                    00000039  0038                    .BLKB   1
                              0039          .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
                    0000003A  0039                    .BLKB   1
                              003A          .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
                    0000003C  003A                    .BLKW   1
                              003C          .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                              003C          .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                              003C          .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
                    0000003D  003C                    .BLKB   1
                              003D          .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                              003D          .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
                    0000003E  003D                    .BLKB   1
                              003E          .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                              003E          .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
                    0000003F  003E                    .BLKB   1
                              003F          .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
                    00000040  003F                    .BLKB   1
                              0040          .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
                    00000044  0040                    .BLKL   1
                              0044          .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
                    00000048  0044                    .BLKL   1
                              0048          .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
                    0000004A  0048                    .BLKW   1
                              004A          .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                              004A          .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
                    0000004B  004A                    .BLKB   1
                              004B          .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
                    0000004C  004B                    .BLKB   1
                              004C          .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
```

```
00000050  004C                        .BLKL   1
                0050        .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                        .BLKW   1
                0052        .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                0052        .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                        .BLKB   1
                0053        .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                        .BLKB   1
                0054        .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                        .BLKL   1
                0058        .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                        .BLKW   1
                005A        .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                        .BLKW   1
                005C        .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                005C        .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                        .BLKL   1
                0060        .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                        .BLKL   1
                0064        .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                0064        .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                        .BLKL   1
                0068        .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                        .BLKL   1
                006C        .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                        .BLKW   1
                006E        .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                        .BLKW   1
                0070        .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                        .BLKB   1
                0071        .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                        .BLKB   1
                0072        .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                        .BLKW   1
                0074        .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                0074        .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074              . = 0
                0000        .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                        .BLKW   1
00000074  0002              . = 116
                0074        .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                        .BLKL   1
                0078        .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                        .BLKL   1
                007C        .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                        .BLKL   1
                0080        .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                        .BLKL   1
                0084        .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                        .BLKL   1
                0088        .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                        .BLKL   1
                008C        .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                        .BLKL   1
                0090        .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                0090        .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090              . = 116
```

```
                     0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
          00000075   0074                  .BLKB   1
                     0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
          00000076   0075                  .BLKB   1
                     0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
          00000077   0076                  .BLKB   1
                     0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
          00000078   0077                  .BLKB   1
                     0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
          0000007C   0078                  .BLKL   1
          0000007E   007C                  .BLKW   1
                     007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
          00000080   007E                  .BLKW   1
                     0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
          00000084   0080                  .BLKL   1
          00000080   0084          . = 128
          00000084   0080                  .BLKL   1
                     0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
          00000088   0084                  .BLKL   1
                     0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
          0000008A   0088                  .BLKW   1
                     008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
          0000008C   008A                  .BLKW   1
                     008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                     008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
          0000008E   008C                  .BLKW   1
                     008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
          00000090   008E                  .BLKW   1
                     0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
          00000092   0090                  .BLKW   1
                     0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
          00000094   0092                  .BLKW   1
                     0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
          00000095   0094                  .BLKB   1
                     0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
          00000096   0095                  .BLKB   1
                     0096          .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
          00000098   0096                  .BLKW   1
          00000096   0098          . = 150
          00000098   0096                  .BLKW   1
                     0098          .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
          0000009C   0098                  .BLKL   1
                     009C          .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
          000000A0   009C                  .BLKL   1
                     00A0          .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
          000000A4   00A0                  .BLKL   1
                     00A4          .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
          000000A6   00A4                  .BLKW   1
                     00A6          .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
          000000A7   00A6                  .BLKB   1
          000000A8   00A7                  .BLKB   1
          00000074   00A8          . = 116
          00000078   0074                  .BLKL   1
          0000007C   0078                  .BLKL   1
          00000080   007C                  .BLKL   1
          00000084   0080                  .BLKL   1
          00000088   0084                  .BLKL   1
```

```
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                  .BLKL   1
00000094  0090                                  .BLKL   1
00000098  0094                                  .BLKL   1
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                  .BLKB   1
          009D                  .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                  .BLKB   1
          009E                  .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                  .BLKB   1
000000A0  009F                                  .BLKB   1
          00A0                  .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                  .BLKB   1
          00A1                  .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                  .BLKB   1
          00A2                  .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                  .BLKB   1
000000A4  00A3                                  .BLKB   1
          00A4                  .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                  .BLKB   1
          00A5                  .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                  .BLKW   1
000000A8  00A7                                  .BLKB   1
          00A8                  .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                  .BLKL   1
000000B0  00AC                                  .BLKL   1
000000B4  00B0                                  .BLKL   1
000000B8  00B4                                  .BLKL   1
          00B8                  .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                  .BLKL   1
          00BC                  .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC                  .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC                  . = 116
          0074                  .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                  .BLKW   1
00000080  007E                                  .BLKW   1
          0000                  .RESTORE
          0000      68          $VCBDEF                         ;DEFINE VCB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .IIF    NB,VCB$L_FCBFL, VCB$L_FCBFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,VCB$L_FCBBL, VCB$L_FCBBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,VCB$W_SIZE,  VCB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,VCB$B_TYPE,  VCB$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,VCB$C_MRKLEN,        VCB$C_MRKLEN:
          000B                  .IIF    NB,VCB$K_MRKLEN,        VCB$K_MRKLEN:
```

```
                    000B            .IIF    NB,VCB$B_STATUS,        VCB$B_STATUS:
0000000C  000B                              .BLKB   1
                    000C            .IIF    NB,VCB$W_TRANS, VCB$W_TRANS:
0000000E  000C                              .BLKW   1
                    000E            .IIF    NB,VCB$W_RVN,   VCB$W_RVN:
00000010  000E                              .BLKW   1
                    0010            .IIF    NB,VCB$L_AQB,   VCB$L_AQB:
00000014  0010                              .BLKL   1
                    0014            .IIF    NB,VCB$T_VOLNAME,       VCB$T_VOLNAME:
00000020  0014                              .BLKB   12
                    0020            .IIF    NB,VCB$L_RVT,   VCB$L_RVT:
00000024  0020                              .BLKL   1
                    0024            .IIF    NB,VCB$C_COMLEN,        VCB$C_COMLEN:
                    0024            .IIF    NB,VCB$K_COMLEN,        VCB$K_COMLEN:
                    0024            .IIF    NB,VCB$L_HOMELBN,       VCB$L_HOMELBN:
00000028  0024                              .BLKL   1
                    0028            .IIF    NB,VCB$L_HOME2LBN,      VCB$L_HOME2LBN:
0000002C  0028                              .BLKL   1
                    002C            .IIF    NB,VCB$L_IXHDR2LBN,     VCB$L_IXHDR2LBN:
00000030  002C                              .BLKL   1
                    0030            .IIF    NB,VCB$L_IBMAPLBN,      VCB$L_IBMAPLBN:
00000034  0030                              .BLKL   1
                    0034            .IIF    NB,VCB$L_SBMAPLBN,      VCB$L_SBMAPLBN:
00000038  0034                              .BLKL   1
                    0038            .IIF    NB,VCB$B_IBMAPSIZE,     VCB$B_IBMAPSIZE:
00000039  0038                              .BLKB   1
                    0039            .IIF    NB,VCB$B_SBMAPSIZE,     VCB$B_SBMAPSIZE:
0000003A  0039                              .BLKB   1
                    003A            .IIF    NB,VCB$B_IBMAPVBN,      VCB$B_IBMAPVBN:
0000003B  003A                              .BLKB   1
                    003B            .IIF    NB,VCB$B_SBMAPVBN,      VCB$B_SBMAPVBN:
0000003C  003B                              .BLKB   1
                    003C            .IIF    NB,VCB$W_CLUSTER,       VCB$W_CLUSTER:
0000003E  003C                              .BLKW   1
                    003E            .IIF    NB,VCB$W_EXTEND,        VCB$W_EXTEND:
00000040  003E                              .BLKW   1
                    0040            .IIF    NB,VCB$L_FREE,  VCB$L_FREE:
00000044  0040                              .BLKL   1
                    0044            .IIF    NB,VCB$L_MAXFILES,      VCB$L_MAXFILES:
00000048  0044                              .BLKL   1
                    0048            .IIF    NB,VCB$B_WINDOW,        VCB$B_WINDOW:
00000049  0048                              .BLKB   1
                    0049            .IIF    NB,VCB$B_LRU_LIM,       VCB$B_LRU_LIM:
0000004A  0049                              .BLKB   1
                    004A            .IIF    NB,VCB$W_FILEPROT,      VCB$W_FILEPROT:
0000004C  004A                              .BLKW   1
                    004C            .IIF    NB,VCB$W_MCOUNT,        VCB$W_MCOUNT:
0000004E  004C                              .BLKW   1
                    004E            .IIF    NB,VCB$B_EOFDELTA,      VCB$B_EOFDELTA:
0000004F  004E                              .BLKB   1
                    004F            .IIF    NB,VCB$B_RESFILES,      VCB$B_RESFILES:
00000050  004F                              .BLKB   1
                    0050            .IIF    NB,VCB$W_RECORDSZ,      VCB$W_RECORDSZ:
00000052  0050                              .BLKW   1
                    0052            .IIF    NB,VCB$B_BLOCKFACT,     VCB$B_BLOCKFACT:
00000053  0052                              .BLKB   1
                    0053            .IIF    NB,VCB$B_STATUS2,       VCB$B_STATUS2:
```

```
00000054  0053                        .BLKB   1
          0054          .IIF    NB,VCB$L_QUOTAFCB,      VCB$L_QUOTAFCB:
00000058  0054                        .BLKL   1
          0058          .IIF    NB,VCB$L_CACHE, VCB$L_CACHE:
0000005C  0058                        .BLKL   1
          005C          .IIF    NB,VCB$L_QUOCACHE,      VCB$L_QUOCACHE:
00000060  005C                        .BLKL   1
          0060          .IIF    NB,VCB$W_QUOSIZE,       VCB$W_QUOSIZE:
00000062  0060                        .BLKW   1
          0062          .IIF    NB,VCB$W_PENDERR,       VCB$W_PENDERR:
00000064  0062                        .BLKW   1
          0064          .IIF    NB,VCB$C_LENGTH,        VCB$C_LENGTH:
          0064          .IIF    NB,VCB$K_LENGTH,        VCB$K_LENGTH:
00000000  0064          . = 0
          0000          .IIF    NB,VCB$L_BLOCKFL,       VCB$L_BLOCKFL:
00000004  0000                        .BLKL   1
          0004          .IIF    NB,VCB$L_BLOCKBL,       VCB$L_BLOCKBL:
00000008  0004                        .BLKL   1
0000000B  0008          . = 11
00000024  000B          . = 36
          0024          .IIF    NB,VCB$L_CUR_FID,       VCB$L_CUR_FID:
          0024          .IIF    NB,VCB$W_CUR_NUM,       VCB$W_CUR_NUM:
00000026  0024                        .BLKW   1
          0026          .IIF    NB,VCB$W_CUR_SEQ,       VCB$W_CUR_SEQ:
00000028  0026                        .BLKW   1
          0028          .IIF    NB,VCB$L_START_FID,     VCB$L_START_FID:
          0028          .IIF    NB,VCB$W_START_NUM,     VCB$W_START_NUM:
0000002A  0028                        .BLKW   1
          002A          .IIF    NB,VCB$W_START_SEQ,     VCB$W_START_SEQ:
0000002C  002A                        .BLKW   1
          002C          .IIF    NB,VCB$W_MODE,  VCB$W_MODE:
0000002E  002C                        .BLKW   1
          002E          .IIF    NB,VCB$B_TM,    VCB$B_TM:
0000002F  002E                        .BLKB   1
          002F          .IIF    NB,VCB$B_CUR_RVN,       VCB$B_CUR_RVN:
00000030  002F                        .BLKB   1
          0030          .IIF    NB,VCB$L_ST_RECORD,     VCB$L_ST_RECORD:
00000034  0030                        .BLKL   1
          0034          .IIF    NB,VCB$L_MVL,   VCB$L_MVL:
00000038  0034                        .BLKL   1
          0038          .IIF    NB,VCB$L_WCB,   VCB$L_WCB:
0000003C  0038                        .BLKL   1
          003C          .IIF    NB,VCB$L_VPFL,  VCB$L_VPFL:
00000040  003C                        .BLKL   1
          0040          .IIF    NB,VCB$L_VPBL,  VCB$L_VPBL:
00000044  0040                        .BLKL   1
          0044          .IIF    NB,VCB$L_USRLBLAST,     VCB$L_USRLBLAST:
00000048  0044                        .BLKL   1
0000000B  0048          . = 11
          000B          .IIF    NB,VCB$B_QNAMECNT,      VCB$B_QNAMECNT:
0000000C  000B                        .BLKB   1
          000C          .IIF    NB,VCB$T_QNAME, VCB$T_QNAME:
00000020  000C                        .BLKB   20
          0000          .RESTORE
          0000    69    $WCBDEF                                 ;DEFINE WCB OFFSETS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
```

```
                    00000000  00BC              .=0
                              0000              .IIF    NB,WCB$L_WLFL,  WCB$L_WLFL:
                    00000004  0000                      .BLKL   1
                              0004              .IIF    NB,WCB$L_WLBL,  WCB$L_WLBL:
                    00000008  0004                      .BLKL   1
                              0008              .IIF    NB,WCB$W_SIZE,  WCB$W_SIZE:
                    0000000A  0008                      .BLKW   1
                              000A              .IIF    NB,WCB$B_TYPE,  WCB$B_TYPE:
                    0000000B  000A                      .BLKB   1
                              000B              .IIF    NB,WCB$B_ACCESS,        WCB$B_ACCESS:
                    0000000C  000B                      .BLKB   1
                              000C              .IIF    NB,WCB$L_PID,   WCB$L_PID:
                    0000000E  000C                      .BLKB   2
                              000E              .IIF    NB,WCB$W_REFCNT,        WCB$W_REFCNT:
                    00000010  000E                      .BLKW   1
                              0010              .IIF    NB,WCB$L_ORGUCB,        WCB$L_ORGUCB:
                    00000014  0010                      .BLKL   1
                              0014              .IIF    NB,WCB$W_ACON,  WCB$W_ACON:
                    00000016  0014                      .BLKW   1
                              0016              .IIF    NB,WCB$W_NMAP,  WCB$W_NMAP:
                    00000018  0016                      .BLKW   1
                              0018              .IIF    NB,WCB$L_FCB,   WCB$L_FCB:
                    0000001C  0018                      .BLKL   1
                              001C              .IIF    NB,WCB$L_RVT,   WCB$L_RVT:
                    00000020  001C                      .BLKL   1
                              0020              .IIF    NB,WCB$L_STVBN, WCB$L_STVBN:
                    00000024  0020                      .BLKL   1
                              0024              .IIF    NB,WCB$C_MAP,   WCB$C_MAP:
                              0024              .IIF    NB,WCB$K_MAP,   WCB$K_MAP:
                              0024              .IIF    NB,WCB$C_LENGTH,        WCB$C_LENGTH:
                              0024              .IIF    NB,WCB$K_LENGTH,        WCB$K_LENGTH:
                              0024              .IIF    NB,WCB$W_P1_COUNT,      WCB$W_P1_COUNT:
                    00000026  0024                      .BLKW   1
                              0026              .IIF    NB,WCB$L_P1_LBN,        WCB$L_P1_LBN:
                    0000002A  0026                      .BLKL   1
                              002A              .IIF    NB,WCB$W_P2_COUNT,      WCB$W_P2_COUNT:
                    0000002C  002A                      .BLKW   1
                              002C              .IIF    NB,WCB$L_P2_LBN,        WCB$L_P2_LBN:
                    00000030  002C                      .BLKL   1
                    00000000  0030              . = 0
                              0000              .IIF    NB,WCB$W_COUNT, WCB$W_COUNT:
                    00000002  0000                      .BLKW   1
                              0002              .IIF    NB,WCB$L_LBN,   WCB$L_LBN:
                    00000006  0002                      .BLKL   1
                    00000000  0006              . = 0
                    FFFFFFFA  0000                      .BLKB   -6
                              FFFA              .IIF    NB,WCB$W_PREVCOUNT,     WCB$W_PREVCOUNT:
                    FFFFFFFC  FFFA                      .BLKW   1
                              FFFC              .IIF    NB,WCB$L_PREVLBN,       WCB$L_PREVLBN:
                    00000000  FFFC                      .BLKL   1
                              0000              .RESTORE
                              0000      70
                              0000      71 ;
                              0000      72 ; LOCAL SYMBOLS
                              0000      73 ;
                              0000      74 ; ARGUMENT LIST OFFSET DEFINITIONS
                              0000      75 ;
```

```
                          0000
                  00000004  0000  CHAN=4                                  ;I/O CHANNEL NUMBER
                        00000000          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                          0000          MODNAM  CANCEL
          4C 45 43 4E 41 43 00' 0000  $MODULE:        .ASCIC  \CANCEL\
                        06  0000
```

```
                                0007    81          .SBTTL  CANCEL I/O ON CHANNEL
                                0007    82 ;+
                                0007    83 ; EXE$CANCEL - CANCEL I/O ON CHANNEL
                                0007    84 ;
                                0007    85 ; THIS SERVICE CANCELS ALL I/O ISSUED TO A DEVICE FORM THE SPECIFIED CHANNEL.
                                0007    86 ;
                                0007    87 ; INPUTS:
                                0007    88 ;
                                0007    89 ;       CHAN(AP) = NUMBER OF THE I/O CHANNEL TO CANCEL I/O FOR.
                                0007    90 ;
                                0007    91 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                0007    92 ;
                                0007    93 ; OUTPUTS:
                                0007    94 ;
                                0007    95 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CANCEL I/O.
                                0007    96 ;
                                0007    97 ;               SS$_EXQUOTA - DIRECT I/O QUOTA EXCEEDED WHILE TRYING TO
                                0007    98 ;                     CANCEL FILE I/O.
                                0007    99 ;
                                0007   100 ;               SS$_INSFMEM - INSUFFICIENT MEMORY AVAILABLE TO ALLOCATE I/O
                                0007   101 ;                     PACKET.
                                0007   102 ;
                                0007   103 ;               SS$_IVCHAN - INVALID CHANNEL NUMBER SPECIFIED.
                                0007   104 ;
                                0007   105 ;               SS$_NOPRIV - SPECIFIED CHANNEL IS NOT ASSIGNED TO A DEVICE
                                0007   106 ;                     OR THE CALLER DOES NOT HAVE SUFFICIENT PRIVILEGE TO
                                0007   107 ;                     ACCESS THE CHANNEL.
                                0007   108 ;
                                0007   109 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0007   110 ;
                                0007   111 ;               SS$_NORMAL - NORMAL COMPLETION.
                                0007   112 ;-
                                0007   113
                          00FC  0007   114          .ENTRY  EXE$CANCEL,^M<R2,R3,R4,R5,R6,R7>
54  00000000'EF    9E    0009   115          MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
    50    04 AC    3C    0010   116          MOVZWL  CHAN(AP),R0             ;GET I/O CHANNEL NUMBER
    00000000'EF    16    0014   117          JSB     IOC$VERIFYCHAN          ;VERIFY I/O CHANNEL NUMBER
          60 50    E9    001A   118          BLBC    R0,50$                  ;IF LBC INVALID CHANNEL
       57    52    D0    001D   119          MOVL    R2,R7                   ;SAVE CHANNEL INDEX
       56    51    D0    0020   120          MOVL    R1,R6                   ;SAVE ADDRESS OF CCB
       55    66    D0    0023   121          MOVL    CCB$L_UCB(R6),R5        ;GET ASSIGNED DEVICE UCB ADDRESS
             7E    DC    0026   122 10$:      MOVPSL  -(SP)                   ;SAVE CURRENT PROCESSOR STATUS
                         0028   123          SETIPL  #IPL$_ASTDEL            ;RAISE TO AST DELIVERY LEVEL
       12    02    DA    0028                       MTPR    #IPL$_ASTDEL,S^#PR$_IPL
          0A A6    B5    002B   124          TSTW    CCB$W_IOC(R6)           ;ANY I/O OUTSTANDING?
             51    13    002E   125          BEQL    60$                     ;IF EQL NO
    53    40 A5    9E    0030   126          MOVAB   UCB$L_IOQFL(R5),R3      ;GET ADDRESS OF I/O QUEUE LISTHEAD
       52    53    D0    0034   127          MOVL    R3,R2                   ;COPY ADDRESS OF I/O QUEUE LISTHEAD
                         0037   128          SETIPL  UCB$B_FIPL(R5)          ;RAISE TO DRIVER FORK IPL
       12    0B A5    DA    0037                       MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
       52    62    D0    003B   129 20$:      MOVL    IRP$L_IOQFL(R2),R2      ;GET ADDRESS OF NEXT I/O PACKET IN QUEUE
       53    52    D1    003E   130          CMPL    R2,R3                   ;END OF QUEUE?
             3E    13    0041   131          BEQL    60$                     ;IF EQL YES
 F3 2A A2    04    E0    0043   132          BBS     #IRP$V_VIRTUAL,IRP$W_STS(R2),20$ ;IF SET, VIRTUAL I/O REQUEST
 60 A4    0C A2    D1    0048   133          CMPL    IRP$L_PID(R2),PCB$L_PID(R4) ;PROCESS ID MATCH?
             EC    12    004D   134          BNEQ    20$                     ;IF NEQ NO
    28 A2    57    B1    004F   135          CMPW    R7,IRP$W_CHAN(R2)       ;I/O CHANNEL NUMBER MATCH?
```

```
                     E6   12  0053   136          BNEQ    20$                             ;IF NEQ NO
                  52 04 A2   D0  0055   137          MOVL    IRP$L_IOQBL(R2),R2              ;GET BACKWARD LINK OF CURRENT ENTRY
                  51 00 B2   0F  0059   138          REMQUE  @IRP$L_IOQFL(R2),R1             ;REMOVE I/O PACKET FROM QUEUE
               04 2A A1 00   E1  005D   139          BBC     #IRP$V_BUFIO,IRP$W_STS(R1),30$ ;IF CLR, DIRECT I/O REQUEST
                  2A A1 02   AA  0062   140          BICW    #IRP$M_FUNC,IRP$W_STS(R1) ;CLEAR BUFFERED READ
            34 A1  0000'8F   3C  0066   141 30$:     MOVZWL  #SS$_CANCEL,IRP$L_MEDIA(R1) ;SET COMPLETION STATUS
         00000000'FF 61   0E  006C   142          INSQUE  IRP$L_IOQFL(R1),@L^IOC$GL_PSBL ;INSERT PACKET IN POST PROCESS QUEUE
                     C6   12  0073   143          BNEQ    20$                             ;IF NEQ NOT FIRST ENTRY IN QUEUE
                                 0075   144          SOFTINT #IPL$_IOPOST                    ;INITIATE SOFTWARE INTERRUPT
                        14 04 DA 0075                MTPR    #IPL$_IOPOST,S^#PR$_SIRR
                     C1   11  0078   145          BRB     20$                             ;
                  50 00'  3C  007A   146 40$:     MOVZWL  S^#SS$_NORMAL,R0                ;SET NORMAL COMPLETION STATUS
                                 007D   147 50$:     SETIPL  #0                              ;ALLOW INTERRUPTS
                        12 00 DA 007D                MTPR    #0,S^#PR$_IPL
                                 0080   148          RET                                     ;
                                 0081   149 60$:     SETIPL  UCB$B_FIPL(R5)                  ;RAISE TO DRIVER FORK LEVEL
                  12 0B A5   DA  0081                MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
                  50 24 A5   D0  0085   150          MOVL    UCB$L_DDB(R5),R0                ;GET ADDRESS OF DDB
                  50 0C A0   D0  0089   151          MOVL    DDB$L_DDT(R0),R0                ;GET ADDRESS OF DDT
                  53 4C A5   D0  008D   152          MOVL    UCB$L_IRP(R5),R3                ;GET CURRENT I/O PACKET ADDRESS
                     52 57   D0  0091   153          MOVL    R7,R2                           ;SET CHANNEL INDEX
                  51 0C A0   D0  0094   154          MOVL    DDT$L_CANCEL(R0),R1             ;GET ADDRESS OF CANCEL I/O ENTRY POINT
                     E3   13  0098   155          BEQL    50$                             ;IF EQL NONE
                  03 51 10   E0  009A   156          BBS     #16,R1,70$                      ;++++ BRANCH IF SUPERVISOR ABSOLUTE
                     51 50   C0  009E   157          ADDL    R0,R1                           ;++++ OFFSET BY DDT BASE
                     61   16  00A1   158 70$:     JSB     (R1)                            ;CALL CANCEL I/O ROUTINE
                  50 0A A6   3C  00A3   159          MOVZWL  CCB$W_IOC(R6),R0                ;GET OUTSTANDING I/O COUNT
                  50 04 A6   C8  00A7   160          BISL    CCB$L_WIND(R6),R0               ;OUTSTANDING I/O OR FILE ACTIVITY?
                     CD   13  00AB   161          BEQL    40$                             ;IF EQL NO
            C8 34 A5   13   E1  00AD   162          BBC     #DEV$V_MNT,UCB$L_DEVCHAR(R5),40$ ;IF CLR, DEVICE DISMOUNTED
            C3 34 A5   18   E0  00B2   163          BBS     #DEV$V_FOR,UCB$L_DEVCHAR(R5),40$ ;IF SET, MOUNTED FOREIGN
               51 5C 8F   9A  00B7   164          MOVZBL  #IRP$C_LENGTH,R1                ;SET LENGTH OF I/O PACKET
            00000000'EF 16   00BB   165          JSB     EXE$ALONONPAGED                 ;ALLOCATE I/O PACKET                    [04]
                  0B 50   E8  00C1   166          BLBS    R0,100$                         ;IF LBS SUCCESSFUL ALLOCATION
            51 0000'8F   3C  00C4   167          MOVZWL  #SS$_INSFMEM,R1                 ;SET INSUFFICIENT MEMORY STATUS
                  50 03   3C  00C9   168          MOVZWL  #RSN$_NPDYNMEM,R0               ;SET NONPAGED DYNAMIC MEMORY RESOURCE NUMBER
                     FF57 31  00CC   169 90$:     BRW     10$
                     3A A4   B7  00CF   170 100$:    DECW    PCB$W_BIOCNT(R4)                ;UPDATE BUFFERED I/O COUNT
                     0A A6   B6  00D2   171          INCW    CCB$W_IOC(R6)                   ;INCREMENT I/O COUNT
                  53 52   D0  00D5   172          MOVL    R2,R3                           ;COPY ADDRESS OF I/O PACKET
                  52 08   C0  00D8   173          ADDL    #IRP$W_SIZE,R2                  ;CALCULATE ADDRESS OF PACKET SIZE
               82 5C 8F   9B  00DB   174          MOVZBW  #IRP$C_LENGTH,(R2)+             ;INSERT SIZE OF I/O REQUEST PACKET
                  82 0A   9B  00DF   175          MOVZBW  #DYN$C_IRP,(R2)+                ;INSERT DATA STRUCTURE TYPE AND ZERO MODE
               82 60 A4   D0  00E2   176          MOVL    PCB$L_PID(R4),(R2)+             ;INSERT CURRENT PROCESS ID
                     82   7C  00E6   177          CLRQ    (R2)+                           ;CLEAR AST ADDRESS AND PARAMETER
               82 04 A6   D0  00E8   178          MOVL    CCB$L_WIND(R6),(R2)+            ;INSERT ADDRESS OF WINDOW
                  82 55   D0  00EC   179          MOVL    R5,(R2)+                        ;INSERT DEVICE UCB ADDRESS
                  82 38   B0  00EF   180          MOVW    #IO$_ACPCONTROL,(R2)+           ;INSERT I/O FUNCTION CODE
                  82 1F   90  00F2   181          MOVB    #31,(R2)+                       ;SET EVENT FLAG NUMBER
                  82 00   90  00F5   182          MOVB    #0,(R2)+                        ;INSERT PROCESS BASE PRIORITY
                     82   D4  00F8   183          CLRL    (R2)+                           ;CLEAR I/O STATUS BLOCK ADDRESS
                  82 57   3C  00FA   184          MOVZWL  R7,(R2)+                        ;INSERT CHANNEL NUMBER AND CLEAR STATUS
                     82   7C  00FD   185          CLRQ    (R2)+                           ;CLEAR BUFFER PARAMETERS
                  50 30 A5   D0  00FF   186          MOVL    UCB$L_VCB(R5),R0                ;GET ADDRESS OF VCB
                     0C A0   B6  0103   187          INCW    VCB$W_TRANS(R0)                 ;UPDATE TRANSACTION COUNT
                                 0106   188
                                 0106   189
```

```
                    FEF7'  31  0106   190          BRW     EXE$QIOACPPKT           ;QUEUE ACP PACKET
                               0109   191
                               0109   192          .END
```

$ER	=	00000001		D		IRP$L_IOSB		00000024		D
$MODULE		00000000	R	D	02	IRP$L_IOST1		00000034		D
CCB$B_AMOD		00000009		D		IRP$L_IOST2		00000038		D
CCB$B_STS		00000008		D		IRP$L_MEDIA		00000034		D
CCB$C_LENGTH		00000010		D		IRP$L_PID		0000000C		D
CCB$K_LENGTH		00000010		D		IRP$L_SEGVBN		00000040		D
CCB$L_DIRP		0000000C		D		IRP$L_SEQNUM		00000048		D
CCB$L_UCB		00000000		D		IRP$L_SVAPTE		0000002C		D
CCB$L_WIND		00000004		D		IRP$L_TT_TERM		00000038		D
CCB$W_IOC		0000000A		D		IRP$L_UCB		0000001C		D
CHAN	=	00000004		D		IRP$L_WIND		00000018		D
DDB$B_ACPCLASS		00000013		D		IRP$M_FUNC	=	00000002		D
DDB$B_TYPE		0000000A		D		IRP$Q_NT_PRVMSK		0000003C		D
DDB$C_LENGTH		00000034		D		IRP$V_BUFIO	=	00000000		D
DDB$K_LENGTH		00000034		D		IRP$V_VIRTUAL	=	00000004		D
DDB$L_ACPD		00000010		D		IRP$W_ABCNT		0000003C		D
DDB$L_DDT		0000000C		D		IRP$W_BCNT		00000032		D
DDB$L_LINK		00000000		D		IRP$W_BOFF		00000030		D
DDB$L_UCB		00000004		D		IRP$W_CHAN		00000028		D
DDB$T_DRVNAME		00000024		D		IRP$W_FUNC		00000020		D
DDB$T_NAME		00000014		D		IRP$W_OBCNT		0000003E		D
DDB$W_SIZE		00000008		D		IRP$W_SIZE		00000008		D
DDT$L_ALTSTART		0000001C		D		IRP$W_STS		0000002A		D
DDT$L_CANCEL		0000000C		D		IRP$W_TT_PRMPT		0000003C		D
DDT$L_FDT		00000008		D		PCB$B_ASTACT		0000000C		D
DDT$L_REGDUMP		00000010		D		PCB$B_ASTEN		0000000D		D
DDT$L_START		00000000		D		PCB$B_PRI		0000000B		D
DDT$L_UNITINIT		00000018		D		PCB$B_PRIB		0000002F		D
DDT$L_UNSOLINT		00000004		D		PCB$B_TYPE		0000000A		D
DDT$W_DIAGBUF		00000014		D		PCB$B_WEFC		0000002E		D
DDT$W_ERRORBUF		00000016		D		PCB$C_LENGTH		0000008C		D
DEV$V_FOR	=	00000018		D		PCB$K_LENGTH		0000008C		D
DEV$V_MNT	=	00000013		D		PCB$L_ARB		00000084		D
DS$AX_SOFTPCB		********		X	02	PCB$L_ASTQBL		00000014		D
DYN$C_IRP	=	0000000A		D		PCB$L_ASTQFL		00000010		D
EXE$ALONONPAGED		********		X	02	PCB$L_EFC2P		00000058		D
EXE$CANCEL		00000007	RG	D	02	PCB$L_EFC3P		0000005C		D
EXE$QIOACPPKT		********		X	02	PCB$L_EFCS		00000050		D
IO$_ACPCONTROL	=	00000038		D		PCB$L_EFCU		00000054		D
IOC$GL_PSBL		********		X	02	PCB$L_EFWM		0000004C		D
IOC$VERIFYCHAN		********		X	02	PCB$L_JIB		00000078		D
IPL$_ASTDEL	=	00000002		D		PCB$L_OWNER		0000001C		D
IPL$_IOPOST	=	00000004		D		PCB$L_PHD		00000064		D
IRP$B_CARCON		00000038		D		PCB$L_PHYPCB		00000018		D
IRP$B_EFN		00000022		D		PCB$L_PID		00000060		D
IRP$B_PRI		00000023		D		PCB$L_PQB		0000004C		D
IRP$B_RMOD		0000000B		D		PCB$L_SQBL		00000004		D
IRP$B_TYPE		0000000A		D		PCB$L_SQFL		00000000		D
IRP$C_LENGTH		0000005C		D		PCB$L_STS		00000024		D
IRP$K_LENGTH		0000005C		D		PCB$L_UIC		00000088		D
IRP$L_ARB		00000050		D		PCB$L_WSSWP		00000020		D
IRP$L_AST		00000010		D		PCB$L_WTIME		00000028		D
IRP$L_ASTPRM		00000014		D		PCB$Q_PRIV		0000007C		D
IRP$L_DIAGBUF		00000044		D		PCB$T_LNAME		00000068		D
IRP$L_EXTEND		0000004C		D		PCB$T_TERMINAL		00000044		D
IRP$L_IOQBL		00000004		D		PCB$W_APTCNT		00000030		D
IRP$L_IOQFL		00000000		D		PCB$W_ASTCNT		00000038		D

PCB$W_BIOCNT		0000003A	D		UCB$L_ASTQBL	00000010	D
PCB$W_BIOLM		0000003C	D		UCB$L_ASTQFL	0000000C	D
PCB$W_DIOCNT		0000003E	D		UCB$L_CPID	0000005C	D
PCB$W_DIOLM		00000040	D		UCB$L_CRB	00000020	D
PCB$W_GPGCNT		00000034	D		UCB$L_DDB	00000024	D
PCB$W_GRP		0000008A	D		UCB$L_DEVCHAR	00000034	D
PCB$W_MEM		00000088	D		UCB$L_DEVDEPEND	0000003C	D
PCB$W_MTXCNT		0000000E	D		UCB$L_DPC	00000080	D
PCB$W_PPGCNT		00000036	D		UCB$L_DUETIM	0000005C	D
PCB$W_PRCCNT		00000042	D		UCB$L_DX_BFPNT	0000009C	D
PCB$W_SIZE		00000008	D		UCB$L_DX_BUF	00000098	D
PCB$W_STATE		0000002C	D		UCB$L_DX_RXDB	000000A0	D
PCB$W_TMBU		00000032	D		UCB$L_EMB	00000078	D
PR$_IPL	=	00000012	D		UCB$L_FIRST	00000014	D
PR$_SIRR	=	00000014	D		UCB$L_FPC	0000000C	D
RSN$_NPDYNMEM	=	00000003	D		UCB$L_FQBL	00000004	D
SIZ...	=	00000001	D		UCB$L_FQFL	00000000	D
SS$_CANCEL		********	X	02	UCB$L_FR3	00000010	D
SS$_INSFMEM		********	X	02	UCB$L_FR4	00000014	D
SS$_NORMAL		********	X	02	UCB$L_IOQBL	00000044	D
UCB$B_AMOD		00000053	D		UCB$L_IOQFL	00000040	D
UCB$B_CEX		00000077	D		UCB$L_IRP	0000004C	D
UCB$B_CM1		0000004A	D		UCB$L_LINK	0000002C	D
UCB$B_CM2		0000004B	D		UCB$L_LOGADR	00000064	D
UCB$B_DEVCLASS		00000038	D		UCB$L_MAXBLOCK	00000084	D
UCB$B_DEVTYPE		00000039	D		UCB$L_MB_MBX	0000007C	D
UCB$B_DIPL		00000052	D		UCB$L_MB_PORT	0000008C	D
UCB$B_DX_SCTCNT		000000A6	D		UCB$L_MB_RAST	00000078	D
UCB$B_ERTCNT		00000070	D		UCB$L_MB_SHB	00000080	D
UCB$B_ERTMAX		00000071	D		UCB$L_MB_WAST	00000074	D
UCB$B_FEX		00000076	D		UCB$L_MB_WIOQBL	00000088	D
UCB$B_FIPL		0000000B	D		UCB$L_MB_WIOQFL	00000084	D
UCB$B_LOCSRV		0000003C	D		UCB$L_MEDIA	0000008C	D
UCB$B_OFFNDX		00000094	D		UCB$L_NT_DATSSB	00000074	D
UCB$B_OFFRTC		00000095	D		UCB$L_NT_INTSSB	00000078	D
UCB$B_REMSRV		0000003D	D		UCB$L_OPCNT	00000060	D
UCB$B_SECTORS		0000003C	D		UCB$L_OWNUIC	0000001C	D
UCB$B_SLAVE		00000074	D		UCB$L_PID	00000028	D
UCB$B_SPR		00000075	D		UCB$L_RQBL	00000004	D
UCB$B_STATE		00000052	D		UCB$L_RQFL	00000000	D
UCB$B_TRACKS		0000003D	D		UCB$L_SVAPTE	00000068	D
UCB$B_TT_CRFILL		0000009D	D		UCB$L_SVPN	00000064	D
UCB$B_TT_DECRF		000000A1	D		UCB$L_TT_DECHAR	000000A8	D
UCB$B_TT_DELFF		000000A2	D		UCB$L_TT_RDUE	0000008C	D
UCB$B_TT_DESPEE		000000A0	D		UCB$L_TT_RTIMOU	000000B8	D
UCB$B_TT_DETYPE		000000A4	D		UCB$L_VCB	00000030	D
UCB$B_TT_LFFILL		0000009E	D		UCB$T_PARTNER	0000000C	D
UCB$B_TT_SPEED		0000009C	D		UCB$W_BCNT	0000006E	D
UCB$B_TYPE		0000000A	D		UCB$W_BCR	00000096	D
UCB$B_VERTSZ		0000003F	D		UCB$W_BOFF	0000006C	D
UCB$C_LENGTH		00000074	D		UCB$W_BUFQUO	00000018	D
UCB$C_MB_LENGTH		00000090	D		UCB$W_BYTESTOGO	0000003E	D
UCB$C_TT_LENGTH		000000BC	D		UCB$W_CHARGE	0000004A	D
UCB$K_LENGTH		00000074	D		UCB$W_CYLINDERS	0000003E	D
UCB$K_MB_LENGTH		00000090	D		UCB$W_DA	0000008C	D
UCB$K_TT_LENGTH		000000BC	D		UCB$W_DC	0000008E	D
UCB$L_AMB		00000054	D		UCB$W_DEVBUFSIZ	0000003A	D

UCB$W_DEVSTS	0000005A	D	VCB$L_RVT	00000020	D		
UCB$W_DIRSEQ	00000088	D	VCB$L_SBMAPLBN	00000034	D		
UCB$W_DSTADDR	00000018	D	VCB$L_START_FID	00000028	D		
UCB$W_DX_BCR	000000A4	D	VCB$L_ST_RECORD	00000030	D		
UCB$W_EC1	00000090	D	VCB$L_USRLBLAST	00000044	D		
UCB$W_EC2	00000092	D	VCB$L_VPBL	00000040	D		
UCB$W_ERRCNT	00000072	D	VCB$L_VPFL	0000003C	D		
UCB$W_FUNC	0000007E	D	VCB$L_WCB	00000038	D		
UCB$W_MB_SEED	00000000	D	VCB$T_QNAME	0000000C	D		
UCB$W_MSGCNT	00000016	D	VCB$T_VOLNAME	00000014	D		
UCB$W_MSGMAX	00000014	D	VCB$W_CLUSTER	0000003C	D		
UCB$W_NT_CHAN	0000007C	D	VCB$W_CUR_NUM	00000024	D		
UCB$W_OFFSET	0000008A	D	VCB$W_CUR_SEQ	00000026	D		
UCB$W_REFC	00000050	D	VCB$W_EXTEND	0000003E	D		
UCB$W_SIZE	00000008	D	VCB$W_FILEPROT	0000004A	D		
UCB$W_SRCADDR	0000001A	D	VCB$W_MCOUNT	0000004C	D		
UCB$W_STS	00000058	D	VCB$W_MODE	0000002C	D		
UCB$W_TT_DESIZE	000000A5	D	VCB$W_PENDERR	00000062	D		
UCB$W_UNIT	00000048	D	VCB$W_QUOSIZE	00000060	D		
UCB$W_VPROT	0000001A	D	VCB$W_RECORDSZ	00000050	D		
VCB$B_BLOCKFACT	00000052	D	VCB$W_RVN	0000000E	D		
VCB$B_CUR_RVN	0000002F	D	VCB$W_SIZE	00000008	D		
VCB$B_EOFDELTA	0000004E	D	VCB$W_START_NUM	00000028	D		
VCB$B_IBMAPSIZE	00000038	D	VCB$W_START_SEQ	0000002A	D		
VCB$B_IBMAPVBN	0000003A	D	VCB$W_TRANS	0000000C	D		
VCB$B_LRU_LIM	00000049	D	WCB$B_ACCESS	0000000B	D		
VCB$B_QNAMECNT	0000000B	D	WCB$B_TYPE	0000000A	D		
VCB$B_RESFILES	0000004F	D	WCB$C_LENGTH	00000024	D		
VCB$B_SBMAPSIZE	00000039	D	WCB$C_MAP	00000024	D		
VCB$B_SBMAPVBN	0000003B	D	WCB$K_LENGTH	00000024	D		
VCB$B_STATUS	0000000B	D	WCB$K_MAP	00000024	D		
VCB$B_STATUS2	00000053	D	WCB$L_FCB	00000018	D		
VCB$B_TM	0000002E	D	WCB$L_LBN	00000002	D		
VCB$B_TYPE	0000000A	D	WCB$L_ORGUCB	00000010	D		
VCB$B_WINDOW	00000048	D	WCB$L_P1_LBN	00000026	D		
VCB$C_COMLEN	00000024	D	WCB$L_P2_LBN	0000002C	D		
VCB$C_LENGTH	00000064	D	WCB$L_PID	0000000C	D		
VCB$C_MRKLEN	0000000B	D	WCB$L_PREVLBN	FFFFFFFC	D		
VCB$K_COMLEN	00000024	D	WCB$L_RVT	0000001C	D		
VCB$K_LENGTH	00000064	D	WCB$L_STVBN	00000020	D		
VCB$K_MRKLEN	0000000B	D	WCB$L_WLBL	00000004	D		
VCB$L_AQB	00000010	D	WCB$L_WLFL	00000000	D		
VCB$L_BLOCKBL	00000004	D	WCB$W_ACON	00000014	D		
VCB$L_BLOCKFL	00000000	D	WCB$W_COUNT	00000000	D		
VCB$L_CACHE	00000058	D	WCB$W_NMAP	00000016	D		
VCB$L_CUR_FID	00000024	D	WCB$W_P1_COUNT	00000024	D		
VCB$L_FCBBL	00000004	D	WCB$W_P2_COUNT	0000002A	D		
VCB$L_FCBFL	00000000	D	WCB$W_PREVCOUNT	FFFFFFFA	D		
VCB$L_FREE	00000040	D	WCB$W_REFCNT	0000000E	D		
VCB$L_HOME2LBN	00000028	D	WCB$W_SIZE	00000008	D		
VCB$L_HOMELBN	00000024	D					
VCB$L_IBMAPLBN	00000030	D					
VCB$L_IXHDR2LBN	0000002C	D					
VCB$L_MAXFILES	00000044	D					
VCB$L_MVL	00000034	D					
VCB$L_QUOCACHE	0000005C	D					
VCB$L_QUOTAFCB	00000054	D					

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000109 (265.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	79 (1)	
$MODULE	00000000-R	79 (1)	
CCB$L_UCB	00000000		#-121 (1)
CCB$L_WIND	00000004		#-160 (1) #-178 (1)
CCB$W_IOC	0000000A		#-124 (1) #-159 (1) #-171 (1)
CHAN	=00000004	77 (1)	#-116 (1)
DDB$L_DDT	0000000C		#-151 (1)
DDT$L_CANCEL	0000000C		#-154 (1)
DEV$V_FOR	=00000018		#-163 (1)
DEV$V_MNT	=00000013		#-162 (1)
DS$AX_SOFTPCB	00000000-XR		115 (1)
DYN$C_IRP	=0000000A		#-175 (1)
EXE$ALONONPAGED	00000000-XR		165 (1)
EXE$CANCEL	00000007-R	114 (1)	
EXE$QIOACPPKT	00000000-XR		#-190 (1)
IO$_ACPCONTROL	=00000038		#-180 (1)
IOC$GL_PSBL	00000000-XR		142 (1)
IOC$VERIFYCHAN	00000000-XR		117 (1)
IPL$_ASTDEL	=00000002		#-123 (1)
IPL$_IOPOST	=00000004		#-144 (1)
IRP$C_LENGTH	0000005C		#-164 (1) #-174 (1)
IRP$L_IOQBL	00000004		#-137 (1)
IRP$L_IOQFL	00000000		#-129 (1) 138 (1) 142 (1)
IRP$L_MEDIA	00000034		#-141 (1)
IRP$L_PID	0000000C		#-133 (1)
IRP$M_FUNC	=00000002		#-140 (1)
IRP$V_BUFIO	=00000000		#-139 (1)
IRP$V_VIRTUAL	=00000004		#-132 (1)
IRP$W_CHAN	00000028		#-135 (1)
IRP$W_SIZE	00000008		#-173 (1)
IRP$W_STS	0000002A		132 (1) 139 (1) #-140 (1)
PCB$L_PID	00000060		#-133 (1) #-176 (1)
PCB$W_BIOCNT	0000003A		#-170 (1)
PR$_IPL	=00000012		#-123 (1) #-128 (1) #-147 (1) #-149 (1)
PR$_SIRR	=00000014		#-144 (1)
RSN$_NPDYNMEM	=00000003		#-168 (1)
SS$_CANCEL	00000000-XR		#-141 (1)
SS$_INSFMEM	00000000-XR		#-167 (1)
SS$_NORMAL	00000000-XR		#-146 (1)
UCB$B_FIPL	0000000B		#-128 (1) #-149 (1)
UCB$L_DDB	00000024		#-150 (1)
UCB$L_DEVCHAR	00000034		162 (1) 163 (1)
UCB$L_IOQFL	00000040		126 (1)
UCB$L_IRP	0000004C		#-152 (1)
UCB$L_VCB	00000030		#-186 (1)
VCB$W_TRANS	0000000C		#-187 (1)

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$CADEF          1           55      (1)     55      (1)
$CCBDEF         1           56      (1)     56      (1)
$DDBDEF         1           57      (1)     57      (1)
$DDTDEF         1           58      (1)     58      (1)
$DEFINI         1           55      (1)     55      (1)     56      (1)     57      (1)     58      (1)     59      (1)
                                            60      (1)     61      (1)     62      (1)     63      (1)     64      (1)
                                            65      (1)     66      (1)     67      (1)     68      (1)     69      (1)
$DEVDEF         1           59      (1)     59      (1)
$DYNDEF         2           60      (1)     60      (1)
$IODEF          15          61      (1)     61      (1)
$IPLDEF         1           62      (1)     62      (1)
$IRPDEF         4           63      (1)     63      (1)
$PCBDEF         4           64      (1)     64      (1)
$PRDEF          5           65      (1)     65      (1)
$RSNDEF         1           66      (1)     66      (1)
$UCBDEF         10          67      (1)     67      (1)
$VCBDEF         6           68      (1)     68      (1)
$WCBDEF         3           69      (1)     69      (1)
MODNAM          1           79      (1)     79      (1)
SETIPL          1           123     (1)     123     (1)     128     (1)     147     (1)     149     (1)
SOFTINT         1           144     (1)     144     (1)
```

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDL            00C0      157    (1)    173    (1)
BBC             00E1      139    (1)    162    (1)
BBS             00E0      132    (1)    156    (1)    163    (1)
BEQL            0013      125    (1)    131    (1)    155    (1)    161    (1)
BICW            00AA      140    (1)
BISL            00C8      160    (1)
BLBC            00E9      118    (1)
BLBS            00E8      166    (1)
BNEQ            0012      134    (1)    136    (1)    143    (1)
BRB             0011      145    (1)
BRW             0031      169    (1)    190    (1)
CLRL            00D4      183    (1)
CLRQ            007C      177    (1)    185    (1)
CMPL            00D1      130    (1)    133    (1)
CMPW            00B1      135    (1)
DECW            00B7      170    (1)
INCW            00B6      171    (1)    187    (1)
INSQUE          000E      142    (1)
JSB             0016      117    (1)    158    (1)    165    (1)
MOVAB           009E      115    (1)    126    (1)
MOVB            0090      181    (1)    182    (1)
MOVL            00D0      119    (1)    120    (1)    121    (1)    127    (1)    129    (1)    137    (1)    150    (1)
                          151    (1)    152    (1)    153    (1)    154    (1)    172    (1)    176    (1)    178    (1)
                          179    (1)    186    (1)
MOVPSL          00DC      122    (1)
MOVW            00B0      180    (1)
MOVZBL          009A      164    (1)
MOVZBW          009B      174    (1)    175    (1)
MOVZWL          003C      116    (1)    141    (1)    146    (1)    159    (1)    167    (1)    168    (1)    184    (1)
MTPR            00DA      123    (1)    128    (1)    144    (1)    147    (1)    149    (1)
REMQUE          000F      138    (1)
RET             0004      148    (1)
TSTW            00B5      124    (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	79	(1)														
.END	192	(1)														
.ENDC	123	(1)	128	(1)	147	(1)	149	(1)								
.ENDM	123	(1)	144	(1)	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)
	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)
	69	(1)	79	(1)												
.ENTRY	114	(1)														
.IDENT	6	(1)														
.IF	123	(1)	128	(1)	147	(1)	149	(1)								
.IFF	123	(1)	128	(1)	147	(1)	149	(1)								
.NOCROSS	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)	69	(1)		
.PAGE	80	(1)														
.PSECT	78	(1)														
.RESTORE	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)	69	(1)		
.SAVE	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)	69	(1)		
.SBTTL	81	(1)														
.TITLE	5	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	1	#-116	(1)										
R0	14	#-116	(1)	#-118	(1)	#-146	(1)	#-150	(1)	#-151	(1)	#-154	(1)
		#-157	(1)	#-159	(1)	#-160	(1)	#-166	(1)	#-168	(1)	#-186	(1)
		#-187	(1)										
R1	12	#-120	(1)	#-138	(1)	139	(1)	#-140	(1)	#-141	(1)	142	(1)
		#-154	(1)	156	(1)	#-157	(1)	158	(1)	#-164	(1)	#-167	(1)
R2	27	114	(1)	#-119	(1)	#-127	(1)	#-129	(1)	#-130	(1)	132	(1)
		#-133	(1)	#-135	(1)	#-137	(1)	138	(1)	#-153	(1)	#-172	(1)
		#-173	(1)	#-174	(1)	#-175	(1)	#-176	(1)	#-177	(1)	#-178	(1)
		#-179	(1)	#-180	(1)	#-181	(1)	#-182	(1)	#-183	(1)	#-184	(1)
		#-185	(1)										
R3	6	114	(1)	#-126	(1)	#-127	(1)	#-130	(1)	#-152	(1)	#-172	(1)
R4	5	114	(1)	#-115	(1)	#-133	(1)	#-170	(1)	#-176	(1)		
R5	11	114	(1)	#-121	(1)	126	(1)	#-128	(1)	#-149	(1)	#-150	(1)
		#-152	(1)	162	(1)	163	(1)	#-179	(1)	#-186	(1)		
R6	8	114	(1)	#-120	(1)	#-121	(1)	#-124	(1)	#-159	(1)	#-160	(1)
		#-171	(1)	#-178	(1)								
R7	5	114	(1)	#-119	(1)	#-135	(1)	#-153	(1)	#-184	(1)		
SP	1	#-122	(1)										

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.21
Command processing	868	00:00:00.21	00:00:00.84
Pass 1	862	00:00:03.10	00:00:04.64
Symbol table sort	0	00:00:00.27	00:00:00.27
Pass 2	84	00:00:00.82	00:00:01.34
Symbol table output	11	00:00:00.06	00:00:00.17
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	4	00:00:00.08	00:00:00.15
Assembler run totals	1854	00:00:04.57	00:00:07.62

The working set limit was 2512 pages.
191815 bytes (375 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1052 non-local and 9 local symbols.
192 source lines were read in Pass 1, producing 70 object records in Pass 2.
112 pages of virtual memory were used to define 27 macros.

```
                          +---------------------------+
                          ! Macro library statistics !
                          +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8              11
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9                 1
SYS$COMMON:[SYSLIB]LIB.MLB;2                            4
SYS$COMMON:[SYSLIB]STARLET.MLB;2                        7
TOTALS (all libraries)                                 23

1496 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CANCEL.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(2) 78 DECLARATIONS
(6) 230 $CHANNEL CALL
(7) 290 $CHANNEL CALL - ADAPTER INIT FUNCTION
(8) 323 $CHANNEL CALL - BUS INIT FUNCTION
(9) 356 $CHANNEL CALL - ABORT FUNCTION
(10) 388 $CHANNEL CALL - ENABLE INTERRUPTS
(11) 435 $CHANNEL CALL - PURGE
(12) 467 $CHANNEL CALL - DISABLE INTERRUPTS
(13) 503 $CHANNEL CALL - CLEAR ADAPTER
(14) 535 $CHANNEL CALL - GET ADAPTER STATUS
(15) 577 $CHANNEL CALL - SET DEFEAT PARITY
(16) 619 $CHANNEL CALL - CLEAR DEFEAT PARITY
(17) 661 $CHANNEL CALL - COMMON CALL RETURN
(18) 705 $SETMAP CALL
(19) 1010 DSP$SHOWMBA
(20) 1051 DSP$SHOWUBI
(21) 1095 $SHOWCHAN CALL
(22) 1147 SHOW_UBI STATUS
(23) 1176 SHOWBITS
(24) 1214 DETERMINE ADAPTER SPECIFIC STATUS
(25) 1244 DETERMINE GENERAL ADAPTER STATUS
(26) 1274 DETERMINE ADAPTER HARDWARE ERROR STATUS
(28) 1305 BUILD THE CHANNEL DATA BLOCK
(29) 1366 SET ADAPTER VECTOR(S)
(30) 1413 CLEAR ADAPTER VECTOR(S)
(31) 1454 INTERRUPT PRE-PROCESS

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CHAN730.MAR
                        0000     2 ; *2    12-MAY-1986 10:49:28 BARANSKI "Cleaning up Object Libraries."
                        0000     3 ; *1     6-FEB-1986 15:14:04 DS ""
                        0000     4 ; DEC/CMS REPLACEMENT HISTORY, Element CHAN730.MAR
                        0000     5         .TITLE  CHAN730 *** CHAN730 Channel services for NEBULA
                        0000     6         .IDENT  /06-11/
                        0000     7         .NLIST  CND
                        0000     8         .LIST   MEB
                        0000     9
                        0000    10 ;
                        0000    11 ; Copyright (c) 1979, 1981, 1983
                        0000    12 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000    13 ;
                        0000    14 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    15 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    16 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    17 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    18 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    19 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    20 ; REMAIN IN DEC.
                        0000    21 ;
                        0000    22 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    23 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    24 ; CORPORATION.
                        0000    25 ;
                        0000    26 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    27 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    28
                        0000    29 ;++
                        0000    30 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
                        0000    31 ;
                        0000    32 ; ABSTRACT:
                        0000    33 ;
                        0000    34 ; ENVIRONMENT:
                        0000    35 ;
                        0000    36 ; AUTHOR:       Roger Riggs     10-Sept-1979    VERSION 01
                        0000    37 ;
                        0000    38 ; MODIFIED BY:
                        0000    39 ;               Roger Riggs, 14-Jan-1979, Version 5.2
                        0000    40 ;         01    Added lists for CVT$T_.... strings referenced in CVTREG
                        0000    41 ;         02    Added MTPR #1,#^X37 to do I/O subsystem clear on INITA
                        0000    42 ;               Roger Riggs, 21-MAY-1980,       Version 5.4
                        0000    43 ;         03    CLRMAP was clearing 512 map register, only 496 exist
                        0000    44 ;               Also fixed algorithm for determining how many maps to set
                        0000    45 ;               from starting address and byte count
                        0000    46 ;               Roger Riggs, 26-Jun-1980,       Version 5.5
                        0000    47 ;         04    Fixed bug which did not allocate enough map registers
                        0000    48 ;               for transfer of 1+n*pages.
                        0000    49 ;
                        0000    50 ;         05    - Jack Stansbury, 23-Oct-1981, Version 6.-
                        0000    51 ;               Removed all calls to $DS_BNERROR and $DS_BERROR since these
                        0000    52 ;               should be used only by the diagnostics. Replaced them with
                        0000    53 ;               their BLBx equivalents. Also added .LIBRARY statements for
                        0000    54 ;               $DS and $DIAG.
                        0000    55 ;
                        0000    56 ;         06    - Jack Stansbury, 26-Oct-1981, Version 6.-
                        0000    57 ;               Changed the .PSECT SEP to WORK, CODE, and DATA
```

```
0000     58 ;                  where appropriate.
0000     59 ;
0000     60 ;        07        - Dave Butenhof, 23-Nov-1981, verison 6.5
0000     61 ;                  Add .LIBRARY statement for DS.MLB, and fixed some
0000     62 ;                  truncation errors.
0000     63 ;
0000     64 ;        08        - Dave Butenhof, 09-May-1981, version 6.5
0000     65 ;                  Add CHM$_INVALIDATE function to DSX$SETMAP routine
0000     66 ;
0000     67 ;        09        - Dave Butenhof, 09-May-1981, version 6.5
0000     68 ;                  Fix assorted register errors
0000     69 ;
0000     70 ;        10        - Dave Butenhof, 17-Dec-1981, Version 6.6
0000     71 ;                  Fix problem with map invalidate: it assumed 496 map
0000     72 ;                  vectors, while there are really 512.
0000     73 ;
0000     74 ;        11        Bob Bergazzi     May 16,1983     Version 6.11
0000     75 ;                  Changed the order of the .LIB statements.
0000     76 ;--
```

```
                 0000          .SBTTL  DECLARATIONS
                 0000    ;
                 0000    ; INCLUDE FILES:
                 0000    ;
                 0000
                 0000    .LIBRARY        /SYS$LIBRARY:LIB/       ;                                [11]
                 0000    .LIBRARY        /$DS/                   ;                                [11]
                 0000    .LIBRARY        /$DIAG/                 ;                                [11]
                 0000
                 0000    ;
                 0000    ; MACROS:
                 0000    ;
                 0000
                 0000    .MACRO  $CHNDEF
                 0000            $OFFDEF CHN,<UNIT,FUNC,ADR1,ADR2>
                 0000    .ENDM   $CHNDEF
                 0000    .MACRO  $STMDEF
                 0000            $OFFDEF STM,<UNIT,FUNC,PHY,BAS,CNT,PTH>
                 0000    .ENDM   $STMDEF
                 0000    .MACRO  $SHWDEF
                 0000            $OFFDEF SHW,<UNIT>
                 0000    .ENDM   $SHWDEF
                 0000
                 0000    ;
                 0000    ; EQUATED SYMBOLS:
                 0000    ;
                 0000
                 0000    $DS_DSDEF
                 0000    $UBADEF                                 ; UBA REGISTER DEFINITIONS
                 0000            .SAVE   LOCAL_BLOCK
                 0000            .RESTORE
                 0000    ;$UBIDEF                                        ; UBI REGISTER DEFINITIONS
                 0000    ; TEMPORARY UBI DEFINITIONS
        00000000 0000    UBI$L_CSR       =       ^X0
        00000800 0000    UBI$L_MAP       =       ^X800
        0000001F 0000    UBI$V_MAP_VALID =       31
        80000000 0000    UBI$M_MAP_VALID = 1 @ UBI$V_MAP_VALID
        00000019 0000    UBI$V_MAP_BO    =       25
        02000000 0000    UBI$M_MAP_BO    = 1 @ UBI$V_MAP_BO
        00000001 0000    UBI$S_MAP_BO    =       1
        00000015 0000    UBI$V_MAP_DPD   =       21
        00000002 0000    UBI$S_MAP_DPD   =       2
        00000000 0000    UBI$V_MAP_PFN   =       0
        0000000F 0000    UBI$S_MAP_PFN   =       15
                 0000
                 0000
        006600B1 0000    DS$_NOSUPPORT   =       DS$_NOTIMP!1    ; ALLOW UNSUPPORTED
                 0000                                            ; FUNCTIONS TO SUCCEDE FOR NOW
                 0000    ; END OF TEMPORARY DEFINITIONS
                 0000    $DS_BITDEF                              ; BIT DEFINITIONS
                 0000    $DS_DSDEF                               ; PROCEDURE RETURN CODE DEFINITIONS
                 0000    $DS_PARDEF                              ; PARAMETER ARG DEFINITIONS
                 0000    $DS_HPODEF                              ; HARDWARE P-TABLE DEFINITIONS
                 0000            .SAVE   LOCAL_BLOCK
                 0000            .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
        00000008 0000            .IIF    NB,.BLKQ,       .BLKQ   1
                 0008            .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
```

```
0000000A  0008              .IIF    NB,.BLKW,       .BLKW   1
          000A              .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
0000000B  000A              .IIF    NB,.BLKB,       .BLKB   1
          000B              .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
0000000C  000B              .IIF    NB,.BLKB,       .BLKB   1
          000C              .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
00000018  000C              .IIF    NB,.BLKB,       .BLKB   12
          0018              .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
0000001C  0018              .IIF    NB,.BLKL,       .BLKL   1
          001C              .IIF    NB,HP$A_DVA,    HP$A_DVA:
00000020  001C              .IIF    NB,.BLKL,       .BLKL   1
          0020              .IIF    NB,HP$A_LINK,   HP$A_LINK:
00000024  0020              .IIF    NB,.BLKL,       .BLKL   1
          0024              .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
00000026  0024              .IIF    NB,.BLKW,       .BLKW   1
          0026              .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
00000032  0026              .IIF    NB,.BLKB,       .BLKB   12
          0032              .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
          0000              .RESTORE
          0000    132 $DS_CHDEF                               ; CHANNEL SERVICE DEFINITIONS
          0000    133 $CHNDEF                                 ; $DS_CHANNEL CALL
          0000    134 $PRDEF                                  ; PROCESSOR REGISTERS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0032              .=0
          0000              .RESTORE
          0000    135 $STMDEF                                 ; $DS_SETMAP CALL
          0000    136 $SHWDEF                                 ; $DS_SHOWCHAN CALL
          0000    140
00000080  0000    141 SHW$K_BUFSIZ    =       128             ; $SHOWCHAN BUFFER SIZE
          0000    142
          0000    143 ; INTERNAL FLAG WORD DEFINITIONS
          0000    144
00000000  0000    145 CDB$V_MBA     = 0                       ; ADAPTER IS MBA (11/750,11/780)
00000001  0000    146 CDB$V_UBA     = 1                       ; ADAPTER IS UBA (11/780)
00000002  0000    147 CDB$V_UBI     = 2                       ; ADAPTER IS UBI (11/750)
00000003  0000    148 CDB$V_OFFSET  = 3                       ; OFFSET MODE
00000004  0000    149 CDB$V_REV     = 4                       ; REVERSE OPERATION
00000005  0000    150 CDB$V_IE      = 5                       ; INTERRUPTS ENABLED
00000001  0000    151 CDB$M_MBA     = 1 @ CDB$V_MBA
00000002  0000    152 CDB$M_UBA     = 1 @ CDB$V_UBA
00000004  0000    153 CDB$M_UBI     = 1 @ CDB$V_UBI
00000008  0000    154 CDB$M_OFFSET  = 1 @ CDB$V_OFFSET
00000010  0000    155 CDB$M_REV     = 1 @ CDB$V_REV
00000020  0000    156 CDB$M_IE      = 1 @ CDB$V_IE
          0000    157
          0000    158 ; STATUS BIT DEFINITIONS FOR CDB$L_ST1
          0000    159
0000000F  0000    160 CHS$M_ERRANY    =       CHS$M_SYSERR!CHS$M_CHNERR!CHS$M_DEVERR!CHS$M_PGMERR
          0000    161
```

```
                                00000000   163          .PSECT  Work, NoExe, NoShr, Wrt, Long          ;                    [06]
                                    0000   164
                                    0000   165 ;
                                    0000   166 ; OWN STORAGE:
                                    0000   167 ;
                                    0000   168
                                    0000   169 .SAVE
                                    0000   170 .PSECT  $ABS$,ABS
                           00000000 0032   171 .=0
                           00000004 0000   172 CDB$L_FLAGS:     .BLKL   1               ; INTERNAL FLAGS
                           00000008 0004   173 CDB$A_TBL:       .BLKL   1               ; HARDWARE P-TABLE ADDRESS
                           0000000C 0008   174 CDB$A_VEC:       .BLKL   1               ; INTERRUPT VECTOR ADDRESS
                           00000010 000C   175 CDB$A_STORE:     .BLKL   1               ; CALLER STATUS ADDRESS
                           00000014 0010   176 CDB$A_ADDR:      .BLKL   1               ; CHANNEL VECTOR ADDRESS
                           00000018 0014   177 CDB$L_ST1:       .BLKL   1               ; STATUS (HARDWARE ERROR)
                           0000001A 0018   178 CDB$W_ST2:       .BLKW   1               ; STATUS (GENERAL)
                           0000001C 001A   179 CDB$W_RVR:       .BLKW   1               ; RECEIVE VECTOR
                           00000020 001C   180 CDB$L_PTH:       .BLKL   1               ; DATA PATH NUMBER
                           00000020 0020   181 CDBSIZ=.
                                 00000000  182 .RESTORE
                                    0000   183 CDB$BLOCK::                              ; CHANNEL CONTROL BLOCK
                           00000020 0000   184 .BLKB   CDBSIZ
                                    0020   185
                                    0020   186 ;       ARGLISTS FOR $CVTREG AND $PRINTB
                                    0020   187 FMT_CVT_ARG:    $DS_CVTREG_L    31,0,0,SHWBUF,SHW$K_BUFSIZ
00000000 00000000 0000001F 0000000B 0020                       .LONG   11, 31, 0, 0
00000000 00000000 00000080 00000068' 0030                      .LONG    SHWBUF, SHW$K_BUFSIZ, 0, 0
00000000 00000000 00000000 00000000  0040                      .LONG    0, 0, 0, 0
                                    0050   188
                                    0050   189 CVT$T_MBASR::
                           0000003D' 0050   190          .ADDRESS        MBA_SR_BIT      ; Address of Bit mnemonics
                                    0054   191                                          ; No additional arguments needed
                                    0054   192 CVT$T_MBAMAP::
                           0000003D' 0054   193          .ADDRESS        MBA_MAP_BIT     ; Address of Bit mnemonics
                                    0058   194                                          ; No additional arguments needed
                                    0058   195 CVT$T_UBADPR::
                           0000003D' 0058   196          .ADDRESS        UBA_DPR_BIT     ; Address of Bit mnemonics
                                    005C   197                                          ; No additional arguments needed
                                    005C   198 CVT$T_UBAMAP::
                           0000003E' 005C   199          .ADDRESS        UBA_MAP_BIT     ; Address of Bit mnemonics
                                    0060   200                                          ; No additional arguments needed
                                    0060   201
                                    0060   202 ; COMMON ERROR MESSAGE FOR DEBUG
                                    0060   203
                                    0060   204 MODNAM  CHANNEL                          ; DEFINE MODULE NAME FOR ERRSUP
                  4C 45 4E 4E 41 48 43 00' 0060      $MODULE:        .ASCIC  \CHANNEL\
                                 07  0060
```

```
                                00000000   206          .PSECT  Data, NoExe, Shr, NoWrt, Long          ;                    [06]
                                    0000   207
                                    0000   208 ; STORAGE FOR $SHOWCHAN - CHANNEL STATUS DUMP
                                    0000   209
                             2F 21 00'  0000   210 FMT_CRLF:        .ASCIC  .!/.
                                02      0000
4E 4E 41 48 43 20 49 42 55 2F 21 00'  0003   211 FMT_UBI_HDR:     .ASCIC  .!/UBI CHANNEL STATUS DUMP!/.
55 44 20 53 55 54 41 54 53 20 4C 45   000F
                      2F 21 50 4D     001B
                               1B     0003
5B 3A 52 53 43 5F 49 42 55 2F 21 00'  001F   212 FMT_UBI_CSR:     .ASCIC  .!/UBI_CSR:[!XL]!_!XL(X);!_!AC.
29 58 28 4C 58 21 5F 21 5D 4C 58 21   002B
            43 41 21 5F 21 3B         0037
                               1D     001F
                                    003D   213 MBA_SR_BIT:
                                    003D   214 MBA_MAP_BIT:
                                    003D   215 UBA_DPR_BIT:
                                00' 003D   216          .ASCIC  ""
                                00  003D
                                    003E   217 UBA_MAP_BIT:
2C 2C 2C 2C 2C 2C 44 49 4C 41 56 00'  003E   218          .ASCIC  "VALID,,,,,,BO,,,,,PFN=21^X"
32 3D 4E 46 50 2C 2C 2C 2C 2C 4F 42   004A
                      58 5E 31        0056
                               1A     003E
```

```
                00000068   220          .PSECT  Work, NoExe, NoShr, Wrt, Long          ;                    [06]
                    0068   221
                    0068   222 ;                                                                               [06]
                    0068   223 ; More OWN Storage                                                             [06]
                    0068   224 ;                                                                               [06]
                    0068   225
000000E9            0068   226 SHWBUF: .BLKB   SHW$K_BUFSIZ+1          ; $SHOWCHAN BUFFER
000000ED            00E9   227 JMPADR: .BLKL   1                       ; VECTOR FOR ADAPTER/DEVICE INTERRUPTS
000000F1            00ED   228 UBIVEC: .BLKL   1                       ; TEMP FOR UBI VECTOR
```

```
                        230          .SBTTL  $CHANNEL CALL
00000000                231          .PSECT  Code, Exe, Shr, NoWrt, Long             ;                    [06]
                        232 ;**
                        233 ; FUNCTIONAL DESCRIPTION:
                        234 ;
                        235 ;       CHANNEL ADAPTER INTERFACE SERVICE
                        236 ;
                        237 ; CALLING SEQUENCE:
                        238 ;
                        239 ;       CALLG   ARGLST, a*DS$CHANNEL
                        240 ;         OR
                        241 ;       CALLS   #4,     a*DS$CHANNEL
                        242 ;
                        243 ; INPUT PARAMETERS:
                        244 ;
                        245 ;       4(AP)   LOGICAL UNIT NUMBER
                        246 ;       8(AP)   FUNCTION CODE
                        247 ;       12(AP)  VECTOR ADDRESS
                        248 ;       16(AP)  STATUS STORE ADDRESS(ADDRESS OF QWORD)
                        249 ;
                        250 ; IMPLICIT INPUTS:      NONE
                        251 ;
                        252 ; OUTPUT PARAMETERS:    NONE
                        253 ;
                        254 ; IMPLICIT OUTPUTS:     NONE
                        255 ;
                        256 ; COMPLETION CODES:
                        257 ;
                        258 ;       DS$_....
                        259 ;
                        260 ; SIDE EFFECTS:
                        261 ;
                        262 ;       The channel may be affected
                        263 ;--
```

```
                    007C  0000   265 .ENTRY  DSX$CHANNEL,^M<R2,R3,R4,R5,R6>
                          0002   266
          52  04 AC      D0  0002   267       MOVL    CHN$_UNIT(AP),R2            ; GET THE UNIT NUMBER
               0301      30  0006   268       BSBW    BLDCDB                      ; FIND/CREATE THE CDB
              03 50      E8  0009   269       BLBS    R0, CHN_DSP             ; IF ERROR,  RETURN WITH                [05]
                          000C   270                                           ; ERROR CODE FROM BLDCDB                [05]
               00A1      31  000C   271       BRW     CHN_CALL_RTN                ; RETURN
                          000F   272
                          000F   273 CHN_DSP:
          52  08 AC      D0  000F   274       MOVL    CHN$_FUNC(AP),R2            ; FUNCTION CODE TO R2
                          0013   275       CASE    R2,LIMIT=#0,< -             ; DISPATCH ON FUNCTION
                          0013   276               CHN_INITA, -                ; INITA
                          0013   277               CHN_INITB, -                ; INITB
                          0013   278               CHN_ABORT, -                ; ABORT
                          0013   279               CHN_PURGE, -                ; PURGE
                          0013   280               CHN_ENINT, -                ; ENINT
                          0013   281               CHN_DSINT, -                ; DSINT
                          0013   282               CHN_CLEAR, -                ; CLEAR
                          0013   283               CHN_STATUS, -               ; STATUS
                          0013   284               CHN_SETDFT, -               ; SET DEFEAT PARITY
                          0013   285               CHN_CLRDFT, -               ; CLEAR DEFEAT PARITY
                          0013   286               >
    09'   00   52    AF    0013         CASEW   R2,#0,S^#<<30001$-30000$>/2>-1
                          0017      30000$:
                    001E' 0017               .SIGNED_WORD    CHN_INITA-30000$
                    002A' 0019               .SIGNED_WORD    CHN_INITB-30000$
                    0037' 001B               .SIGNED_WORD    CHN_ABORT-30000$
                    005F' 001D               .SIGNED_WORD    CHN_PURGE-30000$
                    0040' 001F               .SIGNED_WORD    CHN_ENINT-30000$
                    0068' 0021               .SIGNED_WORD    CHN_DSINT-30000$
                    006D' 0023               .SIGNED_WORD    CHN_CLEAR-30000$
                    0076' 0025               .SIGNED_WORD    CHN_STATUS-30000$
                    0080' 0027               .SIGNED_WORD    CHN_SETDFT-30000$
                    0089' 0029               .SIGNED_WORD    CHN_CLRDFT-30000$
                          002B      30001$:
50  00660020 8F     D0  002B   287       MOVL    #DS$_PROGERR,R0             ; INVALID FUNCTION
               007B   31  0032   288       BRW     CHN_CALL_RTN                ; RETURN
```

```
                     0035   290          .SBTTL  $CHANNEL CALL - ADAPTER INIT FUNCTION
                     0035   291 ;++
                     0035   292 ; FUNCTIONAL DESCRIPTION:
                     0035   293 ;
                     0035   294 ; INITIALIZE ADAPTER HARDWARE
                     0035   295 ;
                     0035   296 ; CALLING SEQUENCE:
                     0035   297 ;
                     0035   298 ; BRW   CHN_INITA
                     0035   299 ;
                     0035   300 ; INPUT PARAMETERS:
                     0035   301 ;
                     0035   302 ;       R5      =       CHANNEL DATA BLOCK
                     0035   303 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0035   304 ;
                     0035   305 ; IMPLICIT INPUTS:      NONE
                     0035   306 ;
                     0035   307 ; OUTPUT PARAMETERS:    NONE
                     0035   308 ;
                     0035   309 ; IMPLICIT OUTPUTS:     NONE
                     0035   310 ;
                     0035   311 ; COMPLETION CODES:
                     0035   312 ;
                     0035   313 ;       DS$_NORMAL      =       SUCCESS
                     0035   314 ;
                     0035   315 ; SIDE EFFECTS:         NONE
                     0035   316 ;--
                     0035   317
                     0035   318 CHN_INITA:
        37   01   DA   0035   319          MTPR    #1,#^X37                ; I/O subsystem clear
50  006600B1 8F   D0   0038   320          MOVL    #DS$_NOSUPPORT,R0       ; NOT SUPPORTED
             68   11   003F   321          BRB     CHN_NORMAL_RTN          ; RETURN
```

```
                     0041   323          .SBTTL  $CHANNEL CALL - BUS INIT FUNCTION
                     0041   324 ;**
                     0041   325 ; FUNCTIONAL DESCRIPTION:
                     0041   326 ;
                     0041   327 ;       NOP FOR 11/730 UNIBUS
                     0041   328 ;
                     0041   329 ; CALLING SEQUENCE:
                     0041   330 ;
                     0041   331 ;       BRW     CHN_INITB
                     0041   332 ;
                     0041   333 ; INPUT PARAMETERS:
                     0041   334 ;
                     0041   335 ;       R5      =       CHANNEL DATA BLOCK
                     0041   336 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0041   337 ;
                     0041   338 ; IMPLICIT INPUTS:      NONE
                     0041   339 ;
                     0041   340 ; OUTPUT PARAMETERS:    NONE
                     0041   341 ;
                     0041   342 ; IMPLICIT OUTPUTS:     NONE
                     0041   343 ;
                     0041   344 ; COMPLETION CODES:
                     0041   345 ;
                     0041   346 ;       DS$_NOSUPPORT   =       NOT SUPPORTED FOR 11/750 UBI OR 11/730
                     0041   347 ;
                     0041   348 ; SIDE EFFECTS: NONE
                     0041   349 ;--
                     0041   350
                     0041   351 CHN_INITB:
          37   01   DA  0041   352          MTPR    #1,#^X37                ; I/O subsystem clear
50   006600B1 8F   D0  0044   353          MOVL    #DS$_NOSUPPORT,R0       ; FUNCTION NOT SUPPORTED
          0062   31  004B   354          BRW     CHN_CALL_RTN            ; RETURN
```

```
                     004E   356          .SBTTL  $CHANNEL CALL - ABORT FUNCTION
                     004E   357 ;**
                     004E   358 ; FUNCTIONAL DESCRIPTION:
                     004E   359 ;
                     004E   360 ;       NOP FOR 11/730
                     004E   361 ;
                     004E   362 ; CALLING SEQUENCE:
                     004E   363 ;
                     004E   364 ;       BRW     CHN_ABORT
                     004E   365 ;
                     004E   366 ; INPUT PARAMETERS:
                     004E   367 ;
                     004E   368 ;       R5      =       CHANNEL DATA BLOCK
                     004E   369 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     004E   370 ;
                     004E   371 ; IMPLICIT INPUTS:      NONE
                     004E   372 ;
                     004E   373 ; OUTPUT PARAMETERS:    NONE
                     004E   374 ;
                     004E   375 ; IMPLICIT OUTPUTS:     NONE
                     004E   376 ;
                     004E   377 ; COMPLETION CODES:
                     004E   378 ;
                     004E   379 ;       DS$_NOSUPPORT   =       FUNCTION NOT SUPPORTED
                     004E   380 ;
                     004E   381 ; SIDE EFFECTS:         NONE
                     004E   382 ;--
                     004E   383
                     004E   384 CHN_ABORT:
 50  006600B1 8F  D0 004E   385          MOVL    #DS$_NOSUPPORT,R0       ; INDICATE NO SUPPORT
              59  11 0055   386          BRB     CHN_CALL_RTN            ; RETURN
```

```
                     0057   388          .SBTTL  $CHANNEL CALL - ENABLE INTERRUPTS
                     0057   389 ;**
                     0057   390 ; FUNCTIONAL DESCRIPTION:
                     0057   391 ;
                     0057   392 ;       ENABLE ADAPTER INTERRUPTS
                     0057   393 ;
                     0057   394 ; CALLING SEQUENCE:
                     0057   395 ;
                     0057   396 ;       BRB     CHN_ENINT
                     0057   397 ;
                     0057   398 ; INPUT PARAMETERS:
                     0057   399 ;
                     0057   400 ;       R5      =       CHANNEL DATA BLOCK
                     0057   401 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0057   402 ;
                     0057   403 ; IMPLICIT INPUTS:      NONE
                     0057   404 ;
                     0057   405 ; OUTPUT PARAMETERS:    NONE
                     0057   406 ;
                     0057   407 ; IMPLICIT OUTPUTS:     NONE
                     0057   408 ;
                     0057   409 ; COMPLETION CODES:
                     0057   410 ;
                     0057   411 ;       DS$_NORMAL      =       SUCCESS
                     0057   412 ;       DS$_IVVECT      =       INVALID VECTOR ($SETVEC)
                     0057   413 ;       DS$_IVADDR      =       INVALID ADDRESS ($SETVEC)
                     0057   414 ;
                     0057   415 ; SIDE EFFECTS:
                     0057   416 ;
                     0057   417 ;       CALLS $SETVEC TO POINT ADAPTER/UUT INTERRUPTS INTO CHANNEL SERVICE
                     0057   418 ;
                     0057   419 ;--
                     0057   420
                     0057   421 CHN_ENINT:
                02EE   30    0057   422          BSBW    SETVEC                  ; SET THE VECTOR'S
               17 50   E9    005A   423          BLBC    R0,40$                  ; IF ERROR RETURN WITH IT
                      005D   424
    50   00660020 8F   D0    005D   425          MOVL    #DS$_PROGERR,R0         ; ASSUME ADDRESS CHECK FAILED
                      0064   426
 08 A5   0C AC   D0    0064   427          MOVL    CHN$_ADR1(AP),CDB$A_VEC(R5)     ; SAVE CALLERS VECTOR
                09   15    0069   428          BLEQ    40$                     ; ZERO OR NEGATIVE IS NOGOOD
 0C A5   10 AC   D0    006B   429          MOVL    CHN$_ADR2(AP),CDB$A_STORE(R5)   ; SAVE CALLERS STORE ADDRESS
                02   15    0070   430          BLEQ    40$                     ; GTR ZERO IS CORRECT
                50   D6    0072   431          INCL    R0                      ; Flag success
                      0074   432
                3A   11    0074   433 40$:     BRB     CHN_CALL_RTN            ; RETURN
```

```
                        0076   435          .SBTTL  $CHANNEL CALL - PURGE
                        0076   436 ;++
                        0076   437 ; FUNCTIONAL DESCRIPTION:
                        0076   438 ;
                        0076   439 ;       NOP FOR 11/730
                        0076   440 ;
                        0076   441 ; CALLING SEQUENCE:
                        0076   442 ;
                        0076   443 ;       BRW     CHN_PURGE
                        0076   444 ;
                        0076   445 ; INPUT PARAMETERS:
                        0076   446 ;
                        0076   447 ;       R5      =       CHANNEL DATA BLOCK
                        0076   448 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                        0076   449 ;
                        0076   450 ; IMPLICIT INPUTS:      NONE
                        0076   451 ;
                        0076   452 ; OUTPUT PARAMETERS:    NONE
                        0076   453 ;
                        0076   454 ; IMPLICIT OUTPUTS:     NONE
                        0076   455 ;
                        0076   456 ; COMPLETION CODES:
                        0076   457 ;
                        0076   458 ;       DS$_NOSUPPORT   = not supported
                        0076   459 ;
                        0076   460 ; SIDE EFFECTS:         NONE
                        0076   461 ;--
                        0076   462
                        0076   463 CHN_PURGE:
 50   006600B1 8F    D0  0076   464           MOVL    #DS$_NOSUPPORT,R0       ; Not supported
                  31    11  007D   465           BRB     CHN_CALL_RTN            ; Return
```

```
                     007F   467          .SBTTL  $CHANNEL CALL - DISABLE INTERRUPTS
                     007F   468 ;**
                     007F   469 ; FUNCTIONAL DESCRIPTION:
                     007F   470 ;
                     007F   471 ;       DISABLE ADAPTER INTERRUPT PROCESSING
                     007F   472 ;
                     007F   473 ; CALLING SEQUENCE:
                     007F   474 ;
                     007F   475 ;       BRW     CHN_DSINT
                     007F   476 ;
                     007F   477 ; INPUT PARAMETERS:
                     007F   478 ;
                     007F   479 ;       R5      =       CHANNEL DATA BLOCK
                     007F   480 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     007F   481 ;
                     007F   482 ; IMPLICIT INPUTS:      NONE
                     007F   483 ;
                     007F   484 ; OUTPUT PARAMETERS:    NONE
                     007F   485 ;
                     007F   486 ; IMPLICIT OUTPUTS:     NONE
                     007F   487 ;
                     007F   488 ; COMPLETION CODES:
                     007F   489 ;
                     007F   490 ;       DS$_NORMAL      =       SUCCESS
                     007F   491 ;       DS$_IVVECT      =       INVALID VECTOR ($CLRVEC)
                     007F   492 ;
                     007F   493 ; SIDE EFFECTS:
                     007F   494 ;
                     007F   495 ;       CALLS $CLRVEC
                     007F   496 ;
                     007F   497 ;--
                     007F   498
                     007F   499 CHN_DSINT:
02F7   30   007F   500          BSBW    CLRVEC                  ; CLEAR VECTOR'S
  2C   11   0082   501          BRB     CHN_CALL_RTN            ; RETURN WITH STATUS IN R0
```

```
                     0084    503          .SBTTL  $CHANNEL CALL - CLEAR ADAPTER
                     0084    504 ;**
                     0084    505 ; FUNCTIONAL DESCRIPTION:
                     0084    506 ;
                     0084    507 ;       CLEAR ADAPTER STATUS
                     0084    508 ;
                     0084    509 ; CALLING SEQUENCE:
                     0084    510 ;
                     0084    511 ;       BRW     CHN_CLEAR
                     0084    512 ;
                     0084    513 ; INPUT PARAMETERS:
                     0084    514 ;
                     0084    515 ;       R5      =       CHANNEL DATA BLOCK
                     0084    516 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0084    517 ;
                     0084    518 ; IMPLICIT INPUTS:      NONE
                     0084    519 ;
                     0084    520 ; OUTPUT PARAMETERS:    NONE
                     0084    521 ;
                     0084    522 ; IMPLICIT OUTPUTS:     NONE
                     0084    523 ;
                     0084    524 ; COMPLETION CODES:
                     0084    525 ;
                     0084    526 ;       DS$_NOSUPPORT   = Function NOPED
                     0084    527 ;
                     0084    528 ; SIDE EFFECTS:         NONE
                     0084    529 ;--
                     0084    530
                     0084    531 CHN_CLEAR:
50   006600B1 8F     D0      0084    532             MOVL    #DS$_NOSUPPORT,R0       ; Not supported
                 23  11      008B    533             BRB     CHN_CALL_RTN
```

```
                     008D   535          .SBTTL  $CHANNEL CALL - GET ADAPTER STATUS
                     008D   536 ;**
                     008D   537 ; FUNCTIONAL DESCRIPTION:
                     008D   538 ;
                     008D   539 ;       PASS ADAPTER STATUS TO CALLER
                     008D   540 ;
                     008D   541 ; CALLING SEQUENCE:
                     008D   542 ;
                     008D   543 ;       BRW     CHN_STATUS
                     008D   544 ;
                     008D   545 ; INPUT PARAMETERS:
                     008D   546 ;
                     008D   547 ;       R5      =       CHANNEL DATA BLOCK
                     008D   548 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     008D   549 ;
                     008D   550 ; IMPLICIT INPUTS:
                     008D   551 ;
                     008D   552 ;       NONE
                     008D   553 ;
                     008D   554 ; OUTPUT PARAMETERS:
                     008D   555 ;
                     008D   556 ;       R0      =       COMPLETION CODE
                     008D   557 ;
                     008D   558 ; IMPLICIT OUTPUTS:
                     008D   559 ;
                     008D   560 ;       NONE
                     008D   561 ;
                     008D   562 ; COMPLETION CODES:
                     008D   563 ;
                     008D   564 ;       DS$_NORMAL      =       SUCCESS
                     008D   565 ;
                     008D   566 ; SIDE EFFECTS:
                     008D   567 ;
                     008D   568 ;       NONE
                     008D   569 ;
                     008D   570 ;--
                     008D   571
                     008D   572 CHN_STATUS:
          14 A5   7C 008D   573          CLRQ    CDB$L_ST1(R5)                   ; Clear all status
10 BC     14 A5   7D 0090   574          MOVQ    CDB$L_ST1(R5),aCHN$_ADR2(AP)    ; PASS STATUS
             12   11 0095   575          BRB     CHN_NORMAL_RTN                  ; RETURN
```

```
                     0097   577          .SBTTL  $CHANNEL CALL - SET DEFEAT PARITY
                     0097   578 ;++
                     0097   579 ; FUNCTIONAL DESCRIPTION:
                     0097   580 ;
                     0097   581 ;       SET DEFEAT DATA PATH PARITY
                     0097   582 ;
                     0097   583 ; CALLING SEQUENCE:
                     0097   584 ;
                     0097   585 ;       BSBW    CHN_SETDFT
                     0097   586 ;
                     0097   587 ; INPUT PARAMETERS:
                     0097   588 ;
                     0097   589 ;       R5      =       CHANNEL DATA BLOCK
                     0097   590 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0097   591 ;
                     0097   592 ; IMPLICIT INPUTS:
                     0097   593 ;
                     0097   594 ;       NONE
                     0097   595 ;
                     0097   596 ; OUTPUT PARAMETERS:
                     0097   597 ;
                     0097   598 ;       R0      =       COMPLETION CODE
                     0097   599 ;
                     0097   600 ; IMPLICIT OUTPUTS:
                     0097   601 ;
                     0097   602 ;       NONE
                     0097   603 ;
                     0097   604 ; COMPLETION CODES:
                     0097   605 ;
                     0097   606 ;       DS$_NORMAL      =       SUCCESS
                     0097   607 ;       DS$_NOSUPPORT   =       FUNCTION NOT SUPPORTED FOR 11/750 UBI
                     0097   608 ;
                     0097   609 ; SIDE EFFECTS:
                     0097   610 ;
                     0097   611 ;       NONE
                     0097   612 ;
                     0097   613 ;--
                     0097   614
                     0097   615 CHN_SETDFT:
50   006600B1 8F     D0    0097   616          MOVL    #DS$_NOSUPPORT,R0       ; INDICATE NO SUPPORT
                 10  11    009E   617          BRB     CHN_CALL_RTN            ; RETURN
```

```
                     00A0    619          .SBTTL  $CHANNEL CALL - CLEAR DEFEAT PARITY
                     00A0    620 ;++
                     00A0    621 ; FUNCTIONAL DESCRIPTION:
                     00A0    622 ;
                     00A0    623 ;       CLEAR DEFEAT DATA PATH PARITY (UBI ONLY)
                     00A0    624 ;
                     00A0    625 ; CALLING SEQUENCE:
                     00A0    626 ;
                     00A0    627 ;       BRW     CHN_CLRDFT
                     00A0    628 ;
                     00A0    629 ; INPUT PARAMETERS:
                     00A0    630 ;
                     00A0    631 ;       R5      =       CHANNEL DATA BLOCK
                     00A0    632 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     00A0    633 ;
                     00A0    634 ; IMPLICIT INPUTS:
                     00A0    635 ;
                     00A0    636 ;       NONE
                     00A0    637 ;
                     00A0    638 ; OUTPUT PARAMETERS:
                     00A0    639 ;
                     00A0    640 ;       R0      =       COMPLETION CODE
                     00A0    641 ;
                     00A0    642 ; IMPLICIT OUTPUTS:
                     00A0    643 ;
                     00A0    644 ;       NONE
                     00A0    645 ;
                     00A0    646 ; COMPLETION CODES:
                     00A0    647 ;
                     00A0    648 ;       DS$_NORMAL      =       SUCCESS
                     00A0    649 ;       DS$_NOSUPPORT   =       NOT SUPPORTED (11/750 UBI)
                     00A0    650 ;
                     00A0    651 ; SIDE EFFECTS:
                     00A0    652 ;
                     00A0    653 ;       NONE
                     00A0    654 ;
                     00A0    655 ;--
                     00A0    656
                     00A0    657 CHN_CLRDFT:
 50  006600B1 8F     D0  00A0    658          MOVL    #DS$_NOSUPPORT,R0               ; NOT SUPPORTED
              07     11  00A7    659          BRB     CHN_CALL_RTN                    ; RETURN
```

```
                     00A9   661          .SBTTL  $CHANNEL CALL - COMMON CALL RETURN
                     00A9   662 ;++
                     00A9   663 ; FUNCTIONAL DESCRIPTION:
                     00A9   664 ;
                     00A9   665 ;       COMMON CALL RETURN
                     00A9   666 ;
                     00A9   667 ; CALLING SEQUENCE:
                     00A9   668 ;
                     00A9   669 ;       BRW     CHN_CALL_RTN
                     00A9   670 ;
                     00A9   671 ; INPUT PARAMETERS:
                     00A9   672 ;
                     00A9   673 ;       R0      =       COMPLETION CODE
                     00A9   674 ;       R5      =       CHANNEL DATA BLOCK
                     00A9   675 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     00A9   676 ;
                     00A9   677 ; IMPLICIT INPUTS:
                     00A9   678 ;
                     00A9   679 ;       NONE
                     00A9   680 ;
                     00A9   681 ; OUTPUT PARAMETERS:
                     00A9   682 ;
                     00A9   683 ;       R0      =       COMPLETION CODE
                     00A9   684 ;
                     00A9   685 ; IMPLICIT OUTPUTS:
                     00A9   686 ;
                     00A9   687 ;       NONE
                     00A9   688 ;
                     00A9   689 ; COMPLETION CODES:
                     00A9   690 ;
                     00A9   691 ;       NONE
                     00A9   692 ;
                     00A9   693 ; SIDE EFFECTS:
                     00A9   694 ;
                     00A9   695 ;       NONE
                     00A9   696 ;
                     00A9   697 ;--
                     00A9   698
                     00A9   699 CHN_NORMAL_RTN:
50   00660001 8F     D0    00A9   700          MOVL    #DS$_NORMAL,R0                  ; FLAG NORMAL RETURN
                     00B0   701
                     00B0   702 CHN_CALL_RTN:
                        04   00B0   703          RET                                     ; RETURN
```

```
00B1    705          .SBTTL  $SETMAP CALL
00B1    706 ;**
00B1    707 ; FUNCTIONAL DESCRIPTION:
00B1    708 ;
00B1    709 ;       SET ADPATER MAPPING
00B1    710 ;
00B1    711 ; CALLING SEQUENCE:
00B1    712 ;
00B1    713 ;       CALLG   ARGLST, a#DSX$SETMAP
00B1    714 ;         OR
00B1    715 ;       CALLS   #6,     a#DSX$SETMAP
00B1    716 ;
00B1    717 ; INPUT PARAMETERS:
00B1    718 ;
00B1    719 ;       4(AP)   LOGICAL UNIT NUMBER
00B1    720 ;       8(AP)   FUNCTION CODE
00B1    721 ;       12(AP)  PHYSICAL ADDRESS(POINTER TO QWORD)
00B1    722 ;       16(AP)  BASE ADDRESS(TO START MAPPING)
00B1    723 ;       20(AP)  BYTE COUNT
00B1    724 ;       24(AP)  DATA PATH NUMBER
00B1    725 ;
00B1    726 ; IMPLICIT INPUTS:
00B1    727 ;
00B1    728 ;       NONE
00B1    729 ;
00B1    730 ; OUTPUT PARAMETERS:
00B1    731 ;
00B1    732 ;       R0      =       COMPLETION CODE
00B1    733 ;
00B1    734 ; IMPLICIT OUTPUTS:
00B1    735 ;
00B1    736 ;       NONE
00B1    737 ;
00B1    738 ; COMPLETION CODES:
00B1    739 ;
00B1    740 ;       DS$_NORMAL      =       SUCCESS
00B1    741 ;       DS$_PROGERR     =       PROGRAM ERROR
00B1    742 ;                                MISSING/INVALID P-TABLE
00B1    743 ;                                ADDRESS/BYTE COUNT INVALID
00B1    744 ;                                INVALID DATA PATH
00B1    745 ;       DS$_IHWE        =       INITIAL ADAPTER HARDWARE ERROR ENCOUNTERED
00B1    746 ;
00B1    747 ; SIDE EFFECTS:
00B1    748 ;
00B1    749 ;       NONE
00B1    750 ;
00B1    751 ;--
```

```
                    0FFC  00B1   753 .ENTRY  DSX$SETMAP,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                          00B3   754
          52  04 AC     D0  00B3   755        MOVL    STM$_UNIT(AP),R2                ; LOGICAL UNIT TO R2
            0250        30  00B7   756        BSBW    BLDCDB                          ; BUILD THE CDB
           03 50        E8  00BA   757        BLBS    R0, 10$                 ; ERROR ?                       [05]
            01C0        31  00BD   758        BRW     STM_CALL_RTN                    ; YES - RETURN
                          00C0   759
          52  0C AC     D0  00C0   760 10$:    MOVL    STM$_PHY(AP),R2                 ; GET THE PHYSICAL ADDRESS
            0184        30  00C4   761        BSBW    RNGCHK                          ; RANGE CHECK
           03 50        E8  00C7   762        BLBS    R0, 20$                 ; RANGE CHECK ERROR ?           [05]
            01B3        31  00CA   763        BRW     STM_CALL_RTN                    ; YES-
                          00CD   764
        1C A5  18 AC     D0  00CD   765 20$:    MOVL    STM$_PTH(AP),CDB$L_PTH(R5)      ; REMEMBER THE DATA PATH
          52  10 AC     D0  00D2   766        MOVL    STM$_BAS(AP),R2                 ; YES- BASE ADDRESS TO R2
          53  0C BC     D0  00D6   767        MOVL    @STM$_PHY(AP),R3                ; START PHYSICAL ADDRESS TO R3
          5A  14 AC     D0  00DA   768        MOVL    STM$_CNT(AP),R10                ; BYTE COUNT TO R10
    50  53    17  09     EF  00DE   769        EXTZV   #9,#23,R3,R0                    ; Get PFN of start page
    58  01FF CA43     9E  00E3   770        MOVAB   511(R10)[R3],R8                 ; Last byte R10+R3-1 round to page
    58  58    17  09     EF  00E9   771        EXTZV   #9,#23,R8,R8                    ; Get PFN of last page
          58  50     C2  00EE   772        SUBL    R0,R8                           ; End-start = maps to alloc
            013B        30  00F1   773        BSBW    BASCHK                          ; CHECK BASE
           03 50        E8  00F4   774        BLBS    R0,60$                          ; IF ERROR RETURN WITH
            0186        31  00F7   775        BRW     STM_CALL_RTN                    ; BASCHK COMPLETION CODE
                          00FA   776 60$:
            0152        30  00FA   777        BSBW    CNTCHK                          ; CHECK COUNT
           03 50        E8  00FD   778        BLBS    R0,70$                          ; IF ERROR RETURN WITH
            017D        31  0100   779        BRW     STM_CALL_RTN                    ; CNTCHK COMPLETION CODE
                          0103   780 70$:
          50  08 AC     D0  0103   781        MOVL    STM$_FUNC(AP),R0                ; GET THE FUNCTION CODE
                          0107   782        CASE    R0,LIMIT=#0,TYPE=L,<-           ;
                          0107   783        NFWDN,-                                 ; NOMAP-FORWARD-NOVALIDATE
                          0107   784        INVALIDATE,-                            ; Invalidate all registers      [08]
                          0107   785        MFWDN,-                                 ; MAP-FORWARD-NOVALIDATE
                          0107   786        MFWDV,-                                 ; MAP-FORWARD-VALIDATE
                          0107   787        NREVN,-                                 ; NOMAP-REVERSE-NOVALIDATE
                          0107   788        INVALID,-                               ; INVALID
                          0107   789        MREVN,-                                 ; MAP-REVERSE-NOVALIDATE
                          0107   790        MREVV,-                                 ; MAP-REVERSE-VALIDATE
                          0107   791        INVALID,-                               ; INVALID
                          0107   792        INVALID,-                               ; INVALID
                          0107   793        MFWDNO,-                                ; MAP-FORWARD-NOVALIDATE-OFFSET
                          0107   794        MFWDVO,-                                ; MAP-FORWARD-VALIDATE-OFFSET
                          0107   795        INVALID,-                               ; INVALID
                          0107   796        INVALID,-                               ; INVALID
                          0107   797        MREVNO,-                                ; MAP-REVERSE-NOVALIDATE-OFFSET
                          0107   798        MREVVO,-                                ; MAP-REVERSE-VALIDATE-OFFSET
                          0107   799        >
0F'   00  50    CF        0107              CASEL   R0,#0,S^#<<30003$-30002$>/2>-1
                          010B         30002$:
                0054'     010B              .SIGNED_WORD    NFWDN-30002$
                002A'     010D              .SIGNED_WORD    INVALIDATE-30002$
                0045'     010F              .SIGNED_WORD    MFWDN-30002$
                0033'     0111              .SIGNED_WORD    MFWDV-30002$
                0087'     0113              .SIGNED_WORD    NREVN-30002$
                0020'     0115              .SIGNED_WORD    INVALID-30002$
                0075'     0117              .SIGNED_WORD    MREVN-30002$
                0060'     0119              .SIGNED_WORD    MREVV-30002$
```

```
                    0020' 011B                .SIGNED_WORD    INVALID-30002$
                    0020' 011D                .SIGNED_WORD    INVALID-30002$
                    00AB' 011F                .SIGNED_WORD    MFWDNO-30002$
                    0096' 0121                .SIGNED_WORD    MFWDVO-30002$
                    0020' 0123                .SIGNED_WORD    INVALID-30002$
                    0020' 0125                .SIGNED_WORD    INVALID-30002$
                    00CF' 0127                .SIGNED_WORD    MREVNO-30002$
                    00BD' 0129                .SIGNED_WORD    MREVVO-30002$
                          012B        30003$:
                          012B    800 INVALID:
50   00660020 8F    D0    012B    801         MOVL    #DS$_PROGERR,R0                 ; INVALID FUNCTION CODE
                 014B    31    0132    802         BRW     STM_CALL_RTN                    ; RETURN
                          0135    803 INVALIDATE:
              65   18    CA    0135    804         BICL    #CDB$M_OFFSET ! CDB$M_REV, -    ;                                [08]
                          0138    805                  CDB$L_FLAGS(R5)                ;                                [08]
                 00E2    30    0138    806         BSBW    CLRMAP                          ; Clear all map regs             [08]
                 013B    31    013B    807         BRW     STM_NORMAL_RTN                  ; Return OK                      [08]
                          013E    808 MFWDV:
                   18    CA    013E    809         BICL    #CDB$M_OFFSET!CDB$M_REV,-
                   65          0140    810                  CDB$L_FLAGS(R5)                ; CLEAR OFFSET AND REVERSE FLAGS
                 00D9    30    0141    811         BSBW    CLRMAP                          ; CLEAR ALL MAPS
                 00A2    30    0144    812         BSBW    SETMAP                          ; SET MAPS FOR FORWARD TRANSFER
                 00E3    30    0147    813         BSBW    SETVAR                          ; SET VAR IF MBA
                 00E1    30    014A    814         BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                 0129    31    014D    815         BRW     STM_NORMAL_RTN                  ; RETURN
                          0150    816
                          0150    817 MFWDN:
                   18    CA    0150    818         BICL    #CDB$M_OFFSET!CDB$M_REV,-
                   65          0152    819                  CDB$L_FLAGS(R5)                ; CLEAR OFFSET AND REVERSE FLAGS
                 0093    30    0153    820         BSBW    SETMAP                          ; SET MAPS FOR FORWARD TRANSFER
                 00D4    30    0156    821         BSBW    SETVAR                          ; SET VAR IF MBA
                 00D2    30    0159    822         BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                 011A    31    015C    823         BRW     STM_NORMAL_RTN                  ; RETURN
                          015F    824
                          015F    825 NFWDN:
                   18    CA    015F    826         BICL    #CDB$M_OFFSET!CDB$M_REV,-
                   65          0161    827                  CDB$L_FLAGS(R5)                ; CLEAR OFFSET AND REVERSE FLAGS
                 00C8    30    0162    828         BSBW    SETVAR                          ; SET VAR IF MBA
                 00C6    30    0165    829         BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                 010E    31    0168    830         BRW     STM_NORMAL_RTN                  ; RETURN
                          016B    831
                          016B    832 MREVV:
              65   08    CA    016B    833         BICL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)   ; CLEAR OFFSET FLAG
              65   10    C8    016E    834         BISL    #CDB$M_REV,CDB$L_FLAGS(R5)      ; SET REVERSE FLAG
                 00A9    30    0171    835         BSBW    CLRMAP                          ; CLEAR ALL MAP ENTRIES
                 0072    30    0174    836         BSBW    SETMAP                          ; SET MAPS FOR REVERSE OPERATION
                 00B3    30    0177    837         BSBW    SETVAR                          ; SET VAR IF MBA
                 00B1    30    017A    838         BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                 00F9    31    017D    839         BRW     STM_NORMAL_RTN                  ; RETURN
                          0180    840
                          0180    841 MREVN:
              65   08    CA    0180    842         BICL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)   ; CLEAR OFFSET FLAG
              65   10    C8    0183    843         BISL    #CDB$M_REV,CDB$L_FLAGS(R5)      ; SET REVERSE FLAG
                 0060    30    0186    844         BSBW    SETMAP                          ; SET MAPPING
                 00A1    30    0189    845         BSBW    SETVAR                          ; SET VAR IF MBA
                 009F    30    018C    846         BSBW    SETCNT                          ; SET BYTE COUNT IF MBA
                 00E7    31    018F    847         BRW     STM_NORMAL_RTN                  ; RETURN
```

```
                     0192   848
                     0192   849 NREVN:
65   08   CA         0192   850          BICL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)    ; CLEAR OFFSET FLAG
65   10   C8         0195   851          BISL    #CDB$M_REV,CDB$L_FLAGS(R5)       ; SET REVERSE FLAG
   0092   30         0198   852          BSBW    SETVAR                           ; SET VAR IF MBA
   0090   30         019B   853          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
   00D8   31         019E   854          BRW     STM_NORMAL_RTN                   ; RETURN
                     01A1   855
                     01A1   856 MFWDVO:
65   10   CA         01A1   857          BICL    #CDB$M_REV,CDB$L_FLAGS(R5)       ; CLEAR REVERSE FLAG
65   08   C8         01A4   858          BISL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)    ; SET OFFSET FLAG
   0073   30         01A7   859          BSBW    CLRMAP                           ; INVALIDATE MAPS
   003C   30         01AA   860          BSBW    SETMAP                           ; SET MAPS
   007D   30         01AD   861          BSBW    SETVAR                           ; SET VAR IF MBA
   007B   30         01B0   862          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
   00C3   31         01B3   863          BRW     STM_NORMAL_RTN                   ; RETURN
                     01B6   864
                     01B6   865 MFWDNO:
65   10   CA         01B6   866          BICL    #CDB$M_REV,CDB$L_FLAGS(R5)       ; CLEAR REVERSE FLAG
65   08   C8         01B9   867          BISL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)    ; SET OFFSET FLAG
   002A   30         01BC   868          BSBW    SETMAP                           ; SET MAPPING
   006B   30         01BF   869          BSBW    SETVAR                           ; SET VAR IF MBA
   0069   30         01C2   870          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
   00B1   31         01C5   871          BRW     STM_NORMAL_RTN                   ; RETURN
                     01C8   872
                     01C8   873 MREVVO:
     18   C8         01C8   874          BISL    #CDB$M_OFFSET!CDB$M_REV,-
     65              01CA   875                  CDB$L_FLAGS(R5)                  ; SET OFFSET AND REVERSE FLAGS
   004F   30         01CB   876          BSBW    CLRMAP                           ; CLEAR MAPS
   0018   30         01CE   877          BSBW    SETMAP                           ; SET MAPPING
   0059   30         01D1   878          BSBW    SETVAR                           ; SET VAR IF MBA
   0057   30         01D4   879          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
   009F   31         01D7   880          BRW     STM_NORMAL_RTN                   ; RETURN
                     01DA   881
                     01DA   882 MREVNO:
     18   C8         01DA   883          BISL    #CDB$M_OFFSET!CDB$M_REV,-
     65              01DC   884                  CDB$L_FLAGS(R5)                  ; SET OFFSET AND REVERSE FLAGS
   0009   30         01DD   885          BSBW    SETMAP                           ; SET MAPPING
   004A   30         01E0   886          BSBW    SETVAR                           ; SET VAR IF MBA
   0048   30         01E3   887          BSBW    SETCNT                           ; SET BYTE COUNT IF MBA
   0090   31         01E6   888          BRW     STM_NORMAL_RTN                   ; RETURN
                     01E9   889
```

```
                        01E9    891 ;+
                        01E9    892 ; ROUTINE TO SET ADAPTER MAP REGISTERS
                        01E9    893 ; CALLED BY: BSBW SETMAP
                        01E9    894 ; R2 = BASE ADDRESS TO START MAPPING
                        01E9    895 ; R3 = START PHYSICAL ADDRESS
                        01E9    896 ; R6 = ADAPTER CSR ADDRESS
                        01E9    897 ; R8 = NUMBER OF PAGES TO MAP
                        01E9    898 ;-
                        01E9    899 SETMAP:
          51    52  D0  01E9    900          MOVL    R2,R1                           ; BASE TO R1
                08  BB  01EC    901          PUSHR   #^M<R3>                         ; SAVE R3
 53   53    F7  8F  78  01EE    902          ASHL    #-9,R3,R3                       ; SHIFT PFN
                50  D4  01F3    903          CLRL    R0                              ; INIT R0
 50   80000000  8F  C8  01F5    904          BISL    #UBI$M_MAP_VALID,R0             ; SET THE VALID BIT
          65    08  D3  01FC    905          BITL    #CDB$M_OFFSET,CDB$L_FLAGS(R5)   ; OFFSET MODE ?
                07  13  01FF    906          BEQL    10$                             ; NO -
 50   02000000  8F  C8  0201    907          BISL    #UBI$M_MAP_BO,R0                ; YES- SET THE BIT
                    0208    908 10$:
          00    53  F0  0208    909          INSV    R3,#UBA$V_MAP_ADDR,-            ; INSERT THE PFN
          50    15          020B    910                  #UBA$S_MAP_ADDR,R0          ; INTO R0
   0800 C641    50  D0  020D    911          MOVL    R0,UBI$L_MAP(R6)[R1]            ; CREATE THE MAP
                53  D6  0213    912          INCL    R3                              ; INC THE ENTRY
                51  D6  0215    913          INCL    R1                              ; INC THE POINTER
             EE 58  F5  0217    914          SOBGTR  R8,10$                          ; DO ALL PAGES
                08  BA  021A    915          POPR    #^M<R3>                         ; RESTORE R3
                    021C    916
                05  021C    917          RSB                                     ; RETURN
                    021D    918
```

```
                                021D   ;+
                                021D   ; Routine to invalidate all adapter map entries
                                021D   ; Called by: Bsbw CLRMAP
                                021D   ; R6 = Adapter CSR address
                                021D   ;-
                                021D
                                021D   CLRMAP:
                     50     D4  021D           CLRL    R0                          ; Clear R0
              0800 C640     D4  021F   10$:    CLRL    UBI$L_MAP(R6)[R0]           ; Clear the map entry
      F3 50   000001FF 8F   F3  0224           AOBLEQ  #511, R0, 10$               ; Clear all 512 entries        [10]
                            05  022C           RSB                                 ; Return
                                022D
                                022D
                                022D   ;+
                                022D   ; ROUTINE TO SET ADAPTER VIRTUAL ADDRESS REGISTER
                                022D   ; CALLED BY BSBW SETVAR
                                022D   ; R2 = BASE ADDRESS
                                022D   ; R3 = START PHYSICAL ADDRESS
                                022D   ; R6 = ADAPTER CSR ADDRESS
                                022D   ; R10 = BYTE COUNT
                                022D   ;-
                                022D   SETVAR:
                            05  022D           RSB                                 ; RETURN
                                022E
                                022E
                                022E   ;+
                                022E   ; ROUTINE TO SET THE ADAPTER BYTE COUNT REGISTER
                                022E   ; CALLED BY: BSBW SETCNT
                                022E   ; R6 = ADAPTER CSR ADDRESS
                                022E   ; R10 = BYTE COUNT
                                022E   ;-
                                022E   SETCNT:
                            05  022E           RSB                                 ; RETURN
                                022F
                                022F   ;+
                                022F   ; ROUTINE TO VERIFY THAT THE BASE ADDRESS
                                022F   ; PLUS THE NUMBER OF PAGES REQUIRED WILL
                                022F   ; FIT WITHIN THE MAPPING SPACE
                                022F   ; R2 = MAP BASE
                                022F   ; R8 = TOTAL PAGES TO MAP
                                022F   ;-
                                022F   BASCHK:
      50   00660020 8F      D0  022F           MOVL    #DS$_PROGERR,R0             ; NO - ERROR
         51   52   58      C1  0236           ADDL3   R8,R2,R1                    ; BASE + TOTAL PAGES
      00000200 8F   51      D1  023A           CMPL    R1, #512                    ; Is top register in range?    [10]
                    07      14  0241           BGTR    10$                         ; YES-
      50   00660001 8F      D0  0243           MOVL    #DS$_NORMAL,R0              ; INDICATE SUCCESS
                                024A
                            05  024A   10$:    RSB                                 ; RETURN
                                024B
                                024B   ;+
                                024B   ; ROUTINE TO VALIDATE TRANSFER ADDRESS
                                024B   ; NOT IMPLEMENTED
                                024B   ;-
                                024B   RNGCHK:
              50   01       D0  024B           MOVL    #1,R0                  ; Success
                            05  024E           RSB                                 ; RETURN
```

```
                                024F   977
                                024F   978 ;+
                                024F   979 ; ROUTINE TO VERIFY THAT REQUESTED BYTE COUNT
                                024F   980 ; WILL FIT IN REQUESTED BUFFER SIZE
                                024F   981 ;-
                                024F   982 CNTCHK:
                          0E    BB     024F   983          PUSHR   #^M<R1,R2,R3>                   ; SAVE R1 - R3
                    52    0C  BC     7D  0251   984          MOVQ    @STM$_PHY(AP),R2                ; BEGIN ADDR-> R2, END -> R3
                  53    01FF  8F     A8  0255   985          BISW2   #^X1FF,R3                       ; FORCE TO END OF PAGE
                        53    52     C2  025A   986          SUBL2   R2,R3                           ; FIND NUMBER OF BYTES
                              53     D6  025D   987          INCL    R3                              ; PLUS ONE
                    51    14  AC     D0  025F   988          MOVL    STM$_CNT(AP),R1                 ; COUNT TO R1
              50  00660001  8F     D0  0263   989          MOVL    #DS$_NORMAL,R0                  ; INITIAL COMPLETION CODE
                        53    51     D1  026A   990          CMPL    R1,R3                           ; WILL BYTE COUNT FIT ?
                              07     15  026D   991          BLEQ    CNTCHK_RTN                      ; YES-
              50  00660020  8F     D0  026F   992          MOVL    #DS$_PROGERR,R0                 ; NO -
                                     0276   993
                                     0276   994 CNTCHK_RTN:
                          0E    BA     0276   995          POPR    #^M<R1,R2,R3>                   ; RESTORE REGISTERS
                                05   0278   996          RSB                                     ; RETURN
                                     0279   997
                                     0279   998 ;+
                                     0279   999 ;
                                     0279  1000 ; SETMAP  "GLOBAL" EXIT
                                     0279  1001 ;
                                     0279  1002 ;-
                                     0279  1003
                                     0279  1004 STM_NORMAL_RTN:
              50  00660001  8F     D0  0279  1005          MOVL    #DS$_NORMAL,R0                  ; IT WORKS
                                     0280  1006
                                     0280  1007 STM_CALL_RTN:
                                04   0280  1008          RET                                     ; RETURN TO CALLER
```

```
                     0281  1010          .SBTTL  DSP$SHOWMBA
                     0281  1011 ;++
                     0281  1012 ; FUNCTIONAL DESCRIPTION:
                     0281  1013 ;
                     0281  1014 ;       THIS ROUTINE WILL PRINT OUT THE CURRENT STATE OF OF THE MBA
                     0281  1015 ;       AS SPECIFIED BY THE ARGLIST
                     0281  1016 ;
                     0281  1017 ; CALLING SEQUENCE:
                     0281  1018 ;
                     0281  1019 ;       CALLS   #1,DSP$SHOWMBA OR
                     0281  1020 ;       CALLG   ARGLIST,DSP$SHOWMBA
                     0281  1021 ;
                     0281  1022 ; INPUT PARAMETERS:
                     0281  1023 ;
                     0281  1024 ;       4(AP)   -> ADDRESS OF MBA CSR
                     0281  1025 ;
                     0281  1026 ; IMPLICIT INPUTS:
                     0281  1027 ;
                     0281  1028 ;       NONE
                     0281  1029 ;
                     0281  1030 ; OUTPUT PARAMETERS:
                     0281  1031 ;
                     0281  1032 ;       NONE
                     0281  1033 ;
                     0281  1034 ; IMPLICIT OUTPUTS:
                     0281  1035 ;
                     0281  1036 ;       NONE
                     0281  1037 ;
                     0281  1038 ; COMPLETION CODES:
                     0281  1039 ;
                     0281  1040 ;       DS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                     0281  1041 ;
                     0281  1042 ; SIDE EFFECTS:
                     0281  1043 ;
                     0281  1044 ;       NONE
                     0281  1045 ;--
                     0281  1046
                 007C  0281  1047 .ENTRY  DSP$SHOWMBA,^M<R2,R3,R4,R5,R6>          ; ENTRY MASK (SAME AS SHOWCHAN)
50  006600B0 8F    D0  0283  1048          MOVL    #DS$_NOTIMP,R0
                   04  028A  1049          RET                                     ; RETURN
```

```
                        028B  1051          .SBTTL  DSP$SHOWUBI
                        028B  1052 ;++
                        028B  1053 ; FUNCTIONAL DESCRIPTION:
                        028B  1054 ;
                        028B  1055 ;       THIS PROCEEDURE WILL INTERPRET THE CONTENTS OF THE UBI
                        028B  1056 ;       POINTED TO BY THE ARGLIST
                        028B  1057 ;
                        028B  1058 ; CALLING SEQUENCE:
                        028B  1059 ;
                        028B  1060 ;       PUSHL   <CSR ADDR>
                        028B  1061 ;       CALLS   #1,DSP$SHOWUBI
                        028B  1062 ; OR    CALLG   <ARGLIST>,DSP$SHOWUBI
                        028B  1063 ;
                        028B  1064 ; INPUT PARAMETERS:
                        028B  1065 ;
                        028B  1066 ;       4(AP)   -> ADDRESS OF UBI CSR
                        028B  1067 ;
                        028B  1068 ; IMPLICIT INPUTS:
                        028B  1069 ;
                        028B  1070 ;       NONE
                        028B  1071 ;
                        028B  1072 ; OUTPUT PARAMETERS:
                        028B  1073 ;
                        028B  1074 ;       NONE
                        028B  1075 ;
                        028B  1076 ; IMPLICIT OUTPUTS:
                        028B  1077 ;
                        028B  1078 ;       NONE
                        028B  1079 ;
                        028B  1080 ; COMPLETION CODES:
                        028B  1081 ;
                        028B  1082 ;       DS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                        028B  1083 ;
                        028B  1084 ; SIDE EFFECTS:
                        028B  1085 ;
                        028B  1086 ;       NONE
                        028B  1087 ;--
                        028B  1088
                 007C   028B  1089 .ENTRY  DSP$SHOWUBI,^M<R2,R3,R4,R5,R6>          ; ENTRY MASK (SAME AS SHOWCHAN)
      56   04 AC    D0   028D  1090         MOVL    4(AP),R6                       ; COPY ADDRESS OF UBI CSR
           002C     30   0291  1091         BSBW    SHOW_UBI                       ; INTERPRET THE UBI ADDRESS
50   00660001 8F    D0   0294  1092         MOVL    #DS$_NORMAL,R0
                     04   029B  1093         RET                                    ; SUCCESS OF COURSE
```

```
                     029C  1095          .SBTTL  $SHOWCHAN CALL
                     029C  1096 ;++
                     029C  1097 ; FUNCTIONAL DESCRIPTION:
                     029C  1098 ;
                     029C  1099 ;       DISPLAY ADAPTER REGISTERS
                     029C  1100 ;
                     029C  1101 ; CALLING SEQUENCE:
                     029C  1102 ;
                     029C  1103 ;       CALLG   ARGLST, a#DS$SHOWCHAN
                     029C  1104 ;         OR
                     029C  1105 ;       CALLS   #1,a#DS$SHOWCHAN
                     029C  1106 ;
                     029C  1107 ; INPUT PARAMETERS:
                     029C  1108 ;
                     029C  1109 ;       4(AP) LOGICAL UNIT NUMBER
                     029C  1110 ;
                     029C  1111 ; IMPLICIT INPUTS:
                     029C  1112 ;
                     029C  1113 ;       NONE
                     029C  1114 ;
                     029C  1115 ; OUTPUT PARAMETERS:
                     029C  1116 ;
                     029C  1117 ;       R0      =       COMPLETION CODE
                     029C  1118 ;
                     029C  1119 ; IMPLICIT OUTPUTS:
                     029C  1120 ;
                     029C  1121 ;       NONE
                     029C  1122 ;
                     029C  1123 ; COMPLETION CODES:
                     029C  1124 ;
                     029C  1125 ;       DS$_NORMAL      =       SUCCESS
                     029C  1126 ;       DS$_PROGERR     =       PROGRAM ERROR
                     029C  1127 ;                                MISSING/INVALID P-TABLE
                     029C  1128 ;
                     029C  1129 ; SIDE EFFECTS:
                     029C  1130 ;
                     029C  1131 ;       NONE
                     029C  1132 ;--
                     029C  1133
                 007C  029C  1134 .ENTRY  DSX$SHOWCHAN,^M<R2,R3,R4,R5,R6>
                     029E  1135
      52   04 AC   D0  029E  1136          MOVL    SHW$_UNIT(AP),R2                ; GET THE LOGICAL UNIT NUMBER
           0065    30  02A2  1137          BSBW    BLDCDB                          ; BUILD THE CDB
         17 50     E9  02A5  1138          BLBC    R0, 20$                         ; Exit if error                 [09]
                     02A8  1139
           0015    30  02A8  1140 10$:     BSBW    SHOW_UBI                        ; INTERPRET STATUS FOR UBI
                     02AB  1141
                     02AB  1142          $DS_PRINTB_S    L^FMT_CRLF              ; SEPARATE THE SHOWCHAN         [07]
    00000000'EF    9F  02AB                        PUSHAB  L^FMT_CRLF
00000000'9F     01    FB  02B1                        CALLS   #$$N, a#DS$PRINTB
50   00660001 8F    D0  02B8  1143          MOVL    #DS$_NORMAL,R0                  ; INDICATE SUCCESS
                     02BF  1144
                 04  02BF  1145 20$:     RET                                     ; RETURN
```

```
                     02C0  1147          .SBTTL  SHOW_UBI STATUS
                     02C0  1148 ;++
                     02C0  1149 ; FUNCTIONAL DESCRIPTION:
                     02C0  1150 ;
                     02C0  1151 ;       SUBROUTINE TO PRINT UBI STATUS
                     02C0  1152 ;
                     02C0  1153 ; CALLING SEQUENCE:
                     02C0  1154 ;
                     02C0  1155 ;       BSBW    SHOW_UBI
                     02C0  1156 ;
                     02C0  1157 ; INPUT PARAMETERS:
                     02C0  1158 ;
                     02C0  1159 ;       R6=ADDRESS OF UBI CSR
                     02C0  1160 ;
                     02C0  1161 ; IMPLICIT INPUTS:      NONE
                     02C0  1162 ;
                     02C0  1163 ; OUTPUT PARAMETERS:    NONE
                     02C0  1164 ;
                     02C0  1165 ; IMPLICIT OUTPUTS:     NONE
                     02C0  1166 ;
                     02C0  1167 ; COMPLETION CODES:     NONE
                     02C0  1168 ;
                     02C0  1169 ; SIDE EFFECTS:         NONE
                     02C0  1170 ;--
                     02C0  1171
                     02C0  1172 SHOW_UBI:
                     02C0  1173          $DS_PRINTB_S    L^FMT_UBI_HDR           ; PRINT THE HEADER              [07]
      00000003'EF    9F    02C0                  PUSHAB    L^FMT_UBI_HDR
00000000'9F    01    FB    02C6                  CALLS     #$$N, @#DS$PRINTB
                     05    02CD  1174          RSB                               ; RETURN
```

```
                                02CE  1176          .SBTTL  SHOWBITS
                                02CE  1177 ;++
                                02CE  1178 ; FUNCTIONAL DESCRIPTION:
                                02CE  1179 ;
                                02CE  1180 ;       SUBROUTINE TO CVTREG AND PRINTB CSR BITS
                                02CE  1181 ;
                                02CE  1182 ; CALLING SEQUENCE:
                                02CE  1183 ;
                                02CE  1184 ;       BSBW    SHOWBITS
                                02CE  1185 ;
                                02CE  1186 ; INPUT PARAMETERS:
                                02CE  1187 ;
                                02CE  1188 ;       R2      ADDRESS OF FAO STRING TO BE OUTPUT
                                02CE  1189 ;       R3      ADDRESS OF DEVICE REGISTER
                                02CE  1190 ;       R4      ADDRESS OF ASCIC BIT DESCRIPTOR STRING
                                02CE  1191 ;
                                02CE  1192 ; IMPLICIT INPUTS:      NONE
                                02CE  1193 ;
                                02CE  1194 ; OUTPUT PARAMETERS:    NONE
                                02CE  1195 ;
                                02CE  1196 ; IMPLICIT OUTPUTS:     NONE
                                02CE  1197 ;
                                02CE  1198 ; COMPLETION CODES:     NONE
                                02CE  1199 ;
                                02CE  1200 ; SIDE EFFECTS:         NONE
                                02CE  1201 ;--
                                02CE  1202
                                02CE  1203 SHOWBITS:
        00000028'EF   63    D0  02CE  1204          MOVL    (R3), FMT_CVT_ARG+8             ; CONVERT DEVICE REGISTER BITS
        0000002C'EF   54    D0  02D5  1205          MOVL    R4,FMT_CVT_ARG+12               ; USING THIS MASK
                                02DC  1206          $DS_CVTREG_G  FMT_CVT_ARG               ; INTO ASCII IN SHWBUF
00000000'9F   00000020'EF   FA  02DC                        CALLG   FMT_CVT_ARG, a#DS$CVTREG
              00000068'EF   DF  02E7  1207          PUSHAL  SHWBUF                          ; PARAM 4 ADDRESS OF BIT ENCODINGS
                        63  DD  02ED  1208          PUSHL   (R3)                            ; PARAM 3 IS REG CONTENTS
    7E  53    C0000000  8F  CB  02EF  1209          BICL3   #3 a 30, R3, -(SP)              ; PARAM 2 IS REGISTER ADDRESS
                        62  9F  02F7  1210          PUSHAB  (R2)                            ; PARAM 1 IS FAO ASCIC STRING
        00000000'9F   04    FB  02F9  1211          CALLS   #4,a#DS$PRINTB                  ; CALL OUTPUT ROUTINE
                            05  0300  1212          RSB                                     ; RETURN
```

```
                     0301  1214          .SBTTL  DETERMINE ADAPTER SPECIFIC STATUS
                     0301  1215 ;++
                     0301  1216 ; FUNCTIONAL DESCRIPTION:
                     0301  1217 ;
                     0301  1218 ;       DETERMINE ADAPTER SPECIFIC STATUS
                     0301  1219 ;
                     0301  1220 ; CALLING SEQUENCE:
                     0301  1221 ;
                     0301  1222 ;       BSBW    ADPSTS
                     0301  1223 ;
                     0301  1224 ; INPUT PARAMETERS:
                     0301  1225 ;
                     0301  1226 ;       R5      =       CHANNEL DATA BLOCK
                     0301  1227 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0301  1228 ;
                     0301  1229 ; IMPLICIT INPUTS:      NONE
                     0301  1230 ;
                     0301  1231 ; OUTPUT PARAMETERS:    NONE
                     0301  1232 ;
                     0301  1233 ; IMPLICIT OUTPUTS:     NONE
                     0301  1234 ;
                     0301  1235 ; COMPLETION CODES:     NONE
                     0301  1236 ;
                     0301  1237 ; SIDE EFFECTS:         NONE
                     0301  1238 ;--
                     0301  1239
                     0301  1240 ADPSTS:
50   D4              0301  1241          CLRL    R0                          ; CLEAR R0
     05              0303  1242          RSB                                 ; RETURN
```

```
                        0304  1244          .SBTTL  DETERMINE GENERAL ADAPTER STATUS
                        0304  1245 ;++
                        0304  1246 ; FUNCTIONAL DESCRIPTION:
                        0304  1247 ;
                        0304  1248 ;       DETERMINE GENERAL ADAPTER STATUS
                        0304  1249 ;
                        0304  1250 ; CALLING SEQUENCE:
                        0304  1251 ;
                        0304  1252 ;       BSBW    GENSTS
                        0304  1253 ;
                        0304  1254 ; INPUT PARAMETERS:
                        0304  1255 ;
                        0304  1256 ;       R5      =       CHANNEL DATA BLOCK
                        0304  1257 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                        0304  1258 ;
                        0304  1259 ; IMPLICIT INPUTS:      NONE
                        0304  1260 ;
                        0304  1261 ; OUTPUT PARAMETERS:    NONE
                        0304  1262 ;
                        0304  1263 ; IMPLICIT OUTPUTS:     NONE
                        0304  1264 ;
                        0304  1265 ; COMPLETION CODES:     NONE
                        0304  1266 ;
                        0304  1267 ; SIDE EFFECTS:         NONE
                        0304  1268 ;--
                        0304  1269
                        0304  1270 GENSTS:
     50   D4            0304  1271          CLRL    R0                              ; CLEAR R0
          05            0306  1272          RSB                                     ; RETURN
```

```
                     0307  1274          .SBTTL  DETERMINE ADAPTER HARDWARE ERROR STATUS
                     0307  1275 ;++
                     0307  1276 ; FUNCTIONAL DESCRIPTION:
                     0307  1277 ;
                     0307  1278 ;       DETERMINE ADAPTER HARDWARE ERROR STATUS
                     0307  1279 ;
                     0307  1280 ; CALLING SEQUENCE:
                     0307  1281 ;
                     0307  1282 ;       BSBW    HDWSTS
                     0307  1283 ;
                     0307  1284 ; INPUT PARAMETERS:
                     0307  1285 ;
                     0307  1286 ;       R5      =       CHANNEL DATA BLOCK
                     0307  1287 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0307  1288 ;
                     0307  1289 ; IMPLICIT INPUTS:      NONE
                     0307  1290 ;
                     0307  1291 ; OUTPUT PARAMETERS:    NONE
                     0307  1292 ;
                     0307  1293 ; IMPLICIT OUTPUTS:     NONE
                     0307  1294 ;
                     0307  1295 ; COMPLETION CODES:     NONE
                     0307  1296 ;
                     0307  1297 ; SIDE EFFECTS:         NONE
                     0307  1298 ;--
                     0307  1299
                     0307  1300 HDWSTS:
 50   D4             0307  1301          CLRL    R0                              ; CLEAR R0
      05             0309  1302          RSB                                     ; RETURN
```

```
                                030A  1304
                                030A  1305          .SBTTL  BUILD THE CHANNEL DATA BLOCK
                                030A  1306 ;**
                                030A  1307 ; FUNCTIONAL DESCRIPTION:
                                030A  1308 ;
                                030A  1309 ; ROUTINE TO BUILD A CHANNEL DATA BLOCK FOR
                                030A  1310 ; LOGICAL UNIT NUMBER FOUND IN R2
                                030A  1311 ;
                                030A  1312 ; CALLING SEQUENCE:
                                030A  1313 ;
                                030A  1314 ;       BSBW    BLDCDB
                                030A  1315 ;
                                030A  1316 ;
                                030A  1317 ; INPUT PARAMETERS:
                                030A  1318 ;
                                030A  1319 ;       R2      =       LOGICAL UNIT NUMBER
                                030A  1320 ;
                                030A  1321 ; IMPLICIT INPUTS:      NONE
                                030A  1322 ;
                                030A  1323 ; OUTPUT PARAMETERS:
                                030A  1324 ;
                                030A  1325 ;       R0      =       COMPLETION CODE
                                030A  1326 ;       R5      =       CDB ADDRESS
                                030A  1327 ;       R6      =       ADAPTER ADDRESS
                                030A  1328 ;
                                030A  1329 ; IMPLICIT OUTPUTS:     NONE
                                030A  1330 ;
                                030A  1331 ; COMPLETION CODES:
                                030A  1332 ;
                                030A  1333 ;       DS$_NORMAL      =       SUCCESS
                                030A  1334 ;       DS$_ERROR       =       ERROR ($GPHARD)
                                030A  1335 ;                                 MISSING/INVALID P-TABLE
                                030A  1336 ;
                                030A  1337 ; SIDE EFFECTS:
                                030A  1338 ;
                                030A  1339 ;       DESTROYS R5, R6
                                030A  1340 ;       SETS FLAG (CDB$M_UBI) TO INDICATE ADAPTER TYPE
                                030A  1341 ;
                                030A  1342 ;--
                                030A  1343
                                030A  1344 BLDCDB:
                                030A  1345          $DS_GPHARD_S    R2,-(SP)                ; GET THE HARDWARE P-TABLE
                     7E   DF    030A                        PUSHAL  -(SP)
                     52   DD    030C                        PUSHL   R2
00000000'9F          02   FB    030E                        CALLS   #2, a#DS$GPHARD
               56    8E   D0    0315  1346          MOVL    (SP)+,R6                        ; GET ADDRESS OF HP-TABLE
                  2C 50   E9    0318  1347          BLBC    R0, BLD_RTN             ; IS THERE ONE ?                        [05]
55     00000000'EF   DE    031B  1348          MOVAL   CDB$BLOCK,R5                    ; CDB TO R5
65     FFFFFFDF 8F   CA    0322  1349          BICL    #^CCDB$M_IE,CDB$L_FLAGS(R5)     ; Clear flags
                           0329  1350
               20 A6    D5  0329  1351 10$:    TSTL    HP$A_LINK(R6)                   ; SEARCH FOR ADAPTER ENTRY
                  06    13  032C  1352          BEQL    20$                             ; FOUND IT
            56 20 A6    D0  032E  1353          MOVL    HP$A_LINK(R6),R6                ; KEEP LOOKING
                  F5    11  0332  1354          BRB     10$                             ;
                           0334  1355 20$:
            04 A5   56    D0  0334  1356          MOVL    R6,CDB$A_TBL(R5)                ; SAVE THE P-TABLE ADDRESS
            56   18 A6    D0  0338  1357          MOVL    HP$A_DEVICE(R6),R6              ; Base of CSR's
```

```
                                 033C  1358
          00 65    02    E3      033C  1359          BBCS    #CDB$V_UBI,CDB$L_FLAGS(R5),40$  ; Flag as an UBI
                                 0340  1360
     50   00660001 8F    D0      0340  1361 40$:     MOVL    #DS$_NORMAL,R0                  ; INDICATE SUCCESS
                                 0347  1362
                                 0347  1363 BLD_RTN:
                          05      0347  1364          RSB                                     ; RETURN
```

```
                                0348  1366          .SBTTL  SET ADAPTER VECTOR(S)
                                0348  1367 ;++
                                0348  1368 ; FUNCTIONAL DESCRIPTION:
                                0348  1369 ;
                                0348  1370 ;       SET SCB VECTOR FOR ADAPTER/DEVICE TO POINT TO CHANNEL SERVICE
                                0348  1371 ;
                                0348  1372 ; CALLING SEQUENCE:
                                0348  1373 ;
                                0348  1374 ;       BSBW    SETVEC
                                0348  1375 ;
                                0348  1376 ; INPUT PARAMETERS:
                                0348  1377 ;
                                0348  1378 ;       R5      =       CHANNEL DATA BLOCK
                                0348  1379 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                                0348  1380 ;
                                0348  1381 ; IMPLICIT INPUTS:      NONE
                                0348  1382 ;
                                0348  1383 ; OUTPUT PARAMETERS:    NONE
                                0348  1384 ;
                                0348  1385 ; IMPLICIT OUTPUTS:     NONE
                                0348  1386 ;
                                0348  1387 ; COMPLETION CODES:
                                0348  1388 ;
                                0348  1389 ;       DS$_NORMAL      =       SUCCESS
                                0348  1390 ;       DS$_IVVECT      =       INVALID VECTOR ($SETVEC)
                                0348  1391 ;       DS$_IVADDR      =       INVALID ADDRESS ($SETVEC)
                                0348  1392 ;
                                0348  1393 ; SIDE EFFECTS:         NONE
                                0348  1394 ;--
                                0348  1395
                                0348  1396 SETVEC:
                  04    BB      0348  1397          PUSHR   #^M<R2>                         ; SAVE A WORKING REGISTER
                                034A  1398
        51    04 A5   D0      034A  1399          MOVL    CDB$A_TBL(R5),R1                ; GET THE HARDWARE P-TABLE
    10 A5     24 A1   3C      034E  1400          MOVZWL  HP$W_VECTOR(R1),CDB$A_ADDR(R5)  ; SAVE CHANNEL VECTOR
          65  20      C8      0353  1401          BISL    #CDB$M_IE,CDB$L_FLAGS(R5)       ; INDICATE INTERRUPTS ENABLED
                                0356  1402
51    00000200'EF     7E      0356  1403          MOVAQ   SCB_BASE+512,R1                 ; Point to SCB
52    00000200'EF     9E      035D  1404          MOVAB   SCB_UNKINT+512,R2               ; Point to image
    50    0100 8F     3C      0364  1405          MOVZWL  #2*<512/4>,R0                   ; Number of longwords to fill
                                0369  1406
          81    82    DE      0369  1407 10$:     MOVAL   (R2)+,(R1)+                     ; Do one SCB page
             FA 50    F5      036C  1408          SOBGTR  R0,10$                          ; Copy whole page
50    00660001 8F     D0      036F  1409          MOVL    #DS$_NORMAL,R0                  ; INDICATE SUCCESS
                  04    BA      0376  1410          POPR    #^M<R2>                         ; RESTORE REGISTERS
                        05      0378  1411          RSB
```

```
                     0379  1413          .SBTTL  CLEAR ADAPTER VECTOR(S)
                     0379  1414 ;**
                     0379  1415 ; FUNCTIONAL DESCRIPTION:
                     0379  1416 ;
                     0379  1417 ;       CLEAR CHANNEL SERVICE POINTERS IN SCB VECTOR SPACE
                     0379  1418 ;
                     0379  1419 ; CALLING SEQUENCE:
                     0379  1420 ;
                     0379  1421 ;       BSBW    CLRVEC
                     0379  1422 ;
                     0379  1423 ; INPUT PARAMETERS:
                     0379  1424 ;
                     0379  1425 ;       R5      =       CHANNEL DATA BLOCK
                     0379  1426 ;       R6      =       CHANNEL CSR REGISTER ADDRESS
                     0379  1427 ;
                     0379  1428 ; IMPLICIT INPUTS:
                     0379  1429 ;
                     0379  1430 ;       NONE
                     0379  1431 ;
                     0379  1432 ; OUTPUT PARAMETERS:
                     0379  1433 ;
                     0379  1434 ;       R0      =       COMPLETION CODE
                     0379  1435 ;
                     0379  1436 ; IMPLICIT OUTPUTS:
                     0379  1437 ;
                     0379  1438 ;       NONE
                     0379  1439 ;
                     0379  1440 ; COMPLETION CODES:
                     0379  1441 ;
                     0379  1442 ;       DS$_NORMAL      =       SUCCESS
                     0379  1443 ;       DS$_IVVECT      =       INVALID VECTOR ($CLRVEC)
                     0379  1444 ;
                     0379  1445 ; SIDE EFFECTS:
                     0379  1446 ;
                     0379  1447 ;       NONE
                     0379  1448 ;--
                     0379  1449
                     0379  1450 CLRVEC:
 65    20    CA      0379  1451          BICL    #CDB$M_IE,CDB$L_FLAGS(R5)       ; CLEAR IE
       05          037C  1452          RSB                                     ; RETURN WITH COMPLETION CODE
```

```
                        037D  1454          .SBTTL  INTERRUPT PRE-PROCESS
                        037D  1455 ;++
                        037D  1456 ; FUNCTIONAL DESCRIPTION:
                        037D  1457 ;
                        037D  1458 ;       THIS ROUTINE WILL PRE-PROCESS ALL CHANNEL ADAPTER INTERRUPTS
                        037D  1459 ;       AND PASS CONTROL TO THE DIAGNOSTIC PROGRAM
                        037D  1460 ;
                        037D  1461 ; CALLING SEQUENCE:
                        037D  1462 ;
                        037D  1463 ;       INTERRUPT
                        037D  1464 ;
                        037D  1465 ; INPUT PARAMETERS:
                        037D  1466 ;
                        037D  1467 ;       NONE
                        037D  1468 ;
                        037D  1469 ; IMPLICIT INPUTS:
                        037D  1470 ;
                        037D  1471 ;       CDB$A_VEC MUST HAVE BEEN PREVIOUSLY SET BY $CHANNEL CALL TO
                        037D  1472 ;       ENABLE INTERRUPTS
                        037D  1473 ;
                        037D  1474 ; OUTPUT PARAMETERS:
                        037D  1475 ;
                        037D  1476 ;       THE CHANNEL STATUS QUADWORD IS GENERATED
                        037D  1477 ;
                        037D  1478 ; IMPLICIT OUTPUTS:
                        037D  1479 ;
                        037D  1480 ;       NONE
                        037D  1481 ;
                        037D  1482 ; COMPLETION CODES:
                        037D  1483 ;
                        037D  1484 ;       NONE
                        037D  1485 ;
                        037D  1486 ; SIDE EFFECTS:
                        037D  1487 ;
                        037D  1488 ;--
```

```
                                037D  1490          .ALIGN  LONG
                                0380  1491 DSI$CHANNEL_VEC::
        06 00000000'EF    05    E0  0380  1492          BBS     #CDB$V_IE,CDB$L_FLAGS+CDB$BLOCK, -
                                0388  1493                          DSI$CHANNEL_COM         ; IS INTERRUPT EXPECTED
                                05    0388  1494          RSB                             ; NO - RETURN TO UNEXPECTED HANDLER
                                      0389  1495
                                      0389  1496          .ALIGN  LONG
                                      038C  1497 DSI$CHANNEL:
                             7E   7C  038C  1498          CLRQ    -(SP)                   ; Dummy space to match UBI
                                      038E  1499 DSI$CHANNEL_COM:
                      0063 8F    BB  038E  1500          PUSHR   #^M<R0,R1,R5,R6>        ; SAVE REGISTERS
                                      0392  1501                                          ; SAVE INTERRUPT LEVEL OF DEVICE
                                      0392  1502          DSBINT  #^X17                   ; & DISABLE DEVICE INTERRUPTS
                      7E   12   DB  0392                  MFPR    S^#PR$_IPL,-(SP)
                      12   17   DA  0395                  MTPR    #^X17,S^#PR$_IPL
                                      0398  1503
                55    00000000'EF   DE  0398  1504          MOVAL   CDB$BLOCK,R5            ; CDB ADDRESS TO R5
                       56   04 A5   D0  039F  1505          MOVL    CDB$A_TBL(R5),R6        ; P-TABLE ADDRESS TO R6
                       56   18 A6   D0  03A3  1506          MOVL    HP$A_DEVICE(R6),R6      ; CHANNEL BASE ADDRESS TO R6
                                      03A7  1507
                          FF5D   30  03A7  1508          BSBW    HDWSTS                  ; GET HARDWARE ERROR STATUS
                       51   50   D0  03AA  1509          MOVL    R0,R1                   ; SAVE THE STATUS
                          FF51   30  03AD  1510          BSBW    ADPSTS                  ; GET ADAPTER SPECIFIC STATUS
                       50   51   C8  03B0  1511          BISL    R1,R0                   ; COMBINE STATUS
                     14 A5   50   D0  03B3  1512          MOVL    R0,CDB$L_ST1(R5)        ; SAVE IT
                          FF4A   30  03B7  1513          BSBW    GENSTS                  ; GET GENERAL STATUS
                     18 A5   50   B0  03BA  1514          MOVW    R0,CDB$W_ST2(R5)        ; SAVE IT
                                      03BE  1515
        18 A5   05   02   8E   F0  03BE  1516          INSV    (SP)+,#CHI$V_IPL,#CHI$S_IPL, -  ; STORE THE IPL OF THE INTERRUPT
                                      03C4  1517                    CDB$W_ST2(R5)
                                      03C4  1518
    1A A5   14 AE    FE00 8F   AB  03C4  1519          BICW3   #^C^X1FF,20(SP),CDB$W_RVR(R5)   ; SAVE THE VECTOR
                     18 A5   02   A8  03CC  1520          BISW    #CHI$M_DEVINT,CDB$W_ST2(R5)     ; FLAG AS DEVICE INTERRUPT
                                      03D0  1521
                       50   0C A5   D0  03D0  1522          MOVL    CDB$A_STORE(R5),R0      ; HANDLER'S STORE AREA TO R0
                       80   14 A5   D0  03D4  1523          MOVL    CDB$L_ST1(R5),(R0)+     ; HARDWARE ERROR STATUS
                       60   18 A5   D0  03D8  1524          MOVL    CDB$W_ST2(R5),(R0)      ; RVR + GENERAL STATUS
                                      03DC  1525
                    14 AE   08 A5   D0  03DC  1526          MOVL    CDB$A_VEC(R5),20(SP)    ; BUILD THE HANDLER ADDRESS
                                      03E1  1527
                                      03E1  1528 INT_DSP:
                      0063 8F    BA  03E1  1529          POPR    #^M<R0,R1,R5,R6>        ; RESTORE R0, R1, R5, R6
                           8E    D5  03E5  1530          TSTL    (SP)+                   ; Remove JSB return
                                 05  03E7  1531          RSB                             ; Dispatch to handler
                                      03E8  1532
                                      03E8  1533
                                      03E8  1534          .END
```

```
$$ARGS                          = 00000001     D
$$N                             = 00000001     D
$$T1                            = 00000008     D
$ER                             = 00000001     D
$MODULE                           00000060 R   D   02
ADPSTS                            00000301 R   D   04
BASCHK                            0000022F R   D   04
BIT...                          = 0000001F     D
BIT0                            = 00000001     D
BIT1                            = 00000002     D
BIT10                           = 00000400     D
BIT11                           = 00000800     D
BIT12                           = 00001000     D
BIT13                           = 00002000     D
BIT14                           = 00004000     D
BIT15                           = 00008000     D
BIT16                           = 00010000     D
BIT17                           = 00020000     D
BIT18                           = 00040000     D
BIT19                           = 00080000     D
BIT2                            = 00000004     D
BIT20                           = 00100000     D
BIT21                           = 00200000     D
BIT22                           = 00400000     D
BIT23                           = 00800000     D
BIT24                           = 01000000     D
BIT25                           = 02000000     D
BIT26                           = 04000000     D
BIT27                           = 08000000     D
BIT28                           = 10000000     D
BIT29                           = 20000000     D
BIT3                            = 00000008     D
BIT30                           = 40000000     D
BIT31                           = 80000000     D
BIT4                            = 00000010     D
BIT5                            = 00000020     D
BIT6                            = 00000040     D
BIT7                            = 00000080     D
BIT8                            = 00000100     D
BIT9                            = 00000200     D
BLDCDB                            0000030A R   D   04
BLD_RTN                           00000347 R   D   04
CDB$A_ADDR                        00000010     D
CDB$A_STORE                       0000000C     D
CDB$A_TBL                         00000004     D
CDB$A_VEC                         00000008     D
CDB$BLOCK                         00000000 RG  D   02
CDB$L_FLAGS                       00000000     D
CDB$L_PTH                         0000001C     D
CDB$L_ST1                         00000014     D
CDB$M_IE                        = 00000020     D
CDB$M_MBA                       = 00000001     D
CDB$M_OFFSET                    = 00000008     D
CDB$M_REV                       = 00000010     D
CDB$M_UBA                       = 00000002     D
CDB$M_UBI                       = 00000004     D
CDB$V_IE                        = 00000005     D
CDB$V_MBA                       = 00000000     D
CDB$V_OFFSET                    = 00000003     D
CDB$V_REV                       = 00000004     D
CDB$V_UBA                       = 00000001     D
CDB$V_UBI                       = 00000002     D
CDB$W_RVR                         0000001A     D
CDB$W_ST2                         00000018     D
CDBSIZ                          = 00000020     D
CHC$_ABORT                      = 00000002     D
CHC$_CLEAR                      = 00000006     D
CHC$_CLRDFT                     = 00000009     D
CHC$_DSINT                      = 00000005     D
CHC$_ENINT                      = 00000004     D
CHC$_INITA                      = 00000000     D
CHC$_INITB                      = 00000001     D
CHC$_PURGE                      = 00000003     D
CHC$_SELF_TEST                  = 0000000B     D
CHC$_SETDFT                     = 00000008     D
CHC$_STATUS                     = 00000007     D
CHC$_STOP                       = 0000000A     D
CHI$M_CHNINT                    = 00000001     D
CHI$M_DEVINT                    = 00000002     D
CHI$M_IPL                       = 0000007C     D
CHI$S_IPL                       = 00000005     D
CHI$V_CHNINT                    = 00000000     D
CHI$V_DEVINT                    = 00000001     D
CHI$V_IPL                       = 00000002     D
CHM$_FORWARD                    = 00000000     D
CHM$_INVALIDATE                 = 00000001     D
CHM$_MAP                        = 00000002     D
CHM$_MFWDN                      = 00000002     D
CHM$_MFWDNO                     = 0000000A     D
CHM$_MFWDV                      = 00000003     D
CHM$_MFWDVO                     = 0000000B     D
CHM$_MREVN                      = 00000006     D
CHM$_MREVNO                     = 0000000E     D
CHM$_MREVV                      = 00000007     D
CHM$_MREVVO                     = 0000000F     D
CHM$_NFWDN                      = 00000000     D
CHM$_NREVN                      = 00000004     D
CHM$_OFFSET                     = 00000008     D
CHM$_REVERSE                    = 00000004     D
CHN$_ADR1                       = 0000000C     D
CHN$_ADR2                       = 00000010     D
CHN$_FUNC                       = 00000008     D
CHN$_NARGS                      = 00000004     D
CHN$_UNIT                       = 00000004     D
CHN_ABORT                         0000004E R   D   04
CHN_CALL_RTN                      000000B0 R   D   04
CHN_CLEAR                         00000084 R   D   04
CHN_CLRDFT                        000000A0 R   D   04
CHN_DSINT                         0000007F R   D   04
CHN_DSP                           0000000F R   D   04
CHN_ENINT                         00000057 R   D   04
CHN_INITA                         00000035 R   D   04
CHN_INITB                         00000041 R   D   04
CHN_NORMAL_RTN                    000000A9 R   D   04
```

```
CHN_PURGE                       00000076 R  D  04      CHS$V_UIE                       = 0000001C     D
CHN_SETDFT                      00000097 R  D  04      CLRMAP                            0000021D R  D  04
CHN_STATUS                      0000008D R  D  04      CLRVEC                            00000379 R  D  04
CHS$M_BADBDP                  = 20000000    D        CNTCHK                            0000024F R  D  04
CHS$M_BADPAR                  = 40000000    D        CNTCHK_RTN                        00000276 R  D  04
CHS$M_BUSIC                   = 00800000    D        CVT$T_MBAMAP                      00000054 RG D  02
CHS$M_BUSINIT                 = 00400000    D        CVT$T_MBASR                       00000050 RG D  02
CHS$M_BUSNXM                  = 08000000    D        CVT$T_UBADPR                      00000058 RG D  02
CHS$M_BUSPDN                  = 01000000    D        CVT$T_UBAMAP                      0000005C RG D  02
CHS$M_CHNDPE                  = 00000040    D        DS$CVTREG                         ********    X  04
CHS$M_CHNERR                  = 00000002    D        DS$GPHARD                         ********    X  04
CHS$M_CHNMPE                  = 00000080    D        DS$K_ERROR                      = 00000002     D
CHS$M_CHPFOT                  = 00000100    D        DS$K_NORMAL                     = 00000001     D
CHS$M_DEVBUS                  = 00000010    D        DS$K_SEVERE                     = 00000004     D
CHS$M_DEVERR                  = 00000004    D        DS$K_SUBSYS                     = 00000066     D
CHS$M_DEVTO                   = 00000020    D        DS$K_WARNING                    = 00000000     D
CHS$M_ERRANY                  = 0000000F    D        DS$PRINTB                         ********    X  04
CHS$M_MBAATN                  = 00100000    D        DS$_AP_NORMAL_BREAK             = 00660150     D
CHS$M_MBACPE                  = 00200000    D        DS$_ARITH                       = 006600D0     D
CHS$M_MBADTB                  = 00040000    D        DS$_ASBE                        = 00660118     D
CHS$M_MBADTC                  = 00080000    D        DS$_BADLINK                     = 006600F0     D
CHS$M_MBAEXC                  = 00010000    D        DS$_BADTYPE                     = 006600E8     D
CHS$M_MBANED                  = 00020000    D        DS$_BIIC                        = 00660120     D
CHS$M_MBAWCKLWR               = 02000000    D        DS$_CHME                        = 006600A8     D
CHS$M_MBAWCKUPR               = 04000000    D        DS$_CHMK                        = 006600E0     D
CHS$M_PGMERR                  = 00000008    D        DS$_DEVNAME                     = 00660108     D
CHS$M_PGMHDE                  = 00000800    D        DS$_ERROR                       = 00660002     D
CHS$M_SYSERR                  = 00000001    D        DS$_FHWE                        = 00660068     D
CHS$M_SYSMEM                  = 00000200    D        DS$_FRAGBUF                     = 00660080     D
CHS$M_SYSSBI                  = 00000400    D        DS$_ICBUSY                      = 006600C8     D
CHS$M_UIE                     = 10000000    D        DS$_ICERR                       = 006600C0     D
CHS$V_BADBDP                  = 0000001D    D        DS$_IHWE                        = 00660060     D
CHS$V_BADPAR                  = 0000001E    D        DS$_ILLCHAR                     = 00660018     D
CHS$V_BUSIC                   = 00000017    D        DS$_ILLPAGCNT                   = 00660078     D
CHS$V_BUSINIT                 = 00000016    D        DS$_ILLUNIT                     = 00660100     D
CHS$V_BUSNXM                  = 0000001B    D        DS$_INIT_FAIL                   = 00660140     D
CHS$V_BUSPDN                  = 00000018    D        DS$_INSFMEM                     = 00660050     D
CHS$V_CHNDPE                  = 00000006    D        DS$_INVCPU                      = 00660130     D
CHS$V_CHNERR                  = 00000001    D        DS$_IPL2HI                      = 006600B8     D
CHS$V_CHNMPE                  = 00000007    D        DS$_IVADDR                      = 00660040     D
CHS$V_CHPFOT                  = 00000008    D        DS$_IVVECT                      = 00660038     D
CHS$V_DEVBUS                  = 00000004    D        DS$_KRNLSTK                     = 00660090     D
CHS$V_DEVERR                  = 00000002    D        DS$_LOGIC                       = 00660070     D
CHS$V_DEVTO                   = 00000005    D        DS$_MCHK                        = 00660088     D
CHS$V_MBAATN                  = 00000014    D        DS$_MEM_ALLOC_ERR               = 00660138     D
CHS$V_MBACPE                  = 00000015    D        DS$_MMOFF                       = 00660058     D
CHS$V_MBADTB                  = 00000012    D        DS$_NEEDUNIT                    = 006600F8     D
CHS$V_MBADTC                  = 00000013    D        DS$_NODE                        = 00660128     D
CHS$V_MBAEXC                  = 00000010    D        DS$_NOPCS                       = 00660110     D
CHS$V_MBANED                  = 00000011    D        DS$_NORMAL                      = 00660001     D
CHS$V_MBAWCKLWR               = 00000019    D        DS$_NOSUPPORT                   = 006600B1     D
CHS$V_MBAWCKUPR               = 0000001A    D        DS$_NOTALLOWMP                  = 00660148     D
CHS$V_PGMERR                  = 00000003    D        DS$_NOTDON                      = 00660030     D
CHS$V_PGMHDE                  = 0000000B    D        DS$_NOTIMP                      = 006600B0     D
CHS$V_SYSERR                  = 00000000    D        DS$_NULLSTR                     = 00660010     D
CHS$V_SYSMEM                  = 00000009    D        DS$_OVERFLOW                    = 00660008     D
CHS$V_SYSSBI                  = 0000000A    D        DS$_POWER                       = 00660098     D
```

```
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_UNSUPPDEV                   = 00660158     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSI$CHANNEL                       0000038C R   D   04
DSI$CHANNEL_COM                   0000038E R   D   04
DSI$CHANNEL_VEC                   00000380 RG  D   04
DSP$SHOWMBA                       00000281 RG  D   04
DSP$SHOWUBI                       0000028B RG  D   04
DSX$CHANNEL                       00000000 RG  D   04
DSX$SETMAP                        000000B1 RG  D   04
DSX$SHOWCHAN                      0000029C RG  D   04
FMT_CRLF                          00000000 R   D   03
FMT_CVT_ARG                       00000020 R   D   02
FMT_UBI_CSR                       0000001F R   D   03
FMT_UBI_HDR                       00000003 R   D   03
GENSTS                            00000304 R   D   04
HDWSTS                            00000307 R   D   04
HP$A_DEPENDENT                    00000032     D
HP$A_DEVICE                       00000018     D
HP$A_DVA                          0000001C     D
HP$A_LINK                         00000020     D
HP$B_DRIVE                        0000000B     D
HP$B_FLAGS                        0000000A     D
HP$Q_DEVICE                       00000000     D
HP$T_DEVICE                       0000000C     D
HP$T_TYPE                         00000026     D
HP$W_SIZE                         00000008     D
HP$W_VECTOR                       00000024     D
INT_DSP                           000003E1 R   D   04
INVALID                           0000012B R   D   04
INVALIDATE                        00000135 R   D   04
JMPADR                            000000E9 R   D   02
MBA_MAP_BIT                       0000003D R   D   03
MBA_SR_BIT                        0000003D R   D   03
MFWDN                             00000150 R   D   04
MFWDNO                            000001B6 R   D   04
MFWDV                             0000013E R   D   04
MFWDVO                            000001A1 R   D   04
MREVN                             00000180 R   D   04
MREVNO                            000001DA R   D   04
MREVV                             0000016B R   D   04
MREVVO                            000001C8 R   D   04
NFWDN                             0000015F R   D   04
NREVN                             00000192 R   D   04
PAR$M_ATDEF                     = 00000010     D
PAR$M_ATHI                      = 00000008     D
PAR$M_ATLO                      = 00000004     D
PAR$M_NODEF                     = 00000002     D
PAR$V_ATDEF                     = 00000004     D
PAR$V_ATHI                      = 00000003     D
PAR$V_ATLO                      = 00000002     D
PAR$V_NODEF                     = 00000001     D
PAR$V_BIN                       = 00000002     D
PAR$V_DEC                       = 0000000A     D
PAR$V_HEX                       = 00000010     D
PAR$V_NO                        = 00000000     D
PAR$V_OCT                       = 00000008     D
PAR$V_YES                       = 00000001     D
PR$_IPL                         = 00000012     D
RNGCHK                            0000024B R   D   04
SCB_BASE                          ********     X   04
SCB_UNKINT                        ********     X   04
SETCNT                            0000022E R   D   04
SETMAP                            000001E9 R   D   04
SETVAR                            0000022D R   D   04
SETVEC                            00000348 R   D   04
SHOWBITS                          000002CE R   D   04
SHOW_UBI                          000002C0 R   D   04
SHW$K_BUFSIZ                    = 00000080     D
SHW$_NARGS                      = 00000001     D
SHW$_UNIT                       = 00000004     D
SHWBUF                            00000068 R   D   02
SIZ...                          = 00000001     D
STM$_BAS                        = 00000010     D
STM$_CNT                        = 00000014     D
STM$_FUNC                       = 00000008     D
STM$_NARGS                      = 00000006     D
STM$_PHY                        = 0000000C     D
STM$_PTH                        = 00000018     D
STM$_UNIT                       = 00000004     D
STM_CALL_RTN                      00000280 R   D   04
STM_NORMAL_RTN                    00000279 R   D   04
UBA$S_MAP_ADDR                  = 00000015     D
UBA$V_MAP_ADDR                  = 00000000     D
UBA_DPR_BIT                       0000003D R   D   03
UBA_MAP_BIT                       0000003E R   D   03
UBI$L_CSR                       = 00000000     D
UBI$L_MAP                       = 00000800     D
UBI$M_MAP_BO                    = 02000000     D
UBI$M_MAP_VALID                 = 80000000     D
UBI$S_MAP_BO                    = 00000001     D
UBI$S_MAP_DPD                   = 00000002     D
UBI$S_MAP_PFN                   = 0000000F     D
UBI$V_MAP_BO                    = 00000019     D
UBI$V_MAP_DPD                   = 00000015     D
UBI$V_MAP_PFN                   = 00000000     D
UBI$V_MAP_VALID                 = 0000001F     D
UBIVEC                            000000ED R   D   02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000032 (50.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	000000F1 (241.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000059 (89.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	000003E8 (1000.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000001	136 (2)	133 (2) 135 (2) 136 (2)
$$N	=00000001	1173 (22)	#-1142 (21) #-1173 (22)
$$T1	=00000008	136 (2)	133 (2) 135 (2) 136 (2)
$ER	=00000001	204 (3)	
$MODULE	00000060-R	204 (3)	
ADPSTS	00000301-R	1240 (24)	#-1510 (31)
BASCHK	0000022F-R	961 (18)	#-773 (18)
BIT...	=0000001F	132 (2)	108 (2) 129 (2) 130 (2) 132 (2)
BIT0	=00000001	128 (2)	
BIT1	=00000002	128 (2)	
BIT10	=00000400	128 (2)	
BIT11	=00000800	128 (2)	
BIT12	=00001000	128 (2)	
BIT13	=00002000	128 (2)	
BIT14	=00004000	128 (2)	
BIT15	=00008000	128 (2)	
BIT16	=00010000	128 (2)	
BIT17	=00020000	128 (2)	
BIT18	=00040000	128 (2)	
BIT19	=00080000	128 (2)	
BIT2	=00000004	128 (2)	
BIT20	=00100000	128 (2)	
BIT21	=00200000	128 (2)	
BIT22	=00400000	128 (2)	
BIT23	=00800000	128 (2)	
BIT24	=01000000	128 (2)	
BIT25	=02000000	128 (2)	
BIT26	=04000000	128 (2)	
BIT27	=08000000	128 (2)	
BIT28	=10000000	128 (2)	
BIT29	=20000000	128 (2)	
BIT3	=00000008	128 (2)	
BIT30	=40000000	128 (2)	
BIT31	=80000000	128 (2)	
BIT4	=00000010	128 (2)	
BIT5	=00000020	128 (2)	
BIT6	=00000040	128 (2)	
BIT7	=00000080	128 (2)	
BIT8	=00000100	128 (2)	
BIT9	=00000200	128 (2)	
BLDCDB	0000030A-R	1344 (28)	#-1137 (21) #-268 (6) #-756 (18)
BLD_RTN	00000347-R	1363 (28)	#-1347 (28)
CDB$A_ADDR	00000010	176 (3)	#-1400 (29)
CDB$A_STORE	0000000C	175 (3)	#-1522 (31) #-429 (10)
CDB$A_TBL	00000004	173 (3)	#-1356 (28) #-1399 (29) #-1505 (31)
CDB$A_VEC	00000008	174 (3)	#-1526 (31) #-427 (10)
CDB$BLOCK	00000000-R	183 (3)	1348 (28) 1492 (31) 1504 (31)
CDB$L_FLAGS	00000000	172 (3)	#-1349 (28) 1359 (28) #-1401 (29) #-1451 (30)
			1492 (31) #-805 (18) #-810 (18) #-819 (18)
			#-827 (18) #-833 (18) #-834 (18) #-842 (18)

```
                                                    #-843    (18)   #-850    (18)   #-851    (18)   #-857    (18)
                                                    #-858    (18)   #-866    (18)   #-867    (18)   #-875    (18)
                                                    #-884    (18)   #-905    (18)
CDB$L_PTH                    0000001C     180  (3)   #-765    (18)
CDB$L_ST1                    00000014     177  (3)   #-1512   (31)   #-1523   (31)   #-573    (14)   #-574    (14)
CDB$M_IE                    =00000020     156  (2)   #-1349   (28)   #-1401   (29)   #-1451   (30)
CDB$M_MBA                   =00000001     151  (2)
CDB$M_OFFSET                =00000008     154  (2)   #-804    (18)   #-809    (18)   #-818    (18)   #-826    (18)
                                                    #-833    (18)   #-842    (18)   #-850    (18)   #-858    (18)
                                                    #-867    (18)   #-874    (18)   #-883    (18)   #-905    (18)
CDB$M_REV                   =00000010     155  (2)   #-804    (18)   #-809    (18)   #-818    (18)   #-826    (18)
                                                    #-834    (18)   #-843    (18)   #-851    (18)   #-857    (18)
                                                    #-866    (18)   #-874    (18)   #-883    (18)
CDB$M_UBA                   =00000002     152  (2)
CDB$M_UBI                   =00000004     153  (2)
CDB$V_IE                    =00000005     150  (2)   #-1492   (31)     156    (2)
CDB$V_MBA                   =00000000     145  (2)     151    (2)
CDB$V_OFFSET                =00000003     148  (2)     154    (2)
CDB$V_REV                   =00000004     149  (2)     155    (2)
CDB$V_UBA                   =00000001     146  (2)     152    (2)
CDB$V_UBI                   =00000002     147  (2)   #-1359   (28)     153    (2)
CDB$W_RVR                    0000001A     179  (3)   #-1519   (31)
CDB$W_ST2                    00000018     178  (3)   #-1514   (31)     1517   (31)   #-1520   (31)   #-1524   (31)
CDBSIZ                      =00000020     181  (3)     184    (3)
CHC$_ABORT                  =00000002     132  (2)
CHC$_CLEAR                  =00000006     132  (2)
CHC$_CLRDFT                 =00000009     132  (2)
CHC$_DSINT                  =00000005     132  (2)
CHC$_ENINT                  =00000004     132  (2)
CHC$_INITA                  =00000000     132  (2)
CHC$_INITB                  =00000001     132  (2)
CHC$_PURGE                  =00000003     132  (2)
CHC$_SELF_TEST              =0000000B     132  (2)
CHC$_SETDFT                 =00000008     132  (2)
CHC$_STATUS                 =00000007     132  (2)
CHC$_STOP                   =0000000A     132  (2)
CHI$M_CHNINT                =00000001     132  (2)
CHI$M_DEVINT                =00000002     132  (2)   #-1520   (31)
CHI$M_IPL                   =0000007C     132  (2)
CHI$S_IPL                   =00000005     132  (2)   #-1516   (31)
CHI$V_CHNINT                =00000000     132  (2)
CHI$V_DEVINT                =00000001     132  (2)
CHI$V_IPL                   =00000002     132  (2)   #-1516   (31)
CHM$_FORWARD                =00000000     132  (2)     132    (2)
CHM$_INVALIDATE             =00000001     132  (2)     132    (2)
CHM$_MAP                    =00000002     132  (2)     132    (2)
CHM$_MFWDN                  =00000002     132  (2)
CHM$_MFWDNO                 =0000000A     132  (2)
CHM$_MFWDV                  =00000003     132  (2)
CHM$_MFWDVO                 =0000000B     132  (2)
CHM$_MREVN                  =00000006     132  (2)
CHM$_MREVNO                 =0000000E     132  (2)
CHM$_MREVV                  =00000007     132  (2)
CHM$_MREVVO                 =0000000F     132  (2)
CHM$_NFWDN                  =00000000     132  (2)
CHM$_NREVN                  =00000004     132  (2)
CHM$_OFFSET                 =00000008     132  (2)     132    (2)
```

```
CHM$_REVERSE                    =00000004     132     (2)       132     (2)
CHN$_ADR1                       =0000000C     133     (2)     #-427     (10)
CHN$_ADR2                       =00000010     133     (2)     #-429     (10)    #-574     (14)
CHN$_FUNC                       =00000008     133     (2)     #-274     (6)
CHN$_NARGS                      =00000004     133     (2)
CHN$_UNIT                       =00000004     133     (2)     #-267     (6)
CHN_ABORT                        0000004E-R   384     (9)       286     (6)
CHN_CALL_RTN                     000000B0-R   702     (17)    #-271     (6)     #-288     (6)     #-354     (8)     #-386     (9)
                                                               #-433     (10)    #-465     (11)    #-501     (12)    #-533     (13)
                                                               #-617     (15)    #-659     (16)
CHN_CLEAR                        00000084-R   531     (13)      286     (6)
CHN_CLRDFT                       000000A0-R   657     (16)      286     (6)
CHN_DSINT                        0000007F-R   499     (12)      286     (6)
CHN_DSP                          0000000F-R   273     (6)     #-269     (6)
CHN_ENINT                        00000057-R   421     (10)      286     (6)
CHN_INITA                        00000035-R   318     (7)       286     (6)
CHN_INITB                        00000041-R   351     (8)       286     (6)
CHN_NORMAL_RTN                   000000A9-R   699     (17)    #-321     (7)     #-575     (14)
CHN_PURGE                        00000076-R   463     (11)      286     (6)
CHN_SETDFT                       00000097-R   615     (15)      286     (6)
CHN_STATUS                       0000008D-R   572     (14)      286     (6)
CHS$M_BADBDP                    =20000000     132     (2)
CHS$M_BADPAR                    =40000000     132     (2)
CHS$M_BUSIC                     =00800000     132     (2)
CHS$M_BUSINIT                   =00400000     132     (2)
CHS$M_BUSNXM                    =08000000     132     (2)
CHS$M_BUSPDN                    =01000000     132     (2)
CHS$M_CHNDPE                    =00000040     132     (2)
CHS$M_CHNERR                    =00000002     132     (2)       132     (2)       160     (2)
CHS$M_CHNMPE                    =00000080     132     (2)
CHS$M_CHPFOT                    =00000100     132     (2)
CHS$M_DEVBUS                    =00000010     132     (2)
CHS$M_DEVERR                    =00000004     132     (2)       132     (2)       160     (2)
CHS$M_DEVTO                     =00000020     132     (2)
CHS$M_ERRANY                    =0000000F     160     (2)
CHS$M_MBAATN                    =00100000     132     (2)
CHS$M_MBACPE                    =00200000     132     (2)
CHS$M_MBADTB                    =00040000     132     (2)
CHS$M_MBADTC                    =00080000     132     (2)
CHS$M_MBAEXC                    =00010000     132     (2)
CHS$M_MBANED                    =00020000     132     (2)
CHS$M_MBAWCKLWR                 =02000000     132     (2)
CHS$M_MBAWCKUPR                 =04000000     132     (2)
CHS$M_PGMERR                    =00000008     132     (2)       132     (2)       160     (2)
CHS$M_PGMHDE                    =00000800     132     (2)
CHS$M_SYSERR                    =00000001     132     (2)       132     (2)       160     (2)
CHS$M_SYSMEM                    =00000200     132     (2)
CHS$M_SYSSBI                    =00000400     132     (2)
CHS$M_UIE                       =10000000     132     (2)
CHS$V_BADBDP                    =0000001D     132     (2)
CHS$V_BADPAR                    =0000001E     132     (2)
CHS$V_BUSIC                     =00000017     132     (2)
CHS$V_BUSINIT                   =00000016     132     (2)
CHS$V_BUSNXM                    =0000001B     132     (2)
CHS$V_BUSPDN                    =00000018     132     (2)
CHS$V_CHNDPE                    =00000006     132     (2)
CHS$V_CHNERR                    =00000001     132     (2)
```

```
CHS$V_CHNMPE                    =00000007     132     (2)
CHS$V_CHPFOT                    =00000008     132     (2)
CHS$V_DEVBUS                    =00000004     132     (2)
CHS$V_DEVERR                    =00000002     132     (2)
CHS$V_DEVTO                     =00000005     132     (2)
CHS$V_MBAATN                    =00000014     132     (2)
CHS$V_MBACPE                    =00000015     132     (2)
CHS$V_MBADTB                    =00000012     132     (2)
CHS$V_MBADTC                    =00000013     132     (2)
CHS$V_MBAEXC                    =00000010     132     (2)
CHS$V_MBANED                    =00000011     132     (2)
CHS$V_MBAWCKLWR                 =00000019     132     (2)
CHS$V_MBAWCKUPR                 =0000001A     132     (2)
CHS$V_PGMERR                    =00000003     132     (2)
CHS$V_PGMHDE                    =0000000B     132     (2)
CHS$V_SYSERR                    =00000000     132     (2)
CHS$V_SYSMEM                    =00000009     132     (2)
CHS$V_SYSSBI                    =0000000A     132     (2)
CHS$V_UIE                       =0000001C     132     (2)
CLRMAP                           0000021D-R   926     (18)    #-806     (18)    #-811     (18)    #-835     (18)    #-859     (18)
                                                               #-876     (18)
CLRVEC                           00000379-R   1450    (30)    #-500     (12)
CNTCHK                           0000024F-R   982     (18)    #-777     (18)
CNTCHK_RTN                       00000276-R   994     (18)    #-991     (18)
CVT$T_MBAMAP                     00000054-R   192     (3)
CVT$T_MBASR                      00000050-R   189     (3)
CVT$T_UBADPR                     00000058-R   195     (3)
CVT$T_UBAMAP                     0000005C-R   198     (3)
DS$CVTREG                        00000000-XR                  1206    (23)
DS$GPHARD                        00000000-XR                  1345    (28)
DS$K_ERROR                      =00000002     129     (2)
DS$K_NORMAL                     =00000001     129     (2)
DS$K_SEVERE                     =00000004     129     (2)
DS$K_SUBSYS                     =00000066     129     (2)     108     (2)     129     (2)
DS$K_WARNING                    =00000000     129     (2)
DS$PRINTB                        00000000-XR                  1142    (21)    1173    (22)    1211    (23)
DS$_AP_NORMAL_BREAK             =00660150     129     (2)
DS$_ARITH                       =006600D0     129     (2)
DS$_ASBE                        =00660118     129     (2)
DS$_BADLINK                     =006600F0     129     (2)
DS$_BADTYPE                     =006600E8     129     (2)
DS$_BIIC                        =00660120     129     (2)
DS$_CHME                        =006600A8     129     (2)
DS$_CHMK                        =006600E0     129     (2)
DS$_DEVNAME                     =00660108     129     (2)
DS$_ERROR                       =00660002     129     (2)
DS$_FHWE                        =00660068     129     (2)
DS$_FRAGBUF                     =00660080     129     (2)
DS$_ICBUSY                      =006600C8     129     (2)
DS$_ICERR                       =006600C0     129     (2)
DS$_IHWE                        =00660060     129     (2)
DS$_ILLCHAR                     =00660018     129     (2)
DS$_ILLPAGCNT                   =00660078     129     (2)
DS$_ILLUNIT                     =00660100     129     (2)
DS$_INIT_FAIL                   =00660140     129     (2)
DS$_INSFMEM                     =00660050     129     (2)
DS$_INVCPU                      =00660130     129     (2)
```

```
DS$_IPL2HI                      =006600B8      129     (2)
DS$_IVADDR                      =00660040      129     (2)
DS$_IVVECT                      =00660038      129     (2)
DS$_KRNLSTK                     =00660090      129     (2)
DS$_LOGIC                       =00660070      129     (2)
DS$_MCHK                        =00660088      129     (2)
DS$_MEM_ALLOC_ERR               =00660138      129     (2)
DS$_MMOFF                       =00660058      129     (2)
DS$_NEEDUNIT                    =006600F8      129     (2)
DS$_NODE                        =00660128      129     (2)
DS$_NOPCS                       =00660110      129     (2)
DS$_NORMAL                      =00660001      129     (2)    #-1005  (18)  #-1092  (20)  #-1143  (21)  #-1361  (28)
                                                              #-1409  (29)  #-700   (17)  #-966   (18)  #-989   (18)
DS$_NOSUPPORT                   =006600B1      129     (2)    #-320   (7)   #-353   (8)   #-385   (9)   #-464   (11)
                                                              #-532   (13)  #-616   (15)  #-658   (16)
DS$_NOTALLOWMP                  =00660148      129     (2)
DS$_NOTDON                      =00660030      129     (2)
DS$_NOTIMP                      =006600B0      129     (2)    #-1048  (19)    108   (2)     125   (2)     129   (2)
DS$_NULLSTR                     =00660010      129     (2)
DS$_OVERFLOW                    =00660008      129     (2)
DS$_POWER                       =00660098      129     (2)
DS$_PROGERR                     =00660020      129     (2)    #-287   (6)   #-425   (10)  #-801   (18)  #-962   (18)
                                                              #-992   (18)
DS$_SEVERE                      =00660004      129     (2)
DS$_TRANSL                      =006600A0      129     (2)
DS$_TRUNCATE                    =00660028      129     (2)
DS$_UNEXPINT                    =006600D8      129     (2)
DS$_UNSUPPDEV                   =00660158      129     (2)
DS$_VASFULL                     =00660048      129     (2)
DS$_WARNING                     =00660000      129     (2)      108   (2)     129   (2)
DSI$CHANNEL                      0000038C-R    1497    (31)
DSI$CHANNEL_COM                  0000038E-R    1499    (31)   #-1493  (31)
DSI$CHANNEL_VEC                  00000380-R    1491    (31)
DSP$SHOWMBA                      00000281-R    1047    (19)
DSP$SHOWUBI                      0000028B-R    1089    (20)
DSX$CHANNEL                      00000000-R    265     (6)
DSX$SETMAP                       000000B1-R    753     (18)
DSX$SHOWCHAN                     0000029C-R    1134    (21)
FMT_CRLF                         00000000-R    210     (4)      1142  (21)
FMT_CVT_ARG                      00000020-R    187     (3)    #-1204  (23)  #-1205  (23)    1206  (23)
FMT_UBI_CSR                      0000001F-R    212     (4)
FMT_UBI_HDR                      00000003-R    211     (4)      1173  (22)
GENSTS                           00000304-R    1270    (25)   #-1513  (31)
HDWSTS                           00000307-R    1300    (26)   #-1508  (31)
HP$A_DEVICE                      00000018                     #-1357  (28)  #-1506  (31)
HP$A_LINK                        00000020                     #-1351  (28)  #-1353  (28)
HP$W_VECTOR                      00000024                     #-1400  (29)
INT_DSP                          000003E1-R    1528    (31)
INVALID                          0000012B-R    800     (18)     799   (18)
INVALIDATE                       00000135-R    803     (18)     799   (18)
JMPADR                           000000E9-R    227     (5)
MBA_MAP_BIT                      0000003D-R    214     (4)      193   (3)
MBA_SR_BIT                       0000003D-R    213     (4)      190   (3)
MFWDN                            00000150-R    817     (18)     799   (18)
MFWDNO                           000001B6-R    865     (18)     799   (18)
MFWDV                            0000013E-R    808     (18)     799   (18)
MFWDVO                           000001A1-R    856     (18)     799   (18)
```

```
MREVN                           00000180-R   841    (18)     799    (18)
MREVNO                          000001DA-R   882    (18)     799    (18)
MREVV                           0000016B-R   832    (18)     799    (18)
MREVVO                          000001C8-R   873    (18)     799    (18)
NFWDN                           0000015F-R   825    (18)     799    (18)
NREVN                           00000192-R   849    (18)     799    (18)
PAR$M_ATDEF                    =00000010     130    (2)
PAR$M_ATHI                     =00000008     130    (2)
PAR$M_ATLO                     =00000004     130    (2)
PAR$M_NODEF                    =00000002     130    (2)
PAR$V_ATDEF                    =00000004     130    (2)
PAR$V_ATHI                     =00000003     130    (2)
PAR$V_ATLO                     =00000002     130    (2)
PAR$V_NODEF                    =00000001     130    (2)
PAR$_BIN                       =00000002     130    (2)
PAR$_DEC                       =0000000A     130    (2)
PAR$_HEX                       =00000010     130    (2)
PAR$_NO                        =00000000     130    (2)
PAR$_OCT                       =00000008     130    (2)
PAR$_YES                       =00000001     130    (2)
PR$_IPL                        =00000012                  #-1502   (31)
RNGCHK                          0000024B-R   974    (18)  #-761    (18)
SCB_BASE                        00000000-XR                 1403    (29)
SCB_UNKINT                      00000000-XR                 1404    (29)
SETCNT                          0000022E-R   951    (18)  #-814    (18)  #-822    (18)  #-829    (18)  #-838    (18)
                                                          #-846    (18)  #-853    (18)  #-862    (18)  #-870    (18)
                                                          #-879    (18)  #-887    (18)
SETMAP                          000001E9-R   899    (18)  #-812    (18)  #-820    (18)  #-836    (18)  #-844    (18)
                                                          #-860    (18)  #-868    (18)  #-877    (18)  #-885    (18)
SETVAR                          0000022D-R   941    (18)  #-813    (18)  #-821    (18)  #-828    (18)  #-837    (18)
                                                          #-845    (18)  #-852    (18)  #-861    (18)  #-869    (18)
                                                          #-878    (18)  #-886    (18)
SETVEC                          00000348-R   1396   (29)  #-422    (10)
SHOWBITS                        000002CE-R   1203   (23)
SHOW_UBI                        000002C0-R   1172   (22)  #-1091   (20)  #-1140   (21)
SHW$K_BUFSIZ                   =00000080     141    (2)     187    (3)     226    (5)
SHW$_NARGS                     =00000001     136    (2)
SHW$_UNIT                      =00000004     136    (2)   #-1136   (21)
SHWBUF                          00000068-R   226    (5)     1207   (23)    187    (3)
SIZ...                         =00000001     132    (2)     130    (2)     132    (2)
STM$_BAS                       =00000010     135    (2)   #-766    (18)
STM$_CNT                       =00000014     135    (2)   #-768    (18)  #-988    (18)
STM$_FUNC                      =00000008     135    (2)   #-781    (18)
STM$_NARGS                     =00000006     135    (2)
STM$_PHY                       =0000000C     135    (2)   #-760    (18)  #-767    (18)  #-984    (18)
STM$_PTH                       =00000018     135    (2)   #-765    (18)
STM$_UNIT                      =00000004     135    (2)   #-755    (18)
STM_CALL_RTN                    00000280-R   1007   (18)  #-758    (18)  #-763    (18)  #-775    (18)  #-779    (18)
                                                          #-802    (18)
STM_NORMAL_RTN                  00000279-R   1004   (18)  #-807    (18)  #-815    (18)  #-823    (18)  #-830    (18)
                                                          #-839    (18)  #-847    (18)  #-854    (18)  #-863    (18)
                                                          #-871    (18)  #-880    (18)  #-888    (18)
UBA$S_MAP_ADDR                 =00000015                  #-910    (18)
UBA$V_MAP_ADDR                 =00000000                  #-909    (18)
UBA_DPR_BIT                     0000003D-R   215    (4)     196    (3)
UBA_MAP_BIT                     0000003E-R   217    (4)     199    (3)
UBI$L_CSR                      =00000000     112    (2)
```

```
UBI$L_MAP                  =00000800     113    (2)    #-911    (18)   #-928    (18)
UBI$M_MAP_BO               =02000000     117    (2)    #-907    (18)
UBI$M_MAP_VALID            =80000000     115    (2)    #-904    (18)
UBI$S_MAP_BO               =00000001     118    (2)
UBI$S_MAP_DPD              =00000002     120    (2)
UBI$S_MAP_PFN              =000000F      122    (2)
UBI$V_MAP_BO               =00000019     116    (2)      117    (2)
UBI$V_MAP_DPD              =00000015     119    (2)
UBI$V_MAP_PFN              =00000000     121    (2)
UBI$V_MAP_VALID            =000001F      114    (2)      115    (2)
UBIVEC                      000000ED-R   228    (5)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CHNDEF	1	91 (2)	133 (2)
$D1_PRINT_S	1	1142 (21)	1142 (21) 1173 (22)
$DEF	1	132 (2)	
$DEFINI	1	109 (2)	109 (2) 131 (2) 134 (2)
$DS_BITDEF	2	128 (2)	128 (2)
$DS_CHCDEF	1	132 (2)	132 (2)
$DS_CHDEF	1	132 (2)	132 (2)
$DS_CHIDEF	1	132 (2)	132 (2)
$DS_CHMDEF	2	132 (2)	132 (2)
$DS_CHSDEF	3	132 (2)	132 (2)
$DS_CVTREG_G	1	1206 (23)	1206 (23)
$DS_CVTREG_L	1	187 (3)	187 (3)
$DS_DSDEF	3	129 (2)	108 (2) 129 (2)
$DS_GPHARD_S	1	1345 (28)	1345 (28)
$DS_HPODEF	2	131 (2)	131 (2)
$DS_PARDEF	1	130 (2)	130 (2)
$DS_PRINTB_S	1	1142 (21)	1142 (21) 1173 (22)
$EQU	1	132 (2)	108 (2) 128 (2) 129 (2) 130 (2) 132 (2)
$EQULS1	1	132 (2)	108 (2) 129 (2) 132 (2)
$EQULST	1	108 (2)	108 (2) 129 (2) 132 (2)
$GBLINI	2	108 (2)	108 (2) 128 (2) 129 (2) 130 (2) 132 (2)
$OFFDEF	1	133 (2)	133 (2) 135 (2) 136 (2)
$PRDEF	5	134 (2)	134 (2)
$SHWDEF	1	97 (2)	136 (2)
$STMDEF	1	94 (2)	135 (2)
$UBADEF	6	109 (2)	109 (2)
$VIELD	1	130 (2)	130 (2) 132 (2)
$VIELD1	1	132 (2)	130 (2) 132 (2)
CASE	1	275 (6)	275 (6) 782 (18)
DSBINT	1	1502 (31)	1502 (31)
MODNAM	1	204 (3)	204 (3)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL3	00C1	963 (18)
AOBLEQ	00F3	929 (18)
ASHL	0078	902 (18)
BBCS	00E3	1359 (28)
BBS	00E0	1492 (31)
BEQL	0013	1352 (28) 906 (18)
BGTR	0014	965 (18)
BICL	00CA	1349 (28) 1451 (30) 804 (18) 809 (18) 818 (18) 826 (18) 833 (18)
		842 (18) 850 (18) 857 (18) 866 (18)
BICL3	00CB	1209 (23)
BICW3	00AB	1519 (31)
BISL	00C8	1401 (29) 1511 (31) 834 (18) 843 (18) 851 (18) 858 (18) 867 (18)
		874 (18) 883 (18) 904 (18) 907 (18)
BISW	00A8	1520 (31)
BISW2	00A8	985 (18)
BITL	00D3	905 (18)
BLBC	00E9	1138 (21) 1347 (28) 423 (10)
BLBS	00E8	269 (6) 757 (18) 762 (18) 774 (18) 778 (18)
BLEQ	0015	428 (10) 430 (10) 991 (18)
BRB	0011	1354 (28) 321 (7) 386 (9) 433 (10) 465 (11) 501 (12) 533 (13)
		575 (14) 617 (15) 659 (16)
BRW	0031	271 (6) 288 (6) 354 (8) 758 (18) 763 (18) 775 (18) 779 (18)
		802 (18) 807 (18) 815 (18) 823 (18) 830 (18) 839 (18) 847 (18)
		854 (18) 863 (18) 871 (18) 880 (18) 888 (18)
BSBW	0030	1091 (20) 1137 (21) 1140 (21) 1508 (31) 1510 (31) 1513 (31) 268 (6)
		422 (10) 500 (12) 756 (18) 761 (18) 773 (18) 777 (18) 806 (18)
		811 (18) 812 (18) 813 (18) 814 (18) 820 (18) 821 (18) 822 (18)
		828 (18) 829 (18) 835 (18) 836 (18) 837 (18) 838 (18) 844 (18)
		845 (18) 846 (18) 852 (18) 853 (18) 859 (18) 860 (18) 861 (18)
		862 (18) 868 (18) 869 (18) 870 (18) 876 (18) 877 (18) 878 (18)
		879 (18) 885 (18) 886 (18) 887 (18)
CALLG	00FA	1206 (23)
CALLS	00FB	1142 (21) 1173 (22) 1211 (23) 1345 (28)
CASEL	00CF	799 (18)
CASEW	00AF	286 (6)
CLRL	00D4	1241 (24) 1271 (25) 1301 (26) 903 (18) 927 (18) 928 (18)
CLRQ	007C	1498 (31) 573 (14)
CMPL	00D1	964 (18) 990 (18)
EXTZV	00EF	769 (18) 771 (18)
INCL	00D6	431 (10) 912 (18) 913 (18) 987 (18)
INSV	00F0	1516 (31) 909 (18)
MFPR	00DB	1502 (31)
MOVAB	009E	1404 (29) 770 (18)
MOVAL	00DE	1348 (28) 1407 (29) 1504 (31)
MOVAQ	007E	1403 (29)
MOVL	00D0	1005 (18) 1048 (19) 1090 (20) 1092 (20) 1136 (21) 1143 (21) 1204 (23)
		1205 (23) 1346 (28) 1353 (28) 1356 (28) 1357 (28) 1361 (28) 1399 (29)
		1409 (29) 1505 (31) 1506 (31) 1509 (31) 1512 (31) 1522 (31) 1523 (31)
		1524 (31) 1526 (31) 267 (6) 274 (6) 287 (6) 320 (7) 353 (8)
		385 (9) 425 (10) 427 (10) 429 (10) 464 (11) 532 (13) 616 (15)

```
                    658    (16)   700    (17)   755    (18)   760    (18)   765    (18)   766    (18)   767    (18)
                    768    (18)   781    (18)   801    (18)   900    (18)   911    (18)   962    (18)   966    (18)
                    975    (18)   988    (18)   989    (18)   992    (18)
MOVQ          007D  574    (14)   984    (18)
MOVW          00B0  1514   (31)
MOVZWL        003C  1400   (29)   1405   (29)
MTPR          00DA  1502   (31)   319    (7)    352    (8)
POPR          00BA  1410   (29)   1529   (31)   915    (18)   995    (18)
PUSHAB        009F  1142   (21)   1173   (22)   1210   (23)
PUSHAL        00DF  1207   (23)   1345   (28)
PUSHL         00DD  1208   (23)   1345   (28)
PUSHR         00BB  1397   (29)   1500   (31)   901    (18)   983    (18)
RET           0004  1008   (18)   1049   (19)   1093   (20)   1145   (21)   703    (17)
RSB           0005  1174   (22)   1212   (23)   1242   (24)   1272   (25)   1302   (26)   1364   (28)   1411   (29)
                    1452   (30)   1494   (31)   1531   (31)   917    (18)   930    (18)   942    (18)   952    (18)
                    968    (18)   976    (18)   996    (18)
SOBGTR        00F5  1408   (29)   914    (18)
SUBL          00C2  772    (18)
SUBL2         00C2  986    (18)
TSTL          00D5  1351   (28)   1530   (31)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	190	(3)	193	(3)	196	(3)	199	(3)								
.ALIGN	1490	(31)	1496	(31)												
.ASCIC	204	(3)	210	(4)	211	(4)	212	(4)	216	(4)	218	(4)				
.BLKB	184	(3)	226	(5)												
.BLKL	172	(3)	173	(3)	174	(3)	175	(3)	176	(3)	177	(3)	180	(3)	227	(5)
	228	(5)														
.BLKW	178	(3)	179	(3)												
.END	1534	(31)														
.ENDC	108	(2)	1142	(21)	1173	(22)	128	(2)	129	(2)	130	(2)	132	(2)	1502	(31)
.ENDM	108	(2)	109	(2)	1142	(21)	1206	(23)	128	(2)	129	(2)	130	(2)	131	(2)
	132	(2)	133	(2)	134	(2)	1345	(28)	135	(2)	136	(2)	1502	(31)	187	(3)
	204	(3)	275	(6)	93	(2)	96	(2)	99	(2)						
.ENDR	108	(2)	1142	(21)	1173	(22)	129	(2)	130	(2)	132	(2)	286	(6)	799	(18)
.ENTRY	1047	(19)	1089	(20)	1134	(21)	265	(6)	753	(18)						
.IDENT	6	(1)														
.IF	108	(2)	1142	(21)	1173	(22)	128	(2)	129	(2)	130	(2)	132	(2)	1502	(31)
.IFF	108	(2)	128	(2)	129	(2)	130	(2)	132	(2)	1502	(31)				
.IIF	108	(2)	128	(2)	129	(2)	130	(2)	132	(2)						
.IRP	108	(2)	1142	(21)	1173	(22)	129	(2)	130	(2)	132	(2)	133	(2)	135	(2)
	136	(2)	286	(6)	799	(18)										
.LIBRARY	83	(2)	84	(2)	85	(2)										
.LIST	107	(2)	133	(2)	135	(2)	136	(2)	138	(2)	139	(2)	8	(1)		
.LONG	187	(3)														
.MACRO	108	(2)	128	(2)	129	(2)	130	(2)	132	(2)	91	(2)	94	(2)	97	(2)
.MDELETE	108	(2)	128	(2)	129	(2)	130	(2)	132	(2)						
.NLIST	105	(2)	106	(2)	133	(2)	135	(2)	136	(2)	137	(2)	7	(1)		
.NOCROSS	109	(2)	131	(2)	134	(2)										
.PAGE	1489	(31)	264	(6)	752	(18)	890	(18)	919	(18)						
.PSECT	163	(3)	170	(3)	206	(4)	220	(5)	231	(6)						
.RESTORE	109	(2)	131	(2)	134	(2)	182	(3)								
.SAVE	109	(2)	131	(2)	134	(2)	169	(3)								
.SBTTL	1010	(19)	1051	(20)	1095	(21)	1147	(22)	1176	(23)	1214	(24)	1244	(25)	1274	(26)
	1305	(28)	1366	(29)	1413	(30)	1454	(31)	230	(6)	290	(7)	323	(8)	356	(9)
	388	(10)	435	(11)	467	(12)	503	(13)	535	(14)	577	(15)	619	(16)	661	(17)
	705	(18)	78	(2)												
.SIGNED_WORD	286	(6)	799	(18)												
.TITLE	5	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	16	#-1090 (20) #-1136 (21) #-267 (6) #-274 (6) #-427 (10) #-429 (10)
		#-574 (14) #-755 (18) #-760 (18) #-765 (18) #-766 (18) #-767 (18)
		#-768 (18) #-781 (18) #-984 (18) #-988 (18)
R0	57	#-1005 (18) #-1048 (19) #-1092 (20) #-1138 (21) #-1143 (21) #-1241 (24)
		#-1271 (25) #-1301 (26) #-1347 (28) #-1361 (28) #-1405 (29) #-1408 (29)
		#-1409 (29) #-1500 (31) #-1509 (31) #-1511 (31) #-1512 (31) #-1514 (31)
		#-1522 (31) #-1523 (31) #-1524 (31) #-1529 (31) #-269 (6) #-287 (6)
		#-320 (7) #-353 (8) #-385 (9) #-423 (10) #-425 (10) #-431 (10)
		#-464 (11) #-532 (13) #-616 (15) #-658 (16) #-700 (17) #-757 (18)
		#-762 (18) #-769 (18) #-772 (18) #-774 (18) #-778 (18) #-781 (18)
		#-799 (18) #-801 (18) #-903 (18) #-904 (18) #-907 (18) 910 (18)
		#-911 (18) #-927 (18) #-928 (18) #-929 (18) #-962 (18) #-966 (18)
		#-975 (18) #-989 (18) #-992 (18)
R1	17	#-1399 (29) #-1400 (29) #-1403 (29) #-1407 (29) #-1500 (31) #-1509 (31)
		#-1511 (31) #-1529 (31) #-900 (18) #-911 (18) #-913 (18) #-963 (18)
		#-964 (18) #-983 (18) #-988 (18) #-990 (18) #-995 (18)
R10	3	753 (18) #-768 (18) 770 (18)
R11	1	753 (18)
R2	24	1047 (19) 1089 (20) 1134 (21) #-1136 (21) 1210 (23) #-1345 (28)
		#-1397 (29) #-1404 (29) 1407 (29) #-1410 (29) 265 (6) #-267 (6)
		#-274 (6) #-286 (6) 753 (18) #-755 (18) #-760 (18) #-766 (18)
		#-900 (18) #-963 (18) #-983 (18) #-984 (18) #-986 (18) #-995 (18)
R3	23	1047 (19) 1089 (20) 1134 (21) #-1204 (23) #-1208 (23) #-1209 (23)
		265 (6) 753 (18) #-767 (18) 769 (18) 770 (18) #-901 (18)
		#-902 (18) #-909 (18) #-912 (18) #-915 (18) #-983 (18) #-985 (18)
		#-986 (18) #-987 (18) #-990 (18) #-995 (18)
R4	6	1047 (19) 1089 (20) 1134 (21) #-1205 (23) 265 (6) 753 (18)
R5	48	1047 (19) 1089 (20) 1134 (21) #-1348 (28) #-1349 (28) #-1356 (28)
		1359 (28) #-1399 (29) #-1400 (29) #-1401 (29) #-1451 (30) #-1500 (31)
		#-1504 (31) #-1505 (31) #-1512 (31) #-1514 (31) 1517 (31) #-1519 (31)
		#-1520 (31) #-1522 (31) #-1523 (31) #-1524 (31) #-1526 (31) #-1529 (31)
		265 (6) #-427 (10) #-429 (10) #-573 (14) #-574 (14) 753 (18)
		#-765 (18) #-805 (18) #-810 (18) #-819 (18) #-827 (18) #-833 (18)
		#-834 (18) #-842 (18) #-843 (18) #-850 (18) #-851 (18) #-857 (18)
		#-858 (18) #-866 (18) #-867 (18) #-875 (18) #-884 (18) #-905 (18)
R6	20	1047 (19) 1089 (20) #-1090 (20) 1134 (21) #-1346 (28) #-1351 (28)
		#-1353 (28) #-1356 (28) #-1357 (28) #-1500 (31) #-1505 (31) #-1506 (31)
		#-1529 (31) 265 (6) 753 (18) #-911 (18) #-928 (18)
R7	1	753 (18)
R8	7	753 (18) #-770 (18) 771 (18) #-772 (18) #-914 (18) #-963 (18)
R9	1	753 (18)
SP	9	#-1209 (23) 1345 (28) #-1346 (28) #-1498 (31) #-1502 (31) #-1516 (31)
		#-1519 (31) #-1526 (31) #-1530 (31)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.19
Command processing	882	00:00:00.24	00:00:00.87
Pass 1	793	00:00:02.54	00:00:04.45
Symbol table sort	0	00:00:00.15	00:00:00.15
Pass 2	47	00:00:00.86	00:00:01.68
Symbol table output	2	00:00:00.04	00:00:00.23
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	9	00:00:00.35	00:00:00.58
Assembler run totals	1758	00:00:04.22	00:00:08.16

The working set limit was 2512 pages.
133288 bytes (261 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 558 non-local and 18 local symbols.
1534 source lines were read in Pass 1, producing 60 object records in Pass 2.
95 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	16
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	2
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	26

657 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CHAN730.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W
```

Table of contents

(1)   67   DECLARATIONS
(2)   100   CHANGE MODE TO KERNEL ROUTINE
(3)   182   ASTEXIT SYSTEM SERVICE
(3)   208   Local intermediate dispatches

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CHMK.MAR
                        0000     2 ; *5    12-MAY-1986 11:09:03 BARANSKI "Cleaning up Object Libraries."
                        0000     3 ; *4     2-APR-1986 08:30:27 SHELHAMER "<no reason given>"
                        0000     4 ; *3     1-APR-1986 13:46:12 SHELHAMER "<no reason given>"
                        0000     5 ; *2     1-APR-1986 12:55:42 SHELHAMER "<no reason given>"
                        0000     6 ; *1     6-FEB-1986 15:14:49 DS ""
                        0000     7 ; DEC/CMS REPLACEMENT HISTORY, Element CHMK.MAR
                        0000     8         .TITLE  CHMK    *** CHMK Change mode to kernel handler
                        0000     9         .IDENT  /10-09/                                                 ;G:2
                        0000    10         .LIST   MEB
                        0000    11         .NLIST  CND
                        0000    12
                        0000    13 ;
                        0000    14 ; COPYRIGHT (c) 1977, 1981, 1985, 1986                               ;G:2
                        0000    15 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000    16 ;
                        0000    17 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    18 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    19 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    20 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    21 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    22 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    23 ; REMAIN IN DEC.
                        0000    24 ;
                        0000    25 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    26 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    27 ; CORPORATION.
                        0000    28 ;
                        0000    29 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    30 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    31
                        0000    32 ;++
                        0000    33 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
                        0000    34 ;
                        0000    35 ; ABSTRACT:
                        0000    36 ;
                        0000    37 ; ENVIRONMENT:
                        0000    38 ;
                        0000    39 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
                        0000    40 ;
                        0000    41 ; MODIFIED BY:
                        0000    42 ;               TOM SOUTTER     29-MAR-78       VERSION 02 (ESSAA-4.01)
                        0000    43 ;       01      ADDED CHMK HANDLER FOR ASTEXIT
                        0000    44 ;               D Butenhof      16-jan-80
                        0000    45 ;       02      Add dispatch for SGIPR (Store/Get Int. Proc. Reg.)
                        0000    46 ;
                        0000    47 ;               Roger Riggs,    12-Mar-1980
                        0000    48 ;       03      Removed CMK$MMON, CMK$MMOFF, added CMK$MMENABLE
                        0000    49 ;               Dave Butenhof, 23-feb-1981, version 6.3
                        0000    50 ;       04      Delete SEP psect, fix trunc. errors
                        0000    51 ;       05      Make dispatch offsets .SIGNED_WORD to report signed
                        0000    52 ;               truncation errors
                        0000    53 ;               - Dave Butenhof, 08-Jun-1981, version 6.4
                        0000    54 ;       06      Add dispatch for CHK$_SETPRT
                        0000    55 ;
                        0000    56 ;       07      - Jack Stansbury, 22-Oct-1981, Version 6.-
                        0000    57 ;               Added .LIBRARY statements for $DS and $DIAG.
```

```
                        0000     58 ;                  Fixed a truncation error.
                        0000     59 ;          08      Jim Shelhamer   Jan 30, 1985    Version 8.1
                        0000     60 ;                  Added CMK$APBOOT dispatch for booting AP.
                        0000     61 ;                                                                  ;G:2
                        0000     62 ;          09      Jim Shelhamer   March 1, 1986   Version 10.0    ;G:2
                        0000     63 ;                  Add AP_CMKCODE for AP to use with user vector ;G:2
                        0000     64 ;                  and CHMK instruction.                         ;G:2
                        0000     65 ;--
```

```
                     0000     67          .SBTTL  DECLARATIONS
                     0000     68 ;
                     0000     69 ; INCLUDE FILES:
                     0000     70 ;
                     0000     71          .LIBRARY        /$DS/           ;                                       [07]
                     0000     72          .LIBRARY        /$DIAG/         ;                                       [07]
                     0000     73
                     0000     74 ;
                     0000     75 ; MACROS:
                     0000     76 ;
                     0000     77
                     0000     78 ;
                     0000     79 ; EQUATED SYMBOLS:
                     0000     80 ;
                     0000     81          APDEF                                                       ;[09] ;G:4
                     0000     82          $PR8SSDEF                                                   ;[09] ;G:4
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .RESTORE
                     0000     83          CMKDEF
                     0000     84          DSFDEF
                     0000     88          $PSLDEF
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .RESTORE
                     0000     89          $PCBDEF
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004             0000                      .BLKL   1
                     0004              .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008             0004                      .BLKL   1
                     0008              .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A             0008                      .BLKW   1
                     000A              .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B             000A                      .BLKB   1
                     000B              .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C             000B                      .BLKB   1
                     000C              .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D             000C                      .BLKB   1
                     000D              .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E             000D                      .BLKB   1
                     000E              .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010             000E                      .BLKW   1
                     0010              .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014             0010                      .BLKL   1
                     0014              .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018             0014                      .BLKL   1
                     0018              .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C             0018                      .BLKL   1
                     001C              .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020             001C                      .BLKL   1
                     0020              .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024             0020                      .BLKL   1
                     0024              .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028             0024                      .BLKL   1
                     0028              .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C             0028                      .BLKL   1
                     002C              .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E             002C                      .BLKW   1
```

```
                    002E                .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                                  .BLKB   1
                    002F                .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                                  .BLKB   1
                    0030                .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                                  .BLKW   1
                    0032                .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                                  .BLKW   1
                    0034                .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                                  .BLKW   1
                    0036                .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                                  .BLKW   1
                    0038                .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                                  .BLKW   1
                    003A                .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                                  .BLKW   1
                    003C                .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                                  .BLKW   1
                    003E                .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                                  .BLKW   1
                    0040                .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                                  .BLKW   1
                    0042                .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                                  .BLKW   1
                    0044                .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                                  .BLKB   8
                    004C                .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                    004C                .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                                  .BLKL   1
                    0050                .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                                  .BLKL   1
                    0054                .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                                  .BLKL   1
                    0058                .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                                  .BLKL   1
                    005C                .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                                  .BLKL   1
                    0060                .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                                  .BLKL   1
                    0064                .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                                  .BLKL   1
                    0068                .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                                  .BLKB   16
                    0078                .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                                  .BLKL   1
                    007C                .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084  007C                                  .BLKQ   1
                    0084                .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                                  .BLKL   1
                    0088                .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                    0088                .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                                  .BLKW   1
                    008A                .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                                  .BLKW   1
                    008C                .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                    008C                .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                    0000                .RESTORE
```

```
                        0000     93
                        0000     94 ;
                        0000     95 ; OWN STORAGE:
                        0000     96 ;
                 00000000       97          .PSECT  DATA, NOEXE, SHR, NOWRT, LONG
                        0000     98          MODNAM  CHMK
           4B 4D 48 43 00' 0000        $MODULE:        .ASCIC  \CHMK\
                        04  0000
```

```
0005    100          .SBTTL  CHANGE MODE TO KERNEL ROUTINE
0005    101 ;++
0005    102 ; FUNCTIONAL DESCRIPTION:
0005    103 ;
0005    104 ;       THIS ROUTINE DISPATCHES TO THE PROPER PRIVILEGED ROUTINE.
0005    105 ;
0005    106 ; CALLING SEQUENCE:
0005    107 ;
0005    108 ;       NONE
0005    109 ;
0005    110 ; INPUT PARAMETERS:
0005    111 ;
0005    112 ;       NONE
0005    113 ;
0005    114 ; IMPLICIT INPUTS:
0005    115 ;
0005    116 ;       NONE
0005    117 ;
0005    118 ; OUTPUT PARAMETERS:
0005    119 ;
0005    120 ;       NONE
0005    121 ;
0005    122 ; IMPLICIT OUTPUTS:
0005    123 ;
0005    124 ;       NONE
0005    125 ;
0005    126 ; COMPLETION CODES:
0005    127 ;
0005    128 ;       NONE
0005    129 ;
0005    130 ; SIDE EFFECTS:
0005    131 ;
0005    132 ;       NONE
0005    133 ;
0005    134 ;--
```

```
                                00000000          .PSECT  WORK, NOEXE, NOSHR, WRT, LONG
                                    0000
                          00000000  0000  CMKCODE:        .LONG   0                   ; SAVED CODE FOR USER VECTOR.
                          00000000  0004  AP_CMKCODE:     .LONG   0                   ; Saved code for user vector on AP. [09] ;G:
                                    0008
                                00000000          .PSECT  CODE, EXE, SHR, NOWRT, LONG
                                    0000          .ALIGN  LONG,0
                                    0000  EXE_CMODKRNL::
                                    0000          BR_IF_AP 5$                         ; Use AP_CMKCODE if on AP       [09]  ;G:2
      12 00000000'EF   08   E1  0000          BBC     #AP$V_AP_Running,AP$GW_Data,30000$
         7E   0000005E 8F   DB  0008          MFPR    #PR8SS$_BINID,-(SP)                 ; Get our node id
   00000000'EF   04   04   ED  000F          CMPZV   #AP$V_AP_Node_Id,#AP$S_AP_Node_Id, - ; Our node = AP?
                      8E       0017                  AP$GW_Data, (SP)+
                      09   13  0018          BEQL    5$                                  ; Yes, branch
                               001A  30000$:                                             ; No, continue
         00000000'EF   6E   D0  001A          MOVL    (SP), CMKCODE                ; Save code                                ;G:2
                      07   11  0021          BRB     7$                           ; Go do dispatch                   [09]  ;G:2
         00000004'EF   6E   D0  0023  5$:     MOVL    (SP), AP_CMKCODE             ; Save code                        [09]  ;G:2
                               002A                                                                                          ;G:2
             0D   00   8E   CF  002A  7$:     CASEL   (SP)+, #0, #CMK$_LAST-1 ; DISPATCH.                                   ;G:2
                         008E' 002E  10$:             .SIGNED_WORD    CMK$ASTEXIT-10$
                         FFD2' 0030                   .SIGNED_WORD    CMK$RCONSOLE-10$
                         FFD2' 0032                   .SIGNED_WORD    CMK$TCONSOLE-10$
                         00AB' 0034                   .SIGNED_WORD    MMENABLE-10$
                         00C3' 0036                   .SIGNED_WORD    APBOOT-10$       ; AP boot routine          [08]
                         FFD2' 0038                   .SIGNED_WORD    CMK$REFRESH-10$
                         001C' 003A                   .SIGNED_WORD    20$-10$
                         FFD2' 003C                   .SIGNED_WORD    CMK$HIBER-10$
                         00B1' 003E                   .SIGNED_WORD    INITSCB-10$
                         00B7' 0040                   .SIGNED_WORD    SETIPL-10$
                         FFD2' 0042                   .SIGNED_WORD    CMK$SGIPR-10$
                         FFD2' 0044                   .SIGNED_WORD    CMK$CANTIM-10$  ; Cancel timer kernel mode rtn
                         FFD2' 0046                   .SIGNED_WORD    CMK$SETIMR-10$  ; Setimr kernel mode routine
                         00BD' 0048                   .SIGNED_WORD    SETPRT-10$       ; Set page protection
                               004A
            00000000'EF   D5  004A  20$:     TSTL    L^DS$GA_CHMKVEC              ; Check for soft vector
                      3C   12  0050          BNEQ    30$
                               0052          BR_IF_AP 25$                         ; AP uses AP_CMKCODE               [09]  ;G:2
      12 00000000'EF   08   E1  0052          BBC     #AP$V_AP_Running,AP$GW_Data,30001$
         7E   0000005E 8F   DB  005A          MFPR    #PR8SS$_BINID,-(SP)                 ; Get our node id
   00000000'EF   04   04   ED  0061          CMPZV   #AP$V_AP_Node_Id,#AP$S_AP_Node_Id, - ; Our node = AP?
                      8E       0069                  AP$GW_Data, (SP)+
                      09   13  006A          BEQL    25$                                 ; Yes, branch
                               006C  30001$:                                             ; No, continue
         50   00000000'EF   D0  006C          MOVL    CMKCODE,R0                   ; Get code so ERRSUP prints it
                      07   11  0073          BRB     27$                          ; Go do errsup                     [09]  ;G:2
         50   00000004'EF   D0  0075  25$:     MOVL    AP_CMKCODE,R0                ; Get ap's code for errsup         [09]  ;G:2
                               007C                                                                                          ;G:2
                               007C  27$:     ERRSUP_S                                                                       ;G:2
            00000000'EF   DF  007C                  PUSHAL  $MODULE
                      00   DD  0082                  PUSHL   #0
                      01   DD  0084                  PUSHL   #$ER
         00000000'EF   03   FB  0086                  CALLS   #3, DS_ERRSUP
                      02       008D          REI
                               008E
                               008E  30$:     BR_IF_AP 40$                         ; AP uses AP_CMKCODE               [09]  ;G:2
      12 00000000'EF   08   E1  008E          BBC     #AP$V_AP_Running,AP$GW_Data,30002$
```

```
        7E   0000005E 8F   DB  0096                MFPR    #PR8SS$_BINID,-(SP)                ; Get our node id
  00000000'EF    04   04   ED  009D                CMPZV   #AP$V_AP_Node_Id,#AP$S_AP_Node_Id, - ; Our node = AP?
                      8E       00A5                        AP$GW_Data, (SP)+
                      08   13  00A6                BEQL    40$                                ; Yes, branch
                               00A8        30002$:                                            ; No, continue
            00000000'EF   DD  00A8   176           PUSHL   CMKCODE                  ; RESTORE CODE ON STACK.                ;G:2
                      06   11  00AE   177           BRB     50$                      ; Go do jump                  [09]  ;G:2
            00000004'EF   DD  00B0   178 40$:      PUSHL   AP_CMKCODE               ; Get ap's code onto stack    [09]  ;G:2
                               00B6   179                                                                                  ;G:2
            00000000'FF   17  00B6   180 50$:      JMP     @L^DS$GA_CHMKVEC         ; Go through user vector                ;G:2
```

```
                                00BC   182          .SBTTL  ASTEXIT SYSTEM SERVICE
                                00BC   183 ;+
                                00BC   184 ; CMK$ASTEXIT - SERVICE TO EXIT AN ACTIVE AST AND ALLOW PENDING ASTS TO
                                00BC   185 ;               BE DELIVERED.
                                00BC   186 ;
                                00BC   187 ; THIS SYSTEM SERVICE IS INVOKED WITH A "CMK ASTEXIT" NOT CONTAINED IN
                                00BC   188 ; A STANDARD SYSTEM SERVICE VECTOR TO AVOID CLUTTERING THE STACK WITH AN
                                00BC   189 ; ADDITIONAL CALL FRAME DURING AST EXIT PROCESSING.
                                00BC   190 ;
                                00BC   191 ; INPUTS:
                                00BC   192 ;       NONE
                                00BC   193 ;
                                00BC   194 ; OUTPUTS:
                                00BC   195 ;       PCB$B_ASTACT IS CLEARED FOR THE ISSUING MODE
                                00BC   196 ;
                                00BC   197 ;-
                                00BC   198
                                00BC   199 CMK$ASTEXIT::                                ;EXIT ACTIVE AST
 50   04 AE   02   16   EF   00BC   200          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,4(SP),R0 ;GET PREVIOUS MODE
                    14   BB   00C2   201          PUSHR   #^M<R2,R4>                      ;SAVE R2,R4
      54   00000000'EF   DE   00C4   202          MOVAL   DS$AX_SOFTPCB,R4                ;GET PCB CURRENT PCB ADDRESS
         00 0C A4   50   E7   00CB   203          BBCCI   R0,PCB$B_ASTACT(R4),10$ ;CLEAR AST ACTIVE BIT FOR MODE
           00000000'EF   16   00D0   204 10$:     JSB     SCH$NEWLVL                      ;COMPUTE NEW AST LEVEL SETTING          [07]
                    14   BA   00D6   205          POPR    #^M<R2,R4>                      ;RESTORE REGISTERS<R2,R4>
                         02   00D8   206          REI                                     ;AND EXIT
```

```
                        00D9   208          .SBTTL  Local intermediate dispatches
                        00D9   209 ;+
                        00D9   210 ; Since CASE offsets have to be word relative, to fix truncation errors,
                        00D9   211 ; dispatch to local JMP L^ way-stations.
                        00D9   212 ;-
                        00D9   213
                        00D9   214 MMENABLE:
00000000'EF   17        00D9   215          JMP     L^CMK$MMENABLE          ; Dispatch memory management on
                        00DF   216
                        00DF   217 INITSCB:
00000000'EF   17        00DF   218          JMP     L^CMK$INITSCB           ; Dispatch initialize SCB
                        00E5   219
00000000'EF   17        00E5   220 SETIPL: JMP     L^CMK$SETIPL            ; Dispatch set IPL
                        00EB   221
                        00EB   222 SETPRT:
00000000'EF   17        00EB   223          JMP     L^CMK$SETPRT            ; Dispatch the trap
                        00F1   224 APBOOT:                                                          ;[08
00000000'EF   17        00F1   225          JMP     L^CMK$APBOOT            ; Dispatch AP boot      [08]
                        00F7   226
                        00F7   227          .END
```

```
$ER                             = 00000002    D
$MODULE                           00000000 R  D   02
AP$GK_PROMPT_WAIT               = 00FFFFFF    D
AP$GW_DATA                        ********  X   04
AP$M_AP_NODE_ID                 = 000000F0    D
AP$M_AP_RUNNING                 = 00000100    D
AP$M_BUFFER_FULL                = 00000800    D
AP$M_DATA_READY                 = 00000200    D
AP$M_PP_NODE_ID                 = 0000000F    D
AP$M_REQUEST                    = 00000400    D
AP$S_AP_NODE_ID                 = 00000004    D
AP$S_PP_NODE_ID                 = 00000004    D
AP$V_AP_NODE_ID                 = 00000004    D
AP$V_AP_RUNNING                 = 00000008    D
AP$V_BUFFER_FULL                = 0000000B    D
AP$V_DATA_READY                 = 00000009    D
AP$V_PP_NODE_ID                 = 00000000    D
AP$V_REQUEST                    = 0000000A    D
AP$_REQ_CONSRX                  = 00000000  G D
AP$_REQ_RXCDVEC                 = 00000001  G D
APBOOT                            000000F1 R  D   04
AP_CMKCODE                        00000004 R  D   03
CMK$APBOOT                        ********  X   04
CMK$ASTEXIT                       000000BC RG D   04
CMK$CANTIM                        ********  X   04
CMK$HIBER                         ********  X   04
CMK$INITSCB                       ********  X   04
CMK$MMENABLE                      ********  X   04
CMK$RCONSOLE                      ********  X   04
CMK$REFRESH                       ********  X   04
CMK$SETIMR                        ********  X   04
CMK$SETIPL                        ********  X   04
CMK$SETPRT                        ********  X   04
CMK$SGIPR                         ********  X   04
CMK$TCONSOLE                      ********  X   04
CMK$_APBOOT                     = 00000004    D
CMK$_ASTEXIT                    = 00000000    D
CMK$_CANTIM                     = 0000000B    D
CMK$_HIBER                      = 00000007    D
CMK$_INITSCB                    = 00000008    D
CMK$_LAST                       = 0000000E    D
CMK$_MMENABLE                   = 00000003    D
CMK$_RCONSOLE                   = 00000001    D
CMK$_REFRESH                    = 00000005    D
CMK$_SCHDWK                     = 00000006    D
CMK$_SETIMR                     = 0000000C    D
CMK$_SETIPL                     = 00000009    D
CMK$_SETPRT                     = 0000000D    D
CMK$_SGIPR                      = 0000000A    D
CMK$_TCONSOLE                   = 00000002    D
CMKCODE                           00000000 R  D   03
DS$AX_SOFTPCB                     ********  X   04
DS$GA_CHMKVEC                     ********  X   04
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_NI                         = 04000000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DS_ERRSUP                         ********  X   04
EXE_CMODKRNL                      00000000 RG D   04
INITSCB                           000000DF R  D   04
```

```
MMENABLE                        000000D9 R  D  04
PCB$B_ASTACT                    0000000C    D
PCB$B_ASTEN                     0000000D    D
PCB$B_PRI                       0000000B    D
PCB$B_PRIB                      0000002F    D
PCB$B_TYPE                      0000000A    D
PCB$B_WEFC                      0000002E    D
PCB$C_LENGTH                    0000008C    D
PCB$K_LENGTH                    0000008C    D
PCB$L_ARB                       00000084    D
PCB$L_ASTQBL                    00000014    D
PCB$L_ASTQFL                    00000010    D
PCB$L_EFC2P                     00000058    D
PCB$L_EFC3P                     0000005C    D
PCB$L_EFCS                      00000050    D
PCB$L_EFCU                      00000054    D
PCB$L_EFWM                      0000004C    D
PCB$L_JIB                       00000078    D
PCB$L_OWNER                     0000001C    D
PCB$L_PHD                       00000064    D
PCB$L_PHYPCB                    00000018    D
PCB$L_PID                       00000060    D
PCB$L_PQB                       0000004C    D
PCB$L_SQBL                      00000004    D
PCB$L_SQFL                      00000000    D
PCB$L_STS                       00000024    D
PCB$L_UIC                       00000088    D
PCB$L_WSSWP                     00000020    D
PCB$L_WTIME                     00000028    D
PCB$Q_PRIV                      0000007C    D
PCB$T_LNAME                     00000068    D
PCB$T_TERMINAL                  00000044    D
PCB$W_APTCNT                    00000030    D
PCB$W_ASTCNT                    00000038    D
PCB$W_BIOCNT                    0000003A    D
PCB$W_BIOLM                     0000003C    D
PCB$W_DIOCNT                    0000003E    D
PCB$W_DIOLM                     00000040    D
PCB$W_GPGCNT                    00000034    D
PCB$W_GRP                       0000008A    D
PCB$W_MEM                       00000088    D
PCB$W_MTXCNT                    0000000E    D
PCB$W_PPGCNT                    00000036    D
PCB$W_PRCCNT                    00000042    D
PCB$W_SIZE                      00000008    D
PCB$W_STATE                     0000002C    D
PCB$W_TMBU                      00000032    D
PR0SS$_BINID                =   0000005E    D
PSL$S_PRVMOD                =   00000002    D
PSL$V_PRVMOD                =   00000016    D
SCH$NEWLVL                      ********  X    04
SETIPL                          000000E5 R  D  04
SETPRT                          000000EB R  D  04
SIZ...                      =   00000001    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000008C (140.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000005 (5.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
WORK	00000008 (8.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	000000F7 (247.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000002	172 (3)	#-172 (3)
$MODULE	00000000-R	98 (1)	172 (3)
AP$GK_PROMPT_WAIT	=0FFFFFFF	81 (1)	
AP$GW_DATA	00000000-XR		144 (3) 167 (3) 175 (3)
AP$M_AP_NODE_ID	=000000F0	81 (1)	
AP$M_AP_RUNNING	=00000100	81 (1)	
AP$M_BUFFER_FULL	=00000800	81 (1)	
AP$M_DATA_READY	=00000200	81 (1)	
AP$M_PP_NODE_ID	=0000000F	81 (1)	
AP$M_REQUEST	=00000400	81 (1)	
AP$S_AP_NODE_ID	=00000004	81 (1)	#-144 (3) #-167 (3) #-175 (3)
AP$S_PP_NODE_ID	=00000004	81 (1)	
AP$V_AP_NODE_ID	=00000004	81 (1)	#-144 (3) #-167 (3) #-175 (3)
AP$V_AP_RUNNING	=00000008	81 (1)	#-144 (3) #-167 (3) #-175 (3)
AP$V_BUFFER_FULL	=0000000B	81 (1)	
AP$V_DATA_READY	=00000009	81 (1)	
AP$V_PP_NODE_ID	=00000000	81 (1)	
AP$V_REQUEST	=0000000A	81 (1)	
AP$_REQ_CONSRX	=00000000	81 (1)	
AP$_REQ_RXCDVEC	=00000001	81 (1)	
APBOOT	000000F1-R	224 (3)	154 (3)
AP_CMKCODE	00000004-R	139 (3)	#-147 (3) #-170 (3) #-178 (3)
BIT...	=00000018	84 (1)	81 (1) 83 (1) 84 (1)
CMK$APBOOT	00000000-XR		225 (3)
CMK$ASTEXIT	000000BC-R	199 (3)	150 (3)
CMK$CANTIM	00000000-XR		161 (3)
CMK$HIBER	00000000-XR		157 (3)
CMK$INITSCB	00000000-XR		218 (3)
CMK$MMENABLE	00000000-XR		215 (3)
CMK$RCONSOLE	00000000-XR		151 (3)
CMK$REFRESH	00000000-XR		155 (3)
CMK$SETIMR	00000000-XR		162 (3)
CMK$SETIPL	00000000-XR		220 (3)
CMK$SETPRT	00000000-XR		223 (3)
CMK$SGIPR	00000000-XR		160 (3)
CMK$TCONSOLE	00000000-XR		152 (3)
CMK$_APBOOT	=00000004	83 (1)	
CMK$_ASTEXIT	=00000000	83 (1)	
CMK$_CANTIM	=0000000B	83 (1)	
CMK$_HIBER	=00000007	83 (1)	
CMK$_INITSCB	=00000008	83 (1)	
CMK$_LAST	=0000000E	83 (1)	#-149 (3)
CMK$_MMENABLE	=00000003	83 (1)	
CMK$_RCONSOLE	=00000001	83 (1)	
CMK$_REFRESH	=00000005	83 (1)	
CMK$_SCHDWK	=00000006	83 (1)	
CMK$_SETIMR	=0000000C	83 (1)	
CMK$_SETIPL	=00000009	83 (1)	
CMK$_SETPRT	=0000000D	83 (1)	
CMK$_SGIPR	=0000000A	83 (1)	

```
CMK$_TCONSOLE                   =00000002      83      (1)
CMKCODE                          00000000-R    138     (3)     #-145    (3)     #-168    (3)     #-176    (3)
DS$AX_SOFTPCB                    00000000-XR                     202     (3)
DS$GA_CHMKVEC                    00000000-XR                   #-165     (3)       180     (3)
DS$M_ABRTFLG                    =00000040      84      (1)
DS$M_BADTIME                    =00100000      84      (1)
DS$M_BATCH                      =00400000      84      (1)
DS$M_BRKCLR                     =00001000      84      (1)
DS$M_BRKPT                      =00000800      84      (1)
DS$M_CHARFLG                    =00000100      84      (1)
DS$M_CMDFLG                     =00000080      84      (1)
DS$M_CTRLC                      =00000001      84      (1)
DS$M_CTRLO                      =00010000      84      (1)
DS$M_DEVFLG                     =00000200      84      (1)
DS$M_DISABLCC                   =01000000      84      (1)
DS$M_DONFLG                     =00002000      84      (1)
DS$M_ERRFLG                     =00000010      84      (1)
DS$M_EXCEPT                     =00080000      84      (1)
DS$M_EXETST                     =00040000      84      (1)
DS$M_HLTFLG                     =00000008      84      (1)
DS$M_LODFLG                     =00000002      84      (1)
DS$M_MEMMGT                     =00008000      84      (1)
DS$M_NI                         =04000000      84      (1)
DS$M_OUTPUT                     =00800000      84      (1)
DS$M_RUBFLG                     =00000020      84      (1)
DS$M_SCRIPT                     =00200000      84      (1)
DS$M_SETIMR                     =02000000      84      (1)
DS$M_STRFLG                     =00000004      84      (1)
DS$M_SUBT                       =00004000      84      (1)
DS$M_SYSFLG                     =00000400      84      (1)
DS$M_TIMED                      =02000000      84      (1)
DS$M_TIMED_OUT                  =08000000      84      (1)
DS$M_TIMRON                     =00020000      84      (1)
DS$V_ABRTFLG                    =00000006      84      (1)
DS$V_BADTIME                    =00000014      84      (1)
DS$V_BATCH                      =00000016      84      (1)
DS$V_BRKCLR                     =0000000C      84      (1)
DS$V_BRKPT                      =0000000B      84      (1)
DS$V_CHARFLG                    =00000008      84      (1)
DS$V_CMDFLG                     =00000007      84      (1)
DS$V_CTRLC                      =00000000      84      (1)
DS$V_CTRLO                      =00000010      84      (1)
DS$V_DEVFLG                     =00000009      84      (1)
DS$V_DISABLCC                   =00000018      84      (1)
DS$V_DONFLG                     =0000000D      84      (1)
DS$V_ERRFLG                     =00000004      84      (1)
DS$V_EXCEPT                     =00000013      84      (1)
DS$V_EXETST                     =00000012      84      (1)
DS$V_HLTFLG                     =00000003      84      (1)
DS$V_LODFLG                     =00000001      84      (1)
DS$V_MEMMGT                     =0000000F      84      (1)
DS$V_NI                         =0000001A      84      (1)
DS$V_OUTPUT                     =00000017      84      (1)
DS$V_RUBFLG                     =00000005      84      (1)
DS$V_SCRIPT                     =00000015      84      (1)
DS$V_SETIMR                     =00000019      84      (1)
DS$V_STRFLG                     =00000002      84      (1)
```

```
DS$V_SUBT                       =0000000E      84      (1)
DS$V_SYSFLG                     =0000000A      84      (1)
DS$V_TIMED                      =00000019      84      (1)
DS$V_TIMED_OUT                  =0000001B      84      (1)
DS$V_TIMRON                     =00000011      84      (1)
DS_ERRSUP                        00000000-XR                   172     (3)
EXE_CMODKRNL                     00000000-R    143     (3)
INITSCB                          000000DF-R    217     (3)       158     (3)
MMENABLE                         000000D9-R    214     (3)       153     (3)
PCB$B_ASTACT                     0000000C                        203     (3)
PR0SS$_BINID                    =0000005E                      #-144     (3)     #-167     (3)     #-175     (3)
PSL$S_PRVMOD                    =00000002                      #-200     (3)
PSL$V_IS                        =0000001A      83      (1)
PSL$V_PRVMOD                    =00000016                      #-200     (3)
SCH$NEWLVL                       00000000-XR                   204     (3)
SETIPL                           000000E5-R    220     (3)       159     (3)
SETPRT                           000000EB-R    222     (3)       163     (3)
SIZ...                          =00000001      84      (1)       81      (1)       84      (1)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO                          SIZE        DEFINITION      REFERENCES...
-----                          ----        ----------      -------------
$DEF                           1           84     (1)
$DEFINI                        1           82     (1)      82     (1)      88     (1)      89     (1)
$EQU                           1           84     (1)      83     (1)
$EQULS1                        1           83     (1)      83     (1)
$EQULST                        1           83     (1)      83     (1)
$GBLINI                        2           81     (1)      81     (1)      83     (1)      84     (1)
$PCBDEF                        4           89     (1)      89     (1)
$PR8SSDEF                      5           82     (1)      82     (1)
$PSLDEF                        2           88     (1)      88     (1)
$PUSHADR                       1           172    (3)      172    (3)
$VIELD                         1           81     (1)      81     (1)      84     (1)
$VIELD1                        1           84     (1)      81     (1)      84     (1)
APDEF                          2           81     (1)      81     (1)
BR_IF_AP                       1           144    (3)      144    (3)      167    (3)      175    (3)
BR_IF_NOT_AP_RUNNING           1           144    (3)      144    (3)      167    (3)      175    (3)
CMKDEF                         1           83     (1)      83     (1)
DSFDEF                         3           84     (1)      84     (1)
ERRSUP_S                       1           172    (3)      172    (3)
MODNAM                         1           98     (1)      98     (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...											
BBC	00E1	144	(3)	167	(3)	175	(3)						
BBCCI	00E7	203	(3)										
BEQL	0013	144	(3)	167	(3)	175	(3)						
BNEQ	0012	166	(3)										
BRB	0011	146	(3)	169	(3)	177	(3)						
CALLS	00FB	172	(3)										
CASEL	00CF	149	(3)										
CMPZV	00ED	144	(3)	167	(3)	175	(3)						
EXTZV	00EF	200	(3)										
JMP	0017	180	(3)	215	(3)	218	(3)	220	(3)	223	(3)	225	(3)
JSB	0016	204	(3)										
MFPR	00DB	144	(3)	167	(3)	175	(3)						
MOVAL	00DE	202	(3)										
MOVL	00D0	145	(3)	147	(3)	168	(3)	170	(3)				
POPR	00BA	205	(3)										
PUSHAL	00DF	172	(3)										
PUSHL	00DD	172	(3)	176	(3)	178	(3)						
PUSHR	00BB	201	(3)										
REI	0002	173	(3)	206	(3)								
TSTL	00D5	165	(3)										

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ALIGN          142     (3)
.ASCIC          98      (1)
.END            227     (3)
.ENDC           172     (3)     81      (1)     83      (1)     84      (1)
.ENDM           144     (3)     172     (3)     81      (1)     82      (1)     83      (1)     84      (1)     88      (1)     89      (1)
                98      (1)
.ENDR           81      (1)     83      (1)     84      (1)
.IDENT          9       (1)
.IF             172     (3)     81      (1)     83      (1)     84      (1)
.IFF            172     (3)     81      (1)     83      (1)     84      (1)
.IIF            172     (3)     81      (1)     83      (1)     84      (1)
.IRP            81      (1)     83      (1)     84      (1)
.LIBRARY        71      (1)     72      (1)
.LIST           10      (1)     87      (1)     91      (1)     92      (1)
.LONG           138     (3)     139     (3)
.MACRO          81      (1)     83      (1)     84      (1)
.MDELETE        81      (1)     83      (1)
.NLIST          11      (1)     85      (1)     86      (1)     90      (1)
.NOCROSS        82      (1)     88      (1)     89      (1)
.PAGE           181     (3)     207     (3)     66      (1)
.PSECT          136     (3)     141     (3)     97      (1)
.RESTORE        82      (1)     88      (1)     89      (1)
.SAVE           82      (1)     88      (1)     89      (1)
.SBTTL          100     (2)     182     (3)     208     (3)     67      (1)
.SIGNED_WORD    150     (3)     151     (3)     152     (3)     153     (3)     154     (3)     155     (3)     156     (3)     157     (3)
                158     (3)     159     (3)     160     (3)     161     (3)     162     (3)     163     (3)
.TITLE          8       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	4	#-168	(3)	#-170	(3)	#-200	(3)	#-203	(3)				
R2	2	#-201	(3)	#-205	(3)								
R4	4	#-201	(3)	#-202	(3)	203	(3)	#-205	(3)				
SP	10	#-144	(3)	#-145	(3)	#-147	(3)	#-149	(3)	#-167	(3)	#-175	(3)
		200	(3)										

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.01	00:00:00.19
Command processing	892	00:00:00.25	00:00:03.06
Pass 1	568	00:00:01.27	00:00:02.92
Symbol table sort	0	00:00:00.08	00:00:00.08
Pass 2	23	00:00:00.29	00:00:00.50
Symbol table output	2	00:00:00.04	00:00:00.08
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	6	00:00:00.14	00:00:00.27
Assembler run totals	1519	00:00:02.09	00:00:07.11

The working set limit was 2400 pages.
67430 bytes (132 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 321 non-local and 13 local symbols.
227 source lines were read in Pass 1, producing 34 object records in Pass 2.
44 pages of virtual memory were used to define 18 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	7
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	16

436 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CHMK.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DI
```

Table of contents


```
(1)      546        Libraries, Macros, Equated Symbols
(1)      591        Work Psect Storage
(1)      624        Data Psect Storage
(1)      632        Error & Status Messages Text
(1)      693        COMMANDS A-C
(1)      748        DIRECTORY, DEATTACH, DESELECT
(1)      799        DEPOSIT, EXIT, EXAMINE
(1)      836        LOAD, NEXT, RUN, HELP, INITPCS
(1)      874        SET, SELECT, SHOW, START, SUMMARY, XDELTA, a
(1)      927        END OF LINE & ILLEGAL Processing
(1)      975        INCOMPLETE, BAD LIST & BAD QUAL Processing
(1)     1026        Flag list
(1)     1082        Device Names,  Event Flags
(1)     1125        File Specifications
(1)     1226        ADDRESS Parrameter.
(1)     1281        Start and Run Qualifiers
(1)     1344        BREAKPOINT Start or Run Qualifiers, DEVICE Qualifiers.
(1)     1390        EXAMINE & DEPOSIT Qualifiers
(1)     1445        DEATTACH, DESELECT, and SELECT Parameters.
(1)     1501        Show Parameters and Qualifiers
(1)     1719        Clear Parameters and Qualifiers
(1)     1782        Set Parameters and Qualifiers
(1)     1952        Set Default Parameters
(1)     2011        DS$Cli Routine - Command Line Interpreter
(1)     2175        CLI Action Routines.
(1)     2356        Null and Context-Saving Action Routines
(1)     2392        Success, Failure, Notnuf, Enuf Action Routines
(1)     2423        Commands A-C Action Routines
(1)     2496        Commands D-E Action Routines
(1)     2588        Commands H-R Action Routines
(1)     2667        Commands S-Z Action Routines
(1)     2897        CRD command
(1)     2982        Filespec, Optional, Required Action Routines
(1)     3015        Start and Run Qualifier Action Routines
(1)     3128        Event and Flags Action Routines
(1)     3197        Base, Address, Break Action Routines
(1)     3241        Base_Qual, Nobase Action Routines                                    ;[107]
(1)     3263        All, Default Action Routines
(1)     3303        Show Calls Routine                          ;       [70]
(1)     3315        CRDTrace Action Routines                    ;       [65]
(1)     3359        Ask_prompt, Bell, Binary, Halt, IEx Action Routines
(1)     3425        Loop, Oper, Prompt Quick Action Routines
(1)     3459        Search, Trace, Verify Action Routines
(1)     3485        Set/Show Page, Width Action Routine
(1)     3521        Set/Show QA Action Routines
(1)     3569        QADef Action Routine
(1)     3595        Efn, Next, Ascii Action Routines
(1)     3632        Register Action Routines
(1)     3672        Backup, Advance, Data Action Routines
(1)     3708        Long, Word, Byte, Hex, Dec, Oct Action Routines
(1)     3758        IF Action Routines
(1)     3851        Xdelta Action Routines
(1)     3873        Device, Brief, Adaptor Action Routines
(1)     3926        Show Select, Do_Cmd, Set Load, Show Load Action Routines
(1)     3963        MM On, Off, and Show Action Routines
(1)     4031        Show Status, Show Support
```

```
                                0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CLI.MAR
                                0000     2 ; *28   12-OCT-1987 11:59:57 MARTIN "KA888 -> KA884 PERMENENTLY"
                                0000     3 ;  27A1 30-JUL-1987 17:10:41 MARTIN "FIX"
                                0000     4 ; *27    2-JUN-1987 09:41:32 MI "FIXED BUG IN FILE DIRECTORY SPECIFICATION"
                                0000     5 ; *26   29-APR-1987 16:05:50 MARTIN "MODIFIED FOR PSTAR"
                                0000     6 ; *25    5-SEP-1986 11:11:25 BARANSKI "<no reason given>"
                                0000     7 ; *24    5-SEP-1986 11:06:01 BARANSKI "TESTING UPDATE"
                                0000     8 ; *23    4-SEP-1986 18:08:14 BARANSKI "<no reason given>"
                                0000     9 ; *22    4-SEP-1986 18:02:50 BARANSKI "<no reason given>"
                                0000    10 ; *21    4-SEP-1986 18:00:25 BARANSKI "<no reason given>"
                                0000    11 ; *20    4-SEP-1986 17:58:30 BARANSKI "<no reason given>"
                                0000    12 ; *19    4-SEP-1986 17:52:55 BARANSKI "<no reason given>"
                                0000    13 ; *18    4-SEP-1986 17:49:05 BARANSKI "<no reason given>"
                                0000    14 ; *17    4-SEP-1986 17:45:36 BARANSKI "UPDATE TEST."
                                0000    15 ; *16   22-JUL-1986 10:12:42 MARTIN "FIXED SHO MM - SET MM ON/OFF CONTROVERSY"
                                0000    16 ; *15   10-JUN-1986 13:50:59 BAGGETT "<no reason given>"
                                0000    17 ; *14   26-MAY-1986 13:30:43 BARANSKI "Done fixing SET LOAD []"
                                0000    18 ; *13   20-MAY-1986 10:26:37 BERGAZZI "/TIME"
                                0000    19 ; *12   19-MAY-1986 16:51:03 BARANSKI "TESTING UPDATE"
                                0000    20 ; *11   19-MAY-1986 15:20:14 BERGAZZI "ADD SUPPPORT FOR /TIME"
                                0000    21 ; *10   12-MAY-1986 11:14:21 BARANSKI "Cleaning up Object Libraries."
                                0000    22 ; *9    18-APR-1986 15:37:07 SHELHAMER "<no reason given>"
                                0000    23 ; *8    14-APR-1986 12:42:47 SHELHAMER "<no reason given>"
                                0000    24 ; *7    14-APR-1986 12:24:08 SHELHAMER "<no reason given>"
                                0000    25 ; *6    14-APR-1986 11:58:49 SHELHAMER "<no reason given>"
                                0000    26 ; *5    14-APR-1986 11:19:58 ANDELLA "<no reason given>"
                                0000    27 ; *4    11-MAR-1986 14:03:00 BAGGETT "FINISHED"
                                0000    28 ; *3    10-MAR-1986 10:47:38 ANDELLA "<no reason given>"
                                0000    29 ; *2    14-FEB-1986 11:05:34 BERGAZZI "CRD ONLINE MESSAGE RESTORED"
                                0000    30 ; *1     6-FEB-1986 15:14:51 DS ""
                                0000    31 ; DEC/CMS REPLACEMENT HISTORY, Element CLI.MAR
                                0000    32         .Title  CLI     DIAGNOSTIC SUPERVISOR COMMAND LINE INTERPRETER
                                0000    33         .Ident  "10.10-28"                                              ;G:28
                                0000    34         .NoShow Conditionals
                                0000    35
                                0000    36 ;++
                                0000    37 ; Copyright (c) 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985,        ;G:26
                                0000    38 ;                 1986, 1987                                             ;G:26
                                0000    39 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                                0000    40 ;
                                0000    41 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                                0000    42 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                                0000    43 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                                0000    44 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                                0000    45 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                                0000    46 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                                0000    47 ; REMAIN IN DEC.
                                0000    48 ;
                                0000    49 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                                0000    50 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                                0000    51 ; CORPORATION.
                                0000    52 ;
                                0000    53 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                                0000    54 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                0000    55 ;
                                0000    56 ; NOTE:  The following Notations apply to the Comments concerning what commands
                                0000    57 ; are legal to be parsed:
```

```
0000     58 ;
0000     59 ;     1 space  = optional spaces
0000     60 ;     2 spaces = required spaces
0000     61 ;     WOrd     = abbreviatable to WO
0000     62 ;     word     = a parameter
0000     63 ;     [word]   = optional word
0000     64
0000     65 ;++
0000     66 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     67 ;
0000     68 ; ABSTRACT:
0000     69 ;
0000     70 ; ENVIRONMENT:
0000     71 ;
0000     72 ; AUTHOR:       KEN CHAPMAN     31-OCT-77       VERSION 01.
0000     73 ;
0000     74 ; MODIFIED BY:
0000     75 ;                NICK HOWGATE    06-NOV-77       VERSION 02
0000     76 ;       01       APT INTERFACE
0000     77 ;
0000     78 ;                TOM SOUTTER     17-NOV-77       VERSION 03.
0000     79 ;       02       DSPR #7 - FIX FOR SHOW EVENT FLAGS.
0000     80 ;       03       MAKE 'ILLEGAL COMMAND' MESSAGE GLOBAL.
0000     81 ;
0000     82 ;                KEN CHAPMAN     05-DEC-77       VERSION 04 (ESSAA-3.05)
0000     83 ;       04       DSPR #45 - REMOVE HALTI.
0000     84 ;
0000     85 ;                KEN CHAPMAN     12-JAN-78       VERSION 05 (ESSAA-3.06)
0000     86 ;       05       DSPR #77 - FIXED VERSION NUMBER BRANCH OF TREE.
0000     87 ;       06       DSPR #100 - FIXED HALT BRANCH OF TREE.
0000     88 ;
0000     89 ;                KEN CHAPMAN     16-FEB-78       VERSION 06 (ESSAA-3.07)
0000     90 ;       07       DSPR #104 - FIXED CLEAR BREAK BRANCH OF TREE.
0000     91 ;       08       ADDED 'COMPAT' FLAG.
0000     92 ;       09       CODE COMPRESSION USING CLI$_STRING MECHANISM.
0000     93 ;
0000     94 ;                KEN CHAPMAN     28-FEB-78       VERSION 07 (ESSAA-4.00)
0000     95 ;       10       SET COMMAND MODE FLAG ON ENTERING CLI.
```

```
0000  ;                NICK HOWGATE    29-MAR-78       VERSION 08 (ESSAA-4.01)
0000  ;        11      FIXED 'CONTINUE' IN APT MODE
0000  ;        12      ADDED PROPER SPELLING OF "HEXADECIMAL".
0000  ;        13      CHANGED 'VRRUN' & 'VRLOAD' TO 'DSV$RUN' & 'DSV$LOAD'
0000  ;                KEN CHAPMAN     06-APR-78
0000  ;        14      MADE 'COMPAT' COMMANDS CONDITIONAL ON 'VM11' DEFINED.
0000  ;
0000  ;                NICK HOWGATE    09-JUN-78       VERSION 09 (ESSAA-4.03).
0000  ;        15      FIX APT ^C PROBLEM
0000  ;
0000  ;                Roger Riggs     17-JUL-78       VERSION 10 (ESSAA-5.00).
0000  ;        16      Fix to let SUPERLINE do the prompt, clean up APT
0000  ;                stuff in area of prompt.
0000  ;
0000  ;                Roger Riggs     21-SEP-78
0000  ;        17      Insert support for new device selection commands.
0000  ;
0000  ;                Roger Riggs     04-FEB-79
0000  ;        18      Insert "SET LOAD <filespec>" command for default load device
0000  ;        22      Add SHOW LOAD command
0000  ;        23      Fix SET LOAD, SET LOOP, SET LOCK interaction
0000  ;                fix SELECT and DESELECT
0000  ;                Fix SE AMBIGIOUS error
0000  ;        24      Raise ASTLVL in command mode to prevent AST delivery
0000  ;                in command mode
0000  ;
0000  ;                Roger Riggs     4-Sept-1979
0000  ;        25      Made command tree and command action routine global for APT
0000  ;
0000  ;                Roger Riggs     6-Nov-1979
0000  ;        26      Changed version number on filespec to be decimal instead
0000  ;                of octal.
0000  ;                Changed MTPR x,PR$_IPL to call set so it's done in KERNEL mode
0000  ;
0000  ;        27      MARION BAGGETT  14-DEC-79
0000  ;                ALLOW SETTING OF MEMORY MANAGEMENT ON OR OFF
0000  ;                ALLOW EXAMINE AND DEPOSIT OF PROCESSOR REGISTERS
0000  ;
0000  ;        28      David Butenhof  09-Jan-80
0000  ;                Fix exam/deposit of processor registers
0000  ;
0000  ;        29      David Butenhof  06-feb-80
0000  ;                Be sure all possible AST levels are disabled to prevent
0000  ;                interrupt-driven diagnostic from continuing after ^C or
0000  ;                breakpoint.
0000  ;
0000  ;        30      David Butenhof  15-feb-80
0000  ;                Add SEARCH operator flag
0000  ;
0000  ;                Roger Riggs, 17-Jan-1980, Version 5.2
0000  ;        31      Corrected handling of command mode for APT, made sure
0000  ;                it typed "DS> " after ^C and processing of any command.
0000  ;        32      Fixed saved IPL to save all 5 bits of IPL instead of just 4.
0000  ;        33      Remove disable of ASTs in CLI so that WAITMS works and
0000  ;                ^C ASTs are delivered.
```

```
0000   153 ;                Dave Butenhof   30-jun-1980, version 5.5
0000   154 ;        34      Add HELP command in PARSE tree, and ACT_HELP action routine
0000   155 ;
0000   156 ;                Dave Butenhof, 02-oct-1980, Version 6.1
0000   157 ;        35      Add DIRECTORY command
0000   158 ;        36      Alter filespec parsing to recognise [000000] form as well as
0000   159 ;                [0,0] for UIC.
0000   160 ;
0000   161 ;                Dave Butenhof, 13-feb-1981, version 6.3
0000   162 ;        37      Change /SECTION:xxx so xxx can be symbol (inc. $ and _).
0000   163 ;        38      Add SET WIDTH command, and alter filespec parse to allow
0000   164 ;                subdirectories in user mode.
0000   165 ;
0000   166 ;        39      Dave Butenhof, 06-Oct-1981, version 6.5
0000   167 ;                Implement SHOW STATUS command
0000   168 ;
0000   169 ;        40      Jack Stansbury, 21-Oct-1981, Version 6.5
0000   170 ;                Added calls to QA$Main for the QA enhancement.
0000   171 ;                Also added .LIBRARY statements for $DIAG, $DS, and LIB.
0000   172 ;
0000   173 ;        41      Jack Stansbury, 22-Oct-1981, Version 6.5
0000   174 ;                Fixed truncation errors.
0000   175 ;
0000   176 ;        42      Jack Stansbury, 25-Oct-1981, Version 6.5
0000   177 ;                Many changes for the /QA qualifier on the
0000   178 ;                RUN and START commands, additional SET and SHOW
0000   179 ;                changes for the QA flags.
0000   180 ;
0000   181 ;        43      Jack Stansbury, 3-Nov-1981, Version 6.5
0000   182 ;                Major revisions to the parse tree. Renumbered all the labels,
0000   183 ;                moved things around, ... Also incorporated checks so that a
0000   184 ;                SET FLAG WIDTH 130 is not allowed (as it was before). Also
0000   185 ;                changed the error messages to mixed case. Also took out T_CRLF,
0000   186 ;                because it was not used. Most of these changes are not marked
0000   187 ;                since so much was done. Also made use of the new CLI$K_SLASH,
0000   188 ;                CLI$K_VALUE, and CLI$K_COMMA character values in calls to
0000   189 ;                $DS_CLI. These allow "/", ":" or "=", and ",", respectively,
0000   190 ;                with optional spaces on either side of them. They were added
0000   191 ;                to PARSE.MAR by Dave Butenhof. Also added /ADAPTER qualifier
0000   192 ;                to the SHOW DEVICE, SELECT, and DESELECT commands. This
0000   193 ;                qualifier
0000   194 ;                requires an adaptor name as its value. Also added SHOW DEFAULT
0000   195 ;                and SHOW BASE commands. Also added DCL-like printout of portion
0000   196 ;                of command line in error. That is,
0000   197 ;                        ?? error message
0000   198 ;                           \portion in error\
0000   199 ;
0000   200 ;        44      Marion Baggett, Jack Stansbury, 05-Oct-1981, Version 6.5
0000   201 ;                Enhanced the SHOW DEVICE AND SHOW SELECTED command to allow
0000   202 ;                the qualifier and print only the device type and name.
0000   203 ;                Added a new command SHOW SUPPORT to show the devices a diag
0000   204 ;                will support.
```

```
0000    206 ;       45      Jack Stansbury, 11-Nov-1981, Version 6.5
0000    207 ;               Changed the DS$PARSE call to DSX$SUPERPARSE. SUPERPARSE does not
0000    208 ;               return when it encounters EOL. Rather, it follows the
0000    209 ;               parse tree until it encounters CLI$K_EXIT. This fixes the
0000    210 ;               problem with SET BASE 0, where DS$PARSE saw EOL, and returned to
0000    211 ;               here with the NOTNUF flag still set. Thus, it was not continuing
0000    212 ;               in the parse tree, where it would have executed the ENUF action
0000    213 ;               routine, and then gone to EOL in the parse tree. DSX$SUPERPARSE
0000    214 ;               does continue, and thus will execute ENUF, and then go to EOL.
0000    215 ;               Also: changed the error handling and reporting. Established
0000    216 ;               different action routines for Illegal command versus Incomplete
0000    217 ;               commands. Checked return from DSX$PARSE and various CLI flags
0000    218 ;               and printed appropriate error messages.
0000    219 ;               Also: changed the DCL-like printout of the command line in error
0000    220 ;               to print the part of the line that was good. This was done
0000    221 ;               because of the strange results from doing it the previous way!
0000    222 ;               Also: added action routines ACT_OPTIONAL, ACT_REQUIRED,
0000    223 ;               ACT_IF_REQUIRED, ACT_IF_FILE_SPEC, ACT_BINARY, and ACT_VERIFY.
0000    224 ;               Also: added recognition of SET BINARY, SET VERIFY, CLEAR BINARY,
0000    225 ;               and CLEAR VERIFY. These are new control flags.
0000    226 ;               Also added ADVANCE action routine. It functions exactly
0000    227 ;               opposite of BACKUP.
0000    228 ;
0000    229 ;       46      Jack Stansbury, 13-Nov-1981, Version 6.5
0000    230 ;               Added CLI$K_CALL, CLI$K_RETURN, and CLI$K_BIFS to various places
0000    231 ;               in parse tree. Dave Butenhof added these new character codes.
0000    232 ;               They are to be used for one-level-deep subr calls. Also added
0000    233 ;               SUCCESS and FAILURE action routines to be used in with
0000    234 ;               the above. Also added a new DS command: EXIT.  Also altered
0000    235 ;               ACT_DO_CMD to avoid dispatching if CLI$L_COMMAND(R2) is 0; this
0000    236 ;               allows ATTACH to 'kill' the command when the /Adapter value is
0000    237 ;               bad. Also added the SHOW SECTIONS command.
0000    238 ;
0000    239 ;       47      Jack Stansbury, 17-Nov-1981, Version 6.5
0000    240 ;               Changed all BSBW's to JSB's.
0000    241 ;
0000    242 ;       48      Dave Butenhof, 18-Nov-1981, version 6.5
0000    243 ;               Add DEATTACH command, and typecodes in messages
0000    244 ;
0000    245 ;       49      Jack Stansbury, 21-Nov-1981, Version 6.5
0000    246 ;               Took out the DSQA constants and the calls to QA_MAIN.
0000    247 ;               It appears these will not be needed for the 6.5 DS version.
0000    248 ;
0000    249 ;       50      Dave Butenhof, 04-Feb-1982, Version 6.6
0000    250 ;               Use the new CLI$K_FILE parse code for filename/type parse
0000    251 ;
0000    252 ;       51      Dave Butenhof, 05-Feb-1982, Version 6.6
0000    253 ;               Add new command SHOW MEMORY.
```

```
                                0000    255 ;       52      Jack Stansbury, 18-Feb-1982, Version 6.6
                                0000    256 ;               Changed the HELP command parsing so that HSOW will not result
                                0000    257 ;               in the HELP command looking for DS command SOW! That is, if
                                0000    258 ;               spaces are not found after the minimum characters for HELP
                                0000    259 ;               (H in this case), then there must be an EOL. If not EOL, it
                                0000    260 ;               is an error. Also, I added the VSDP Supervisor command. This
                                0000    261 ;               is for CRD use. Right now, it simply prints "Command not
                                0000    262 ;               implemented".
                                0000    263 ;
                                0000    264 ;       53      Dave Butenhof, 09-Mar-1982, version 6.6
                                0000    265 ;               Change file parsing to accept subdirectories in standalone.
                                0000    266 ;
                                0000    267 ;       54      Dave Butenhof, 10-Mar-1982, version 6.6
                                0000    268 ;               Correct HELP parsing to accept 'H<CR>' as well as 'H '.
                                0000    269 ;
                                0000    270 ;       55      Jack Stansbury, 23-Mar-1982, Version 6.7
                                0000    271 ;               Added some macros (Br_If_Not_...) and took out some .LIST
                                0000    272 ;               statements.
                                0000    273 ;
                                0000    274 ;       56      Jack Stansbury, 10-Apr-1982, Version 6.7
                                0000    275 ;               Added support for running CRD-AutoTest on-line via the VSDP
                                0000    276 ;               CLI command.
                                0000    277 ;
                                0000    278 ;       57      Dave Butenhof, 13-Apr-1982, version 6.7
                                0000    279 ;               Add command to enable/disable generic name enforcement,
                                0000    280 ;               as backup. 'Set/Clear Enforce'
                                0000    281 ;
                                0000    282 ;       58      Jack Stansbury, 6-May-1982, Version 6.8
                                0000    283 ;               Added code in DS$Cli to not clear the ^C bit in the DS flags
                                0000    284 ;               longword if CRD-AutoTest was running. This is to allow ^C's
                                0000    285 ;               to be caught faster and easier by CRD.
                                0000    286 ;
                                0000    287 ;       59      Richard Brown, 19-May_82, Version 6.8
                                0000    288 ;               Addition of DEBUG command.  This involves placing the command
                                0000    289 ;               in the command line interpreter decision tree and adding an
                                0000    290 ;               action routine for the command.
                                0000    291 ;
                                0000    292 ;       60      Jack Stansbury, 24-May-1982, Version 6.8
                                0000    293 ;               Changed the "VSDP" command to "CRD".
                                0000    294 ;
                                0000    295 ;       61      Dave Butenhof, 25-May-1982, version 6.8
                                0000    296 ;               Restore poor SHOW MM command, which got kicked out by
                                0000    297 ;               SHOW MEMORY (note, however, that SHOW MEMORY did not
                                0000    298 ;               intend to do this, and should not be accused of bullying).
                                0000    299 ;
                                0000    300 ;       62      Richard Brown, 7-July-82, Version 6.8
                                0000    301 ;               Addition of /DEBUG qualifier to RUN and START commands.
                                0000    302 ;
                                0000    303 ;       63      Jack Stansbury, 15-July-1982, Version 6.9
                                0000    304 ;               Added code to call CRD Menu Test when the CRD command is typed.
```

```
                     0000   306 ;       64      Jack Stansbury, 24-July-1982, Version 6.9
                     0000   307 ;               Added code (commented out for now) that will check to see if the
                     0000   308 ;               CRD Menu Test command (CRD) is being called in user mode. If so,
                     0000   309 ;               it will print an error message. NOTE: this will have to be taken
                     0000   310 ;               out before the Supervisor is frozen for 6.9!!!!
                     0000   311 ;
                     0000   312 ;       65      Jack Stansbury, 8-September-1982, Version 6.9
                     0000   313 ;               Added three new commands: SET CRDTRACE, CLEAR CRDTRACE, and
                     0000   314 ;               SHOW CRDTRACE. These three all have to do with a new DSA bit
                     0000   315 ;               called DSA$V_CRD_Trace. This bit, when set, will cause the CRD
                     0000   316 ;               routines to print trace and debugging information for the user.
                     0000   317 ;
                     0000   318 ;       66      Jack Stansbury, 15-September-1982, Version 6.9
                     0000   319 ;               Added code in the ACT_CRD routine that clears the ^C handlers
                     0000   320 ;               and the ^C DS flag, and disable ^C's.
                     0000   321 ;
                     0000   322 ;       67      Jack Stansbury, 23-September-1982, Version 6.9
                     0000   323 ;               Changed the CRDTRACE qualifier to CRD/TRACE. The command are
                     0000   324 ;               now:
                     0000   325 ;               SET CRD/TRACE, CLEAR CRD/TRACE, and SHOW CRD.
                     0000   326 ;               Also changed the code
                     0000   327 ;               a bit at the activation of CRD - if CRD's initialization routine
                     0000   328 ;               returns failure, the DSR$Unload_CRD routine is now called.
                     0000   329 ;
                     0000   330 ;       68      Richard Brown, 27-Sep-82, Version 6.9
                     0000   331 ;               Changed default debugger name from DELTA to VDSDEBUG.
                     0000   332 ;
                     0000   333 ;       69      Jack Stansbury, March 3, 1983, Version 6.11
                     0000   334 ;               Added the SET CRD/DEBUG/TRACE command. This exact syntax
                     0000   335 ;               is required to minimize the number of people finding out
                     0000   336 ;               about and using this command. It turns on debugging mode in
                     0000   337 ;               CRD, and a LOT of output results!
                     0000   338 ;
                     0000   339 ;       70      Jack Stansbury, March 7, 1983, Version 6.11
                     0000   340 ;               Added the SHOW CALLS command. The routine is in the CRDImage
                     0000   341 ;               module.
                     0000   342 ;
                     0000   343 ;       71      Bob Bergazzi    28-March-1983   Version 6.11
                     0000   344 ;               Added the SET/SHOW PAGE command for video console terminals
                     0000   345 ;               in standalone mode. Routines are in PRINT.
                     0000   346 ;
                     0000   347 ;       72      John Ciukaj     4-April-1983    version 6.11
                     0000   348 ;               - Changed CRD Command Action routine (ACT_CRD) to
                     0000   349 ;                 allow online debugging of CRD AUTO Offline
                     0000   350 ;                 test package and offline initiation of
                     0000   351 ;                 CRD AUTO Offline by CRD Command.
                     0000   352 ;               - Changed references to CRD bits in DSA Flags to
                     0000   353 ;                 distinguish between online and offline.
                     0000   354 ;
                     0000   355 ;       73      Bob Bergazzi    8-Apr-1983      Version 6.11
                     0000   356 ;               Cleared Line_count when we put out a DS>, so that if we are SET
                     0000   357 ;               PAGEing, our first screen isn't wasted with stuff already seen.
                     0000   358 ;
                     0000   359 ;       74      Bob Bergazzi    May 16,1983     Version 6.11
                     0000   360 ;               Changed the order of the .LIB statements.
                     0000   361 ;
                     0000   362 ;       75      Bob Bergazzi    29-Aug-1983     Version 6.13
```

```
                0000   363 ;                Added the SET MEMORY command.
```

```
                                0000    365 ;       76      Bob Bergazzi    20-Sep-1983     Version 6.13
                                0000    366 ;               Added the /TIME switch to the RUN and START commands.
                                0000    367 ;               Commented out for release 13 and 14.
                                0000    368 ;
                                0000    369 ;       77      Bob Bergazzi    21-Oct-1983     Version 6.13
                                0000    370 ;               Added the INITPCS command.
                                0000    371 ;
                                0000    372 ;       78      Domenic Andella 8-Nov-83        Version 6.14
                                0000    373 ;               Added branches to tree at GET_FILE_SPEC, and
                                0000    374 ;               START_AND_RUN routines. These modifications
                                0000    375 ;               are to allow '$' and '_' to be included in the
                                0000    376 ;               directory, filename, and file type when using
                                0000    377 ;               a script filename, SET LOAD cmd, LOAD cmd, RUN,
                                0000    378 ;               and DIRECTORY cmd. These changes are necessary
                                0000    379 ;               for VDS to run under VMS V4.
                                0000    380 ;
                                0000    381 ;       79      RE MUSE         8 MAY 1984      Version 7.0
                                0000    382 ;               allow device names of the form 'node$ggan'
                                0000    383 ;
                                0000    384 ;       80      Bob Bergazzi    May 21, 1984    Version 7.0
                                0000    385 ;               Added /WIDE switch to the DIRECTORY command so
                                0000    386 ;               that filenames will be displayed one per line.
                                0000    387 ;               This fixes the problem of truncation of long filenames.
                                0000    388 ;
                                0000    389 ;       81      Ted Salem       Aug. 2, 1984    Version 7.1
                                0000    390 ;               Added SHOW TESTS command which will print the number
                                0000    391 ;               and title of each test of the current diag. program.
                                0000    392 ;
                                0000    393 ;       82      Ted Salem       Aug. 6, 1984    Version 7.1
                                0000    394 ;               Added /NOALLOCATE switch to the SELECT command so
                                0000    395 ;               that a device does not have to be allocated.
                                0000    396 ;
                                0000    397 ;       83      Jim Shelhamer   20 AUG 1984     Version 7.1
                                0000    398 ;               Corrected fuctioning of EXAMINE for address A or F by
                                0000    399 ;               eliminating a 'backup' of the input string.
                                0000    400 ;
                                0000    401 ;       84      M. Baggett      27-Sep-1984     Version 7.1
                                0000    402 ;               Temporary fix to allow CRD to run online.
                                0000    403 ;
                                0000    404 ;       85      Ron Parker      1-NOV-1984      Version 7.1
                                0000    405 ;               Changed GT_MENU ... message to elimate 11/730 reference
                                0000    406 ;
                                0000    407 ;       86      Ron Parker      6-NOV-1984
                                0000    408 ;               Cleaned up ACT_CRD routine and added Menutest_Online
                                0000    409 ;               function.
                                0000    410 ;
                                0000    411 ;       87      M. Baggett      14-Nov-1984
                                0000    412 ;               Allow generic name formats for form '$m$ggan'/.
                                0000    413 ;
                                0000    414 ;       88      Jim Shelhamer   21-Nov-1984     Version 8.0
                                0000    415 ;               Add Boot command for starting attached processor.
                                0000    416 ;               Only works for Scorpio, all others return error message.
                                0000    417 ;
                                0000    418 ;       89      Amy Iorio       29-Nov-1984     Version 8.0
                                0000    419 ;               Add SHOW ENFORCE command.
```

```
                                0000    421 ;       90      Bob Bergazzi    2-Jan-1985      Version 8.1
                                0000    422 ;               Fix conflict between SET PROMPT and SET PAGE.
                                0000    423 ;
                                0000    424 ;       91      Ted Salem       22-Jan-1985     Version 8.1
                                0000    425 ;               Fixed /noallocate bug.
                                0000    426 ;
                                0000    427 ;       92      Jim Shelhamer   Jan 30, 1985    Version 8.1
                                0000    428 ;               Modified Boot action routine to give error message in user-mode.
                                0000    429 ;               Also, added dispatch via CMK to CMK$APBOOT.
                                0000    430 ;
                                0000    431 ;       93      Jim Shelhamer   Feb 20, 1984    Version 8.1
                                0000    432 ;               Modified Boot action routine to save node id of Ap
                                0000    433 ;               to be booted.
                                0000    434 ;
                                0000    435 ;       94      M. Baggett      20-FEB-1985     Version 8.1
                                0000    436 ;               Added commands SET ULTRIX,SET ODS and change file parser
                                0000    437 ;               to recognize ULTRIX structure.
                                0000    438 ;
                                0000    439 ;       95      Bob Bergazzi    Feb. 26, 1985   Version 8.1
                                0000    440 ;               Enable edit [76] (/TIME).
                                0000    441 ;
                                0000    442 ;       96      Bob Bergazzi    Mar. 4, 1985    Version 8.1
                                0000    443 ;               Disable edit 95.
                                0000    444 ;
                                0000    445 ;       97      Ted Salem       Mar. 5, 1985    Version 8.1
                                0000    446 ;               Added DUMP command - for the NI only.
                                0000    447 ;
                                0000    448 ;       98      Bob Bergazzi    Mar. 12, 1985   Version 8.1
                                0000    449 ;               ACT_MMON does a MOVL into a byte field. Fix.
                                0000    450 ;
                                0000    451 ;       99      Domenic Andella 28-March-1985   Version 8.2
                                0000    452 ;               Add CLI 'next' flag to be set upon receipt
                                0000    453 ;               of NEXT command for use by MP.
                                0000    454 ;
                                0000    455 ;       100     Jim Shelhamer   Apr 10, 1985    Version 8.2
                                0000    456 ;               Moved saving of AP node number to after it has
                                0000    457 ;               been parsed.
                                0000    458 ;
                                0000    459 ;       101     Ted Salem       April 30, 1985  Version 8.2
                                0000    460 ;               Added SET CPU xxx command for MP systems.
                                0000    461 ;
                                0000    462 ;       102     M. Baggett      20-May-1985     Version 8.2
                                0000    463 ;               Correct ULTRIX load string spec.
                                0000    464 ;               Added ULTRIX flags for key commands.
                                0000    465 ;
                                0000    466 ;       103     Domenic Andella 22-May-1985     Version 8.2
                                0000    467 ;               Added clear mp condition flag in ACT_run,
                                0000    468 ;               start,next, and continue routines.
                                0000    469 ;
                                0000    470 ;       104     M. Baggett      23-May-1985     Version 8.2
                                0000    471 ;               Fixed ULTRIX parsing.
                                0000    472 ;
                                0000    473 ;       105     M. Baggett      29 -May-1985    Version 8.2
                                0000    474 ;               Fix bugs in RUN and START file parsing routines.
                                0000    475 ;
                                0000    476 ;       106     M. Baggett      4-Jun-1985      Version 8.3
                                0000    477 ;               Corrected ultrix parsing.
```

```
0000   478 ;
0000   479 ;       107     J. Baranski     21-Jun-1985     Version 8.3
0000   480 ;               Added /BASE and /NOBASE qualifiers to SET BREAKPOINTS,
0000   481 ;               CLEAR BREAKPOINTS, EXAMINE and DEPOSIT.
0000   482 ;
0000   483 ;       108     A. Iorio        18-July-1985    Version 8.3
0000   484 ;               Added SET ASK_PROMPT, CLEAR ASK_PROMPT, and ACT_ASK_PROMPT.
0000   485 ;
0000   486 ;       109     J. Baranski     26-Jun-1985     Version 8.3
0000   487 ;               General Comments & code Cleanup.
0000   488 ;
0000   489 ;       110     J. Baranski      1-Nov-1985     Version 9.1
0000   490 ;               Un Comment Out START/TIME.
0000   491 ;
0000   492 ;       111     Domenic Andella 15-Nov-1085     Version 9.2
0000   493 ;               Fixed some SET CPU bugs, modified ACT_CONTINUE to
0000   494 ;               prohibit printing of pc on an active mp system.
0000   495 ;
0000   496 ;       112     J. Baranski     21-Nov-1985     Version 9.1
0000   497 ;               Fix $DS_CLI CLI$K_FILE,... To accept $ and _.
0000   498 ;
0000   499 ;       113     Jim Shelhamer   Dec 5, 1985     Version 9.0
0000   500 ;               Fix parsing of Ultrix file specifiers.
0000   501 ;
0000   502 ;       114     Jim Shelhamer   Jan 13, 1986    Version 9.1
0000   503 ;               Remove saving of ap node number. It is now done
0000   504 ;               in APBOOT8200.
0000   505 ;
0000   506 ;       115     Bob Bergazzi    12-Feb-1986     Version 10.0            ;G:2
0000   507 ;               Added back the message about not running CRD online     ;G:2
0000   508 ;                                                                       ;G:3
0000   509 ;       116     Domenic Andella 20-Feb-1986     Version 10.0            ;G:3
0000   510 ;               Add SHOW CPU command to be used in concert with the     ;G:3
0000   511 ;               Set Cpu command to indicate which cpu's (primary or     ;G:3
0000   512 ;               KAn) gpr's or ipr's are to be referenced.               ;G:3
0000   513 ;                                                                       ;G:3
0000   514 ;       117     M. Baggett      11-March-1986   Version 10              ;G:4
0000   515 ;               Allow alpha value in ULTRIX partition field and         ;G:4
0000   516 ;               remove the SET ODS and SET ULTRIX command.              ;G:4
0000   517 ;                                                                       ;G:4
0000   518 ;       118     Domenic Andella 10-April-1986   Version 10              ;G:5
0000   519 ;               Fix some SHOW/SET CPU bugs, (1)-change kaaaa            ;G:5
0000   520 ;               to ka880, (2)-shorten mp$gt_device_name field,          ;G:5
0000   521 ;               and, (3)-reinit "primary" cpu in ACT_SETCPU.            ;G:5
0000   522 ;                                                                       ;G:5
0000   523 ;       119     Jim Shelhamer   14-Apr-1986     Version 10.0            ;G:6
0000   524 ;               Added check for AP on SET MEMORY command.               ;G:6
0000   525 ;                                                                       ;G:6
0000   526 ;       120     Bob Bergazzi    16-May-1986     Version 10.1            ;G:11
0000   527 ;               Added support for /TIME.                                ;G:11
0000   528 ;                                                                       ;G:14
0000   529 ;       121     Jim Baranski    21-May-1986     Version 10.1            ;G:14
0000   530 ;               Fix "SET LOAD []", and "EXAMINE A".                     ;G:14
0000   531 ;                                                                       ;G:16
0000   532 ;       122     I.Martin/S. Meier 14-July-1986  Version 10.2            ;G:16
0000   533 ;               In ACT_SHOWMM, replaced TSTB DS$GB_MM_ENB with          ;G:16
0000   534 ;               a TSTL IPR 38.                                          ;G:16
```

```
0000   535 ;                                                                                  ;G:16
0000   536 ;       123     Ingrid Martin   7-April-1987    Version 10.7                       ;G:26
0000   537 ;               Updates to allow KArrr to be attached CPU                          ;G:28
0000   538 ;                                                                                  ;G:27
0000   539 ;               Jason Mi        27-May-1987     Version 10.9                       ;G:27
0000   540 ;               Fixed bug in file directory specification.                         ;G:27
0000   541 ;                                                                                  ;G:28
0000   542 ;               Ingrid Martin   5-OCt-1987      Version 10.10                      ;G:28
0000   543 ;               KARRR is now KA884 (permenently).                                  ;G:28
0000   544 ;--
```

```
                     0000   546          .SBTTL  Libraries, Macros, Equated Symbols
                     0000   547 ;
                     0000   548 ; INCLUDE FILES:
                     0000   549 ;
                     0000   550          .LIBRARY        /SYS$LIBRARY:LIB/        ;                    [40]   ;G:2
                     0000   551          .Library        /$CRD/                   ;                    [56]   ;G:2
                     0000   552          .LIBRARY        /$DS/                    ;                    [40]   ;G:2
                     0000   553          .LIBRARY        /$DIAG/                  ;                    [40]   ;G:2
                     0000   554 ;
                     0000   555 ; EXTERNAL ROUTINES
                     0000   556 ;
                     0000   557          .EXTRN          DUMP_SELF                ; [97]
                     0000   558          .WEAK           DUMP_SELF                ; [97]
                     0000   559
                     0000   560
                     0000   561          .WEAK   MP$GL_CONDITION_FLAGS            ; [103]
                     0000   562 ;
                     0000   563 ; MACROS:
                     0000   564 ;
                     0000   565 .MACRO  CASEDEF ACT, N
                     0000   566          $$N=$$N+1
                     0000   567          ACT     = N
                     0000   568          .WORD   ACT_'ACT - 10$
                     0000   569 .ENDM   CASEDEF
                     0000   570 ;
                     0000   571 ; EQUATED SYMBOLS:
                     0000   572 ;
            00000001 0000   573 SS$_NORMAL = 1
                     0000   574
                     0000   575          $DS_CLIDEF                           ; CLI opcode defs
                     0000   576          $DS_HPODEF                           ; PTABLE offset defs.     [101]
                     0000            .SAVE   LOCAL_BLOCK
                     0000            .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
            00000008 0000            .IIF    NB,.BLKQ,       .BLKQ   1
                     0008            .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
            0000000A 0008            .IIF    NB,.BLKW,       .BLKW   1
                     000A            .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
            0000000B 000A            .IIF    NB,.BLKB,       .BLKB   1
                     000B            .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
            0000000C 000B            .IIF    NB,.BLKB,       .BLKB   1
                     000C            .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
            00000018 000C            .IIF    NB,.BLKB,       .BLKB   12
                     0018            .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
            0000001C 0018            .IIF    NB,.BLKL,       .BLKL   1
                     001C            .IIF    NB,HP$A_DVA,    HP$A_DVA:
            00000020 001C            .IIF    NB,.BLKL,       .BLKL   1
                     0020            .IIF    NB,HP$A_LINK,   HP$A_LINK:
            00000024 0020            .IIF    NB,.BLKL,       .BLKL   1
                     0024            .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
            00000026 0024            .IIF    NB,.BLKW,       .BLKW   1
                     0026            .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
            00000032 0026            .IIF    NB,.BLKB,       .BLKB   12
                     0032            .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                     0000            .RESTORE
                     0000   577          $DS_DSADEF                           ; APT mail box defs
                     0000   578          $DS_DSDEF                            ; Supervisor return codes
                     0000   579          CLIDEF                               ; CLI data structure
```

```
                     0000   580          DSFDEF                          ; Supervisor internal flags
                     0000   581          $IPLDEF                         ; IPL level defs
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 FE70                .=0
                     0000                .RESTORE
                     0000   582          $PRDEF                          ; Processor register defs
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 FE70                .=0
                     0000                .RESTORE
                     0000   583          $PSLDEF                         ; PSL field defs
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 FE70                .=0
                     0000                .RESTORE
                     0000   584          $DS_TypeDef                     ; Define type codes                    [48]
                     0000   585          $CRD_Literals                   ; Define CRD equated symbols           [56]
                     0000   586          CMKDEF                          ; CMK dispatch codes                   [92]
                     0000   587          APDEF                           ; AP offsets in AP$GW_DATA             [93]
                     0000   588          $PR8SSDEF                       ; Scorpio defs                         [119
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 FE70                .=0
                     0000                .RESTORE
                     0000   589
```

```
                                0000   591          .SUBTITLE       Work Psect Storage
                                0000   592 ;
                                0000   593 ; OWN STORAGE:
                                0000   594 ;
                            00000000   595          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                0000   596
                                0000   597 DS$GL_CLIBASE::
                            0000044C   598          .BLKB   CLI$K_SIZE
                                044C   599
                                044C   600 Q_GOOD_CMD:                          ; The portion of the command line that  [43]
                            00000454   601          .BLKQ   1                   ;   that is good. This is a descriptor. [43]
                                0454   602
                                0454   603 B_SUCCESS:                          ; Used to store SUCCESS or FAILURE      [46]
                            00000455   604          .BLKB   1                   ; for CLI subroutines                   [46]
                                0455   605
                                0455   606
                                0455   607
                59 52 41 4D 49 52 50   0455   608 PRIMARY_NAME: .ASCII    \PRIMARY\       ; Used as an argument in set cpu command [10
                                045C   609
                   34 38 38 41 4B 00'  045C   610 KA884_CPU :   .ASCIC    \KA884\         ; Valid processor type to be attached [123]
                                   05   045C
                   30 38 38 41 4B 00'  0462   611 KA880_CPU :   .ASCIC    \KA880\         ; Valid procssor type to be attached [118] ;
                                   05   0462
                   30 32 38 41 4B 00'  0468   612 KA820_CPU :   .ASCIC    \KA820\         ; Valid procssor type to be attached [111] ;
                                   05   0468
                                0046E   613
                            00000000    046E   614 PTABLE: .LONG   0                   ; Storage for the address of a p-table [101]
                                0472   615
                                  00    0472   616 MP$GB_SETCPU :: .BYTE   0           ; Set CPU flag                          [101] ;G:5
                                0473   617
                            00000000    0473   618 MP$GL_LUN ::    .LONG   0           ; Storage for Logical unit number [101] ;G:5
                                0477   619
                                  08    0477   620 MP$GT_DEVICE_NAME::     .byte  8        ;                                       [118] ;G:5
20 59 52 41 4D 49 52 50   0478   621          .ASCII \PRIMARY \ ; Preload SHOW CPU to indicate primary [118]          ;G:5
                                0480   622
```

```
                                        0480    624          .SUBTITLE       Data Psect Storage
                                        0480    625 ;
                                        0480    626 ; STRING STORAGE
                                        0480    627 ;
                                    00000000    628          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                        0000    629
                                        0000    630          MODNAM  CLI
                              49 4C 43 00'  0000      $MODULE:        .ASCIC  \CLI\
                                          03  0000
                                        0004    631
                                        0004    632          .SUBTITLE       Error & Status Messages Text
                                        0004    633
                                        0004    634 T_PRGERR:
6D 6D 61 72 67 6F 72 50 20 3F 3F 00'    0004    635          .ASCIC  "?? Programming error!!"                        ;[43
   21 21 72 6F 72 72 65 20 67 6E 69     0010
                                    16  0004
                                        001B    636
                                        001B    637 T_ILLCMD:
20 6C 61 67 65 6C 6C 49 20 3F 3F 00'    001B    638          .ASCIC  "?? Illegal command!/   \!AS\!/"               ;[43
20 20 20 2F 21 64 6E 61 6D 6D 6F 63     0027
                  2F 21 5C 53 41 21 5C  0033
                                    1E  001B
                                        003A    639
                                        003A    640 T_INCOMPLETE:
65 6C 70 6D 6F 63 6E 49 20 3F 3F 00'    003A    641          .ASCIC  "?? Incomplete command line!/   \!AS\!/"        ;[43
6C 20 64 6E 61 6D 6D 6F 63 20 65 74     0046
53 41 21 5C 20 20 20 2F 21 65 6E 69     0052
                              2F 21 5C  005E
                                    26  003A
                                        0061    642
                                        0061    643 T_ILLEFN:                                                   ;[45
20 6C 61 67 65 6C 6C 49 20 3F 3F 00'    0061    644          .ASCIC  "?? Illegal event flag number. Must be 1 to 23.!/"  ;[45
6E 20 67 61 6C 66 20 74 6E 65 76 65     006D
20 74 73 75 4D 20 2E 72 65 62 6D 75     0079
21 2E 33 32 20 6F 74 20 31 20 65 62     0085
                                    2F  0091
                                    30  0061
                                        0092    645
                                        0092    646 T_CONT:
69 75 6E 69 74 6E 6F 43 20 2E 2E 00'    0092    647          .ASCIC  ".. Continuing from !XL!/"                      ;[43
21 4C 58 21 20 6D 6F 72 66 20 67 6E     009E
                                    2F  00AA
                                    18  0092
                                        00AB    648
                                        00AB    649 T_MP_CONT:
75 6E 69 74 6E 6F 43 20 2E 2E 2E 00'    00AB    650          .ASCIC  "... Continuing !/"                            ;[11
                  2F 21 20 67 6E 69     00B7
                                    11  00AB
                                        00BD    651
                                        00BD    652 T_EMESSG1::
20 64 6E 61 6D 6D 6F 43 20 3F 3F 00'    00BD    653          .ASCIC "?? Command not valid in user mode !/"          ;[43
6E 69 20 64 69 6C 61 76 20 74 6F 6E     00C9
21 20 65 64 6F 6D 20 72 65 73 75 20     00D5
                                    2F  00E1
                                    24  00BD
                                        00E2    654
                                        00E2    655 T_EMESSG2:
```

```
20 64 6E 61 6D 6D 6F 43 20 3F 3F 00'  00E2   656          .ASCIC "?? Command only valid when running VDS Via NI !/"                ;[97
77 20 64 69 6C 61 76 20 79 6C 6E 6F   00EE
20 67 6E 69 6E 6E 75 72 20 6E 65 68   00FA
21 20 49 4E 20 61 69 56 20 53 44 56   0106
                                 2F   0112
                                 30   00E2
                                      0113   657
                                      0113   658 T_EMESSG3:
21 20 65 63 69 76 65 44 20 3F 3F 00'  0113   659          .ASCIC "?? Device !AS is not attached or diagnostic has not been started. !/
74 61 20 74 6F 6E 20 73 69 20 53 41   011F
69 64 20 72 6F 20 64 65 68 63 61 74   012B
73 61 68 20 63 69 74 73 6F 6E 67 61   0137
74 73 20 6E 65 65 62 20 74 6F 6E 20   0143
20 20 20 2F 21 20 2E 64 65 74 72 61   014F
69 76 65 64 20 68 63 61 74 74 41 20   015B
20 74 72 61 74 73 20 72 6F 20 65 63   0167
20 2E 63 69 74 73 6F 6E 67 61 69 64   0173
                           2F 21      017F
                              6D      0113
                                      0181   660                  "   Attach device or start diagnostic. !/"                  ;[10
                                      0181   661
                                      0181   662 T_EMESSG4:
74 20 65 63 69 76 65 44 20 3F 3F 00'  0181   663          .ASCIC "?? Device type must be !AC or !AC. !/"                ;       [101
20 65 62 20 74 73 75 6D 20 65 70 79   018D
20 2E 43 41 21 20 72 6F 20 43 41 21   0199
                           2F 21      01A5
                              25      0181
                                      01A7   664
                                      01A7   665 T_EMESSG5:                                                          ;[119] ;G:6
20 64 6E 61 6D 6D 6F 43 20 3F 3F 00'  01A7   666          .ASCIC "?? Command not valid from AP. !/"                    ;[119] ;G:6
72 66 20 64 69 6C 61 76 20 74 6F 6E   01B3
         2F 21 20 2E 50 41 20 6D 6F   01BF
                                 20   01A7
                                      01C8   667                                                                       ;G:6
                                      01C8   668 T_SHOWMM:
61 6E 61 6D 20 79 72 6F 6D 65 4D 00'  01C8   669          .ASCIC  "Memory management is !AC.!/"                        ;[43
41 21 20 73 69 20 74 6E 65 6D 65 67   01D4
                        2F 21 2E 43   01E0
                                 1B   01C8
                                      01E4   670
                                      01E4   671 T_ON:
                              6E 6F 00'  01E4   672          .ASCIC  "on"                                                ;[43
                                 02   01E4
                                      01E7   673
                                      01E7   674 T_OFF:
                           66 66 6F 00'  01E7   675          .ASCIC  "off"                                               ;[43
                                 03   01E7
                                      01EB   676
                                      01EB   677 GT_Menu_Error::                                                       ; [67]
6E 69 20 72 6F 72 72 45 20 3F 3F 00'  01EB   678          .Ascic  "?? Error in starting the CRD MENU TEST!!!/"         ;       [85]
68 74 20 67 6E 69 74 72 61 74 73 20   01F7
54 20 55 4E 45 4D 20 44 52 43 20 65   0203
            2F 21 21 21 54 53 45      020F
                                 2A   01EB
                                      0216   679
                                      0216   680 T_User_Mode_Menu_Error:                                              ; [115] ;G:2
6E 6E 61 63 20 75 6F 59 20 3F 3F 00'  0216   681          .Ascic  "?? You cannot run VAX-11/730 CRD MENU TEST in user mode!/"   ;G:2
```

```
31 2D 58 41 56 20 6E 75 72 20 74 6F  0222
45 4D 20 44 52 43 20 30 33 37 2F 31  022E
75 20 6E 69 20 54 53 45 54 20 55 4E  023A
      2F 21 65 64 6F 6D 20 72 65 73  0246
                                 39  0216
                                     0250   682                                                                  ;G:2
                                     0250   683 T_CRD_Trace_Output:                                                ; [65]
61 72 54 20 44 52 43 20 65 68 54 00' 0250   684         .Ascic  "The CRD Trace flag is !AC.!/"                    ; [65]
21 20 73 69 20 67 61 6C 66 20 65 63  025C
                     2F 21 2E 43 41  0268
                                 1C  0250
                                     026D   685
                                     026D   686 T_ShowEnforce:                                        ; [89]
65 63 72 6F 66 6E 65 20 65 68 54 00' 026D   687         .Ascic  "The enforce flag is !AC.!/"                ; [89]
43 41 21 20 73 69 20 67 61 6C 66 20  0279
                           2F 21 2E  0285
                                 1A  026D
                                     0288   688                                                                  ;G:3
                                     0288   689 T_SHOWCPU:                                            ;[116]      ;G:3
21 74 65 73 20 73 69 20 43 41 21 00' 0288   690         .ASCIC  "!AC is set!/"                          ;[116]      ;G:3
                                 2F  0294
                                 0C  0288
                                     0295   691                                                                  ;G:3
```

```
                        0295   693          .SUBTITLE COMMANDS A-C
                        0295   694 ;+
                        0295   695 ; Command line interpretor decision tree.
                        0295   696 ;-
                        0295   697
                        0295   698 ;
                        0295   699 ; Basic commands.
                        0295   700 ;
                        0295   701 COMMAND_TREE::
                        0295   702          $DS_CLI CLI$K_SPACE,    0,      100$                    ; Optional <SP>'s
              00 84     0295                       .BYTE   CLI$K_SPACE,0
               0004'    0297                       .WORD   100$-<.-2>
                        0299   703
                        0299   704 100$:    $DS_CLI <^A"A">,        NOTNUF, 115$                    ; A             [88]
              18'41     0299                       .BYTE   ^A"A",NOTNUF
               001F'    029B                       .WORD   115$-<.-2>
                        029D   705 ;
                        029D   706 ; ABort
                        029D   707 ;
                        029D   708          $DS_CLI CLI$K_STRING,   ABORT,  110$,   <'BORT'>        ; ABORT
              06'8B     029D                       .BYTE   CLI$K_STRING,ABORT
               000D'    029F                       .WORD   110$-<.-2>
        54 52 4F 42 00' 02A1                       .ASCIC  'BORT'
                  04    02A1
                        02A6   709          $DS_CLI CLI$K_BR,       ENUF,   EOL
              17'82     02A6                       .BYTE   CLI$K_BR,ENUF
               0209'    02A8                       .WORD   EOL-<.-2>
                        02AA   710 ;
                        02AA   711 ; ATtach<not parsed by CLI>
                        02AA   712 ;
                        02AA   713 110$:    $DS_CLI CLI$K_STRING,   ATTACH, ILLEGAL,<'TTACH'>       ; ATTACH
              07'8B     02AA                       .BYTE   CLI$K_STRING,ATTACH
               021D'    02AC                       .WORD   ILLEGAL-<.-2>
     48 43 41 54 54 00' 02AE                       .ASCIC  'TTACH'
                  05    02AE
                        02B4   714          $DS_CLI CLI$K_EXIT,     ENUF                            ; Enough input gotte
              17'81     02B4                       .BYTE   CLI$K_EXIT,ENUF
               0004'    02B6                       .WORD   30005$-<.-2>
                        02B8       30005$:
                        02B8   715 ;
                        02B8   716 ;
                        02B8   717 ; Boot  num_node
                        02B8   718 ;
                        02B8   719 115$:    $DS_CLI CLI$K_STRING,   BOOT,   120$,   <'BOOT'>        ; BOOT          [88]
              08'8B     02B8                       .BYTE   CLI$K_STRING,BOOT
               0015'    02BA                       .WORD   120$-<.-2>
        54 4F 4F 42 00' 02BC                       .ASCIC  'BOOT'
                  04    02BC
                        02C1   720          $DS_CLI CLI$K_SPACE,    NOTNUF, ILLEGAL                 ; Need more     [88]
              18'84     02C1                       .BYTE   CLI$K_SPACE,NOTNUF
               0206'    02C3                       .WORD   ILLEGAL-<.-2>
                        02C5   721          $DS_CLI CLI$K_NUM,      DATA,   ILLEGAL                 ; Get node number [8
              66'85     02C5                       .BYTE   CLI$K_NUM,DATA
               0202'    02C7                       .WORD   ILLEGAL-<.-2>
                        02C9   722          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;               [88]
              17'82     02C9                       .BYTE   CLI$K_BR,ENUF
               01E6'    02CB                       .WORD   EOL-<.-2>
```

```
                          02CD   723 ;
                          02CD   724 ;+
                          02CD   725 ;       The Commands Starting with C:
                          02CD   726 ;                                                                    [65]
                          02CD   727 ;       CLear                                                        [65]
                          02CD   728 ;       COntinue                                                     [65]
                          02CD   729 ;       CRd                                                          [65]
                          02CD   730 ;-
                          02CD   731 120$:   $DS_CLI <^A"C">,        NOTNUF, 140$                  ; C
                    18'43 02CD                       .BYTE   ^A"C",NOTNUF
                     002C' 02CF                       .WORD   140$-<.-2>
                          02D1   732 ;
                          02D1   733 ; CLear...
                          02D1   734 ;
                          02D1   735         $DS_CLI CLI$K_STRING,   CLEAR,  130$,   <'LEAR'>        ; CLEAR
                    09'8B 02D1                       .BYTE   CLI$K_STRING,CLEAR
                     000D' 02D3                       .WORD   130$-<.-2>
           52 41 45 4C 00' 02D5                       .ASCIC  'LEAR'
                       04  02D5
                          02DA   736         $DS_CLI CLI$K_BR,       NOTNUF, CLEAR_PARAMS            ; Get rest of line
                    18'82 02DA                       .BYTE   CLI$K_BR,NOTNUF
                     099E' 02DC                       .WORD   CLEAR_PARAMS-<.-2>
                          02DE   737 ;
                          02DE   738 ; COntinue
                          02DE   739 ;
                          02DE   740 130$:   $DS_CLI CLI$K_STRING,   CONTINUE,135$,<'ONTINUE'>       ; CONTINUE      [60]
                    0A'8B 02DE                       .BYTE   CLI$K_STRING,CONTINUE
                     0010' 02E0                       .WORD   135$-<.-2>
     45 55 4E 49 54 4E 4F 00' 02E2                       .ASCIC  'ONTINUE'
                       07  02E2
                          02EA   741         $DS_CLI CLI$K_BR,       ENUF,   EOL
                    17'82 02EA                       .BYTE   CLI$K_BR,ENUF
                     01C5' 02EC                       .WORD   EOL-<.-2>
                          02EE   742 ;                                                                    [60]
                          02EE   743 ; CRd                                                                [60]
                          02EE   744 ;                                                                    [60]
                          02EE   745 135$:   $DS_Cli CLI$K_String,   CRD,    Illegal,<'RD'>          ; CRD           [65]
                    27'8B 02EE                       .BYTE   CLI$K_String,CRD
                     01D9' 02F0                       .WORD   Illegal-<.-2>
                 44 52 00' 02F2                       .ASCIC  'RD'
                       02  02F2
                          02F5   746         $DS_Cli CLI$K_BR,       ENUF,   EOL                     ; Must be EOL   [65]
                    17'82 02F5                       .BYTE   CLI$K_BR,ENUF
                     01BA' 02F7                       .WORD   EOL-<.-2>
```

```
                          02F9   748          .SUBTITLE       DIRECTORY, DEATTACH, DESELECT
                          02F9   749 ;
                          02F9   750 ;       The Commands Starting with D:
                          02F9   751 ;
                          02F9   752 ;       Deposit
                          02F9   753 ;       DIrectory
                          02F9   754 ;       DEAttach                                                    [48]
                          02F9   755 ;       DEBug                                                       [59]
                          02F9   756 ;       DESelect
                          02F9   757 ;       DUmp
                          02F9   758 ;
                          02F9   759 140$:   $DS_CLI <^A"D">,      NOTNUF,   200$                    ; D
                 18'44    02F9                       .BYTE   ^A"D",NOTNUF
                  00AA'   02FB                       .WORD   200$-<.-2>
                          02FD   760 ;
                          02FD   761 ; DIrectory[ / Wide]  <get_file_spec>
                          02FD   762 ;
                          02FD   763          $DS_CLI CLI$K_STRING,   DIRECTORY,150$, <'IRECTORY'>    ; DIRECTORY
                 10'8B    02FD                       .BYTE   CLI$K_STRING,DIRECTORY
                  0026'   02FF                       .WORD   150$-<.-2>
      59 52 4F 54 43 45 52 49 00' 0301               .ASCIC  'IRECTORY'
                    08    0301
                          030A   764          $DS_CLI CLI$K_SLASH,    NOTNUF, 145$                      ; /             [80]
                 18'90    030A                       .BYTE   CLI$K_SLASH,NOTNUF
                  000D'   030C                       .WORD   145$-<.-2>
                          030E   765          $DS_CLI CLI$K_STRING,   DIR_WIDE,BAD_QUAL,<'WIDE'>      ; /WIDE         [80]
                 8C'8B    030E                       .BYTE   CLI$K_STRING,DIR_WIDE
                  01ED'   0310                       .WORD   BAD_QUAL-<.-2>
                45 44 49 57 00' 0312               .ASCIC  'WIDE'
                    04    0312
                          0317   766 145$:   $DS_CLI CLI$K_SPACE,    0,      EOL                       ; If not <SP>'s, EOL
                 00 84    0317                       .BYTE   CLI$K_SPACE,0
                  0198'   0319                       .WORD   EOL-<.-2>
                          031B   767          $DS_CLI CLI$K_CALL,     OPTIONAL,GET_FILE_SPEC            ; Optional file-spec
                 2A'92    031B                       .BYTE   CLI$K_CALL,OPTIONAL
                  0333'   031D                       .WORD   GET_FILE_SPEC-<.-2>
                          031F   768          $DS_CLI CLI$K_BR,       ENUF,   EOL                       ; Done with command
                 17'82    031F                       .BYTE   CLI$K_BR,ENUF
                  0190'   0321                       .WORD   EOL-<.-2>
                          0323   769 ;
                          0323   770 ; DUmp[  num_node]
                          0323   771 ;
                          0323   772 150$:   $DS_CLI CLI$K_STRING,   DUMP,   160$,   <'UMP'>          ; DUMP          [97]
                 11'8B    0323                       .BYTE   CLI$K_STRING,DUMP
                  001C'   0325                       .WORD   160$-<.-2>
            50 4D 55 00'  0327               .ASCIC  'UMP'
                    03    0327
                          032B   773          $DS_CLI CLI$K_SPACE,    0,      155$                      ; Optional:     [97]
                 00 84    032B                       .BYTE   CLI$K_SPACE,0
                  000C'   032D                       .WORD   155$-<.-2>
                          032F   774          $DS_CLI CLI$K_NUM,      DATA,   ILLEGAL                   ;
                 66'85    032F                       .BYTE   CLI$K_NUM,DATA
                  0198'   0331                       .WORD   ILLEGAL-<.-2>
                          0333   775          $DS_CLI CLI$K_BR,       ENUF,   EOL                       ;
                 17'82    0333                       .BYTE   CLI$K_BR,ENUF
                  017C'   0335                       .WORD   EOL-<.-2>
                          0337   776 155$:   $DS_CLI CLI$K_NUM,      NULL_DATA,EOL                     ; More  ?       [97]
```

```
                     67'85  0337                        .BYTE   CLI$K_NUM,NULL_DATA
                      0178' 0339                        .WORD   EOL-<.-2>
                            033B   777          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;                       [97]
                     17'82  033B                        .BYTE   CLI$K_BR,ENUF
                      0174' 033D                        .WORD   EOL-<.-2>
                            033F   778
                            033F   779 160$:     $DS_CLI <^A"E">,        0,      191$                    ; DE
                     00 45  033F                        .BYTE   ^A"E",0
                      0044' 0341                        .WORD   191$-<.-2>
                            0343   780 ;
                            0343   781 ; DEAttach...
                            0343   782 ;
                            0343   783          $DS_CLI CLI$K_STRING,   DEATTACH,170$,  <'ATTACH'>      ; DEATTACH      [48]
                     0B'8B  0343                        .BYTE   CLI$K_STRING,DEATTACH
                      000F' 0345                        .WORD   170$-<.-2>
          48 43 41 54 54 41 00' 0347                    .ASCIC  'ATTACH'
                        06  0347
                            034E   784          $DS_CLI CLI$K_BR,       NOTNUF, DEATTACH_PARAMS         ; Get params    [48]
                     18'82  034E                        .BYTE   CLI$K_BR,NOTNUF
                      0653' 0350                        .WORD   DEATTACH_PARAMS-<.-2>
                            0352   785 ;                                                                [59]
                            0352   786 ; DEBug[=<file_spec>]                                           [59]
                            0352   787 ;
                            0352   788 170$:     $DS_CLI CLI$K_STRING,   DEBUG,  180$,   <'BUG'>         ; DEBUG         [59]
                     0C'8B  0352                        .BYTE   CLI$K_STRING,DEBUG
                      0018' 0354                        .WORD   180$-<.-2>
                47 55 42 00' 0356                       .ASCIC  'BUG'
                        03  0356
                            035A   789          $DS_CLI <^A/=/>,        0,      175$                    ; IF NOT '=', EOL[59
                     00 3D  035A                        .BYTE   ^A/=/,0
                      000C' 035C                        .WORD   175$-<.-2>
                            035E   790          $DS_CLI CLI$K_CALL,     OPTIONAL,GET_FILE_SPEC          ; OPT. FILENAME [59]
                     2A'92  035E                        .BYTE   CLI$K_CALL,OPTIONAL
                      02F0' 0360                        .WORD   GET_FILE_SPEC-<.-2>
                            0362   791          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ; Done with Command
                     17'82  0362                        .BYTE   CLI$K_BR,ENUF
                      014D' 0364                        .WORD   EOL-<.-2>
                            0366   792 175$:     $DS_CLI CLI$K_BR,       DEFAULT_DBG,EOL                 ; USE DEFLT NAME[59]
                     0E'82  0366                        .BYTE   CLI$K_BR,DEFAULT_DBG
                      0149' 0368                        .WORD   EOL-<.-2>
                            036A   793 ;
                            036A   794 ; DESelect...
                            036A   795 ;
                            036A   796 180$:     $DS_CLI CLI$K_STRING,   DESELECT,190$,  <'SELECT'>      ; DESELECT
                     7A'8B  036A                        .BYTE   CLI$K_STRING,DESELECT
                      000F' 036C                        .WORD   190$-<.-2>
          54 43 45 4C 45 53 00' 036E                    .ASCIC  'SELECT'
                        06  036E
                            0375   797          $DS_CLI CLI$K_BR,       NOTNUF, DESELECT_PARAMS         ; Get qualifiers
                     18'82  0375                        .BYTE   CLI$K_BR,NOTNUF
                      0674' 0377                        .WORD   DESELECT_PARAMS-<.-2>
```

```
                         0379   799          .SUBTITLE       DEPOSIT, EXIT, EXAMINE
                         0379   800
                         0379   801 ; Deposit<qualifiers>  address<qualifiers>  data<qualifiers>
                         0379   802 ;
                         0379   803 190$:   $DS_CLI CLI$K_STRING,   0,      191$,   <'POSIT'>       ; DEPOSIT
                   00 8B 0379                       .BYTE   CLI$K_STRING,0
                    000A' 037B                      .WORD   191$-<.-2>
          54 49 53 4F 50 00' 037D                   .ASCIC  'POSIT'
                      05  037D
                         0383   804 191$:   $DS_CLI CLI$K_CALL,     DEPOSIT,EXA_DEP_QUALS          ; Get Qualifiers
                   0F'92 0383                       .BYTE   CLI$K_CALL,DEPOSIT
                    056E' 0385                      .WORD   EXA_DEP_QUALS-<.-2>
                         0387   805          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required Space
                   00 84 0387                       .BYTE   CLI$K_SPACE,0
                    0140' 0389                      .WORD   ILLEGAL-<.-2>
                         038B   806          $DS_CLI CLI$K_CALL,     0,      GET_ADDRESS             ; Get Address
                   00 92 038B                       .BYTE   CLI$K_CALL,0
                    03E7' 038D                      .WORD   GET_ADDRESS-<.-2>
                         038F   807          $DS_CLI CLI$K_CALL,     0,      EXA_DEP_QUALS           ; Get Qualifiers
                   00 92 038F                       .BYTE   CLI$K_CALL,0
                    0562' 0391                      .WORD   EXA_DEP_QUALS-<.-2>
                         0393   808          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required Space
                   00 84 0393                       .BYTE   CLI$K_SPACE,0
                    0134' 0395                      .WORD   ILLEGAL-<.-2>
                         0397   809          $DS_CLI CLI$K_NUM,      DATA,   ILLEGAL                 ; Get Data
                   66'85 0397                       .BYTE   CLI$K_NUM,DATA
                    0130' 0399                      .WORD   ILLEGAL-<.-2>
                         039B   810          $DS_CLI CLI$K_CALL,     0,      EXA_DEP_QUALS           ; Get Qualifiers
                   00 92 039B                       .BYTE   CLI$K_CALL,0
                    0556' 039D                      .WORD   EXA_DEP_QUALS-<.-2>
                         039F   811          $DS_CLI CLI$K_BR,       0,      EOL                     ; End Of Line
                   00 82 039F                       .BYTE   CLI$K_BR,0
                    0110' 03A1                      .WORD   EOL-<.-2>
                         03A3   812 ;
                         03A3   813 ;        The Commands Starting with E:
                         03A3   814 ;
                         03A3   815 ;        Examine
                         03A3   816 ;        EXIt
                         03A3   817 ;
                         03A3   818 200$:   $DS_CLI <^A"E">,        NOTNUF, 235$                    ; E
                   18'45 03A3                       .BYTE   ^A"E",NOTNUF
                    0031' 03A5                      .WORD   235$-<.-2>
                         03A7   819          $DS_CLI <^A"X">,        NOTNUF, 220$                    ; EX
                   18'58 03A7                       .BYTE   ^A"X",NOTNUF
                    000F' 03A9                      .WORD   220$-<.-2>
                         03AB   820
                         03AB   821 ;
                         03AB   822 ; EXIt
                         03AB   823 ;
                         03AB   824          $DS_CLI CLI$K_STRING,   EXIT,   220$,   <'IT'>          ; EXIT
                   13'8B 03AB                       .BYTE   CLI$K_STRING,EXIT
                    000B' 03AD                      .WORD   220$-<.-2>
                54 49 00' 03AF                      .ASCIC  'IT'
                      02  03AF
                         03B2   825          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ; EXIT <EOL>
                   17'82 03B2                       .BYTE   CLI$K_BR,ENUF
                    00FD' 03B4                      .WORD   EOL-<.-2>
```

```
                        03B6    826 ;
                        03B6    827 ; Examine<qualifiers>  num_address<qualifiers>
                        03B6    828 ;
                        03B6    829 220$:   $DS_CLI CLI$K_STRING,   0,      230$,   <'AMINE'>       ; EXAMINE
                 00 8B  03B6                        .BYTE   CLI$K_STRING,0
                  000A' 03B8                        .WORD   230$-<.-2>
      45 4E 49 4D 41 00' 03BA                        .ASCIC  'AMINE'
                    05  03BA
                        03C0    830 230$:   $DS_CLI CLI$K_CALL,     EXAMINE,EXA_DEP_QUALS
                12'92   03C0                        .BYTE   CLI$K_CALL,EXAMINE
                 0531'  03C2                        .WORD   EXA_DEP_QUALS-<.-2>
                        03C4    831         $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required Space
                 00 84  03C4                        .BYTE   CLI$K_SPACE,0
                  0103' 03C6                        .WORD   ILLEGAL-<.-2>
                        03C8    832         $DS_CLI CLI$K_CALL,     0,      GET_ADDRESS
                 00 92  03C8                        .BYTE   CLI$K_CALL,0
                  03AA' 03CA                        .WORD   GET_ADDRESS-<.-2>
                        03CC    833         $DS_CLI CLI$K_CALL,     0,      EXA_DEP_QUALS
                 00 92  03CC                        .BYTE   CLI$K_CALL,0
                  0525' 03CE                        .WORD   EXA_DEP_QUALS-<.-2>
                        03D0    834         $DS_CLI CLI$K_BR,       0,      EOL                     ; End Of Line
                 00 82  03D0                        .BYTE   CLI$K_BR,0
                  00DF' 03D2                        .WORD   EOL-<.-2>
```

```
                                03D4   836          .SUBTITLE LOAD, NEXT, RUN, HELP, INITPCS
                                03D4   837 ;
                                03D4   838 ; Help<not_parsed_by_cli>
                                03D4   839 ;
                                03D4   840 235$:    $DS_CLI CLI$K_STRING,   HELP,   240$,   <'HELP'>        ; HELP
                      14'8B     03D4                        .BYTE   CLI$K_STRING,HELP
                       0019'    03D6                        .WORD   240$-<.-2>
                50 4C 45 48 00' 03D8                        .ASCIC  'HELP'
                          04    03D8
                                03DD   841          $DS_CLI CLI$K_EOL,      0,      237$                    ; 'H<CR>'
                      00 91     03DD                        .BYTE   CLI$K_EOL,0
                       0008'    03DF                        .WORD   237$-<.-2>
                                03E1   842          $DS_CLI CLI$K_EXIT,     ENUF                            ;
                      17'81     03E1                        .BYTE   CLI$K_EXIT,ENUF
                       0004'    03E3                        .WORD   30061$-<.-2>
                                03E5            30061$:
                                03E5   843 237$:    $DS_CLI CLI$K_SPACE,    HELP,   ILLEGAL                 ; Required space(s)
                      14'84     03E5                        .BYTE   CLI$K_SPACE,HELP
                       00E2'    03E7                        .WORD   ILLEGAL-<.-2>
                                03E9   844          $DS_CLI CLI$K_EXIT,     ENUF                            ; Enough input gotte
                      17'81     03E9                        .BYTE   CLI$K_EXIT,ENUF
                       0004'    03EB                        .WORD   30063$-<.-2>
                                03ED            30063$:
                                03ED   845 ;
                                03ED   846 ; Initpcs                                                        begi
                                03ED   847 ;
                                03ED   848 240$:    $DS_CLI CLI$K_STRING,   INITPCS,245$,   <'INITPCS'>     ; INITPCS
                      8B'8B     03ED                        .BYTE   CLI$K_STRING,INITPCS
                       0010'    03EF                        .WORD   245$-<.-2>
        53 43 50 54 49 4E 49 00' 03F1                        .ASCIC  'INITPCS'
                          07    03F1
                                03F9   849          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;                   end
                      17'82     03F9                        .BYTE   CLI$K_BR,ENUF
                       00B6'    03FB                        .WORD   EOL-<.-2>
                                03FD   850 ;
                                03FD   851 ; Load  <file_spec>
                                03FD   852 ;
                                03FD   853 245$:    $DS_CLI CLI$K_STRING,   LOAD,   250$,   <'LOAD'>        ; LOAD
                      15'8B     03FD                        .BYTE   CLI$K_STRING,LOAD
                       0015'    03FF                        .WORD   250$-<.-2>
                44 41 4F 4C 00' 0401                        .ASCIC  'LOAD'
                          04    0401
                                0406   854          $DS_CLI CLI$K_SPACE,    NOTNUF, ILLEGAL                 ; <SP>'s req.
                      18'84     0406                        .BYTE   CLI$K_SPACE,NOTNUF
                       00C1'    0408                        .WORD   ILLEGAL-<.-2>
                                040A   855          $DS_CLI CLI$K_CALL,     REQUIRED,GET_FILE_SPEC          ; File-spec required
                      2B'92     040A                        .BYTE   CLI$K_CALL,REQUIRED
                       0244'    040C                        .WORD   GET_FILE_SPEC-<.-2>
                                040E   856          $DS_CLI CLI$K_BR,       ENUF,   EOL
                      17'82     040E                        .BYTE   CLI$K_BR,ENUF
                       00A1'    0410                        .WORD   EOL-<.-2>
                                0412   857 ;
                                0412   858 ; Next[  <dec_mnumber>]
                                0412   859 ;
                                0412   860 250$:    $DS_CLI CLI$K_STRING,   NEXT_INST,260$, <'NEXT'>        ; NEXT
                      16'8B     0412                        .BYTE   CLI$K_STRING,NEXT_INST
                       0015'    0414                        .WORD   260$-<.-2>
```

```
54 58 45 4E 00' 0416                        .ASCIC  'NEXT'
             04 0416
                041B   861          $DS_CLI CLI$K_SPACE,    0,      EOL
          00 84 041B                        .BYTE   CLI$K_SPACE,0
           0094' 041D                        .WORD   EOL-<.-2>
                041F   862          $DS_CLI CLI$K_DEC,      DATA,   ILLEGAL                 ; If not dec., ILLEG
          66'8A 041F                        .BYTE   CLI$K_DEC,DATA
           00A8' 0421                        .WORD   ILLEGAL-<.-2>
                0423   863          $DS_CLI CLI$K_BR,       ENUF,   EOL
          17'82 0423                        .BYTE   CLI$K_BR,ENUF
           008C' 0425                        .WORD   EOL-<.-2>
                0427   864 ;
                0427   865 ; Run[<qualifiers>]  <file_spec>[<qualifiers>]
                0427   866 ;
                0427   867 260$:    $DS_CLI CLI$K_STRING,   RUN,    270$,   <'RUN'>         ; RUN
          19'8B 0427                        .BYTE   CLI$K_STRING,RUN
           001C' 0429                        .WORD   270$-<.-2>
     4E 55 52 00' 042B                        .ASCIC  'RUN'
             03 042B
                042F   868          $DS_CLI CLI$K_CALL,     0,      START_RUN_QUALS         ; Read Qaulifiers
          00 92 042F                        .BYTE   CLI$K_CALL,0
           039E' 0431                        .WORD   START_RUN_QUALS-<.-2>
                0433   869          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required spaces.
          00 84 0433                        .BYTE   CLI$K_SPACE,0
           0094' 0435                        .WORD   ILLEGAL-<.-2>
                0437   870          $DS_CLI CLI$K_CALL,     0,      GET_FILE_SPEC           ; Read File Specific
          00 92 0437                        .BYTE   CLI$K_CALL,0
           0217' 0439                        .WORD   GET_FILE_SPEC-<.-2>
                043B   871          $DS_CLI CLI$K_CALL,     0,      START_RUN_QUALS         ; Read Qualifiers
          00 92 043B                        .BYTE   CLI$K_CALL,0
           0392' 043D                        .WORD   START_RUN_QUALS-<.-2>
                043F   872          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ; Done With CoMMAND
          17'82 043F                        .BYTE   CLI$K_BR,ENUF
           0070' 0441                        .WORD   EOL-<.-2>
```

```
                    0443   874          .SUBTITLE       SET, SELECT, SHOW, START, SUMMARY, XDELTA, a
                    0443   875 ;
                    0443   876 ;        The Commands Starting with S:
                    0443   877 ;
                    0443   878 ;        SET
                    0443   879 ;        SELect
                    0443   880 ;        SHow
                    0443   881 ;        STart
                    0443   882 ;        SUmmary
                    0443   883 ;
                    0443   884 270$:    $DS_CLI <^A"S">,        NOTNUF, 320$                    ; S                [60]
            18'53   0443                        .BYTE   ^A"S",NOTNUF
             0049'  0445                        .WORD   320$-<.-2>
                    0447   885 ;
                    0447   886 ; SET...
                    0447   887 ;
                    0447   888          $DS_CLI <^A"E">,        NOTNUF, 290$                    ; SE
            18'45   0447                        .BYTE   ^A"E",NOTNUF
             0019'  0449                        .WORD   290$-<.-2>
                    044B   889          $DS_CLI <^A"T">,        SET,    280$                    ; SET
            1B'54   044B                        .BYTE   ^A"T",SET
             0008'  044D                        .WORD   280$-<.-2>
                    044F   890          $DS_CLI CLI$K_BR,       NOTNUF, SET_PARAMS              ; Get parameters
            18'82   044F                        .BYTE   CLI$K_BR,NOTNUF
             08C5'  0451                        .WORD   SET_PARAMS-<.-2>
                    0453   891 ;
                    0453   892 ; SELect...
                    0453   893 ;                                                                        ;G:27
                    0453   894 280$:    $DS_CLI CLI$K_STRING,   SELECT, ILLEGAL,<'LECT'>        ; SELECT
            7B'8B   0453                        .BYTE   CLI$K_STRING,SELECT
             0074'  0455                        .WORD   ILLEGAL-<.-2>
     54 43 45 4C 00'  0457                        .ASCIC  'LECT'
                04   0457
                    045C   895          $DS_CLI CLI$K_BR,       NOTNUF, SELECT_PARAMS
            18'82   045C                        .BYTE   CLI$K_BR,NOTNUF
             05AD'  045E                        .WORD   SELECT_PARAMS-<.-2>
                    0460   896 ;
                    0460   897 ; SHow...
                    0460   898 ;
                    0460   899 290$:    $DS_CLI CLI$K_STRING,   SHOW,   300$,   <'HOW'>         ; SHOW
            1D'8B   0460                        .BYTE   CLI$K_STRING,SHOW
             000C'  0462                        .WORD   300$-<.-2>
        57 4F 48 00'  0464                        .ASCIC  'HOW'
                03   0464
                    0468   900          $DS_CLI CLI$K_BR,       NOTNUF, SHOW_PARAMS
            18'82   0468                        .BYTE   CLI$K_BR,NOTNUF
             05D8'  046A                        .WORD   SHOW_PARAMS-<.-2>
                    046C   901 ;
                    046C   902 ; STart[<qualifiers>]
                    046C   903 ;
                    046C   904 300$:    $DS_CLI CLI$K_STRING,   START,  310$,   <'TART'>        ; START
            25'8B   046C                        .BYTE   CLI$K_STRING,START
             0011'  046E                        .WORD   310$-<.-2>
     54 52 41 54 00'  0470                        .ASCIC  'TART'
                04   0470
                    0475   905          $DS_CLI CLI$K_CALL,     0,      START_RUN_QUALS         ; Read Qualifiers
            00 92   0475                        .BYTE   CLI$K_CALL,0
```

```
                        0358' 0477                      .WORD   START_RUN_QUALS-<.-2>
                              0479    906       $DS_CLI CLI$K_BR,       ENUF,   EOL
                            17'82  0479                 .BYTE   CLI$K_BR,ENUF
                             0036' 047B                 .WORD   EOL-<.-2>
                              047D    907 ;
                              047D    908 ; SUmmary
                              047D    909 ;
                              047D    910 310$:   $DS_CLI CLI$K_STRING,   SUMMARY,ILLEGAL,<'UMMARY'>      ; SUMMARY
                            26'8B  047D                 .BYTE   CLI$K_STRING,SUMMARY
                             004A' 047F                 .WORD   ILLEGAL-<.-2>
                  59 52 41 4D 4D 55 00' 0481            .ASCIC  'UMMARY'
                                06  0481
                              0488    911       $DS_CLI CLI$K_BR,       ENUF,   EOL
                            17'82  0488                 .BYTE   CLI$K_BR,ENUF
                             0027' 048A                 .WORD   EOL-<.-2>
                              048C    912
                              048C    913 ;
                              048C    914 ; Xdeltaa <file_spec>
                              048C    915 ;
                              048C    916 320$:   $DS_CLI CLI$K_BIF,      IF_XDELTA,330$
                            74'83  048C                 .BYTE   CLI$K_BIF,IF_XDELTA
                             0013' 048E                 .WORD   330$-<.-2>
                              0490    917       $DS_CLI CLI$K_STRING,   XDELTA, 330$,   <'XDELTA'>      ; XDELTA
                            75'8B  0490                 .BYTE   CLI$K_STRING,XDELTA
                             000F' 0492                 .WORD   330$-<.-2>
                  41 54 4C 45 44 58 00' 0494            .ASCIC  'XDELTA'
                                06  0494
                              049B    918       $DS_CLI CLI$K_BR,       ENUF,   EOL
                            17'82  049B                 .BYTE   CLI$K_BR,ENUF
                             0014' 049D                 .WORD   EOL-<.-2>
                              049F    919 ;
                              049F    920 ; a <file-spec>
                              049F    921 ;
                              049F    922 330$:   $DS_CLI <^A"a">,        SCRIPT, EOL                     ; a
                            1A'40  049F                 .BYTE   ^A"a",SCRIPT
                             0010' 04A1                 .WORD   EOL-<.-2>
                              04A3    923       $DS_CLI CLI$K_SPACE,    NOTNUF, 340$                    ; Optional <SP>'s
                            18'84  04A3                 .BYTE   CLI$K_SPACE,NOTNUF
                             0004' 04A5                 .WORD   340$-<.-2>
                              04A7    924 340$:   $DS_CLI CLI$K_CALL      REQUIRED,GET_FILE_SPEC          ; File-spec required
                            2B'92  04A7                 .BYTE   CLI$K_CALL,REQUIRED
                             01A7' 04A9                 .WORD   GET_FILE_SPEC-<.-2>
                              04AB    925       $DS_CLI CLI$K_BR,       ENUF,   EOL                     ; Done With Command
                            17'82  04AB                 .BYTE   CLI$K_BR,ENUF
                             0004' 04AD                 .WORD   EOL-<.-2>
```

```
                        04AF    927          .SUBTITLE       END OF LINE & ILLEGAL Processing
                        04AF    928 ;
                        04AF    929 ; END OF LINE.
                        04AF    930 ;
                        04AF    931 ; If really EOL, return SUCCESSfully. If not, advance past the next character
                        04AF    932 ; (the one that is illegal), save the context at this point, and return.
                        04AF    933 ;
                        04AF    934
                        04AF    935 EOL:    $DS_CLI CLI$K_SPACE,    0,      100$                    ; Jump past spaces
                  00 84 04AF                    .BYTE   CLI$K_SPACE,0
                   0004' 04B1                   .WORD   100$-<.-2>
                        04B3    936 100$:   $DS_CLI CLI$K_EOL,      0,      110$                    ; If not EOL, branch
                  00 91 04B3                    .BYTE   CLI$K_EOL,0
                   0008' 04B5                   .WORD   110$-<.-2>
                        04B7    937         $DS_CLI CLI$K_EXIT                                      ; Exit successfully
                  00 81 04B7                    .BYTE   CLI$K_EXIT,0
                   0004' 04B9                   .WORD   30102$-<.-2>
                        04BB        30102$:
                        04BB    938
                        04BB    939 110$:   $DS_CLI CLI$K_BR,       ADVANCE,120$                    ; Jump past illegal
                  65'82 04BB                    .BYTE   CLI$K_BR,ADVANCE
                   0004' 04BD                   .WORD   120$-<.-2>
                        04BF    940 120$:   $DS_CLI CLI$K_BR,       ILL_CMD,130$                    ; Save current point
                  01'82 04BF                    .BYTE   CLI$K_BR,ILL_CMD
                   0004' 04C1                   .WORD   130$-<.-2>
                        04C3    941 130$:   $DS_CLI CLI$K_ERROR                                     ; Indicate ILLEGAL
                  00 80 04C3                    .BYTE   CLI$K_ERROR,0
                   0004' 04C5                   .WORD   30105$-<.-2>
                        04C7        30105$:
                        04C7    942 ;
                        04C7    943 ; Errors.
                        04C7    944 ;
                        04C7    945
                        04C7    946 ILLEGAL:
                        04C7    947
                        04C7    948 ;
                        04C7    949 ; Illegal command error:
                        04C7    950 ;
                        04C7    951 ; This is called with the parse character pointer pointing to the illegal
                        04C7    952 ; character. This will usually be a character within an unrecognized
                        04C7    953 ; qualifier or parameter such as the "X" in "SET WIDTX". It is possible
                        04C7    954 ; for this to be called when a command is actually INCOMPLETE. This
                        04C7    955 ; happens, for example, in recognizing "SET BREAKPOINTS" but failing
                        04C7    956 ; to see any (required) spaces after the "BREAKPOINTS". If no spaces
                        04C7    957 ; are seen, ILLEGAL is called even though the next symbol could
                        04C7    958 ; actually be EOL (implying INCOMPLETE). If it is not an EOL, then there
                        04C7    959 ; is an illegal character (e.g., the "X" above) and the command is
                        04C7    960 ; ILLEGAL. If it is an EOL, then the required address is missing and the
                        04C7    961 ; command is INCOMPLETE. Therefore, EOL is checked below to make a
                        04C7    962 ; determination about the kind of error that has occurred.
                        04C7    963 ;
                        04C7    964 100$:   $DS_CLI CLI$K_EOL,      0,      110$                    ; If not EOL, branch
                  00 91 04C7                    .BYTE   CLI$K_EOL,0
                   0008' 04C9                   .WORD   110$-<.-2>
                        04CB    965         $DS_CLI CLI$K_ERROR     INC_CMD                         ; If EOL, exit w/INC
                  02'80 04CB                    .BYTE   CLI$K_ERROR,INC_CMD
                   0004' 04CD                   .WORD   30107$-<.-2>
```

```
                        04CF        30107$:
                        04CF    966 ;
                        04CF    967 ;
                        04CF    968 ; Not really EOL, somthing illegal is there. First, get rid of any spaces
                        04CF    969 ; that are there and then advance past the illegal character.
                        04CF    970 ;
                        04CF    971 110$:   $DS_CLI CLI$K_SPACE,    0,      115$                    ; Jump past any spac
                  00 84 04CF                .BYTE   CLI$K_SPACE,0
                   0004' 04D1               .WORD   115$-<.-2>
                        04D3    972 115$:   $DS_CLI CLI$K_BR,       ADVANCE,120$                    ; Jump past illegal
                  65'82 04D3                .BYTE   CLI$K_BR,ADVANCE
                   0004' 04D5               .WORD   120$-<.-2>
                        04D7    973 120$:   $DS_CLI CLI$K_ERROR,    ILL_CMD                         ; Print error mess.
                  01'80 04D7                .BYTE   CLI$K_ERROR,ILL_CMD
                   0004' 04D9               .WORD   30110$-<.-2>
                        04DB        30110$:
```

```
                04DB          .SUBTITLE       INCOMPLETE, BAD LIST & BAD QUAL Processing
                04DB    INCOMPLETE:
                04DB    ;
                04DB    ;
                04DB    ; Incomplete command error:
                04DB    ;
                04DB          $DS_CLI CLI$K_ERROR,    INC_CMD                         ; Print error mess.
        02'80   04DB                  .BYTE   CLI$K_ERROR,INC_CMD
         0004'  04DD                  .WORD   30111$-<.-2>
                04DF    30111$:
                04DF
                04DF
                04DF    BAD_LIST:
                04DF    ;
                04DF    ; We have a bad list. This is branched to at the end of a
                04DF    ; name recognition list if a name is unrecognized. This
                04DF    ; checks for cases of
                04DF    ;
                04DF    ;       ,,      => INCOMPLETE
                04DF    ;       ,<EOL>  => INCOMPLETE
                04DF    ;       ,???    => ILLEGAL
                04DF    ;
                04DF          $DS_CLI CLI$K_COMMA,    0,      110$                    ; If not comma, bran
        00 8E   04DF                  .BYTE   CLI$K_COMMA,0
         0008'  04E1                  .WORD   110$-<.-2>
                04E3          $DS_CLI CLI$K_ERROR,    INC_CMD                         ; ,,
        02'80   04E3                  .BYTE   CLI$K_ERROR,INC_CMD
         0004'  04E5                  .WORD   30113$-<.-2>
                04E7    30113$:
                04E7
                04E7    110$:   $DS_CLI CLI$K_EOL,      0,      120$                    ; If not EOL, branch
        00 91   04E7                  .BYTE   CLI$K_EOL,0
         0008'  04E9                  .WORD   120$-<.-2>
                04EB          $DS_CLI CLI$K_ERROR     INC_CMD                         ; ,<EOL>
        02'80   04EB                  .BYTE   CLI$K_ERROR,INC_CMD
         0004'  04ED                  .WORD   30115$-<.-2>
                04EF    30115$:
                04EF
                04EF    120$:   $DS_CLI CLI$K_SPACE,    0,      130$                    ; Jump past any spac
        00 84   04EF                  .BYTE   CLI$K_SPACE,0
         0004'  04F1                  .WORD   130$-<.-2>
                04F3    130$:   $DS_CLI CLI$K_BR,       ADVANCE,140$                    ; Jump past illegal
        65'82   04F3                  .BYTE   CLI$K_BR,ADVANCE
         0004'  04F5                  .WORD   140$-<.-2>
                04F7    140$:   $DS_CLI CLI$K_ERROR,    ILL_CMD                         ; ILLEGAL
        01'80   04F7                  .BYTE   CLI$K_ERROR,ILL_CMD
         0004'  04F9                  .WORD   30118$-<.-2>
                04FB    30118$:
                04FB
                04FB
                04FB    BAD_QUAL:
                04FB    ;
                04FB    ; We have a bad qualifier. This is branched to at the end of a
                04FB    ; qualifier recognition sub-tree if a qualifier is not
                04FB    ; recognized. This checks for cases of
                04FB    ;
                04FB    ;       //      => INCOMPLETE
```

```
                        04FB  1012 ;       /<EOL> => INCOMPLETE
                        04FB  1013 ;       /???   => ILLEGAL
                        04FB  1014 ;
                        04FB  1015 ;
                        04FB  1016         $DS_CLI CLI$K_SLASH,    0,      110$                    ; If not /, branch
                00 90   04FB                       .BYTE   CLI$K_SLASH,0
                 0008'  04FD                       .WORD   110$-<.-2>
                        04FF  1017         $DS_CLI CLI$K_ERROR,    INC_CMD                         ; //
                02'80   04FF                       .BYTE   CLI$K_ERROR,INC_CMD
                 0004'  0501                       .WORD   30120$-<.-2>
                        0503       30120$:
                        0503  1018
                        0503  1019 110$:   $DS_CLI CLI$K_EOL,      0,      120$                    ; /???
                00 91   0503                       .BYTE   CLI$K_EOL,0
                 0008'  0505                       .WORD   120$-<.-2>
                        0507  1020         $DS_CLI CLI$K_ERROR     INC_CMD                         ; /<EOL>
                02'80   0507                       .BYTE   CLI$K_ERROR,INC_CMD
                 0004'  0509                       .WORD   30122$-<.-2>
                        050B       30122$:
                        050B  1021
                        050B  1022 120$:   $DS_CLI CLI$K_SPACE,    0,      130$                    ; Jump past any spac
                00 84   050B                       .BYTE   CLI$K_SPACE,0
                 0004'  050D                       .WORD   130$-<.-2>
                        050F  1023 130$:   $DS_CLI CLI$K_BR,       ADVANCE,140$                    ; Jump past illegal
                65'82   050F                       .BYTE   CLI$K_BR,ADVANCE
                 0004'  0511                       .WORD   140$-<.-2>
                        0513  1024 140$:   $DS_CLI CLI$K_ERROR,    ILL_CMD                         ; ILLEGAL
                01'80   0513                       .BYTE   CLI$K_ERROR,ILL_CMD
                 0004'  0515                       .WORD   30125$-<.-2>
                        0517       30125$:
```

```
                                0517  1026          .SUBTITLE        Flag list
                                0517  1027 ;
                                0517  1028 ; Read Flags list for SET FLAGS and CLEAR FLAGS.  The flags are:  ALl,
                                0517  1029 ; ASk_prompt, BEll, BInary, Halt, IE1, IE2, IE3, IES, Loop, Operator, Prompt,
                                0517  1030 ; Quick, Search, Trace, Verify.
                                0517  1031 ;
                                0517  1032 ; FLAGS_LIST:  [Flags  ]x , x , ...
                                0517  1033 ;
                                0517  1034 FLAGS_LIST:
                                0517  1035          $DS_CLI CLI$K_STRING,   0,      FLAGS_LOOP,<'FLAGS'>    ; Optional FLAGS
                        00 8B   0517                        .BYTE   CLI$K_STRING,0
                         000E'  0519                        .WORD   FLAGS_LOOP-<.-2>
              53 47 41 4C 46 00'  051B                        .ASCIC  'FLAGS'
                           05   051B
                                0521  1036          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ;
                        00 84   0521                        .BYTE   CLI$K_SPACE,0
                         FFA6   0523                        .WORD   ILLEGAL-<.-2>
                                0525  1037
                                0525  1038 FLAGS_LOOP:
                                0525  1039          $DS_CLI <^A"A">,        NOTNUF, 100$                    ; ALl           [108
                        18'41   0525                        .BYTE   ^A"A",NOTNUF
                         0021'  0527                        .WORD   100$-<.-2>
                                0529  1040          $DS_CLI CLI$K_STRING,   ALL,    90$,    <'LL'>          ;
                        41'8B   0529                        .BYTE   CLI$K_STRING,ALL
                         000B'  052B                        .WORD   90$-<.-2>
                     4C 4C 00'  052D                        .ASCIC  'LL'
                           02   052D
                                0530  1041          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
                        00 82   0530                        .BYTE   CLI$K_BR,0
                         00CC'  0532                        .WORD   END_FLAGS_LOOP-<.-2>
                                0534  1042
                                0534  1043 90$:     $DS_CLI CLI$K_STRING,   ASK_PROMPT,100$,<'SK_PROMPT'>   ; ASk_prompt    [108
                        42'8B   0534                        .BYTE   CLI$K_STRING,ASK_PROMPT
                         0012'  0536                        .WORD   100$-<.-2>
  54 50 4D 4F 52 50 5F 4B 53 00'  0538                        .ASCIC  'SK_PROMPT'
                           09   0538
                                0542  1044          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
                        00 82   0542                        .BYTE   CLI$K_BR,0
                         00BA'  0544                        .WORD   END_FLAGS_LOOP-<.-2>
                                0546  1045
                                0546  1046 100$:    $DS_CLI <^A"B">,        NOTNUF, 120$                    ; BEll
                        18'42   0546                        .BYTE   ^A"B",NOTNUF
                         001E'  0548                        .WORD   120$-<.-2>
                                054A  1047          $DS_CLI CLI$K_STRING,   BELL,   110$,   <'ELL'>         ;
                        44'8B   054A                        .BYTE   CLI$K_STRING,BELL
                         000C'  054C                        .WORD   110$-<.-2>
                   4C 4C 45 00'  054E                        .ASCIC  'ELL'
                           03   054E
                                0552  1048          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
                        00 82   0552                        .BYTE   CLI$K_BR,0
                         00AA'  0554                        .WORD   END_FLAGS_LOOP-<.-2>
                                0556  1049
                                0556  1050 110$:    $DS_CLI CLI$K_STRING,   BINARY, BAD_LIST,<'INARY'>      ; BInary        [45]
                        45'8B   0556                        .BYTE   CLI$K_STRING,BINARY
                         FF89   0558                        .WORD   BAD_LIST-<.-2>
              59 52 41 4E 49 00'  055A                        .ASCIC  'INARY'
                           05   055A
```

```
                    0560  1051          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;                [45]
          00 82     0560                        .BYTE   CLI$K_BR,0
           009C'    0562                        .WORD   END_FLAGS_LOOP-<.-2>
                    0564  1052
                    0564  1053 120$:    $DS_CLI CLI$K_STRING,   HALT,   130$,   <'HALT'>        ; Halt
          46'8B     0564                        .BYTE   CLI$K_STRING,HALT
           000D'    0566                        .WORD   130$-<.-2>
 54 4C 41 48 00'    0568                        .ASCIC  'HALT'
             04     0568
                    056D  1054          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
          00 82     056D                        .BYTE   CLI$K_BR,0
           008F'    056F                        .WORD   END_FLAGS_LOOP-<.-2>
                    0571  1055
                    0571  1056 130$:    $DS_CLI <^A"I">,        NOTNUF, 170$                    ; IE1
          18'49     0571                        .BYTE   ^A"I",NOTNUF
           0028'    0573                        .WORD   170$-<.-2>
                    0575  1057          $DS_CLI <^A"E">,        NOTNUF, BAD_LIST                ;
          18'45     0575                        .BYTE   ^A"E",NOTNUF
           FF6A     0577                        .WORD   BAD_LIST-<.-2>
                    0579  1058          $DS_CLI <^A"1">,        IE1,    140$                    ;
          47'31     0579                        .BYTE   ^A"1",IE1
           0008'    057B                        .WORD   140$-<.-2>
                    057D  1059          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
          00 82     057D                        .BYTE   CLI$K_BR,0
           007F'    057F                        .WORD   END_FLAGS_LOOP-<.-2>
                    0581  1060
                    0581  1061 140$:    $DS_CLI <^A"2">,        IE2,    150$                    ; IE2
          48'32     0581                        .BYTE   ^A"2",IE2
           0008'    0583                        .WORD   150$-<.-2>
                    0585  1062          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
          00 82     0585                        .BYTE   CLI$K_BR,0
           0077'    0587                        .WORD   END_FLAGS_LOOP-<.-2>
                    0589  1063
                    0589  1064 150$:    $DS_CLI <^A"3">,        IE3,    160$                    ; IE3
          49'33     0589                        .BYTE   ^A"3",IE3
           0008'    058B                        .WORD   160$-<.-2>
                    058D  1065          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
          00 82     058D                        .BYTE   CLI$K_BR,0
           006F'    058F                        .WORD   END_FLAGS_LOOP-<.-2>
                    0591  1066
                    0591  1067 160$:    $DS_CLI <^A"S">,        IES,    BAD_LIST                ; IES
          4A'53     0591                        .BYTE   ^A"S",IES
           FF4E     0593                        .WORD   BAD_LIST-<.-2>
                    0595  1068          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
          00 82     0595                        .BYTE   CLI$K_BR,0
           0067'    0597                        .WORD   END_FLAGS_LOOP-<.-2>
                    0599  1069
                    0599  1070 170$:    $DS_CLI CLI$K_STRING,   LOOP,   180$,   <'LOOP'>        ; Loop
          4B'8B     0599                        .BYTE   CLI$K_STRING,LOOP
           000D'    059B                        .WORD   180$-<.-2>
 50 4F 4F 4C 00'    059D                        .ASCIC  'LOOP'
             04     059D
                    05A2  1071          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
          00 82     05A2                        .BYTE   CLI$K_BR,0
           005A'    05A4                        .WORD   END_FLAGS_LOOP-<.-2>
                    05A6  1072
                    05A6  1073 180$:    $DS_CLI CLI$K_STRING,   OPER,   190$,   <'OPERATOR'>    ; Operator
```

```
                        4C'8B  05A6                        .BYTE   CLI$K_STRING,OPER
                         0011' 05A8                        .WORD   190$-<.-2>
     52 4F 54 41 52 45 50 4F 00' 05AA                        .ASCIC  'OPERATOR'
                           08  05AA
                               05B3  1074          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
                        00 82  05B3                        .BYTE   CLI$K_BR,0
                         0049' 05B5                        .WORD   END_FLAGS_LOOP-<.-2>
                               05B7  1075
                               05B7  1076 190$:    $DS_CLI CLI$K_STRING,   PROMPT, 200$,   <'PROMPT'>      ; Prompt
                        4D'8B  05B7                        .BYTE   CLI$K_STRING,PROMPT
                         000F' 05B9                        .WORD   200$-<.-2>
           54 50 4D 4F 52 50 00' 05BB                        .ASCIC  'PROMPT'
                           06  05BB
                               05C2  1077          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
                        00 82  05C2                        .BYTE   CLI$K_BR,0
                         003A' 05C4                        .WORD   END_FLAGS_LOOP-<.-2>
                               05C6  1078
                               05C6  1079 200$:    $DS_CLI CLI$K_STRING,   QUICK,  210$,   <'QUICK'>       ; Quick         [42]
                        4E'8B  05C6                        .BYTE   CLI$K_STRING,QUICK
                         000E' 05C8                        .WORD   210$-<.-2>
              4B 43 49 55 51 00' 05CA                        .ASCIC  'QUICK'
                           05  05CA
                               05D0  1080          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
                        00 82  05D0                        .BYTE   CLI$K_BR,0
                         002C' 05D2                        .WORD   END_FLAGS_LOOP-<.-2>
```

```
                                05D4  1082          .SUBTITLE       Device Names,  Event Flags
                                05D4  1083
                                05D4  1084 210$:    $DS_CLI CLI$K_STRING,   SEARCH, 220$,   <'SEARCH'>      ; Search
                          4F'8B 05D4                        .BYTE   CLI$K_STRING,SEARCH
                           000F' 05D6                       .WORD   220$-<.-2>
                48 43 52 41 45 53 00' 05D8                .ASCIC  'SEARCH'
                             06  05D8
                                05DF  1085          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;
                          00 82 05DF                        .BYTE   CLI$K_BR,0
                           001D' 05E1                       .WORD   END_FLAGS_LOOP-<.-2>
                                05E3  1086
                                05E3  1087 220$:    $DS_CLI CLI$K_STRING,   TRACE,  230$,   <'TRACE'>       ; Trace         [42]
                          50'8B 05E3                        .BYTE   CLI$K_STRING,TRACE
                           000E' 05E5                       .WORD   230$-<.-2>
                   45 43 41 52 54 00' 05E7                  .ASCIC  'TRACE'
                             05  05E7
                                05ED  1088          $DS_CLI CLI$K_BR,       0,      END_FLAGS_LOOP          ;               [42]
                          00 82 05ED                        .BYTE   CLI$K_BR,0
                           000F' 05EF                       .WORD   END_FLAGS_LOOP-<.-2>
                                05F1  1089
                                05F1  1090 230$:    $DS_CLI CLI$K_STRING,   VERIFY, BAD_LIST,<'VERIFY'>     ; Verify        [45]
                          51'8B 05F1                        .BYTE   CLI$K_STRING,VERIFY
                            FEEE 05F3                       .WORD   BAD_LIST-<.-2>
                59 46 49 52 45 56 00' 05F5                .ASCIC  'VERIFY'
                             06  05F5
                                05FC  1091
                                05FC  1092 END_FLAGS_LOOP:
                                05FC  1093          $DS_CLI CLI$K_COMMA,    NOTNUF, EOL                     ; ,
                          18'8E 05FC                        .BYTE   CLI$K_COMMA,NOTNUF
                            FEB3 05FE                       .WORD   EOL-<.-2>
                                0600  1094          $DS_CLI CLI$K_BR,       0,      FLAGS_LOOP              ; Get another flag
                          00 82 0600                        .BYTE   CLI$K_BR,0
                            FF25 0602                       .WORD   FLAGS_LOOP-<.-2>
                                0604  1095 ;
                                0604  1096 ; Finish scanning SELECT... and DESELECT... and SHOW DEVICE...
                                0604  1097 ; DEVICE NAME LIST:  device_name , device_name , ...
                                0604  1098 ;
                                0604  1099 DEVICE_NAMES:
                                0604  1100          $DS_CLI CLI$K_SPACE,    0,      140$                    ; Required/Optional
                          00 84 0604                        .BYTE   CLI$K_SPACE,0
                           0018' 0606                       .WORD   140$-<.-2>
                                0608  1101 109$:    $DS_CLI CLI$K_BR,       FILESPEC,110$
                          28'82 0608                        .BYTE   CLI$K_BR,FILESPEC
                           0004' 060A                       .WORD   110$-<.-2>
                                060C  1102 110$:    $DS_CLI CLI$K_SYMBOL,   FILEND, 140$                    ; Alpha-nums
                          29'8D 060C                        .BYTE   CLI$K_SYMBOL,FILEND
                           0010' 060E                       .WORD   140$-<.-2>
                                0610  1103          $DS_CLI <^A":">,        0,      120$                    ; Optional :
                          00 3A 0610                        .BYTE   ^A":",0
                           0004' 0612                       .WORD   120$-<.-2>
                                0614  1104 120$:    $DS_CLI CLI$K_COMMA,    DO_CMD, 150$                    ; Do comm. for each
                          7E'8E 0614                        .BYTE   CLI$K_COMMA,DO_CMD
                           000C' 0616                       .WORD   150$-<.-2>
                                0618  1105          $DS_CLI CLI$K_BR,       0,      109$                    ; Get next one
                          00 82 0618                        .BYTE   CLI$K_BR,0
                            FFF0 061A                       .WORD   109$-<.-2>
                                061C  1106
```

```
                        061C  1107 140$:   $DS_CLI CLI$K_BIF,      IF_REQUIRED,ILLEGAL                ; If name req., bran
                  6F'83 061C                       .BYTE   CLI$K_BIF,IF_REQUIRED
                   FEAB 061E                       .WORD   ILLEGAL-<.-2>
                        0620  1108 150$:   $DS_CLI CLI$K_BR,       ENUF,   EOL                        ; Assume EOL
                  17'82 0620                       .BYTE   CLI$K_BR,ENUF
                   FE8F 0622                       .WORD   EOL-<.-2>
                        0624  1109 ;
                        0624  1110 ; EVENT FLAGS:  [  Flags]  flag_num , flag_num , ...
                        0624  1111 ;               [  Flags]  All
                        0624  1112 ;
                        0624  1113 EVENT_FLAGS:
                        0624  1114         $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                    ; Required <SP>'s
                  00 84 0624                       .BYTE   CLI$K_SPACE,0
                   FEA3 0626                       .WORD   ILLEGAL-<.-2>
                        0628  1115         $DS_CLI CLI$K_STRING,   FLAGS,  130$,   <'FLAGS'>          ; "EVENT FLAGS"
                  3B'8B 0628                       .BYTE   CLI$K_STRING,FLAGS
                   000E' 062A                       .WORD   130$-<.-2>
     53 47 41 4C 46 00' 062C                       .ASCIC  'FLAGS'
                     05 062C
                        0632  1116         $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                    ; Required <SP>'s
                  00 84 0632                       .BYTE   CLI$K_SPACE,0
                   FE95 0634                       .WORD   ILLEGAL-<.-2>
                        0636  1117
                        0636  1118 130$:   $DS_CLI CLI$K_STRING,   ALL,    140$,   <'ALL'>            ; "EVENT FLAGS ALL"
                  41'8B 0636                       .BYTE   CLI$K_STRING,ALL
                   000C' 0638                       .WORD   140$-<.-2>
           4C 4C 41 00' 063A                       .ASCIC  'ALL'
                     03 063A
                        063E  1119 135$:   $DS_CLI CLI$K_BR,       0,      EOL
                  00 82 063E                       .BYTE   CLI$K_BR,0
                   FE71 0640                       .WORD   EOL-<.-2>
                        0642  1120
                        0642  1121 140$:   $DS_CLI CLI$K_DEC,      EFN,    BAD_LIST                   ; "EVENT FLAGS xx"
                  58'8A 0642                       .BYTE   CLI$K_DEC,EFN
                   FE9D 0644                       .WORD   BAD_LIST-<.-2>
                        0646  1122         $DS_CLI CLI$K_COMMA,    NOTNUF, 135$                       ; ,
                  18'8E 0646                       .BYTE   CLI$K_COMMA,NOTNUF
                   FFF8 0648                       .WORD   135$-<.-2>
                        064A  1123         $DS_CLI CLI$K_BR,       0,      140$
                  00 82 064A                       .BYTE   CLI$K_BR,0
                   FFF8 064C                       .WORD   140$-<.-2>
```

```
                        064E  1125          .SUBTITLE       File Specifications
                        064E  1126 ;
                        064E  1127 ;        ODS file specifier                                                      ;G:4
                        064E  1128 ;                                                                                ;G:4
                        064E  1129 ;        [device_name:][directory_name]file_name.file_type;file_version           ;G:14
                        064E  1130 ;
                        064E  1131 ;        ULTRIX file specifier                                                   ;G:4
                        064E  1132 ;                                                                                ;G:4
                        064E  1133 ;        [device_name][(0,partition)][name1/name2 - - -]                         ;G:4
                        064E  1134 ;                                                                                ;G:4
                        064E  1135 GET_FILE_SPEC:
                        064E  1136          $DS_CLI CLI$K_BR,       FILESPEC,100$
                28'82   064E                .BYTE   CLI$K_BR,FILESPEC
                 0004'  0650                .WORD   100$-<.-2>
                        0652  1137 100$:    $DS_CLI CLI$K_BIF,      ULTRIX_ODS, 300$                        ; Check if U
                1C'83   0652                .BYTE   CLI$K_BIF,ULTRIX_ODS
                 005C'  0654                .WORD   300$-<.-2>
                        0656  1138
                        0656  1139          $DS_CLI CLI$K_FILE,     0,      110$                            ; Could be Device, o
                00 95   0656                .BYTE   CLI$K_FILE,0
                 0008'  0658                .WORD   110$-<.-2>
                        065A  1140          $DS_CLI <^A":">,        0,      140$                            ; Was Device Name.
                00 3A   065A                .BYTE   ^A":",0
                 0034'  065C                .WORD   140$-<.-2>
                        065E  1141
                        065E  1142 110$:    $DS_CLI <^A"[">,        0,      130$                            ; Get Directory Coul
                00 5B   065E                .BYTE   ^A"[",0
                 002C'  0660                .WORD   130$-<.-2>
                        0662  1143          $DS_CLI CLI$K_FILE,     0,      120$                            ; FILE spec fits bot
                00 95   0662                .BYTE   CLI$K_FILE,0
                 0024'  0664                .WORD   120$-<.-2>
                        0666  1144          $DS_CLI CLI$K_COMMA,    0,      115$                            ; for an UIC. ;G:27
                00 8E   0666                .BYTE   CLI$K_COMMA,0
                 0014'  0668                .WORD   115$-<.-2>
                        066A  1145          $DS_CLI CLI$K_FILE,     0,      ILLEGAL                         ;               ;G:28
                00 95   066A                .BYTE   CLI$K_FILE,0
                 FE5D   066C                .WORD   ILLEGAL-<.-2>
                        066E  1146          $DS_CLI CLI$K_BR,       0,      120$
                00 82   066E                .BYTE   CLI$K_BR,0
                 0018'  0670                .WORD   120$-<.-2>
                        0672  1147
                        0672  1148 112$:    $DS_CLI <^A"]">,        0,      117$                            ; Check for [] Direc
                00 5D   0672                .BYTE   ^A"]",0
                 000C'  0674                .WORD   117$-<.-2>
                        0676  1149          $DS_CLI CLI$K_BR,       0,      130$                            ; Go Get File Name.
                00 82   0676                .BYTE   CLI$K_BR,0
                 0014'  0678                .WORD   130$-<.-2>
                        067A  1150                                                                              ;G:14
                        067A  1151 115$:    $DS_CLI <^A".">,        0,      120$                            ; Check for more Dir
                00 2E   067A                .BYTE   ^A".",0
                 000C'  067C                .WORD   120$-<.-2>
                        067E  1152 117$:    $DS_CLI CLI$K_FILE,     0,      ILLEGAL                         ; Get Next Dir Name.
                00 95   067E                .BYTE   CLI$K_FILE,0
                 FE49   0680                .WORD   ILLEGAL-<.-2>
                        0682  1153          $DS_CLI CLI$K_BR,       0,      115$                            ; Get More.
                00 82   0682                .BYTE   CLI$K_BR,0
                 FFF8   0684                .WORD   115$-<.-2>
```

```
                 0686  1154
                 0686  1155 120$:   $DS_CLI <^A"]">,        0,      ILLEGAL                  ; End the Dir.
          00 5D  0686                        .BYTE   ^A"]",0
           FE41  0688                        .WORD   ILLEGAL-<.-2>
                 068A  1156
                 068A  1157 130$:   $DS_CLI CLI$K_FILE,     0,      140$                     ; Get File Name. [11
          00 95  068A                        .BYTE   CLI$K_FILE,0
           0004' 068C                        .WORD   140$-<.-2>
                 068E  1158
                 068E  1159 140$:   $DS_CLI <^A".">,        0,      150$                     ; Check for Type.
          00 2E  068E                        .BYTE   ^A".",0
           0008' 0690                        .WORD   150$-<.-2>
                 0692  1160         $DS_CLI CLI$K_FILE,     0,      150$                     ; Get Type. [112]
          00 95  0692                        .BYTE   CLI$K_FILE,0
           0004' 0694                        .WORD   150$-<.-2>
                 0696  1161
                 0696  1162 150$:   $DS_CLI <^A";">,        0,      190$                     ; Check for Version.
          00 3B  0696                        .BYTE   ^A";",0
           0008' 0698                        .WORD   190$-<.-2>
                 069A  1163         $DS_CLI CLI$K_DEC,      0,      190$
          00 8A  069A                        .BYTE   CLI$K_DEC,0
           0004' 069C                        .WORD   190$-<.-2>
                 069E  1164
                 069E  1165 190$:   $DS_CLI CLI$K_BR,       FILEND,191$                      ; End Of File String
          29'82  069E                        .BYTE   CLI$K_BR,FILEND
           0004' 06A0                        .WORD   191$-<.-2>
                 06A2  1166 191$:   $DS_CLI CLI$K_BIF,      IF_FILE_SPEC,200$                ; If file-spec, bran
          70'83  06A2                        .BYTE   CLI$K_BIF,IF_FILE_SPEC
           0008' 06A4                        .WORD   200$-<.-2>
                 06A6  1167         $DS_CLI CLI$K_BIF,      IF_REQUIRED,INCOMPLETE           ; If required, branc
          6F'83  06A6                        .BYTE   CLI$K_BIF,IF_REQUIRED
           FE35  06A8                        .WORD   INCOMPLETE-<.-2>
                 06AA  1168 200$:   $DS_CLI CLI$K_RETURN,   ENUF                             ; Done, Return to Co
          17'93  06AA                        .BYTE   CLI$K_RETURN,ENUF
           0004' 06AC                        .WORD   30204$-<.-2>
                 06AE       30204$:
                 06AE  1169 ;
                 06AE  1170 ; ULTRIX FILES STRING PARSE
                 06AE  1171 ;
                 06AE  1172 300$:   $DS_CLI CLI$K_FILE,     0,      308$                     ; Alpha-nums    [106
          00 95  06AE                        .BYTE   CLI$K_FILE,0
           0008' 06B0                        .WORD   308$-<.-2>
                 06B2  1173         $DS_CLI <^A":">,        0,      350$                     ; :
          00 3A  06B2                        .BYTE   ^A":",0
           008C' 06B4                        .WORD   350$-<.-2>
                 06B6  1174 308$:   $DS_CLI <^A"(">,        0,      350$                     ; Partition.    [106
          00 28  06B6                        .BYTE   ^A"(",0
           0088' 06B8                        .WORD   350$-<.-2>
                 06BA  1175 309$:   $DS_CLI CLI$K_DEC,      0,      INCOMPLETE               ; Search for    [106
          00 8A  06BA                        .BYTE   CLI$K_DEC,0
           FE21  06BC                        .WORD   INCOMPLETE-<.-2>
                 06BE  1176 310$:   $DS_CLI CLI$K_COMMA,    0,      INCOMPLETE               ; (##,##)       [94]
          00 8E  06BE                        .BYTE   CLI$K_COMMA,0
           FE1D  06C0                        .WORD   INCOMPLETE-<.-2>
                 06C2  1177         $DS_CLI CLI$K_DEC,      0,      320$                     ;               [117
          00 8A  06C2                        .BYTE   CLI$K_DEC,0
           0008' 06C4                        .WORD   320$-<.-2>
```

```
                        06C6  1178          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06C6                        .BYTE   CLI$K_BR,0
 0070'                  06C8                        .WORD   338$-<.-2>
                        06CA  1179 320$:    $DS_CLI <^A"A">,        0,      321$                    ;                       [117
00 41                   06CA                        .BYTE   ^A"A",0
 0008'                  06CC                        .WORD   321$-<.-2>
                        06CE  1180          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06CE                        .BYTE   CLI$K_BR,0
 0068'                  06D0                        .WORD   338$-<.-2>
                        06D2  1181 321$:    $DS_CLI <^A"a">,        0,      322$                    ;                       [117
00 61                   06D2                        .BYTE   ^A"a",0
 0008'                  06D4                        .WORD   322$-<.-2>
                        06D6  1182          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06D6                        .BYTE   CLI$K_BR,0
 0060'                  06D8                        .WORD   338$-<.-2>
                        06DA  1183 322$:    $DS_CLI <^A"B">,        0,      323$                    ;                       [117
00 42                   06DA                        .BYTE   ^A"B",0
 0008'                  06DC                        .WORD   323$-<.-2>
                        06DE  1184          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06DE                        .BYTE   CLI$K_BR,0
 0058'                  06E0                        .WORD   338$-<.-2>
                        06E2  1185 323$:    $DS_CLI <^A"b">,        0,      324$                    ;                       [117
00 62                   06E2                        .BYTE   ^A"b",0
 0008'                  06E4                        .WORD   324$-<.-2>
                        06E6  1186          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06E6                        .BYTE   CLI$K_BR,0
 0050'                  06E8                        .WORD   338$-<.-2>
                        06EA  1187 324$:    $DS_CLI <^A"C">,        0,      325$                    ;                       [117
00 43                   06EA                        .BYTE   ^A"C",0
 0008'                  06EC                        .WORD   325$-<.-2>
                        06EE  1188          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06EE                        .BYTE   CLI$K_BR,0
 0048'                  06F0                        .WORD   338$-<.-2>
                        06F2  1189 325$:    $DS_CLI <^A"c">,        0,      326$                    ;                       [117
00 63                   06F2                        .BYTE   ^A"c",0
 0008'                  06F4                        .WORD   326$-<.-2>
                        06F6  1190          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06F6                        .BYTE   CLI$K_BR,0
 0040'                  06F8                        .WORD   338$-<.-2>
                        06FA  1191 326$:    $DS_CLI <^A"D">,        0,      327$                    ;                       [117
00 44                   06FA                        .BYTE   ^A"D",0
 0008'                  06FC                        .WORD   327$-<.-2>
                        06FE  1192          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   06FE                        .BYTE   CLI$K_BR,0
 0038'                  0700                        .WORD   338$-<.-2>
                        0702  1193 327$:    $DS_CLI <^A"d">,        0,      328$                    ;                       [117
00 64                   0702                        .BYTE   ^A"d",0
 0008'                  0704                        .WORD   328$-<.-2>
                        0706  1194          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   0706                        .BYTE   CLI$K_BR,0
 0030'                  0708                        .WORD   338$-<.-2>
                        070A  1195 328$:    $DS_CLI <^A"E">,        0,      329$                    ;                       [117
00 45                   070A                        .BYTE   ^A"E",0
 0008'                  070C                        .WORD   329$-<.-2>
                        070E  1196          $DS_CLI CLI$K_BR,       0,      338$                    ;                       [117
00 82                   070E                        .BYTE   CLI$K_BR,0
 0028'                  0710                        .WORD   338$-<.-2>
```

```
                 0712  1197 329$:   $DS_CLI <^A"e">,        0,      330$                    ;                [117
        00 65    0712                        .BYTE   ^A"e",0
         0008'   0714                        .WORD   330$-<.-2>
                 0716  1198         $DS_CLI CLI$K_BR,       0,      338$                    ;                [117
        00 82    0716                        .BYTE   CLI$K_BR,0
         0020'   0718                        .WORD   338$-<.-2>
                 071A  1199 330$:   $DS_CLI <^A"F">,        0,      331$                    ;                [117
        00 46    071A                        .BYTE   ^A"F",0
         0008'   071C                        .WORD   331$-<.-2>
                 071E  1200         $DS_CLI CLI$K_BR,       0,      338$                    ;                [117
        00 82    071E                        .BYTE   CLI$K_BR,0
         0018'   0720                        .WORD   338$-<.-2>
                 0722  1201 331$:   $DS_CLI <^A"f">,        0,      332$                    ;                [117
        00 66    0722                        .BYTE   ^A"f",0
         0008'   0724                        .WORD   332$-<.-2>
                 0726  1202         $DS_CLI CLI$K_BR,       0,      338$                    ;                [117
        00 82    0726                        .BYTE   CLI$K_BR,0
         0010'   0728                        .WORD   338$-<.-2>
                 072A  1203 332$:   $DS_CLI <^A"G">,        0,      333$                    ;                [117
        00 47    072A                        .BYTE   ^A"G",0
         0008'   072C                        .WORD   333$-<.-2>
                 072E  1204         $DS_CLI CLI$K_BR,       0,      338$                    ;                [117
        00 82    072E                        .BYTE   CLI$K_BR,0
         0008'   0730                        .WORD   338$-<.-2>
                 0732  1205 333$:   $DS_CLI <^A"g">,        0,      INCOMPLETE              ;                [117
        00 67    0732                        .BYTE   ^A"g",0
         FDA9    0734                        .WORD   INCOMPLETE-<.-2>
                 0736  1206 338$:   $DS_CLI <^A")">,        0,      INCOMPLETE              ; Partition.     [94]
        00 29    0736                        .BYTE   ^A")",0
         FDA5    0738                        .WORD   INCOMPLETE-<.-2>
                 073A  1207 ;                                                                                [113
                 073A  1208 ; file [/file---] [. [file]]                                                     [113
                 073A  1209 ;  "." and .file are also accepted - these are allowable Ultrix file names       [113
                 073A  1210 ;                                                                                [113
                 073A  1211 340$:   $DS_CLI CLI$K_FILE,     0,      350$                    ; file           [113
        00 95    073A                        .BYTE   CLI$K_FILE,0
         0004'   073C                        .WORD   350$-<.-2>
                 073E  1212 350$:   $DS_CLI CLI$K_EOL,      0,      360$                    ; file eol       [113
        00 91    073E                        .BYTE   CLI$K_EOL,0
         0008'   0740                        .WORD   360$-<.-2>
                 0742  1213         $DS_CLI CLI$K_BR,       0,      190$                    ; ok, exit       [113
        00 82    0742                        .BYTE   CLI$K_BR,0
         FF5C    0744                        .WORD   190$-<.-2>
                 0746  1214 360$:   $DS_CLI <^A"/">,        0,      380$                    ; file/          [113
        00 2F    0746                        .BYTE   ^A"/",0
         0014'   0748                        .WORD   380$-<.-2>
                 074A  1215         $DS_CLI CLI$K_EOL,      NOTNUF, 370$                    ; get rest       [113
        18'91    074A                        .BYTE   CLI$K_EOL,NOTNUF
         0008'   074C                        .WORD   370$-<.-2>
                 074E  1216         $DS_CLI CLI$K_BR,       0,      ILLEGAL                 ;  else error    [113
        00 82    074E                        .BYTE   CLI$K_BR,0
         FD79    0750                        .WORD   ILLEGAL-<.-2>
                 0752  1217 370$:   $DS_CLI CLI$K_FILE,     0,      ILLEGAL                 ; file/file      [113
        00 95    0752                        .BYTE   CLI$K_FILE,0
         FD75    0754                        .WORD   ILLEGAL-<.-2>
                 0756  1218         $DS_CLI CLI$K_BR,       0,      350$                    ; look for more [113
        00 82    0756                        .BYTE   CLI$K_BR,0
```

```
                FFE8  0758                              .WORD   350$-<.-2>
                      075A  1219 380$:   $DS_CLI <^A".">,      0,      ILLEGAL                 ; file.         [113
              00 2E   075A                              .BYTE   ^A".",0
               FD6D   075C                              .WORD   ILLEGAL-<.-2>
                      075E  1220         $DS_CLI CLI$K_EOL,     0,      390$                    ; file. eol     [113
              00 91   075E                              .BYTE   CLI$K_EOL,0
               0008'  0760                              .WORD   390$-<.-2>
                      0762  1221         $DS_CLI CLI$K_BR,      0,      190$                    ; ok, exit      [113
              00 82   0762                              .BYTE   CLI$K_BR,0
               FF3C   0764                              .WORD   190$-<.-2>
                      0766  1222 390$:   $DS_CLI CLI$K_FILE,    0,      ILLEGAL                 ; file.file     [113
              00 95   0766                              .BYTE   CLI$K_FILE,0
               FD61   0768                              .WORD   ILLEGAL-<.-2>
                      076A  1223         $DS_CLI CLI$K_EOL,     0,      ILLEGAL                 ; nothing more  [113
              00 91   076A                              .BYTE   CLI$K_EOL,0
               FD5D   076C                              .WORD   ILLEGAL-<.-2>
                      076E  1224         $DS_CLI CLI$K_BR,      0,      190$                    ; exit          [113
              00 82   076E                              .BYTE   CLI$K_BR,0
               FF30   0770                              .WORD   190$-<.-2>
```

```
                0772  1226          .SUBTITLE ADDRESS Parrameter.
                0772  1227 ;
                0772  1228 ; Read a generic address for Examine, Deposit...
                0772  1229 ; ADDRESS:   address
                0772  1230 ; address is one of: AP, FP, PC, PSl, Pxx, Rxx, SP, number
                0772  1231 ;
                0772  1232 GET_ADDRESS:
                0772  1233 ;
                0772  1234 ; AP    This could be AP, or it could be Axxx = address.
                0772  1235 ;
                0772  1236          $DS_CLI <^A"A">,      REG12,    120$                    ; AP
        5E'41   0772                        .BYTE   ^A"A",REG12
         000C'  0774                        .WORD   120$-<.-2>
                0776  1237          $DS_CLI <^A"P">,      REGP,     200$                    ; [121]       ;G:14
        63'50   0776                        .BYTE   ^A"P",REGP
         0053'  0778                        .WORD   200$-<.-2>
                077A  1238          $DS_CLI CLI$K_RETURN                                    ; Return      ;G:14
        00 93   077A                        .BYTE   CLI$K_RETURN,0
         0004'  077C                        .WORD   30256$-<.-2>
                077E        30256$:
                077E  1239 ;
                077E  1240 ; FP    This could be FP, or it could be Fxxx = address.
                077E  1241 ;
                077E  1242
                077E  1243 120$:    $DS_CLI <^A"F">,      REG13,    130$                    ; FP
        5F'46   077E                        .BYTE   ^A"F",REG13
         000C'  0780                        .WORD   130$-<.-2>
                0782  1244          $DS_CLI <^A"P">,      REGP,     200$                    ; [121]       ;G:14
        63'50   0782                        .BYTE   ^A"P",REGP
         0047'  0784                        .WORD   200$-<.-2>
                0786  1245          $DS_CLI CLI$K_RETURN                                    ; Return
        00 93   0786                        .BYTE   CLI$K_RETURN,0
         0004'  0788                        .WORD   30259$-<.-2>
                078A        30259$:
                078A  1246 ;
                078A  1247 ; PC
                078A  1248 ;
                078A  1249 130$:    $DS_CLI <^A"P">,        0,      160$                    ; PC
        00 50   078A                        .BYTE   ^A"P",0
         001F'  078C                        .WORD   160$-<.-2>
                078E  1250          $DS_CLI <^A"C">,      REG15,    140$                    ;
        61'43   078E                        .BYTE   ^A"C",REG15
         0008'  0790                        .WORD   140$-<.-2>
                0792  1251          $DS_CLI CLI$K_RETURN                                    ; Return
        00 93   0792                        .BYTE   CLI$K_RETURN,0
         0004'  0794                        .WORD   30262$-<.-2>
                0796        30262$:
                0796  1252 ;
                0796  1253 ; PSL
                0796  1254 ;
                0796  1255 140$:    $DS_CLI CLI$K_STRING, REG16,    150$,   <'SL'>          ; PSL
        62'8B   0796                        .BYTE   CLI$K_STRING,REG16
         000B'  0798                        .WORD   150$-<.-2>
     4C 53 00'  079A                        .ASCIC  'SL'
           02   079A
                079D  1256          $DS_CLI CLI$K_RETURN                                    ; Return
        00 93   079D                        .BYTE   CLI$K_RETURN,0
```

```
                0004' 079F                              .WORD   30264$-<.-2>
                      07A1          30264$:
                      07A1    1257 ;
                      07A1    1258 ; Pxx    processor register
                      07A1    1259 ;
                      07A1    1260 150$:    $DS_CLI CLI$K_NUM,    PREGN,    ILLEGAL                 ; Pxx
              32'85   07A1                            .BYTE   CLI$K_NUM,PREGN
               FD26   07A3                            .WORD   ILLEGAL-<.-2>
                      07A5    1261          $DS_CLI CLI$K_RETURN                                      ; Return
              00 93   07A5                            .BYTE   CLI$K_RETURN,0
               0004'  07A7                            .WORD   30266$-<.-2>
                      07A9          30266$:
                      07A9    1262 ;
                      07A9    1263 ; Rnum   general register
                      07A9    1264 ;
                      07A9    1265 160$:    $DS_CLI <^A"R">,      REGN,     170$                    ; Rxx
              5D'52   07A9                            .BYTE   ^A"R",REGN
               000C'  07AB                            .WORD   170$-<.-2>
                      07AD    1266          $DS_CLI CLI$K_DEC,    ADDRESS,  ILLEGAL                 ;
              3D'8A   07AD                            .BYTE   CLI$K_DEC,ADDRESS
               FD1A   07AF                            .WORD   ILLEGAL-<.-2>
                      07B1    1267          $DS_CLI CLI$K_RETURN                                      ; Return
              00 93   07B1                            .BYTE   CLI$K_RETURN,0
               0004'  07B3                            .WORD   30269$-<.-2>
                      07B5          30269$:
                      07B5    1268 ;
                      07B5    1269 ; SP
                      07B5    1270 ;
                      07B5    1271 170$:    $DS_CLI <^A"S">,      REG14,    180$                    ; SP
              60'53   07B5                            .BYTE   ^A"S",REG14
               000C'  07B7                            .WORD   180$-<.-2>
                      07B9    1272          $DS_CLI <^A"P">,      REGP,     ILLEGAL                 ;
              63'50   07B9                            .BYTE   ^A"P",REGP
               FD0E   07BB                            .WORD   ILLEGAL-<.-2>
                      07BD    1273          $DS_CLI CLI$K_RETURN                                      ; Return
              00 93   07BD                            .BYTE   CLI$K_RETURN,0
               0004'  07BF                            .WORD   30272$-<.-2>
                      07C1          30272$:
                      07C1    1274 ;
                      07C1    1275 ; address
                      07C1    1276 ;
                      07C1    1277 180$:    $DS_CLI CLI$K_NUM,    ADDRESS,  ILLEGAL                 ; Address is data
              3D'85   07C1                            .BYTE   CLI$K_NUM,ADDRESS
               FD06   07C3                            .WORD   ILLEGAL-<.-2>
                      07C5    1278 190$:    $DS_CLI CLI$K_RETURN                                      ; Return.
              00 93   07C5                            .BYTE   CLI$K_RETURN,0
               0004'  07C7                            .WORD   30274$-<.-2>
                      07C9          30274$:
                      07C9    1279 200$:    $DS_CLI CLI$K_BR,     BACKUP,   180$                    ; Backup  and try re
              64'82   07C9                            .BYTE   CLI$K_BR,BACKUP
               FFF8   07CB                            .WORD   180$-<.-2>
```

```
                        07CD  1281          .SUBTITLE       Start and Run Qualifiers
                        07CD  1282 ;
                        07CD  1283 START_RUN_QUALS:
                        07CD  1284 ;
                        07CD  1285 ; Qualifiers for the START and RUN commands.
                        07CD  1286 ; This allows optional spaces before and after the :'s and the /'s.
                        07CD  1287 ;
                        07CD  1288          $DS_CLI CLI$K_SLASH,    NOTNUF, 200$                    ; Read Qualifiers
                18'90   07CD                        .BYTE   CLI$K_SLASH,NOTNUF
                 00C2'  07CF                        .WORD   200$-<.-2>
                        07D1  1289 ;                                                                                [62]
                        07D1  1290 ; Start or Run Qualifier:  / Debug                                               [62]
                        07D1  1291 ;                                                                                [62]
                        07D1  1292          $DS_CLI CLI$K_STRING,   DEBUG_SWITCH,90$,<'DEBUG'>      ; /DEBUG       [62]
                0D'8B   07D1                        .BYTE   CLI$K_STRING,DEBUG_SWITCH
                 000E'  07D3                        .WORD   90$-<.-2>
           47 55 42 45 44 00'  07D5                 .ASCIC  'DEBUG'
                   05   07D5
                        07DB  1293          $DS_CLI CLI$K_BR,       0,      START_RUN_QUALS         ; NEXT QUALIFIER[62]
                00 82   07DB                        .BYTE   CLI$K_BR,0
                 FFF2   07DD                        .WORD   START_RUN_QUALS-<.-2>
                        07DF  1294 ;
                        07DF  1295 ; Start or Run Qualifier:  / Passes : dec_passes
                        07DF  1296 ;
                        07DF  1297 90$:     $DS_CLI CLI$K_STRING,   NOTNUF, 100$,   <'PASSES'>      ; /PASSES      [42]
                18'8B   07DF                        .BYTE   CLI$K_STRING,NOTNUF
                 0017'  07E1                        .WORD   100$-<.-2>
        53 45 53 53 41 50 00'  07E3                 .ASCIC  'PASSES'
                   06   07E3
                        07EA  1298          $DS_CLI CLI$K_VALUE,    NOTNUF, ILLEGAL                 ; : or =
                18'8F   07EA                        .BYTE   CLI$K_VALUE,NOTNUF
                 FCDD   07EC                        .WORD   ILLEGAL-<.-2>
                        07EE  1299          $DS_CLI CLI$K_DEC,      PASS,   ILLEGAL                 ; /PASSES:decmal
                34'8A   07EE                        .BYTE   CLI$K_DEC,PASS
                 FCD9   07F0                        .WORD   ILLEGAL-<.-2>
                        07F2  1300          $DS_CLI CLI$K_BR,       0,      START_RUN_QUALS
                00 82   07F2                        .BYTE   CLI$K_BR,0
                 FFDB   07F4                        .WORD   START_RUN_QUALS-<.-2>
                        07F6  1301 ;
                        07F6  1302 ; Start or Run Qualifier:  / Qa
                        07F6  1303 ;
                        07F6  1304 100$:    $DS_CLI CLI$K_STRING    QA,     110$,   <'QA'>          ; /QA          [42]
                35'8B   07F6                        .BYTE   CLI$K_STRING,QA
                 000B'  07F8                        .WORD   110$-<.-2>
              41 51 00'  07FA                       .ASCIC  'QA'
                   02   07FA
                        07FD  1305          $DS_CLI CLI$K_BR        0,      START_RUN_QUALS         ;              [42]
                00 82   07FD                        .BYTE   CLI$K_BR,0
                 FFD0   07FF                        .WORD   START_RUN_QUALS-<.-2>
                        0801  1306 ;
                        0801  1307 ; Start or Run Qualifier:  / SEction : sym_secname
                        0801  1308 ;
                        0801  1309 110$:    $DS_CLI <^A"S">,        NOTNUF, 125$                    ; S
                18'53   0801                        .BYTE   ^A"S",NOTNUF
                 0032'  0803                        .WORD   125$-<.-2>
                        0805  1310
                        0805  1311          $DS_CLI CLI$K_STRING,   NOTNUF, 120$,   <'ECTION'>      ; /SECTION
```

```
                      18'8B  0805                    .BYTE   CLI$K_STRING,NOTNUF
                       0017' 0807                    .WORD   120$-<.-2>
          4E 4F 49 54 43 45 00' 0809                 .ASCIC  'ECTION'
                         06  0809
                             0810  1312        $DS_CLI CLI$K_VALUE,    NOTNUF, ILLEGAL                 ; : or =
                      18'8F  0810                    .BYTE   CLI$K_VALUE,NOTNUF
                       FCB7  0812                    .WORD   ILLEGAL-<.-2>
                             0814  1313        $DS_CLI CLI$K_SYMBOL,   SECTION,ILLEGAL                 ;                [37]
                      33'8D  0814                    .BYTE   CLI$K_SYMBOL,SECTION
                       FCB3  0816                    .WORD   ILLEGAL-<.-2>
                             0818  1314        $DS_CLI CLI$K_BR,       0,      START_RUN_QUALS         ;                [42]
                      00 82  0818                    .BYTE   CLI$K_BR,0
                       FFB5  081A                    .WORD   START_RUN_QUALS-<.-2>
                             081C  1315 ;
                             081C  1316 ; Start or Run Qualifier:  / SUbtest : dec_nsubtest
                             081C  1317 ;
                             081C  1318 120$:    $DS_CLI CLI$K_STRING,   NOTNUF, BAD_QUAL,<'UBTEST'>     ; /SUBTEST
                      18'8B  081C                    .BYTE   CLI$K_STRING,NOTNUF
                       FCDF  081E                    .WORD   BAD_QUAL-<.-2>
          54 53 45 54 42 55 00' 0820                 .ASCIC  'UBTEST'
                         06  0820
                             0827  1319        $DS_CLI CLI$K_VALUE,    NOTNUF, ILLEGAL                 ; : or =
                      18'8F  0827                    .BYTE   CLI$K_VALUE,NOTNUF
                       FCA0  0829                    .WORD   ILLEGAL-<.-2>
                             082B  1320        $DS_CLI CLI$K_DEC,      SUBTEST,ILLEGAL
                      2E'8A  082B                    .BYTE   CLI$K_DEC,SUBTEST
                       FC9C  082D                    .WORD   ILLEGAL-<.-2>
                             082F  1321        $DS_CLI CLI$K_BR,       0,      START_RUN_QUALS         ;                [42]
                      00 82  082F                    .BYTE   CLI$K_BR,0
                       FF9E  0831                    .WORD   START_RUN_QUALS-<.-2>
                             0833  1322 ;
                             0833  1323 ; Start or Run Qualifier:  /TIME:dddd-hh:mm:ss.cc             ; begin      [76]
                             0833  1324 ;
                             0833  1325 125$:    $DS_CLI <^A"T">,        NOTNUF, BAD_QUAL            ; T          [37]
                      18'54  0833                    .BYTE   ^A"T",NOTNUF
                       FCC8  0835                    .WORD   BAD_QUAL-<.-2>
                             0837  1326
                             0837  1327 130$:    $DS_CLI CLI$K_STRING,   NOTNUF, 140$,<'IME'>        ; /TIME      [76]
                      18'8B  0837                    .BYTE   CLI$K_STRING,NOTNUF
                       003C' 0839                    .WORD   140$-<.-2>
                45 4D 49 00' 083B                    .ASCIC  'IME'
                         03  083B
                             083F  1328        $DS_CLI CLI$K_VALUE,    NOTNUF, ILLEGAL                 ; : or =     [76]
                      18'8F  083F                    .BYTE   CLI$K_VALUE,NOTNUF
                       FC88  0841                    .WORD   ILLEGAL-<.-2>
                             0843  1329        $DS_CLI CLI$K_BR,       BEGINTIME, 131$                 ; Mark start[76]
                      8D'82  0843                    .BYTE   CLI$K_BR,BEGINTIME
                       0004' 0845                    .WORD   131$-<.-2>
                             0847  1330                                                              ; of string [76]
                             0847  1331 131$:    $DS_CLI CLI$K_DEC,      0,      133$                ; dddd       [76]
                      00 8A  0847                    .BYTE   CLI$K_DEC,0
                       000C' 0849                    .WORD   133$-<.-2>
                             084B  1332        $DS_CLI <^A"-">,        0,      133$                ; SP         [76]
                      00 2D  084B                    .BYTE   ^A"-",0
                       0008' 084D                    .WORD   133$-<.-2>
                             084F  1333 132$:    $DS_CLI CLI$K_DEC,      0,      133$                ; hh         [76]
                      00 8A  084F                    .BYTE   CLI$K_DEC,0
```

```
                0004' 0851                              .WORD   133$-<.-2>
                      0853  1334 133$:    $DS_CLI CLI$K_VALUE,    0,      134$                    ; : or =    [76]
             00 8F    0853                              .BYTE   CLI$K_VALUE,0
              0004'   0855                              .WORD   134$-<.-2>
                      0857  1335 134$:    $DS_CLI CLI$K_DEC,      0,      135$                    ; mm        [76]
             00 8A    0857                              .BYTE   CLI$K_DEC,0
              0004'   0859                              .WORD   135$-<.-2>
                      085B  1336 135$:    $DS_CLI CLI$K_VALUE,    0,      136$                    ; : or =    [76]
             00 8F    085B                              .BYTE   CLI$K_VALUE,0
              0004'   085D                              .WORD   136$-<.-2>
                      085F  1337 136$:    $DS_CLI CLI$K_DEC,      0,      137$                    ; ss        [76]
             00 8A    085F                              .BYTE   CLI$K_DEC,0
              0004'   0861                              .WORD   137$-<.-2>
                      0863  1338 137$:    $DS_CLI <^A".">,        0,      138$                    ; .         [76]
             00 2E    0863                              .BYTE   ^A".",0
              0004'   0865                              .WORD   138$-<.-2>
                      0867  1339 138$:    $DS_CLI CLI$K_DEC,      0,      139$                    ; cc        [76]
             00 8A    0867                              .BYTE   CLI$K_DEC,0
              0004'   0869                              .WORD   139$-<.-2>
                      086B  1340 139$:    $DS_CLI CLI$K_BIF,      TIME,   ILLEGAL                 ; Check the [76]
             8E'83    086B                              .BYTE   CLI$K_BIF,TIME
              FC5C    086D                              .WORD   ILLEGAL-<.-2>
                      086F  1341                                                                    ; string    [76]
                      086F  1342    $DS_CLI CLI$K_BR,       0,      START_RUN_QUALS
             00 82    086F                              .BYTE   CLI$K_BR,0
              FF5E    0871                              .WORD   START_RUN_QUALS-<.-2>
```

```
                        0873  1344          .SUBTITLE       BREAKPOINT Start or Run Qualifiers, DEVICE Qualifiers.
                        0873  1345 ;
                        0873  1346 ; Start or Run Qualifier:  / TEst : dec_start : dec_end
                        0873  1347 ;
                        0873  1348 140$:    $DS_CLI CLI$K_STRING,   NOTNUF, 141$,   <'EST'>          ; /TEST          [37]
              18'8B     0873                        .BYTE   CLI$K_STRING,NOTNUF
               0008'    0875                        .WORD   141$-<.-2>
        54 53 45 00'    0877                        .ASCIC  'EST'
                  03    0877
                        087B  1349 141$:    $DS_CLI CLI$K_VALUE,    NOTNUF, ILLEGAL                  ; : or =
              18'8F     087B                        .BYTE   CLI$K_VALUE,NOTNUF
               FC4C     087D                        .WORD   ILLEGAL-<.-2>
                        087F  1350          $DS_CLI CLI$K_DEC,      TEST,   ILLEGAL                  ; /TEST:x
              2C'8A     087F                        .BYTE   CLI$K_DEC,TEST
               FC48     0881                        .WORD   ILLEGAL-<.-2>
                        0883  1351          $DS_CLI CLI$K_VALUE,    NOTNUF, START_RUN_QUALS          ; : or =
              18'8F     0883                        .BYTE   CLI$K_VALUE,NOTNUF
               FF4A     0885                        .WORD   START_RUN_QUALS-<.-2>
                        0887  1352          $DS_CLI CLI$K_DEC,      LAST,   ILLEGAL                  ; /TEST:x:y
              2D'8A     0887                        .BYTE   CLI$K_DEC,LAST
               FC40     0889                        .WORD   ILLEGAL-<.-2>
                        088B  1353          $DS_CLI CLI$K_BR,       0,      START_RUN_QUALS          ;
              00 82     088B                        .BYTE   CLI$K_BR,0
               FF42     088D                        .WORD   START_RUN_QUALS-<.-2>
                        088F  1354
                        088F  1355 200$:    $DS_CLI CLI$K_RETURN,   ENUF                             ; End of Qualifiers
              17'93     088F                        .BYTE   CLI$K_RETURN,ENUF
               0004'    0891                        .WORD   30315$-<.-2>
                        0893        30315$:
                        0893  1356 ;
                        0893  1357 ; <break_qual> = [ / Nobase]
                        0893  1358 ;                 [ / Base : num_base]
                        0893  1359 ;
                        0893  1360 BREAK_QUALS:
                        0893  1361          $DS_CLI CLI$K_SLASH,    0,      110$                     ; Look For A Qualifi
                 00 90  0893                        .BYTE   CLI$K_SLASH,0
                  0028' 0895                        .WORD   110$-<.-2>
                        0897  1362
                        0897  1363          $DS_CLI CLI$K_STRING,   0,      100$,   <'BASE'>         ; Look For / Base
                 00 8B  0897                        .BYTE   CLI$K_STRING,0
                  0015' 0899                        .WORD   100$-<.-2>
           45 53 41 42 00' 089B                     .ASCIC  'BASE'
                     04   089B
                        08A0  1364          $DS_CLI CLI$K_VALUE,    0,      ILLEGAL                  ;
                 00 8F  08A0                        .BYTE   CLI$K_VALUE,0
                  FC27  08A2                        .WORD   ILLEGAL-<.-2>
                        08A4  1365          $DS_CLI CLI$K_NUM,      BASE_QUAL,ILLEGAL                ; Read Temp Base
                3F'85   08A4                        .BYTE   CLI$K_NUM,BASE_QUAL
                  FC23  08A6                        .WORD   ILLEGAL-<.-2>
                        08A8  1366          $DS_CLI CLI$K_BR,       0,      BREAK_QUALS              ; Look For Next Qual
                 00 82  08A8                        .BYTE   CLI$K_BR,0
                  FFEB  08AA                        .WORD   BREAK_QUALS-<.-2>
                        08AC  1367
                        08AC  1368 100$:    $DS_CLI CLI$K_STRING,   NOBASE, BAD_QUAL,<'NOBASE'>      ; Look For / Nobase
                40'8B   08AC                        .BYTE   CLI$K_STRING,NOBASE
                  FC4F  08AE                        .WORD   BAD_QUAL-<.-2>
     45 53 41 42 4F 4E 00' 08B0                     .ASCIC  'NOBASE'
```

```
                         06  08B0
                             08B7  1369          $DS_CLI CLI$K_BR,       0,      BREAK_QUALS             ; Look For Next Qual
                      00 82  08B7                        .BYTE   CLI$K_BR,0
                       FFDC  08B9                        .WORD   BREAK_QUALS-<.-2>
                             08BB  1370
                             08BB  1371 110$:    $DS_CLI CLI$K_RETURN                                    ; No More Qualifiers
                      00 93  08BB                        .BYTE   CLI$K_RETURN,0
                       0004' 08BD                        .WORD   30323$-<.-2>
                             08BF        30323$:
                             08BF  1372 ;
                             08BF  1373 ; <device_qual> = [ / Brief]
                             08BF  1374 ;                  [ / Adapter : alnum_adpater[:]]
                             08BF  1375 ;
                             08BF  1376 DEVICE_QUALS:
                             08BF  1377          $DS_CLI CLI$K_SLASH,    NOTNUF, 130$                    ; /
                      18'90  08BF                        .BYTE   CLI$K_SLASH,NOTNUF
                       002E' 08C1                        .WORD   130$-<.-2>
                             08C3  1378
                             08C3  1379          $DS_CLI CLI$K_STRING,   0,      120$,   <'ADAPTER'>     ; / ADAPTER
                      00 8B  08C3                        .BYTE   CLI$K_STRING,0
                       001C' 08C5                        .WORD   120$-<.-2>
      52 45 54 50 41 44 41 00' 08C7                     .ASCIC  'ADAPTER'
                         07  08C7
                             08CF  1380          $DS_CLI CLI$K_VALUE,    BEGINADAPTER,ILLEGAL                    ; / ADAPTER
                      78'8F  08CF                        .BYTE   CLI$K_VALUE,BEGINADAPTER
                       FBF8  08D1                        .WORD   ILLEGAL-<.-2>
                             08D3  1381          $DS_CLI CLI$K_ALNUM,    ADAPTER,ILLEGAL                 ; Required name
                      79'87  08D3                        .BYTE   CLI$K_ALNUM,ADAPTER
                       FBF4  08D5                        .WORD   ILLEGAL-<.-2>
                             08D7  1382          $DS_CLI <^A":">,        0,      110$                    ; Optional :
                      00 3A  08D7                        .BYTE   ^A":",0
                       0004' 08D9                        .WORD   110$-<.-2>
                             08DB  1383 110$:    $DS_CLI CLI$K_BR,       ENUF,   DEVICE_QUALS            ; Any more quals?
                      17'82  08DB                        .BYTE   CLI$K_BR,ENUF
                       FFE4  08DD                        .WORD   DEVICE_QUALS-<.-2>
                             08DF  1384
                             08DF  1385 120$:    $DS_CLI CLI$K_STRING,   BRIEF,  BAD_QUAL,<'BRIEF'>      ; DEVICE / BRIEF
                      77'8B  08DF                        .BYTE   CLI$K_STRING,BRIEF
                       FC1C  08E1                        .WORD   BAD_QUAL-<.-2>
            46 45 49 52 42 00' 08E3                     .ASCIC  'BRIEF'
                         05  08E3
                             08E9  1386          $DS_CLI CLI$K_BR,       ENUF,   DEVICE_QUALS                    ; Any more q
                      17'82  08E9                        .BYTE   CLI$K_BR,ENUF
                       FFD6  08EB                        .WORD   DEVICE_QUALS-<.-2>
                             08ED  1387
                             08ED  1388 130$:    $DS_CLI CLI$K_RETURN
                      00 93  08ED                        .BYTE   CLI$K_RETURN,0
                       0004' 08EF                        .WORD   30332$-<.-2>
                             08F1        30332$:
```

```
                        08F1  1390          .SUBTITLE EXAMINE & DEPOSIT Qualifiers
                        08F1  1391 ;
                        08F1  1392 ; Read EXAMINE and DEPOSIT Qualifiers.  Qualifiers are:
                        08F1  1393 ;
                        08F1  1394 ;        / Base : number
                        08F1  1395 ;        / NObase
                        08F1  1396 ;        / Byte
                        08F1  1397 ;        / Word
                        08F1  1398 ;        / Longword
                        08F1  1399 ;        / Octal
                        08F1  1400 ;        / Hexadecimal
                        08F1  1401 ;        / Ascii
                        08F1  1402 ;        / Next : number
                        08F1  1403 ;
                        08F1  1404 EXA_DEP_QUALS:
                        08F1  1405          $DS_CLI CLI$K_SLASH,    0,      210$                    ; /
                  00 90 08F1                .BYTE   CLI$K_SLASH,0
                   00AC' 08F3               .WORD   210$-<.-2>
                        08F5  1406
                        08F5  1407          $DS_CLI CLI$K_STRING,   ASCII,  110$,   <'ASCII'>       ; / ASCII
                  5C'8B 08F5                .BYTE   CLI$K_STRING,ASCII
                   000E' 08F7               .WORD   110$-<.-2>
      49 49 43 53 41 00' 08F9               .ASCIC  'ASCII'
                      05 08F9
                        08FF  1408          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                  00 82 08FF                .BYTE   CLI$K_BR,0
                   FFF2 0901                .WORD   EXA_DEP_QUALS-<.-2>
                        0903  1409
                        0903  1410 110$:    $DS_CLI <^A"B">,        0,      130$                    ; / BASE
                  00 42 0903                .BYTE   ^A"B",0
                   0024' 0905               .WORD   130$-<.-2>
                        0907  1411          $DS_CLI CLI$K_STRING,   0,      120$,   <'ASE'>         ;
                  00 8B 0907                .BYTE   CLI$K_STRING,0
                   0014' 0909               .WORD   120$-<.-2>
            45 53 41 00' 090B               .ASCIC  'ASE'
                      03 090B
                        090F  1412          $DS_CLI CLI$K_VALUE,    0,      ILLEGAL                 ;
                  00 8F 090F                .BYTE   CLI$K_VALUE,0
                   FBB8 0911                .WORD   ILLEGAL-<.-2>
                        0913  1413          $DS_CLI CLI$K_NUM,      BASE_QUAL,ILLEGAL               ; Read Temp Base
                  3F'85 0913                .BYTE   CLI$K_NUM,BASE_QUAL
                   FBB4 0915                .WORD   ILLEGAL-<.-2>
                        0917  1414          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                  00 82 0917                .BYTE   CLI$K_BR,0
                   FFDA 0919                .WORD   EXA_DEP_QUALS-<.-2>
                        091B  1415
                        091B  1416 120$:    $DS_CLI CLI$K_STRING,   0,      121$    ,<'YTE'>        ; / B OR / BYTE
                  00 8B 091B                .BYTE   CLI$K_STRING,0
                   0008' 091D               .WORD   121$-<.-2>
            45 54 59 00' 091F               .ASCIC  'YTE'
                      03 091F
                        0923  1417 121$:    $DS_CLI CLI$K_BR,       BYTE,   EXA_DEP_QUALS           ; Look For More Qual
                  6A'82 0923                .BYTE   CLI$K_BR,BYTE
                   FFCE 0925                .WORD   EXA_DEP_QUALS-<.-2>
                        0927  1418
                        0927  1419 130$:    $DS_CLI CLI$K_STRING, DEC,      140$,   <'DECIMAL'>     ; / DECIMAL
                  6C'8B 0927                .BYTE   CLI$K_STRING,DEC
```

```
                          0010'  0929                          .WORD   140$-<.-2>
          4C 41 4D 49 43 45 44 00'  092B                          .ASCIC  'DECIMAL'
                             07  092B
                                 0933  1420          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                          00 82  0933                          .BYTE   CLI$K_BR,0
                           FFBE  0935                          .WORD   EXA_DEP_QUALS-<.-2>
                                 0937  1421
                                 0937  1422 140$:    $DS_CLI CLI$K_STRING, HEX,      150$,   <'HEXADECIMAL'> ; / HEXADECIMAL
                          6B'8B  0937                          .BYTE   CLI$K_STRING,HEX
                          0014'  0939                          .WORD   150$-<.-2>
4C 41 4D 49 43 45 44 41 58 45 48 00'  093B                          .ASCIC  'HEXADECIMAL'
                             0B  093B
                                 0947  1423          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                          00 82  0947                          .BYTE   CLI$K_BR,0
                           FFAA  0949                          .WORD   EXA_DEP_QUALS-<.-2>
                                 094B  1424
                                 094B  1425 150$:    $DS_CLI CLI$K_STRING, LONG,     160$,   <'LONGWORD'>    ; / LONGWORD
                          68'8B  094B                          .BYTE   CLI$K_STRING,LONG
                          0011'  094D                          .WORD   160$-<.-2>
        44 52 4F 57 47 4E 4F 4C 00'  094F                          .ASCIC  'LONGWORD'
                             08  094F
                                 0958  1426          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                          00 82  0958                          .BYTE   CLI$K_BR,0
                           FF99  095A                          .WORD   EXA_DEP_QUALS-<.-2>
                                 095C  1427
                                 095C  1428 160$:    $DS_CLI <^A"N">,        0,      190$                    ; / NObase
                          00 4E  095C                          .BYTE   ^A"N",0
                          0026'  095E                          .WORD   190$-<.-2>
                                 0960  1429          $DS_CLI CLI$K_STRING,   NOBASE, 170$,   <'OBASE'>       ;
                          40'8B  0960                          .BYTE   CLI$K_STRING,NOBASE
                          000E'  0962                          .WORD   170$-<.-2>
                45 53 41 42 4F 00'  0964                          .ASCIC  'OBASE'
                             05  0964
                                 096A  1430          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                          00 82  096A                          .BYTE   CLI$K_BR,0
                           FF87  096C                          .WORD   EXA_DEP_QUALS-<.-2>
                                 096E  1431
                                 096E  1432 170$:    $DS_CLI CLI$K_STRING,   0,      171$,   <'EXT'>         ; /N OR /NEXT
                          00 8B  096E                          .BYTE   CLI$K_STRING,0
                          0008'  0970                          .WORD   171$-<.-2>
                     54 58 45 00'  0972                          .ASCIC  'EXT'
                             03  0972
                                 0976  1433 171$:    $DS_CLI CLI$K_VALUE,  NOTNUF,   ILLEGAL                 ;
                          18'8F  0976                          .BYTE   CLI$K_VALUE,NOTNUF
                           FB51  0978                          .WORD   ILLEGAL-<.-2>
                                 097A  1434          $DS_CLI CLI$K_DEC,    NEXT,     ILLEGAL                 ;
                          59'8A  097A                          .BYTE   CLI$K_DEC,NEXT
                           FB4D  097C                          .WORD   ILLEGAL-<.-2>
                                 097E  1435          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                          00 82  097E                          .BYTE   CLI$K_BR,0
                           FF73  0980                          .WORD   EXA_DEP_QUALS-<.-2>
                                 0982  1436
                                 0982  1437 190$:    $DS_CLI CLI$K_STRING, OCT,      200$,   <'OCTAL'>       ; / OCTAL
                          6D'8B  0982                          .BYTE   CLI$K_STRING,OCT
                          000E'  0984                          .WORD   200$-<.-2>
                4C 41 54 43 4F 00'  0986                          .ASCIC  'OCTAL'
                             05  0986
```

```
                          098C  1438          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                   00 82  098C                        .BYTE   CLI$K_BR,0
                    FF65  098E                        .WORD   EXA_DEP_QUALS-<.-2>
                          0990  1439
                          0990  1440 200$:    $DS_CLI CLI$K_STRING, WORD,     BAD_QUAL,<'WORD'>       ; / WORD
                   69'8B  0990                        .BYTE   CLI$K_STRING,WORD
                    FB6B  0992                        .WORD   BAD_QUAL-<.-2>
               44 52 4F 57 00' 0994                   .ASCIC  'WORD'
                      04  0994
                          0999  1441          $DS_CLI CLI$K_BR,       0,      EXA_DEP_QUALS           ; Look For More Qual
                   00 82  0999                        .BYTE   CLI$K_BR,0
                    FF58  099B                        .WORD   EXA_DEP_QUALS-<.-2>
                          099D  1442
                          099D  1443 210$:    $DS_CLI CLI$K_RETURN                                    ; No More Qualifiers
                   00 93  099D                        .BYTE   CLI$K_RETURN,0
                    0004' 099F                        .WORD   30360$-<.-2>
                          09A1        30360$:
```

```
                        09A1  1445          .SUBTITLE       DEATTACH, DESELECT, and SELECT Parameters.
                        09A1  1446 ;
                        09A1  1447 ;        DEAttach / Adapter : <adapter>  <filespec>
                        09A1  1448 ;        DEAttach  <filespec> / Adapter : <adapter>
                        09A1  1449 ;
                        09A1  1450 DEATTACH_PARAMS:                                                     ;
                        09A1  1451          $DS_CLI CLI$K_SLASH,    NOTNUF, 110$                        ; If not /, get name
                18'90   09A1                .BYTE   CLI$K_SLASH,NOTNUF
                 0020'  09A3                .WORD   110$-<.-2>
                        09A5  1452
                        09A5  1453          $DS_CLI CLI$K_STRING,   0,      BAD_QUAL,<'ADAPTER'>        ; / ADAPTER
                00 8B   09A5                .BYTE   CLI$K_STRING,0
                 FB56   09A7                .WORD   BAD_QUAL-<.-2>
        52 45 54 50 41 44 41 00'  09A9      .ASCIC  'ADAPTER'
                   07   09A9
                        09B1  1454          $DS_CLI CLI$K_VALUE,    NOTNUF, ILLEGAL                     ; Required :
                18'8F   09B1                .BYTE   CLI$K_VALUE,NOTNUF
                 FB16   09B3                .WORD   ILLEGAL-<.-2>
                        09B5  1455          $DS_CLI CLI$K_BR,       BEGINADAPTER,100$                   ; Start of name
                78'82   09B5                .BYTE   CLI$K_BR,BEGINADAPTER
                 0004'  09B7                .WORD   100$-<.-2>
                        09B9  1456 100$:    $DS_CLI CLI$K_ALNUM,    ADAPTER,        ILLEGAL             ; Required alpha-num
                79'87   09B9                .BYTE   CLI$K_ALNUM,ADAPTER
                 FB0E   09BB                .WORD   ILLEGAL-<.-2>
                        09BD  1457          $DS_CLI <^A":">,        0,      110$                        ; Optional :
                00 3A   09BD                .BYTE   ^A":",0
                 0004'  09BF                .WORD   110$-<.-2>
                        09C1  1458
                        09C1  1459 110$:    $DS_CLI CLI$K_BIF,      IF_FILE_SPEC,140$                   ; End if device
                70'83   09C1                .BYTE   CLI$K_BIF,IF_FILE_SPEC
                 0020'  09C3                .WORD   140$-<.-2>
                        09C5  1460          $DS_CLI CLI$K_SPACE,    FILESPEC,ILLEGAL                    ; Required <SP>'s, I
                28'84   09C5                .BYTE   CLI$K_SPACE,FILESPEC
                 FB02   09C7                .WORD   ILLEGAL-<.-2>
                        09C9  1461 120$:    $DS_CLI CLI$K_ALNUM,    FILEND, INCOMPLETE                  ; Device name
                29'87   09C9                .BYTE   CLI$K_ALNUM,FILEND
                 FB12   09CB                .WORD   INCOMPLETE-<.-2>
                        09CD  1462          $DS_CLI <^A"$">,        FILEND, 125$                        ; Optional '$
                29'24   09CD                .BYTE   ^A"$",FILEND
                 0008'  09CF                .WORD   125$-<.-2>
                        09D1  1463          $DS_CLI CLI$K_BR,       0,      120$                        ;
                00 82   09D1                .BYTE   CLI$K_BR,0
                 FFF8   09D3                .WORD   120$-<.-2>
                        09D5  1464
                        09D5  1465 125$:    $DS_CLI <^A":">,        0,      130$                        ; Optional ':'
                00 3A   09D5                .BYTE   ^A":",0
                 0004'  09D7                .WORD   130$-<.-2>
                        09D9  1466 130$:    $DS_CLI CLI$K_BIF,      IF_ADAPTER,140$                     ; End if adapter
                71'83   09D9                .BYTE   CLI$K_BIF,IF_ADAPTER
                 0008'  09DB                .WORD   140$-<.-2>
                        09DD  1467          $DS_CLI CLI$K_BR,       NOTNUF, DEATTACH_PARAMS             ; Get adapter
                18'82   09DD                .BYTE   CLI$K_BR,NOTNUF
                 FFC4   09DF                .WORD   DEATTACH_PARAMS-<.-2>
                        09E1  1468 140$:    $DS_CLI CLI$K_BIF,      IF_ADAPTER,EOL                      ; EOL
                71'83   09E1                .BYTE   CLI$K_BIF,IF_ADAPTER
                 FACE   09E3                .WORD   EOL-<.-2>
                        09E5  1469          $DS_CLI CLI$K_BR,       0,      INCOMPLETE                  ; No good
```

```
                     00 82  09E5                        .BYTE   CLI$K_BR,0
                      FAF6  09E7                        .WORD   INCOMPLETE-<.-2>
                            09E9  1470 ;
                            09E9  1471 ; DESelect[ / Adapter : link_name[:]]  <device_names>
                            09E9  1472 ;
                            09E9  1473 DESELECT_PARAMS:
                            09E9  1474          $DS_CLI CLI$K_SLASH,    NOTNUF, 100$                    ; Look for Qualifier
                     18'90  09E9                        .BYTE   CLI$K_SLASH,NOTNUF
                      001C' 09EB                        .WORD   100$-<.-2>
                            09ED  1475          $DS_CLI CLI$K_STRING,   0,      BAD_QUAL,<'ADAPTER'>    ; /ADAPTER:<adapter>
                     00 8B  09ED                        .BYTE   CLI$K_STRING,0
                      FB0E  09EF                        .WORD   BAD_QUAL-<.-2>
         52 45 54 50 41 44 41 00' 09F1                  .ASCIC  'ADAPTER'
                        07  09F1
                            09F9  1476          $DS_CLI CLI$K_VALUE,    BEGINADAPTER,ILLEGAL            ; Required :
                     78'8F  09F9                        .BYTE   CLI$K_VALUE,BEGINADAPTER
                      FACE  09FB                        .WORD   ILLEGAL-<.-2>
                            09FD  1477          $DS_CLI CLI$K_ALNUM,    ADAPTER,ILLEGAL                 ; Adapter Name.
                     79'87  09FD                        .BYTE   CLI$K_ALNUM,ADAPTER
                      FACA  09FF                        .WORD   ILLEGAL-<.-2>
                            0A01  1478          $DS_CLI <^A":">,        0,      100$                    ; Optional :
                     00 3A  0A01                        .BYTE   ^A":",0
                      0004' 0A03                        .WORD   100$-<.-2>
                            0A05  1479
                            0A05  1480 100$:    $DS_CLI CLI$K_BR,       REQUIRED,DEVICE_NAMES           ; Get device names
                     2B'82  0A05                        .BYTE   CLI$K_BR,REQUIRED
                      FBFF  0A07                        .WORD   DEVICE_NAMES-<.-2>
                            0A09  1481 ;
                            0A09  1482 ; SELect<qualifiers>  <device_names>
                            0A09  1483 ;
                            0A09  1484 ; <qualifier> = [ / Noallocate]
                            0A09  1485 ;               [ / Adapter : link_name[:]]
                            0A09  1486 ;
                            0A09  1487 SELECT_PARAMS:
                            0A09  1488          $DS_CLI CLI$K_SLASH,    NOTNUF, 300$                    Look for Qualifiers
                     18'90  0A09                        .BYTE   CLI$K_SLASH,NOTNUF
                      0033' 0A0B                        .WORD   300$-<.-2>
                            0A0D  1489
                            0A0D  1490          $DS_CLI CLI$K_STRING,   0,      200$,   <'ADAPTER'>     ; /ADAPTER:<adapter>
                     00 8B  0A0D                        .BYTE   CLI$K_STRING,0
                      001C' 0A0F                        .WORD   200$-<.-2>
         52 45 54 50 41 44 41 00' 0A11                  .ASCIC  'ADAPTER'
                        07  0A11
                            0A19  1491          $DS_CLI CLI$K_VALUE,    BEGINADAPTER,ILLEGAL            ; Required :
                     78'8F  0A19                        .BYTE   CLI$K_VALUE,BEGINADAPTER
                      FAAE  0A1B                        .WORD   ILLEGAL-<.-2>
                            0A1D  1492          $DS_CLI CLI$K_ALNUM,    ADAPTER,ILLEGAL                 ; Adapter Name.
                     79'87  0A1D                        .BYTE   CLI$K_ALNUM,ADAPTER
                      FAAA  0A1F                        .WORD   ILLEGAL-<.-2>
                            0A21  1493          $DS_CLI <^A":">,        0,      100$                    ; Optional :
                     00 3A  0A21                        .BYTE   ^A":",0
                      0004' 0A23                        .WORD   100$-<.-2>
                            0A25  1494 100$:    $DS_CLI CLI$K_BR,       0,      SELECT_PARAMS           ; Look for more
                     00 82  0A25                        .BYTE   CLI$K_BR,0
                      FFE4  0A27                        .WORD   SELECT_PARAMS-<.-2>
                            0A29  1495
                            0A29  1496 200$:    $DS_CLI CLI$K_STRING,   NO_ALLOC,BAD_QUAL,<'NOALLOCATE'>; /NOALLOCATE
```

```
                                7C'8B  0A29                    .BYTE   CLI$K_STRING,NO_ALLOC
                                 FAD2  0A2B                    .WORD   BAD_QUAL-<.-2>
   45 54 41 43 4F 4C 4C 41 4F 4E 00'  0A2D                    .ASCIC  'NOALLOCATE'
                                   0A  0A2D
                                       0A38  1497          $DS_CLI CLI$K_BR,       0,      SELECT_PARAMS           ; Look for more
                                00 82  0A38                    .BYTE   CLI$K_BR,0
                                 FFD1  0A3A                    .WORD   SELECT_PARAMS-<.-2>
                                       0A3C  1498
                                       0A3C  1499 300$:    $DS_CLI CLI$K_BR,       REQUIRED,DEVICE_NAMES           ; Get Device Names
                                2B'82  0A3C                    .BYTE   CLI$K_BR,REQUIRED
                                 FBC8  0A3E                    .WORD   DEVICE_NAMES-<.-2>
```

```
                        0A40  1501          .SUBTITLE       Show Parameters and Qualifiers
                        0A40  1502 ;
                        0A40  1503 ; SHow parameters:
                        0A40  1504 ;
                        0A40  1505 ;       SHow  BAse
                        0A40  1506 ;       SHow  BReakpoints
                        0A40  1507 ;       SHow  CAlls                                                  [70]          ;G:3
                        0A40  1508 ;       SHow  CRd                                                    [67]          ;G:3
                        0A40  1509 ;       SHow  CPU                                                    [116]         ;G:3
                        0A40  1510 ;       SHow  DEFaults
                        0A40  1511 ;       SHow  DEVice
                        0A40  1512 ;       SHow  ENforce                                                [89]          ;G:3
                        0A40  1513 ;       SHow  EVent[  Flags]
                        0A40  1514 ;       SHow  FLags
                        0A40  1515 ;       SHow  Load
                        0A40  1516 ;       SHow  MEmory
                        0A40  1517 ;       SHow  MM
                        0A40  1518 ;       SHow  Page
                        0A40  1519 ;       SHow  QACkloop
                        0A40  1520 ;       SHow  QADefaults
                        0A40  1521 ;       SHow  QAErrorprints
                        0A40  1522 ;       SHow  QAMultiplepass
                        0A40  1523 ;       SHow  QASubtestloops
                        0A40  1524 ;       SHow  QATestloops
                        0A40  1525 ;       SHow  SECtions
                        0A40  1526 ;       SHow  SElected
                        0A40  1527 ;       SHow  STatus
                        0A40  1528 ;       SHow  SUpport
                        0A40  1529 ;       SHow  Tests                                                                [81]
                        0A40  1530 ;       SHow  Width
                        0A40  1531 ;       SHow<qualifiers>  Device
                        0A40  1532 ;       SHow<qualifiers>  Selected
                        0A40  1533 ;
                        0A40  1534 SHOW_PARAMS:
                        0A40  1535          $DS_CLI CLI$K_SPACE,    0,      310$                        ; If no <SP>'s, chec
                  00 84  0A40                       .BYTE   CLI$K_SPACE,0
                   0210' 0A42                       .WORD   310$-<.-2>
                        0A44  1536          $DS_CLI <^A"B">,        NOTNUF, 110$                        ; B
                  18'42  0A44                       .BYTE   ^A"B",NOTNUF
                   0023' 0A46                       .WORD   110$-<.-2>
                        0A48  1537 ;
                        0A48  1538 ; SHow  BAse
                        0A48  1539 ;
                        0A48  1540          $DS_CLI CLI$K_STRING,   BASE,   100$,   <'ASE'>         ; BAse
                  3C'8B  0A48                       .BYTE   CLI$K_STRING,BASE
                   000C' 0A4A                       .WORD   100$-<.-2>
            45 53 41 00' 0A4C                       .ASCIC  'ASE'
                     03  0A4C
                        0A50  1541          $DS_CLI CLI$K_BR,       ENUF,   EOL
                  17'82  0A50                       .BYTE   CLI$K_BR,ENUF
                   FA5F  0A52                       .WORD   EOL-<.-2>
                        0A54  1542 ;
                        0A54  1543 ; SHow  BReakpoints
                        0A54  1544 ;
                        0A54  1545 100$:    $DS_CLI CLI$K_STRING,   BREAK,  ILLEGAL,<'REAKPOINTS'>  ; BReakpoints
                  3E'8B  0A54                       .BYTE   CLI$K_STRING,BREAK
                   FA73  0A56                       .WORD   ILLEGAL-<.-2>
```

```
ZZ-ENSAA-10.10  Show Parameters and Qualifiers                    3-MAR-1988      Fiche 4 Frame C12        Sequence 763
CLI                        DIAGNOSTIC SUPERVISOR COMMAND LINE INTER 11-NOV-1987 17:26:56  VAX/VMS Macro V04-00        Page  57
10.10-28                   Show Parameters and Qualifiers            6-OCT-1987 20:26:22  [DS.LOCAL.RELEASE.WORK]CLI.MAR;1  (1)

   53 54 4E 49 4F 50 4B 41 45 52 00' 0A58                        .ASCIC  'REAKPOINTS'
                              0A   0A58
                                   0A63  1546          $DS_CLI CLI$K_BR,       ENUF,   EOL
                             17'82 0A63                        .BYTE   CLI$K_BR,ENUF
                              FA4C 0A65                        .WORD   EOL-<.-2>
                                   0A67  1547 ;
                                   0A67  1548 ; SHow  CRd                                                          [67]
                                   0A67  1549 ;
                                   0A67  1550 110$:    $DS_CLI <^A"C">,        NOTNUF, 115$                    ; SHow C        [70]
                             18'43 0A67                        .BYTE   ^A"C",NOTNUF
                              0027' 0A69                       .WORD   115$-<.-2>
                                   0A6B  1551          $DS_CLI CLI$K_STRING,   SHOW_CRD_TRACE,112$,<'RD'>      ; SHow CRd      [70]
                             22'8B 0A6B                        .BYTE   CLI$K_STRING,SHOW_CRD_TRACE
                              000B' 0A6D                       .WORD   112$-<.-2>
                          44 52 00' 0A6F                       .ASCIC  'RD'
                                02  0A6F
                                   0A72  1552          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;               [65]
                             17'82 0A72                        .BYTE   CLI$K_BR,ENUF
                              FA3D 0A74                        .WORD   EOL-<.-2>
```

```
                        0A76  1554 ;
                        0A76  1555 ; SHow  CAlls
                        0A76  1556 ;
                        0A76  1557 112$:   $DS_CLI CLI$K_STRING,   SHOW_CALLS,113$,<'ALLS'>  ; SHow CAlls [116]  ;G:3
                 23'8B  0A76                       .BYTE   CLI$K_STRING,SHOW_CALLS
                  000D' 0A78                       .WORD   113$-<.-2>
           53 4C 4C 41 00' 0A7A                    .ASCIC  'ALLS'
                    04  0A7A
                        0A7F  1558         $DS_CLI CLI$K_BR,       ENUF,   EOL                 ;             [70]  ;G:3
                 17'82  0A7F                       .BYTE   CLI$K_BR,ENUF
                  FA30  0A81                       .WORD   EOL-<.-2>
                        0A83  1559                                                                                ;G:3
                        0A83  1560 ;                                                                              ;G:3
                        0A83  1561 ; SHow Cpu                                                 ; SHOW CPU [116]    ;G:3
                        0A83  1562 ;                                                                              ;G:3
                        0A83  1563 113$:   $DS_CLI CLI$K_STRING,   SHOWCPU,ILLEGAL,<'PU'>     ; [116]             ;G:3
                 24'8B  0A83                       .BYTE   CLI$K_STRING,SHOWCPU
                  FA44  0A85                       .WORD   ILLEGAL-<.-2>
                55 50 00' 0A87                     .ASCIC  'PU'
                    02  0A87
                        0A8A  1564         $DS_CLI CLI$K_BR,       ENUF,   EOL                 ; [116]             ;G:3
                 17'82  0A8A                       .BYTE   CLI$K_BR,ENUF
                  FA25  0A8C                       .WORD   EOL-<.-2>
                        0A8E  1565                                                                                ;G:3
                        0A8E  1566 ;
                        0A8E  1567 ;
                        0A8E  1568 ; SHow  D . . .
                        0A8E  1569 ;
                        0A8E  1570 115$:   $DS_CLI <^A"D">,        NOTNUF, 150$                    ; D
                 18'44  0A8E                       .BYTE   ^A"D",NOTNUF
                  0028' 0A90                       .WORD   150$-<.-2>
                        0A92  1571         $DS_CLI <^A"E">,        NOTNUF, ILLEGAL                 ; DE
                 18'45  0A92                       .BYTE   ^A"E",NOTNUF
                  FA35  0A94                       .WORD   ILLEGAL-<.-2>
                        0A96  1572 ;
                        0A96  1573 ; SHow  DEFaults
                        0A96  1574 ;
                        0A96  1575         $DS_CLI CLI$K_STRING,   DEFAULT,        120$,   <'FAULTS'>      ; DEFaults
                 43'8B  0A96                       .BYTE   CLI$K_STRING,DEFAULT
                  000F' 0A98                       .WORD   120$-<.-2>
     53 54 4C 55 41 46 00' 0A9A                    .ASCIC  'FAULTS'
                    06  0A9A
                        0AA1  1576         $DS_CLI CLI$K_BR,       ENUF,   EOL
                 17'82  0AA1                       .BYTE   CLI$K_BR,ENUF
                  FA0E  0AA3                       .WORD   EOL-<.-2>
                        0AA5  1577 ;
                        0AA5  1578 ; SHow  DEVice[<device_quals>][  <device_names>]                   ; See end of SHOW se
                        0AA5  1579 ;
                        0AA5  1580 120$:   $DS_CLI CLI$K_STRING,   DEVICE, ILLEGAL,<'VICE'>         ; DEVice
                 76'8B  0AA5                       .BYTE   CLI$K_STRING,DEVICE
                  FA22  0AA7                       .WORD   ILLEGAL-<.-2>
           45 43 49 56 00' 0AA9                    .ASCIC  'VICE'
                    04  0AA9
                        0AAE  1581         $DS_CLI CLI$K_CALL,     0,      DEVICE_QUALS            ; Read Device Quals
                 00 92  0AAE                       .BYTE   CLI$K_CALL,0
                  FE11  0AB0                       .WORD   DEVICE_QUALS-<.-2>
                        0AB2  1582         $DS_CLI CLI$K_BR,       OPTIONAL,DEVICE_NAMES           ; Get optional names
```

```
                     2A'82  0AB2                        .BYTE   CLI$K_BR,OPTIONAL
                      FB52  0AB4                        .WORD   DEVICE_NAMES-<.-2>
                            0AB6  1583 ;
                            0AB6  1584 ; SHow  ENforce
                            0AB6  1585 ;
                            0AB6  1586 150$:    $DS_CLI <^A"E">,        NOTNUF, 160$                    ; E (skip to 'F')
                     18'45  0AB6                        .BYTE   ^A"E",NOTNUF
                      002E' 0AB8                        .WORD   160$-<.-2>
                            0ABA  1587          $DS_CLI CLI$K_STRING,   SHOWENFORCE,155$,<'NFORCE'>     ; ENforce
                     38'8B  0ABA                        .BYTE   CLI$K_STRING,SHOWENFORCE
                      000F' 0ABC                        .WORD   155$-<.-2>
           45 43 52 4F 46 4E 00' 0ABE                   .ASCIC  'NFORCE'
                        06  0ABE
                            0AC5  1588          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;
                     17'82  0AC5                        .BYTE   CLI$K_BR,ENUF
                      F9EA  0AC7                        .WORD   EOL-<.-2>
                            0AC9  1589 ;
                            0AC9  1590 ; SHow  EVent[  Flags]
                            0AC9  1591 ;
                            0AC9  1592 155$:    $DS_CLI CLI$K_STRING,   EVENT,  160$,   <'VENT'>        ; EVent
                     3A'8B  0AC9                        .BYTE   CLI$K_STRING,EVENT
                      001B' 0ACB                        .WORD   160$-<.-2>
              54 4E 45 56 00' 0ACD                      .ASCIC  'VENT'
                        04  0ACD
                            0AD2  1593          $DS_CLI CLI$K_SPACE,    ENUF,   EOL                     ;
                     17'84  0AD2                        .BYTE   CLI$K_SPACE,ENUF
                      F9DD  0AD4                        .WORD   EOL-<.-2>
                            0AD6  1594          $DS_CLI CLI$K_STRING,   ENUF,   EOL,    <'FLAGS'>       ; EVent  Flags
                     17'8B  0AD6                        .BYTE   CLI$K_STRING,ENUF
                      F9D9  0AD8                        .WORD   EOL-<.-2>
           53 47 41 4C 46 00' 0ADA                      .ASCIC  'FLAGS'
                        05  0ADA
                            0AE0  1595          $DS_CLI CLI$K_BR,       ENUF,   EOL
                     17'82  0AE0                        .BYTE   CLI$K_BR,ENUF
                      F9CF  0AE2                        .WORD   EOL-<.-2>
                            0AE4  1596 ;
                            0AE4  1597 ; SHow Flags
                            0AE4  1598 ;
                            0AE4  1599 160$:    $DS_CLI CLI$K_STRING,   FLAGS,  170$,   <'FLAGS'>       ; Flags
                     3B'8B  0AE4                        .BYTE   CLI$K_STRING,FLAGS
                      000E' 0AE6                        .WORD   170$-<.-2>
           53 47 41 4C 46 00' 0AE8                      .ASCIC  'FLAGS'
                        05  0AE8
                            0AEE  1600          $DS_CLI CLI$K_BR,       ENUF,   EOL
                     17'82  0AEE                        .BYTE   CLI$K_BR,ENUF
                      F9C1  0AF0                        .WORD   EOL-<.-2>
                            0AF2  1601 ;
                            0AF2  1602 ; SHow Load
                            0AF2  1603 ;
                            0AF2  1604 170$:    $DS_CLI CLI$K_STRING,   SHOWLOAD,180$,  <'LOAD'>        ; Load
                     80'8B  0AF2                        .BYTE   CLI$K_STRING,SHOWLOAD
                      000D' 0AF4                        .WORD   180$-<.-2>
              44 41 4F 4C 00' 0AF6                      .ASCIC  'LOAD'
                        04  0AF6
                            0AFB  1605          $DS_CLI CLI$K_BR,       ENUF,   EOL
                     17'82  0AFB                        .BYTE   CLI$K_BR,ENUF
                      F9B4  0AFD                        .WORD   EOL-<.-2>
```

```
                                  0AFF  1607 ;
                                  0AFF  1608 ; SHow M...
                                  0AFF  1609 ;
                                  0AFF  1610 180$:   $DS_Cli <^A"M">,        0,      190$                    ; Show  M       [62]
                          00 4D   0AFF                        .BYTE   ^A"M",0
                           005A'  0B01                        .WORD   190$-<.-2>
                                  0B03  1611 ;
                                  0B03  1612 ; SHow  MEmory<qualifiers>
                                  0B03  1613 ;
                                  0B03  1614         $DS_CLI CLI$K_STRING,   SHOWMEM,189$,   <'EMORY'>       ; MEmory        [62]
                          85'8B   0B03                        .BYTE   CLI$K_STRING,SHOWMEM
                           004C'  0B05                        .WORD   189$-<.-2>
                 59 52 4F 4D 45 00'  0B07                        .ASCIC  'EMORY'
                             05   0B07
                                  0B0D  1615 185$:   $DS_CLI CLI$K_SLASH,    NOTNUF, EOL
                          18'90   0B0D                        .BYTE   CLI$K_SLASH,NOTNUF
                           F9A2   0B0F                        .WORD   EOL-<.-2>
                                  0B11  1616
                                  0B11  1617         $DS_CLI CLI$K_STRING,   SHOMEMALL,186$, <'ALL'>         ; / All
                          89'8B   0B11                        .BYTE   CLI$K_STRING,SHOMEMALL
                           000C'  0B13                        .WORD   186$-<.-2>
                     4C 4C 41 00'  0B15                        .ASCIC  'ALL'
                             03   0B15
                                  0B19  1618         $DS_CLI CLI$K_BR,       ENUF,   185$
                          17'82   0B19                        .BYTE   CLI$K_BR,ENUF
                           FFF4   0B1B                        .WORD   185$-<.-2>
                                  0B1D  1619
                                  0B1D  1620 186$:   $DS_CLI CLI$K_STRING,   SHOMEMBUF,187$, <'BUFFER'>      ; / Buffer
                          87'8B   0B1D                        .BYTE   CLI$K_STRING,SHOMEMBUF
                           000F'  0B1F                        .WORD   187$-<.-2>
              52 45 46 46 55 42 00'  0B21                        .ASCIC  'BUFFER'
                             06   0B21
                                  0B28  1621         $DS_CLI CLI$K_BR,       ENUF,   185$
                          17'82   0B28                        .BYTE   CLI$K_BR,ENUF
                           FFE5   0B2A                        .WORD   185$-<.-2>
                                  0B2C  1622
                                  0B2C  1623 187$:   $DS_CLI CLI$K_STRING,   SHOMEMDAT,188$, <'DATA_STRUCTURE'>; / Data_structure
                          88'8B   0B2C                        .BYTE   CLI$K_STRING,SHOMEMDAT
                           0017'  0B2E                        .WORD   188$-<.-2>
54 43 55 52 54 53 5F 41 54 41 44 00'  0B30                        .ASCIC  'DATA_STRUCTURE'
                        45 52 55   0B3C
                             0E   0B30
                                  0B3F  1624         $DS_CLI CLI$K_BR,       ENUF,   185$
                          17'82   0B3F                        .BYTE   CLI$K_BR,ENUF
                           FFCE   0B41                        .WORD   185$-<.-2>
                                  0B43  1625
                                  0B43  1626 188$:   $DS_CLI CLI$K_STRING,   SHOMEMMAP,BAD_QUAL,<'MAP'>      ; / Map
                          86'8B   0B43                        .BYTE   CLI$K_STRING,SHOMEMMAP
                           F9B8   0B45                        .WORD   BAD_QUAL-<.-2>
                     50 41 4D 00'  0B47                        .ASCIC  'MAP'
                             03   0B47
                                  0B4B  1627         $DS_CLI CLI$K_BR,       ENUF,   185$
                          17'82   0B4B                        .BYTE   CLI$K_BR,ENUF
                           FFC2   0B4D                        .WORD   185$-<.-2>
                                  0B4F  1628 ;
                                  0B4F  1629 ; SHow MM
                                  0B4F  1630 ;
```

```
                                0B4F  1631 189$:   $DS_CLI CLI$K_STRING,   SHOWMM,ILLEGAL, <'M'>            ; MM              [62]
                          31'8B  0B4F                       .BYTE   CLI$K_STRING,SHOWMM
                           F978  0B51                       .WORD   ILLEGAL-<.-2>
                          4D 00' 0B53                       .ASCIC  'M'
                             01  0B53
                                0B55  1632         $DS_CLI CLI$K_BR,       ENUF,   EOL
                          17'82  0B55                       .BYTE   CLI$K_BR,ENUF
                           F95A  0B57                       .WORD   EOL-<.-2>
                                0B59  1633 ;
                                0B59  1634 ; SHow PAGE
                                0B59  1635 ;
                                0B59  1636 190$:   $DS_CLI CLI$K_STRING,   PAGE,   195$,   <'PAGE'>         ; Page            [71]
                          5A'8B  0B59                       .BYTE   CLI$K_STRING,PAGE
                           000D' 0B5B                       .WORD   195$-<.-2>
                 45 47 41 50 00' 0B5D                       .ASCIC  'PAGE'
                             04  0B5D
                                0B62  1637         $DS_CLI CLI$K_BR        ENUF,   EOL                     ;                 [71]
                          17'82  0B62                       .BYTE   CLI$K_BR,ENUF
                           F94D  0B64                       .WORD   EOL-<.-2>
                                0B66  1638 ;
                                0B66  1639 ; SHow  QACklooploops
                                0B66  1640 ;
                                0B66  1641 195$:   $DS_CLI <^A"Q">         NOTNUF, 250$                    ; Q               [42]
                          18'51  0B66                       .BYTE   ^A"Q",NOTNUF
                           007D' 0B68                       .WORD   250$-<.-2>
                                0B6A  1642         $DS_CLI <^A"A">         NOTNUF, ILLEGAL                 ; QA              [42]
                          18'41  0B6A                       .BYTE   ^A"A",NOTNUF
                           F95D  0B6C                       .WORD   ILLEGAL-<.-2>
                                0B6E  1643         $DS_CLI CLI$K_STRING    QACL,   200$,   <'CKLOOPLOOPS'> ; QACklooploops [42]
                          53'8B  0B6E                       .BYTE   CLI$K_STRING,QACL
                           0014' 0B70                       .WORD   200$-<.-2>
53 50 4F 4F 4C 50 4F 4F 4C 4B 43 00' 0B72                       .ASCIC  'CKLOOPLOOPS'
                             0B  0B72
                                0B7E  1644         $DS_CLI CLI$K_BR        ENUF,   EOL                     ;                 [42]
                          17'82  0B7E                       .BYTE   CLI$K_BR,ENUF
                           F931  0B80                       .WORD   EOL-<.-2>
                                0B82  1645 ;
                                0B82  1646 ; SHow  QADefaults
                                0B82  1647 ;
                                0B82  1648 200$:   $DS_CLI CLI$K_STRING    QADEF,  210$,   <'DEFAULTS'>    ; QADefaults      [42]
                          57'8B  0B82                       .BYTE   CLI$K_STRING,QADEF
                           0011' 0B84                       .WORD   210$-<.-2>
        53 54 4C 55 41 46 45 44 00' 0B86                       .ASCIC  'DEFAULTS'
                             08  0B86
                                0B8F  1649         $DS_CLI CLI$K_BR        ENUF,   EOL                     ;                 [42]
                          17'82  0B8F                       .BYTE   CLI$K_BR,ENUF
                           F920  0B91                       .WORD   EOL-<.-2>
                                0B93  1650 ;
                                0B93  1651 ; SHow  QAErrorprints
                                0B93  1652 ;
                                0B93  1653 210$:   $DS_CLI CLI$K_STRING    QAEP,   220$,   <'ERRORPRINTS'> ; QAErrorprints [42]
                          52'8B  0B93                       .BYTE   CLI$K_STRING,QAEP
                           0014' 0B95                       .WORD   220$-<.-2>
53 54 4E 49 52 50 52 4F 52 52 45 00' 0B97                       .ASCIC  'ERRORPRINTS'
                             0B  0B97
                                0BA3  1654         $DS_CLI CLI$K_BR        ENUF,   EOL                     ;                 [42]
                          17'82  0BA3                       .BYTE   CLI$K_BR,ENUF
```

```
                          F90C  0BA5                        .WORD   EOL-<.-2>
                                0BA7  1655 ;
                                0BA7  1656 ; SHow QAMultiplepass
                                0BA7  1657 ;
                                0BA7  1658 220$:   $DS_CLI CLI$K_STRING    QAMP,   230$,   <'MULTIPLEPASS'>; QAMultpilepass[42]
                             56'8B  0BA7                        .BYTE   CLI$K_STRING,QAMP
                              0015' 0BA9                        .WORD   230$-<.-2>
53 41 50 45 4C 50 49 54 4C 55 4D 00' 0BAB                      .ASCIC  'MULTIPLEPASS'
                                53  0BB7
                                0C  0BAB
                                0BB8  1659         $DS_CLI CLI$K_BR        ENUF,   EOL                     ;               [42]
                             17'82  0BB8                        .BYTE   CLI$K_BR,ENUF
                              F8F7  0BBA                        .WORD   EOL-<.-2>
                                0BBC  1660 ;
                                0BBC  1661 ; SHow  QASubtestloops
                                0BBC  1662 ;
                                0BBC  1663 230$:   $DS_CLI CLI$K_STRING    QASL,   240$,   <'SUBTESTLOOPS'>; QASubtestloops[42]
                             55'8B  0BBC                        .BYTE   CLI$K_STRING,QASL
                              0015' 0BBE                        .WORD   240$-<.-2>
50 4F 4F 4C 54 53 45 54 42 55 53 00' 0BC0                      .ASCIC  'SUBTESTLOOPS'
                                53  0BCC
                                0C  0BC0
                                0BCD  1664         $DS_CLI CLI$K_BR        ENUF,   EOL                     ;               [42]
                             17'82  0BCD                        .BYTE   CLI$K_BR,ENUF
                              F8E2  0BCF                        .WORD   EOL-<.-2>
                                0BD1  1665 ;
                                0BD1  1666 ; SHow QATestloops
                                0BD1  1667 ;
                                0BD1  1668 240$:   $DS_CLI CLI$K_STRING    QATL,   ILLEGAL,<'TESTLOOPS'>   ; QATestloops   [42]
                             54'8B  0BD1                        .BYTE   CLI$K_STRING,QATL
                              F8F6  0BD3                        .WORD   ILLEGAL-<.-2>
53 50 4F 4F 4C 54 53 45 54 00' 0BD5                            .ASCIC  'TESTLOOPS'
                                09  0BD5
                                0BDF  1669         $DS_CLI CLI$K_BR        ENUF,   EOL                     ;               [42]
                             17'82  0BDF                        .BYTE   CLI$K_BR,ENUF
                              F8D0  0BE1                        .WORD   EOL-<.-2>
                                0BE3  1670 ;
                                0BE3  1671 ; SHow  SECtions
                                0BE3  1672 ;
                                0BE3  1673 250$:   $DS_CLI <^A"S">,        NOTNUF, 300$                    ; S
                             18'53  0BE3                        .BYTE   ^A"S",NOTNUF
                              0051' 0BE5                        .WORD   300$-<.-2>
                                0BE7  1674         $DS_CLI <^A"E">,        NOTNUF, 280$                    ; SE
                             18'45  0BE7                        .BYTE   ^A"E",NOTNUF
                              0030' 0BE9                        .WORD   280$-<.-2>
                                0BEB  1675 252$:   $DS_CLI CLI$K_STRING,   SHOWSECTIONS,255$,<'CTIONS'>    ; SECtions
                             83'8B  0BEB                        .BYTE   CLI$K_STRING,SHOWSECTIONS
                              000F' 0BED                        .WORD   255$-<.-2>
                  53 4E 4F 49 54 43 00' 0BEF                    .ASCIC  'CTIONS'
                                06  0BEF
                                0BF6  1676         $DS_CLI CLI$K_BR,       ENUF,   EOL
                             17'82  0BF6                        .BYTE   CLI$K_BR,ENUF
                              F8B9  0BF8                        .WORD   EOL-<.-2>
                                0BFA  1677 ;
                                0BFA  1678 ; SHow  SElected[ / Brief]                                      ; See end of SHOW se
                                0BFA  1679 ;
                                0BFA  1680 255$:   $DS_CLI CLI$K_STRING,   0,      258$,   <'LECTED'>      ; SElected      [39]
```

```
                     00 8B  0BFA                          .BYTE   CLI$K_STRING,0
                      000B' 0BFC                          .WORD   258$-<.-2>
           44 45 54 43 45 4C 00' 0BFE                     .ASCIC  'LECTED'
                        06  0BFE
                            0C05  1681 258$:   $DS_CLI CLI$K_SLASH,    NOTNUF, 260$                    ; /
                     18'90  0C05                          .BYTE   CLI$K_SLASH,NOTNUF
                      000E' 0C07                          .WORD   260$-<.-2>
                            0C09  1682         $DS_CLI CLI$K_STRING,   BRIEF,  BAD_QUAL,<'BRIEF'>      ; SELected /BRIEF
                     77'8B  0C09                          .BYTE   CLI$K_STRING,BRIEF
                      F8F2  0C0B                          .WORD   BAD_QUAL-<.-2>
              46 45 49 52 42 00' 0C0D                    .ASCIC  'BRIEF'
                        05  0C0D
                            0C13  1683 260$:   $DS_CLI CLI$K_BR,       SHOWSEL,EOL
                     7D'82  0C13                          .BYTE   CLI$K_BR,SHOWSEL
                      F89C  0C15                          .WORD   EOL-<.-2>
                            0C17  1684 ;
                            0C17  1685 ; SHow  STatus
                            0C17  1686 ;
                            0C17  1687 280$:   $DS_CLI CLI$K_STRING,   SHOWSTATUS,290$,<'TATUS'>       ; STatus        [39]
                     82'8B  0C17                          .BYTE   CLI$K_STRING,SHOWSTATUS
                      000E' 0C19                          .WORD   290$-<.-2>
              53 55 54 41 54 00' 0C1B                    .ASCIC  'TATUS'
                        05  0C1B
                            0C21  1688         $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;               [39]
                     17'82  0C21                          .BYTE   CLI$K_BR,ENUF
                      F88E  0C23                          .WORD   EOL-<.-2>
                            0C25  1689 ;
                            0C25  1690 ; SHow  SUpport
                            0C25  1691 ;
                            0C25  1692 290$:   $DS_CLI CLI$K_STRING,   SHOWSUP,ILLEGAL,<'UPPORT'>      ; SUpport
                     81'8B  0C25                          .BYTE   CLI$K_STRING,SHOWSUP
                      F8A2  0C27                          .WORD   ILLEGAL-<.-2>
           54 52 4F 50 50 55 00' 0C29                     .ASCIC  'UPPORT'
                        06  0C29
                            0C30  1693         $DS_CLI CLI$K_BR,       ENUF,   EOL
                     17'82  0C30                          .BYTE   CLI$K_BR,ENUF
                      F87F  0C32                          .WORD   EOL-<.-2>
                            0C34  1694 ;
                            0C34  1695 ; SHow  Tests
                            0C34  1696 ;
                            0C34  1697 300$:   $DS_CLI CLI$K_STRING,   SHOWTESTS,305$, <'TESTS'>       ; Tests         [81]
                     84'8B  0C34                          .BYTE   CLI$K_STRING,SHOWTESTS
                      000E' 0C36                          .WORD   305$-<.-2>
              53 54 53 45 54 00' 0C38                    .ASCIC  'TESTS'
                        05  0C38
                            0C3E  1698         $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;               [81]
                     17'82  0C3E                          .BYTE   CLI$K_BR,ENUF
                      F871  0C40                          .WORD   EOL-<.-2>
                            0C42  1699 ;
                            0C42  1700 ; SHow  Width
                            0C42  1701 ;
                            0C42  1702 305$:   $DS_CLI CLI$K_STRING,   WIDTH,  310$,   <'WIDTH'>       ; Width         [38]
                     5B'8B  0C42                          .BYTE   CLI$K_STRING,WIDTH
                      000E' 0C44                          .WORD   310$-<.-2>
              48 54 44 49 57 00' 0C46                    .ASCIC  'WIDTH'
                        05  0C46
                            0C4C  1703         $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;               [38]
```

```
                        17'82  0C4C                         .BYTE   CLI$K_BR,ENUF
                         F863  0C4E                         .WORD   EOL-<.-2>
                               0C50  1704 ;
                               0C50  1705 ; SHow<device_quals>  <option>  <device_names>
                               0C50  1706 ;
                               0C50  1707 310$:   $DS_CLI CLI$K_CALL,     0,      DEVICE_QUALS            ; Read Quals
                        00 92  0C50                         .BYTE   CLI$K_CALL,0
                         FC6F  0C52                         .WORD   DEVICE_QUALS-<.-2>
                               0C54  1708 330$:   $DS_CLI CLI$K_SPACE,    NOTNUF, ILLEGAL                 ; Required spaces
                        18'84  0C54                         .BYTE   CLI$K_SPACE,NOTNUF
                         F873  0C56                         .WORD   ILLEGAL-<.-2>
                               0C58  1709
                               0C58  1710         $DS_CLI CLI$K_STRING,   DEVICE, 340$,   <'DEVICE'>      ; Device
                        76'8B  0C58                         .BYTE   CLI$K_STRING,DEVICE
                         000F' 0C5A                         .WORD   340$-<.-2>
           45 43 49 56 45 44 00' 0C5C                         .ASCIC  'DEVICE'
                           06  0C5C
                               0C63  1711         $DS_CLI CLI$K_BR,       OPTIONAL,DEVICE_NAMES           ; Optional device na
                        2A'82  0C63                         .BYTE   CLI$K_BR,OPTIONAL
                         F9A1  0C65                         .WORD   DEVICE_NAMES-<.-2>
                               0C67  1712
                               0C67  1713 340$:   $DS_CLI CLI$K_STRING,   SHOWSEL,ILLEGAL,<'SELECTED'>    ; Selected      [39]
                        7D'8B  0C67                         .BYTE   CLI$K_STRING,SHOWSEL
                         F860  0C69                         .WORD   ILLEGAL-<.-2>
     44 45 54 43 45 4C 45 53 00' 0C6B                         .ASCIC  'SELECTED'
                           08  0C6B
                               0C74  1714         $DS_CLI CLI$K_BR,       ENUF,   EOL
                        17'82  0C74                         .BYTE   CLI$K_BR,ENUF
                         F83B  0C76                         .WORD   EOL-<.-2>
                               0C78  1715 ;
                               0C78  1716 ; End of SHOW.
                               0C78  1717 ;
```

```
                        0C78  1719          .SUBTITLE       Clear Parameters and Qualifiers
                        0C78  1720 ;
                        0C78  1721 ; CLEAR parameters.
                        0C78  1722 ;
                        0C78  1723 ;        CLear  BReakpoints  ...
                        0C78  1724 ;        CLear  Crd  ...
                        0C78  1725 ;        CLear  ENforce
                        0C78  1726 ;        CLear  EVent  ...
                        0C78  1727 ;        CLear  Flags  ...
                        0C78  1728 ;        CLear  ...
                        0C78  1729 ;
                        0C78  1730
                        0C78  1731 CLEAR_PARAMS:
                        0C78  1732          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required <SP>'s
                 00 84  0C78                        .BYTE   CLI$K_SPACE,0
                  F84F  0C7A                        .WORD   ILLEGAL-<.-2>
                        0C7C  1733          $DS_CLI <^A"B">,        NOTNUF, 116$                    ;"BREAKPOINTS" OR "B
                18'42   0C7C                        .BYTE   ^A"B",NOTNUF
                 003E'  0C7E                        .WORD   116$-<.-2>
                        0C80  1734 ;
                        0C80  1735 ; CLear  BReakpoints<qualifiers>  num_address<qualifiers>
                        0C80  1736 ; CLear  BReakpoints<qualifiers>  ALl<qualifiers>
                        0C80  1737 ;
                        0C80  1738          $DS_CLI CLI$K_STRING,   BREAK,  115$,   <'REAKPOINTS'>  ;
                 3E'8B  0C80                        .BYTE   CLI$K_STRING,BREAK
                  0036' 0C82                        .WORD   115$-<.-2>
53 54 4E 49 4F 50 4B 41 45 52 00'  0C84             .ASCIC  'REAKPOINTS'
                    0A  0C84
                        0C8F  1739          $DS_CLI CLI$K_CALL,     0,      BREAK_QUALS             ; Look For Qualifier
                 00 92  0C8F                        .BYTE   CLI$K_CALL,0
                  FC04  0C91                        .WORD   BREAK_QUALS-<.-2>
                        0C93  1740          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL
                 00 84  0C93                        .BYTE   CLI$K_SPACE,0
                  F834  0C95                        .WORD   ILLEGAL-<.-2>
                        0C97  1741
                        0C97  1742          $DS_CLI <^A"A">,        0,      105$                    ;"All" Or Address
                 00 41  0C97                        .BYTE   ^A"A",0
                  0013' 0C99                        .WORD   105$-<.-2>
                        0C9B  1743          $DS_CLI CLI$K_STRING,   ALL,    100$,   <'LL'>
                 41'8B  0C9B                        .BYTE   CLI$K_STRING,ALL
                  000B' 0C9D                        .WORD   100$-<.-2>
              4C 4C 00' 0C9F                        .ASCIC  'LL'
                    02  0C9F
                        0CA2  1744          $DS_CLI CLI$K_BR,       0,      110$                    ; Go Look For Quals
                 00 82  0CA2                        .BYTE   CLI$K_BR,0
                  000C' 0CA4                        .WORD   110$-<.-2>
                        0CA6  1745
                        0CA6  1746 100$:    $DS_CLI CLI$K_BR,       BACKUP, 105$                    ;Back Up Over A
                 64'82  0CA6                        .BYTE   CLI$K_BR,BACKUP
                  0004' 0CA8                        .WORD   105$-<.-2>
                        0CAA  1747 105$:    $DS_CLI CLI$K_NUM,      ADDRESS,ILLEGAL                 ;Read Address
                 3D'85  0CAA                        .BYTE   CLI$K_NUM,ADDRESS
                  F81D  0CAC                        .WORD   ILLEGAL-<.-2>
                        0CAE  1748
                        0CAE  1749 110$:    $DS_CLI CLI$K_CALL,     0,      BREAK_QUALS             ; Look For Qualifier
                 00 92  0CAE                        .BYTE   CLI$K_CALL,0
                  FBE5  0CB0                        .WORD   BREAK_QUALS-<.-2>
```

```
                        0CB2  1750          $DS_CLI CLI$K_BR,       0,      EOL                          ; Done Parsing
                 00 82  0CB2                        .BYTE   CLI$K_BR,0
                  F7FD  0CB4                        .WORD   EOL-<.-2>
                        0CB6  1751
                        0CB6  1752 115$:    $DS_CLI CLI$K_BR,       BACKUP, 170$                         ; Backup over Bell
                 64'82  0CB6                        .BYTE   CLI$K_BR,BACKUP
                  005A' 0CB8                        .WORD   170$-<.-2>
                        0CBA  1753 ;
                        0CBA  1754 ; CLear  Crd<qualifiers>                                              ; NOTE: CL C = CL C/
                        0CBA  1755 ;
                        0CBA  1756 116$:    $DS_CLI CLI$K_STRING,   0,      120$,   <'CRD'>              ; CRD                [69]
                 00 8B  0CBA                        .BYTE   CLI$K_STRING,0
                  0036' 0CBC                        .WORD   120$-<.-2>
              44 52 43 00' 0CBE                     .ASCIC  'CRD'
                    03  0CBE
                        0CC2  1757          $DS_Cli CLI$K_SLASH,    0,      117$,                        ; CRD/               [69]
                 00 90  0CC2                        .BYTE   CLI$K_SLASH,0
                  000E' 0CC4                        .WORD   117$-<.-2>
                        0CC6  1758          $DS_Cli CLI$K_STRING,   0,      118$,   <'TRACE'>            ; CRD/TRACE          [69]
                 00 8B  0CC6                        .BYTE   CLI$K_STRING,0
                  000E' 0CC8                        .WORD   118$-<.-2>
           45 43 41 52 54 00' 0CCA                  .ASCIC  'TRACE'
                    05  0CCA
                        0CD0  1759 117$:    $DS_CLI CLI$K_BR,       CLEAR_CRD_TRACE,EOL                                  ;
                 1E'82  0CD0                        .BYTE   CLI$K_BR,CLEAR_CRD_TRACE
                  F7DF  0CD2                        .WORD   EOL-<.-2>
                        0CD4  1760
                        0CD4  1761 118$:    $DS_CLI CLI$K_STRING,   NOTNUF, BAD_QUAL,<'DEBUG'>           ; CRD/DEBUG          [69]
                 18'8B  0CD4                        .BYTE   CLI$K_STRING,NOTNUF
                  F827  0CD6                        .WORD   BAD_QUAL-<.-2>
           47 55 42 45 44 00' 0CD8                  .ASCIC  'DEBUG'
                    05  0CD8
                        0CDE  1762          $DS_CLI CLI$K_SLASH,    NOTNUF, BAD_QUAL,                    ; CRD/DEBUG/         [69]
                 18'90  0CDE                        .BYTE   CLI$K_SLASH,NOTNUF
                  F81D  0CE0                        .WORD   BAD_QUAL-<.-2>
                        0CE2  1763          $DS_CLI CLI$K_STRING,   0,      BAD_QUAL,<'TRACE'>           ; CRD/DEBUG/TRACE[69
                 00 8B  0CE2                        .BYTE   CLI$K_STRING,0
                  F819  0CE4                        .WORD   BAD_QUAL-<.-2>
           45 43 41 52 54 00' 0CE6                  .ASCIC  'TRACE'
                    05  0CE6
                        0CEC  1764          $DS_CLI CLI$K_BR,       CLEAR_CRD_DEBUG,EOL                  ;                    [69]
                 21'82  0CEC                        .BYTE   CLI$K_BR,CLEAR_CRD_DEBUG
                  F7C3  0CEE                        .WORD   EOL-<.-2>
                        0CF0  1765 ;
                        0CF0  1766 ; CLear  ENforce
                        0CF0  1767 ;
                        0CF0  1768 120$:    $DS_CLI <^A"E">,        0,      170$                         ; an E?
                 00 45  0CF0                        .BYTE   ^A"E",0
                  0020' 0CF2                        .WORD   170$-<.-2>
                        0CF4  1769          $DS_CLI CLI$K_STRING,   CLEARENFORCE,125$, <'NFORCE'>        ; ENFORCE
                 37'8B  0CF4                        .BYTE   CLI$K_STRING,CLEARENFORCE
                  000F' 0CF6                        .WORD   125$-<.-2>
        45 43 52 4F 46 4E 00' 0CF8                  .ASCIC  'NFORCE'
                    06  0CF8
                        0CFF  1770          $DS_CLI CLI$K_BR,       0,      EOL                          ;
                 00 82  0CFF                        .BYTE   CLI$K_BR,0
                  F7B0  0D01                        .WORD   EOL-<.-2>
```

```
                        0D03  1771 ;
                        0D03  1772 ; CLear  EVent  [Flags  ]flag_num , flag_num , ...
                        0D03  1773 ; CLear  EVent  [Flags  ]All
                        0D03  1774 ;
                        0D03  1775 125$:   $DS_CLI CLI$K_STRING,   EVENT,  Illegal,<'VENT'>        ; EVENT
                3A'8B   0D03                       .BYTE   CLI$K_STRING,EVENT
                 F7C4   0D05                       .WORD   Illegal-<.-2>
      54 4E 45 56 00'   0D07                       .ASCIC  'VENT'
                   04   0D07
                        0D0C  1776         $DS_CLI CLI$K_BR,       0,      EVENT_FLAGS             ; Go To Generic Pars
                00 82   0D0C                       .BYTE   CLI$K_BR,0
                 F918   0D0E                       .WORD   EVENT_FLAGS-<.-2>
                        0D10  1777 ;
                        0D10  1778 ; CLear  [FLAGS  ]x , x , x , ...
                        0D10  1779 ;
                        0D10  1780 170$:   $DS_CLI CLI$K_BR,       0,      FLAGS_LIST              ; Read Flags list
                00 82   0D10                       .BYTE   CLI$K_BR,0
                 F807   0D12                       .WORD   FLAGS_LIST-<.-2>
```

```
                        0D14  1782          .SUBTITLE       Set Parameters and Qualifiers
                        0D14  1783 ;
                        0D14  1784 ; SET parameters and qualifiers.
                        0D14  1785 ;
                        0D14  1786 ;        SET  BASE
                        0D14  1787 ;        SET  BREAKPOINTS...
                        0D14  1788 ;        SET  CPU
                        0D14  1789 ;        SET  CRD/TRACE
                        0D14  1790 ;        SET  DEFAULTS
                        0D14  1791 ;        SET  ENFORCE
                        0D14  1792 ;        SET  EVENT...
                        0D14  1793 ;        SET  FLAGS...
                        0D14  1794 ;        SET  y,y,y,...
                        0D14  1795 ;        SET  LOAD
                        0D14  1796 ;        SET  MEMORY...
                        0D14  1797 ;        SET  MM...
                        0D14  1798 ;        SET  PAGE
                        0D14  1799 ;        SET  QA...
                        0D14  1800 ;        SET  WIDTH
                        0D14  1801 ;
                        0D14  1802 SET_PARAMS:
                        0D14  1803          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required <SP>'s
                  00 84 0D14                .BYTE   CLI$K_SPACE,0
                   F7B3 0D16                .WORD   ILLEGAL-<.-2>
                        0D18  1804          $DS_CLI <^A"B">,        NOTNUF, 101$                    ; B             [65]
                18'42   0D18                .BYTE   ^A"B",NOTNUF
                 003F'  0D1A                .WORD   101$-<.-2>
                        0D1C  1805 ;
                        0D1C  1806 ; SET  BAse  address
                        0D1C  1807 ;
                        0D1C  1808          $DS_CLI CLI$K_STRING,   BASE,   90$,    <'ASE'>          ; BASE
                   3C'8B 0D1C               .BYTE   CLI$K_STRING,BASE
                    0014' 0D1E              .WORD   90$-<.-2>
               45 53 41 00' 0D20            .ASCIC  'ASE'
                      03    0D20
                        0D24  1809          $DS_CLI CLI$K_SPACE,    NOTNUF, ILLEGAL                 ; Required <SP>'s
                   18'84 0D24               .BYTE   CLI$K_SPACE,NOTNUF
                    F7A3 0D26               .WORD   ILLEGAL-<.-2>
                        0D28  1810          $DS_CLI CLI$K_NUM,      ADDRESS,ILLEGAL                 ; BASE Address
                   3D'85 0D28               .BYTE   CLI$K_NUM,ADDRESS
                    F79F 0D2A               .WORD   ILLEGAL-<.-2>
                        0D2C  1811          $DS_CLI CLI$K_BR,       ENUF,   EOL
                   17'82 0D2C               .BYTE   CLI$K_BR,ENUF
                    F783 0D2E               .WORD   EOL-<.-2>
                        0D30  1812 ;
                        0D30  1813 ; SET  BReakpoints<qualifiers>  num_address<qualifiers>
                        0D30  1814 ;
                        0D30  1815 90$:     $DS_CLI CLI$K_STRING,   BREAK,  115$,   <'REAKPOINTS'>  ;
                   3E'8B 0D30               .BYTE   CLI$K_STRING,BREAK
                    0023' 0D32              .WORD   115$-<.-2>
53 54 4E 49 4F 50 4B 41 45 52 00' 0D34      .ASCIC  'REAKPOINTS'
                      0A    0D34
                        0D3F  1816          $DS_CLI CLI$K_CALL,     0,      BREAK_QUALS             ; Look For Qualifier
                   00 92 0D3F               .BYTE   CLI$K_CALL,0
                    FB54 0D41               .WORD   BREAK_QUALS-<.-2>
                        0D43  1817          $DS_CLI CLI$K_SPACE,    0,      ILLEGAL
                   00 84 0D43               .BYTE   CLI$K_SPACE,0
```

```
                     F784  0D45                       .WORD   ILLEGAL-<.-2>
                           0D47  1818          $DS_CLI CLI$K_NUM,      ADDRESS,ILLEGAL                 ;Read Address
                   3D'85   0D47                       .BYTE   CLI$K_NUM,ADDRESS
                     F780  0D49                       .WORD   ILLEGAL-<.-2>
                           0D4B  1819          $DS_CLI CLI$K_CALL,     0,      BREAK_QUALS             ; Look For Qualifier
                   00 92   0D4B                       .BYTE   CLI$K_CALL,0
                     FB48  0D4D                       .WORD   BREAK_QUALS-<.-2>
                           0D4F  1820          $DS_CLI CLI$K_BR,       0,      EOL                     ; Done Parsing
                   00 82   0D4F                       .BYTE   CLI$K_BR,0
                     F760  0D51                       .WORD   EOL-<.-2>
                           0D53  1821
                           0D53  1822 115$:    $DS_CLI CLI$K_BR,       BACKUP, 350$                    ; Backup over Bell
                   64'82   0D53                       .BYTE   CLI$K_BR,BACKUP
                    01A9'  0D55                       .WORD   350$-<.-2>
                           0D57  1823 ;
                           0D57  1824 ; SET  CPu  name
                           0D57  1825 ;
                           0D57  1826 101$:    $DS_CLI <^A"C">,        NOTNUF, 120$                    ; C
                   18'43   0D57                       .BYTE   ^A"C",NOTNUF
                     0054' 0D59                       .WORD   120$-<.-2>
                           0D5B  1827          $DS_CLI CLI$K_STRING,   0,      112$,   <'PU'>          ; CPU
                   00 8B   0D5B                       .BYTE   CLI$K_STRING,0
                     001B' 0D5D                       .WORD   112$-<.-2>
                55 50 00'  0D5F                       .ASCIC  'PU'
                       02  0D5F
                           0D62  1828          $DS_CLI CLI$K_SPACE,    NOTNUF, ILLEGAL                 ; Required <SP>'s
                   18'84   0D62                       .BYTE   CLI$K_SPACE,NOTNUF
                     F765  0D64                       .WORD   ILLEGAL-<.-2>
                           0D66  1829          $DS_CLI CLI$K_BR,       FILESPEC,110$                   ; Put device name in
                   28'82   0D66                       .BYTE   CLI$K_BR,FILESPEC
                     0004' 0D68                       .WORD   110$-<.-2>
                           0D6A  1830 110$:    $DS_CLI CLI$K_ALNUM,    FILEND, 118$                    ; CLI$Q_FILE quadwor
                   29'87   0D6A                       .BYTE   CLI$K_ALNUM,FILEND
                     0025' 0D6C                       .WORD   118$-<.-2>
                           0D6E  1831          $DS_CLI <^A":">,        0,      111$                    ; Optional :
                   00 3A   0D6E                       .BYTE   ^A":",0
                     0004' 0D70                       .WORD   111$-<.-2>
                           0D72  1832 111$:    $DS_CLI CLI$K_BR,       SETCPU, EOL                     ; stop here     [101
                   39'82   0D72                       .BYTE   CLI$K_BR,SETCPU
                     F73D  0D74                       .WORD   EOL-<.-2>
                           0D76  1833 ;
                           0D76  1834 ; SET  Crd[ / Trace]
                           0D76  1835 ; SET  Crd[ / Debug / Trace ]
                           0D76  1836 ;
                           0D76  1837 112$:    $DS_CLI CLI$K_STRING,   0,      120$,   <'RD'>          ; CRD           [69]
                   00 8B   0D76                       .BYTE   CLI$K_STRING,0
                     0035' 0D78                       .WORD   120$-<.-2>
                44 52 00'  0D7A                       .ASCIC  'RD'
                       02  0D7A
                           0D7D  1838          $DS_Cli CLI$K_SLASH,    0,      117$,                   ; CRD/          [69]
                   00 90   0D7D                       .BYTE   CLI$K_SLASH,0
                     000E' 0D7F                       .WORD   117$-<.-2>
                           0D81  1839          $DS_Cli CLI$K_STRING,   0,      118$,   <'TRACE'>       ; CRD/TRACE     [69]
                   00 8B   0D81                       .BYTE   CLI$K_STRING,0
                     000E' 0D83                       .WORD   118$-<.-2>
45 43 41 52 54 00'  0D85                       .ASCIC  'TRACE'
                       05  0D85
```

```
                                0D8B  1840 117$:   $DS_CLI CLI$K_BR,       SET_CRD_TRACE,EOL                ;                [69]
                          1F'82  0D8B                        .BYTE   CLI$K_BR,SET_CRD_TRACE
                           F724  0D8D                        .WORD   EOL-<.-2>
                                0D8F  1841
                                0D8F  1842 118$:   $DS_CLI CLI$K_STRING,   NOTNUF, BAD_QUAL,<'DEBUG'>       ; CRD/DEBUG      [69]
                          18'8B  0D8F                        .BYTE   CLI$K_STRING,NOTNUF
                           F76C  0D91                        .WORD   BAD_QUAL-<.-2>
                47 55 42 45 44 00' 0D93                        .ASCIC  'DEBUG'
                             05  0D93
                                0D99  1843         $DS_CLI CLI$K_SLASH,    NOTNUF, BAD_QUAL,                ; CRD/DEBUG/     [69]
                          18'90  0D99                        .BYTE   CLI$K_SLASH,NOTNUF
                           F762  0D9B                        .WORD   BAD_QUAL-<.-2>
                                0D9D  1844         $DS_CLI CLI$K_STRING,   0,      BAD_QUAL,<'TRACE'>       ; CRD/DEBUG/TRACE[69
                          00 8B  0D9D                        .BYTE   CLI$K_STRING,0
                           F75E  0D9F                        .WORD   BAD_QUAL-<.-2>
                45 43 41 52 54 00' 0DA1                        .ASCIC  'TRACE'
                             05  0DA1
                                0DA7  1845         $DS_CLI CLI$K_BR,       SET_CRD_DEBUG,EOL                ;                [69]
                          20'82  0DA7                        .BYTE   CLI$K_BR,SET_CRD_DEBUG
                           F708  0DA9                        .WORD   EOL-<.-2>
                                0DAB  1846 ;
                                0DAB  1847 ; SET  Defaults...
                                0DAB  1848 ;
                                0DAB  1849 120$:   $DS_CLI CLI$K_STRING,   DEFAULT,130$,   <'DEFAULTS'>     ; DEFAULTS
                          43'8B  0DAB                        .BYTE   CLI$K_STRING,DEFAULT
                           0011' 0DAD                        .WORD   130$-<.-2>
        53 54 4C 55 41 46 45 44 00' 0DAF                        .ASCIC  'DEFAULTS'
                             08  0DAF
                                0DB8  1850         $DS_CLI CLI$K_BR,       NOTNUF, SET_DEFAULT_PARAMS       ; Get parameters
                          18'82  0DB8                        .BYTE   CLI$K_BR,NOTNUF
                           0148' 0DBA                        .WORD   SET_DEFAULT_PARAMS-<.-2>
                                0DBC  1851 ;
                                0DBC  1852 ; SET  ENforce
                                0DBC  1853 ;
                                0DBC  1854 130$:   $Ds_Cli <^A"E">,        0,      190$                     ; an E?
                          00 45  0DBC                        .BYTE   ^A"E",0
                           0020' 0DBE                        .WORD   190$-<.-2>
                                0DC0  1855         $DS_CLI CLI$K_STRING,   0,      135$, <'NFORCE'>         ; ENFORCE
                          00 8B  0DC0                        .BYTE   CLI$K_STRING,0
                           000F' 0DC2                        .WORD   135$-<.-2>
            45 43 52 4F 46 4E 00' 0DC4                        .ASCIC  'NFORCE'
                             06  0DC4
                                0DCB  1856         $DS_CLI CLI$K_BR,       SETENFORCE,EOL                   ;
                          36'82  0DCB                        .BYTE   CLI$K_BR,SETENFORCE
                           F6E4  0DCD                        .WORD   EOL-<.-2>
                                0DCF  1857 ;
                                0DCF  1858 ; SET  EVent  [Flags  ]flag_num , flag_num , ...
                                0DCF  1859 ; SET  EVent  [Flags  ]All
                                0DCF  1860 ;
                                0DCF  1861 135$:   $DS_CLI CLI$K_STRING,   EVENT,  ILLEGAL,<'VENT'>         ; EVENT
                          3A'8B  0DCF                        .BYTE   CLI$K_STRING,EVENT
                           F6F8  0DD1                        .WORD   ILLEGAL-<.-2>
                54 4E 45 56 00' 0DD3                        .ASCIC  'VENT'
                             04  0DD3
                                0DD8  1862         $DS_CLI CLI$K_BR,       0,      EVENT_FLAGS              ; Generic Event Pars
                          00 82  0DD8                        .BYTE   CLI$K_BR,0
                           F84C  0DDA                        .WORD   EVENT_FLAGS-<.-2>
```

```
                        0DDC  1863 ;
                        0DDC  1864 ; SET  LOAd  <file-spec>
                        0DDC  1865 ;
                        0DDC  1866 190$:   $DS_CLI <^A"L">,         NOTNUF, 220$                    ; L
                 18'4C  0DDC                .BYTE   ^A"L",NOTNUF
                  0023' 0DDE                .WORD   220$-<.-2>
                        0DE0  1867          $DS_CLI <^A"O">,         NOTNUF, ILLEGAL                 ; LO
                 18'4F  0DE0                .BYTE   ^A"O",NOTNUF
                  F6E7  0DE2                .WORD   ILLEGAL-<.-2>
                        0DE4  1868          $DS_CLI CLI$K_STRING,    SETLOAD,200$,   <'AD'>          ; LOAD
                 7F'8B  0DE4                .BYTE   CLI$K_STRING,SETLOAD
                  0013' 0DE6                .WORD   200$-<.-2>
              44 41 00' 0DE8                .ASCIC  'AD'
                    02  0DE8
                        0DEB  1869          $DS_CLI CLI$K_SPACE,     NOTNUF, ILLEGAL                 ; Required spaces
                 18'84  0DEB                .BYTE   CLI$K_SPACE,NOTNUF
                  F6DC  0DED                .WORD   ILLEGAL-<.-2>
                        0DEF  1870          $DS_CLI CLI$K_CALL,      REQUIRED,GET_FILE_SPEC          ; Required file-spec
                 2B'92  0DEF                .BYTE   CLI$K_CALL,REQUIRED
                  F85F  0DF1                .WORD   GET_FILE_SPEC-<.-2>
                        0DF3  1871          $DS_CLI CLI$K_BR,        ENUF, EOL                       ; Done With Command
                 17'82  0DF3                .BYTE   CLI$K_BR,ENUF
                  F6BC  0DF5                .WORD   EOL-<.-2>
                        0DF7  1872
                        0DF7  1873 200$:   $DS_CLI CLI$K_BR,        BACKUP, 210$                    ; Backup over "0"
                 64'82  0DF7                .BYTE   CLI$K_BR,BACKUP
                  0004' 0DF9                .WORD   210$-<.-2>
                        0DFB  1874 210$:   $DS_CLI CLI$K_BR,        BACKUP, 350$                    ; Backup over "L", R
                 64'82  0DFB                .BYTE   CLI$K_BR,BACKUP
                  0101' 0DFD                .WORD   350$-<.-2>
                        0DFF  1875 ;                                                                [75]
                        0DFF  1876 ; SET  MEmory  n
                        0DFF  1877 ;                                                                [75]
                        0DFF  1878 220$:   $DS_CLI <^A"M">,         NOTNUF, 235$                    ; M             [75]
                 18'4D  0DFF                .BYTE   ^A"M",NOTNUF
                  003E' 0E01                .WORD   235$-<.-2>
                        0E03  1879          $DS_CLI CLI$K_STRING,    SETMEM, 225$,   <'EMORY'>       ; MEMORY        [75]
                 8A'8B  0E03                .BYTE   CLI$K_STRING,SETMEM
                  0016' 0E05                .WORD   225$-<.-2>
        59 52 4F 4D 45 00' 0E07             .ASCIC  'EMORY'
                    05  0E07
                        0E0D  1880          $DS_CLI CLI$K_SPACE,     0,      ILLEGAL                 ; Required <SP>'s[75
                 00 84  0E0D                .BYTE   CLI$K_SPACE,0
                  F6BA  0E0F                .WORD   ILLEGAL-<.-2>
                        0E11  1881          $DS_CLI CLI$K_DEC,       DATA,   ILLEGAL                 ; data          [75]
                 66'8A  0E11                .BYTE   CLI$K_DEC,DATA
                  F6B6  0E13                .WORD   ILLEGAL-<.-2>
                        0E15  1882          $DS_CLI CLI$K_BR,        ENUF,   EOL                     ;               [75]
                 17'82  0E15                .BYTE   CLI$K_BR,ENUF
                  F69A  0E17                .WORD   EOL-<.-2>
```

```
                        0E19  1884 ; SET  MM  ON
                        0E19  1885 ; SET  MM  OFf
                        0E19  1886 ;
                        0E19  1887 225$:   $DS_CLI CLI$K_STRING,   NOTNUF, 235$,   <'MM'>          ; MM              [75]
                  18'8B 0E19                       .BYTE   CLI$K_STRING,NOTNUF
                   0024' 0E1B                      .WORD   235$-<.-2>
               4D 4D 00' 0E1D                      .ASCIC  'MM'
                     02  0E1D
                        0E20  1888         $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Required <SP>'s
                  00 84 0E20                       .BYTE   CLI$K_SPACE,0
                   F6A7 0E22                       .WORD   ILLEGAL-<.-2>
                        0E24  1889         $DS_CLI <^A"O">,        NOTNUF, ILLEGAL                 ; MM O
                  18'4F 0E24                       .BYTE   ^A"O",NOTNUF
                   F6A3 0E26                       .WORD   ILLEGAL-<.-2>
                        0E28  1890         $DS_CLI CLI$K_STRING,   MMON,   230$,   <'N'>           ; MM ON
                  2F'8B 0E28                       .BYTE   CLI$K_STRING,MMON
                   000A' 0E2A                      .WORD   230$-<.-2>
                  4E 00' 0E2C                      .ASCIC  'N'
                     01  0E2C
                        0E2E  1891         $DS_CLI CLI$K_BR,       ENUF,   EOL
                  17'82 0E2E                       .BYTE   CLI$K_BR,ENUF
                   F681 0E30                       .WORD   EOL-<.-2>
                        0E32  1892
                        0E32  1893 230$:   $DS_CLI CLI$K_STRING,   MMOFF,  ILLEGAL,<'FF'>          ; MM OFF
                  30'8B 0E32                       .BYTE   CLI$K_STRING,MMOFF
                   F695 0E34                       .WORD   ILLEGAL-<.-2>
               46 46 00' 0E36                      .ASCIC  'FF'
                     02  0E36
                        0E39  1894         $DS_CLI CLI$K_BR,       ENUF,   EOL
                  17'82 0E39                       .BYTE   CLI$K_BR,ENUF
                   F676 0E3B                       .WORD   EOL-<.-2>
                        0E3D  1895 ;
                        0E3D  1896 ; SET  PAGE  n
                        0E3D  1897 ;
                        0E3D  1898 235$:   $DS_CLI <^A"P">,        NOTNUF, 240$                 ;                 [71]
                  18'50 0E3D                       .BYTE   ^A"P",NOTNUF
                   001C' 0E3F                      .WORD   240$-<.-2>
                        0E41  1899         $DS_CLI CLI$K_STRING,   PAGE,   237$,   <'AGE'>         ; PAGE            [71]
                  5A'8B 0E41                       .BYTE   CLI$K_STRING,PAGE
                   0014' 0E43                      .WORD   237$-<.-2>
            45 47 41 00' 0E45                      .ASCIC  'AGE'
                     03  0E45
                        0E49  1900         $DS_CLI CLI$K_SPACE,    NOTNUF, ILLEGAL                 ; required space[71]
                  18'84 0E49                       .BYTE   CLI$K_SPACE,NOTNUF
                   F67E 0E4B                       .WORD   ILLEGAL-<.-2>
                        0E4D  1901         $DS_CLI CLI$K_DEC,      DATA,   ILLEGAL                 ; data            [71]
                  66'8A 0E4D                       .BYTE   CLI$K_DEC,DATA
                   F67A 0E4F                       .WORD   ILLEGAL-<.-2>
                        0E51  1902         $DS_CLI CLI$K_BR,       ENUF,   EOL                     ;                 [71]
                  17'82 0E51                       .BYTE   CLI$K_BR,ENUF
                   F65E 0E53                       .WORD   EOL-<.-2>
                        0E55  1903
                        0E55  1904 237$:   $DS_CLI CLI$K_BR,       BACKUP, 350$                ; Back up 'P', Check Flags L
                  64'82 0E55                       .BYTE   CLI$K_BR,BACKUP
                   00A7' 0E57                      .WORD   350$-<.-2>
```

```
                        0E59  1906 ; SET  QACklooploops  x
                        0E59  1907 ; SET  QADefaults
                        0E59  1908 ; SET  QAErrorprints  x
                        0E59  1909 ; SET  QAMultiplepass  x
                        0E59  1910 ; SET  QASubtestloops  x
                        0E59  1911 ; SET  QATestloops  x
                        0E59  1912 ;
                        0E59  1913 240$:   $DS_CLI <^A"Q">,        NOTNUF, 300$                    ; Q              [42]
                 18'51  0E59                .BYTE   ^A"Q",NOTNUF
                  008D' 0E5B                .WORD   300$-<.-2>
                        0E5D  1914          $DS_CLI <^A"A">,        NOTNUF, 295$                    ; QA             [42]
                 18'41  0E5D                .BYTE   ^A"A",NOTNUF
                  0085' 0E5F                .WORD   295$-<.-2>
                        0E61  1915
                        0E61  1916          $DS_CLI CLI$K_STRING,   QACL,   250$,   <'CKLOOPLOOPS'> ; QACKLOOPLOOPS [42]
                 53'8B  0E61                .BYTE   CLI$K_STRING,QACL
                  0014' 0E63                .WORD   250$-<.-2>
53 50 4F 4F 4C 50 4F 4F 4C 4B 43 00' 0E65   .ASCIC  'CKLOOPLOOPS'
                    0B  0E65
                        0E71  1917          $DS_CLI CLI$K_BR,       0,      291$                    ; Read QA Data
                 00 82  0E71                .BYTE   CLI$K_BR,0
                  0065' 0E73                .WORD   291$-<.-2>
                        0E75  1918
                        0E75  1919 250$:   $DS_CLI CLI$K_STRING,   QADEF,  260$,   <'DEFAULTS'>    ; QADEFAULTS     [42]
                 57'8B  0E75                .BYTE   CLI$K_STRING,QADEF
                  0011' 0E77                .WORD   260$-<.-2>
      53 54 4C 55 41 46 45 44 00' 0E79      .ASCIC  'DEFAULTS'
                    08  0E79
                        0E82  1920          $DS_CLI CLI$K_BR,       ENUF,   EOL                     ; No data        [42]
                 17'82  0E82                .BYTE   CLI$K_BR,ENUF
                  F62D  0E84                .WORD   EOL-<.-2>
                        0E86  1921
                        0E86  1922 260$:   $DS_CLI CLI$K_STRING,   QAEP,   270$,   <'ERRORPRINTS'> ; QAERRORPRINTS [42]
                 52'8B  0E86                .BYTE   CLI$K_STRING,QAEP
                  0014' 0E88                .WORD   270$-<.-2>
53 54 4E 49 52 50 52 4F 52 52 45 00' 0E8A   .ASCIC  'ERRORPRINTS'
                    0B  0E8A
                        0E96  1923          $DS_CLI CLI$K_BR,       0,      291$                    ; Read QA Data
                 00 82  0E96                .BYTE   CLI$K_BR,0
                  0040' 0E98                .WORD   291$-<.-2>
                        0E9A  1924
                        0E9A  1925 270$:   $DS_CLI CLI$K_STRING,   QAMP,   280$,   <'MULTIPLEPASS'>; QAMULTIPLEPASS[42]
                 56'8B  0E9A                .BYTE   CLI$K_STRING,QAMP
                  0015' 0E9C                .WORD   280$-<.-2>
53 41 50 45 4C 50 49 54 4C 55 4D 00' 0E9E   .ASCIC  'MULTIPLEPASS'
                    53  0EAA
                    0C  0E9E
                        0EAB  1926          $DS_CLI CLI$K_BR,       0,      291$                    ; Read QA Data
                 00 82  0EAB                .BYTE   CLI$K_BR,0
                  002B' 0EAD                .WORD   291$-<.-2>
                        0EAF  1927
                        0EAF  1928 280$:   $DS_CLI CLI$K_STRING,   QASL,   290$,   <'SUBTESTLOOPS'>; QASUBTESTLOOPS[42]
                 55'8B  0EAF                .BYTE   CLI$K_STRING,QASL
                  0015' 0EB1                .WORD   290$-<.-2>
50 4F 4F 4C 54 53 45 54 42 55 53 00' 0EB3   .ASCIC  'SUBTESTLOOPS'
                    53  0EBF
                    0C  0EB3
```

```
                        0EC0  1929          $DS_CLI CLI$K_BR,        0,      291$                    ; Read QA Data
                 00 82  0EC0                        .BYTE   CLI$K_BR,0
                  0016' 0EC2                        .WORD   291$-<.-2>
                        0EC4  1930
                        0EC4  1931 290$:    $DS_CLI CLI$K_STRING,   QATL,   ILLEGAL,<'TESTLOOPS'>   ; QATESTLOOPS   [42]
                 54'8B  0EC4                        .BYTE   CLI$K_STRING,QATL
                  F603  0EC6                        .WORD   ILLEGAL-<.-2>
53 50 4F 4F 4C 54 53 45 54 00' 0EC8                 .ASCIC  'TESTLOOPS'
                    09  0EC8
                        0ED2  1932          $DS_CLI CLI$K_BR,        0,      291$                    ; Read QA Data
                 00 82  0ED2                        .BYTE   CLI$K_BR,0
                  0004' 0ED4                        .WORD   291$-<.-2>
                        0ED6  1933
                        0ED6  1934 291$:    $DS_CLI CLI$K_SPACE,    0,      ILLEGAL                 ; Read QA Data
                 00 84  0ED6                        .BYTE   CLI$K_SPACE,0
                  F5F1  0ED8                        .WORD   ILLEGAL-<.-2>
                        0EDA  1935          $DS_CLI CLI$K_DEC,      DATA,   ILLEGAL                 ;
                 66'8A  0EDA                        .BYTE   CLI$K_DEC,DATA
                  F5ED  0EDC                        .WORD   ILLEGAL-<.-2>
                        0EDE  1936          $DS_CLI CLI$K_BR,        0,      EOL                     ;
                 00 82  0EDE                        .BYTE   CLI$K_BR,0
                  F5D1  0EE0                        .WORD   EOL-<.-2>
                        0EE2  1937
                        0EE2  1938 295$:    $DS_CLI CLI$K_BR,        BACKUP, 350$                    ; Back up 'Q', Check
                 64'82  0EE2                        .BYTE   CLI$K_BR,BACKUP
                  001A' 0EE4                        .WORD   350$-<.-2>
                        0EE6  1939 ;
                        0EE6  1940 ;
                        0EE6  1941 ; SET WIDTH x
                        0EE6  1942 ;
                        0EE6  1943 300$:    $DS_CLI CLI$K_STRING,   WIDTH,  350$,<'WIDTH'>; WIDTH                   ;G:4
              5B'8B     0EE6                        .BYTE   CLI$K_STRING,WIDTH
               0016'    0EE8                        .WORD   350$-<.-2>
48 54 44 49 57 00'      0EEA                        .ASCIC  'WIDTH'
                   05   0EEA
                        0EF0  1944          $DS_CLI CLI$K_SPACE,    NOTNUF, ILLEGAL                 ; Required <SP>'s
              18'84     0EF0                        .BYTE   CLI$K_SPACE,NOTNUF
               F5D7     0EF2                        .WORD   ILLEGAL-<.-2>
                        0EF4  1945          $DS_CLI CLI$K_DEC,      DATA,   ILLEGAL                 ; WIDTH x
              66'8A     0EF4                        .BYTE   CLI$K_DEC,DATA
               F5D3     0EF6                        .WORD   ILLEGAL-<.-2>
                        0EF8  1946          $DS_CLI CLI$K_BR,        ENUF,   EOL
              17'82     0EF8                        .BYTE   CLI$K_BR,ENUF
               F5B7     0EFA                        .WORD   EOL-<.-2>
                        0EFC  1947 ;
                        0EFC  1948 ; SET  [FLAGS  ]x , x , x , ...
                        0EFC  1949 ;
                        0EFC  1950 350$:    $DS_CLI CLI$K_BR,        0,      FLAGS_LIST              ; Read Flags list
              00 82     0EFC                        .BYTE   CLI$K_BR,0
               F61B     0EFE                        .WORD   FLAGS_LIST-<.-2>
```

```
                     0F00  1952          .SUBTITLE       Set Default Parameters
                     0F00  1953 ;
                     0F00  1954 ; SET DEFAULT parameters.
                     0F00  1955 ;
                     0F00  1956 ; This allows a list of sizes and radixes, each one separated
                     0F00  1957 ; by a comma. Spaces may appear on either side of the commas.
                     0F00  1958 ; If conflicting radixes (sizes) are given, the "biggest" one
                     0F00  1959 ; is used. That is,
                     0F00  1960 ;       HEXADECIMAL > DECIMAL > OCTAL
                     0F00  1961 ;       LONGWORD > WORD > BYTE
                     0F00  1962 ;
                     0F00  1963
                     0F00  1964 SET_DEFAULT_PARAMS:
                     0F00  1965          $DS_CLI CLI$K_SPACE     0,      ILLEGAL                 ; Required <SP>'s
                00 84  0F00                      .BYTE   CLI$K_SPACE,0
                 F5C7  0F02                      .WORD   ILLEGAL-<.-2>
```

```
                                0F04  1967 ;
                                0F04  1968 ; SET DEFAULT BYTE
                                0F04  1969 ;
                                0F04  1970 100$:   $DS_CLI CLI$K_STRING, BYTE,     110$,   <'BYTE'>         ; BYTE
                      6A'8B     0F04                       .BYTE   CLI$K_STRING,BYTE
                       000D'    0F06                       .WORD   110$-<.-2>
                 45 54 59 42 00' 0F08                       .ASCIC  'BYTE'
                         04     0F08
                                0F0D  1971         $DS_CLI CLI$K_BR,        0,      160$
                      00 82     0F0D                       .BYTE   CLI$K_BR,0
                       0054'    0F0F                       .WORD   160$-<.-2>
                                0F11  1972 ;
                                0F11  1973 ;  SET DEFAULT DECIMAL
                                0F11  1974 ;
                                0F11  1975 110$:   $DS_CLI CLI$K_STRING, DEC,      120$,   <'DECIMAL'>      ; DECIMAL
                      6C'8B     0F11                       .BYTE   CLI$K_STRING,DEC
                       0010'    0F13                       .WORD   120$-<.-2>
           4C 41 4D 49 43 45 44 00' 0F15                    .ASCIC  'DECIMAL'
                         07     0F15
                                0F1D  1976         $DS_CLI CLI$K_BR,        0,      160$
                      00 82     0F1D                       .BYTE   CLI$K_BR,0
                       0044'    0F1F                       .WORD   160$-<.-2>
                                0F21  1977 ;
                                0F21  1978 ; SET DEFAULT HEXADECIMAL
                                0F21  1979 ;
                                0F21  1980 120$:   $DS_CLI CLI$K_STRING, HEX,      130$,   <'HEXADECIMAL'> ; HEXADECIMAL
                      6B'8B     0F21                       .BYTE   CLI$K_STRING,HEX
                       0014'    0F23                       .WORD   130$-<.-2>
4C 41 4D 49 43 45 44 41 58 45 48 00' 0F25                  .ASCIC  'HEXADECIMAL'
                         0B     0F25
                                0F31  1981         $DS_CLI CLI$K_BR,        0,      160$
                      00 82     0F31                       .BYTE   CLI$K_BR,0
                       0030'    0F33                       .WORD   160$-<.-2>
                                0F35  1982 ;
                                0F35  1983 ; SET DEFAULT LONGWORD
                                0F35  1984 ;
                                0F35  1985 130$:   $DS_CLI CLI$K_STRING, LONG,     140$,   <'LONGWORD'>     ; LONGWORD
                      68'8B     0F35                       .BYTE   CLI$K_STRING,LONG
                       0011'    0F37                       .WORD   140$-<.-2>
        44 52 4F 57 47 4E 4F 4C 00' 0F39                    .ASCIC  'LONGWORD'
                         08     0F39
                                0F42  1986         $DS_CLI CLI$K_BR,        0,      160$
                      00 82     0F42                       .BYTE   CLI$K_BR,0
                       001F'    0F44                       .WORD   160$-<.-2>
                                0F46  1987 ;
                                0F46  1988 ; SET DEFAULT OCTAL
                                0F46  1989 ;
                                0F46  1990 140$:   $DS_CLI CLI$K_STRING, OCT,      150$,   <'OCTAL'>        ; OCTAL
                      6D'8B     0F46                       .BYTE   CLI$K_STRING,OCT
                       000E'    0F48                       .WORD   150$-<.-2>
                 4C 41 54 43 4F 00' 0F4A                    .ASCIC  'OCTAL'
                         05     0F4A
                                0F50  1991         $DS_CLI CLI$K_BR,        0,      160$
                      00 82     0F50                       .BYTE   CLI$K_BR,0
                       0011'    0F52                       .WORD   160$-<.-2>
                                0F54  1992 ;
                                0F54  1993 ; SET DEFAULT WORD
```

```
ZZ-ENSAA-10.10  Set Default Parameters                              3-MAR-1988       Fiche 4 Frame J13          Sequence 783
CLI                      DIAGNOSTIC SUPERVISOR COMMAND LINE INTER 11-NOV-1987 17:26:56  VAX/VMS Macro V04-00          Page  77
10.10-28                 Set Default Parameters                    6-OCT-1987 20:26:22  [DS.LOCAL.RELEASE.WORK]CLI.MAR;1  (1)

                         0F54  1994 ;
                         0F54  1995 150$:   $DS_CLI CLI$K_STRING, WORD,     BAD_LIST,<'WORD'>       ; WORD
                69'8B    0F54                       .BYTE   CLI$K_STRING,WORD
                 F58B    0F56                       .WORD   BAD_LIST-<.-2>
          44 52 4F 57 00' 0F58                       .ASCIC  'WORD'
                   04    0F58
                         0F5D  1996         $DS_CLI CLI$K_BR,        0,      160$
                00 82    0F5D                       .BYTE   CLI$K_BR,0
                 0004'   0F5F                       .WORD   160$-<.-2>
                         0F61  1997 ;
                         0F61  1998 ; If there is a comma, get it and go back for more.
                         0F61  1999 ; If not, it must be end of line time.
                         0F61  2000 ;
                         0F61  2001 160$:   $DS_CLI CLI$K_COMMA,  NOTNUF,   170$                    ; ,
                18'8E    0F61                       .BYTE   CLI$K_COMMA,NOTNUF
                 0008'   0F63                       .WORD   170$-<.-2>
                         0F65  2002         $DS_CLI CLI$K_BR,        0,      100$                   ; Go back for more
                00 82    0F65                       .BYTE   CLI$K_BR,0
                 FF9F    0F67                       .WORD   100$-<.-2>
                         0F69  2003 ;
                         0F69  2004 ; End of SET DEFAULT parameters. Assume EOL.
                         0F69  2005 ;
                         0F69  2006 170$:   $DS_CLI CLI$K_BR,     ENUF,     EOL                    ; Enough input
                17'82    0F69                       .BYTE   CLI$K_BR,ENUF
                 F546    0F6B                       .WORD   EOL-<.-2>
                         0F6D  2007 ;
                         0F6D  2008 ; End of SET DEFAULT
                         0F6D  2009 ;
```

```
                     0F6D  2011          .SBTTL  DS$Cli Routine - Command Line Interpreter
                 00000000  2012          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000  2013 ;++
                     0000  2014 ; FUNCTIONAL DESCRIPTION:
                     0000  2015 ;
                     0000  2016 ;       This routine provides the basic command interpreter for the
                     0000  2017 ;       supervisor.
                     0000  2018 ;
                     0000  2019 ; CALLING SEQUENCE:
                     0000  2020 ;
                     0000  2021 ;       DS$CLI ()
                     0000  2022 ;
                     0000  2023 ; INPUT PARAMETERS:
                     0000  2024 ;
                     0000  2025 ;       AP      Address of R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP,PC,PSL
                     0000  2026 ;
                     0000  2027 ;
                     0000  2028 ; IMPLICIT INPUTS:
                     0000  2029 ;
                     0000  2030 ;       NONE
                     0000  2031 ;
                     0000  2032 ; OUTPUT PARAMETERS:
                     0000  2033 ;
                     0000  2034 ;       NONE
                     0000  2035 ;
                     0000  2036 ; IMPLICIT OUTPUTS:
                     0000  2037 ;
                     0000  2038 ;       NONE
                     0000  2039 ;
                     0000  2040 ; COMPLETION CODES:
                     0000  2041 ;
                     0000  2042 ;       NONE
                     0000  2043 ;
                     0000  2044 ; SIDE EFFECTS:
                     0000  2045 ;
                     0000  2046 ;       ALL COMMAND SUBROUTINES ARE ENTERED WITH R2 POINTING TO
                     0000  2047 ;       "DS$GL_CLIBASE".
                     0000  2048 ;--
```

```
                        003C  0000  2050          .ENTRY  DS$CLI,^M<R2,R3,R4,R5>
                              0002  2051 RCLI:
          00000000'EF    7C   0002  2052          clrq    L^q_adapter                      ; Clear adapter string          [48]
          00000000'EF    94   0008  2053          ClrB    L^ShowMemoryFlags                ; Clear flags for show mem
     52   0000044C'EF    DE   000E  2054          MOVAL   L^DS$GL_CLIBASE+CLI$K_SIZE,R2    ; START AT THE END
                    72   D4   0015  2062          CLRL    -(R2)                            ; AFTER THIS IT IS QUAD ALIGNED
     50   00000089 8F    D0   0017  2064          MOVL    #CLI$K_SIZE/8,R0                 ; NUMBER OF QUAD WORDS REMAINING
                              001E  2065
                    72   7C   001E  2066 10$:     CLRQ    -(R2)                            ; CLEAR QUAD
                 FB 50   F5   0020  2067          SOBGTR  R0,10$                           ; CLEAR OTHER QUADWORDS
                              0023  2068
                              0023  2069 20$:     Br_If_Not_APT   CLI                      ; If not in APT mode, branch    [55]
     0000FE00'EF    1F   E1   0023                BBC     #DSA$V_Apt, -                    ; If bit not set, branch
                    47        002A                          L^DSA$GL_Flags, CLI            ;
  22 00000000'EF    00   E4   002B  2070          BBSC    #DS$V_CTRLC,DS$GL_FLAGS, 30$     ; Branch if ^C typed
          00000000'EF    16   0033  2071          JSB     APT_MSG                          ; WAIT FOR APT MESSAGE TO BE ACKED
          00000000'EF    16   0039  2072          JSB     KB_CHECK_APT                     ; CHECK APT
          0000FE04'EF    D5   003F  2073          TSTL    L^DSA$GL_APTCOM                  ; YES- IS THERE A COMMAND ?     [41]
                    DC   13   0045  2074          BEQL    20$                              ; NO - WAIT UNTIL THERE IS ONE
  04 A2   00000000'EF    DE   0047  2075          MOVAL   L^APT_COM,CLI$L_COMMAND(R2)      ; SET APT COMMAND DISPATCH      [41]
          00000000'EF    16   004F  2076          JSB     APT_COM                          ; ENTER APT COMMAND PROCESSOR
                              0055  2077
                              0055  2078 30$:     Clear_CtrlO                              ; Clear the ^O flag             [55]
     00000000'EF    10   E4   0055                BBSC    #DS$V_CtrlO, -           ; Branch if CTRLO bit is set to
                    00        005C                          L^DS$GL_Flags, 30600$ ; ...
                              005D        30600$:                                  ; ... here. Clear bit regardless.
                              005D  2079          $PRINT  #ds$k_type_ds_prompt, -          ; Print out the prompt          [48]
                              005D  2080                    #ds$k_printi, -                ; .. use PRINTI                 [48]
                              005D  2081                    L^DS$GT_PROMPT                 ; Print "DS> " prompt, tells    [41]
          00000000'EF    9F   005D                        PUSHAB  L^DS$GT_PROMPT
                7E   00   9A   0063                        movzbl  #ds$k_printi, -(sp)
                7E   01   98   0066                        cvtbl   #ds$k_type_ds_prompt, -(sp)
     00000000'GF    03   FB   0069                        CALLS   #$$N+2, G^DSX$PRINT
                              0070  2082                                                   ;  apt we're done               [41]
                    90   11   0070  2083          BRB     RCLI                             ; CHECK FOR ANOTHER COMMAND
```

```
                          0072  2085
                          0072  2086 CLI:
                          0072  2087          Br_If_CRD_AutoTest_Off  5$      ; If running under CRD Auto Test, branch[72]
0000FE00'EF   0B   E0     0072                BBS     #DSA$V_CRD_Autotest_Off, -      ; If bit set, branch            [36]
              12          0079                  L^DSA$GL_Flags, 5$            ;                               [36]
                          007A  2088          Br_If_CRD_MenuTest_Off  5$      ; If running under CRD Menu Test, branch[72]
0000FE00'EF   10   E0     007A                BBS     #DSA$V_CRD_MenuTest_Off, -      ; If bit set, branch            [36]
              0A          0081                  L^DSA$GL_Flags, 5$            ;                               [36]
                          0082  2089          Br_If_CRD_MenuTest_On   5$      ; If running under CRD Menu Test, branch[72]
0000FE00'EF   12   E0     0082                BBS     #DSA$V_CRD_MenuTest_On, -       ; If bit set, branch            [36]
              02          0089                  L^DSA$GL_Flags, 5$            ;                               [36]
              08   11     008A  2090          Brb     10$                     ; If not CRD, branch around             [67]
                          008C  2091
     00000000'EF   16     008C  2092 5$:      Jsb     KB_Check                ; Check for any ^C's                    [67]
              08   11     0092  2093          Brb     20$                     ; Branch around clearing ^C flag        [58]
                          0094  2094
                          0094  2095 10$:     Clear_CtrlC                     ; Clear the ^C flag                     [58]
00000000'EF   00   E4     0094                BBSC    #DS$V_CtrlC, -          ; Branch if CTRLC bit is set to         [29]
              00          009B                  L^DS$GL_Flags, 30601$ ; ...                                   [29]
                          009C        30601$:                               ; ... here. Clear bit regardless.       [29]
                          009C  2096
     00000000'EF   9F     009C  2097 20$:     PUSHAB  L^DS$GT_PROMPT          ; ADDRESS OF PROMPT                     [58]
     00000100 8F   DD     00A2  2098          PUSHL   #CLI$K_BUFSIZ           ; SIZE OF BUFFER
           34 A2   DF     00A8  2099          PUSHAL  CLI$Q_BUFQWD(R2)        ; WHERE TO PUT LENGTH
           3C A2   9F     00AB  2100          PUSHAB  CLI$T_BUFFER(R2)        ; WHERE TO PUT TEXT
00000000'EF   04   FB     00AE  2101          CALLS   #4,L^DS_SUPERLINE       ; GET SUPERVISOR LINE                   [41]
                          00B5  2102
     00000000'EF   D4     00B5  2103          CLRL    L^Line_count            ; Reinit # lines output for SET PAGE    [73]
                          00BB  2104
     50   0000'8F   B1     00BB  2105          CMPW    #SS$_ENDOFFILE,R0       ; EOF?
              09   12     00C0  2106          BNEQ    70$                     ; BRANCH IF NOT EOF
                          00C2  2107          $EXIT_S                         ; RETURN TO VMS COMMAND LEVEL
              01   DD     00C2                        PUSHL   #1
00000000'GF   01   FB     00C4                        CALLS   #1,G^SYS$EXIT
                          00CB  2108
           03 50   E8     00CB  2109 70$:     BLBS    R0,700$                 ; If no errors, continue                [41]
            FF31   31     00CE  2110          BRW     RCLI                    ; If error on input, try again          [41]
                          00D1  2111
                          00D1  2112 700$:                                    ;                                       [41]
              51   D5     00D1  2113          TSTL    R1                      ; CHECK FOR NULL STRING.
              03   12     00D3  2114          BNEQ    71$                     ; Don't loop if not zero                [40]
            FF2A   31     00D5  2115          BRW     RCLI                    ; Loop if zero                          [40]
                          00D8  2116
                          00D8  2117 71$:                                     ;                                       [40]
     38 A2   3C A2   DE   00D8  2118          MOVAL   CLI$T_BUFFER(R2), -
                          00DD  2119                    CLI$Q_BUFQWD+4(R2)
```

```
                          00DD  2121
                          00DD  2122 ;
                          00DD  2123 ; Parse the Supervisor line.
                          00DD  2124 ;
                          00DD  2125
          0000013D'EF   DF  00DD  2126          PUSHAL  L^CLI_ACTION            ; Call the special Supervisor          [45]
          00000295'EF   DF  00E3  2127          PUSHAL  L^COMMAND_TREE          ;  parser. The only difference between [45]
                34 A2   7F  00E9  2128          PUSHAQ  CLI$Q_BUFQWD(R2)        ;  it and the regular DS$PARSE is that [45]
    00000000'9F   03   FB  00EC  2129          CALLS   #3, a#DSX$SUPERPARSE    ;  it does not return when EOL (i.e.,  [45]
                          00F3  2130                                          ;  it continues parsing until CLI$K_EXIT[45]
                          00F3  2131                                          ;  is seen).                           [45]
                          00F3  2132
             3C 50   E8   00F3  2133          BLBS    R0, 90$                 ; If parsed successfully, branch       [45]
                          00F6  2134
                          00F6  2135 ;
                          00F6  2136 ; Parsed unsuccessfully. Check NOTNUF flag. If it is set,
                          00F6  2137 ; implies an illegal command (see ACT_INC_CMD). If it is
                          00F6  2138 ; clear, implies an incomplete command (see ACT_INC_CMD).
                          00F6  2139 ;
                          00F6  2140
    1A 00000000 E2   01   E0  00F6  2141          BBS     #CLI$V_NOTNUF, -        ; If NOTNUF set, branch                [45]
                          00FE  2142                    L^CLI$L_FLAGS (R2),75$;                                      [45]
                          00FE  2143
                          00FE  2144 ;
                          00FE  2145 ; ILLEGAL COMMAND
                          00FE  2146 ;
                          00FE  2147
          0000044C'EF   9F  00FE  2148          PUSHAB  L^Q_GOOD_CMD            ; Push desc. of good part              [43]
          0000001B'EF   9F  0104  2149          PUSHAB  L^T_ILLCMD              ; Push ILLEGAL COMMAND                 [43]
                   00   DD  010A  2150          pushl   #ds$k_printi            ; use PRINTI                           [48]
                   15   DD  010C  2151          pushl   #ds$k_type_command_err  ; Command error code                  [48]
    00000000'GF    04   FB  010E  2152          CALLS   #4, G^DSX$PRINT         ;                                      [48]
                FEEA   31  0115  2153          BRW     RCLI
                          0118  2154
                          0118  2155 ;
                          0118  2156 ; INCOMPLETE COMMAND
                          0118  2157 ;
                          0118  2158
          0000044C'EF   9F  0118  2159 75$:     PUSHAB  L^Q_GOOD_CMD            ; Push desc. of good part              [43]
          0000003A'EF   9F  011E  2160          PUSHAB  L^T_INCOMPLETE          ; Push INCOMPLETE COMMAND              [43]
                   00   DD  0124  2161          pushl   #ds$k_printi            ; use PRINTI                           [48]
                   15   DD  0126  2162          pushl   #ds$k_type_command_err  ; Command error code                  [48]
    00000000'GF    04   FB  0128  2163          CALLS   #4, G^DSX$PRINT         ;                                      [48]
                FED0   31  012F  2164          BRW     RCLI
                          0132  2165
                          0132  2166 ;
                          0132  2167 ; No errors anywhere!
                          0132  2168 ;
                          0132  2169
             50   04 A2   D0  0132  2170 90$:     MOVL    CLI$L_COMMAND(R2),R0    ; TO GET ADDRESS OF EXECUTION ROUTINE.
                   02   13  0136  2171          BEQL    99$                     ; SKIP CALL IF NULL
                   60   16  0138  2172          JSB     (R0)                    ; CALL FINAL ACTION ROUTINE
                FEC5   31  013A  2173 99$:     BRW     RCLI                    ; CONTINUE COMMAND SCAN
```

```
                     2175          .SBTTL  CLI Action Routines.
                     2176 ;++
                     2177 ; FUNCTIONAL DESCRIPTION:
                     2178 ;
                     2179 ;       THESE ROUTINES SAVE THE COMMAND LINE INFORMATION AS IT IS BEING PARSED.
                     2180 ;
                     2181 ; CALLING SEQUENCE:
                     2182 ;
                     2183 ;       SUBROUTINE CALL FROM THE "PARSE" SERVICE ROUTINE.
                     2184 ;
                     2185 ; INPUT PARAMETERS:
                     2186 ;
                     2187 ;       R0      = ACTION CODE FROM TREE.
                     2188 ;       R1      = DATA, IF CLI$K_NUM.
                     2189 ;       R8      = POINTER TO NEXT CHARACTER.
                     2190 ;
                     2191 ; IMPLICIT INPUTS:      NONE
                     2192 ;
                     2193 ; OUTPUT PARAMETERS:    NONE
                     2194 ;
                     2195 ; IMPLICIT OUTPUTS:     NONE
                     2196 ;
                     2197 ; COMPLETION CODES:     NONE
                     2198 ;
                     2199 ; SIDE EFFECTS:         NONE
                     2200 ;--
```

```
                        013D  2202
                        013D  2203 CLI_ACTION::
            8E'8F    00   50    8F  013D  2204          CASEB   R0,#0, #$$$
                      00000000  0142  2205 10$:     $$N=0
                        0142  2206
                        0142  2207          CASEDEF NULL,\$$N                       ; Must be first         [43]
                   0134' 0142                .WORD   ACT_NULL - 10$
                        0144  2208          CASEDEF ILL_CMD,\$$N                    ;                       [43]
                   0135' 0144                .WORD   ACT_ILL_CMD - 10$
                        0146  2209          CASEDEF INC_CMD,\$$N                    ;                       [43]
                   013B' 0146                .WORD   ACT_INC_CMD - 10$
                        0148  2210          CASEDEF CONTEXT,\$$N                    ;                       [43]
                   013F' 0148                .WORD   ACT_CONTEXT - 10$
                        014A  2211          CASEDEF SUCCESS,\$$N                    ;                       [46]
                   0154' 014A                .WORD   ACT_SUCCESS - 10$
                        014C  2212          CASEDEF FAILURE,\$$N                    ;                       [46]
                   015D' 014C                .WORD   ACT_FAILURE - 10$
                        014E  2213          CASEDEF ABORT,\$$N
                   0181' 014E                .WORD   ACT_ABORT - 10$
                        0150  2214          CASEDEF ATTACH,\$$N
                   0170' 0150                .WORD   ACT_ATTACH - 10$
                        0152  2215          CASEDEF BOOT,\$$N                       ;                       [88]
                   018A' 0152                .WORD   ACT_BOOT - 10$
                        0154  2216          CASEDEF CLEAR,\$$N
                   01BD' 0154                .WORD   ACT_CLEAR - 10$
                        0156  2217          CASEDEF CONTINUE,\$$N
                   01CA' 0156                .WORD   ACT_CONTINUE - 10$
                        0158  2218          CASEDEF DEATTACH,\$$N                   ;                       [48]
                   021A' 0158                .WORD   ACT_DEATTACH - 10$
                        015A  2219          CASEDEF DEBUG,\$$N                      ;                       [59]
                   0223' 015A                .WORD   ACT_DEBUG - 10$
                        015C  2220          CASEDEF DEBUG_SWITCH,\$$N               ;                       [62]
                   0662' 015C                .WORD   ACT_DEBUG_SWITCH - 10$
                        015E  2221          CASEDEF DEFAULT_DBG,\$$N                ;                       [59]
                   022C' 015E                .WORD   ACT_DEFAULT_DBG - 10$
                        0160  2222          CASEDEF DEPOSIT,\$$N
                   0246' 0160                .WORD   ACT_DEPOSIT - 10$
                        0162  2223          CASEDEF DIRECTORY,\$$N
                   025C' 0162                .WORD   ACT_DIRECTORY - 10$
                        0164  2224          CASEDEF DUMP,\$$N                       ;                       [97]
                   0273' 0164                .WORD   ACT_DUMP - 10$
                        0166  2225          CASEDEF EXAMINE,\$$N
                   029E' 0166                .WORD   ACT_EXAMINE - 10$
                        0168  2226          CASEDEF EXIT,\$$N                       ;                       [46]
                   02AB' 0168                .WORD   ACT_EXIT - 10$
                        016A  2227          CASEDEF HELP,\$$N
                   02B4' 016A                .WORD   ACT_HELP - 10$
                        016C  2228          CASEDEF LOAD,\$$N
                   031D' 016C                .WORD   ACT_LOAD - 10$
                        016E  2229          CASEDEF NEXT_INST,\$$N
                   0330' 016E                .WORD   ACT_NEXT_INST - 10$
                        0170  2230          CASEDEF ENUF,\$$N
                   016B' 0170                .WORD   ACT_ENUF - 10$
                        0172  2231          CASEDEF NOTNUF,\$$N
                   0166' 0172                .WORD   ACT_NOTNUF - 10$
                        0174  2232          CASEDEF RUN,\$$N
                   034D' 0174                .WORD   ACT_RUN - 10$
```

```
                     0176  2233          CASEDEF SCRIPT,\$$N
0558'                0176                .WORD   ACT_SCRIPT - 10$
                     0178  2234          CASEDEF SET,\$$N
039F'                0178                .WORD   ACT_SET - 10$
                     017A  2235          CASEDEF ULTRIX_ODS,\$$N                                 ;               [106
04E6'                017A                .WORD   ACT_ULTRIX_ODS - 10$
                     017C  2236          CASEDEF SHOW,\$$N
051B'                017C                .WORD   ACT_SHOW - 10$
                     017E  2237          CASEDEF Clear_CRD_Trace,\$$N                            ;               [65]
085B'                017E                .WORD   ACT_Clear_CRD_Trace - 10$
                     0180  2238          CASEDEF Set_CRD_Trace,\$$N                              ;               [65]
0851'                0180                .WORD   ACT_Set_CRD_Trace - 10$
                     0182  2239          CASEDEF Set_CRD_Debug,\$$N                              ;               [69]
0897'                0182                .WORD   ACT_Set_CRD_Debug - 10$
                     0184  2240          CASEDEF Clear_CRD_Debug,\$$N                            ;               [69]
08A0'                0184                .WORD   ACT_Clear_CRD_Debug - 10$
                     0186  2241          CASEDEF Show_CRD_Trace,\$$N                             ;               [65]
0865'                0186                .WORD   ACT_Show_CRD_Trace - 10$
                     0188  2242          CASEDEF Show_Calls,\$$N                                 ;               [70]
0844'                0188                .WORD   ACT_Show_Calls - 10$
                     018A  2243          CASEDEF SHOWCPU,\$$N                                    ; [116]         ;G:3
04AC'                018A                .WORD   ACT_SHOWCPU - 10$
                     018C  2244          CASEDEF START,\$$N
0520'                018C                .WORD   ACT_START - 10$
                     018E  2245          CASEDEF SUMMARY,\$$N
0561'                018E                .WORD   ACT_SUMMARY - 10$
                     0190  2246          CASEDEF CRD,\$$N                                        ;               [60]
056A'                0190                .WORD   ACT_CRD - 10$
                     0192  2247          CASEDEF FILESPEC,\$$N
0649'                0192                .WORD   ACT_FILESPEC - 10$
                     0194  2248          CASEDEF FILEND,\$$N
0651'                0194                .WORD   ACT_FILEND - 10$
                     0196  2249          CASEDEF OPTIONAL,\$$N                                   ;               [45]
0658'                0196                .WORD   ACT_OPTIONAL - 10$
                     0198  2250          CASEDEF REQUIRED,\$$N                                   ;               [45]
065D'                0198                .WORD   ACT_REQUIRED - 10$
                     019A  2251          CASEDEF TEST,\$$N
0687'                019A                .WORD   ACT_TEST - 10$
                     019C  2252          CASEDEF LAST,\$$N
068C'                019C                .WORD   ACT_LAST - 10$
                     019E  2253          CASEDEF SUBTEST,\$$N
0682'                019E                .WORD   ACT_SUBTEST - 10$
                     01A0  2254          CASEDEF MMON,\$$N
0B31'                01A0                .WORD   ACT_MMON - 10$
                     01A2  2255          CASEDEF MMOFF,\$$N
0B60'                01A2                .WORD   ACT_MMOFF - 10$
                     01A4  2256          CASEDEF SHOWMM,\$$N
0B8E'                01A4                .WORD   ACT_SHOWMM - 10$
                     01A6  2257          CASEDEF PREGN,\$$N
09EB'                01A6                .WORD   ACT_PREGN - 10$
                     01A8  2258          CASEDEF SECTION,\$$N
0678'                01A8                .WORD   ACT_SECTION - 10$
                     01AA  2259          CASEDEF PASS,\$$N
066E'                01AA                .WORD   ACT_PASS - 10$
                     01AC  2260          CASEDEF QA,\$$N                                         ;               [42]
0673'                01AC                .WORD   ACT_QA - 10$
                     01AE  2261          CaseDef SetEnforce,\$$N                                 ;               [57]
```

```
0722' 01AE                .WORD   ACT_SetEnforce - 10$
      01B0  2262          CaseDef ClearEnforce,\$$N                                       ;               [57]
0726' 01B0                .WORD   ACT_ClearEnforce - 10$
      01B2  2263          CASEDEF SHOWENFORCE,\$$N                                        ;               [89]
0734' 01B2                .WORD   ACT_SHOWENFORCE - 10$
      01B4  2264          CASEDEF SETCPU,\$$N                                             ;               [101
03AC' 01B4                .WORD   ACT_SETCPU - 10$
      01B6  2265          CASEDEF EVENT,\$$N
0765' 01B6                .WORD   ACT_EVENT - 10$
      01B8  2266          CASEDEF FLAGS,\$$N
078E' 01B8                .WORD   ACT_FLAGS - 10$
      01BA  2267          CASEDEF BASE,\$$N
07A3' 01BA                .WORD   ACT_BASE - 10$
      01BC  2268          CASEDEF ADDRESS,\$$N
07B9' 01BC                .WORD   ACT_ADDRESS - 10$
      01BE  2269          CASEDEF BREAK,\$$N
07C6' 01BE                .WORD   ACT_BREAK - 10$
      01C0  2270          CASEDEF BASE_QUAL,\$$N
07ED' 01C0                .WORD   ACT_BASE_QUAL - 10$
      01C2  2271          CASEDEF NOBASE,\$$N
07F9' 01C2                .WORD   ACT_NOBASE - 10$
      01C4  2272          CASEDEF ALL,\$$N
0804' 01C4                .WORD   ACT_ALL - 10$
      01C6  2273          CASEDEF ASK_PROMPT,\$$N                                         ;               [108
08A8' 01C6                .WORD   ACT_ASK_PROMPT - 10$
      01C8  2274          CASEDEF DEFAULT,\$$N
0825' 01C8                .WORD   ACT_DEFAULT - 10$
      01CA  2275          CASEDEF BELL,\$$N
08AE' 01CA                .WORD   ACT_BELL - 10$
      01CC  2276          CASEDEF BINARY,\$$N                                             ;               [45]
08B4' 01CC                .WORD   ACT_BINARY - 10$
      01CE  2277          CASEDEF HALT,\$$N
08BA' 01CE                .WORD   ACT_HALT - 10$
      01D0  2278          CASEDEF IE1,\$$N
08C0' 01D0                .WORD   ACT_IE1 - 10$
      01D2  2279          CASEDEF IE2,\$$N
08C6' 01D2                .WORD   ACT_IE2 - 10$
      01D4  2280          CASEDEF IE3,\$$N
08CC' 01D4                .WORD   ACT_IE3 - 10$
      01D6  2281          CASEDEF IES,\$$N
08D2' 01D6                .WORD   ACT_IES - 10$
      01D8  2282          CASEDEF LOOP,\$$N
08D8' 01D8                .WORD   ACT_LOOP - 10$
      01DA  2283          CASEDEF OPER,\$$N
08DE' 01DA                .WORD   ACT_OPER - 10$
      01DC  2284          CASEDEF PROMPT,\$$N
08E4' 01DC                .WORD   ACT_PROMPT - 10$
      01DE  2285          CASEDEF QUICK,\$$N
08EA' 01DE                .WORD   ACT_QUICK - 10$
      01E0  2286          CASEDEF SEARCH,\$$N
08F0' 01E0                .WORD   ACT_SEARCH - 10$
      01E2  2287          CASEDEF TRACE,\$$N
08F6' 01E2                .WORD   ACT_TRACE - 10$
      01E4  2288          CASEDEF VERIFY,\$$N                                             ;               [45]
08FC' 01E4                .WORD   ACT_VERIFY - 10$
      01E6  2289          CASEDEF QAEP,\$$N                                               ;               [42]
093E' 01E6                .WORD   ACT_QAEP - 10$
```

```
                01E8  2290        CASEDEF QACL,\$$N                                  ;          [42]
        0944'   01E8              .WORD   ACT_QACL - 10$
                01EA  2291        CASEDEF QATL,\$$N                                  ;          [42]
        094A'   01EA              .WORD   ACT_QATL - 10$
                01EC  2292        CASEDEF QASL,\$$N                                  ;          [42]
        0950'   01EC              .WORD   ACT_QASL - 10$
                01EE  2293        CASEDEF QAMP,\$$N                                  ;          [42]
        0956'   01EE              .WORD   ACT_QAMP - 10$
                01F0  2294        CASEDEF QADEF,\$$N                                 ;          [42]
        0978'   01F0              .WORD   ACT_QADEF - 10$
                01F2  2295        CASEDEF EFN,\$$N
        099A'   01F2              .WORD   ACT_EFN - 10$
                01F4  2296        CASEDEF NEXT,\$$N
        09BA'   01F4              .WORD   ACT_NEXT - 10$
                01F6  2297        CASEDEF PAGE,\$$N                                  ;          [71]
        0902'   01F6              .WORD   ACT_PAGE - 10$
                01F8  2298        CASEDEF WIDTH,\$$N                                 ;          [38]
        0920'   01F8              .WORD   ACT_WIDTH - 10$
                01FA  2299        CASEDEF ASCII,\$$N
        09BF'   01FA              .WORD   ACT_ASCII - 10$
                01FC  2300        CASEDEF REGN,\$$N
        09C4'   01FC              .WORD   ACT_REGN - 10$
                01FE  2301        CASEDEF REG12,\$$N
        09C9'   01FE              .WORD   ACT_REG12 - 10$
                0200  2302        CASEDEF REG13,\$$N
        09CE'   0200              .WORD   ACT_REG13 - 10$
                0202  2303        CASEDEF REG14,\$$N
        09D3'   0202              .WORD   ACT_REG14 - 10$
                0204  2304        CASEDEF REG15,\$$N
        09D8'   0204              .WORD   ACT_REG15 - 10$
                0206  2305        CASEDEF REG16,\$$N
        09DE'   0206              .WORD   ACT_REG16 - 10$
                0208  2306        CASEDEF REGP,\$$N
        09E2'   0208              .WORD   ACT_REGP - 10$
                020A  2307        CASEDEF BACKUP,\$$N
        09F2'   020A              .WORD   ACT_BACKUP - 10$
                020C  2308        CASEDEF ADVANCE,\$$N                               ;          [45]
        09F7'   020C              .WORD   ACT_ADVANCE - 10$
                020E  2309        CASEDEF DATA,\$$N
        09FC'   020E              .WORD   ACT_DATA - 10$
                0210  2310        CASEDEF NULL_DATA,\$$N                             ;          [97]
        0A01'   0210              .WORD   ACT_NULL_DATA - 10$
                0212  2311        CASEDEF LONG,\$$N
        0A0A'   0212              .WORD   ACT_LONG - 10$
                0214  2312        CASEDEF WORD,\$$N
        0A0F'   0214              .WORD   ACT_WORD - 10$
                0216  2313        CASEDEF BYTE,\$$N
        0A14'   0216              .WORD   ACT_BYTE - 10$
                0218  2314        CASEDEF HEX,\$$N
        0A19'   0218              .WORD   ACT_HEX - 10$
                021A  2315        CASEDEF DEC,\$$N
        0A1E'   021A              .WORD   ACT_DEC - 10$
                021C  2316        CASEDEF OCT,\$$N
        0A23'   021C              .WORD   ACT_OCT - 10$
                021E  2317        CASEDEF IF_RUN,\$$N
        0A28'   021E              .WORD   ACT_IF_RUN - 10$
                0220  2318        CASEDEF IF_REQUIRED,\$$N                           ;          [45]
```

```
                0A36' 0220                .WORD   ACT_IF_REQUIRED - 10$
                      0222  2319          CASEDEF IF_FILE_SPEC,\$$N                       ;               [45]
                0A3F' 0222                .WORD   ACT_IF_FILE_SPEC - 10$
                      0224  2320          CASEDEF IF_ADAPTER,\$$N                         ;               [48]
                0A49' 0224                .WORD   ACT_IF_ADAPTER - 10$
                      0226  2321          CASEDEF IF_REG_OR_ADR,\$$N                      ;               [43]
                0A56' 0226                .WORD   ACT_IF_REG_OR_ADR - 10$
                      0228  2322          CASEDEF IFNOTUSER,\$$N                          ;               [38]
                0A64' 0228                .WORD   ACT_IFNOTUSER - 10$
                      022A  2323          CASEDEF IF_XDELTA,\$$N
                0A71' 022A                .WORD   ACT_IF_XDELTA - 10$
                      022C  2324          CASEDEF XDELTA,\$$N
                0A86' 022C                .WORD   ACT_XDELTA - 10$
                      022E  2325          CASEDEF DEVICE,\$$N
                0A90' 022E                .WORD   ACT_DEVICE - 10$
                      0230  2326          CASEDEF BRIEF,\$$N                              ;               [44]
                0AA6' 0230                .WORD   ACT_BRIEF - 10$
                      0232  2327          CASEDEF BEGINADAPTER,\$$N                       ;               [43]
                0ADB' 0232                .WORD   ACT_BEGINADAPTER - 10$
                      0234  2328          CASEDEF ADAPTER,\$$N                            ;               [43]
                0AE9' 0234                .WORD   ACT_ADAPTER - 10$
                      0236  2329          CASEDEF DESELECT,\$$N
                0253' 0236                .WORD   ACT_DESELECT - 10$
                      0238  2330          CASEDEF SELECT,\$$N
                0385' 0238                .WORD   ACT_SELECT - 10$
                      023A  2331          CASEDEF NO_ALLOC,\$$N                           ;               [82]
                0396' 023A                .WORD   ACT_NO_ALLOC - 10$
                      023C  2332          CASEDEF SHOWSEL,\$$N
                0B00' 023C                .WORD   ACT_SHOWSEL - 10$
                      023E  2333          CASEDEF DO_CMD,\$$N
                0B16' 023E                .WORD   ACT_DO_CMD - 10$
                      0240  2334          CASEDEF SETLOAD,\$$N
                0B1F' 0240                .WORD   ACT_SETLOAD - 10$
                      0242  2335          CASEDEF SHOWLOAD,\$$N
                0B28' 0242                .WORD   ACT_SHOWLOAD - 10$
                      0244  2336          CASEDEF SHOWSUP,\$$N                            ;               [44]
                0BF8' 0244                .WORD   ACT_SHOWSUP - 10$
                      0246  2337          CASEDEF SHOWSTATUS,\$$N                         ;               [39]
                0BEF' 0246                .WORD   ACT_SHOWSTATUS - 10$
                      0248  2338          CASEDEF SHOWSECTIONS,\$$N                       ;               [46]
                0C01' 0248                .WORD   ACT_SHOWSECTIONS - 10$
                      024A  2339          CASEDEF SHOWTESTS,\$$N                          ;           [81]
                0C0A' 024A                .WORD   ACT_SHOWTESTS - 10$
                      024C  2340          CaseDef ShowMem,\$$N
                0C13' 024C                .WORD   ACT_ShowMem - 10$
                      024E  2341          CaseDef ShoMemMap,\$$N
                0C1C' 024E                .WORD   ACT_ShoMemMap - 10$
                      0250  2342          CaseDef ShoMemBuf,\$$N
                0C20' 0250                .WORD   ACT_ShoMemBuf - 10$
                      0252  2343          CaseDef ShoMemDat,\$$N
                0C25' 0252                .WORD   ACT_ShoMemDat - 10$
                      0254  2344          CaseDef ShoMemAll,\$$N
                0C2A' 0254                .WORD   ACT_ShoMemAll - 10$
                      0256  2345          CASEDEF SETMEM,\$$N                             ;               [75]
                0C36' 0256                .WORD   ACT_SETMEM - 10$
                      0258  2346          CASEDEF INITPCS,\$$N                            ;               [77]
                02F5' 0258                .WORD   ACT_INITPCS - 10$
```

```
                          025A  2347          CASEDEF DIR_WIDE,\$$N                                  ;                [80]
                     026B' 025A                .WORD   ACT_DIR_WIDE - 10$
                          025C  2348          CASEDEF BEGINTIME,\$$N                                 ;        [76]
                     0691' 025C                .WORD   ACT_BEGINTIME - 10$
                          025E  2349          CASEDEF TIME,\$$N                                      ;        [76]
                     069B' 025E                .WORD   ACT_TIME - 10$
                          0260  2350
                 0000008E 0260  2351          $$$=$$N-1
                          0260  2352
                          0260  2353 20$:     ERRSUP_S  MSGADR=L^T_PRGERR     ;*** INVALID ACTION CODE????          [41]
        00000000'EF   DF  0260                        PUSHAL  $MODULE
        00000004'EF   DF  0266                        PUSHAL  L^T_PRGERR
                 01   DD  026C                        PUSHL   #$ER
   00000000'EF   03   FB  026E                        CALLS   #3, DS_ERRSUP
                      05  0275  2354          RSB
```

```
                                  0276          .SUBTITLE       Null and Context-Saving Action Routines
                                  0276  ;+
                                  0276  ; NULL ACTION ROUTINE.
                                  0276  ; This action routine must remain here. It is the action routine that is
                                  0276  ; done when the parser executes the 0 action routine.
                                  0276  ;-
                                  0276  ACT_NULL:
                            05    0276          RSB                                 ; RETURN
                                  0277
                                  0277  ;+
                                  0277  ; ILL_CMD Action routine.
                                  0277  ;-
                                  0277  ACT_ILL_CMD:                                ;                                [43]
          06 62    01      E5    0277          BBCC    #CLI$V_NOTNUF, -            ;                                [43]
                                  027B                    CLI$L_FLAGS (R2), ACT_CONTEXT ;                              [43]
                    04      11    027B  10$:    BRB     ACT_CONTEXT                 ;                                [43]
                                  027D
                                  027D  ;+
                                  027D  ; INC_CMD Action routine.
                                  027D  ;-
                                  027D  ACT_INC_CMD:                                ;                                [43]
          00 62    01      E3    027D          BBCS    #CLI$V_NOTNUF, -            ;                                [43]
                                  0281                    CLI$L_FLAGS (R2), ACT_CONTEXT ;                              [43]
                                  0281
                                  0281  ;+
                                  0281  ; CONTEXT Action routine.
                                  0281  ; Branch to when error. Save descriptor of good portion of command line.
                                  0281  ;-
                                  0281
                                  0281  ACT_CONTEXT:                                ;                                [43]
  00000450'EF    38 A2      D0    0281          MOVL    CLI$Q_BUFQWD+4(R2), -       ; Address of good portion        [43]
                                  0289                    Q_GOOD_CMD+4                ;                                [43]
     58    00000450'EF      C3    0289          SUBL3   Q_GOOD_CMD+4, -             ;                                [43]
           0000044C'EF            0290                    R8, Q_GOOD_CMD              ; Length                         [43]
                            05    0295          RSB                                 ; Return                         [43]
```

```
                                0296  2392          .SUBTITLE       Success, Failure, Notnuf, Enuf Action Routines
                                0296  2393 ;+
                                0296  2394 ; SUCCESS Action routine.
                                0296  2395 ;-
                                0296  2396 ACT_SUCCESS:
      00 00000454'EF    00  E3  0296  2397          BBCS    #0, B_SUCCESS, 10$      ; Indicate SUCCESS                      [46]
                            05  029E  2398 10$:     RSB
                                029F  2399
                                029F  2400 ;+
                                029F  2401 ; FAILURE Action routine.
                                029F  2402 ;-
                                029F  2403 ACT_FAILURE:
      00 00000454'EF    00  E5  029F  2404          BBCC    #0, B_SUCCESS, 10$      ; Indicate FAILURE                      [46]
                            05  02A7  2405 10$:     RSB
                                02A8  2406
                                02A8  2407 ;+
                                02A8  2408 ; NOTNUF ACTION ROUTINE.
                                02A8  2409 ;-
                                02A8  2410 ACT_NOTNUF:
            00 62   01  E3  02A8  2411          BBCS    #CLI$V_NOTNUF, -        ; SET NOT ENOUGH FLAG.
                                02AC  2412                    CLI$L_FLAGS(R2), 10$
                            05  02AC  2413 10$:     RSB                             ; RETURN
                                02AD  2414
                                02AD  2415 ;+
                                02AD  2416 ; ENUF Action routine.
                                02AD  2417 ;-
                                02AD  2418 ACT_ENUF:
   00 62    01  E4  02AD  2419          BBSC    #CLI$V_NOTNUF, -        ; INDICATE SUFFICIENT INFO.
                                02B1  2420                    CLI$L_FLAGS(R2), 10$
                            05  02B1  2421 10$:     RSB                             ; RETURN.
```

```
                                02B2          .SUBTITLE       Commands A-C Action Routines
                                02B2  ;+
                                02B2  ; ATTACH COMMAND.
                                02B2  ;-
                                02B2  ACT_ATTACH:
          08 A2    59    D0     02B2          MOVL    R9,CLI$Q_FILE(R2)        ; SAVE LENGTH OF STRING
          0C A2    58    D0     02B6          MOVL    R8,CLI$Q_FILE+4(R2)      ; SAVE ADDRESS OF REMAINDER
  04 A2   00000000'EF    9E     02BA          MOVAB   L^DSV$ATTACH,CLI$L_COMMAND(R2) ; SET COMMAND PROCESSOR          [41]
                         05     02C2          RSB
                                02C3
                                02C3  ;+
                                02C3  ; ABORT COMMAND.
                                02C3  ;-
                                02C3  ACT_ABORT:
  04 A2   00000000'EF    DE     02C3          MOVAL   L^VRABORT, -             ; LOAD COMMAND ROUTINE ADDRESS.         [41]
                                02CB                    CLI$L_COMMAND(R2)
                         05     02CB          RSB
                                02CC
                                02CC  ;+
                                02CC  ; BOOT COMMAND
                                02CC  ;-
                                02CC  ACT_BOOT:
      04 A2   D2'AF      DE     02CC          MOVAL   B^10$,CLI$L_COMMAND(R2)          ; Load command routine address  [88]
                         05     02D1          RSB                                      ;                               [88]
                                02D2
                                02D2  10$:    BR_IF_NOT_USER  20$                      ; If standalone, continue       [92]
   0000FE00'EF    1C    E1     02D2          BBC     #DSA$V_User, -                   ; If bit not set, branch
                  12            02D9                    L^DSA$GL_Flags, 20$              ;
        000000BD'EF      9F     02DA          PUSHAB  L^T_EMESSG1                      ;                               [92]
                  01    DD     02E0          pushl   #ds$k_printf                     ; Print with PRINTF             [92]
                  15    DD     02E2          pushl   #ds$k_type_command_err           ; Command error code            [92]
   00000000'GF    03    FB     02E4          CALLS   #3, G^DSX$PRINT                  ; Print                         [92]
                         05     02EB          RSB                                      ;                               [92]
                                02EC
                                02EC  ; In order to insure the state of the PP upon stopping the AP, we will do an    [92]
                                02EC  ; INIT_CONTEXT and a BRW BEGIN. Therefore, we will not return here after the    [92]
                                02EC  ; CMK call. We will return here is we are not on an AP system.                  [92]
                                02EC
                                02EC  20$:    CMK     APBOOT                           ; Go do boot.                   [92]
                  7E    DC     02EC          MOVPSL  -(SP)
          06 6E   1A    E0     02EE          BBS     #PSL$V_IS, (SP), 30602$
                  8E    D4     02F2          CLRL    (SP)+
                  04    BC     02F4          CHMK    #CMK$_APBOOT
                  06    11     02F6          BRB     30603$
        00000000'EF      16     02F8  30602$: JSB     CMK$APBOOT
                                02FE  30603$:
                         05     02FE          RSB                                      ;                               [92]
                                02FF
                                02FF  ;+
                                02FF  ; CLEAR COMMAND.
                                02FF  ;-
                                02FF  ACT_CLEAR:
  04 A2   00000000'EF    DE     02FF          MOVAL   L^VRCLRFLG, -            ; LOAD COMMAND ROUTINE ADDRESS.         [41]
                                0307                    CLI$L_COMMAND(R2)
          00 62    02    E2     0307          BBSS    #CLI$V_CLEAR, -          ; SET CLR FLG.
                                030B                    CLI$L_FLAGS(R2), 10$
                         05     030B  10$:    RSB                              ; RETURN.
```

```
                          030C  2471
                          030C  2472 ;+
                          030C  2473 ; CONTINUE COMMAND.
                          030C  2474 ;-
                          030C  2475 ACT_CONTINUE:
00000000'EF   02    8A    030C  2476          BICB2   #2,L^DEBUG$GB_FLAGS     ; CLEAR STEP FLAG                          [41]
     00000000'EF    D4    0313  2477          CLRL    MP$GL_CONDITION_FLAGS   ; Clear mp condition flags                 [103
04 A2 00000322'EF   9E    0319  2478          MOVAB   10$,CLI$L_COMMAND(R2)   ; SET COMMAND ADDRESS
                       05    0321  2479          RSB
                          0322  2480 10$:
     00000000'EF    D5    0322  2481          TSTL MP$GR_BOOTED_PROCESSORS    ; Are we an active mp system? [111]
                 15    13    0328  2482          BEQL    20$                     ; Branch if not active mp     [111]
                          032A  2483          $PRINT  #ds$k_type_command_out,-; Print command output        [111]
                          032A  2484                    #ds$k_printi, -       ; .. use PRINTI               [111]
                          032A  2485                    L^T_MP_CONT           ; .. Continue message         [111]
     000000AB'EF    9F    032A                        PUSHAB  L^T_MP_CONT
         7E    00    9A    0330                        movzbl  #ds$k_printi, -(sp)
         7E    16    98    0333                        cvtbl   #ds$k_type_command_out, -(sp)
00000000'GF    03    FB    0336                        CALLS   #$$N+2, G^DSX$PRINT
                 16    11    033D  2486          BRB     30$                     ;...                          [111]
                          033F  2487 20$:     $PRINT  #ds$k_type_command_out,-; Print command output      [48][111]
                          033F  2488                    #ds$k_printf, -       ; .. use PRINTF                [48]
                          033F  2489                    L^T_CONT, -           ; .. Continue message          [48]
                          033F  2490                    60(AP)                ; .. Type PC                   [48]
            3C  AC    DD    033F                        PUSHL   60(AP)
     00000092'EF    9F    0342                        PUSHAB  L^T_CONT
         7E    01    9A    0348                        movzbl  #ds$k_printf, -(sp)
         7E    16    98    034B                        cvtbl   #ds$k_type_command_out, -(sp)
00000000'GF    04    FB    034E                        CALLS   #$$N+2, G^DSX$PRINT
     00000000'EF    16    0355  2491 30$:     JSB     SCRIPT$CONT             ; Continue script if present  [111]
                          035B  2492
                          035B  2493 ACT_CONT::
                       04 035B  2494 10$:     RET                             ; RETURN TO CLI'S CALLER
```

```
                                035C          .SUBTITLE       Commands D-E Action Routines
                                035C  ;+
                                035C  ; DEATTACH command
                                035C  ;-
                                035C  act_deattach:                                  ;                                        [48]
     04 A2   00000000'EF   DE   035C          moval   L^dsv$deattach, -               ; Load command routine address           [48]
                                0364                    cli$l_command(r2)             ;                                        [48]
                           05   0364          rsb                                     ; Just return                            [48]
                                0365
                                0365  ;+                                                                                        [59]
                                0365  ; DEBUG COMMAND                                                                           [59]
                                0365  ;-                                                                                        [59]
                                0365
                                0365  ACT_DEBUG:                                                ;                               [59]
     04 A2   00000000'EF   DE   0365          MOVAL   L^DSV$DEBUG,CLI$L_COMMAND(R2)           ; LOAD ADDR. OF DEBUG COMMAND   [59]
                           05   036D          RSB                                             ; RETURN                        [59]
                                036E
                                036E  ACT_DEFAULT_DBG::                                ; BUILD FILENAME DESC. FOR DEFAULT DBG  [59]
     08 A2   0000037F'EF   9A   036E          MOVZBL  DBG_NAME,CLI$Q_FILE(R2)         ; GET CHARACTER COUNT                   [59]
     0C A2   00000380'EF   DE   0376          MOVAL   DBG_NAME+1, -                   ; GET ADDRESS OF FILENAME STRING        [59]
                                037E                          CLI$Q_FILE+4(R2);                                                 [59]
                           05   037E          RSB                                     ; RETURN                                [59]
  47 55 42 45 44 53 44 56 00'   037F  DBG_NAME:       .ASCIC /VDSDEBUG/               ; DEFAULT DEBUGGER NAME                 [68]
                           08   037F
                                0388
                                0388  ;+
                                0388  ; DEPOSIT COMMAND.
                                0388  ;-
                                0388  ACT_DEPOSIT:
     04 A2   00000000'EF   DE   0388          MOVAL   L^VRDEPOSIT, -                  ; LOAD COMMAND ROUTINE ADDRESS.         [41]
                                0390                    CLI$L_COMMAND(R2)
              00 62   19   E2   0390          BBSS    #CLI$V_DEPOSIT, -               ; SET DEPOSIT FLG.
                                0394                    CLI$L_FLAGS(R2), 10$
                           05   0394  10$:    RSB
                                0395
                                0395  ;+
                                0395  ; DESELECT
                                0395  ;-
                                0395  ACT_DESELECT:
     04 A2   00000000'EF   9E   0395          MOVAB   L^DSV$DESELECT,CLI$L_COMMAND(R2) ; SET COMMAND ROUTINE                  [41]
                           05   039D          RSB
                                039E
                                039E  ;
                                039E  ; Directory command
                                039E  ;-
                                039E
                                039E  ACT_DIRECTORY:
          00000000'EF      DE   039E          MOVAL   L^DSV$DIRECTORY,-               ; Set dispatch address for command
                04 A2           03A4                    CLI$L_COMMAND(R2)
          00000000'EF      94   03A6          CLRB    L^WIDE_FLAG                     ; WIDE_FLAG in DIRECTORY module         [80]
                           05   03AC          RSB
                                03AD
                                03AD  ACT_DIR_WIDE:                                  ;                                        [80]
     00000000'EF   01      90   03AD          MOVB    #1,L^WIDE_FLAG                  ; WIDE_FLAG in DIRECTORY module         [80]
                           05   03B4          RSB                                     ;                                        [80]
                                03B5
                                03B5  ;+
```

```
                        03B5  2552 ; DUMP COMMAND
                        03B5  2553 ;-
                        03B5  2554
                        03B5  2555 ACT_DUMP:
    04 A2   BB'AF   DE  03B5  2556          MOVAL   B^10$,CLI$L_COMMAND(R2)         ; Load command routine address  [97]
                    05  03BA  2557          RSB                                     ;                               [97]
                        03BB  2558
                        03BB  2559 10$:     BR_IF_NI        20$                     ; If network is in use, continue[97]
00000000'EF   1A    E0  03BB                BBS     #DS$V_NI, -                     ; If bit set, branch            [49]
                12      03C2                  L^DS$GL_Flags, 20$                    ;                               [49]
     000000E2'EF    9F  03C3  2560          PUSHAB  L^T_EMESSG2                     ;                               [97]
                01  DD  03C9  2561          pushl   #ds$k_printf                    ; Print with PRINTF             [97]
                15  DD  03CB  2562          pushl   #ds$k_type_command_err          ; Command error code            [97]
00000000'GF     03  FB  03CD  2563          CALLS   #3, G^DSX$PRINT                 ; Print                         [97]
                05  03D4  2564          RSB                                     ;                               [97]
                        03D5  2565
              1C A2 DD  03D5  2566 20$:     PUSHL   CLI$L_DATA(R2)                  ; Push dump size                [97]
00000000'EF     01  FB  03D8  2567          CALLS   #1,DUMP_SELF
                05  03DF  2568          RSB                                     ;                               [97]
                        03E0  2569
                        03E0  2570 ;+
                        03E0  2571 ; EXAMINE COMMAND.
                        03E0  2572 ;-
                        03E0  2573 ACT_EXAMINE:
  04 A2   00000000'EF   DE  03E0  2574          MOVAL   L^VREXAMINE, -          ; LOAD COMMAND ROUTINE ADDRESS. [41]
                        03E8  2575                    CLI$L_COMMAND(R2)
     00 62      05    E2  03E8  2576          BBSS    #CLI$V_EXAM, -          ; CHECK FOR EXAMINE.
                        03EC  2577                    CLI$L_FLAGS(R2), 10$
                    05  03EC  2578 10$:     RSB
                        03ED  2579
                        03ED  2580 ;+
                        03ED  2581 ; EXIT Action routine.
                        03ED  2582 ;-
                        03ED  2583 ACT_EXIT:
  04 A2   00000000'EF   DE  03ED  2584          MOVAL   L^DSV$EXIT, -           ; Load EXIT routine address     [46]
                        03F5  2585                    CLI$L_COMMAND (R2)        ;                               [46]
                    05  03F5  2586          RSB                                 ; Return                        [46]
```

```
                                03F6  2588          .SUBTITLE       Commands H-R Action Routines
                                03F6  2589 ;+
                                03F6  2590 ; HELP Command action routine
                                03F6  2591 ;-
                                03F6  2592
                                03F6  2593 ACT_HELP:
        00000000'EF    16       03F6  2594          JSB     SCRIPT$FLUSH            ; Flush scripts if console command
        00000000'EF    16       03FC  2595          JSB     DS_CLEANUP              ; Clean up program if necessary
     50 00000000'EF    9E       0402  2596          MOVAB   HELPINFO,R0             ; Place to save pointers
         04 A0   58    D0       0409  2597          MOVL    R8,4(R0)                ; Save address
            60   59    D0       040D  2598          MOVL    R9,(R0)                 ; Save length
        00000000'EF    16       0410  2599          JSB     INIT_CONTEXT            ; Re-init stacks, PCB, ...
     7E 00000000'EF    7D       0416  2600          MOVQ    HELPINFO,-(SP)          ; Restore descriptor
                 6E    7F       041D  2601          PUSHAQ  (SP)                    ; and set pointer to it
     00000000'EF    01    FB    041F  2602          CALLS   #1,L^DSX$HELP           ; Call the HELP routine
                 8E    7C       0426  2603          CLRQ    (SP)+                   ; Remove descriptor
              06 50    E8       0428  2604          BLBS    R0,20$                  ; If no HELP error, continue
        00000000'EF    16       042B  2605          JSB     DSR$COMPLETION          ; Type error text
        00000000'EF    17       0431  2606 20$:     JMP     BEGIN                   ; Can't return--so start over
                                0437  2607
                                0437  2608 ;+
                                0437  2609 ; INITPCS                                                          begin   [77]
                                0437  2610 ;-
                                0437  2611
                                0437  2612 ACT_INITPCS:
        04 A2    3D'AF    9E    0437  2613          MOVAB   B^10$,CLI$L_COMMAND(R2) ; Load command routine address
                          05    043C  2614          RSB
                                043D  2615
                                043D  2616 10$:     BR_IF_NOT_USER  20$
     0000FE00'EF    1C    E1    043D                BBC     #DSA$V_User, -          ; If bit not set, branch
                          12    0444                  L^DSA$GL_Flags, 20$           ;
          000000BD'EF    9F    0445  2617          PUSHAB  L^T_EMESSG1             ;
                          01    DD    044B  2618          pushl   #ds$k_printf            ; Print with PRINTF
                          15    DD    044D  2619          pushl   #ds$k_type_command_err  ; Command error code
     00000000'GF    03    FB    044F  2620          CALLS   #3, G^DSX$PRINT         ; Print
                          05    0456  2621          RSB
                                0457  2622
     00000000'EF    00    FB    0457  2623 20$:     CALLS   #0,DSV$INITPCS          ; Routine to do the work
                          05    045E  2624          RSB
                                045F  2625
                                045F  2626 ;+
                                045F  2627 ; LOAD COMMAND.
                                045F  2628 ;-
                                045F  2629 ACT_LOAD:
  04 A2    00000000'EF    DE    045F  2630          MOVAL   L^DSV$LOAD, -           ; Load command routine address
                                0467  2631            CLI$L_COMMAND(R2)
           62    40 8F    88    0467  2632          BISB2   #1@CLI$V_LOAD, -
                                046B  2633            CLI$L_FLAGS(R2)               ; SET LOAD COMMAND
     18 A2    0000'8F    3C    046B  2634          MOVZWL  #DS$A_PRGBGN,CLI$L_ADDRESS(R2)  ; DEFAULT LOAD ADDRESS
                          05    0471  2635          RSB
                                0472  2636
                                0472  2637 ;+
                                0472  2638 ; NEXT Action routine.
                                0472  2639 ;-
                                0472  2640 ACT_NEXT_INST:
        00000000'EF    D4    0472  2641          CLRL    MP$GL_CONDITION_FLAGS   ; Clear mp condition flags              [103
         1C A2   01    D0    0478  2642          MOVL    #1, CLI$L_DATA(R2)      ; DEFAULT IS ONE STEP
```

```
                0444 C2    02   88  047C  2643          BISB2   #1aCLI$V_NEXT, -          ; Set next flag                          [99]
                                     0481  2644                  CLI$L_FLAGS2(R2)          ;                                        [99]
            00000000'EF    02   88  0481  2645          BISB2   #2,L^DEBUG$GB_FLAGS       ; SET STEP FLAG                          [41]
             04 A2   FECF CF    9E  0488  2646          MOVAB   W^ACT_CONT,CLI$L_COMMAND(R2)    ; SET PROCESSING ADDRESS
                                 05  048E  2647          RSB
                                     048F  2648
                                     048F  2649 ;+
                                     048F  2650 ; RUN COMMAND.
                                     048F  2651 ;-
                                     048F  2652 ACT_RUN:                                                                        ;G:27
                00000000'EF    D4  048F  2653          CLRL    MP$GL_CONDITION_FLAGS     ; Clear mp condition flags      [103] ;G:11
                00000000'EF    7C  0495  2654          CLRQ    L^BIN_TIME                ; Init time to run.             [120] ;G:11
   00000000'EF   02000000 8F    CA  049B  2655          BICL2   #DS$M_TIMED,DS$GL_FLAGS   ; Init /TIME flag               [120] ;G:11
                0444 C2    08   C8  04A6  2656          BISL2   #CLI$M_START_OR_RUN, -    ; Indicate a RUN command        [120] ;G:11
                                     04AB  2657                  CLI$L_FLAGS2(R2)          ;                               [120] ;G:11
                  00 62    15   E3  04AB  2658          BBCS    #CLI$V_RUN, -
                                     04AF  2659                    CLI$L_FLAGS (R2), 5$    ; Set RUN flag                  [45]  ;G:11
         04 A2   00000000'EF    DE  04AF  2660 5$:      MOVAL   L^DSV$RUN, -              ; LOAD COMMAND ROUTINE ADDRESS. [41]  ;G:11
                                     04B7  2661                    CLI$L_COMMAND(R2)
                  2C A2    01   D0  04B7  2662          MOVL    #1,CLI$L_PASS(R2)         ; Default pass count is 1
         03 0000FE00'EF    0E   E1  04BB  2663          BBC     #DSA$V_SEARCH,L^DSA$GL_FLAGS,10$ ; Branch if search not set[41] ;G:1
                       2C A2    D4  04C3  2664          CLRL    CLI$L_PASS(R2)            ; If in SEARCH, default to 0.
                                 05  04C6  2665 10$:     RSB                               ; Return
```

```
                                04C7          .SUBTITLE       Commands S-Z Action Routines
                                04C7
                                04C7  ;+
                                04C7  ; SELECT
                                04C7  ;-
                                04C7  ACT_SELECT:
     04 A2   00000000'EF    9E  04C7          MOVAB   L^DSV$SELECT,CLI$L_COMMAND(R2)  ; SET COMMAND ROUTINE          [41]
     00 00000444 E2   00    E5  04CF          BBCC    #CLI$V_NOALLOC, -               ; CLEAR NO ALLOCATE FLAG       [91]
                                04D7                    L^CLI$L_FLAGS2(R2),10$        ;                              [91]
                          05    04D7  10$:    RSB
                                04D8  ;+
                                04D8  ; SELECT/NOALLOCATE                                                            [82]
                                04D8  ;-
                                04D8  ACT_NO_ALLOC::
     00 00000444 E2   00    E2  04D8          BBSS    #CLI$V_NOALLOC, -               ; SET NO ALLOCATE FLAG         [82]
                                04E0                    L^CLI$L_FLAGS2(R2),10$        ;                              [82]
                          05    04E0  10$:    RSB                                     ;                              [82]
                                04E1  ;
                                04E1  ;+
                                04E1  ; SET COMMAND.
                                04E1  ;-
                                04E1  ACT_SET:
     04 A2   00000000'EF    DE  04E1          MOVAL   L^VRSETFLG, -           ; LOAD COMMAND ROUTINE ADDRESS.        [41]
                                04E9                    CLI$L_COMMAND(R2)
              00 62   03    E2  04E9          BBSS    #CLI$V_SET, -           ; SET SET FLG.
                                04ED                    CLI$L_FLAGS(R2), 10$
                          05    04ED  10$:    RSB                             ; RETURN.
                                04EE
                                04EE  ;                                                                              [101
                                04EE  ;+                                                                             [101
                                04EE  ; SET CPU COMMAND                                                              [101
                                04EE  ;-                                                                             [101
                                04EE  ;                                                                              [101
                                04EE  ACT_SETCPU:
                                04EE
                 00FF 8F    BB  04EE          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7>    ; SAVE REGISTERS        [101]
                                04F2
                                04F2  5$:     Br_If_Not_User  10$             ; Branch around if not user mode       [101
        0000FE00'EF   1C    E1  04F2          BBC     #DSA$V_User, -                  ; If bit not set, branch
                      14        04F9                    L^DSA$GL_Flags, 10$           ;
            000000BD'EF    9F  04FA          PUSHAB  L^T_EMESSG1                     ;                              [101
                      01    DD  0500          pushl   #ds$k_printf                    ; Print with PRINTF            [101
                      15    DD  0502          pushl   #ds$k_type_command_err          ; Command error code           [101
        00000000'GF   03    FB  0504          CALLS   #3, G^DSX$PRINT                 ; Print                        [101
                    00DB    31  050B          BRW     500$                            ; Return                       [101
                                050E
                 56   52    D0  050E  10$:    MOVL    R2,R6                   ; PUT BASE OF CLI IN R6        [101]
                                0511
                                0511  ;       1) see if name stored in cli$Q_file is 'primary'                [101]
                                0511
              55   08 A6    9A  0511          MOVZBL  CLI$Q_FILE(R6),R5               ; Store string length [111]
  00000455'EF   0C B6   55  29  0515          CMPC3   R5,@CLI$Q_FILE+4(r6),PRIMARY_NAME       ;       [101][111]
                                051E
                                051E  ;       2) if so clear ap_set_flag, exit                                [101]
                                051E                                          ; else
                      03    12  051E                  BNEQ    90$             ; BRANCH to find Ptable of device      [101
                                0520
```

```
                     00B4   31  0520  2722                BRW     400$            ; JUMP to clear flag and exit           [101
                                0523  2723
                                0523  2724 ;       3) else search for LUN                                                  [101
                                0523  2725
                00000473'EF    D4  0523  2726 90$:    CLRL    MP$GL_LUN       ; Reset the log. unit number            [101]
                                0529  2727
                                0529  2728 100$:    $DS_GPHARD_S    MP$GL_LUN, - ; Get the Ptable for the           [101]
                                0529  2729                            PTABLE      ; specified l.u.n.                  [101]
                0000046E'EF    DF  0529                        PUSHAL  PTABLE
                00000473'EF    DD  052F                        PUSHL   MP$GL_LUN
             00000000'9F   02   FB  0535                        CALLS   #2, a#DS$GPHARD
                      01   50   D1  053C  2730         CMPL    R0,#SS$_NORMAL  ; If no more Ptables exist then         [101]
                           7F   12  053F  2731         BNEQ    300$            ; branch to print error and exit        [101]
                                0541  2732
                0000046E'EF    D0  0541  2733         MOVL    PTABLE, R7      ; Else, move the Ptable address in R7   [101]
                55   08 A6     9A  0548  2734         MOVZBL  CLI$Q_FILE(R6),R5               ; Store string length [111]
          0D A7   0C B6   55    29  054C  2735         CMPC3   R5, -           ; and compare it to the generic name[101][111]
                                0552  2736                         aCLI$Q_FILE+4(r6),HP$T_DEVICE+1(R7) ; [101]
                                0552  2737
                           08   13  0552  2738         BEQL    200$            ; If found then check if valid          [101]
                                0554  2739
                00000473'EF    D6  0554  2740         INCL    MP$GL_LUN       ; Try the next log. unit number         [101]
                           CD   11  055A  2741         BRB     100$            ;                                       [101]
                                055C  2742
                                055C  2743 ;       4) if Ptable exist see if device type is a cpu. If so, set flag and exit
                                055C  2744
                                055C  2745
                                055C  2746 200$:
                55   26 A7     9A  055C  2747         MOVZBL  HP$T_TYPE(R7),R5                ; Store string length [111][123] ;G:
0000045D'EF     27 A7   55    29  0560  2748         CMPC3   R5, -   ; and compare it to the device type  [101][111][123]  ;G:26
                                0569  2749                         HP$T_TYPE+1(r7),KA884_CPU+1     ;          [101][118][123]  ;G:28
                                0569  2750                                                                             ;G:26
                           43   13  0569  2751         BEQL    250$                    ; branch if device type is valid[101] ;G:26
                                056B  2752                                                                             ;G:26
                55   26 A7     9A  056B  2753         MOVZBL  HP$T_TYPE(R7),R5                ; Store string length [111]
00000463'EF     27 A7   55    29  056F  2754         CMPC3   R5, -   ; and compare it to the device type  [101][111]
                                0578  2755                         HP$T_TYPE+1(r7),KA880_CPU+1     ;          [101][118]       ;G:5
                                0578  2756
                           34   13  0578  2757         BEQL    250$                    ; branch if device type is valid[101]
                                057A  2758
                55   26 A7     9A  057A  2759         MOVZBL  HP$T_TYPE(R7),R5                ; Store string length [111]
00000469'EF     27 A7   55    29  057E  2760         CMPC3   R5, -   ; and compare it to the device type    [101][111]
                                0587  2761                         HP$T_TYPE+1(r7),KA820_CPU+1     ;          [101][111]
                                0587  2762
                           25   13  0587  2763         BEQL    250$                    ; branch if device type is valid[101]
                                0589  2764                                             ; else ...                      [101]
                                0589  2765
                0000045C'EF    9F  0589  2766         PUSHAB  KA884_CPU               ; Push device type      [123]       ;G:28
                00000462'EF    9F  058F  2767         PUSHAB  KA880_CPU               ; Push device type      [101][118]  ;G:5
                00000468'EF    9F  0595  2768         PUSHAB  KA820_CPU               ; Push device type      [101][111]
                00000181'EF    9F  059B  2769         PUSHAB  L^T_EMESSG4             ; Error in device type          [101]
                           01   DD  05A1  2770         pushl   #ds$k_printf            ; Print with PRINTF             [101]
                           15   DD  05A3  2771         pushl   #ds$k_type_command_err  ; Command error code           [101]
                00000000'GF    05  FB  05A5  2772         CALLS   #5, G^DSX$PRINT         ; Print                         [101]
                                05AC  2773
                           29   11  05AC  2774         BRB     400$
                                05AE  2775
```

```
                                05AE  2776 250$:
                 0D A7   08   28  05AE  2777          MOVC3   #8,HP$T_DEVICE+1(R7),- ; SAVE device name if we need    [118] ;G:5
              00000478'EF        05B2  2778                      MP$GT_DEVICE_NAME+1  ; to print it later            [101] ;G:5
          00000472'EF   01   90  05B7  2779          MOVB    #1,MP$GB_SETCPU ; Set flag and                          [101]
                        29   11  05BE  2780          BRB     500$            ; exit error                            [101]
                                05C0  2781
                                05C0  2782
                                05C0  2783 ;         5) else print error in command statement, exit.
                                05C0  2784
                                05C0  2785 300$:
                                05C0  2786
              00000008 E6   9F  05C0  2787          PUSHAB  L^CLI$Q_FILE(R6)        ; PUSH the QUAD  of device name [101]
                                05C6  2788
              00000113'EF   9F  05C6  2789          PUSHAB  L^T_EMESSG3             ;                          [101]     ;G:5
                        01   DD  05CC  2790          pushl   #ds$k_printf            ; Print with PRINTF        [101]     ;G:5
                        15   DD  05CE  2791          pushl   #ds$k_type_command_err  ; Command error code       [101]     ;G:5
          00000000'GF   04   FB  05D0  2792          CALLS   #4, G^DSX$PRINT         ; Print                    [101]     ;G:5
                                05D7  2793
                                05D7  2794
              00000472'EF   94  05D7  2795 400$:    CLRB    MP$GB_SETCPU            ; clear the flag       [101]         ;G:5
00000478'EF   00000455'EF   08   28  05DD  2796          MOVC3   #8,PRIMARY_NAME,     -  ; Reload "primary" cpu  [118]         ;G:5
                                05E9  2797                      MP$GT_DEVICE_NAME+1     ;                      [118]         ;G:5
                                05E9  2798
                                05E9  2799 500$:
                   00FF 8F   BA  05E9  2800          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7>    ; RESTORE REGISTERS     [101]
                                05ED  2801
                            05  05ED  2802          RSB                             ; RETURN.                      [101]
                                05EE  2803                                                                                ;G:3
                                05EE  2804 ;                                                                              ;G:3
                                05EE  2805 ;+                                                                             ;G:3
                                05EE  2806 ; SHOW CPU COMMAND                                          [116]              ;G:3
                                05EE  2807 ;-                                                                             ;G:3
                                05EE  2808 ;                                                                              ;G:3
                                05EE  2809 ACT_SHOWCPU:                                                ;[116]             ;G:3
          04 A2    F4'AF   9E  05EE  2810          MOVAB   B^10$,CLI$L_COMMAND(R2)         ; Set command processor [116] ;G:3
                            05  05F3  2811          RSB                                                                   ;G:3
                                05F4  2812                                                                                ;G:3
                                05F4  2813 10$:     Br_If_Not_User  20$     ; Branch around if not in user mode[116]      ;G:3
          0000FE00'EF   1C   E1  05F4                  BBC     #DSA$V_User, -                  ; If bit not set, branch
                        14       05FB                            L^DSA$GL_Flags, 20$           ;
                                05FC  2814          $PRINT  #ds$k_type_command_err, -       ; Print command error [116]   ;G:3
                                05FC  2815                    #ds$k_printf, -               ; .. use PRINTF       [116]   ;G:3
                                05FC  2816                    L^T_EMESSG1                   ; Can't do            [116]   ;G:3
              000000BD'EF   9F  05FC                        PUSHAB  L^T_EMESSG1
                   7E   01   9A  0602                        movzbl  #ds$k_printf, -(sp)
                   7E   15   98  0605                        cvtbl   #ds$k_type_command_err, -(sp)
          00000000'GF   03   FB  0608                        CALLS   #$$N+2, G^DSX$PRINT
                            05  060F  2817          RSB                                     ;                     [116]   ;G:3
                                0610  2818                                                                                ;G:3
              00000477'EF   9F  0610  2819 20$:     PUSHAB  MP$GT_DEVICE_NAME               ; PRIMARY or KAn ascii[116]   ;G:3
              00000288'EF   9F  0616  2820          PUSHAB  T_SHOWCPU                       ; Address of text     [116]   ;G:3
                        01   DD  061C  2821          pushl   #ds$k_printf                    ; Print with PRINTF   [116]   ;G:3
                        16   DD  061E  2822          pushl   #ds$k_type_command_out          ; Command output code [116]   ;G:3
          00000000'GF   04   FB  0620  2823          CALLS   #4, G^DSX$PRINT                 ; Print               [116]   ;G:3
                            05  0627  2824          RSB                                     ; All done            [116]   ;G:3
                                0628  2825                                                                                ;G:3
                                0628  2826 ;
```

```
                                0628  2827 ;+
                                0628  2828 ; Check command currently given.
                                0628  2829 ; Set flag in R0 if SET LOAD command AND  file spec should be treated as ULTRIX form
                                0628  2830 ; OR
                                0628  2831 ; Set flag if ULTRIX flag set.
                                0628  2832 ; ELSE
                                0628  2833 ; Clear R0.
                                0628  2834 ;-
                                0628  2835 ACT_ULTRIX_ODS:
                          03    BB  0628  2836          PUSHR   #^M<R0,R1>              ; Save register                          [106
                          6E    D4  062A  2837          CLRL    (SP)                    ; Initialize                             [106
                50  00000000'EF   DE  062C  2838          MOVAL   DSV$SETLOAD, R0         ; Address of SET LOAD routine            [106
                     04 A2    50   D1  0633  2839          CMPL    R0,CLI$L_COMMAND(R2)    ; Check if same command                  [106
                          13    12  0637  2840          BNEQ    10$                     ; Branch if not SET LOAD command        [106
             68   59   5B    8F   3A  0639  2841          LOCC    #^A"[", R9, (R8)        ; Find how to parse filespec             [106
                          1A    12  063E  2842          BNEQ    20$                     ; Branch to treat as ODS filespec        [106
                68   59    3C   3A  0640  2843          LOCC    #^A"<", R9, (R8)        ; Find how to parse filespec             [106
                          14    12  0644  2844          BNEQ    20$                     ; Branch to treat as ODS filespec        [106
                68   59    28   3A  0646  2845          LOCC    #^A"(", R9, (R8)        ;
                          0C    12  064A  2846          BNEQ    15$                     ; Branch if ULTRIX                       [106
                                0  064C  2847
                 00000000'8F    E1  064C  2848 10$:      BBC     #ULTRIX_V_MODE,-        ; Use current state for parsing          [106
              02 00000000'EF        0652  2849                  L^ULTRIX_FLAGS,20$      ; Test flag for ULTRIX                   [106
                          6E    D6  0658  2850 15$:      INCL    (SP)                    ; Set to indicate proceed as ULTRIX      [106
                          03    BA  065A  2851 20$:      POPR    #^M<R0,R1>              ; Restore register                       [106
                          05    65C  2852               RSB                             ; Return                                 [106
                                065D  2853 ;
                                065D  2854
                                065D  2855 ;+
                                065D  2856 ; SHOW COMMAND.
                                065D  2857 ;-
                                065D  2858 ACT_SHOW:
                    00 62   04    E2  065D  2859          BBSS    #CLI$V_SHOW, -          ; SET SHOW FLG.
                                      0661  2860            CLI$L_FLAGS(R2), 10$
                          05    65C  2861 10$:     RSB                             ; RETURN.
                                0662  2862
                                0662  2863 ;+
                                0662  2864 ; START COMMAND.
                                0662  2865 ;-
                                0662  2866 ACT_START:
                 00000000'EF    D4  0662  2867          CLRL    MP$GL_CONDITION_FLAGS   ; Clear mp condition flags      [103] ;G:11
                 00000000'EF    7C  0668  2868          CLRQ    L^BIN_TIME              ; Init time to run.             [120] ;G:11
00000000'EF      02000000 8F    CA  066E  2869          BICL2   #DS$M_TIMED,DS$GL_FLAGS ; Init /TIME flag               [120] ;G:11
               0444 C2    08    C8  0679  2870          BISL2   #CLI$M_START_OR_RUN, -   ; Indicate a START command     [120] ;G:11
                                067E  2871                  CLI$L_FLAGS2(R2)        ;                               [120] ;G:11
                    00 62   15    E5  067E  2872          BBCC    #CLI$V_RUN, -           ;                               [45]  ;G:11
                                0682  2873            CLI$L_FLAGS (R2), 5$    ; Clear RUN flag                [45]  ;G:11
          04 A2     00000000'EF    DE  0682  2874 5$:      MOVAL   L^VRSTART, -            ; LOAD COMMAND ROUTINE ADDRESS. [41]  ;G:11
                                068A  2875                  CLI$L_COMMAND(R2)
                     2C A2    01   D0  068A  2876          MOVL    #1,CLI$L_PASS(R2)       ; Default pass count is 1
          03 0000FE00'EF        0E   E1  068E  2877          BBC     #DSA$V_SEARCH,L^DSA$GL_FLAGS,10$ ; Branch if search not set[41] ;G:1
                        2C A2   D4  0696  2878          CLRL    CLI$L_PASS(R2)          ; If in SEARCH, default to 0.
                          05    699  2879 10$:     RSB                             ; Return
                                069A  2880
                                069A  2881 ;+
                                069A  2882 ; SCRIPT COMMAND.
                                069A  2883 ;-
```

```
                                069A  ACT_SCRIPT:
     04 A2   00000000'EF   DE   069A          MOVAL   L^SCRIPT$OPEN, -        ; LOAD COMMAND ROUTINE ADDRESS.        [41]
                                06A2                    CLI$L_COMMAND(R2)
                           05   06A2          RSB                                                                    ;G:27
                                06A3
                                06A3  ;+
                                06A3  ; SUMMARY COMMAND.
                                06A3  ;-
                                06A3  ACT_SUMMARY:
     04 A2   00000000'EF   DE   06A3          MOVAL   L^VRSUMMARY, -          ; LOAD COMMAND ROUTINE ADDRESS.        [41]
                                06AB                    CLI$L_COMMAND(R2)
                           05   06AB          RSB
```

```
                                06AC  2897          .SUBTITLE       CRD command
                                06AC  2898
                                06AC  2899 ;+                                                                    [60]  ;G:2
                                06AC  2900 ; CRD COMMAND.                                                        [60]  ;G:2
                                06AC  2901 ;-                                                                    [60]  ;G:2
                                06AC  2902                                                                             ;G:2
                                06AC  2903 ACT_CRD:                                 ;                            [60]  ;G:2
                                06AC  2904          BR_IF_NOT_USER  50$             ;                            [115] ;G:2
          0000FE00'EF    1C E1  06AC                BBC     #DSA$V_User, -                  ; If bit not set, branch
                            16  06B3                  L^DSA$GL_Flags, 50$           ;
                                06B4  2905                                                                             ;G:2
                                06B4  2906          ;+                                                                 ;G:2
                                06B4  2907          ; Print an error message since command not permitted in User mode.[115] ;G:2
                                06B4  2908          ;                                                                  ;G:27
                                06B4  2909          ;-                                                                 ;G:2
                                06B4  2910          $Print  #DS$K_Type_Command_Err,-         ; Print illegal command [115] ;G:2
                                06B4  2911                    #DS$K_PrintI, -            ; . . . use PrintI      [115] ;G:2
                                06B4  2912                    L^T_User_Mode_Menu_Error   ; . . . the format string [115] ;G:2
              00000216'EF    9F 06B4                PUSHAB  L^T_User_Mode_Menu_Error
                    7E    00 9A 06BA                movzbl  #DS$K_PrintI, -(sp)
                    7E    15 98 06BD                cvtbl   #DS$K_Type_Command_Err, -(sp)
          00000000'GF    03 FB 06C0                CALLS   #$$N+2, G^DSX$PRINT
                       00C0 31 06C7  2913          BRW     310$                        ; Exit                       [115] ;G:2
                                06CA  2914
                                06CA  2915          ;   This section sets things up
    0000FE00'EF    00050800 8F D3  06CA  2916 50$:  BITL    #<DSA$M_CRD_Autotest_Off ! -                                   ;G:2
                                06D5  2917                    DSA$M_CRD_Menutest_Off ! -
                                06D5  2918                    DSA$M_CRD_Menutest_On>, DSA$GL_Flags                     ;[72] ;G:2
                          03    13 06D5  2919          BEQL    100$                        ; Branch if NO bits set ; [86] ;G:2
                        0094    31 06D7  2920          BRW     300$                        ; Branch to ERROR message
                                06DA  2921
              00000000'EF    01 91  06DA  2922 100$:  CMPB    #1, CRD$GL_CRD_Command_Debug                             ; [72] ;G:2
                             0D 12  06E1  2923          BNEQ    110$                        ; Branch if NOT equal    ; [86] ;G:2
    0000FE00'EF    00010000 8F C8  06E3  2924 105$:  BISL2   #DSA$M_CRD_Menutest_Off, DSA$GL_Flags                   ; [72] ;G:2
                             2E 11  06EE  2925          BRB     210$                                                     ; [86] ;G:2
                                06F0  2926                                                                             ;G:27
              00000000'EF    02 91  06F0  2927 110$:  CMPB    #2,CRD$GL_CRD_Command_Debug                              ; [72] ;G:2
                             0D 12  06F7  2928          BNEQ    120$                        ; Branch if NOT equal    ; [86] ;G:2
    0000FE00'EF    00000800 8F C8  06F9  2929          BISL2   #DSA$M_CRD_Autotest_Off, DSA$GL_Flags                  ; [72] ;G:2
                             18 11  0704  2930          BRB     210$                                                     ; [86] ;G:2
                                0706  2931
          03 0000FE00'EF    1C E0  0706  2932 120$:  BBS     #DSA$V_User, DSA$GL_Flags, 130$ ; Branch IF user mode    ; [86] ;G:2
                           FFD2 31 070E  2933          BRW     105$                        ; Branch to ERROR message
    0000FE00'EF    00040000 8F C8  0711  2934 130$:  BISL2   #DSA$M_CRD_Menutest_On,  DSA$GL_Flags                   ; [72] ;G:2
                             00 11  071C  2935          BRB     210$                                                     ; [86] ;G:2
                                071E  2936
                                071E  2937          ;   This section envokes CRD
                00000000'EF    16 071E  2938 210$:  Jsb     L^DS_Cleanup                ; Clean up the old diagnostic  [67]  ;G:2
                                0724  2939
                                0724  2940          Br_If_User      215$                ; If User_Mode, branch around  [67]  ;G:2
              0000FE00'EF    1C E0  0724                BBS     #DSA$V_User, -                  ; If bit set, branch             [28]
                             07     072B                  L^DSA$GL_Flags, 215$          ;                                [28]
              00000000'EF    00 FB  072C  2941          Calls   #0, L^MapFree               ; Map free memory            [67]  ;G:2
                                0733  2942
                                0733  2943          ; Clear the DS's 1st and 2cd ^C handler. Clear the user's ^C handler
                00000000'EF    D4 0733  2944 215$:  ClrL    L^DS$GA_DS_Ctrl_C_First                                  ; [67] ;G:2
                00000000'EF    D4 0739  2945          ClrL    L^DS$GA_DS_Ctrl_C_Second                                 ; [66] ;G:2
```

```
                00000000'EF   D4  073F  2946          ClrL    L^DS$L_UserCntrlC                                          ; [66] ;G:2
                                  0745  2947
                                  0745  2948          ; Clear the ^C handler
        00 00000000'EF   00   E4  0745  2949          Bbsc    #DS$V_CtrlC, L^DS$GL_Flags, 220$                           ; [67] ;G:2
                                  074D  2950
                                  074D  2951          ; Disable ^C's for now (set the bit)
        00 00000000'EF   18   E2  074D  2952 220$:    Bbss    #DS$V_DisablCC, L^DS$GL_Flags, 230$                        ; [67] ;G:2
                                  0755  2953
                                  0755  2954          ; Load the CRD image file
           00000000'EF   00   FB  0755  2955 230$:    Calls   #0, L^DSR$Load_CRD                                         ; [67] ;G:2
                                  075C  2956
                                  075C  2957          ;+
                                  075C  2958          ; Call the CRD's initialization routine. If it returns success, we[67] ;G:2
                                  075C  2959          ; are on our way. If it returns failure, print a message and call [67] ;G:2
                                  075C  2960          ; the DSR$Unload_CRD routine so that the CRD region can be      [67]  ;G:2
                                  075C  2961          ; deallocated.                                                  [67]  ;G:2
                                  075C  2962          ;-
                        00   DD  075C  2963          PushL   #0                      ; Push 0 as param-4             [63]  ;G:2
                        00   DD  075E  2964          PushL   #0                      ; Push 0 as param-3             [63]  ;G:2
                        00   DD  0760  2965          PushL   #CRD$K_CRD_Initialization ; Push Initialization as param-2[63] ;G:2
                        00   DD  0762  2966          PushL   #CRD$K_Hook_Point       ; Push Hook_Point as param-1      [63] ;G:2
           00000000'FF   04   FB  0764  2967          CallS   #4, -                   ; Call the CRD$DS_Interface routine[63] ;G:2
                                  076B  2968                    @L^DS$GA_CRD_DS_Interface ; ... to initialize CRD        [63] ;G:2
                     1C 50   E8  076B  2969          BLBS    R0, 310$                ; Branch if Sucess                [67] ;G:2
                                  076E  2970
                                  076E  2971          ;+
                                  076E  2972          ; Come to here if: 1. Already running CRD and the CRD command is   [67] ;G:2
                                  076E  2973          ; issued, or, 2. If the CRD imitialization routine return failure. [67] ;G:2
                                  076E  2974          ; This should never happen, but . . .                             [67] ;G:2
                                  076E  2975          ;-
                                  076E  2976 300$:    $PRINT  #DS$K_TYPE_COMMAND_ERR, #DS$K_PRINTI, L^GT_Menu_Error      ;[63] ;G:
                000001EB'EF   9F  076E                        PUSHAB  L^GT_Menu_Error
                      7E   00   9A  0774                        movzbl  #DS$K_PRINTI, -(sp)
                      7E   15   98  0777                        cvtbl   #DS$K_TYPE_COMMAND_ERR, -(sp)
           00000000'GF   03   FB  077A                        CALLS   #$$N+2, G^DSX$PRINT
                        00   DD  0781  2977          PushL   #0                      ; Don't return if CRD image loaded ;[67] ;G:
           00000000'EF   01   FB  0783  2978          Calls   #1, L^DSR$Unload_CRD                                       ;[67] ;G:
                              05  078A  2979 310$:    RSB                                                                ;[67] ;G:
                                  078B  2980
```

```
                                078B  2982          .SUBTITLE       Filespec, Optional, Required Action Routines
                                078B  2983 ;+
                                078B  2984 ; FILESPEC ACTION ROUTINE.
                                078B  2985 ;-
                                078B  2986 ACT_FILESPEC:
                  0C A2   68    DE  078B  2987          MOVAL   (R8), CLI$Q_FILE+4(R2)  ; SAVE CHARACTER POINTER.
                       08 A2    D4  078F  2988          CLRL    CLI$Q_FILE(R2)          ; CLEAR LENGTH OF STRING
                                05  0792  2989          RSB                             ; RETURN
                                    0793  2990
                                    0793  2991 ;+
                                    0793  2992 ; FILEND ACTION ROUTINE.
                                    0793  2993 ;-
                                    0793  2994 ACT_FILEND:
                  58    0C A2   C3  0793  2995          SUBL3   CLI$Q_FILE+4(R2), -     ; CALC CHARACTER COUNT.
                       08 A2        0797  2996                    R8, CLI$Q_FILE(R2)
                                05  0799  2997          RSB                             ; RETURN.
                                    079A  2998
                                    079A  2999 ;+
                                    079A  3000 ; OPTIONAL Action routine.
                                    079A  3001 ;-
                                    079A  3002 ACT_OPTIONAL:                            ;                               [45]
                  00 62    00   E5  079A  3003          BBCC    #CLI$V_REQUIRED, -      ; Clear required bit            [45]
                                    079E  3004                    CLI$L_FLAGS (R2), 10$ ;                               [45]
                                05  079E  3005 10$:     RSB                             ; Return                        [45]
                                    079F  3006
                                    079F  3007 ;+
                                    079F  3008 ; REQUIRED Action routine.
                                    079F  3009 ;-
                                    079F  3010 ACT_REQUIRED:                            ;                               [45]
                  00 62    00   E3  079F  3011          BBCS    #CLI$V_REQUIRED, -      ; Set required bit              [45]
                                    07A3  3012                    CLI$L_FLAGS (R2), 10$ ;                               [45]
                                05  07A3  3013 10$:     RSB                             ; Return                        [45]
```

```
                                07A4          .SUBTITLE       Start and Run Qualifier Action Routines
                                07A4
                                07A4  ;+                                                                                [62]
                                07A4  ; /DEBUG ACTION ROUTINE                                                           [62]
                                07A4  ;-                                                                                [62]
                                07A4  ACT_DEBUG_SWITCH:                                ;                                [62]
        40000000 8F      C8     07A4          BISL2   #1@DSA$V_LOAD_DEBUGGER,-;SET INDICATOR SO DS WILL LOAD          [62]
        0000FE00'EF             07AA                  DSA$GL_FLAGS             ;  DEBUGGER LATER                      [62]
                         05     07AF          RSB                              ;RETURN                                [62]
                                07B0
                                07B0  ;+
                                07B0  ; /PASSES ACTION ROUTINE.
                                07B0  ;-
                                07B0  ACT_PASS:
         2C A2    5A      D0     07B0          MOVL    R10, CLI$L_PASS(R2)      ; SAVE NUMBER OF PASSES.
                         05     07B4          RSB                              ; RETURN.
                                07B5
                                07B5  ;+
                                07B5  ; /QA Action routine
                                07B5  ;-
                                07B5  ACT_QA:
         00 62    07      E2     07B5          BBSS    #CLI$V_QA, -                          ; Set QA flag on          [42]
                                07B9                          CLI$L_FLAGS(R2),10$           ;                         [42]
                         05     07B9  10$:    RSB                                           ;                         [42]
                                07BA
                                07BA  ;+
                                07BA  ; /SECTION ACTION ROUTINE.
                                07BA  ;-
                                07BA  ACT_SECTION:
      10 A2    58    51    C3     07BA          SUBL3   R1, R8, -                ; CALCULATE SECTION NAME SIZE.
                                07BF                    CLI$Q_SECTION(R2)
         14 A2    51      D0     07BF          MOVL    R1, CLI$Q_SECTION+4(R2) ; ADDRESS OF SECTION NAME.
                         05     07C3          RSB                              ; RETURN.
                                07C4
                                07C4  ;+
                                07C4  ; /SUBTEST ACTION ROUTINE.
                                07C4  ;-
                                07C4  ACT_SUBTEST:
         28 A2    5A      D0     07C4          MOVL    R10, CLI$L_SUBT(R2)      ; SAVE SUBTEST NUMBER.
                         05     07C8          RSB                              ; RETURN.
                                07C9
                                07C9  ;+
                                07C9  ; /TEST ACTION ROUTINE.
                                07C9  ;-
                                07C9  ACT_TEST:
         20 A2    5A      D0     07C9          MOVL    R10, CLI$L_TEST(R2)      ; SAVE FIRST TEST NUMBER.
                         05     07CD          RSB                              ; RETURN.
                                07CE
                                07CE  ;+
                                07CE  ; LAST ACTION ROUTINE.
                                07CE  ;-
                                07CE  ACT_LAST:
         24 A2    5A      D0     07CE          MOVL    R10, CLI$L_LAST(R2)      ; SAVE LAST TEST NUMBER.
                         05     07D2          RSB                              ; RETURN.
                                07D3
                                07D3  ;+
                                07D3  ; BEGINTIME ACTION ROUTINE for /TIME switch     ;                                [76]
```

```
                                  07D3  ;-
                                  07D3
                                  07D3  ACT_BEGINTIME:                                  ;                                  [76]
          0440 C2    68    DE     07D3          MOVAL   (R8),CLI$Q_TIME+4(R2)           ; Save current addr of time string [76]
               043C C2    D4     07D8          CLRL    CLI$Q_TIME(R2)                  ; Clear the length field           [76]
                          05     07DC          RSB                                     ;                                  [76]
                                  07DD
                                  07DD  ;+
                                  07DD  ; /TIME ACTION ROUTINE                          ;                                  [76]
                                  07DD  ;-
                                  07DD
                                  07DD  ACT_TIME:                                       ;                                  [76]
               007E 8F    BB     07DD          PUSHR   #^M<R1,R2,R3,R4,R5,R6>          ; MOVC3 destroys R0-R5, LOCC       [76]
                                  07E1                                                  ; destroys R0,R1, don't need R0    [76]
                                  07E1                                                  ; Need R6 to hold R2 temporarily[76]
               56    52    D0     07E1          MOVL    R2,R6                           ; Save R2 (DS$GL_CLIBASE)          [76]
   043C C6   58   0440 C6   C3     07E4          SUBL3   CLI$Q_TIME+4(R6),R8, -          ; Calculate length of time...      [76]
                                  07EC                  CLI$Q_TIME(R6)                  ; string                           [76]
                                  07EC
   0440 D6   043C C6   2D    3A     07EC          LOCC    #^A"-",CLI$Q_TIME(R6), -        ; a "-" means a value for day     [76]
                                  07F4                  @CLI$Q_TIME+4(R6)               ; was typed.                       [76]
                     3B    12     07F4          BNEQ    20$                             ; Branch if found                  [76]
                                  07F6
                                  07F6                                                  ; Not found, copy string to        [76]
                                  07F6                                                  ; stack, in order to insert "0 "[76]
                     7E    B5     07F6  10$:    TSTW    -(SP)                           ; Allocate space for "0 "          [76]
          5E   043C C6    C2     07F8          SUBL2   CLI$Q_TIME(R6),SP               ; Allocate space for string        [76]
          6E   2030 8F    B0     07FD          MOVW    #^A"0 ",(SP)                    ; "0 " means 0 days                [76]
 02 AE   0440 D6   043C C6   28  0802          MOVC3   CLI$Q_TIME(R6), -               ; Move the string from the CLI     [76]
                                  080B                  @CLI$Q_TIME+4(R6),2(SP)         ; data structure to the stack      [76]
          0440 C6    6E    DE     080B          MOVAL   (SP),CLI$Q_TIME+4(R6)           ; Set new address of string        [76]
          043C C6    02    C0     0810          ADDL2   #2,CLI$Q_TIME(R6)               ; Add 2 to the string length       [76]
                                  0815          $BINTIM_S CLI$Q_TIME(R6),L^BIN_TIME ; Convert ASCII to binary time[76]
           00000000'EF    7F     0815                  PUSHAQ  L^BIN_TIME
               043C C6    7F     081B                  PUSHAQ  CLI$Q_TIME(R6)
       00000000'GF    02    FB     081F                  CALLS   #2,G^SYS$BINTIM
          5E   043C C6    C0     0826          ADDL2   CLI$Q_TIME(R6),SP               ; Deallocate stack space           [76]
               007E 8F    BA     082B          POPR    #^M<R1,R2,R3,R4,R5,R6>          ; These registers matter           [76]
                     29    11     082F          BRB     40$
                                  0831
          51   0440 C6    D1     0831  20$:    CMPL    CLI$Q_TIME+4(R6),R1             ; "-" first char in string?        [76]
                     0A    12     0836          BNEQ    30$                             ; Branch if not                    [76]
               043C C6    D7     0838          DECL    CLI$Q_TIME(R6)                  ; Decr string count (rid of "-")[76]
               0440 C6    D6     083C          INCL    CLI$Q_TIME+4(R6)                ; Incr string address              [76]
                     B4    11     0840          BRB     10$                             ; Go pad in a "0 "                 [76]
                                  0842
                                  0842                                                  ; Here if a day value preceded     [76]
                                  0842                                                  ; the dash                         [76]
               61    20    90     0842  30$:    MOVB    #^A" ",(R1)                     ; Change the "-" to " "            [76]
                                  0845          $BINTIM_S CLI$Q_TIME(R6),L^BIN_TIME ; Convert ascii to binary time[76]
           00000000'EF    7F     0845                  PUSHAQ  L^BIN_TIME
               043C C6    7F     084B                  PUSHAQ  CLI$Q_TIME(R6)
       00000000'GF    02    FB     084F                  CALLS   #2,G^SYS$BINTIM
               007E 8F    BA     0856          POPR    #^M<R1,R2,R3,R4,R5,R6>          ; These registers matter           [76]
                                  085A
                 04 50    E8     085A  40$:    BLBS    R0,50$                          ; Branch if success                [76]
               50    01    D0     085D          MOVL    #1,R0                           ; Failure, set R0 for CLI$K_BIF [76]
```

```
          05  0860  3123          RSB                          ;                                [76]
              0861  3124
     50   D4  0861  3125 50$:     CLRL    R0                   ; Success, set R0 for CLI$K_BIF [76]
          05  0863  3126          RSB                          ;                                [76]
```

```
                                0864          .SUBTITLE       Event and Flags Action Routines
                                0864
                                0864  ;+
                                0864  ; Set/Clear Enforce action routines.  Note that 'SET ENFORCE'
                                0864  ; actually CLEARS the flag, and 'CLEAR ENFORCE' SETS the flag.
                                0864  ; The physical flag is actually 'inhibit enforce'.
                                0864  ;-
                                0864  .Enable LSB
                                0864  Act_SetEnforce:                                  ;                                                [57]
                    50    94    0864          ClrB    R0                               ; Clear value                                    [57]
                    03    11    0866          BrB     Common_Enforce                   ;                                                [57]
                                0868  Act_ClearEnforce:                                ;                                                [57]
              50    01    90    0868          MovB    #1, R0                           ; Set value                                      [57]
                                086B  Common_Enforce:                                  ;                                                [57]
     00000000'EF    50    90    086B          MovB    R0, L^Ds$GB_Inhibit_Naming           ; transfer value                           [57]
                04 A2    D4    0872          ClrL    Cli$L_Command(R2)                ; Make sure no command set                       [57]
                          05    0875          Rsb                                      ;                                                [57]
                                0876  .Disable LSB
                                0876
                                0876  ;+
                                0876  ;  SHOW ENFORCE ROUTINE.                                                                          [89]
                                0876  ;-
                                0876  ACT_SHOWENFORCE:                                      ;                                            [89]
     50    000001E4'EF    9E    0876          MOVAB   L^T_ON,R0                            ; Assume flag set                            [89]
       07 00000000'EF    E9    087D          BLBC    L^DS$GB_INHIBIT_NAMING,10$           ; Check for set flag                         [89]
     50    000001E7'EF    9E    0884          MOVAB   L^T_OFF,R0                           ; Replace with off value                     [89]
                                088B
                                088B  10$:    $PRINT  #DS$K_TYPE_COMMAND_OUT,-             ; Print status of...                         [89]
                                088B                    #DS$K_PRINTI,-                     ; enforce flag...                            [89]
                                088B                    L^T_SHOWENFORCE,-
                                088B                    R0                                 ; on or off.                                 [89]
                    50    DD    088B                  PUSHL   R0
          0000026D'EF    9F    088D                  PUSHAB  L^T_SHOWENFORCE
              7E    00    9A    0893                  movzbl  #DS$K_PRINTI, -(sp)
              7E    16    98    0896                  cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
     00000000'GF    04    FB    0899                  CALLS   #$$N+2, G^DSX$PRINT
                                08A0
          00000004 E2    D4    08A0          CLRL    L^CLI$L_COMMAND(R2)                  ; Make sure no command routine is se
                          05    08A6          RSB                                          ; Return                                     [89]
                                08A7
                                08A7  ;+
                                08A7  ; EVENT ACTION ROUTINE.
                                08A7  ;-
                                08A7  ACT_EVENT:
           00 62    08    E2    08A7          BBSS    #CLI$V_EVENT, -                  ; SET 'EVENT FLAG' FLAG.
                                08AB                    CLI$L_FLAGS(R2), 10$
           0A 62    03    E1    08AB  10$:    BBC     #CLI$V_SET, -                    ; BR IF SET COMMAND.
                                08AF                    CLI$L_FLAGS(R2), 20$
     04 A2    00000000'EF    DE    08AF          MOVAL   L^VRSETEF, -                     ; COMMAND.                                       [41]
                                08B7                    CLI$L_COMMAND(R2)
                    16    11    08B7          BRB     40$
           0A 62    02    E1    08B9  20$:    BBC     #CLI$V_CLEAR, -                  ; BR IF CLEAR COMMAND.
                                08BD                    CLI$L_FLAGS(R2), 30$
     04 A2    00000000'EF    DE    08BD          MOVAL   L^VRCLREF, -                     ; COMMAND.                                       [41]
                                08C5                    CLI$L_COMMAND(R2)
                    08    11    08C5          BRB     40$
     04 A2    00000000'EF    DE    08C7  30$:    MOVAL   L^VRSHOWEF, -                    ; COMMAND.                                       [41]
```

```
                                08CF                          CLI$L_COMMAND(R2)
                          05    08CF  40$:    RSB                                  ; RETURN.
                                08D0
                                08D0  ;+
                                08D0  ; FLAGS ACTION ROUTINE.
                                08D0  ;-
                                08D0  ACT_FLAGS:
                10 62   08   E0 08D0          BBS     #CLI$V_EVENT, -              ; 'EVENT FLAG' FLAG SET ?
                                08D4                    CLI$L_FLAGS(R2), 20$       ; YES, THEN DO NOTHING HERE.
                00 62   09   E2 08D4          BBSS    #CLI$V_FLAGS, -              ; SET 'FLAGS' FLAG.
                                08D8                    CLI$L_FLAGS(R2), 10$
                08 62   04   E1 08D8  10$:    BBC     #CLI$V_SHOW, -               ; BR IF NOT SHOW COMMAND.
                                08DC                    CLI$L_FLAGS(R2), 20$
        04 A2   00000000'EF  DE 08DC          MOVAL   L^VRSHOWFLG, -               ; COMMAND.                                [41]
                                08E4                    CLI$L_COMMAND(R2)
                          05    08E4  20$:    RSB                                  ; RETURN.
```

```
                                08E5  3197          .SUBTITLE       Base, Address, Break Action Routines
                                08E5  3198
                                08E5  3199 ;+
                                08E5  3200 ; BASE ACTION ROUTINE.
                                08E5  3201 ;-
                                08E5  3202 ACT_BASE:
            09 62    04    E1   08E5  3203          BBC     #CLI$V_SHOW, -          ; Branch if not SHOW command          [43]
                                08E9  3204            CLI$L_FLAGS (R2), 10$ ;                                       [43]
     04 A2  00000000'EF    DE   08E9  3205          MOVAL   L^VRSHOWBASE, -         ; "SHOW BASE" command                 [43]
                                08F1  3206            CLI$L_COMMAND (R2)      ;                                     [43]
                          05    08F1  3207          RSB                             ;                                     [43]
     04 A2  00000000'EF    DE   08F2  3208 10$:     MOVAL   L^VRSETBASE, -          ; "SET BASE" COMMAND.                [41]
                                08FA  3209            CLI$L_COMMAND(R2)
                          05    08FA  3210          RSB                             ; RETURN.
                                08FB  3211
                                08FB  3212 ;+
                                08FB  3213 ; ADDRESS ACTION ROUTINE.
                                08FB  3214 ;-
                                08FB  3215 ACT_ADDRESS:
            18 A2    5A    D0   08FB  3216          MOVL    R10, CLI$L_ADDRESS(R2)
  00 00000000 E2     0B    E2   08FF  3217          BBSS    #CLI$V_ADR, -           ; Got address
                                0907  3218            L^CLI$L_FLAGS (R2), 10$
                          05    0907  3219 10$:     RSB
                                0908  3220
                                0908  3221 ;+
                                0908  3222 ; BREAK ACTION ROUTINE.
                                0908  3223 ;-
                                0908  3224 ACT_BREAK:
            00 62    0A    E2   0908  3225          BBSS    #CLI$V_BREAK, -          ; SET 'BREAKPOINT' FLAG.
                                090C  3226            CLI$L_FLAGS(R2), 10$
            09 62    03    E1   090C  3227 10$:     BBC     #CLI$V_SET, -           ; BR IF NOT 'SET' COMMAND.
                                0910  3228            CLI$L_FLAGS(R2), 20$
     04 A2  00000000'EF    DE   0910  3229          MOVAL   L^VRSETBRK, -           ; 'SET BREAK' COMMAND.                [41]
                                0918  3230            CLI$L_COMMAND(R2)
                          05    0918  3231          RSB                             ; RETURN.
            09 62    02    E1   0919  3232 20$:     BBC     #CLI$V_CLEAR, -         ; BR IF NOT 'CLEAR' COMMAND.
                                091D  3233            CLI$L_FLAGS(R2), 30$
     04 A2  00000000'EF    DE   091D  3234          MOVAL   L^VRCLRBRK, -           ; 'CLEAR BREAK' COMMAND.              [41]
                                0925  3235            CLI$L_COMMAND(R2)
                          05    0925  3236          RSB                             ; RETURN.
     04 A2  00000000'EF    DE   0926  3237 30$:     MOVAL   L^VRSHOWBRK, -          ; 'SHOW BREAK' COMMAND.              [41]
                                092E  3238            CLI$L_COMMAND(R2)
                          05    092E  3239          RSB
```

```
                                092F  3241          .SUBTITLE       Base_Qual, Nobase Action Routines
                                092F  3242 ;+
                                092F  3243 ; BASE_QUAL ACTION ROUTINE
                                092F  3244 ;-
                                092F  3245
                                092F  3246 ACT_BASE_QUAL:
    0448 C2    5A    D0         092F  3247          MOVL    R10, CLI$L_TEMP_BASE(R2)        ;SAVE TEMP BASE VALUE
00  0444 C2    02    E2         0934  3248          BBSS    #CLI$V_BASE_QUAL,       -
                                093A  3249                  CLI$L_FLAGS2(R2), 10$
                          05    093A  3250 10$:     RSB                                     ;RETURN
                                093B  3251
                                093B  3252
                                093B  3253 ;+
                                093B  3254 ; NOBASE QUALIFIER ACTION ROUTINE
                                093B  3255 ;-
                                093B  3256
                                093B  3257 ACT_NOBASE:
        0448 C2    D4         093B  3258          CLRL    CLI$L_TEMP_BASE(R2)             ;ZERO TEMP BASE VALUE
00  0444 C2    02    E2         093F  3259          BBSS    #CLI$V_BASE_QUAL,       -
                                0945  3260                  CLI$L_FLAGS2(R2), 10$
                          05    0945  3261 10$:     RSB                                     ;RETURN
```

```
                                0946  3263          .SUBTITLE       All, Default Action Routines
                                0946  3264 ;+
                                0946  3265 ; ALL ACTION ROUTINE.
                                0946  3266 ;-
                                0946  3267
                                0946  3268 ACT_ALL:
                1C A2    00     D2  0946  3269          MCOML   #0, CLI$L_DATA(R2)
                     04 A2     D5  094A  3270          TSTL    CLI$L_COMMAND(R2)
                           01    13  094D  3271          BEQL    10$
                           05    0F  094F  3272          RSB
                0A 62      03    E1  0950  3273 10$:     BBC     #CLI$V_SET, -           ; BR IF NOT 'SET' COMMAND.
                                     0954  3274                    CLI$L_FLAGS(R2), 20$
        04 A2   00000000'EF    DE  0954  3275          MOVAL   L^VRSETFLG, -           ;                                       [41]
                                     095C  3276                    CLI$L_COMMAND(R2)
                           08    11  095C  3277          BRB     30$
        04 A2   00000000'EF    DE  095E  3278 20$:     MOVAL   L^VRCLRFLG, -           ;                                       [41]
                                     0966  3279                    CLI$L_COMMAND(R2)
                           05    05  0966  3280 30$:     RSB                             ; RETURN.
                                0967  3281
                                0967  3282 ;+
                                0967  3283 ; DEFAULT ACTION ROUTINE.
                                0967  3284 ;-
                                0967  3285 ACT_DEFAULT:
                16 62      09    E0  0967  3286          BBS     #CLI$V_FLAGS, -         ; If "SET FLAGS", branch.               [43]
                                     096B  3287                    CLI$L_FLAGS(R2), 20$
                                     096B  3288
                09 62      04    E1  096B  3289          BBC     #CLI$V_SHOW, -          ; Branch if not SHOW command            [43]
                                     096F  3290                    CLI$L_FLAGS (R2), 10$ ;                                 [43]
        04 A2   00000000'EF    DE  096F  3291          MOVAL   L^VRSHOWDFLT, -         ; "SHOW DEFAULT" command                [43]
                                     0977  3292                    CLI$L_COMMAND (R2)    ;                                 [43]
                           05    05  0977  3293          RSB                             ;                                       [43]
                                0978  3294
        04 A2   00000000'EF    DE  0978  3295 10$:     MOVAL   L^VRSETDFLT, -          ; "SET DEFAULT" command.               [41]
                                     0980  3296                    CLI$L_COMMAND(R2)
                           05    05  0980  3297          RSB                             ;                                       [43]
                                0981  3298
                00 62      0C    E2  0981  3299 20$:     BBSS    #CLI$V_DEFAULT, -       ; Set DEFAULT flag for                  [43]
                                     0985  3300                    CLI$L_FLAGS(R2), 30$  ;   "SET FLAGS DEFAULT" command.  [43]
                           05    05  0985  3301 30$:     RSB                             ;                                       [43]
```

```
                                0986  3303          .SUBTITLE       Show Calls Routine                                      ;       [70]
                                0986  3304
                                0986  3305 ;+                                                                                      [70]
                                0986  3306 ; Show Calls action routine.                                                          [70]
                                0986  3307 ;-                                                                                      [70]
                                0986  3308
                                0986  3309 Act_Show_Calls:                                                                       ;       [70]
        04 A2   00000000'EF     DE    0986  3310          MovAL   L^VRShow_Calls, -       ; SHOW CALLS command routine            [70]
                                      098E  3311                    CLI$L_Command (R2)    ;                                       [70]
              1C A2   00        D0    098E  3312          MovL    #0, CLI$L_Data(R2)      ; 0 => show all call frames             [70]
                                05    0992  3313          Rsb                             ; Return to caller                      [70]
```

```
                          0993  3315          .SUBTITLE       CRDTrace Action Routines                        ;       [65]
                          0993  3316
                          0993  3317 ;+
                          0993  3318 ; Set/Clear/Show CRDTrace action routines.                                  [65]
                          0993  3319 ;-
                          0993  3320
                          0993  3321 Act_Set_CRD_Trace:                          ;                             [65]
0000FE00'EF   11    E3    0993  3322          Bbcs    #DSA$V_CRD_Trace, -       ; Set the bit on              [65]
              00          099A  3323            L^DSA$GL_Flags, 10$             ;                             [65]
                          099B  3324
              35    11    099B  3325 10$:     BrB     Common_CRD_Trace_Debug    ; Branch to the common stuff  [69]
                          099D  3326
                          099D  3327 Act_Clear_CRD_Trace:                        ;                             [65]
0000FE00'EF   11    E5    099D  3328          Bbcc    #DSA$V_CRD_Trace, -       ; Clear the bit - set it off  [65]
              00          09A4  3329            L^DSA$GL_Flags, 10$             ;                             [65]
                          09A5  3330
              2B    11    09A5  3331 10$:     BrB     Common_CRD_Trace_Debug    ; Branch to the common stuff  [69]
                          09A7  3332
                          09A7  3333 Act_Show_CRD_Trace:                         ;                             [65]
50    000001E4'EF   9E    09A7  3334          MovAB   L^T_On, R0                ; Assume the bit is turned on [65]
0000FE00'EF   11    E0    09AE  3335          Bbs     #DSA$V_CRD_Trace, -       ; If the bit is turned on, . . . [65]
              07          09B5  3336            L^DSA$GL_Flags, -               ; . . . then branch to 10$, . . . [65]
                          09B6  3337            10$                             ; . . . and print it.         [65]
50    000001E7'EF   9E    09B6  3338          MovAB   L^T_Off, R0               ; The bit is off.             [65]
                          09BD  3339
                          09BD  3340 10$:     $Print  #DS$K_Type_Command_Out,-; Print a message giving the status of [65]
                          09BD  3341            #DS$K_PrintI,           -; . . . the CRDTrace flag.           [65]
                          09BD  3342            L^T_CRD_Trace_Output,-;                                       [65]
                          09BD  3343            R0                              ; . . . "on" or "off".        [65]
              50    DD    09BD                  PUSHL   R0
        00000250'EF   9F    09BF                  PUSHAB  L^T_CRD_Trace_Output
           7E   00    9A    09C5                  movzbl  #DS$K_PrintI, -(sp)
           7E   16    98    09C8                  cvtbl   #DS$K_Type_Command_Out, -(sp)
00000000'GF   04    FB    09CB                  CALLS   #$$N+2, G^DSX$PRINT
                          09D2  3344
                          09D2  3345 Common_CRD_Trace_Debug:                     ; The common stuff            [69]
        00000004 E2   D4    09D2  3346          ClrL    L^Cli$L_Command(R2)       ; Make sure no command routine is set [65]
              05          09D8  3347          Rsb                               ; Return to the DS$Cli routine [65]
                          09D9  3348
                          09D9  3349 Act_Set_CRD_Debug:                          ; Set the CRD debug bit - for CRD debug [69]
                          09D9  3350                                             ; ... statements              [69]
00000000'EF   01    D0    09D9  3351          MovL    #1, L^DS$GL_CRD_Debug     ; 1 => debug is on.           [69]
              B1    11    09E0  3352          Brb     Act_Set_CRD_Trace         ; Turn Trace on also          [69]
                          09E2  3353
                          09E2  3354 Act_Clear_CRD_Debug:                        ; Clear the CRD debug bit - for CRD [69]
                          09E2  3355                                             ; ... debug statements        [69]
        00000000'EF   D4    09E2  3356          ClrL    L^DS$GL_CRD_Debug         ; 0 => no debug statements    [69]
              B3    11    09E8  3357          Brb     Act_Clear_CRD_Trace       ; Clear Trace also            [69]
```

```
                          09EA  3359          .SUBTITLE       Ask_prompt, Bell, Binary, Halt, IEx Action Routines
                          09EA  3360
                          09EA  3361 ;+
                          09EA  3362 ; ASK_PROMPT ACTION ROUTINE.                                          ;          [108
                          09EA  3363 ;-
                          09EA  3364 ACT_ASK_PROMPT:
00 1C A2   13   E2        09EA  3365          BBSS    #DSA$V_ASK_PROMPT, -
                          09EF  3366                    CLI$L_DATA(R2), 10$
                    05    09EF  3367 10$:     RSB
                          09F0  3368
                          09F0  3369 ;+
                          09F0  3370 ; BELL ACTION ROUTINE.
                          09F0  3371 ;-
                          09F0  3372 ACT_BELL:
00 1C A2   03   E2        09F0  3373          BBSS    #DSA$V_BELL, -
                          09F5  3374                    CLI$L_DATA(R2), 10$
                    05    09F5  3375 10$:     RSB
                          09F6  3376
                          09F6  3377 ;+
                          09F6  3378 ; BINARY ACTION ROUTINE.
                          09F6  3379 ;-
                          09F6  3380 ACT_BINARY:                                                           ;          [45]
00 1C A2   01   E2        09F6  3381          BBSS    #DSA$V_BINARY, -                                     ;          [45]
                          09FB  3382                    CLI$L_DATA(R2), 10$                              ;          [45]
                    05    09FB  3383 10$:     RSB                                                        ;          [45]
                          09FC  3384
                          09FC  3385 ;+
                          09FC  3386 ; HALT ACTION ROUTINE.
                          09FC  3387 ;-
                          09FC  3388 ACT_HALT:
00 1C A2   00   E2        09FC  3389          BBSS    #DSA$V_HALT, -
                          0A01  3390                    CLI$L_DATA(R2), 10$
                    05    0A01  3391 10$:     RSB
                          0A02  3392
                          0A02  3393 ;+
                          0A02  3394 ; IE1 ACTION ROUTINE.
                          0A02  3395 ;-
                          0A02  3396 ACT_IE1:
00 1C A2   04   E2        0A02  3397          BBSS    #DSA$V_IE1, -
                          0A07  3398                    CLI$L_DATA(R2), 10$
                    05    0A07  3399 10$:     RSB
                          0A08  3400
                          0A08  3401 ;+
                          0A08  3402 ; IE2 ACTION ROUTINE.
                          0A08  3403 ;-
                          0A08  3404 ACT_IE2:
00 1C A2   05   E2        0A08  3405          BBSS    #DSA$V_IE2, -
                          0A0D  3406                    CLI$L_DATA(R2), 10$
                    05    0A0D  3407 10$:     RSB
                          0A0E  3408
                          0A0E  3409 ;+
                          0A0E  3410 ; IE3 ACTION ROUTINE.
                          0A0E  3411 ;-
                          0A0E  3412 ACT_IE3:
00 1C A2   06   E2        0A0E  3413          BBSS    #DSA$V_IE3, -
                          0A13  3414                    CLI$L_DATA(R2), 10$
                    05    0A13  3415 10$:     RSB
```

```
                          0A14  3416
                          0A14  3417 ;+
                          0A14  3418 ; IES ACTION ROUTINE.
                          0A14  3419 ;-
                          0A14  3420 ACT_IES:
          00 1C A2   07   E2  0A14  3421          BBSS    #DSA$V_IES, -
                          0A19  3422                    CLI$L_DATA(R2), 10$
                       05 0A19  3423 10$:    RSB
```

```
                          0A1A  3425          .SUBTITLE       Loop, Oper, Prompt Quick Action Routines
                          0A1A  3426
                          0A1A  3427 ;+
                          0A1A  3428 ; LOOP ACTION ROUTINE.
                          0A1A  3429 ;-
                          0A1A  3430 ACT_LOOP:
00 1C A2    02    E2      0A1A  3431          BBSS    #DSA$V_LOOP, -
                          0A1F  3432                    CLI$L_DATA(R2), 10$
                    05    0A1F  3433 10$:     RSB
                          0A20  3434
                          0A20  3435 ;+
                          0A20  3436 ; OPER ACTION ROUTINE.
                          0A20  3437 ;-
                          0A20  3438 ACT_OPER:
00 1C A2    0C    E2      0A20  3439          BBSS    #DSA$V_OPER, -
                          0A25  3440                    CLI$L_DATA(R2), 10$
                    05    0A25  3441 10$:     RSB
                          0A26  3442
                          0A26  3443 ;+
                          0A26  3444 ; PROMPT ACTION ROUTINE.
                          0A26  3445 ;-
                          0A26  3446 ACT_PROMPT:
00 1C A2    0D    E2      0A26  3447          BBSS    #DSA$V_PROMPT, -
                          0A2B  3448                    CLI$L_DATA(R2), 10$
                    05    0A2B  3449 10$:     RSB
                          0A2C  3450
                          0A2C  3451 ;+
                          0A2C  3452 ; QUICK ACTION ROUTINE.
                          0A2C  3453 ;-
                          0A2C  3454 ACT_QUICK:
00 1C A2    08    E2      0A2C  3455          BBSS    #DSA$V_QUICK, -
                          0A31  3456                    CLI$L_DATA(R2), 10$
                    05    0A31  3457 10$:     RSB
```

```
                          0A32  3459          .SUBTITLE       Search, Trace, Verify Action Routines
                          0A32  3460
                          0A32  3461 ;+
                          0A32  3462 ; SEARCH action routine
                          0A32  3463 ;-
                          0A32  3464 ACT_SEARCH:
00 1C A2    0E    E2      0A32  3465          BBSS    #DSA$V_SEARCH, -
                          0A37  3466                    CLI$L_DATA(R2),10$
                    05    0A37  3467 10$:     RSB
                          0A38  3468
                          0A38  3469 ;+
                          0A38  3470 ; TRACE ACTION ROUTINE.
                          0A38  3471 ;-
                          0A38  3472 ACT_TRACE:
00 1C A2    0A    E2      0A38  3473          BBSS    #DSA$V_TRACE, -
                          0A3D  3474                    CLI$L_DATA(R2), 10$
                    05    0A3D  3475 10$:     RSB
                          0A3E  3476
                          0A3E  3477 ;+
                          0A3E  3478 ; VERIFY ACTION ROUTINE.
                          0A3E  3479 ;-
                          0A3E  3480 ACT_VERIFY:                                          ;       [45]
00 1C A2    09    E2      0A3E  3481          BBSS    #DSA$V_VERIFY, -                    ;       [45]
                          0A43  3482                    CLI$L_DATA(R2), 10$               ;       [45]
                    05    0A43  3483 10$:     RSB                                       ;       [45]
```

```
                                0A44          .SUBTITLE       Set/Show Page, Width Action Routine
                                0A44
                                0A44  ;+
                                0A44  ; SET/SHOW PAGE action routine
                                0A44  ;-
                                0A44  ACT_PAGE:                                 ;                                       [71]
          0D 62    04    E0     0A44          BBS     #CLI$V_SHOW, -            ; If SHOW command                       [71]
                                0A48                   CLI$L_FLAGS(R2), 10$     ;  branch                               [71]
          12 62    03    E1     0A48          BBC     #CLI$V_SET, -             ; If not SET command                    [71]
                                0A4C                   CLI$L_FLAGS(R2), 20$     ;  no command                           [71]
   04 A2   00000000'EF    9E     0A4C          MOVAB   DSV$SETPAGE, -            ;                                       [71]
                                0A54                   CLI$L_COMMAND(R2)        ; Set command address                   [71]
                          05     0A54          RSB                               ;                                       [71]
   04 A2   00000000'EF    9E     0A55  10$:    MOVAB   DSV$SHOWPAGE, -           ; Set SHOW command address              [71]
                                0A5D                   CLI$L_COMMAND(R2)        ;                                       [71]
                          05     0A5D          RSB                               ;                                       [71]
                04 A2     D4     0A5E  20$:    CLRL    CLI$L_COMMAND(R2)         ; Make sure no routine address          [71]
                          05     0A61          RSB                               ;                                       [71]
                                0A62
                                0A62  ;+
                                0A62  ; SET/SHOW WIDTH action routine
                                0A62  ;-
                                0A62  ACT_WIDTH:                                ;                                       [38]
          0D 62    04    E0     0A62          BBS     #CLI$V_SHOW, -            ; If SHOW command                       [38]
                                0A66                   CLI$L_FLAGS(R2), 10$     ;  branch                               [38]
          12 62    03    E1     0A66          BBC     #CLI$V_SET, -             ; If not SET command                    [38]
                                0A6A                   CLI$L_FLAGS(R2), 20$     ;  no command                           [38]
   04 A2   00000000'EF    9E     0A6A          MOVAB   VRSETWIDTH, -             ;                                       [38]
                                0A72                   CLI$L_COMMAND(R2)        ; Set command address                   [38]
                          05     0A72          RSB
   04 A2   00000000'EF    9E     0A73  10$:    MOVAB   VRSHOWWIDTH, -            ; Set SHOW command address              [38]
                                0A7B                   CLI$L_COMMAND(R2)        ;                                       [38]
                          05     0A7B          RSB
                04 A2     D4     0A7C  20$:    CLRL    CLI$L_COMMAND(R2)         ; Make sure no routine address          [38]
                          05     0A7F          RSB                               ;                                       [38]
```

```
                                0A80          .SUBTITLE       Set/Show QA Action Routines
                                0A80
                                0A80  ;                                                                                [42]
                                0A80  ; SET/SHOW QAxxxxxxxxx flags Action routine                                      [42]
                                0A80  ;   This is patterned after the SET/SHOW width action routine above              [42]
                                0A80  ;                                                                                [42]
                                0A80  ACT_QAEP:                                                  ;                     [42]
          18 62    1B    E2     0A80          BBSS    #CLI$V_QAERRORPRINTS, -                    ; Set the bit on      [42]
                                0A84                          CLI$L_FLAGS(R2), QA_COMMON         ;   reqardless        [42]
                   16    11     0A84          BRB     QA_COMMON                                  ; Branch to common stuff[42]
                                0A86
                                0A86  ACT_QACL:                                                  ;                     [42]
          12 62    1C    E2     0A86          BBSS    #CLI$V_QACKLOOPLOOPS, -                    ; Set the bit on      [42]
                                0A8A                          CLI$L_FLAGS(R2), QA_COMMON         ;   reqardless        [42]
                   10    11     0A8A          BRB     QA_COMMON                                  ; Branch to common stuff[42]
                                0A8C
                                0A8C  ACT_QATL:                                                  ;                     [42]
          0C 62    1D    E2     0A8C          BBSS    #CLI$V_QATESTLOOPS, -                      ; Set the bit on      [42]
                                0A90                          CLI$L_FLAGS(R2), QA_COMMON         ;   reqardless        [42]
                   0A    11     0A90          BRB     QA_COMMON                                  ; Branch to common stuff[42]
                                0A92
                                0A92  ACT_QASL:                                                  ;                     [42]
          06 62    1E    E2     0A92          BBSS    #CLI$V_QASUBTESTLOOPS, -                   ; Set the bit on      [42]
                                0A96                          CLI$L_FLAGS(R2), QA_COMMON         ;   reqardless        [42]
                   04    11     0A96          BRB     QA_COMMON                                  ; Branch to common stuff[42]
                                0A98
                                0A98  ACT_QAMP:                                                  ;                     [42]
          00 62    1F    E2     0A98          BBSS    #CLI$V_QAMULTIPLEPASS, -                   ; Set the bit on      [42]
                                0A9C                          CLI$L_FLAGS(R2), QA_COMMON         ;   reqardless        [42]
                                0A9C                                                             ; Cont onto common stuff[42]
                                0A9C
                                0A9C  QA_COMMON:                                                 ;                     [42]
          0D 62    04    E0     0A9C          BBS     #CLI$V_SHOW, -                             ; If SHOW, branch to 10$[42]
                                0AA0                          CLI$L_FLAGS(R2), 10$               ;   If not, ...       [42]
          12 62    03    E1     0AA0          BBC     #CLI$V_SET, -                              ; If not SET or SHOW,  [42]
                                0AA4                          CLI$L_FLAGS(R2), 20$               ;    branch to 20$    [42]
                                0AA4                                                             ; If SET, ...         [42]
  04 A2   00000000'EF    9E     0AA4          MOVAB   VRSETQA, -                                 ; Move address of SETQA [42]
                                0AAC                          CLI$L_COMMAND(R2)                  ;   routine to command [42]
                                0AAC                                                             ;   address           [42]
                         05     0AAC          RSB
  04 A2   00000000'EF    9E     0AAD  10$:    MOVAB   VRSHOWQA, -                                ; Move address of SHOWQA[42]
                                0AB5                          CLI$L_COMMAND(R2)                  ;   routine to command [42]
                                0AB5                                                             ;   address
                         05     0AB5          RSB
                04 A2    D4     0AB6  20$:    CLRL    CLI$L_COMMAND(R2)                          ; No routine address  [42]
                         05     0AB9          RSB                                                ;                     [42]
```

```
                                0ABA          .SUBTITLE       QADef Action Routine
                                0ABA
                                0ABA  ACT_QADEF:                                                  ;                                [42]
          DE 62    0C    E2     0ABA          BBSS    #CLI$V_DEFAULT, -                           ; Set the DEFAULT bit            [42]
                                0ABE                          CLI$L_FLAGS(R2), QA_COMMON          ;   on regardless                [42]
                                0ABE                                                              ; The DEFAULT bit isn't          [42]
                                0ABE                                                              ;   really for QA, but           [42]
                                0ABE                                                              ;   will use it anyway!          [42]
                                0ABE
          0D 62    04    E0     0ABE          BBS     #CLI$V_SHOW, -                              ; If SHOW, branch to 10$         [42]
                                0AC2                          CLI$L_FLAGS(R2), 10$                ;   If not, ...                  [42]
          12 62    03    E1     0AC2          BBC     #CLI$V_SET, -                               ; If not SET or SHOW,            [42]
                                0AC6                          CLI$L_FLAGS(R2), 20$                ;    branch to 20$               [42]
                                0AC6                                                              ; If SET, ...                    [42]
  04 A2   00000000'EF    9E     0AC6          MOVAB   VRSETQA, -                                  ; Move address of SETQA          [42]
                                0ACE                          CLI$L_COMMAND(R2)                   ;   routine to command           [42]
                                0ACE                                                              ;   address                      [42]
                         05     0ACE          RSB
                                0ACF
  04 A2   00000000'EF    9E     0ACF  10$:    MOVAB   VRSHOWQA, -                                 ; Move address of SHOWQA         [42]
                                0AD7                          CLI$L_COMMAND(R2)                   ;   routine to command           [42]
                                0AD7                                                              ;   address
                         05     0AD7          RSB
                 04 A2   D4     0AD8  20$:    CLRL    CLI$L_COMMAND(R2)                           ; No routine address             [42]
                         05     0ADB          RSB                                                 ;                                [42]
```

```
                                0ADC  3595          .SUBTITLE       Efn, Next, Ascii Action Routines
                                0ADC  3596
                                0ADC  3597 ;+
                                0ADC  3598 ; EFN ACTION ROUTINE.
                                0ADC  3599 ;-
                                0ADC  3600 ACT_EFN:
                          5A    D5  0ADC  3601 10$:     TSTL    R10                     ; CHECK FOR EFN 0.
                          0A    13  0ADE  3602          BEQL    30$
                       17 5A    D1  0AE0  3603          CMPL    R10, #23                ; CHECK FOR IN RANGE.
                          05    1A  0AE3  3604          BGTRU   30$
                    11 1C A2    5A  E3  0AE5  3605 20$:     BBCS    R10, CLI$L_DATA(R2), 40$
                                0AEA  3606 30$:                                         ; Print illegal event flag number       [45]
                 00000061'EF    9F  0AEA  3607          PUSHAB  L^T_ILLEFN              ;                                       [45]
                          00    DD  0AF0  3608          pushl   #ds$k_printi            ; Use printi                            [48]
                          15    DD  0AF2  3609          pushl   #ds$k_type_command_err  ; Command error                         [48]
              00000000'GF 03    FB  0AF4  3610          CALLS   #3, G^DSX$PRINT         ;                                       [48]
                                0AFB  3611
                                0AFB  3612          ;BBCS   #CLI$V_ERROR, -         ; Illegal event flag number             [45]
                                0AFB  3613          ;         CLI$L_FLAGS(R2),40$   ;                                       [45]
                                0AFB  3614
                          05    0AFB  3615 40$:     RSB                             ; RETURN
                                0AFC  3616
                                0AFC  3617 ;+
                                0AFC  3618 ; NEXT ACTION ROUTINE.
                                0AFC  3619 ;-
                                0AFC  3620 ACT_NEXT:
                       30 A2    5A  D0  0AFC  3621          MOVL    R10, CLI$L_NEXT(R2)     ; SAVE NEXT COUNT.
                          05    0B00  3622          RSB                             ; RETURN
                                0B01  3623
                                0B01  3624 ;+
                                0B01  3625 ; ASCII ACTION ROUTINE.
                                0B01  3626 ;-
                                0B01  3627 ACT_ASCII:
                 00 62    13    E2  0B01  3628          BBSS    #CLI$V_ASCII, -
                                0B05  3629                    CLI$L_FLAGS(R2), 10$
                          05    0B05  3630 10$:     RSB                             ; RETURN.
```

```
                                0B06  3632          .SUBTITLE       Register Action Routines
                                0B06  3633
                                0B06  3634 ;+
                                0B06  3635 ; REGISTER ACTION ROUTINES.
                                0B06  3636 ;-
                                0B06  3637 ACT_REGN:
                00 62   14  E2  0B06  3638          BBSS    #CLI$V_REG, -
                                0B0A  3639                    CLI$L_FLAGS(R2), 10$
                            05  0B0A  3640 10$:     RSB                             ; RETURN.
                                0B0B  3641
                                0B0B  3642 ACT_REG12:
                18 A2   0C  D0  0B0B  3643          MOVL    #12, CLI$L_ADDRESS(R2)  ; AP.
                            05  0B0F  3644          RSB
                                0B10  3645 ACT_REG13:
                18 A2   0D  D0  0B10  3646          MOVL    #13, CLI$L_ADDRESS(R2)  ; FP.
                            05  0B14  3647          RSB
                                0B15  3648 ACT_REG14:
                18 A2   0E  D0  0B15  3649          MOVL    #14, CLI$L_ADDRESS(R2)  ; SP.
                            05  0B19  3650          RSB
                                0B1A  3651 ACT_REG15:
                18 A2   0F  D0  0B1A  3652          MOVL    #15, CLI$L_ADDRESS(R2)  ; PC.
                            04  11  0B1E  3653          BRB     ACT_REGP
                                0B20  3654
                                0B20  3655 ACT_REG16:
                18 A2   10  D0  0B20  3656          MOVL    #16, CLI$L_ADDRESS(R2)  ; PSL.
                                0B24  3657 ACT_REGP:
                00 62   14  E2  0B24  3658          BBSS    #CLI$V_REG, -
                                0B28  3659                    CLI$L_FLAGS(R2), ACT_REGNUF
                                0B28  3660 ACT_REGNUF:
                00 62   0B  E2  0B28  3661          BBSS    #CLI$V_ADR, -
                                0B2C  3662                    CLI$L_FLAGS(R2), 10$
                            05  0B2C  3663 10$:     RSB                             ; RETURN.
                                0B2D  3664
                                0B2D  3665 ;+
                                0B2D  3666 ; PROCESSOR REGISTER ACTION ROUTINES
                                0B2D  3667 ;-
                                0B2D  3668 ACT_PREGN:
                00 62   1A  E2  0B2D  3669          BBSS    #CLI$V_PREG,CLI$L_FLAGS(R2),10$ ; Set command type bit
                    FDC7    31  0B31  3670 10$:     BRW     ACT_ADDRESS                     ; Store register code
```

```
                     0B34  3672          .SUBTITLE       Backup, Advance, Data Action Routines
                     0B34  3673
                     0B34  3674 ;+
                     0B34  3675 ; BACKUP ACTION ROUTINE.
                     0B34  3676 ;-
                     0B34  3677 ACT_BACKUP:
                58   D7  0B34  3678          DECL    R8                      ; Backup one character in the
                59   D6  0B36  3679          INCL    R9                      ; line. Increment the remaining
                     0B38  3680                                              ; character count.
                     05  0B38  3681          RSB                             ; Return
                     0B39  3682
                     0B39  3683 ;+
                     0B39  3684 ; ADVANCE Action routine.
                     0B39  3685 ;-
                     0B39  3686 ACT_ADVANCE:
                58   D6  0B39  3687          INCL    R8                      ; Advance one character in      [45]
                59   D7  0B3B  3688          DECL    R9                      ; line. Decrement the remaining [45]
                     0B3D  3689                                              ; character count.              [45]
                     05  0B3D  3690          RSB                             ; Return                        [45]
                     0B3E  3691 ;+
                     0B3E  3692 ; DATA ACTION ROUTINE.
                     0B3E  3693 ;-
                     0B3E  3694 ACT_DATA:
             1C A2   5A  D0  0B3E  3695          MOVL    R10, CLI$L_DATA(R2)     ; SAVE THE DATA.
                     05  0B42  3696          RSB                             ; RETURN.
```

```
                                0B43  3698
                                0B43  3699
                                0B43  3700 ;*
                                0B43  3701 ; DATA ACTION ROUTINE.
                                0B43  3702 ;-
                                0B43  3703 ACT_NULL_DATA:
      1C A2   FFFFFFFF 8F   D0  0B43  3704          MOVL    #-1, CLI$L_DATA(R2)     ; Store a default value of -1   [97]
                            05  0B4B  3705          RSB
                                0B4C  3706
```

```
                                0B4C  3708          .SUBTITLE       Long, Word, Byte, Hex, Dec, Oct Action Routines
                                0B4C  3709
                                0B4C  3710 ;+
                                0B4C  3711 ; LONG ACTION ROUTINE.
                                0B4C  3712 ;-
                                0B4C  3713 ACT_LONG:
                00 62   0F   E3 0B4C  3714          BBCS    #CLI$V_LONG, -
                                0B50  3715                    CLI$L_FLAGS(R2), 10$
                             05 0B50  3716 10$:     RSB                             ; RETURN.
                                0B51  3717
                                0B51  3718 ;+
                                0B51  3719 ; WORD ACTION ROUTINE.
                                0B51  3720 ;-
                                0B51  3721 ACT_WORD:
                00 62   0E   E3 0B51  3722          BBCS    #CLI$V_WORD, -
                                0B55  3723                    CLI$L_FLAGS(R2), 10$
                             05 0B55  3724 10$:     RSB                             ; RETURN.
                                0B56  3725
                                0B56  3726 ;+
                                0B56  3727 ; BYTE ACTION ROUTINE.
                                0B56  3728 ;-
                                0B56  3729 ACT_BYTE:
                00 62   0D   E3 0B56  3730          BBCS    #CLI$V_BYTE, -
                                0B5A  3731                    CLI$L_FLAGS(R2), 10$
                             05 0B5A  3732 10$:     RSB                             ; RETURN.
                                0B5B  3733
                                0B5B  3734 ;+
                                0B5B  3735 ; HEX ACTION ROUTINE.
                                0B5B  3736 ;-
                                0B5B  3737 ACT_HEX:
             00 62   12   E3    0B5B  3738          BBCS    #CLI$V_HEX, -
                                0B5F  3739                    CLI$L_FLAGS(R2), 10$
                             05 0B5F  3740 10$:     RSB                             ; RETURN.
                                0B60  3741
                                0B60  3742 ;+
                                0B60  3743 ; DEC ACTION ROUTINE.
                                0B60  3744 ;-
                                0B60  3745 ACT_DEC:
             00 62   10   E3    0B60  3746          BBCS    #CLI$V_DEC, -
                                0B64  3747                    CLI$L_FLAGS(R2), 10$
                             05 0B64  3748 10$:     RSB                             ; RETURN.
                                0B65  3749
                                0B65  3750 ;+
                                0B65  3751 ; OCT ACTION ROUTINE.
                                0B65  3752 ;-
                                0B65  3753 ACT_OCT:
             00 62   11   E3    0B65  3754          BBCS    #CLI$V_OCT, -
                                0B69  3755                    CLI$L_FLAGS(R2), 10$
                             05 0B69  3756 10$:     RSB                             ; RETURN.
```

```
                                0B6A          .SUBTITLE       IF Action Routines
                                0B6A
                                0B6A  ;+
                                0B6A  ; IF_RUN ACTION ROUTINE.
                                0B6A  ;
                                0B6A  ;       Returns:
                                0B6A  ;               1 => RUN command AND no file-spec has been gotten yet
                                0B6A  ;               0 => Either:
                                0B6A  ;                       Not the RUN command
                                0B6A  ;                    OR the file-spec has already been gotten
                                0B6A  ;-
                                0B6A  ACT_IF_RUN:
                  50    D4      0B6A          CLRL    R0                      ; Assume negative.
          07 62   15    E1      0B6C          BBC     #CLI$V_RUN, -
                                0B70                    CLI$L_FLAGS(R2), 10$  ; If not RUN, return with 0.
              08 A2    D5      0B70          TSTL    CLI$Q_FILE (R2)         ; Check the char. count for the file-spec.
                  02    12      0B73          BNEQ    10$                     ; If > 0, => file-spec is there.
                  50    D6      0B75          INCL    R0                      ; RUN and no file-spec found.
                        05      0B77  10$:    RSB                             ; Return.
                                0B78
                                0B78  ;+
                                0B78  ; IF_REQUIRED Action routine.
                                0B78  ;
                                0B78  ;       Returns:
                                0B78  ;               1 => Something was required for the command
                                0B78  ;               0 => Something was optional for the command
                                0B78  ;-
                                0B78  ACT_IF_REQUIRED:                        ;                                    [45]
                  50    D4      0B78          CLRL    R0                      ;                                    [45]
          02 62   00    E1      0B7A          BBC     #CLI$V_REQUIRED, -      ;                                    [45]
                                0B7E                    CLI$L_FLAGS (R2), 10$ ;                                    [45]
                  50    D6      0B7E          INCL    R0                      ;                                    [45]
                        05      0B80  10$:    RSB                             ;                                    [45]
                                0B81
                                0B81  ;+
                                0B81  ; IF_FILE_SPEC Action routine.
                                0B81  ;
                                0B81  ;       Returns:
                                0B81  ;               1 => A file-spec is present
                                0B81  ;               0 => No file-spec present
                                0B81  ;-
                                0B81  ACT_IF_FILE_SPEC:                       ;                                    [45]
                  50    D4      0B81          CLRL    R0                      ; Assume no file-spec present        [45]
              08 A2    D5      0B83          TSTL    CLI$Q_FILE (R2)         ; See if the char-count is > zero    [45]
                  02    13      0B86          BEQL    10$                     ; If = 0, branch and return 0        [45]
                  50    D6      0B88          INCL    R0                      ; If > 0, return 1                   [45]
                        05      0B8A  10$:    RSB                             ;                                    [45]
                                0B8B
                                0B8B  ;+
                                0B8B  ; IF_ADAPTER Action routine.
                                0B8B  ;
                                0B8B  ;       Returns:
                                0B8B  ;               1 => A adapter is present
                                0B8B  ;               0 => No adapter present
                                0B8B  ;-
                                0B8B  ACT_IF_ADAPTER:                         ;                                    [48]
                  50    D4      0B8B          CLRL    R0                      ; Assume no adapter present          [48]
```

```
                00000000'EF   D5  0B8D  3815          TSTL    L^q_adapter             ; See if the char-count is > zero       [48]
                         02   13  0B93  3816          BEQL    10$                     ; If = 0, branch and return 0           [48]
                         50   D6  0B95  3817          INCL    R0                      ; If > 0, return 1                      [48]
                              05  0B97  3818 10$:     RSB                             ;                                       [48]
```

```
                                0B98  ;+
                                0B98  ; IF_REG_OR_ADR Routine.
                                0B98  ;
                                0B98  ;       Returns:
                                0B98  ;               1 => A register has been given OR an address has been given
                                0B98  ;               0 => No register AND no address were given
                                0B98  ;
                                0B98  ACT_IF_REG_OR_ADR:
                    50    D4    0B98          CLRL    R0                      ; Assume not set
          03 62     14    E1    0B9A          BBC     #CLI$V_REG, -           ; If not REG, branch
                                0B9E                    CLI$L_FLAGS (R2), 20$
                    50    D6    0B9E  10$:    INCL    R0                      ; One of them was set
                          05    0BA0          RSB                             ; Leave with 1
          F9 62     0B    E0    0BA1  20$:    BBS     #CLI$V_ADR, -           ; If ADR, branch
                                0BA5                    CLI$L_FLAGS (R2), 10$
                          05    0BA5          RSB                             ; Leave with 0
                                0BA6
                                0BA6  ;+
                                0BA6  ; IFNOTUSER action routine
                                0BA6  ;
                                0BA6  ;       Returns:
                                0BA6  ;               1 => Not user mode
                                0BA6  ;               0 => User mode
                                0BA6  ;-
                                0BA6  ACT_IFNOTUSER:                          ;                                    [38]
                    50    D4    0BA6          CLRL    R0                      ; Pre-assume failure                 [38]
    0000FE00'EF     1C    E0    0BA8          BBS     #DSA$V_USER, -          ;                                    [38]
                    02          0BAF                    DSA$GL_FLAGS, 10$     ; Branch if not user mode            [38]
                    50    D6    0BB0          INCL    R0                      ; Set R0 to cause branch             [38]
                          05    0BB2  10$:    RSB                             ;                                    [38]
```

```
                                0BB3  3851          .SUBTITLE       Xdelta Action Routines
                                0BB3  3852
                                0BB3  3853 ;+
                                0BB3  3854 ; VAX DEBUGGER ACTION
                                0BB3  3855 ;-
                                0BB3  3856
                                0BB3  3857          .WEAK   XDELBPT,XDELTBIT
                                0BB3  3858 ACT_IF_XDELTA:
     50    00000000'8F    D0   0BB3  3859          MOVL    #XDELBPT,R0             ; Is Xdelta linked?
                     08    13   0BBA  3860          BEQL    10$                     ; Branch if not
     0000FE00'EF     1C    E1   0BBC  3861          BBC     #DSA$V_USER, -
                     03         0BC3  3862                  DSA$GL_FLAGS, 20$       ; Branch if S/A
              50     01    D0   0BC4  3863 10$:     MOVL    #1,R0                   ; Make scanner take branch to error
                           05   0BC7  3864 20$:     RSB                             ; Return
                                0BC8  3865
                                0BC8  3866 ACT_XDELTA:
  04 A2    00000000'EF    9E   0BC8  3867          MOVAB   INI$BRK, -
                                0BD0  3868                  CLI$L_COMMAND(R2)       ; Set command address, Cause breakpoint
                           05   0BD0  3869          RSB                             ; Return to scanner
                                0BD1  3870
                           05   0BD1  3871          RSB                             ; RETURN FOR NEXT COMMAND
```

```
                                0BD2  3873          .SUBTITLE       Device, Brief, Adaptor Action Routines
                                0BD2  3874
                                0BD2  3875 ;+
                                0BD2  3876 ; SHOW DEVICE and SHOW DEVICE/BRIEF
                                0BD2  3877 ;-
                                0BD2  3878 ACT_DEVICE:
                 09 62   1B  E0 0BD2  3879          BBS     #CLI$V_BRIEF,CLI$L_FLAGS(R2),5$ ; Check for brief flag.          [44]
                                0BD6  3880
          04 A2  00000000'EF  9E 0BD6  3881          MOVAB   L^DSV$SHOWDEVICE,CLI$L_COMMAND(R2) ; SHOW DEV COMMAND ROUTINE    [44]
                             05 0BDE  3882          RSB                                     ;                                [44]
                                0BDF  3883
          04 A2  00000000'EF  9E 0BDF  3884 5$:      MOVAB   L^DSV$SHOWDEVICEB,CLI$L_COMMAND(R2)     ; Show device brief     [44]
                             05 0BE7  3885          RSB                                     ;                                [44]
                                0BE8  3886
                                0BE8  3887 ;+
                                0BE8  3888 ; /BRIEF Action routine.
                                0BE8  3889 ;-
                                0BE8  3890 ACT_BRIEF:                                      ;                                [44]
                 00 62   1B  E2 0BE8  3891          BBSS    #CLI$V_BRIEF,CLI$L_FLAGS(R2),20$        ; Set flag              [44]
                             01 BB 0BEC  3892 20$:      PUSHR   #^M<R0>                         ; Save register          [44]
             50  00000000'EF  9E 0BEE  3893          MOVAB   L^DSV$SHOWDEVICE,R0             ; Get address            [44]
                  04 A2   50 D1 0BF5  3894          CMPL    R0,CLI$L_COMMAND(R2)            ; Check current command. [44]
                             0A 12 0BF9  3895          BNEQ    30$                             ; Skip if not show device. [44]
          04 A2  00000000'EF  9E 0BFB  3896          MOVAB   L^DSV$SHOWDEVICEB,CLI$L_COMMAND(R2) ; Change to BRIEF form.    [44]
                             15 11 0C03  3897          BRB     50$                             ;                        [44]
             50  00000000'EF  9E 0C05  3898 30$:      MOVAB   L^DSV$SHOWSELECT,R0             ; Get address            [44]
                  04 A2   50 D1 0C0C  3899          CMPL    R0,CLI$L_COMMAND(R2)            ; Check current command. [44]
                             08 12 0C10  3900          BNEQ    50$                             ;                        [44]
          04 A2  00000000'EF  9E 0C12  3901          MOVAB   L^DSV$SHOWSELECTB,CLI$L_COMMAND(R2)  ; Change to BRIEF form.  [44]
                             01 BA 0C1A  3902 50$:      POPR    #^M<R0>                         ; Restore register       [44]
                             05 0C1C  3903          RSB                                     ;                        [44]
                                0C1D  3904
                                0C1D  3905 ;+
                                0C1D  3906 ; BEGINADAPTER Action routine
                                0C1D  3907 ; Adaptor name storage action routines for /ADAPTER qualifier
                                0C1D  3908 ; on SELECT, DESELECT, and SHOW DEVICE commands.
                                0C1D  3909 ;-
                                0C1D  3910 ACT_BEGINADAPTER:
       00000004'EF   68 DE 0C1D  3911          MOVAL   (R8), L^Q_ADAPTER+4             ; Save current pointer   [43]
            00000000'EF D4 0C24  3912          CLRL    L^Q_ADAPTER                     ; Clear length field     [43]
                             05 0C2A  3913          RSB                                     ; Return to caller       [43]
                                0C2B  3914
                                0C2B  3915 ;+
                                0C2B  3916 ; ADAPTER Action routine.
                                0C2B  3917 ;-
                                0C2B  3918 ACT_ADAPTER:
00000000'EF   58   00000004'EF  C3 0C2B  3919          SUBL3   L^Q_ADAPTER+4, R8, -            ; Calculate count        [43]
                                0C37  3920                    L^Q_ADAPTER               ;                        [43]
                             08 13 0C37  3921          BEQL    10$                             ;                        [43]
          00 00000000 E2   18 E2 0C39  3922          BBSS    #CLI$V_ADAPTER, -               ; Set adaptor bit        [43]
                                0C41  3923                    L^CLI$L_FLAGS (R2), 10$         ;                        [43]
                             05 0C41  3924 10$:      RSB                                     ;                        [43]
```

```
                                0C42  3926          .SUBTITLE       Show Select, Do_Cmd, Set Load, Show Load Action Routines
                                0C42  3927
                                0C42  3928 ;+
                                0C42  3929 ; SHOW SELECT/BRIEF and SHOW SELECT
                                0C42  3930 ;-
                                0C42  3931 ACT_SHOWSEL:
                09 62    1B     E0  0C42  3932          BBS     #CLI$V_BRIEF,CLI$L_FLAGS(R2),5$ ; Check for brief flags          [44]
                                0C46  3933
          04 A2    00000000'EF  9E  0C46  3934          MOVAB   L^DSV$SHOWSELECT,CLI$L_COMMAND(R2) ; SET COMMAND ROUTINE          [41]
                                05  0C4E  3935          RSB                                                              [44]
                                0C4F  3936
          04 A2    00000000'EF  9E  0C4F  3937 5$:      MOVAB   L^DSV$SHOWSELECTB,CLI$L_COMMAND(R2)     ; Show select brief     [44]
                                05  0C57  3938          RSB                                                              [44]
                                0C58  3939
                                0C58  3940 ;+
                                0C58  3941 ; DO_CoMmanD
                                0C58  3942 ;-
                                0C58  3943 ACT_DO_CMD:
                50    04 A2     D0  0C58  3944          MOVL    CLI$L_COMMAND(R2), R0           ; Check value before dispatch   [46]
                      02        13  0C5C  3945          BEQL    10$                             ; Don't dispatch if 0           [46]
                      60        16  0C5E  3946          JSB     (R0)                            ; Perform the routine           [46]
                                05  0C60  3947 10$:     RSB                                     ;                               [46]
                                0C61  3948
                                0C61  3949 ;+
                                0C61  3950 ; SET LOAD COMMAND
                                0C61  3951 ;-
                                0C61  3952 ACT_SETLOAD:
          04 A2    00000000'EF  9E  0C61  3953          MOVAB   DSV$SETLOAD,CLI$L_COMMAND(R2)   ; Set the processor command
                                05  0C69  3954          RSB
                                0C6A  3955
                                0C6A  3956 ;+
                                0C6A  3957 ; SHOW LOAD DEFAULTS
                                0C6A  3958 ;-
                                0C6A  3959 ACT_SHOWLOAD:
          04 A2    00000000'EF  9E  0C6A  3960          MOVAB   DSV$SHOWLOAD,CLI$L_COMMAND(R2)  ; Set the processor address
                                05  0C72  3961          RSB
```

```
                        0C73  3963          .SUBTITLE       MM On, Off, and Show Action Routines
                        0C73  3964
                        0C73  3965 ;+
                        0C73  3966 ; SET MEMORY MANAGEMENT ON ROUTINE
                        0C73  3967 ;-
                        0C73  3968 ACT_MMON:
    04 A2    79'AF   9E  0C73  3969          MOVAB   B^5$,CLI$L_COMMAND(R2); MEMORY ON ROUTINE
                     05 0C78  3970          RSB
                        0C79  3971
                        0C79  3972 5$:      Br_If_Not_User  10$             ; Branch around if not user mode        [55]
0000FE00'EF    1C   E1 0C79               BBC     #DSA$V_User, -                  ; If bit not set, branch
                   12   0C80                 L^DSA$GL_Flags, 10$             ;
     000000BD'EF   9F  0C81  3973          PUSHAB  L^T_EMESSG1                     ;                               [41]
                   01  DD 0C87  3974          pushl   #ds$k_printf                    ; Print with PRINTF             [48]
                   15  DD 0C89  3975          pushl   #ds$k_type_command_err          ; Command error code           [48]
00000000'GF    03   FB 0C8B  3976          CALLS   #3, G^DSX$PRINT                 ; Print                         [48]
                     05 0C92  3977          RSB
                        0C93  3978
00000000'EF    01   90 0C93  3979 10$:     MOVB    #1,DS$GB_MM_ENB                 ; Flag MM ENABLED               [98]
00000000'EF    00   FB 0C9A  3980          CALLS   #0,DSX$MMON
                     05 0CA1  3981          RSB
                        0CA2  3982
                        0CA2  3983
                        0CA2  3984 ;+
                        0CA2  3985 ; SET MEMORY MANAGEMENT OFF ROUTINE
                        0CA2  3986 ;-
                        0CA2  3987 ACT_MMOFF:
    04 A2    A8'AF   9E  0CA2  3988          MOVAB   B^5$,CLI$L_COMMAND(R2); MEMORY OFF ROUTINE
                     05 0CA7  3989          RSB
                        0CA8  3990
                        0CA8  3991 5$:      Br_If_Not_User  10$             ; Branch around if not in user mode     [55]
0000FE00'EF    1C   E1 0CA8               BBC     #DSA$V_User, -                  ; If bit not set, branch
                   12   0CAF                 L^DSA$GL_Flags, 10$             ;
     000000BD'EF   9F  0CB0  3992          PUSHAB  L^T_EMESSG1                     ;                               [41]
                   01  DD 0CB6  3993          pushl   #ds$k_printf                    ; Print with PRINTF             [48]
                   15  DD 0CB8  3994          pushl   #ds$k_type_command_err          ; Command error code           [48]
00000000'GF    03   FB 0CBA  3995          CALLS   #3, G^DSX$PRINT                 ; Print                         [48]
                     05 0CC1  3996          RSB
                        0CC2  3997
     00000000'EF   94  0CC2  3998 10$:     CLRB    DS$GB_MM_ENB                    ; Allow it to turn MM OFF
00000000'EF    00   FB 0CC8  3999          CALLS   #0,DSX$MMOFF
                     05 0CCF  4000          RSB
                        0CD0  4001
                        0CD0  4002 ;+
                        0CD0  4003 ; SHOW STATUS OF MEMORY MANAGEMENT
                        0CD0  4004 ;-
                        0CD0  4005 ACT_SHOWMM:
    04 A2    D6'AF   9E  0CD0  4006          MOVAB   B^10$,CLI$L_COMMAND(R2)         ; Set command processor
                     05 0CD5  4007          RSB
                        0CD6  4008
                        0CD6  4009 10$:     Br_If_Not_User  20$             ; Branch around if not in user mode     [55]
0000FE00'EF    1C   E1 0CD6               BBC     #DSA$V_User, -                  ; If bit not set, branch
                   14   0CDD                 L^DSA$GL_Flags, 20$             ;
                        0CDE  4010          $PRINT  #ds$k_type_command_err, -       ; Print command error           [48]
                        0CDE  4011                  #ds$k_printf, -             ; .. use PRINTF                 [48]
                        0CDE  4012                  L^T_EMESSG1                 ; Can't do                      [48]
     000000BD'EF   9F  0CDE                 PUSHAB  L^T_EMESSG1
```

```
                7E   01   9A  0CE4                          movzbl  #ds$k_printf, -(sp)
                7E   15   98  0CE7                          cvtbl   #ds$k_type_command_err, -(sp)
      00000000'GF    03   FB  0CEA                          CALLS   #$$N+2, G^DSX$PRINT
                          05  0CF1  4013          RSB
                              0CF2  4014
          000001E7'EF    9F  0CF2  4015 20$:     PUSHAB  T_OFF                          ; Assume MM OFF
                    2A   BB  0CF8  4016          PUSHR   #^M<R1,R3,R5>                  ; Save R1,R3 & R5                [122
                55   01   D0  0CFA  4017          MOVL    #1,R5                          ; Pass examine code              [122
                53   38   D0  0CFD  4018          MOVL    #PR$_MAPEN,R3                  ; Pass IPR number                [122
                              0D00  4019          CMK     SGIPR                          ; Ch mode and examine IPR38      [122
                    7E   DC  0D00                MOVPSL  -(SP)
             06 6E   1A   E0  0D02                BBS     #PSL$V_IS, (SP), 30604$
                    8E   D4  0D06                CLRL    (SP)+
                    0A   BC  0D08                CHMK    #CMK$_SGIPR
                    06   11  0D0A                BRB     30605$
          00000000'EF    16  0D0C       30604$:  JSB     CMK$SGIPR
                              0D12       30605$:
                    51   95  0D12  4020          TSTB    R1                             ; is mm on?                      [122
                    2A   BA  0D14  4021          POPR    #^M<R1,R3,R5>                  ; Restore R1,R3 & R5             [122
                    07   13  0D16  4022          BEQL    30$                            ; Branch if not
      6E  000001E4'EF    9E  0D18  4023          MOVAB   T_ON,(SP)                      ; Change insert
                              0D1F  4024
          000001C8'EF    9F  0D1F  4025 30$:     PUSHAB  T_SHOWMM                       ; Address of text
                    01   DD  0D25  4026          pushl   #ds$k_printf                   ; Print with PRINTF              [48]
                    16   DD  0D27  4027          pushl   #ds$k_type_command_out         ; Command output code            [48]
      00000000'GF    04   FB  0D29  4028          CALLS   #4, G^DSX$PRINT                ; Print                          [48]
                          05  0D30  4029          RSB                                    ; All done
```

```
                                0D31          .SUBTITLE       Show Status, Show Support
                                0D31
                                0D31  ;+
                                0D31  ; SHOW STATUS
                                0D31  ;-
                                0D31  ACT_SHOWSTATUS:                                                   ;                [39]
04 A2   00000000'EF     9E      0D31          MOVAB   VRSHOWSTATUS, CLI$L_COMMAND(R2)                   ;                [39]
                        05      0D39          RSB
                                0D3A
                                0D3A  ;+
                                0D3A  ; SHOW SUPPORT
                                0D3A  ;-
                                0D3A  ACT_SHOWSUP:                                      ;                                [44]
04 A2   00000000'EF     9E      0D3A          MOVAB   L^VRSUPPORT,CLI$L_COMMAND(R2)     ; Set command routine          [44]
                        05      0D42          RSB
                                0D43
                                0D43  ;+                                                        ;                        [46]
                                0D43  ; SHOW SECTIONS                                           ;                        [46]
                                0D43  ;-                                                        ;                        [46]
                                0D43  ACT_SHOWSECTIONS:                                         ;                        [46]
04 A2   00000000'EF     9E      0D43          MOVAB   L^VRSHOWSECTIONS,CLI$L_COMMAND(R2)        ; Set command routine   [46]
                        05      0D4B          RSB                                               ;                        [46]
                                0D4C  ;
                                0D4C  ;
                                0D4C  ACT_SHOWTESTS:                                            ;                [81]
04 A2   00000000'EF     9E      0D4C          MOVAB   L^DSV$SHOWTESTS,CLI$L_COMMAND(R2)         ; SHOW TESTS     [81]
                        05      0D54          RSB                                               ;                [81]
                                0D55  ;
                                0D55  ;+
                                0D55  ; SHOW MEMORY
                                0D55  ;-
                                0D55  Act_ShowMem:                                              ;                        [51]
04 A2   00000000'EF     9E      0D55          MovAB   L^DSV$ShowMemory, Cli$L_Command(R2)       ; Show Memory address   [51]
                        05      0D5D          Rsb                                               ;                        [51]
                                0D5E      .Enable LSB                                           ;                        [51]
                                0D5E  Act_ShoMemMap:                                            ; /Map                   [51]
                  50    D4      0D5E          ClrL    R0                                        ;                        [51]
                  0D    11      0D60          BrB     10$                                       ;                        [51]
                                0D62  Act_ShoMemBuf:                                            ; /Buffer                [51]
            50    01    D0      0D62          MovL    #1, R0                                    ;                        [51]
                  08    11      0D65          BrB     10$                                       ;                        [51]
                                0D67  Act_ShoMemDat:                                            ; /Data_Structure        [51]
            50    02    D0      0D67          MovL    #2, R0                                    ;                        [51]
                  03    11      0D6A          BrB     10$                                       ;                        [51]
                                0D6C  Act_ShoMemAll:                                            ; /All                   [51]
            50    03    D0      0D6C          MovL    #3, R0                                    ;                        [51]
00 00000000'EF    50    E2      0D6F  10$:    BbsS    R0, L^ShowMemoryFlags, 20$                ; set the bit            [51]
                        05      0D77  20$:    Rsb                                               ; and return             [51]
                                0D78      .Disable LSB
                                0D78
                                0D78  ;+
                                0D78  ; SET MEMORY
                                0D78  ;-
                                0D78
                                0D78  ACT_SETMEM:                                       ;                        [75]  ;G:6
04 A2   00000D81'EF     9E      0D78          MOVAB   5$, CLI$L_COMMAND(R2)             ; Address of following   [75]  ;G:6
                        05      0D80          RSB                                       ; routine                [75]  ;G:6
```

```
                                0D81  4088                                                                                  ;G:6
                                0D81  4089 5$:     BR_IF_NOT_USER  10$                          ;                    [75]  ;G:6
      0000FE00'EF    1C    E1   0D81               BBC     #DSA$V_User, -                       ; If bit not set, branch
                     12         0D88                         L^DSA$GL_Flags, 10$                ;
                                0D89  4090                                                                                  ;G:6
          000000BD'EF    9F    0D89  4091          PUSHAB  L^T_EMESSG1                          ; Not allowed in user mode[75] ;G:6
                     01    DD   0D8F  4092          PUSHL   #DS$K_PRINTF                         ; Tell user          [75]  ;G:6
                     15    DD   0D91  4093          PUSHL   #DS$K_TYPE_COMMAND_ERR               ;                    [75]  ;G:6
      00000000'GF    03    FB   0D93  4094          CALLS   #3, G^DSX$PRINT                      ;                    [75]  ;G:6
                         05    0D9A  4095          RSB                                          ;                    [75]  ;G:6
                                0D9B  4096
   12 00000000'EF    08    E1   0D9B  4097 10$:     BBC     #AP$V_AP_RUNNING,AP$GW_DATA,20$      ; Not allowed on AP  [119] ;G:8
          000001A7'EF    9F    0DA3  4098          PUSHAB  L^T_EMESSG5                          ; Tell user          [119] ;G:6
                     01    DD   0DA9  4099          PUSHL   #DS$K_PRINTF                         ;                    [119] ;G:6
                     15    DD   0DAB  4100          PUSHL   #DS$K_TYPE_COMMAND_ERR               ;                    [119] ;G:6
      00000000'GF    03    FB   0DAD  4101          CALLS   #3, G^DSX$PRINT                      ;                    [119] ;G:6
                         05    0DB4  4102          RSB                                          ; And exit           [119] ;G:9
                                0DB5  4103
                                0DB5  4104 20$:                                                 ;                    [119] ;G:6
          00000000'EF    16    0DB5  4105          JSB     L^VRSETMEM                           ; Routine to do work [75]  ;G:6
                         05    0DBB  4106          RSB                                          ;                    [75]  ;G:6
                                0DBC  4107
                                0DBC  4108          .END
```

$$$	=	0000008E		D		ACT_HEX		00000B5B	R	D	04
$$N	=	00000001		D		ACT_IE1		00000A02	R	D	04
$ER	=	00000002		D		ACT_IE2		00000A08	R	D	04
$MODULE		00000000	R	D	03	ACT_IE3		00000A0E	R	D	04
ABORT	=	00000006		D		ACT_IES		00000A14	R	D	04
ACT_ABORT		000002C3	R	D	04	ACT_IFNOTUSER		00000BA6	R	D	04
ACT_ADAPTER		00000C2B	R	D	04	ACT_IF_ADAPTER		00000B8B	R	D	04
ACT_ADDRESS		000008FB	R	D	04	ACT_IF_FILE_SPEC		00000B81	R	D	04
ACT_ADVANCE		00000B39	R	D	04	ACT_IF_REG_OR_ADR		00000B98	R	D	04
ACT_ALL		00000946	R	D	04	ACT_IF_REQUIRED		00000B78	R	D	04
ACT_ASCII		00000B01	R	D	04	ACT_IF_RUN		00000B6A	R	D	04
ACT_ASK_PROMPT		000009EA	R	D	04	ACT_IF_XDELTA		00000BB3	R	D	04
ACT_ATTACH		000002B2	R	D	04	ACT_ILL_CMD		00000277	R	D	04
ACT_BACKUP		00000B34	R	D	04	ACT_INC_CMD		0000027D	R	D	04
ACT_BASE		000008E5	R	D	04	ACT_INITPCS		00000437	R	D	04
ACT_BASE_QUAL		0000092F	R	D	04	ACT_LAST		000007CE	R	D	04
ACT_BEGINADAPTER		00000C1D	R	D	04	ACT_LOAD		0000045F	R	D	04
ACT_BEGINTIME		000007D3	R	D	04	ACT_LONG		00000B4C	R	D	04
ACT_BELL		000009F0	R	D	04	ACT_LOOP		00000A1A	R	D	04
ACT_BINARY		000009F6	R	D	04	ACT_MMOFF		00000CA2	R	D	04
ACT_BOOT		000002CC	R	D	04	ACT_MMON		00000C73	R	D	04
ACT_BREAK		00000908	R	D	04	ACT_NEXT		00000AFC	R	D	04
ACT_BRIEF		00000BE8	R	D	04	ACT_NEXT_INST		00000472	R	D	04
ACT_BYTE		00000B56	R	D	04	ACT_NOBASE		0000093B	R	D	04
ACT_CLEAR		000002FF	R	D	04	ACT_NOTNUF		000002A8	R	D	04
ACT_CLEARENFORCE		00000868	R	D	04	ACT_NO_ALLOC		000004D8	RG	D	04
ACT_CLEAR_CRD_DEBUG		000009E2	R	D	04	ACT_NULL		00000276	R	D	04
ACT_CLEAR_CRD_TRACE		0000099D	R	D	04	ACT_NULL_DATA		00000B43	R	D	04
ACT_CONT		0000035B	RG	D	04	ACT_OCT		00000B65	R	D	04
ACT_CONTEXT		00000281	R	D	04	ACT_OPER		00000A20	R	D	04
ACT_CONTINUE		0000030C	R	D	04	ACT_OPTIONAL		0000079A	R	D	04
ACT_CRD		000006AC	R	D	04	ACT_PAGE		00000A44	R	D	04
ACT_DATA		00000B3E	R	D	04	ACT_PASS		000007B0	R	D	04
ACT_DEATTACH		0000035C	R	D	04	ACT_PREGN		00000B2D	R	D	04
ACT_DEBUG		00000365	R	D	04	ACT_PROMPT		00000A26	R	D	04
ACT_DEBUG_SWITCH		000007A4	R	D	04	ACT_QA		000007B5	R	D	04
ACT_DEC		00000B60	R	D	04	ACT_QACL		00000A86	R	D	04
ACT_DEFAULT		00000967	R	D	04	ACT_QADEF		00000ABA	R	D	04
ACT_DEFAULT_DBG		0000036E	RG	D	04	ACT_QAEP		00000A80	R	D	04
ACT_DEPOSIT		00000388	R	D	04	ACT_QAMP		00000A98	R	D	04
ACT_DESELECT		00000395	R	D	04	ACT_QASL		00000A92	R	D	04
ACT_DEVICE		00000BD2	R	D	04	ACT_QATL		00000A8C	R	D	04
ACT_DIRECTORY		0000039E	R	D	04	ACT_QUICK		00000A2C	R	D	04
ACT_DIR_WIDE		000003AD	R	D	04	ACT_REG12		00000B0B	R	D	04
ACT_DO_CMD		00000C58	R	D	04	ACT_REG13		00000B10	R	D	04
ACT_DUMP		000003B5	R	D	04	ACT_REG14		00000B15	R	D	04
ACT_EFN		00000ADC	R	D	04	ACT_REG15		00000B1A	R	D	04
ACT_ENUF		000002AD	R	D	04	ACT_REG16		00000B20	R	D	04
ACT_EVENT		000008A7	R	D	04	ACT_REGN		00000B06	R	D	04
ACT_EXAMINE		000003E0	R	D	04	ACT_REGNUF		00000B28	R	D	04
ACT_EXIT		000003ED	R	D	04	ACT_REGP		00000B24	R	D	04
ACT_FAILURE		0000029F	R	D	04	ACT_REQUIRED		0000079F	R	D	04
ACT_FILEND		00000793	R	D	04	ACT_RUN		0000048F	R	D	04
ACT_FILESPEC		0000078B	R	D	04	ACT_SCRIPT		0000069A	R	D	04
ACT_FLAGS		000008D0	R	D	04	ACT_SEARCH		00000A32	R	D	04
ACT_HALT		000009FC	R	D	04	ACT_SECTION		000007BA	R	D	04
ACT_HELP		000003F6	R	D	04	ACT_SELECT		000004C7	R	D	04

```
ACT_SET                         000004E1 R   D  04
ACT_SETCPU                      000004EE R   D  04
ACT_SETENFORCE                  00000864 R   D  04
ACT_SETLOAD                     00000C61 R   D  04
ACT_SETMEM                      00000D78 R   D  04
ACT_SET_CRD_DEBUG               000009D9 R   D  04
ACT_SET_CRD_TRACE               00000993 R   D  04
ACT_SHOMEMALL                   00000D6C R   D  04
ACT_SHOMEMBUF                   00000D62 R   D  04
ACT_SHOMEMDAT                   00000D67 R   D  04
ACT_SHOMEMMAP                   00000D5E R   D  04
ACT_SHOW                        0000065D R   D  04
ACT_SHOWCPU                     000005EE R   D  04
ACT_SHOWENFORCE                 00000876 R   D  04
ACT_SHOWLOAD                    00000C6A R   D  04
ACT_SHOWMEM                     00000D55 R   D  04
ACT_SHOWMM                      00000CD0 R   D  04
ACT_SHOWSECTIONS                00000D43 R   D  04
ACT_SHOWSEL                     00000C42 R   D  04
ACT_SHOWSTATUS                  00000D31 R   D  04
ACT_SHOWSUP                     00000D3A R   D  04
ACT_SHOWTESTS                   00000D4C R   D  04
ACT_SHOW_CALLS                  00000986 R   D  04
ACT_SHOW_CRD_TRACE              000009A7 R   D  04
ACT_START                       00000662 R   D  04
ACT_SUBTEST                     000007C4 R   D  04
ACT_SUCCESS                     00000296 R   D  04
ACT_SUMMARY                     000006A3 R   D  04
ACT_TEST                        000007C9 R   D  04
ACT_TIME                        000007DD R   D  04
ACT_TRACE                       00000A38 R   D  04
ACT_ULTRIX_ODS                  00000628 R   D  04
ACT_VERIFY                      00000A3E R   D  04
ACT_WIDTH                       00000A62 R   D  04
ACT_WORD                        00000B51 R   D  04
ACT_XDELTA                      00000BC8 R   D  04
ADAPTER                       = 00000079     D
ADDRESS                       = 0000003D     D
ADVANCE                       = 00000065     D
ALL                           = 00000041     D
AP$GK_PROMPT_WAIT             = 00FFFFFF     D
AP$GW_DATA                      ********   X    04
AP$M_AP_NODE_ID               = 000000F0     D
AP$M_AP_RUNNING               = 00000100     D
AP$M_BUFFER_FULL              = 00000800     D
AP$M_DATA_READY               = 00000200     D
AP$M_PP_NODE_ID               = 0000000F     D
AP$M_REQUEST                  = 00000400     D
AP$S_AP_NODE_ID               = 00000004     D
AP$S_PP_NODE_ID               = 00000004     D
AP$V_AP_NODE_ID               = 00000004     D
AP$V_AP_RUNNING               = 00000008     D
AP$V_BUFFER_FULL              = 0000000B     D
AP$V_DATA_READY               = 00000009     D
AP$V_PP_NODE_ID               = 00000000     D
AP$V_REQUEST                  = 0000000A     D
AP$_REQ_CONSRX                = 00000000   G D
AP$_REQ_RXCDVEC                 00000001   G D
APT_COM                         ********     X    04
APT_MSG                         ********     X    04
ASCII                         = 0000005C       D
ASK_PROMPT                    = 00000042       D
ATTACH                        = 00000007       D
BACKUP                        = 00000064       D
BAD_LIST                        000004DF R     D    03
BAD_QUAL                        000004FB R     D    03
BASE                          = 0000003C       D
BASE_QUAL                     = 0000003F       D
BEGIN                           ********     X    04
BEGINADAPTER                  = 00000078       D
BEGINTIME                     = 0000008D       D
BELL                          = 00000044       D
BINARY                        = 00000045       D
BIN_TIME                        ********     X    04
BIT...                        = 0000000C       D
BOOT                          = 00000008       D
BREAK                         = 0000003E       D
BREAK_QUALS                     00000893 R     D    03
BRIEF                         = 00000077       D
BYTE                          = 0000006A       D
B_SUCCESS                       00000454 R     D    02
CLEAR                         = 00000009       D
CLEARENFORCE                  = 00000037       D
CLEAR_CRD_DEBUG               = 00000021       D
CLEAR_CRD_TRACE               = 0000001E       D
CLEAR_PARAMS                    00000C78 R     D    03
CLI                             00000072 R     D    04
CLI$K_ALNUM                   = 00000087       D
CLI$K_ALPHA                   = 00000086       D
CLI$K_BIF                     = 00000083       D
CLI$K_BIFS                    = 00000094       D
CLI$K_BR                      = 00000082       D
CLI$K_BUFSIZ                  = 00000100       D
CLI$K_CALL                    = 00000092       D
CLI$K_COMMA                   = 0000008E       D
CLI$K_DEC                     = 0000008A       D
CLI$K_EOL                     = 00000091       D
CLI$K_ERROR                   = 00000080       D
CLI$K_EXIT                    = 00000081       D
CLI$K_FILE                    = 00000095       D
CLI$K_HEX                     = 00000089       D
CLI$K_KEYWORD                 = 0000008C       D
CLI$K_NUM                     = 00000085       D
CLI$K_OCT                     = 00000088       D
CLI$K_RETURN                  = 00000093       D
CLI$K_SIZE                      0000044C       D
CLI$K_SLASH                   = 00000090       D
CLI$K_SPACE                   = 00000084       D
CLI$K_STRING                  = 0000008B       D
CLI$K_SYMBOL                  = 0000008D       D
CLI$K_VALUE                   = 0000008F       D
CLI$L_ADDRESS                   00000018       D
CLI$L_COMMAND                   00000004       D
CLI$L_DATA                      0000001C       D
```

```
CLI$L_FLAGS                     00000000    D
CLI$L_FLAGS2                    00000444    D
CLI$L_LAST                      00000024    D
CLI$L_NEXT                      00000030    D
CLI$L_PASS                      0000002C    D
CLI$L_SUBT                      00000028    D
CLI$L_TEMP_BASE                 00000448    D
CLI$L_TEST                      00000020    D
CLI$M_START_OR_RUN            = 00000008    D
CLI$Q_BUFQWD                    00000034    D
CLI$Q_FILE                      00000008    D
CLI$Q_SECTION                   00000010    D
CLI$Q_TIME                      0000043C    D
CLI$T_BUFFER                    0000003C    D
CLI$V_ADAPTER                 = 00000018    D
CLI$V_ADR                     = 0000000B    D
CLI$V_ASCII                   = 00000013    D
CLI$V_BASE_QUAL               = 00000002    D
CLI$V_BREAK                   = 0000000A    D
CLI$V_BRIEF                   = 0000001B    D
CLI$V_BYTE                    = 0000000D    D
CLI$V_CLEAR                   = 00000002    D
CLI$V_DEC                     = 00000010    D
CLI$V_DEFAULT                 = 0000000C    D
CLI$V_DEPOSIT                 = 00000019    D
CLI$V_EVENT                   = 00000008    D
CLI$V_EXAM                    = 00000005    D
CLI$V_FLAGS                   = 00000009    D
CLI$V_HEX                     = 00000012    D
CLI$V_KERNEL                  = 00000017    D
CLI$V_LOAD                    = 00000006    D
CLI$V_LONG                    = 0000000F    D
CLI$V_NEXT                    = 00000001    D
CLI$V_NOALLOC                 = 00000000    D
CLI$V_NOTNUF                  = 00000001    D
CLI$V_OCT                     = 00000011    D
CLI$V_PREG                    = 0000001A    D
CLI$V_QA                      = 00000007    D
CLI$V_QACKLOOPLOOPS           = 0000001C    D
CLI$V_QAERRORPRINTS           = 0000001B    D
CLI$V_QAMULTIPLEPASS          = 0000001F    D
CLI$V_QASUBTESTLOOPS          = 0000001E    D
CLI$V_QATESTLOOPS             = 0000001D    D
CLI$V_REG                     = 00000014    D
CLI$V_REQUIRED                = 00000000    D
CLI$V_RUN                     = 00000015    D
CLI$V_SET                     = 00000003    D
CLI$V_SHOW                    = 00000004    D
CLI$V_START_OR_RUN            = 00000003    D
CLI$V_VALSEC                  = 00000016    D
CLI$V_WORD                    = 0000000E    D
CLI_ACTION                      0000013D RG D   04
CMK$APBOOT                      ******** X    04
CMK$SGIPR                       ******** X    04
CMK$_APBOOT                   = 00000004    D
CMK$_ASTEXIT                  = 00000000    D
CMK$_CANTIM                   = 0000000B    D
CMK$_HIBER                    = 00000007    D
CMK$_INITSCB                  = 00000008    D
CMK$_LAST                     = 0000000E    D
CMK$_MMENABLE                 = 00000003    D
CMK$_RCONSOLE                 = 00000001    D
CMK$_REFRESH                  = 00000005    D
CMK$_SCHDWK                   = 00000006    D
CMK$_SETIMR                   = 0000000C    D
CMK$_SETIPL                   = 00000009    D
CMK$_SETPRT                   = 0000000D    D
CMK$_SGIPR                    = 0000000A    D
CMK$_TCONSOLE                 = 00000002    D
COMMAND_TREE                    00000295 RG D   03
COMMON_CRD_TRACE_DEBUG          000009D2 R  D   04
COMMON_ENFORCE                  0000086B R  D   04
CONTEXT                       = 00000003    D
CONTINUE                      = 0000000A    D
CRD                           = 00000027    D
CRD$GL_CRD_COMMAND_DEBUG        ******** X    04
CRD$K_CRD_INITIALIZATION      = 00000000  G D
CRD$K_DS_CONTROL_C            = 00000001  G D
CRD$K_DS_FLAGS                = 00000000  G D
CRD$K_FAILURE                 = 00000000    D
CRD$K_HOOK_POINT              = 00000000  G D
CRD$K_LAST_ACCESS_CODE        = 00000002  G D
CRD$K_LAST_DS_VARIABLE        = 00000002  G D
CRD$K_LAST_FUNCTION           = 00000002  G D
CRD$K_LAST_HOOK_POINT         = 00000001  G D
CRD$K_READ                    = 00000000  G D
CRD$K_SUCCESS                 = 00000001    D
CRD$K_TYPECODE                = 00000001  G D
CRD$K_WRITE                   = 00000001  G D
DATA                          = 00000066    D
DBG_NAME                        0000037F R  D   04
DEATTACH                      = 0000000B    D
DEATTACH_PARAMS                 000009A1 R  D   03
DEBUG                         = 0000000C    D
DEBUG$GB_FLAGS                  ******** X    04
DEBUG_SWITCH                  = 0000000D    D
DEC                           = 0000006C    D
DEFAULT                       = 00000043    D
DEFAULT_DBG                   = 0000000E    D
DEPOSIT                       = 0000000F    D
DESELECT                      = 0000007A    D
DESELECT_PARAMS                 000009E9 R  D   03
DEVICE                        = 00000076    D
DEVICE_NAMES                    00000604 R  D   03
DEVICE_QUALS                    000008BF R  D   03
DIRECTORY                     = 00000010    D
DIR_WIDE                      = 0000008C    D
DO_CMD                        = 0000007E    D
DS$A_PRGBGN                     ******** X    04
DS$CLI                          00000000 RG D   04
DS$GA_CRD_DS_INTERFACE          ******** X    04
DS$GA_DS_CTRL_C_FIRST           ******** X    04
DS$GA_DS_CTRL_C_SECOND          ******** X    04
DS$GB_INHIBIT_NAMING            ******** X    04
```

```
DS$GB_MM_ENB                    ********   X   04
DS$GL_CLIBASE                   00000000 RG  D  02
DS$GL_CRD_DEBUG                 ********   X   04
DS$GL_FLAGS                     ********   X   04
DS$GPHARD                       ********   X   04
DS$GT_PROMPT                    ********   X   04
DS$K_ERROR                    = 00000002     D
DS$K_NORMAL                   = 00000001     D
DS$K_PRINTB                   = 00000002     D
DS$K_PRINTF                   = 00000001     D
DS$K_PRINTI                   = 00000000     D
DS$K_PRINTX                   = 00000003     D
DS$K_SEVERE                   = 00000004     D
DS$K_SUBSYS                   = 00000066     D
DS$K_TYPE_ABORT_PROGRAM       = 00000014     D
DS$K_TYPE_ABORT_TEST          = 00000013     D
DS$K_TYPE_COMMAND_ERR         = 00000015     D
DS$K_TYPE_COMMAND_OUT         = 00000016     D
DS$K_TYPE_CRD_AUTOTEST        = 0000001A     D
DS$K_TYPE_DS_PROMPT           = 00000001     D
DS$K_TYPE_DS_START            = 0000001D     D
DS$K_TYPE_ERRDEV              = 00000008     D
DS$K_TYPE_ERRHARD             = 00000006     D
DS$K_TYPE_ERROR_BODY          = 00000009     D
DS$K_TYPE_ERROR_END           = 0000000A     D
DS$K_TYPE_ERRPREP             = 0000001B     D
DS$K_TYPE_ERRSOFT             = 00000007     D
DS$K_TYPE_ERRSUP              = 00000004     D
DS$K_TYPE_ERRSYS              = 00000005     D
DS$K_TYPE_ERR_HALT            = 0000000D     D
DS$K_TYPE_EXCEPTION           = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD      = 0000000B     D
DS$K_TYPE_FIRST_PASS          = 00000011     D
DS$K_TYPE_GENERAL             = 00000000     D
DS$K_TYPE_GENERAL_ERROR       = 00000003     D
DS$K_TYPE_NO_TESTS            = 00000012     D
DS$K_TYPE_PARAM_ERROR         = 0000001C     D
DS$K_TYPE_PROGRAM_END         = 00000010     D
DS$K_TYPE_PROGRAM_INFO        = 00000017     D
DS$K_TYPE_PROGRAM_START       = 0000000F     D
DS$K_TYPE_QIO_INVADP          = 00000024     D
DS$K_TYPE_QIO_NODRIVER        = 00000022     D
DS$K_TYPE_QIO_WRONGVER        = 00000023     D
DS$K_TYPE_SCRIPT_ECHO         = 00000021     D
DS$K_TYPE_SCRIPT_PNF          = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT       = 00000020     D
DS$K_TYPE_SCRIPT_SKIP         = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR      = 00000019     D
DS$K_TYPE_START_ERR           = 00000018     D
DS$K_TYPE_START_LIST          = 00000025     D
DS$K_TYPE_SUMMARY             = 0000000E     D
DS$K_TYPE_USER_PROMPT         = 00000002     D
DS$K_WARNING                  = 00000000     D
DS$L_USERCNTRLC                 ********   X   04
DS$M_ABRTFLG                  = 00000040     D
DS$M_BADTIME                  = 00100000     D
DS$M_BATCH                    = 00400000     D
DS$M_BRKCLR                   = 00001000     D
DS$M_BRKPT                    = 00000800     D
DS$M_CHARFLG                  = 00000100     D
DS$M_CMDFLG                   = 00000080     D
DS$M_CTRLC                    = 00000001     D
DS$M_CTRLO                    = 00010000     D
DS$M_DEVFLG                   = 00000200     D
DS$M_DISABLCC                 = 01000000     D
DS$M_DONFLG                   = 00002000     D
DS$M_ERRFLG                   = 00000010     D
DS$M_EXCEPT                   = 00080000     D
DS$M_EXETST                   = 00040000     D
DS$M_HLTFLG                   = 00000008     D
DS$M_LODFLG                   = 00000002     D
DS$M_MEMMGT                   = 00008000     D
DS$M_NI                       = 04000000     D
DS$M_OUTPUT                   = 00800000     D
DS$M_RUBFLG                   = 00000020     D
DS$M_SCRIPT                   = 00200000     D
DS$M_SETIMR                   = 02000000     D
DS$M_STRFLG                   = 00000004     D
DS$M_SUBT                     = 00004000     D
DS$M_SYSFLG                   = 00000400     D
DS$M_TIMED                    = 02000000     D
DS$M_TIMED_OUT                = 08000000     D
DS$M_TIMRON                   = 00020000     D
DS$V_ABRTFLG                  = 00000006     D
DS$V_BADTIME                  = 00000014     D
DS$V_BATCH                    = 00000016     D
DS$V_BRKCLR                   = 0000000C     D
DS$V_BRKPT                    = 0000000B     D
DS$V_CHARFLG                  = 00000008     D
DS$V_CMDFLG                   = 00000007     D
DS$V_CTRLC                    = 00000000     D
DS$V_CTRLO                    = 00000010     D
DS$V_DEVFLG                   = 00000009     D
DS$V_DISABLCC                 = 00000018     D
DS$V_DONFLG                   = 0000000D     D
DS$V_ERRFLG                   = 00000004     D
DS$V_EXCEPT                   = 00000013     D
DS$V_EXETST                   = 00000012     D
DS$V_HLTFLG                   = 00000003     D
DS$V_LODFLG                   = 00000001     D
DS$V_MEMMGT                   = 0000000F     D
DS$V_NI                       = 0000001A     D
DS$V_OUTPUT                   = 00000017     D
DS$V_RUBFLG                   = 00000005     D
DS$V_SCRIPT                   = 00000015     D
DS$V_SETIMR                   = 00000019     D
DS$V_STRFLG                   = 00000002     D
DS$V_SUBT                     = 0000000E     D
DS$V_SYSFLG                   = 0000000A     D
DS$V_TIMED                    = 00000019     D
DS$V_TIMED_OUT                = 0000001B     D
DS$V_TIMRON                   = 00000011     D
DS$_AP_NORMAL_BREAK           = 00660150     D
DS$_ARITH                     = 006600D0     D
```

DS$_ASBE	= 00660118	D	DSA$GL_SID	0000FE14	D			
DS$_BADLINK	= 006600F0	D	DSA$GL_SUBTNO	0000FE4C	D			
DS$_BADTYPE	= 006600E8	D	DSA$GL_TESTNO	0000FE50	D			
DS$_BIIC	= 00660120	D	DSA$GL_UNITS	0000FE0C	D			
DS$_CHME	= 006600A8	D	DSA$GQ_MSGPTR	0000FE68	D			
DS$_CHMK	= 006600E0	D	DSA$GT_DEVNAM	0000FE5C	D			
DS$_DEVNAME	= 00660108	D	DSA$M_CRD_AUTOTEST_OFF	= 00000800	D			
DS$_ERROR	= 00660002	D	DSA$M_CRD_MENUTEST_OFF	= 00010000	D			
DS$_FHWE	= 00660068	D	DSA$M_CRD_MENUTEST_ON	= 00040000	D			
DS$_FRAGBUF	= 00660080	D	DSA$V_APT	= 0000001F	D			
DS$_ICBUSY	= 006600C8	D	DSA$V_ASK_PROMPT	= 00000013	D			
DS$_ICERR	= 006600C0	D	DSA$V_BELL	= 00000003	D			
DS$_IHWE	= 00660060	D	DSA$V_BINARY	= 00000001	D			
DS$_ILLCHAR	= 00660018	D	DSA$V_CRD_AUTOTEST_OFF	= 0000000B	D			
DS$_ILLPAGCNT	= 00660078	D	DSA$V_CRD_MENUTEST_OFF	= 00000010	D			
DS$_ILLUNIT	= 00660100	D	DSA$V_CRD_MENUTEST_ON	= 00000012	D			
DS$_INIT_FAIL	= 00660140	D	DSA$V_CRD_TRACE	= 00000011	D			
DS$_INSFMEM	= 00660050	D	DSA$V_HALT	= 00000000	D			
DS$_INVCPU	= 00660130	D	DSA$V_IE1	= 00000004	D			
DS$_IPL2HI	= 006600B8	D	DSA$V_IE2	= 00000005	D			
DS$_IVADDR	= 00660040	D	DSA$V_IE3	= 00000006	D			
DS$_IVVECT	= 00660038	D	DSA$V_IES	= 00000007	D			
DS$_KRNLSTK	= 00660090	D	DSA$V_LOAD_DEBUGGER	= 0000001E	D			
DS$_LOGIC	= 00660070	D	DSA$V_LOOP	= 00000002	D			
DS$_MCHK	= 00660088	D	DSA$V_OPER	= 0000000C	D			
DS$_MEM_ALLOC_ERR	= 00660138	D	DSA$V_PROMPT	= 0000000D	D			
DS$_MMOFF	= 00660058	D	DSA$V_QUICK	= 00000008	D			
DS$_NEEDUNIT	= 006600F8	D	DSA$V_SEARCH	= 0000000E	D			
DS$_NODE	= 00660128	D	DSA$V_TRACE	= 0000000A	D			
DS$_NOPCS	= 00660110	D	DSA$V_USER	= 0000001C	D			
DS$_NORMAL	= 00660001	D	DSA$V_VERIFY	= 00000009	D			
DS$_NOSUPPORT	= 006600B1	D	DSR$COMPLETION	********	X	04		
DS$_NOTALLOWMP	= 00660148	D	DSR$LOAD_CRD	********	X	04		
DS$_NOTDON	= 00660030	D	DSR$UNLOAD_CRD	********	X	04		
DS$_NOTIMP	= 006600B0	D	DSV$ATTACH	********	X	04		
DS$_NULLSTR	= 00660010	D	DSV$DEATTACH	********	X	04		
DS$_OVERFLOW	= 00660008	D	DSV$DEBUG	********	X	04		
DS$_POWER	= 00660098	D	DSV$DESELECT	********	X	04		
DS$_PROGERR	= 00660020	D	DSV$DIRECTORY	********	X	04		
DS$_SEVERE	= 00660004	D	DSV$EXIT	********	X	04		
DS$_TRANSL	= 006600A0	D	DSV$INITPCS	********	X	04		
DS$_TRUNCATE	= 00660028	D	DSV$LOAD	********	X	04		
DS$_UNEXPINT	= 006600D8	D	DSV$RUN	********	X	04		
DS$_UNSUPPDEV	= 00660158	D	DSV$SELECT	********	X	04		
DS$_VASFULL	= 00660048	D	DSV$SETLOAD	********	X	04		
DS$_WARNING	= 00660000	D	DSV$SETPAGE	********	X	04		
DSA$AL_APTMAIL	0000FE00	D	DSV$SHOWDEVICE	********	X	04		
DSA$AT_APTTXT	0000FA00	D	DSV$SHOWDEVICEB	********	X	04		
DSA$GL_APTCOM	0000FE04	D	DSV$SHOWLOAD	********	X	04		
DSA$GL_DEVLEN	0000FE58	D	DSV$SHOWMEMORY	********	X	04		
DSA$GL_ERRNO	0000FE44	D	DSV$SHOWPAGE	********	X	04		
DSA$GL_EVENT	0000FE48	D	DSV$SHOWSELECT	********	X	04		
DSA$GL_FLAGS	0000FE00	D	DSV$SHOWSELECTB	********	X	04		
DSA$GL_MSGTYP	0000FE40	D	DSV$SHOWTESTS	********	X	04		
DSA$GL_PASSES	0000FE08	D	DSX$HELP	********	X	04		
DSA$GL_PASSNO	0000FE54	D	DSX$MMOFF	********	X	04		
DSA$GL_SECTNO	0000FE10	D	DSX$MMON	********	X	04		

```
DSX$PRINT                    ********   X   04
DSX$SUPERPARSE               ********   X   04
DS_CLEANUP                   ********   X   04
DS_ERRSUP                    ********   X   04
DS_SUPERLINE                 ********   X   04
DUMP                       = 00000011    D
DUMP_SELF                    ********W GX   00
EFN                        = 00000058    D
END_FLAGS_LOOP               000005FC R  D  03
ENUF                       = 00000017    D
EOL                          000004AF R  D  03
EVENT                      = 0000003A    D
EVENT_FLAGS                  00000624 R  D  03
EXAMINE                    = 00000012    D
EXA_DEP_QUALS                000008F1 R  D  03
EXIT                       = 00000013    D
FAILURE                    = 00000005    D
FALSE                      = 00000000    D
FILEND                     = 00000029    D
FILESPEC                   = 00000028    D
FLAGS                      = 0000003B    D
FLAGS_LIST                   00000517 R  D  03
FLAGS_LOOP                   00000525 R  D  03
GET_ADDRESS                  00000772 R  D  03
GET_FILE_SPEC                0000064E R  D  03
GT_MENU_ERROR                000001EB RG D  03
HALT                       = 00000046    D
HELP                       = 00000014    D
HELPINFO                     ********   X   04
HEX                        = 0000006B    D
HP$A_DEPENDENT               00000032    D
HP$A_DEVICE                  00000018    D
HP$A_DVA                     0000001C    D
HP$A_LINK                    00000020    D
HP$B_DRIVE                   0000000B    D
HP$B_FLAGS                   0000000A    D
HP$Q_DEVICE                  00000000    D
HP$T_DEVICE                  0000000C    D
HP$T_TYPE                    00000026    D
HP$W_SIZE                    00000008    D
HP$W_VECTOR                  00000024    D
IE1                        = 00000047    D
IE2                        = 00000048    D
IE3                        = 00000049    D
IES                        = 0000004A    D
IFNOTUSER                  = 00000073    D
IF_ADAPTER                 = 00000071    D
IF_FILE_SPEC               = 00000070    D
IF_REG_OR_ADR              = 00000072    D
IF_REQUIRED                = 0000006F    D
IF_RUN                     = 0000006E    D
IF_XDELTA                  = 00000074    D
ILLEGAL                      000004C7 R  D  03
ILL_CMD                    = 00000001    D
INCOMPLETE                   000004DB R  D  03
INC_CMD                    = 00000002    D
INI$BRK                      ********   X   04
INITPCS                    = 0000008B    D
INIT_CONTEXT                 ********   X   04
KA820_CPU                    00000468 R  D  02
KA880_CPU                    00000462 R  D  02
KA884_CPU                    0000045C R  D  02
KB_CHECK                     ********   X   04
KB_CHECK_APT                 ********   X   04
LAST                       = 0000002D    D
LINE_COUNT                   ********   X   04
LOAD                       = 00000015    D
LONG                       = 00000068    D
LOOP                       = 0000004B    D
MAPFREE                      ********   X   04
MMOFF                      = 00000030    D
MMON                       = 0000002F    D
MP$GB_SETCPU                 00000472 RG D  02
MP$GL_CONDITION_FLAGS        ********W GX   00
MP$GL_LUN                    00000473 RG D  02
MP$GR_BOOTED_PROCESSORS      ********   X   04
MP$GT_DEVICE_NAME            00000477 RG D  02
NEXT                       = 00000059    D
NEXT_INST                  = 00000016    D
NOBASE                     = 00000040    D
NOTNUF                     = 00000018    D
NO_ALLOC                   = 0000007C    D
NULL                       = 00000000    D
NULL_DATA                  = 00000067    D
OCT                        = 0000006D    D
OFF                        = 00000000    D
ON                         = 00000001    D
OPER                       = 0000004C    D
OPTIONAL                   = 0000002A    D
PAGE                       = 0000005A    D
PASS                       = 00000034    D
PR$_MAPEN                  = 00000038    D
PREGN                      = 00000032    D
PRIMARY_NAME                 00000455 R  D  02
PROMPT                     = 0000004D    D
PSL$V_IS                   = 0000001A    D
PTABLE                       0000046E R  D  02
QA                         = 00000035    D
QACL                       = 00000053    D
QADEF                      = 00000057    D
QAEP                       = 00000052    D
QAMP                       = 00000056    D
QASL                       = 00000055    D
QATL                       = 00000054    D
QA_COMMON                    00000A9C R  D  04
QUICK                      = 0000004E    D
Q_ADAPTER                    ********   X   04
Q_GOOD_CMD                   0000044C R  D  02
RCLI                         00000002 R  D  04
REG12                      = 0000005E    D
REG13                      = 0000005F    D
REG14                      = 00000060    D
REG15                      = 00000061    D
REG16                      = 00000062    D
```

```
REGN                        = 0000005D    D
REGP                        = 00000063    D
REQUIRED                    = 0000002B    D
RUN                         = 00000019    D
SCRIPT                      = 0000001A    D
SCRIPT$CONT                   ********   X   04
SCRIPT$FLUSH                  ********   X   04
SCRIPT$OPEN                   ********   X   04
SEARCH                      = 0000004F    D
SECTION                     = 00000033    D
SELECT                      = 0000007B    D
SELECT_PARAMS                 00000A09 R  D  03
SET                         = 0000001B    D
SETCPU                      = 00000039    D
SETENFORCE                  = 00000036    D
SETLOAD                     = 0000007F    D
SETMEM                      = 0000008A    D
SET_CRD_DEBUG               = 00000020    D
SET_CRD_TRACE               = 0000001F    D
SET_DEFAULT_PARAMS            00000F00 R  D  03
SET_PARAMS                    00000D14 R  D  03
SHOMEMALL                   = 00000089    D
SHOMEMBUF                   = 00000087    D
SHOMEMDAT                   = 00000088    D
SHOMEMMAP                   = 00000086    D
SHOW                        = 0000001D    D
SHOWCPU                     = 00000024    D
SHOWENFORCE                 = 00000038    D
SHOWLOAD                    = 00000080    D
SHOWMEM                     = 00000085    D
SHOWMEMORYFLAGS               ********   X   04
SHOWMM                      = 00000031    D
SHOWSECTIONS                = 00000083    D
SHOWSEL                     = 0000007D    D
SHOWSTATUS                  = 00000082    D
SHOWSUP                     = 00000081    D
SHOWTESTS                   = 00000084    D
SHOW_CALLS                  = 00000023    D
SHOW_CRD_TRACE              = 00000022    D
SHOW_PARAMS                   00000A40 R  D  03
SIZ...                      = 00000001    D
SS$_ENDOFFILE                 ********   X   04
SS$_NORMAL                  = 00000001    D
START                       = 00000025    D
START_RUN_QUALS               000007CD R  D  03
SUBTEST                     = 0000002E    D
SUCCESS                     = 00000004    D
SUMMARY                     = 00000026    D
SYS$BINTIM                    ********  GX   04
SYS$EXIT                      ********  GX   04
TEST                        = 0000002C    D
TIME                        = 0000008E    D
TRACE                       = 00000050    D
TRUE                        = 00000001    D
T_CONT                        00000092 R  D  03
T_CRD_TRACE_OUTPUT            00000250 R  D  03
T_EMESSG1                     000000BD RG D  03
T_EMESSG2                     000000E2 R  D  03
T_EMESSG3                     00000113 R  D  03
T_EMESSG4                     00000181 R  D  03
T_EMESSG5                     000001A7 R  D  03
T_ILLCMD                      0000001B R  D  03
T_ILLEFN                      00000061 R  D  03
T_INCOMPLETE                  0000003A R  D  03
T_MP_CONT                     000000AB R  D  03
T_OFF                         000001E7 R  D  03
T_ON                          000001E4 R  D  03
T_PRGERR                      00000004 R  D  03
T_SHOWCPU                     00000288 R  D  03
T_SHOWENFORCE                 0000026D R  D  03
T_SHOWMM                      000001C8 R  D  03
T_USER_MODE_MENU_ERROR        00000216 R  D  03
ULTRIX_FLAGS                  ********   X   04
ULTRIX_ODS                  = 0000001C    D
ULTRIX_V_MODE                 ********   X   04
VERIFY                      = 00000051    D
VRABORT                       ********   X   04
VRCLRBRK                      ********   X   04
VRCLREF                       ********   X   04
VRCLRFLG                      ********   X   04
VRDEPOSIT                     ********   X   04
VREXAMINE                     ********   X   04
VRSETBASE                     ********   X   04
VRSETBRK                      ********   X   04
VRSETDFLT                     ********   X   04
VRSETEF                       ********   X   04
VRSETFLG                      ********   X   04
VRSETMEM                      ********   X   04
VRSETQA                       ********   X   04
VRSETWIDTH                    ********   X   04
VRSHOWBASE                    ********   X   04
VRSHOWBRK                     ********   X   04
VRSHOWDFLT                    ********   X   04
VRSHOWEF                      ********   X   04
VRSHOWFLG                     ********   X   04
VRSHOWQA                      ********   X   04
VRSHOWSECTIONS                ********   X   04
VRSHOWSTATUS                  ********   X   04
VRSHOWWIDTH                   ********   X   04
VRSHOW_CALLS                  ********   X   04
VRSTART                       ********   X   04
VRSUMMARY                     ********   X   04
VRSUPPORT                     ********   X   04
WIDE_FLAG                     ********   X   04
WIDTH                       = 0000005B    D
WORD                        = 00000069    D
XDELBPT                       ********W GX   04
XDELTA                      = 00000075    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000480 (1152.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000F6D (3949.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000DBC (3516.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=0000008E	2351 (1)	#-2204 (1)
$$N	=00000001	4012 (1)	#-2081 (1) 2207 (1) 2208 (1) 2209 (1)
			2210 (1) 2211 (1) 2212 (1) 2213 (1)
			2214 (1) 2215 (1) 2216 (1) 2217 (1)
			2218 (1) 2219 (1) 2220 (1) 2221 (1)
			2222 (1) 2223 (1) 2224 (1) 2225 (1)
			2226 (1) 2227 (1) 2228 (1) 2229 (1)
			2230 (1) 2231 (1) 2232 (1) 2233 (1)
			2234 (1) 2235 (1) 2236 (1) 2237 (1)
			2238 (1) 2239 (1) 2240 (1) 2241 (1)
			2242 (1) 2243 (1) 2244 (1) 2245 (1)
			2246 (1) 2247 (1) 2248 (1) 2249 (1)
			2250 (1) 2251 (1) 2252 (1) 2253 (1)
			2254 (1) 2255 (1) 2256 (1) 2257 (1)
			2258 (1) 2259 (1) 2260 (1) 2261 (1)
			2262 (1) 2263 (1) 2264 (1) 2265 (1)
			2266 (1) 2267 (1) 2268 (1) 2269 (1)
			2270 (1) 2271 (1) 2272 (1) 2273 (1)
			2274 (1) 2275 (1) 2276 (1) 2277 (1)
			2278 (1) 2279 (1) 2280 (1) 2281 (1)
			2282 (1) 2283 (1) 2284 (1) 2285 (1)
			2286 (1) 2287 (1) 2288 (1) 2289 (1)
			2290 (1) 2291 (1) 2292 (1) 2293 (1)
			2294 (1) 2295 (1) 2296 (1) 2297 (1)
			2298 (1) 2299 (1) 2300 (1) 2301 (1)
			2302 (1) 2303 (1) 2304 (1) 2305 (1)
			2306 (1) 2307 (1) 2308 (1) 2309 (1)
			2310 (1) 2311 (1) 2312 (1) 2313 (1)
			2314 (1) 2315 (1) 2316 (1) 2317 (1)
			2318 (1) 2319 (1) 2320 (1) 2321 (1)
			2322 (1) 2323 (1) 2324 (1) 2325 (1)
			2326 (1) 2327 (1) 2328 (1) 2329 (1)
			2330 (1) 2331 (1) 2332 (1) 2333 (1)
			2334 (1) 2335 (1) 2336 (1) 2337 (1)
			2338 (1) 2339 (1) 2340 (1) 2341 (1)
			2342 (1) 2343 (1) 2344 (1) 2345 (1)
			2346 (1) 2347 (1) 2348 (1) 2349 (1)
			2351 (1) #-2485 (1) 2490 (1) #-2816 (1)
			#-2912 (1) #-2976 (1) 3158 (1) 3343 (1)
			#-4012 (1)
$ER	=00000002	2353 (1)	#-2353 (1)
$MODULE	00000000-R	630 (1)	2353 (1)
ABORT	=00000006	2213 (1)	708 (1)
ACT_ABORT	000002C3-R	2436 (1)	2213 (1)
ACT_ADAPTER	00000C2B-R	3918 (1)	2328 (1)
ACT_ADDRESS	000008FB-R	3215 (1)	2268 (1) #-3670 (1)
ACT_ADVANCE	00000B39-R	3686 (1)	2308 (1)
ACT_ALL	00000946-R	3268 (1)	2272 (1)
ACT_ASCII	00000B01-R	3627 (1)	2299 (1)
ACT_ASK_PROMPT	000009EA-R	3364 (1)	2273 (1)

ACT_ATTACH	000002B2-R	2427	(1)	2214	(1)						
ACT_BACKUP	00000B34-R	3677	(1)	2307	(1)						
ACT_BASE	000008E5-R	3202	(1)	2267	(1)						
ACT_BASE_QUAL	0000092F-R	3246	(1)	2270	(1)						
ACT_BEGINADAPTER	00000C1D-R	3910	(1)	2327	(1)						
ACT_BEGINTIME	000007D3-R	3074	(1)	2348	(1)						
ACT_BELL	000009F0-R	3372	(1)	2275	(1)						
ACT_BINARY	000009F6-R	3380	(1)	2276	(1)						
ACT_BOOT	000002CC-R	2444	(1)	2215	(1)						
ACT_BREAK	00000908-R	3224	(1)	2269	(1)						
ACT_BRIEF	00000BE8-R	3890	(1)	2326	(1)						
ACT_BYTE	00000B56-R	3729	(1)	2313	(1)						
ACT_CLEAR	000002FF-R	2465	(1)	2216	(1)						
ACT_CLEARENFORCE	00000868-R	3139	(1)	2262	(1)						
ACT_CLEAR_CRD_DEBUG	000009E2-R	3354	(1)	2240	(1)						
ACT_CLEAR_CRD_TRACE	0000099D-R	3327	(1)	2237	(1)	#-3357	(1)				
ACT_CONT	0000035B-R	2493	(1)	2646	(1)						
ACT_CONTEXT	00000281-R	2385	(1)	2210	(1)	#-2370	(1)	#-2371	(1)	#-2378	(1)
ACT_CONTINUE	0000030C-R	2475	(1)	2217	(1)						
ACT_CRD	000006AC-R	2903	(1)	2246	(1)						
ACT_DATA	00000B3E-R	3694	(1)	2309	(1)						
ACT_DEATTACH	0000035C-R	2500	(1)	2218	(1)						
ACT_DEBUG	00000365-R	2509	(1)	2219	(1)						
ACT_DEBUG_SWITCH	000007A4-R	3020	(1)	2220	(1)						
ACT_DEC	00000B60-R	3745	(1)	2315	(1)						
ACT_DEFAULT	00000967-R	3285	(1)	2274	(1)						
ACT_DEFAULT_DBG	0000036E-R	2513	(1)	2221	(1)						
ACT_DEPOSIT	00000388-R	2523	(1)	2222	(1)						
ACT_DESELECT	00000395-R	2533	(1)	2329	(1)						
ACT_DEVICE	00000BD2-R	3878	(1)	2325	(1)						
ACT_DIRECTORY	0000039E-R	2541	(1)	2223	(1)						
ACT_DIR_WIDE	000003AD-R	2547	(1)	2347	(1)						
ACT_DO_CMD	00000C58-R	3943	(1)	2333	(1)						
ACT_DUMP	000003B5-R	2555	(1)	2224	(1)						
ACT_EFN	00000ADC-R	3600	(1)	2295	(1)						
ACT_ENUF	000002AD-R	2418	(1)	2230	(1)						
ACT_EVENT	000008A7-R	3166	(1)	2265	(1)						
ACT_EXAMINE	000003E0-R	2573	(1)	2225	(1)						
ACT_EXIT	000003ED-R	2583	(1)	2226	(1)						
ACT_FAILURE	0000029F-R	2403	(1)	2212	(1)						
ACT_FILEND	00000793-R	2994	(1)	2248	(1)						
ACT_FILESPEC	0000078B-R	2986	(1)	2247	(1)						
ACT_FLAGS	000008D0-R	3186	(1)	2266	(1)						
ACT_HALT	000009FC-R	3388	(1)	2277	(1)						
ACT_HELP	000003F6-R	2593	(1)	2227	(1)						
ACT_HEX	00000B5B-R	3737	(1)	2314	(1)						
ACT_IE1	00000A02-R	3396	(1)	2278	(1)						
ACT_IE2	00000A08-R	3404	(1)	2279	(1)						
ACT_IE3	00000A0E-R	3412	(1)	2280	(1)						
ACT_IES	00000A14-R	3420	(1)	2281	(1)						
ACT_IFNOTUSER	00000BA6-R	3844	(1)	2322	(1)						
ACT_IF_ADAPTER	00000B8B-R	3813	(1)	2320	(1)						
ACT_IF_FILE_SPEC	00000B81-R	3799	(1)	2319	(1)						
ACT_IF_REG_OR_ADR	00000B98-R	3827	(1)	2321	(1)						
ACT_IF_REQUIRED	00000B78-R	3785	(1)	2318	(1)						
ACT_IF_RUN	00000B6A-R	3769	(1)	2317	(1)						
ACT_IF_XDELTA	00000BB3-R	3858	(1)	2323	(1)						

ACT_ILL_CMD	00000277-R	2368	(1)	2208	(1)			
ACT_INC_CMD	0000027D-R	2376	(1)	2209	(1)			
ACT_INITPCS	00000437-R	2612	(1)	2346	(1)			
ACT_LAST	000007CE-R	3066	(1)	2252	(1)			
ACT_LOAD	0000045F-R	2629	(1)	2228	(1)			
ACT_LONG	00000B4C-R	3713	(1)	2311	(1)			
ACT_LOOP	00000A1A-R	3430	(1)	2282	(1)			
ACT_MMOFF	00000CA2-R	3987	(1)	2255	(1)			
ACT_MMON	00000C73-R	3968	(1)	2254	(1)			
ACT_NEXT	00000AFC-R	3620	(1)	2296	(1)			
ACT_NEXT_INST	00000472-R	2640	(1)	2229	(1)			
ACT_NOBASE	0000093B-R	3257	(1)	2271	(1)			
ACT_NOTNUF	000002A8-R	2410	(1)	2231	(1)			
ACT_NO_ALLOC	000004D8-R	2680	(1)	2331	(1)			
ACT_NULL	00000276-R	2362	(1)	2207	(1)			
ACT_NULL_DATA	00000B43-R	3703	(1)	2310	(1)			
ACT_OCT	00000B65-R	3753	(1)	2316	(1)			
ACT_OPER	00000A20-R	3438	(1)	2283	(1)			
ACT_OPTIONAL	0000079A-R	3002	(1)	2249	(1)			
ACT_PAGE	00000A44-R	3490	(1)	2297	(1)			
ACT_PASS	000007B0-R	3028	(1)	2259	(1)			
ACT_PREGN	00000B2D-R	3668	(1)	2257	(1)			
ACT_PROMPT	00000A26-R	3446	(1)	2284	(1)			
ACT_QA	000007B5-R	3035	(1)	2260	(1)			
ACT_QACL	00000A86-R	3532	(1)	2290	(1)			
ACT_QADEF	00000ABA-R	3571	(1)	2294	(1)			
ACT_QAEP	00000A80-R	3527	(1)	2289	(1)			
ACT_QAMP	00000A98-R	3547	(1)	2293	(1)			
ACT_QASL	00000A92-R	3542	(1)	2292	(1)			
ACT_QATL	00000A8C-R	3537	(1)	2291	(1)			
ACT_QUICK	00000A2C-R	3454	(1)	2285	(1)			
ACT_REG12	00000B0B-R	3642	(1)	2301	(1)			
ACT_REG13	00000B10-R	3645	(1)	2302	(1)			
ACT_REG14	00000B15-R	3648	(1)	2303	(1)			
ACT_REG15	00000B1A-R	3651	(1)	2304	(1)			
ACT_REG16	00000B20-R	3655	(1)	2305	(1)			
ACT_REGN	00000B06-R	3637	(1)	2300	(1)			
ACT_REGNUF	00000B28-R	3660	(1)	#-3659	(1)			
ACT_REGP	00000B24-R	3657	(1)	2306	(1)	#-3653	(1)	
ACT_REQUIRED	0000079F-R	3010	(1)	2250	(1)			
ACT_RUN	0000048F-R	2652	(1)	2232	(1)			
ACT_SCRIPT	0000069A-R	2884	(1)	2233	(1)			
ACT_SEARCH	00000A32-R	3464	(1)	2286	(1)			
ACT_SECTION	000007BA-R	3043	(1)	2258	(1)			
ACT_SELECT	000004C7-R	2672	(1)	2330	(1)			
ACT_SET	000004E1-R	2688	(1)	2234	(1)			
ACT_SETCPU	000004EE-R	2700	(1)	2264	(1)			
ACT_SETENFORCE	00000864-R	3136	(1)	2261	(1)			
ACT_SETLOAD	00000C61-R	3952	(1)	2334	(1)			
ACT_SETMEM	00000D78-R	4085	(1)	2345	(1)			
ACT_SET_CRD_DEBUG	000009D9-R	3349	(1)	2239	(1)			
ACT_SET_CRD_TRACE	00000993-R	3321	(1)	2238	(1)	#-3352	(1)	
ACT_SHOMEMALL	00000D6C-R	4075	(1)	2344	(1)			
ACT_SHOMEMBUF	00000D62-R	4069	(1)	2342	(1)			
ACT_SHOMEMDAT	00000D67-R	4072	(1)	2343	(1)			
ACT_SHOMEMMAP	00000D5E-R	4066	(1)	2341	(1)			
ACT_SHOW	0000065D-R	2858	(1)	2236	(1)			

```
ACT_SHOWCPU                     000005EE-R    2809    (1)       2243    (1)
ACT_SHOWENFORCE                 00000876-R    3150    (1)       2263    (1)
ACT_SHOWLOAD                    00000C6A-R    3959    (1)       2335    (1)
ACT_SHOWMEM                     00000D55-R    4062    (1)       2340    (1)
ACT_SHOWMM                      00000CD0-R    4005    (1)       2256    (1)
ACT_SHOWSECTIONS                00000D43-R    4050    (1)       2338    (1)
ACT_SHOWSEL                     00000C42-R    3931    (1)       2332    (1)
ACT_SHOWSTATUS                  00000D31-R    4036    (1)       2337    (1)
ACT_SHOWSUP                     00000D3A-R    4043    (1)       2336    (1)
ACT_SHOWTESTS                   00000D4C-R    4055    (1)       2339    (1)
ACT_SHOW_CALLS                  00000986-R    3309    (1)       2242    (1)
ACT_SHOW_CRD_TRACE              000009A7-R    3333    (1)       2241    (1)
ACT_START                       00000662-R    2866    (1)       2244    (1)
ACT_SUBTEST                     000007C4-R    3052    (1)       2253    (1)
ACT_SUCCESS                     00000296-R    2396    (1)       2211    (1)
ACT_SUMMARY                     000006A3-R    2892    (1)       2245    (1)
ACT_TEST                        000007C9-R    3059    (1)       2251    (1)
ACT_TIME                        000007DD-R    3083    (1)       2349    (1)
ACT_TRACE                       00000A38-R    3472    (1)       2287    (1)
ACT_ULTRIX_ODS                  00000628-R    2835    (1)       2235    (1)
ACT_VERIFY                      00000A3E-R    3480    (1)       2288    (1)
ACT_WIDTH                       00000A62-R    3507    (1)       2298    (1)
ACT_WORD                        00000B51-R    3721    (1)       2312    (1)
ACT_XDELTA                      00000BC8-R    3866    (1)       2324    (1)
ADAPTER                        =00000079      2328    (1)       1381    (1)      1456    (1)      1477    (1)      1492    (1)
ADDRESS                        =0000003D      2268    (1)       1266    (1)      1277    (1)      1747    (1)      1810    (1)
                                                                 1818    (1)
ADVANCE                        =00000065      2308    (1)       1001    (1)      1023    (1)      939     (1)      972     (1)
ALL                            =00000041      2272    (1)       1040    (1)      1118    (1)      1743    (1)
AP$GK_PROMPT_WAIT              =0FFFFFFF      587     (1)
AP$GW_DATA                      00000000-XR                      4097    (1)
AP$M_AP_NODE_ID                =000000F0      587     (1)
AP$M_AP_RUNNING                =00000100      587     (1)
AP$M_BUFFER_FULL               =00000800      587     (1)
AP$M_DATA_READY                =00000200      587     (1)
AP$M_PP_NODE_ID                =0000000F      587     (1)
AP$M_REQUEST                   =00000400      587     (1)
AP$S_AP_NODE_ID                =00000004      587     (1)
AP$S_PP_NODE_ID                =00000004      587     (1)
AP$V_AP_NODE_ID                =00000004      587     (1)
AP$V_AP_RUNNING                =00000008      587     (1)     #-4097    (1)
AP$V_BUFFER_FULL               =0000000B      587     (1)
AP$V_DATA_READY                =00000009      587     (1)
AP$V_PP_NODE_ID                =00000000      587     (1)
AP$V_REQUEST                   =0000000A      587     (1)
AP$_REQ_CONSRX                 =00000000      587     (1)
AP$_REQ_RXCDVEC                =00000001      587     (1)
APT_COM                         00000000-XR                      2075    (1)      2076    (1)
APT_MSG                         00000000-XR                      2071    (1)
ASCII                          =0000005C      2299    (1)       1407    (1)
ASK_PROMPT                     =00000042      2273    (1)       1043    (1)
ATTACH                         =00000007      2214    (1)       713     (1)
BACKUP                         =00000064      2307    (1)       1279    (1)      1746    (1)      1752    (1)      1822    (1)
                                                                 1873    (1)      1874    (1)      1904    (1)      1938    (1)
BAD_LIST                        000004DF-R    984     (1)       1050    (1)      1057    (1)      1067    (1)      1090    (1)
                                                                 1121    (1)      1995    (1)
BAD_QUAL                        000004FB-R    1005    (1)       1318    (1)      1325    (1)      1368    (1)      1385    (1)
```

				1440	(1)	1453	(1)	1475	(1)	1496	(1)		
				1626	(1)	1682	(1)	1761	(1)	1762	(1)		
				1763	(1)	1842	(1)	1843	(1)	1844	(1)		
				765	(1)								
BASE	=0000003C	2267	(1)	1540	(1)	1808	(1)						
BASE_QUAL	=0000003F	2270	(1)	1365	(1)	1413	(1)						
BEGIN	00000000-XR			2606	(1)								
BEGINADAPTER	=00000078	2327	(1)	1380	(1)	1455	(1)	1476	(1)	1491	(1)		
BEGINTIME	=0000008D	2348	(1)	1329	(1)								
BELL	=00000044	2275	(1)	1047	(1)								
BINARY	=00000045	2276	(1)	1050	(1)								
BIN_TIME	00000000-XR			#-2654	(1)	#-2868	(1)	3104	(1)	3118	(1)		
BIT...	=0000000C	587	(1)	575	(1)	578	(1)	580	(1)	584	(1)		
				585	(1)	586	(1)	587	(1)				
BOOT	=00000008	2215	(1)	719	(1)								
BREAK	=0000003E	2269	(1)	1545	(1)	1738	(1)	1815	(1)				
BREAK_QUALS	00000893-R	1360	(1)	1366	(1)	1369	(1)	1739	(1)	1749	(1)		
				1816	(1)	1819	(1)						
BRIEF	=00000077	2326	(1)	1385	(1)	1682	(1)						
BYTE	=0000006A	2313	(1)	1417	(1)	1970	(1)						
B_SUCCESS	00000454-R	603	(1)	2397	(1)	2404	(1)						
CLEAR	=00000009	2216	(1)	735	(1)								
CLEARENFORCE	=00000037	2262	(1)	1769	(1)								
CLEAR_CRD_DEBUG	=00000021	2240	(1)	1764	(1)								
CLEAR_CRD_TRACE	=0000001E	2237	(1)	1759	(1)								
CLEAR_PARAMS	00000C78-R	1731	(1)	736	(1)								
CLI	00000072-R	2086	(1)	#-2069	(1)								
CLI$K_ALNUM	=00000087	575	(1)	1381	(1)	1456	(1)	1461	(1)	1477	(1)		
				1492	(1)	1830	(1)						
CLI$K_ALPHA	=00000086	575	(1)										
CLI$K_BIF	=00000083	575	(1)	1107	(1)	1137	(1)	1166	(1)	1167	(1)		
				1340	(1)	1459	(1)	1466	(1)	1468	(1)		
				916	(1)								
CLI$K_BIFS	=00000094	575	(1)										
CLI$K_BR	=00000082	575	(1)	1001	(1)	1023	(1)	1041	(1)	1044	(1)		
				1048	(1)	1051	(1)	1054	(1)	1059	(1)		
				1062	(1)	1065	(1)	1068	(1)	1071	(1)		
				1074	(1)	1077	(1)	1080	(1)	1085	(1)		
				1088	(1)	1094	(1)	1101	(1)	1105	(1)		
				1108	(1)	1119	(1)	1123	(1)	1136	(1)		
				1146	(1)	1149	(1)	1153	(1)	1165	(1)		
				1178	(1)	1180	(1)	1182	(1)	1184	(1)		
				1186	(1)	1188	(1)	1190	(1)	1192	(1)		
				1194	(1)	1196	(1)	1198	(1)	1200	(1)		
				1202	(1)	1204	(1)	1213	(1)	1216	(1)		
				1218	(1)	1221	(1)	1224	(1)	1279	(1)		
				1293	(1)	1300	(1)	1305	(1)	1314	(1)		
				1321	(1)	1329	(1)	1342	(1)	1353	(1)		
				1366	(1)	1369	(1)	1383	(1)	1386	(1)		
				1408	(1)	1414	(1)	1417	(1)	1420	(1)		
				1423	(1)	1426	(1)	1430	(1)	1435	(1)		
				1438	(1)	1441	(1)	1455	(1)	1463	(1)		
				1467	(1)	1469	(1)	1480	(1)	1494	(1)		
				1497	(1)	1499	(1)	1541	(1)	1546	(1)		
				1552	(1)	1558	(1)	1564	(1)	1576	(1)		
				1582	(1)	1588	(1)	1595	(1)	1600	(1)		
				1605	(1)	1618	(1)	1621	(1)	1624	(1)		

```
                                                1627  (1)     1632  (1)     1637  (1)     1644  (1)
                                                1649  (1)     1654  (1)     1659  (1)     1664  (1)
                                                1669  (1)     1676  (1)     1683  (1)     1688  (1)
                                                1693  (1)     1698  (1)     1703  (1)     1711  (1)
                                                1714  (1)     1744  (1)     1746  (1)     1750  (1)
                                                1752  (1)     1759  (1)     1764  (1)     1770  (1)
                                                1776  (1)     1780  (1)     1811  (1)     1820  (1)
                                                1822  (1)     1829  (1)     1832  (1)     1840  (1)
                                                1845  (1)     1850  (1)     1856  (1)     1862  (1)
                                                1871  (1)     1873  (1)     1874  (1)     1882  (1)
                                                1891  (1)     1894  (1)     1902  (1)     1904  (1)
                                                1917  (1)     1920  (1)     1923  (1)     1926  (1)
                                                1929  (1)     1932  (1)     1936  (1)     1938  (1)
                                                1946  (1)     1950  (1)     1971  (1)     1976  (1)
                                                1981  (1)     1986  (1)     1991  (1)     1996  (1)
                                                2002  (1)     2006  (1)     709   (1)     722   (1)
                                                736   (1)     741   (1)     746   (1)     768   (1)
                                                775   (1)     777   (1)     784   (1)     791   (1)
                                                792   (1)     797   (1)     811   (1)     825   (1)
                                                834   (1)     849   (1)     856   (1)     863   (1)
                                                872   (1)     890   (1)     895   (1)     900   (1)
                                                906   (1)     911   (1)     918   (1)     925   (1)
                                                939   (1)     940   (1)     972   (1)
CLI$K_BUFSIZ        =00000100   579   (1)     #-2098  (1)     579   (1)
CLI$K_CALL          =00000092   575   (1)     1581  (1)     1707  (1)     1739  (1)     1749  (1)
                                                1816  (1)     1819  (1)     1870  (1)     767   (1)
                                                790   (1)     804   (1)     806   (1)     807   (1)
                                                810   (1)     830   (1)     832   (1)     833   (1)
                                                855   (1)     868   (1)     870   (1)     871   (1)
                                                905   (1)     924   (1)
CLI$K_COMMA         =0000008E   575   (1)     1093  (1)     1104  (1)     1122  (1)     1144  (1)
                                                1176  (1)     2001  (1)     994   (1)
CLI$K_DEC           =0000008A   575   (1)     1121  (1)     1163  (1)     1175  (1)     1177  (1)
                                                1266  (1)     1299  (1)     1320  (1)     1331  (1)
                                                1333  (1)     1335  (1)     1337  (1)     1339  (1)
                                                1350  (1)     1352  (1)     1434  (1)     1881  (1)
                                                1901  (1)     1935  (1)     1945  (1)     862   (1)
CLI$K_EOL           =00000091   575   (1)     1019  (1)     1212  (1)     1215  (1)     1220  (1)
                                                1223  (1)     841   (1)     936   (1)     964   (1)
                                                997   (1)
CLI$K_ERROR         =00000080   575   (1)     1002  (1)     1017  (1)     1020  (1)     1024  (1)
                                                941   (1)     965   (1)     973   (1)     981   (1)
                                                995   (1)     998   (1)
CLI$K_EXIT          =00000081   575   (1)     714   (1)     842   (1)     844   (1)     937   (1)
CLI$K_FILE          =00000095   575   (1)     1139  (1)     1143  (1)     1145  (1)     1152  (1)
                                                1157  (1)     1160  (1)     1172  (1)     1211  (1)
                                                1217  (1)     1222  (1)
CLI$K_HEX           =00000089   575   (1)
CLI$K_KEYWORD       =0000008C   575   (1)     1000  (1)     1001  (1)     1002  (1)     1016  (1)
                                                1017  (1)     1019  (1)     1020  (1)     1022  (1)
                                                1023  (1)     1024  (1)     1035  (1)     1036  (1)
                                                1039  (1)     1040  (1)     1041  (1)     1043  (1)
                                                1044  (1)     1046  (1)     1047  (1)     1048  (1)
                                                1050  (1)     1051  (1)     1053  (1)     1054  (1)
                                                1056  (1)     1057  (1)     1058  (1)     1059  (1)
                                                1061  (1)     1062  (1)     1064  (1)     1065  (1)
                                                1067  (1)     1068  (1)     1070  (1)     1071  (1)
```

1073	(1)	1074	(1)	1076	(1)	1077	(1)
1079	(1)	1080	(1)	1084	(1)	1085	(1)
1087	(1)	1088	(1)	1090	(1)	1093	(1)
1094	(1)	1100	(1)	1101	(1)	1102	(1)
1103	(1)	1104	(1)	1105	(1)	1107	(1)
1108	(1)	1114	(1)	1115	(1)	1116	(1)
1118	(1)	1119	(1)	1121	(1)	1122	(1)
1123	(1)	1136	(1)	1137	(1)	1139	(1)
1140	(1)	1142	(1)	1143	(1)	1144	(1)
1145	(1)	1146	(1)	1148	(1)	1149	(1)
1151	(1)	1152	(1)	1153	(1)	1155	(1)
1157	(1)	1159	(1)	1160	(1)	1162	(1)
1163	(1)	1165	(1)	1166	(1)	1167	(1)
1168	(1)	1172	(1)	1173	(1)	1174	(1)
1175	(1)	1176	(1)	1177	(1)	1178	(1)
1179	(1)	1180	(1)	1181	(1)	1182	(1)
1183	(1)	1184	(1)	1185	(1)	1186	(1)
1187	(1)	1188	(1)	1189	(1)	1190	(1)
1191	(1)	1192	(1)	1193	(1)	1194	(1)
1195	(1)	1196	(1)	1197	(1)	1198	(1)
1199	(1)	1200	(1)	1201	(1)	1202	(1)
1203	(1)	1204	(1)	1205	(1)	1206	(1)
1211	(1)	1212	(1)	1213	(1)	1214	(1)
1215	(1)	1216	(1)	1217	(1)	1218	(1)
1219	(1)	1220	(1)	1221	(1)	1222	(1)
1223	(1)	1224	(1)	1236	(1)	1237	(1)
1238	(1)	1243	(1)	1244	(1)	1245	(1)
1249	(1)	1250	(1)	1251	(1)	1255	(1)
1256	(1)	1260	(1)	1261	(1)	1265	(1)
1266	(1)	1267	(1)	1271	(1)	1272	(1)
1273	(1)	1277	(1)	1278	(1)	1279	(1)
1288	(1)	1292	(1)	1293	(1)	1297	(1)
1298	(1)	1299	(1)	1300	(1)	1304	(1)
1305	(1)	1309	(1)	1311	(1)	1312	(1)
1313	(1)	1314	(1)	1318	(1)	1319	(1)
1320	(1)	1321	(1)	1325	(1)	1327	(1)
1328	(1)	1329	(1)	1331	(1)	1332	(1)
1333	(1)	1334	(1)	1335	(1)	1336	(1)
1337	(1)	1338	(1)	1339	(1)	1340	(1)
1342	(1)	1348	(1)	1349	(1)	1350	(1)
1351	(1)	1352	(1)	1353	(1)	1355	(1)
1361	(1)	1363	(1)	1364	(1)	1365	(1)
1366	(1)	1368	(1)	1369	(1)	1371	(1)
1377	(1)	1379	(1)	1380	(1)	1381	(1)
1382	(1)	1383	(1)	1385	(1)	1386	(1)
1388	(1)	1405	(1)	1407	(1)	1408	(1)
1410	(1)	1411	(1)	1412	(1)	1413	(1)
1414	(1)	1416	(1)	1417	(1)	1419	(1)
1420	(1)	1422	(1)	1423	(1)	1425	(1)
1426	(1)	1428	(1)	1429	(1)	1430	(1)
1432	(1)	1433	(1)	1434	(1)	1435	(1)
1437	(1)	1438	(1)	1440	(1)	1441	(1)
1443	(1)	1451	(1)	1453	(1)	1454	(1)
1455	(1)	1456	(1)	1457	(1)	1459	(1)
1460	(1)	1461	(1)	1462	(1)	1463	(1)
1465	(1)	1466	(1)	1467	(1)	1468	(1)
1469	(1)	1474	(1)	1475	(1)	1476	(1)

1477 (1) 1478 (1) 1480 (1) 1488 (1)
1490 (1) 1491 (1) 1492 (1) 1493 (1)
1494 (1) 1496 (1) 1497 (1) 1499 (1)
1535 (1) 1536 (1) 1540 (1) 1541 (1)
1545 (1) 1546 (1) 1550 (1) 1551 (1)
1552 (1) 1557 (1) 1558 (1) 1563 (1)
1564 (1) 1570 (1) 1571 (1) 1575 (1)
1576 (1) 1580 (1) 1581 (1) 1582 (1)
1586 (1) 1587 (1) 1588 (1) 1592 (1)
1593 (1) 1594 (1) 1595 (1) 1599 (1)
1600 (1) 1604 (1) 1605 (1) 1610 (1)
1614 (1) 1615 (1) 1617 (1) 1618 (1)
1620 (1) 1621 (1) 1623 (1) 1624 (1)
1626 (1) 1627 (1) 1631 (1) 1632 (1)
1636 (1) 1637 (1) 1641 (1) 1642 (1)
1643 (1) 1644 (1) 1648 (1) 1649 (1)
1653 (1) 1654 (1) 1658 (1) 1659 (1)
1663 (1) 1664 (1) 1668 (1) 1669 (1)
1673 (1) 1674 (1) 1675 (1) 1676 (1)
1680 (1) 1681 (1) 1682 (1) 1683 (1)
1687 (1) 1688 (1) 1692 (1) 1693 (1)
1697 (1) 1698 (1) 1702 (1) 1703 (1)
1707 (1) 1708 (1) 1710 (1) 1711 (1)
1713 (1) 1714 (1) 1732 (1) 1733 (1)
1738 (1) 1739 (1) 1740 (1) 1742 (1)
1743 (1) 1744 (1) 1746 (1) 1747 (1)
1749 (1) 1750 (1) 1752 (1) 1756 (1)
1757 (1) 1758 (1) 1759 (1) 1761 (1)
1762 (1) 1763 (1) 1764 (1) 1768 (1)
1769 (1) 1770 (1) 1775 (1) 1776 (1)
1780 (1) 1803 (1) 1804 (1) 1808 (1)
1809 (1) 1810 (1) 1811 (1) 1815 (1)
1816 (1) 1817 (1) 1818 (1) 1819 (1)
1820 (1) 1822 (1) 1826 (1) 1827 (1)
1828 (1) 1829 (1) 1830 (1) 1831 (1)
1832 (1) 1837 (1) 1838 (1) 1839 (1)
1840 (1) 1842 (1) 1843 (1) 1844 (1)
1845 (1) 1849 (1) 1850 (1) 1854 (1)
1855 (1) 1856 (1) 1861 (1) 1862 (1)
1866 (1) 1867 (1) 1868 (1) 1869 (1)
1870 (1) 1871 (1) 1873 (1) 1874 (1)
1878 (1) 1879 (1) 1880 (1) 1881 (1)
1882 (1) 1887 (1) 1888 (1) 1889 (1)
1890 (1) 1891 (1) 1893 (1) 1894 (1)
1898 (1) 1899 (1) 1900 (1) 1901 (1)
1902 (1) 1904 (1) 1913 (1) 1914 (1)
1916 (1) 1917 (1) 1919 (1) 1920 (1)
1922 (1) 1923 (1) 1925 (1) 1926 (1)
1928 (1) 1929 (1) 1931 (1) 1932 (1)
1934 (1) 1935 (1) 1936 (1) 1938 (1)
1943 (1) 1944 (1) 1945 (1) 1946 (1)
1950 (1) 1965 (1) 1970 (1) 1971 (1)
1975 (1) 1976 (1) 1980 (1) 1981 (1)
1985 (1) 1986 (1) 1990 (1) 1991 (1)
1995 (1) 1996 (1) 2001 (1) 2002 (1)
2006 (1) 702 (1) 704 (1) 708 (1)
709 (1) 713 (1) 714 (1) 719 (1)

```
                                                  720    (1)    721    (1)    722    (1)    731    (1)
                                                  735    (1)    736    (1)    740    (1)    741    (1)
                                                  745    (1)    746    (1)    759    (1)    763    (1)
                                                  764    (1)    765    (1)    766    (1)    767    (1)
                                                  768    (1)    772    (1)    773    (1)    774    (1)
                                                  775    (1)    776    (1)    777    (1)    779    (1)
                                                  783    (1)    784    (1)    788    (1)    789    (1)
                                                  790    (1)    791    (1)    792    (1)    796    (1)
                                                  797    (1)    803    (1)    804    (1)    805    (1)
                                                  806    (1)    807    (1)    808    (1)    809    (1)
                                                  810    (1)    811    (1)    818    (1)    819    (1)
                                                  824    (1)    825    (1)    829    (1)    830    (1)
                                                  831    (1)    832    (1)    833    (1)    834    (1)
                                                  840    (1)    841    (1)    842    (1)    843    (1)
                                                  844    (1)    848    (1)    849    (1)    853    (1)
                                                  854    (1)    855    (1)    856    (1)    860    (1)
                                                  861    (1)    862    (1)    863    (1)    867    (1)
                                                  868    (1)    869    (1)    870    (1)    871    (1)
                                                  872    (1)    884    (1)    888    (1)    889    (1)
                                                  890    (1)    894    (1)    895    (1)    899    (1)
                                                  900    (1)    904    (1)    905    (1)    906    (1)
                                                  910    (1)    911    (1)    916    (1)    917    (1)
                                                  918    (1)    922    (1)    923    (1)    924    (1)
                                                  925    (1)    935    (1)    936    (1)    937    (1)
                                                  939    (1)    940    (1)    941    (1)    964    (1)
                                                  965    (1)    971    (1)    972    (1)    973    (1)
                                                  981    (1)    994    (1)    995    (1)    997    (1)
                                                  998    (1)
CLI$K_NUM                  =00000085    575    (1)   1260    (1)   1277    (1)   1365    (1)   1413    (1)
                                                  1747    (1)   1810    (1)   1818    (1)    721    (1)
                                                  774    (1)    776    (1)    809    (1)
CLI$K_OCT                  =00000088    575    (1)
CLI$K_RETURN               =00000093    575    (1)   1168    (1)   1238    (1)   1245    (1)   1251    (1)
                                                  1256    (1)   1261    (1)   1267    (1)   1273    (1)
                                                  1278    (1)   1355    (1)   1371    (1)   1388    (1)
                                                  1443    (1)
CLI$K_SIZE                  0000044C    579    (1)   2054    (1)   2055    (1)   2058    (1)   2061    (1)
                                                 #-2064    (1)    598    (1)
CLI$K_SLASH                =00000090    575    (1)   1016    (1)   1288    (1)   1361    (1)   1377    (1)
                                                  1405    (1)   1451    (1)   1474    (1)   1488    (1)
                                                  1615    (1)   1681    (1)   1757    (1)   1762    (1)
                                                  1838    (1)   1843    (1)    764    (1)
CLI$K_SPACE                =00000084    575    (1)   1000    (1)   1022    (1)   1036    (1)   1100    (1)
                                                  1114    (1)   1116    (1)   1460    (1)   1535    (1)
                                                  1593    (1)   1708    (1)   1732    (1)   1740    (1)
                                                  1803    (1)   1809    (1)   1817    (1)   1828    (1)
                                                  1869    (1)   1880    (1)   1888    (1)   1900    (1)
                                                  1934    (1)   1944    (1)   1965    (1)    702    (1)
                                                  720    (1)    766    (1)    773    (1)    805    (1)
                                                  808    (1)    831    (1)    843    (1)    854    (1)
                                                  861    (1)    869    (1)    923    (1)    935    (1)
                                                  971    (1)
CLI$K_STRING               =0000008B    575    (1)   1000    (1)   1001    (1)   1002    (1)   1016    (1)
                                                  1017    (1)   1019    (1)   1020    (1)   1022    (1)
                                                  1023    (1)   1024    (1)   1035    (1)   1036    (1)
                                                  1039    (1)   1040    (1)   1041    (1)   1043    (1)
                                                  1044    (1)   1046    (1)   1047    (1)   1048    (1)
```

1050 (1) 1051 (1) 1053 (1) 1054 (1)
1056 (1) 1057 (1) 1058 (1) 1059 (1)
1061 (1) 1062 (1) 1064 (1) 1065 (1)
1067 (1) 1068 (1) 1070 (1) 1071 (1)
1073 (1) 1074 (1) 1076 (1) 1077 (1)
1079 (1) 1080 (1) 1084 (1) 1085 (1)
1087 (1) 1088 (1) 1090 (1) 1093 (1)
1094 (1) 1100 (1) 1101 (1) 1102 (1)
1103 (1) 1104 (1) 1105 (1) 1107 (1)
1108 (1) 1114 (1) 1115 (1) 1116 (1)
1118 (1) 1119 (1) 1121 (1) 1122 (1)
1123 (1) 1136 (1) 1137 (1) 1139 (1)
1140 (1) 1142 (1) 1143 (1) 1144 (1)
1145 (1) 1146 (1) 1148 (1) 1149 (1)
1151 (1) 1152 (1) 1153 (1) 1155 (1)
1157 (1) 1159 (1) 1160 (1) 1162 (1)
1163 (1) 1165 (1) 1166 (1) 1167 (1)
1168 (1) 1172 (1) 1173 (1) 1174 (1)
1175 (1) 1176 (1) 1177 (1) 1178 (1)
1179 (1) 1180 (1) 1181 (1) 1182 (1)
1183 (1) 1184 (1) 1185 (1) 1186 (1)
1187 (1) 1188 (1) 1189 (1) 1190 (1)
1191 (1) 1192 (1) 1193 (1) 1194 (1)
1195 (1) 1196 (1) 1197 (1) 1198 (1)
1199 (1) 1200 (1) 1201 (1) 1202 (1)
1203 (1) 1204 (1) 1205 (1) 1206 (1)
1211 (1) 1212 (1) 1213 (1) 1214 (1)
1215 (1) 1216 (1) 1217 (1) 1218 (1)
1219 (1) 1220 (1) 1221 (1) 1222 (1)
1223 (1) 1224 (1) 1236 (1) 1237 (1)
1238 (1) 1243 (1) 1244 (1) 1245 (1)
1249 (1) 1250 (1) 1251 (1) 1255 (1)
1256 (1) 1260 (1) 1261 (1) 1265 (1)
1266 (1) 1267 (1) 1271 (1) 1272 (1)
1273 (1) 1277 (1) 1278 (1) 1279 (1)
1288 (1) 1292 (1) 1293 (1) 1297 (1)
1298 (1) 1299 (1) 1300 (1) 1304 (1)
1305 (1) 1309 (1) 1311 (1) 1312 (1)
1313 (1) 1314 (1) 1318 (1) 1319 (1)
1320 (1) 1321 (1) 1325 (1) 1327 (1)
1328 (1) 1329 (1) 1331 (1) 1332 (1)
1333 (1) 1334 (1) 1335 (1) 1336 (1)
1337 (1) 1338 (1) 1339 (1) 1340 (1)
1342 (1) 1348 (1) 1349 (1) 1350 (1)
1351 (1) 1352 (1) 1353 (1) 1355 (1)
1361 (1) 1363 (1) 1364 (1) 1365 (1)
1366 (1) 1368 (1) 1369 (1) 1371 (1)
1377 (1) 1379 (1) 1380 (1) 1381 (1)
1382 (1) 1383 (1) 1385 (1) 1386 (1)
1388 (1) 1405 (1) 1407 (1) 1408 (1)
1410 (1) 1411 (1) 1412 (1) 1413 (1)
1414 (1) 1416 (1) 1417 (1) 1419 (1)
1420 (1) 1422 (1) 1423 (1) 1425 (1)
1426 (1) 1428 (1) 1429 (1) 1430 (1)
1432 (1) 1433 (1) 1434 (1) 1435 (1)
1437 (1) 1438 (1) 1440 (1) 1441 (1)
1443 (1) 1451 (1) 1453 (1) 1454 (1)

1455 (1) 1456 (1) 1457 (1) 1459 (1)
1460 (1) 1461 (1) 1462 (1) 1463 (1)
1465 (1) 1466 (1) 1467 (1) 1468 (1)
1469 (1) 1474 (1) 1475 (1) 1476 (1)
1477 (1) 1478 (1) 1480 (1) 1488 (1)
1490 (1) 1491 (1) 1492 (1) 1493 (1)
1494 (1) 1496 (1) 1497 (1) 1499 (1)
1535 (1) 1536 (1) 1540 (1) 1541 (1)
1545 (1) 1546 (1) 1550 (1) 1551 (1)
1552 (1) 1557 (1) 1558 (1) 1563 (1)
1564 (1) 1570 (1) 1571 (1) 1575 (1)
1576 (1) 1580 (1) 1581 (1) 1582 (1)
1586 (1) 1587 (1) 1588 (1) 1592 (1)
1593 (1) 1594 (1) 1595 (1) 1599 (1)
1600 (1) 1604 (1) 1605 (1) 1610 (1)
1614 (1) 1615 (1) 1617 (1) 1618 (1)
1620 (1) 1621 (1) 1623 (1) 1624 (1)
1626 (1) 1627 (1) 1631 (1) 1632 (1)
1636 (1) 1637 (1) 1641 (1) 1642 (1)
1643 (1) 1644 (1) 1648 (1) 1649 (1)
1653 (1) 1654 (1) 1658 (1) 1659 (1)
1663 (1) 1664 (1) 1668 (1) 1669 (1)
1673 (1) 1674 (1) 1675 (1) 1676 (1)
1680 (1) 1681 (1) 1682 (1) 1683 (1)
1687 (1) 1688 (1) 1692 (1) 1693 (1)
1697 (1) 1698 (1) 1702 (1) 1703 (1)
1707 (1) 1708 (1) 1710 (1) 1711 (1)
1713 (1) 1714 (1) 1732 (1) 1733 (1)
1738 (1) 1739 (1) 1740 (1) 1742 (1)
1743 (1) 1744 (1) 1746 (1) 1747 (1)
1749 (1) 1750 (1) 1752 (1) 1756 (1)
1757 (1) 1758 (1) 1759 (1) 1761 (1)
1762 (1) 1763 (1) 1764 (1) 1768 (1)
1769 (1) 1770 (1) 1775 (1) 1776 (1)
1780 (1) 1803 (1) 1804 (1) 1808 (1)
1809 (1) 1810 (1) 1811 (1) 1815 (1)
1816 (1) 1817 (1) 1818 (1) 1819 (1)
1820 (1) 1822 (1) 1826 (1) 1827 (1)
1828 (1) 1829 (1) 1830 (1) 1831 (1)
1832 (1) 1837 (1) 1838 (1) 1839 (1)
1840 (1) 1842 (1) 1843 (1) 1844 (1)
1845 (1) 1849 (1) 1850 (1) 1854 (1)
1855 (1) 1856 (1) 1861 (1) 1862 (1)
1866 (1) 1867 (1) 1868 (1) 1869 (1)
1870 (1) 1871 (1) 1873 (1) 1874 (1)
1878 (1) 1879 (1) 1880 (1) 1881 (1)
1882 (1) 1887 (1) 1888 (1) 1889 (1)
1890 (1) 1891 (1) 1893 (1) 1894 (1)
1898 (1) 1899 (1) 1900 (1) 1901 (1)
1902 (1) 1904 (1) 1913 (1) 1914 (1)
1916 (1) 1917 (1) 1919 (1) 1920 (1)
1922 (1) 1923 (1) 1925 (1) 1926 (1)
1928 (1) 1929 (1) 1931 (1) 1932 (1)
1934 (1) 1935 (1) 1936 (1) 1938 (1)
1943 (1) 1944 (1) 1945 (1) 1946 (1)
1950 (1) 1965 (1) 1970 (1) 1971 (1)
1975 (1) 1976 (1) 1980 (1) 1981 (1)

				1985 (1) 1986 (1) 1990 (1) 1991 (1)
				1995 (1) 1996 (1) 2001 (1) 2002 (1)
				2006 (1) 702 (1) 704 (1) 708 (1)
				709 (1) 713 (1) 714 (1) 719 (1)
				720 (1) 721 (1) 722 (1) 731 (1)
				735 (1) 736 (1) 740 (1) 741 (1)
				745 (1) 746 (1) 759 (1) 763 (1)
				764 (1) 765 (1) 766 (1) 767 (1)
				768 (1) 772 (1) 773 (1) 774 (1)
				775 (1) 776 (1) 777 (1) 779 (1)
				783 (1) 784 (1) 788 (1) 789 (1)
				790 (1) 791 (1) 792 (1) 796 (1)
				797 (1) 803 (1) 804 (1) 805 (1)
				806 (1) 807 (1) 808 (1) 809 (1)
				810 (1) 811 (1) 818 (1) 819 (1)
				824 (1) 825 (1) 829 (1) 830 (1)
				831 (1) 832 (1) 833 (1) 834 (1)
				840 (1) 841 (1) 842 (1) 843 (1)
				844 (1) 848 (1) 849 (1) 853 (1)
				854 (1) 855 (1) 856 (1) 860 (1)
				861 (1) 862 (1) 863 (1) 867 (1)
				868 (1) 869 (1) 870 (1) 871 (1)
				872 (1) 884 (1) 888 (1) 889 (1)
				890 (1) 894 (1) 895 (1) 899 (1)
				900 (1) 904 (1) 905 (1) 906 (1)
				910 (1) 911 (1) 916 (1) 917 (1)
				918 (1) 922 (1) 923 (1) 924 (1)
				925 (1) 935 (1) 936 (1) 937 (1)
				939 (1) 940 (1) 941 (1) 964 (1)
				965 (1) 971 (1) 972 (1) 973 (1)
				981 (1) 994 (1) 995 (1) 997 (1)
				998 (1)
CLI$K_SYMBOL	=0000008D	575	(1)	1102 (1) 1313 (1)
CLI$K_VALUE	=0000008F	575	(1)	1298 (1) 1312 (1) 1319 (1) 1328 (1)
				1334 (1) 1336 (1) 1349 (1) 1351 (1)
				1364 (1) 1380 (1) 1412 (1) 1433 (1)
				1454 (1) 1476 (1) 1491 (1)
CLI$L_ADDRESS	00000018	579	(1)	#-2634 (1) #-3216 (1) #-3643 (1) #-3646 (1)
				#-3649 (1) #-3652 (1) #-3656 (1)
CLI$L_COMMAND	00000004	579	(1)	#-2075 (1) #-2170 (1) #-2430 (1) #-2438 (1)
				#-2445 (1) #-2467 (1) #-2478 (1) #-2502 (1)
				#-2510 (1) #-2525 (1) #-2534 (1) #-2543 (1)
				#-2556 (1) #-2575 (1) #-2585 (1) #-2613 (1)
				#-2631 (1) #-2646 (1) #-2661 (1) #-2673 (1)
				#-2690 (1) #-2810 (1) #-2839 (1) #-2875 (1)
				#-2886 (1) #-2894 (1) #-3143 (1) #-3160 (1)
				#-3172 (1) #-3177 (1) #-3180 (1) #-3194 (1)
				#-3206 (1) #-3209 (1) #-3230 (1) #-3235 (1)
				#-3238 (1) #-3270 (1) #-3276 (1) #-3279 (1)
				#-3292 (1) #-3296 (1) #-3311 (1) #-3346 (1)
				#-3496 (1) #-3499 (1) #-3501 (1) #-3513 (1)
				#-3516 (1) #-3518 (1) #-3559 (1) #-3563 (1)
				#-3566 (1) #-3584 (1) #-3589 (1) #-3592 (1)
				#-3868 (1) #-3881 (1) #-3884 (1) #-3894 (1)
				#-3896 (1) #-3899 (1) #-3901 (1) #-3934 (1)
				#-3937 (1) #-3944 (1) #-3953 (1) #-3960 (1)
				#-3969 (1) #-3988 (1) #-4006 (1) #-4037 (1)

```
                                                #-4044  (1)     #-4051  (1)     #-4056  (1)     #-4063  (1)
                                                #-4086  (1)
CLI$L_DATA                 0000001C     579 (1) #-2566  (1)     #-2642  (1)     #-3269  (1)     #-3312  (1)
                                                  3366  (1)       3374  (1)       3382  (1)       3390  (1)
                                                  3398  (1)       3406  (1)       3414  (1)       3422  (1)
                                                  3432  (1)       3440  (1)       3448  (1)       3456  (1)
                                                  3466  (1)       3474  (1)       3482  (1)       3605  (1)
                                                #-3695  (1)     #-3704  (1)
CLI$L_FLAGS                00000000     579 (1)   2142  (1)       2370  (1)       2378  (1)       2412  (1)
                                                  2420  (1)       2469  (1)       2527  (1)       2577  (1)
                                                #-2633  (1)       2659  (1)       2692  (1)       2860  (1)
                                                  2873  (1)       3004  (1)       3012  (1)       3037  (1)
                                                  3168  (1)       3170  (1)       3175  (1)       3188  (1)
                                                  3190  (1)       3192  (1)       3204  (1)       3218  (1)
                                                  3226  (1)       3228  (1)       3233  (1)       3274  (1)
                                                  3287  (1)       3290  (1)       3300  (1)       3492  (1)
                                                  3494  (1)       3509  (1)       3511  (1)       3529  (1)
                                                  3534  (1)       3539  (1)       3544  (1)       3549  (1)
                                                  3554  (1)       3556  (1)       3573  (1)       3579  (1)
                                                  3581  (1)       3629  (1)       3639  (1)       3659  (1)
                                                  3662  (1)       3669  (1)       3715  (1)       3723  (1)
                                                  3731  (1)       3739  (1)       3747  (1)       3755  (1)
                                                  3772  (1)       3788  (1)       3830  (1)       3834  (1)
                                                  3879  (1)       3891  (1)       3923  (1)       3932  (1)
CLI$L_FLAGS2               00000444     579 (1) #-2644  (1)     #-2657  (1)     2675  (1)      2682  (1)
                                                #-2871  (1)       3249  (1)       3260  (1)
CLI$L_LAST                 00000024     579 (1) #-3067  (1)
CLI$L_NEXT                 00000030     579 (1) #-3621  (1)
CLI$L_PASS                 0000002C     579 (1) #-2662  (1)     #-2664  (1)     #-2876  (1)     #-2878  (1)
                                                #-3029  (1)
CLI$L_SUBT                 00000028     579 (1) #-3053  (1)
CLI$L_TEMP_BASE            00000448     579 (1) #-3247  (1)     #-3258  (1)
CLI$L_TEST                 00000020     579 (1) #-3060  (1)
CLI$M_START_OR_RUN        =00000008     579 (1) #-2656  (1)     #-2870  (1)
CLI$Q_BUFQWD               00000034     579 (1)   2099  (1)     #-2119  (1)       2128  (1)     #-2386  (1)
CLI$Q_FILE                 00000008     579 (1) #-2428  (1)     #-2429  (1)     #-2514  (1)     #-2516  (1)
                                                #-2715  (1)       2716  (1)     #-2734  (1)       2736  (1)
                                                  2787  (1)     #-2987  (1)     #-2988  (1)     #-2995  (1)
                                                #-2996  (1)     #-3773  (1)     #-3801  (1)
CLI$Q_SECTION              00000010     579 (1) #-3045  (1)     #-3046  (1)
CLI$Q_TIME                 0000043C     579 (1) #-3075  (1)     #-3076  (1)     #-3088  (1)     #-3089  (1)
                                                #-3091  (1)       3092  (1)     #-3098  (1)     #-3100  (1)
                                                  3101  (1)     #-3102  (1)     #-3103  (1)       3104  (1)
                                                #-3105  (1)     #-3109  (1)     #-3111  (1)     #-3112  (1)
                                                  3118  (1)
CLI$T_BUFFER               0000003C     579 (1)   2100  (1)       2118  (1)
CLI$V_ADAPTER             =00000018     579 (1) #-3922  (1)
CLI$V_ADR                 =0000000B     579 (1) #-3217  (1)     #-3661  (1)     #-3833  (1)
CLI$V_ASCII               =00000013     579 (1) #-3628  (1)
CLI$V_BASE_QUAL           =00000002     579 (1) #-3248  (1)     #-3259  (1)
CLI$V_BREAK               =0000000A     579 (1) #-3225  (1)
CLI$V_BRIEF               =0000001B     579 (1) #-3879  (1)     #-3891  (1)     #-3932  (1)
CLI$V_BYTE                =0000000D     579 (1) #-3730  (1)
CLI$V_CLEAR               =00000002     579 (1) #-2468  (1)     #-3174  (1)     #-3232  (1)
CLI$V_DEC                 =00000010     579 (1) #-3746  (1)
CLI$V_DEFAULT             =0000000C     579 (1) #-3299  (1)     #-3572  (1)
CLI$V_DEPOSIT             =00000019     579 (1) #-2526  (1)
```

```
CLI$V_EVENT                     =00000008     579    (1)    #-3167   (1)    #-3187   (1)
CLI$V_EXAM                      =00000005     579    (1)    #-2576   (1)
CLI$V_FLAGS                     =00000009     579    (1)    #-3189   (1)    #-3286   (1)
CLI$V_HEX                       =00000012     579    (1)    #-3738   (1)
CLI$V_KERNEL                    =00000017     579    (1)
CLI$V_LOAD                      =00000006     579    (1)    #-2632   (1)
CLI$V_LONG                      =0000000F     579    (1)    #-3714   (1)
CLI$V_NEXT                      =00000001     579    (1)    #-2643   (1)
CLI$V_NOALLOC                   =00000000     579    (1)    #-2674   (1)    #-2681   (1)
CLI$V_NOTNUF                    =00000001     579    (1)    #-2141   (1)    #-2369   (1)    #-2377   (1)    #-2411   (1)
                                                            #-2419   (1)
CLI$V_OCT                       =00000011     579    (1)    #-3754   (1)
CLI$V_PREG                      =0000001A     579    (1)    #-3669   (1)
CLI$V_QA                        =00000007     579    (1)    #-3036   (1)
CLI$V_QACKLOOPLOOPS             =0000001C     579    (1)    #-3533   (1)
CLI$V_QAERRORPRINTS             =0000001B     579    (1)    #-3528   (1)
CLI$V_QAMULTIPLEPASS            =0000001F     579    (1)    #-3548   (1)
CLI$V_QASUBTESTLOOPS            =0000001E     579    (1)    #-3543   (1)
CLI$V_QATESTLOOPS               =0000001D     579    (1)    #-3538   (1)
CLI$V_REG                       =00000014     579    (1)    #-3638   (1)    #-3658   (1)    #-3829   (1)
CLI$V_REQUIRED                  =00000000     579    (1)    #-3003   (1)    #-3011   (1)    #-3787   (1)
CLI$V_RUN                       =00000015     579    (1)    #-2658   (1)    #-2872   (1)    #-3771   (1)
CLI$V_SET                       =00000003     579    (1)    #-2691   (1)    #-3169   (1)    #-3227   (1)    #-3273   (1)
                                                            #-3493   (1)    #-3510   (1)    #-3555   (1)    #-3580   (1)
CLI$V_SHOW                      =00000004     579    (1)    #-2859   (1)    #-3191   (1)    #-3203   (1)    #-3289   (1)
                                                            #-3491   (1)    #-3508   (1)    #-3553   (1)    #-3578   (1)
CLI$V_START_OR_RUN              =00000003     579    (1)      579    (1)
CLI$V_VALSEC                    =00000016     579    (1)
CLI$V_WORD                      =0000000E     579    (1)    #-3722   (1)
CLI_ACTION                       0000013D-R   2203   (1)      2126   (1)
CMK$APBOOT                       00000000-XR                 2459   (1)
CMK$SGIPR                        00000000-XR                 4019   (1)
CMK$_APBOOT                     =00000004     586    (1)    #-2459   (1)
CMK$_ASTEXIT                    =00000000     586    (1)
CMK$_CANTIM                     =0000000B     586    (1)
CMK$_HIBER                      =00000007     586    (1)
CMK$_INITSCB                    =00000008     586    (1)
CMK$_LAST                       =0000000E     586    (1)
CMK$_MMENABLE                   =00000003     586    (1)
CMK$_RCONSOLE                   =00000001     586    (1)
CMK$_REFRESH                    =00000005     586    (1)
CMK$_SCHDWK                     =00000006     586    (1)
CMK$_SETIMR                     =0000000C     586    (1)
CMK$_SETIPL                     =00000009     586    (1)
CMK$_SETPRT                     =0000000D     586    (1)
CMK$_SGIPR                      =0000000A     586    (1)    #-4019   (1)
CMK$_TCONSOLE                   =00000002     586    (1)
COMMAND_TREE                     00000295-R   701    (1)      2127   (1)
COMMON_CRD_TRACE_DEBUG           000009D2-R   3345   (1)    #-3325   (1)    #-3331   (1)
COMMON_ENFORCE                   0000086B-R   3141   (1)    #-3138   (1)
CONTEXT                         =00000003     2210   (1)
CONTINUE                        =0000000A     2217   (1)      740    (1)
CRD                             =00000027     2246   (1)      745    (1)
CRD$GL_CRD_COMMAND_DEBUG         00000000-XR                 #-2922   (1)    #-2927   (1)
CRD$K_CRD_INITIALIZATION        =00000000     585    (1)    #-2965   (1)
CRD$K_DS_CONTROL_C              =00000001     585    (1)
CRD$K_DS_FLAGS                  =00000000     585    (1)
```

```
CRD$K_FAILURE                   =00000000     585    (1)
CRD$K_HOOK_POINT                =00000000     585    (1)    #-2966   (1)
CRD$K_LAST_ACCESS_CODE          =00000002     585    (1)
CRD$K_LAST_DS_VARIABLE          =00000002     585    (1)
CRD$K_LAST_FUNCTION             =00000002     585    (1)
CRD$K_LAST_HOOK_POINT           =00000001     585    (1)
CRD$K_READ                      =00000000     585    (1)
CRD$K_SUCCESS                   =00000001     585    (1)
CRD$K_TYPECODE                  =00000001     585    (1)
CRD$K_WRITE                     =00000001     585    (1)
DATA                            =00000066    2309    (1)     1881   (1)     1901   (1)     1935   (1)     1945   (1)
                                                             721   (1)      774   (1)      809   (1)      862   (1)
DBG_NAME                         0000037F-R  2518    (1)    #-2514   (1)     2515   (1)
DEATTACH                        =0000000B    2218    (1)     783   (1)
DEATTACH_PARAMS                  000009A1-R  1450    (1)     1467   (1)      784   (1)
DEBUG                           =0000000C    2219    (1)     788   (1)
DEBUG$GB_FLAGS                   00000000-XR                #-2476   (1)   #-2645   (1)
DEBUG_SWITCH                    =0000000D    2220    (1)     1292   (1)
DEC                             =0000006C    2315    (1)     1419   (1)     1975   (1)
DEFAULT                         =00000043    2274    (1)     1575   (1)     1849   (1)
DEFAULT_DBG                     =0000000E    2221    (1)     792   (1)
DEPOSIT                         =0000000F    2222    (1)     804   (1)
DESELECT                        =0000007A    2329    (1)     796   (1)
DESELECT_PARAMS                  000009E9-R  1473    (1)     797   (1)
DEVICE                          =00000076    2325    (1)     1580   (1)     1710   (1)
DEVICE_NAMES                     00000604-R  1099    (1)     1480   (1)     1499   (1)     1582   (1)     1711   (1)
DEVICE_QUALS                     000008BF-R  1376    (1)     1383   (1)     1386   (1)     1581   (1)     1707   (1)
DIRECTORY                       =00000010    2223    (1)     763   (1)
DIR_WIDE                        =0000008C    2347    (1)     765   (1)
DO_CMD                          =0000007E    2333    (1)     1104   (1)
DS$A_PRGBGN                      00000000-XR                #-2634   (1)
DS$CLI                           00000000-R  2050    (1)
DS$GA_CRD_DS_INTERFACE           00000000-XR                2968   (1)
DS$GA_DS_CTRL_C_FIRST            00000000-XR               #-2944   (1)
DS$GA_DS_CTRL_C_SECOND           00000000-XR               #-2945   (1)
DS$GB_INHIBIT_NAMING             00000000-XR               #-3142   (1)   #-3152   (1)
DS$GB_MM_ENB                     00000000-XR               #-3979   (1)   #-3998   (1)
DS$GL_CLIBASE                    00000000-R   597    (1)     2054   (1)
DS$GL_CRD_DEBUG                  00000000-XR               #-3351   (1)   #-3356   (1)
DS$GL_FLAGS                      00000000-XR                2070   (1)     2078   (1)     2095   (1)     2559   (1)
                                                          #-2655   (1)   #-2869   (1)     2949   (1)     2952   (1)
DS$GPHARD                        00000000-XR                2729   (1)
DS$GT_PROMPT                     00000000-XR                2081   (1)     2097   (1)
DS$K_ERROR                      =00000002     578    (1)
DS$K_NORMAL                     =00000001     578    (1)
DS$K_PRINTB                     =00000002     584    (1)
DS$K_PRINTF                     =00000001     584    (1)   #-2450   (1)   #-2490   (1)   #-2561   (1)   #-2618   (1)
                                                          #-2706   (1)   #-2770   (1)   #-2790   (1)   #-2816   (1)
                                                          #-2821   (1)   #-3974   (1)   #-3993   (1)   #-4012   (1)
                                                          #-4026   (1)   #-4092   (1)   #-4099   (1)
DS$K_PRINTI                     =00000000     584    (1)   #-2081   (1)   #-2150   (1)   #-2161   (1)   #-2485   (1)
                                                          #-2912   (1)   #-2976   (1)   #-3158   (1)   #-3343   (1)
                                                          #-3608   (1)
DS$K_PRINTX                     =00000003     584    (1)
DS$K_SEVERE                     =00000004     578    (1)
DS$K_SUBSYS                     =00000066     578    (1)     578   (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     584    (1)
```

```
DS$K_TYPE_ABORT_TEST            =00000013     584     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     584     (1)    #-2151   (1)    #-2162   (1)    #-2451   (1)    #-2562   (1)
                                                             #-2619   (1)    #-2707   (1)    #-2771   (1)    #-2791   (1)
                                                             #-2816   (1)    #-2912   (1)    #-2976   (1)    #-3609   (1)
                                                             #-3975   (1)    #-3994   (1)    #-4012   (1)    #-4093   (1)
                                                             #-4100   (1)
DS$K_TYPE_COMMAND_OUT           =00000016     584     (1)    #-2485   (1)    #-2490   (1)    #-2822   (1)    #-3158   (1)
                                                             #-3343   (1)    #-4027   (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     584     (1)
DS$K_TYPE_DS_PROMPT             =00000001     584     (1)    #-2081   (1)
DS$K_TYPE_DS_START              =0000001D     584     (1)
DS$K_TYPE_ERRDEV                =00000008     584     (1)
DS$K_TYPE_ERRHARD               =00000006     584     (1)
DS$K_TYPE_ERROR_BODY            =00000009     584     (1)
DS$K_TYPE_ERROR_END             =0000000A     584     (1)
DS$K_TYPE_ERRPREP               =0000001B     584     (1)
DS$K_TYPE_ERRSOFT               =00000007     584     (1)
DS$K_TYPE_ERRSUP                =00000004     584     (1)
DS$K_TYPE_ERRSYS                =00000005     584     (1)
DS$K_TYPE_ERR_HALT              =0000000D     584     (1)
DS$K_TYPE_EXCEPTION             =0000000C     584     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     584     (1)
DS$K_TYPE_FIRST_PASS            =00000011     584     (1)
DS$K_TYPE_GENERAL               =00000000     584     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     584     (1)
DS$K_TYPE_NO_TESTS              =00000012     584     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     584     (1)
DS$K_TYPE_PROGRAM_END           =00000010     584     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     584     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     584     (1)
DS$K_TYPE_QIO_INVADP            =00000024     584     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     584     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     584     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     584     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     584     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     584     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     584     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     584     (1)
DS$K_TYPE_START_ERR             =00000018     584     (1)
DS$K_TYPE_START_LIST            =00000025     584     (1)
DS$K_TYPE_SUMMARY               =0000000E     584     (1)
DS$K_TYPE_USER_PROMPT           =00000002     584     (1)
DS$K_WARNING                    =00000000     578     (1)
DS$L_USERCNTRLC                  00000000-XR                    #-2946   (1)
DS$M_ABRTFLG                    =00000040     580     (1)
DS$M_BADTIME                    =00100000     580     (1)
DS$M_BATCH                      =00400000     580     (1)
DS$M_BRKCLR                     =00001000     580     (1)
DS$M_BRKPT                      =00000800     580     (1)
DS$M_CHARFLG                    =00000100     580     (1)
DS$M_CMDFLG                     =00000080     580     (1)
DS$M_CTRLC                      =00000001     580     (1)
DS$M_CTRLO                      =00010000     580     (1)
DS$M_DEVFLG                     =00000200     580     (1)
DS$M_DISABLCC                   =01000000     580     (1)
DS$M_DONFLG                     =00002000     580     (1)
DS$M_ERRFLG                     =00000010     580     (1)
```

```
DS$M_EXCEPT                     =00080000     580     (1)
DS$M_EXETST                     =00040000     580     (1)
DS$M_HLTFLG                     =00000008     580     (1)
DS$M_LODFLG                     =00000002     580     (1)
DS$M_MEMMGT                     =00008000     580     (1)
DS$M_NI                         =04000000     580     (1)
DS$M_OUTPUT                     =00800000     580     (1)
DS$M_RUBFLG                     =00000020     580     (1)
DS$M_SCRIPT                     =00200000     580     (1)
DS$M_SETIMR                     =02000000     580     (1)
DS$M_STRFLG                     =00000004     580     (1)
DS$M_SUBT                       =00004000     580     (1)
DS$M_SYSFLG                     =00000400     580     (1)
DS$M_TIMED                      =02000000     580     (1)     #-2655    (1)     #-2869    (1)
DS$M_TIMED_OUT                  =08000000     580     (1)
DS$M_TIMRON                     =00020000     580     (1)
DS$V_ABRTFLG                    =00000006     580     (1)
DS$V_BADTIME                    =00000014     580     (1)
DS$V_BATCH                      =00000016     580     (1)
DS$V_BRKCLR                     =0000000C     580     (1)
DS$V_BRKPT                      =0000000B     580     (1)
DS$V_CHARFLG                    =00000008     580     (1)
DS$V_CMDFLG                     =00000007     580     (1)
DS$V_CTRLC                      =00000000     580     (1)     #-2070    (1)     #-2095    (1)     #-2949    (1)
DS$V_CTRLO                      =00000010     580     (1)     #-2078    (1)
DS$V_DEVFLG                     =00000009     580     (1)
DS$V_DISABLCC                   =00000018     580     (1)     #-2952    (1)
DS$V_DONFLG                     =0000000D     580     (1)
DS$V_ERRFLG                     =00000004     580     (1)
DS$V_EXCEPT                     =00000013     580     (1)
DS$V_EXETST                     =00000012     580     (1)
DS$V_HLTFLG                     =00000003     580     (1)
DS$V_LODFLG                     =00000001     580     (1)
DS$V_MEMMGT                     =0000000F     580     (1)
DS$V_NI                         =0000001A     580     (1)     #-2559    (1)
DS$V_OUTPUT                     =00000017     580     (1)
DS$V_RUBFLG                     =00000005     580     (1)
DS$V_SCRIPT                     =00000015     580     (1)
DS$V_SETIMR                     =00000019     580     (1)
DS$V_STRFLG                     =00000002     580     (1)
DS$V_SUBT                       =0000000E     580     (1)
DS$V_SYSFLG                     =0000000A     580     (1)
DS$V_TIMED                      =00000019     580     (1)
DS$V_TIMED_OUT                  =0000001B     580     (1)
DS$V_TIMRON                     =00000011     580     (1)
DS$_AP_NORMAL_BREAK             =00660150     578     (1)
DS$_ARITH                       =006600D0     578     (1)
DS$_ASBE                        =00660118     578     (1)
DS$_BADLINK                     =006600F0     578     (1)
DS$_BADTYPE                     =006600E8     578     (1)
DS$_BIIC                        =00660120     578     (1)
DS$_CHME                        =006600A8     578     (1)
DS$_CHMK                        =006600E0     578     (1)
DS$_DEVNAME                     =00660108     578     (1)
DS$_ERROR                       =00660002     578     (1)
DS$_FHWE                        =00660068     578     (1)
DS$_FRAGBUF                     =00660080     578     (1)
```

```
DS$_ICBUSY                      =006600C8     578     (1)
DS$_ICERR                       =006600C0     578     (1)
DS$_IHWE                        =00660060     578     (1)
DS$_ILLCHAR                     =00660018     578     (1)
DS$_ILLPAGCNT                   =00660078     578     (1)
DS$_ILLUNIT                     =00660100     578     (1)
DS$_INIT_FAIL                   =00660140     578     (1)
DS$_INSFMEM                     =00660050     578     (1)
DS$_INVCPU                      =00660130     578     (1)
DS$_IPL2HI                      =006600B8     578     (1)
DS$_IVADDR                      =00660040     578     (1)
DS$_IVVECT                      =00660038     578     (1)
DS$_KRNLSTK                     =00660090     578     (1)
DS$_LOGIC                       =00660070     578     (1)
DS$_MCHK                        =00660088     578     (1)
DS$_MEM_ALLOC_ERR               =00660138     578     (1)
DS$_MMOFF                       =00660058     578     (1)
DS$_NEEDUNIT                    =006600F8     578     (1)
DS$_NODE                        =00660128     578     (1)
DS$_NOPCS                       =00660110     578     (1)
DS$_NORMAL                      =00660001     578     (1)
DS$_NOSUPPORT                   =006600B1     578     (1)
DS$_NOTALLOWMP                  =00660148     578     (1)
DS$_NOTDON                      =00660030     578     (1)
DS$_NOTIMP                      =006600B0     578     (1)       578     (1)
DS$_NULLSTR                     =00660010     578     (1)
DS$_OVERFLOW                    =00660008     578     (1)
DS$_POWER                       =00660098     578     (1)
DS$_PROGERR                     =00660020     578     (1)
DS$_SEVERE                      =00660004     578     (1)
DS$_TRANSL                      =006600A0     578     (1)
DS$_TRUNCATE                    =00660028     578     (1)
DS$_UNEXPINT                    =006600D8     578     (1)
DS$_UNSUPPDEV                   =00660158     578     (1)
DS$_VASFULL                     =00660048     578     (1)
DS$_WARNING                     =00660000     578     (1)       578     (1)
DSA$GL_APTCOM                    0000FE04                     #-2073    (1)
DSA$GL_FLAGS                     0000FE00                       2069    (1)      2087    (1)      2088    (1)      2089    (1)
                                                                2448    (1)      2616    (1)      2663    (1)      2704    (1)
                                                                2813    (1)      2877    (1)      2904    (1)    #-2918    (1)
                                                              #-2924    (1)    #-2929    (1)      2932    (1)    #-2934    (1)
                                                                2940    (1)    #-3022    (1)      3323    (1)      3329    (1)
                                                                3336    (1)      3847    (1)      3862    (1)      3972    (1)
                                                                3991    (1)      4009    (1)      4089    (1)
DSA$M_CRD_AUTOTEST_OFF          =00000800                    #-2916    (1)    #-2929    (1)
DSA$M_CRD_MENUTEST_OFF          =00010000                    #-2917    (1)    #-2924    (1)
DSA$M_CRD_MENUTEST_ON           =00040000                    #-2918    (1)    #-2934    (1)
DSA$V_APT                       =0000001F                    #-2069    (1)
DSA$V_ASK_PROMPT                =00000013                    #-3365    (1)
DSA$V_BELL                      =00000003                    #-3373    (1)
DSA$V_BINARY                    =00000001                    #-3381    (1)
DSA$V_CRD_AUTOTEST_OFF          =0000000B                    #-2087    (1)
DSA$V_CRD_MENUTEST_OFF          =00000010                    #-2088    (1)
DSA$V_CRD_MENUTEST_ON           =00000012                    #-2089    (1)
DSA$V_CRD_TRACE                 =00000011                    #-3322    (1)    #-3328    (1)    #-3335    (1)
DSA$V_HALT                      =00000000                    #-3389    (1)
DSA$V_IE1                       =00000004                    #-3397    (1)
```

```
DSA$V_IE2                  =00000005                  #-3405   (1)
DSA$V_IE3                  =00000006                  #-3413   (1)
DSA$V_IES                  =00000007                  #-3421   (1)
DSA$V_LOAD_DEBUGGER        =0000001E                  #-3021   (1)
DSA$V_LOOP                 =00000002                  #-3431   (1)
DSA$V_OPER                 =0000000C                  #-3439   (1)
DSA$V_PROMPT               =0000000D                  #-3447   (1)
DSA$V_QUICK                =00000008                  #-3455   (1)
DSA$V_SEARCH               =0000000E                  #-2663   (1)    #-2877   (1)    #-3465   (1)
DSA$V_TRACE                =0000000A                  #-3473   (1)
DSA$V_USER                 =0000001C                  #-2448   (1)    #-2616   (1)    #-2704   (1)    #-2813   (1)
                                                      #-2904   (1)    #-2932   (1)    #-2940   (1)    #-3846   (1)
                                                      #-3861   (1)    #-3972   (1)    #-3991   (1)    #-4009   (1)
                                                      #-4089   (1)
DSA$V_VERIFY               =00000009                  #-3481   (1)
DSR$COMPLETION              00000000-XR                2605   (1)
DSR$LOAD_CRD                00000000-XR                2955   (1)
DSR$UNLOAD_CRD              00000000-XR                2978   (1)
DSV$ATTACH                  00000000-XR                2430   (1)
DSV$DEATTACH                00000000-XR                2501   (1)
DSV$DEBUG                   00000000-XR                2510   (1)
DSV$DESELECT                00000000-XR                2534   (1)
DSV$DIRECTORY               00000000-XR                2542   (1)
DSV$EXIT                    00000000-XR                2584   (1)
DSV$INITPCS                 00000000-XR                2623   (1)
DSV$LOAD                    00000000-XR                2630   (1)
DSV$RUN                     00000000-XR                2660   (1)
DSV$SELECT                  00000000-XR                2673   (1)
DSV$SETLOAD                 00000000-XR                2838   (1)      3953   (1)
DSV$SETPAGE                 00000000-XR                3495   (1)
DSV$SHOWDEVICE              00000000-XR                3881   (1)      3893   (1)
DSV$SHOWDEVICEB             00000000-XR                3884   (1)      3896   (1)
DSV$SHOWLOAD                00000000-XR                3960   (1)
DSV$SHOWMEMORY              00000000-XR                4063   (1)
DSV$SHOWPAGE                00000000-XR                3498   (1)
DSV$SHOWSELECT              00000000-XR                3898   (1)      3934   (1)
DSV$SHOWSELECTB             00000000-XR                3901   (1)      3937   (1)
DSV$SHOWTESTS               00000000-XR                4056   (1)
DSX$HELP                    00000000-XR                2602   (1)
DSX$MMOFF                   00000000-XR                3999   (1)
DSX$MMON                    00000000-XR                3980   (1)
DSX$PRINT                   00000000-XR                2081   (1)      2152   (1)      2163   (1)      2452   (1)
                                                       2485   (1)      2490   (1)      2563   (1)      2620   (1)
                                                       2708   (1)      2772   (1)      2792   (1)      2816   (1)
                                                       2823   (1)      2912   (1)      2976   (1)      3158   (1)
                                                       3343   (1)      3610   (1)      3976   (1)      3995   (1)
                                                       4012   (1)      4028   (1)      4094   (1)      4101   (1)
DSX$SUPERPARSE              00000000-XR                2129   (1)
DS_CLEANUP                  00000000-XR                2595   (1)      2938   (1)
DS_ERRSUP                   00000000-XR                2353   (1)
DS_SUPERLINE                00000000-XR                2101   (1)
DUMP                       =00000011      2224   (1)   772    (1)
DUMP_SELF                   00000000-XR                2567   (1)      557    (1)      558    (1)
EFN                        =00000058      2295   (1)   1121   (1)
END_FLAGS_LOOP              000005FC-R     1092   (1)   1041   (1)      1044   (1)      1048   (1)      1051   (1)
                                                       1054   (1)      1059   (1)      1062   (1)      1065   (1)
                                                       1068   (1)      1071   (1)      1074   (1)      1077   (1)
```

				1080	(1)	1085	(1)	1088	(1)				
ENUF	=00000017	2230	(1)	1108	(1)	1168	(1)	1355	(1)	1383	(1)		
				1386	(1)	1541	(1)	1546	(1)	1552	(1)		
				1558	(1)	1564	(1)	1576	(1)	1588	(1)		
				1593	(1)	1594	(1)	1595	(1)	1600	(1)		
				1605	(1)	1618	(1)	1621	(1)	1624	(1)		
				1627	(1)	1632	(1)	1637	(1)	1644	(1)		
				1649	(1)	1654	(1)	1659	(1)	1664	(1)		
				1669	(1)	1676	(1)	1688	(1)	1693	(1)		
				1698	(1)	1703	(1)	1714	(1)	1811	(1)		
				1871	(1)	1882	(1)	1891	(1)	1894	(1)		
				1902	(1)	1920	(1)	1946	(1)	2006	(1)		
				709	(1)	714	(1)	722	(1)	741	(1)		
				746	(1)	768	(1)	775	(1)	777	(1)		
				791	(1)	825	(1)	842	(1)	844	(1)		
				849	(1)	856	(1)	863	(1)	872	(1)		
				906	(1)	911	(1)	918	(1)	925	(1)		
EOL	000004AF-R	935	(1)	1093	(1)	1108	(1)	1119	(1)	1468	(1)		
				1541	(1)	1546	(1)	1552	(1)	1558	(1)		
				1564	(1)	1576	(1)	1588	(1)	1593	(1)		
				1594	(1)	1595	(1)	1600	(1)	1605	(1)		
				1615	(1)	1632	(1)	1637	(1)	1644	(1)		
				1649	(1)	1654	(1)	1659	(1)	1664	(1)		
				1669	(1)	1676	(1)	1683	(1)	1688	(1)		
				1693	(1)	1698	(1)	1703	(1)	1714	(1)		
				1750	(1)	1759	(1)	1764	(1)	1770	(1)		
				1811	(1)	1820	(1)	1832	(1)	1840	(1)		
				1845	(1)	1856	(1)	1871	(1)	1882	(1)		
				1891	(1)	1894	(1)	1902	(1)	1920	(1)		
				1936	(1)	1946	(1)	2006	(1)	709	(1)		
				722	(1)	741	(1)	746	(1)	766	(1)		
				768	(1)	775	(1)	776	(1)	777	(1)		
				791	(1)	792	(1)	811	(1)	825	(1)		
				834	(1)	849	(1)	856	(1)	861	(1)		
				863	(1)	872	(1)	906	(1)	911	(1)		
				918	(1)	922	(1)	925	(1)				
EVENT	=0000003A	2265	(1)	1592	(1)	1775	(1)	1861	(1)				
EVENT_FLAGS	00000624-R	1113	(1)	1776	(1)	1862	(1)						
EXAMINE	=00000012	2225	(1)	830	(1)								
EXA_DEP_QUALS	000008F1-R	1404	(1)	1408	(1)	1414	(1)	1417	(1)	1420	(1)		
				1423	(1)	1426	(1)	1430	(1)	1435	(1)		
				1438	(1)	1441	(1)	804	(1)	807	(1)		
				810	(1)	830	(1)	833	(1)				
EXIT	=00000013	2226	(1)	824	(1)								
FAILURE	=00000005	2212	(1)										
FALSE	=00000000	585	(1)										
FILEND	=00000029	2248	(1)	1102	(1)	1165	(1)	1461	(1)	1462	(1)		
				1830	(1)								
FILESPEC	=00000028	2247	(1)	1101	(1)	1136	(1)	1460	(1)	1829	(1)		
FLAGS	=0000003B	2266	(1)	1115	(1)	1599	(1)						
FLAGS_LIST	00000517-R	1034	(1)	1780	(1)	1950	(1)						
FLAGS_LOOP	00000525-R	1038	(1)	1035	(1)	1094	(1)						
GET_ADDRESS	00000772-R	1232	(1)	806	(1)	832	(1)						
GET_FILE_SPEC	0000064E-R	1135	(1)	1870	(1)	767	(1)	790	(1)	855	(1)		
				870	(1)	924	(1)						
GT_MENU_ERROR	000001EB-R	677	(1)	2976	(1)								
HALT	=00000046	2277	(1)	1053	(1)								

```
HELP                            =00000014     2227    (1)       840    (1)       843    (1)
HELPINFO                         00000000-XR                    2596    (1)    #-2600    (1)
HEX                             =0000006B     2314    (1)      1422    (1)      1980    (1)
HP$T_DEVICE                      0000000C                      2736    (1)      2777    (1)
HP$T_TYPE                        00000026                   #-2747    (1)      2749    (1)    #-2753    (1)      2755    (1)
                                                            #-2759    (1)      2761    (1)
IE1                             =00000047     2278    (1)      1058    (1)
IE2                             =00000048     2279    (1)      1061    (1)
IE3                             =00000049     2280    (1)      1064    (1)
IES                             =0000004A     2281    (1)      1067    (1)
IFNOTUSER                       =00000073     2322    (1)
IF_ADAPTER                      =00000071     2320    (1)      1466    (1)      1468    (1)
IF_FILE_SPEC                    =00000070     2319    (1)      1166    (1)      1459    (1)
IF_REG_OR_ADR                   =00000072     2321    (1)
IF_REQUIRED                     =0000006F     2318    (1)      1107    (1)      1167    (1)
IF_RUN                          =0000006E     2317    (1)
IF_XDELTA                       =00000074     2323    (1)       916    (1)
ILLEGAL                          000004C7-R    946    (1)      1036    (1)      1107    (1)      1114    (1)      1116    (1)
                                                             1145    (1)      1152    (1)      1155    (1)      1216    (1)
                                                             1217    (1)      1219    (1)      1222    (1)      1223    (1)
                                                             1260    (1)      1266    (1)      1272    (1)      1277    (1)
                                                             1298    (1)      1299    (1)      1312    (1)      1313    (1)
                                                             1319    (1)      1320    (1)      1328    (1)      1340    (1)
                                                             1349    (1)      1350    (1)      1352    (1)      1364    (1)
                                                             1365    (1)      1380    (1)      1381    (1)      1412    (1)
                                                             1413    (1)      1433    (1)      1434    (1)      1454    (1)
                                                             1456    (1)      1460    (1)      1476    (1)      1477    (1)
                                                             1491    (1)      1492    (1)      1545    (1)      1563    (1)
                                                             1571    (1)      1580    (1)      1631    (1)      1642    (1)
                                                             1668    (1)      1692    (1)      1708    (1)      1713    (1)
                                                             1732    (1)      1740    (1)      1747    (1)      1775    (1)
                                                             1803    (1)      1809    (1)      1810    (1)      1817    (1)
                                                             1818    (1)      1828    (1)      1861    (1)      1867    (1)
                                                             1869    (1)      1880    (1)      1881    (1)      1888    (1)
                                                             1889    (1)      1893    (1)      1900    (1)      1901    (1)
                                                             1931    (1)      1934    (1)      1935    (1)      1944    (1)
                                                             1945    (1)      1965    (1)       713    (1)       720    (1)
                                                              721    (1)       745    (1)       774    (1)       805    (1)
                                                              808    (1)       809    (1)       831    (1)       843    (1)
                                                              854    (1)       862    (1)       869    (1)       894    (1)
                                                              910    (1)
ILL_CMD                         =00000001     2208    (1)      1002    (1)      1024    (1)       940    (1)       973    (1)
INCOMPLETE                       000004DB-R    976    (1)      1167    (1)      1175    (1)      1176    (1)      1205    (1)
                                                             1206    (1)      1461    (1)      1469    (1)
INC_CMD                         =00000002     2209    (1)      1017    (1)      1020    (1)       965    (1)       981    (1)
                                                              995    (1)       998    (1)
INI$BRK                          00000000-XR                    3867    (1)
INITPCS                         =0000008B     2346    (1)       848    (1)
INIT_CONTEXT                     00000000-XR                    2599    (1)
KA820_CPU                        00000468-R    612    (1)      2761    (1)      2768    (1)
KA880_CPU                        00000462-R    611    (1)      2755    (1)      2767    (1)
KA884_CPU                        0000045C-R    610    (1)      2749    (1)      2766    (1)
KB_CHECK                         00000000-XR                    2092    (1)
KB_CHECK_APT                     00000000-XR                    2072    (1)
LAST                            =0000002D     2252    (1)      1352    (1)
LINE_COUNT                       00000000-XR                 #-2103    (1)
LOAD                            =00000015     2228    (1)       853    (1)
```

```
LONG                          =00000068     2311   (1)      1425   (1)      1985   (1)
LOOP                          =0000004B     2282   (1)      1070   (1)
MAPFREE                        00000000-XR                   2941   (1)
MMOFF                         =00000030     2255   (1)      1893   (1)
MMON                          =0000002F     2254   (1)      1890   (1)
MP$GB_SETCPU                   00000472-R   616    (1)    #-2779   (1)    #-2795   (1)
MP$GL_CONDITION_FLAGS          00000000-XR                #-2477   (1)    #-2641   (1)    #-2653   (1)    #-2867   (1)
                                                            561    (1)
MP$GL_LUN                      00000473-R   618    (1)    #-2726   (1)    #-2729   (1)    #-2740   (1)
MP$GR_BOOTED_PROCESSORS        00000000-XR                #-2481   (1)
MP$GT_DEVICE_NAME              00000477-R   620    (1)      2778   (1)      2797   (1)      2819   (1)
NEXT                          =00000059     2296   (1)      1434   (1)
NEXT_INST                     =00000016     2229   (1)      860    (1)
NOBASE                        =00000040     2271   (1)      1368   (1)      1429   (1)
NOTNUF                        =00000018     2231   (1)      1039   (1)      1046   (1)      1056   (1)      1057   (1)
                                                            1093   (1)      1122   (1)      1215   (1)      1288   (1)
                                                            1297   (1)      1298   (1)      1309   (1)      1311   (1)
                                                            1312   (1)      1318   (1)      1319   (1)      1325   (1)
                                                            1327   (1)      1328   (1)      1348   (1)      1349   (1)
                                                            1351   (1)      1377   (1)      1433   (1)      1451   (1)
                                                            1454   (1)      1467   (1)      1474   (1)      1488   (1)
                                                            1536   (1)      1550   (1)      1570   (1)      1571   (1)
                                                            1586   (1)      1615   (1)      1641   (1)      1642   (1)
                                                            1673   (1)      1674   (1)      1681   (1)      1708   (1)
                                                            1733   (1)      1761   (1)      1762   (1)      1804   (1)
                                                            1809   (1)      1826   (1)      1828   (1)      1842   (1)
                                                            1843   (1)      1850   (1)      1866   (1)      1867   (1)
                                                            1869   (1)      1878   (1)      1887   (1)      1889   (1)
                                                            1898   (1)      1900   (1)      1913   (1)      1914   (1)
                                                            1944   (1)      2001   (1)      704    (1)      720    (1)
                                                            731    (1)      736    (1)      759    (1)      764    (1)
                                                            784    (1)      797    (1)      818    (1)      819    (1)
                                                            854    (1)      884    (1)      888    (1)      890    (1)
                                                            895    (1)      900    (1)      923    (1)
NO_ALLOC                      =0000007C     2331   (1)      1496   (1)
NULL                          =00000000     2207   (1)
NULL_DATA                     =00000067     2310   (1)      776    (1)
OCT                           =0000006D     2316   (1)      1437   (1)      1990   (1)
OFF                           =00000000     585    (1)
ON                            =00000001     585    (1)
OPER                          =0000004C     2283   (1)      1073   (1)
OPTIONAL                      =0000002A     2249   (1)      1582   (1)      1711   (1)      767    (1)      790    (1)
PAGE                          =0000005A     2297   (1)      1636   (1)      1899   (1)
PASS                          =00000034     2259   (1)      1299   (1)
PR$_MAPEN                     =00000038                   #-4018   (1)
PREGN                         =00000032     2257   (1)      1260   (1)
PRIMARY_NAME                   00000455-R   608    (1)      2716   (1)      2796   (1)
PROMPT                        =0000004D     2284   (1)      1076   (1)
PSL$V_IS                      =0000001A     586    (1)    #-2459   (1)    #-4019   (1)
PTABLE                         0000046E-R   614    (1)      2729   (1)    #-2733   (1)
QA                            =00000035     2260   (1)      1304   (1)
QACL                          =00000053     2290   (1)      1643   (1)      1916   (1)
QADEF                         =00000057     2294   (1)      1648   (1)      1919   (1)
QAEP                          =00000052     2289   (1)      1653   (1)      1922   (1)
QAMP                          =00000056     2293   (1)      1658   (1)      1925   (1)
QASL                          =00000055     2292   (1)      1663   (1)      1928   (1)
QATL                          =00000054     2291   (1)      1668   (1)      1931   (1)
```

```
QA_COMMON                  00000A9C-R   3552   (1)    #-3529   (1)    #-3530   (1)    #-3534   (1)    #-3535   (1)
                                                      #-3539   (1)    #-3540   (1)    #-3544   (1)    #-3545   (1)
                                                      #-3549   (1)    #-3573   (1)
QUICK                      =0000004E     2285   (1)      1079   (1)
Q_ADAPTER                  00000000-XR                #-2052   (1)    #-3815   (1)    #-3911   (1)    #-3912   (1)
                                                      #-3919   (1)    #-3920   (1)
Q_GOOD_CMD                 0000044C-R   600    (1)      2148   (1)      2159   (1)    #-2387   (1)    #-2388   (1)
                                                      #-2389   (1)
RCLI                       00000002-R   2051   (1)    #-2083   (1)    #-2110   (1)    #-2115   (1)    #-2153   (1)
                                                      #-2164   (1)    #-2173   (1)
REG12                      =0000005E     2301   (1)      1236   (1)
REG13                      =0000005F     2302   (1)      1243   (1)
REG14                      =00000060     2303   (1)      1271   (1)
REG15                      =00000061     2304   (1)      1250   (1)
REG16                      =00000062     2305   (1)      1255   (1)
REGN                       =0000005D     2300   (1)      1265   (1)
REGP                       =00000063     2306   (1)      1237   (1)      1244   (1)      1272   (1)
REQUIRED                   =0000002B     2250   (1)      1480   (1)      1499   (1)      1870   (1)       855   (1)
                                                        924   (1)
RUN                        =00000019     2232   (1)      867    (1)
SCRIPT                     =0000001A     2233   (1)      922    (1)
SCRIPT$CONT                00000000-XR                  2491   (1)
SCRIPT$FLUSH               00000000-XR                  2594   (1)
SCRIPT$OPEN                00000000-XR                  2885   (1)
SEARCH                     =0000004F     2286   (1)      1084   (1)
SECTION                    =00000033     2258   (1)      1313   (1)
SELECT                     =0000007B     2330   (1)      894    (1)
SELECT_PARAMS              00000A09-R   1487   (1)      1494   (1)      1497   (1)      895    (1)
SET                        =0000001B     2234   (1)      889    (1)
SETCPU                     =00000039     2264   (1)      1832   (1)
SETENFORCE                 =00000036     2261   (1)      1856   (1)
SETLOAD                    =0000007F     2334   (1)      1868   (1)
SETMEM                     =0000008A     2345   (1)      1879   (1)
SET_CRD_DEBUG              =00000020     2239   (1)      1845   (1)
SET_CRD_TRACE              =0000001F     2238   (1)      1840   (1)
SET_DEFAULT_PARAMS          00000F00-R   1964   (1)      1850   (1)
SET_PARAMS                  00000D14-R   1802   (1)      890    (1)
SHOMEMALL                  =00000089     2344   (1)      1617   (1)
SHOMEMBUF                  =00000087     2342   (1)      1620   (1)
SHOMEMDAT                  =00000088     2343   (1)      1623   (1)
SHOMEMMAP                  =00000086     2341   (1)      1626   (1)
SHOW                       =0000001D     2236   (1)      899    (1)
SHOWCPU                    =00000024     2243   (1)      1563   (1)
SHOWENFORCE                =00000038     2263   (1)      1587   (1)
SHOWLOAD                   =00000080     2335   (1)      1604   (1)
SHOWMEM                    =00000085     2340   (1)      1614   (1)
SHOWMEMORYFLAGS             00000000-XR                #-2053   (1)      4077   (1)
SHOWMM                     =00000031     2256   (1)      1631   (1)
SHOWSECTIONS               =00000083     2338   (1)      1675   (1)
SHOWSEL                    =0000007D     2332   (1)      1683   (1)      1713   (1)
SHOWSTATUS                 =00000082     2337   (1)      1687   (1)
SHOWSUP                    =00000081     2336   (1)      1692   (1)
SHOWTESTS                  =00000084     2339   (1)      1697   (1)
SHOW_CALLS                 =00000023     2242   (1)      1557   (1)
SHOW_CRD_TRACE             =00000022     2241   (1)      1551   (1)
SHOW_PARAMS                 00000A40-R   1534   (1)      900    (1)
SIZ...                     =00000001     587    (1)      580    (1)      587    (1)
```

```
SS$_ENDOFFILE                 00000000-XR                  #-2105   (1)
SS$_NORMAL                   =00000001     573    (1)     #-2730   (1)
START                        =00000025     2244   (1)       904    (1)
START_RUN_QUALS               000007CD-R   1283   (1)       1293   (1)      1300   (1)      1305   (1)      1314   (1)
                                                           1321   (1)      1342   (1)      1351   (1)      1353   (1)
                                                           868    (1)      871    (1)      905    (1)
SUBTEST                      =0000002E     2253   (1)       1320   (1)
SUCCESS                      =00000004     2211   (1)
SUMMARY                      =00000026     2245   (1)       910    (1)
SYS$BINTIM                    00000000-XR                   3104   (1)      3118   (1)
SYS$EXIT                      00000000-XR                   2107   (1)
TEST                         =0000002C     2251   (1)       1350   (1)
TIME                         =0000008E     2349   (1)       1340   (1)
TRACE                        =00000050     2287   (1)       1087   (1)
TRUE                         =00000001     585    (1)
T_CONT                        00000092-R   646    (1)       2490   (1)
T_CRD_TRACE_OUTPUT            00000250-R   683    (1)       3343   (1)
T_EMESSG1                     000000BD-R   652    (1)       2449   (1)      2617   (1)      2705   (1)      2816   (1)
                                                           3973   (1)      3992   (1)      4012   (1)      4091   (1)
T_EMESSG2                     000000E2-R   655    (1)       2560   (1)
T_EMESSG3                     00000113-R   658    (1)       2789   (1)
T_EMESSG4                     00000181-R   662    (1)       2769   (1)
T_EMESSG5                     000001A7-R   665    (1)       4098   (1)
T_ILLCMD                      0000001B-R   637    (1)       2149   (1)
T_ILLEFN                      00000061-R   643    (1)       3607   (1)
T_INCOMPLETE                  0000003A-R   640    (1)       2160   (1)
T_MP_CONT                     000000AB-R   649    (1)       2485   (1)
T_OFF                         000001E7-R   674    (1)       3153   (1)      3338   (1)      4015   (1)
T_ON                          000001E4-R   671    (1)       3151   (1)      3334   (1)      4023   (1)
T_PRGERR                      00000004-R   634    (1)       2353   (1)
T_SHOWCPU                     00000288-R   689    (1)       2820   (1)
T_SHOWENFORCE                 0000026D-R   686    (1)       3158   (1)
T_SHOWMM                      000001C8-R   668    (1)       4025   (1)
T_USER_MODE_MENU_ERROR        00000216-R   680    (1)       2912   (1)
ULTRIX_FLAGS                  00000000-XR                   2849   (1)
ULTRIX_ODS                   =0000001C     2235   (1)       1137   (1)
ULTRIX_V_MODE                 00000000-XR                  #-2848   (1)
VERIFY                       =00000051     2288   (1)       1090   (1)
VRABORT                       00000000-XR                   2437   (1)
VRCLRBRK                      00000000-XR                   3234   (1)
VRCLREF                       00000000-XR                   3176   (1)
VRCLRFLG                      00000000-XR                   2466   (1)      3278   (1)
VRDEPOSIT                     00000000-XR                   2524   (1)
VREXAMINE                     00000000-XR                   2574   (1)
VRSETBASE                     00000000-XR                   3208   (1)
VRSETBRK                      00000000-XR                   3229   (1)
VRSETDFLT                     00000000-XR                   3295   (1)
VRSETEF                       00000000-XR                   3171   (1)
VRSETFLG                      00000000-XR                   2689   (1)      3275   (1)
VRSETMEM                      00000000-XR                   4105   (1)
VRSETQA                       00000000-XR                   3558   (1)      3583   (1)
VRSETWIDTH                    00000000-XR                   3512   (1)
VRSHOWBASE                    00000000-XR                   3205   (1)
VRSHOWBRK                     00000000-XR                   3237   (1)
VRSHOWDFLT                    00000000-XR                   3291   (1)
VRSHOWEF                      00000000-XR                   3179   (1)
VRSHOWFLG                     00000000-XR                   3193   (1)
```

```
VRSHOWQA                        00000000-XR                     3562   (1)      3588   (1)
VRSHOWSECTIONS                  00000000-XR                     4051   (1)
VRSHOWSTATUS                    00000000-XR                     4037   (1)
VRSHOWWIDTH                     00000000-XR                     3515   (1)
VRSHOW_CALLS                    00000000-XR                     3310   (1)
VRSTART                         00000000-XR                     2874   (1)
VRSUMMARY                       00000000-XR                     2893   (1)
VRSUPPORT                       00000000-XR                     4044   (1)
WIDE_FLAG                       00000000-XR                   *-2544   (1)    *-2548   (1)
WIDTH                           =0000005B       2298   (1)     1702   (1)      1943   (1)
WORD                            =00000069       2312   (1)     1440   (1)      1995   (1)
XDELBPT                          00000000-XR                    3857   (1)    *-3859   (1)
XDELTA                          =00000075       2324   (1)     917    (1)
XDELTBIT                         00000000-XR                    3857   (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BINTIM_S	1	3104 (1)	3104 (1) 3118 (1)
$CRD_LITERALS	5	585 (1)	585 (1)
$D1_PRINT_S	1	2081 (1)	2081 (1) 2485 (1) 2490 (1) 2816 (1) 2912 (1) 2976 (1) 3158 (1) 3343 (1) 4012 (1)
$DEF	1	587 (1)	
$DEFINI	1	576 (1)	576 (1) 577 (1) 581 (1) 582 (1) 583 (1) 588 (1)
$DS_CLI	1	702 (1)	1000 (1) 1001 (1) 1002 (1) 1016 (1) 1017 (1) 1019 (1) 1020 (1) 1022 (1) 1023 (1) 1024 (1) 1035 (1) 1036 (1) 1039 (1) 1040 (1) 1041 (1) 1043 (1) 1044 (1) 1046 (1) 1047 (1) 1048 (1) 1050 (1) 1051 (1) 1053 (1) 1054 (1) 1056 (1) 1057 (1) 1058 (1) 1059 (1) 1061 (1) 1062 (1) 1064 (1) 1065 (1) 1067 (1) 1068 (1) 1070 (1) 1071 (1) 1073 (1) 1074 (1) 1076 (1) 1077 (1) 1079 (1) 1080 (1) 1084 (1) 1085 (1) 1087 (1) 1088 (1) 1090 (1) 1093 (1) 1094 (1) 1100 (1) 1101 (1) 1102 (1) 1103 (1) 1104 (1) 1105 (1) 1107 (1) 1108 (1) 1114 (1) 1115 (1) 1116 (1) 1118 (1) 1119 (1) 1121 (1) 1122 (1) 1123 (1) 1136 (1) 1137 (1) 1139 (1) 1140 (1) 1142 (1) 1143 (1) 1144 (1) 1145 (1) 1146 (1) 1148 (1) 1149 (1) 1151 (1) 1152 (1) 1153 (1) 1155 (1) 1157 (1) 1159 (1) 1160 (1) 1162 (1) 1163 (1) 1165 (1) 1166 (1) 1167 (1) 1168 (1) 1172 (1) 1173 (1) 1174 (1) 1175 (1) 1176 (1) 1177 (1) 1178 (1) 1179 (1) 1180 (1) 1181 (1) 1182 (1) 1183 (1) 1184 (1) 1185 (1) 1186 (1) 1187 (1) 1188 (1) 1189 (1) 1190 (1) 1191 (1) 1192 (1) 1193 (1) 1194 (1) 1195 (1) 1196 (1) 1197 (1) 1198 (1) 1199 (1) 1200 (1) 1201 (1) 1202 (1) 1203 (1) 1204 (1) 1205 (1) 1206 (1) 1211 (1) 1212 (1) 1213 (1) 1214 (1) 1215 (1) 1216 (1) 1217 (1) 1218 (1) 1219 (1) 1220 (1) 1221 (1) 1222 (1) 1223 (1) 1224 (1) 1236 (1) 1237 (1) 1238 (1) 1243 (1) 1244 (1) 1245 (1) 1249 (1) 1250 (1) 1251 (1) 1255 (1) 1256 (1) 1260 (1) 1261 (1) 1265 (1) 1266 (1) 1267 (1) 1271 (1) 1272 (1) 1273 (1) 1277 (1) 1278 (1) 1279 (1) 1288 (1) 1292 (1) 1293 (1) 1297 (1) 1298 (1) 1299 (1) 1300 (1) 1304 (1)

1305	(1)	1309	(1)	1311	(1)	1312	(1)
1313	(1)	1314	(1)	1318	(1)	1319	(1)
1320	(1)	1321	(1)	1325	(1)	1327	(1)
1328	(1)	1329	(1)	1331	(1)	1332	(1)
1333	(1)	1334	(1)	1335	(1)	1336	(1)
1337	(1)	1338	(1)	1339	(1)	1340	(1)
1342	(1)	1348	(1)	1349	(1)	1350	(1)
1351	(1)	1352	(1)	1353	(1)	1355	(1)
1361	(1)	1363	(1)	1364	(1)	1365	(1)
1366	(1)	1368	(1)	1369	(1)	1371	(1)
1377	(1)	1379	(1)	1380	(1)	1381	(1)
1382	(1)	1383	(1)	1385	(1)	1386	(1)
1388	(1)	1405	(1)	1407	(1)	1408	(1)
1410	(1)	1411	(1)	1412	(1)	1413	(1)
1414	(1)	1416	(1)	1417	(1)	1419	(1)
1420	(1)	1422	(1)	1423	(1)	1425	(1)
1426	(1)	1428	(1)	1429	(1)	1430	(1)
1432	(1)	1433	(1)	1434	(1)	1435	(1)
1437	(1)	1438	(1)	1440	(1)	1441	(1)
1443	(1)	1451	(1)	1453	(1)	1454	(1)
1455	(1)	1456	(1)	1457	(1)	1459	(1)
1460	(1)	1461	(1)	1462	(1)	1463	(1)
1465	(1)	1466	(1)	1467	(1)	1468	(1)
1469	(1)	1474	(1)	1475	(1)	1476	(1)
1477	(1)	1478	(1)	1480	(1)	1488	(1)
1490	(1)	1491	(1)	1492	(1)	1493	(1)
1494	(1)	1496	(1)	1497	(1)	1499	(1)
1535	(1)	1536	(1)	1540	(1)	1541	(1)
1545	(1)	1546	(1)	1550	(1)	1551	(1)
1552	(1)	1557	(1)	1558	(1)	1563	(1)
1564	(1)	1570	(1)	1571	(1)	1575	(1)
1576	(1)	1580	(1)	1581	(1)	1582	(1)
1586	(1)	1587	(1)	1588	(1)	1592	(1)
1593	(1)	1594	(1)	1595	(1)	1599	(1)
1600	(1)	1604	(1)	1605	(1)	1610	(1)
1614	(1)	1615	(1)	1617	(1)	1618	(1)
1620	(1)	1621	(1)	1623	(1)	1624	(1)
1626	(1)	1627	(1)	1631	(1)	1632	(1)
1636	(1)	1637	(1)	1641	(1)	1642	(1)
1643	(1)	1644	(1)	1648	(1)	1649	(1)
1653	(1)	1654	(1)	1658	(1)	1659	(1)
1663	(1)	1664	(1)	1668	(1)	1669	(1)
1673	(1)	1674	(1)	1675	(1)	1676	(1)
1680	(1)	1681	(1)	1682	(1)	1683	(1)
1687	(1)	1688	(1)	1692	(1)	1693	(1)
1697	(1)	1698	(1)	1702	(1)	1703	(1)
1707	(1)	1708	(1)	1710	(1)	1711	(1)
1713	(1)	1714	(1)	1732	(1)	1733	(1)
1738	(1)	1739	(1)	1740	(1)	1742	(1)
1743	(1)	1744	(1)	1746	(1)	1747	(1)
1749	(1)	1750	(1)	1752	(1)	1756	(1)
1757	(1)	1758	(1)	1759	(1)	1761	(1)
1762	(1)	1763	(1)	1764	(1)	1768	(1)
1769	(1)	1770	(1)	1775	(1)	1776	(1)
1780	(1)	1803	(1)	1804	(1)	1808	(1)
1809	(1)	1810	(1)	1811	(1)	1815	(1)
1816	(1)	1817	(1)	1818	(1)	1819	(1)

```
                                                                1820   (1)     1822   (1)     1826   (1)     1827   (1)
                                                                1828   (1)     1829   (1)     1830   (1)     1831   (1)
                                                                1832   (1)     1837   (1)     1838   (1)     1839   (1)
                                                                1840   (1)     1842   (1)     1843   (1)     1844   (1)
                                                                1845   (1)     1849   (1)     1850   (1)     1854   (1)
                                                                1855   (1)     1856   (1)     1861   (1)     1862   (1)
                                                                1866   (1)     1867   (1)     1868   (1)     1869   (1)
                                                                1870   (1)     1871   (1)     1873   (1)     1874   (1)
                                                                1878   (1)     1879   (1)     1880   (1)     1881   (1)
                                                                1882   (1)     1887   (1)     1888   (1)     1889   (1)
                                                                1890   (1)     1891   (1)     1893   (1)     1894   (1)
                                                                1898   (1)     1899   (1)     1900   (1)     1901   (1)
                                                                1902   (1)     1904   (1)     1913   (1)     1914   (1)
                                                                1916   (1)     1917   (1)     1919   (1)     1920   (1)
                                                                1922   (1)     1923   (1)     1925   (1)     1926   (1)
                                                                1928   (1)     1929   (1)     1931   (1)     1932   (1)
                                                                1934   (1)     1935   (1)     1936   (1)     1938   (1)
                                                                1943   (1)     1944   (1)     1945   (1)     1946   (1)
                                                                1950   (1)     1965   (1)     1970   (1)     1971   (1)
                                                                1975   (1)     1976   (1)     1980   (1)     1981   (1)
                                                                1985   (1)     1986   (1)     1990   (1)     1991   (1)
                                                                1995   (1)     1996   (1)     2001   (1)     2002   (1)
                                                                2006   (1)     702    (1)     704    (1)     708    (1)
                                                                709    (1)     713    (1)     714    (1)     719    (1)
                                                                720    (1)     721    (1)     722    (1)     731    (1)
                                                                735    (1)     736    (1)     740    (1)     741    (1)
                                                                745    (1)     746    (1)     759    (1)     763    (1)
                                                                764    (1)     765    (1)     766    (1)     767    (1)
                                                                768    (1)     772    (1)     773    (1)     774    (1)
                                                                775    (1)     776    (1)     777    (1)     779    (1)
                                                                783    (1)     784    (1)     788    (1)     789    (1)
                                                                790    (1)     791    (1)     792    (1)     796    (1)
                                                                797    (1)     803    (1)     804    (1)     805    (1)
                                                                806    (1)     807    (1)     808    (1)     809    (1)
                                                                810    (1)     811    (1)     818    (1)     819    (1)
                                                                824    (1)     825    (1)     829    (1)     830    (1)
                                                                831    (1)     832    (1)     833    (1)     834    (1)
                                                                840    (1)     841    (1)     842    (1)     843    (1)
                                                                844    (1)     848    (1)     849    (1)     853    (1)
                                                                854    (1)     855    (1)     856    (1)     860    (1)
                                                                861    (1)     862    (1)     863    (1)     867    (1)
                                                                868    (1)     869    (1)     870    (1)     871    (1)
                                                                872    (1)     884    (1)     888    (1)     889    (1)
                                                                890    (1)     894    (1)     895    (1)     899    (1)
                                                                900    (1)     904    (1)     905    (1)     906    (1)
                                                                910    (1)     911    (1)     916    (1)     917    (1)
                                                                918    (1)     922    (1)     923    (1)     924    (1)
                                                                925    (1)     935    (1)     936    (1)     937    (1)
                                                                939    (1)     940    (1)     941    (1)     964    (1)
                                                                965    (1)     971    (1)     972    (1)     973    (1)
                                                                981    (1)     994    (1)     995    (1)     997    (1)
                                                                998    (1)
$DS_CLIDEF                      1          575    (1)          575    (1)
$DS_DSADEF                      5          577    (1)          577    (1)
$DS_DSDEF                       3          578    (1)          578    (1)
$DS_GPHARD_S                    1          2728   (1)          2728   (1)
$DS_HPODEF                      2          576    (1)          576    (1)
```

```
$DS_TYPEDEF                 4        584    (1)    584    (1)
$EQU                        1        587    (1)    575    (1)    578    (1)    584    (1)    585    (1)
                                                   586    (1)
$EQULS1                     1        586    (1)    575    (1)    578    (1)    584    (1)    585    (1)
                                                   586    (1)
$EQULST                     1        575    (1)    575    (1)    578    (1)    584    (1)    585    (1)
                                                   586    (1)
$EXIT_S                     1        2107   (1)    2107   (1)
$GBLINI                     2        575    (1)    575    (1)    578    (1)    580    (1)    584    (1)
                                                   585    (1)    586    (1)    587    (1)
$IPLDEF                     1        581    (1)    581    (1)
$PR8SSDEF                   5        588    (1)    588    (1)
$PRDEF                      5        582    (1)    582    (1)
$PRINT                      2        2079   (1)    2079   (1)    2483   (1)    2487   (1)    2814   (1)
                                                   2910   (1)    2976   (1)    3155   (1)    3340   (1)
                                                   4010   (1)
$PSLDEF                     2        583    (1)    583    (1)
$PUSHADR                    1        2353   (1)    2353   (1)    3104   (1)    3118   (1)
$VIELD                      1                      580    (1)    587    (1)
$VIELD1                     1        587    (1)    580    (1)    587    (1)
APDEF                       2        587    (1)    587    (1)
BR_IF_CRD_AUTOTEST_OFF      1        2087   (1)    2087   (1)
BR_IF_CRD_MENUTEST_OFF      1        2088   (1)    2088   (1)
BR_IF_CRD_MENUTEST_ON       1        2089   (1)    2089   (1)
BR_IF_NI                    1        2559   (1)    2559   (1)
BR_IF_NOT_APT               1        2069   (1)    2069   (1)
BR_IF_NOT_USER              1        2448   (1)    2448   (1)    2616   (1)    2704   (1)    2813   (1)
                                                   2904   (1)    3972   (1)    3991   (1)    4009   (1)
                                                   4089   (1)
BR_IF_USER                  1        2940   (1)    2940   (1)
CASEDEF                     1        565    (1)    2207   (1)    2208   (1)    2209   (1)    2210   (1)
                                                   2211   (1)    2212   (1)    2213   (1)    2214   (1)
                                                   2215   (1)    2216   (1)    2217   (1)    2218   (1)
                                                   2219   (1)    2220   (1)    2221   (1)    2222   (1)
                                                   2223   (1)    2224   (1)    2225   (1)    2226   (1)
                                                   2227   (1)    2228   (1)    2229   (1)    2230   (1)
                                                   2231   (1)    2232   (1)    2233   (1)    2234   (1)
                                                   2235   (1)    2236   (1)    2237   (1)    2238   (1)
                                                   2239   (1)    2240   (1)    2241   (1)    2242   (1)
                                                   2243   (1)    2244   (1)    2245   (1)    2246   (1)
                                                   2247   (1)    2248   (1)    2249   (1)    2250   (1)
                                                   2251   (1)    2252   (1)    2253   (1)    2254   (1)
                                                   2255   (1)    2256   (1)    2257   (1)    2258   (1)
                                                   2259   (1)    2260   (1)    2261   (1)    2262   (1)
                                                   2263   (1)    2264   (1)    2265   (1)    2266   (1)
                                                   2267   (1)    2268   (1)    2269   (1)    2270   (1)
                                                   2271   (1)    2272   (1)    2273   (1)    2274   (1)
                                                   2275   (1)    2276   (1)    2277   (1)    2278   (1)
                                                   2279   (1)    2280   (1)    2281   (1)    2282   (1)
                                                   2283   (1)    2284   (1)    2285   (1)    2286   (1)
                                                   2287   (1)    2288   (1)    2289   (1)    2290   (1)
                                                   2291   (1)    2292   (1)    2293   (1)    2294   (1)
                                                   2295   (1)    2296   (1)    2297   (1)    2298   (1)
                                                   2299   (1)    2300   (1)    2301   (1)    2302   (1)
                                                   2303   (1)    2304   (1)    2305   (1)    2306   (1)
                                                   2307   (1)    2308   (1)    2309   (1)    2310   (1)
                                                   2311   (1)    2312   (1)    2313   (1)    2314   (1)
```

				2315	(1)	2316	(1)	2317	(1)	2318	(1)
				2319	(1)	2320	(1)	2321	(1)	2322	(1)
				2323	(1)	2324	(1)	2325	(1)	2326	(1)
				2327	(1)	2328	(1)	2329	(1)	2330	(1)
				2331	(1)	2332	(1)	2333	(1)	2334	(1)
				2335	(1)	2336	(1)	2337	(1)	2338	(1)
				2339	(1)	2340	(1)	2341	(1)	2342	(1)
				2343	(1)	2344	(1)	2345	(1)	2346	(1)
				2347	(1)	2348	(1)	2349	(1)		
CLEAR_CTRLC	1	2095	(1)	2095	(1)						
CLEAR_CTRLO	1	2078	(1)	2078	(1)						
CLIDEF	4	579	(1)	579	(1)						
CMK	1	2459	(1)	2459	(1)	4019	(1)				
CMKDEF	1	586	(1)	586	(1)						
DSFDEF	3	580	(1)	580	(1)						
ERRSUP_S	1	2353	(1)	2353	(1)						
MODNAM	1	630	(1)	630	(1)						

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL2	00C0	3103 (1) 3105 (1)
BBC	00E1	2069 (1) 2448 (1) 2616 (1) 2663 (1) 2704 (1) 2813 (1) 2848 (1)
		2877 (1) 2904 (1) 3169 (1) 3174 (1) 3191 (1) 3203 (1) 3227 (1)
		3232 (1) 3273 (1) 3289 (1) 3493 (1) 3510 (1) 3555 (1) 3580 (1)
		3771 (1) 3787 (1) 3829 (1) 3861 (1) 3972 (1) 3991 (1) 4009 (1)
		4089 (1) 4097 (1)
BBCC	00E5	2369 (1) 2404 (1) 2674 (1) 2872 (1) 3003 (1) 3328 (1)
BBCS	00E3	2377 (1) 2397 (1) 2411 (1) 2658 (1) 3011 (1) 3322 (1) 3605 (1)
		3714 (1) 3722 (1) 3730 (1) 3738 (1) 3746 (1) 3754 (1)
BBS	00E0	2087 (1) 2088 (1) 2089 (1) 2141 (1) 2459 (1) 2559 (1) 2932 (1)
		2940 (1) 3187 (1) 3286 (1) 3335 (1) 3491 (1) 3508 (1) 3553 (1)
		3578 (1) 3833 (1) 3846 (1) 3879 (1) 3932 (1) 4019 (1)
BBSC	00E4	2070 (1) 2078 (1) 2095 (1) 2419 (1) 2949 (1)
BBSS	00E2	2468 (1) 2526 (1) 2576 (1) 2681 (1) 2691 (1) 2859 (1) 2952 (1)
		3036 (1) 3167 (1) 3189 (1) 3217 (1) 3225 (1) 3248 (1) 3259 (1)
		3299 (1) 3365 (1) 3373 (1) 3381 (1) 3389 (1) 3397 (1) 3405 (1)
		3413 (1) 3421 (1) 3431 (1) 3439 (1) 3447 (1) 3455 (1) 3465 (1)
		3473 (1) 3481 (1) 3528 (1) 3533 (1) 3538 (1) 3543 (1) 3548 (1)
		3572 (1) 3628 (1) 3638 (1) 3658 (1) 3661 (1) 3669 (1) 3891 (1)
		3922 (1) 4077 (1)
BEQL	0013	2074 (1) 2171 (1) 2482 (1) 2738 (1) 2751 (1) 2757 (1) 2763 (1)
		2919 (1) 3271 (1) 3602 (1) 3802 (1) 3816 (1) 3860 (1) 3921 (1)
		3945 (1) 4022 (1)
BGTRU	001A	3604 (1)
BICB2	008A	2476 (1)
BICL2	00CA	2655 (1) 2869 (1)
BISB2	0088	2632 (1) 2643 (1) 2645 (1)
BISL2	00C8	2656 (1) 2870 (1) 2924 (1) 2929 (1) 2934 (1) 3021 (1)
BITL	00D3	2916 (1)
BLBC	00E9	3152 (1)
BLBS	00E8	2109 (1) 2133 (1) 2604 (1) 2969 (1) 3121 (1)
BNEQ	0012	2106 (1) 2114 (1) 2720 (1) 2731 (1) 2840 (1) 2842 (1) 2844 (1)
		2846 (1) 2923 (1) 2928 (1) 3093 (1) 3110 (1) 3774 (1) 3895 (1)
		3900 (1)
BRB	0011	2083 (1) 2090 (1) 2093 (1) 2371 (1) 2459 (1) 2486 (1) 2741 (1)
		2774 (1) 2780 (1) 2925 (1) 2930 (1) 2935 (1) 3107 (1) 3113 (1)
		3138 (1) 3173 (1) 3178 (1) 3277 (1) 3325 (1) 3331 (1) 3352 (1)
		3357 (1) 3530 (1) 3535 (1) 3540 (1) 3545 (1) 3653 (1) 3897 (1)
		4019 (1) 4068 (1) 4071 (1) 4074 (1)
BRW	0031	2110 (1) 2115 (1) 2153 (1) 2164 (1) 2173 (1) 2709 (1) 2722 (1)
		2913 (1) 2920 (1) 2933 (1) 3670 (1)
CALLS	00FB	2081 (1) 2101 (1) 2107 (1) 2129 (1) 2152 (1) 2163 (1) 2353 (1)
		2452 (1) 2485 (1) 2490 (1) 2563 (1) 2567 (1) 2602 (1) 2620 (1)
		2623 (1) 2708 (1) 2729 (1) 2772 (1) 2792 (1) 2816 (1) 2823 (1)
		2912 (1) 2941 (1) 2955 (1) 2967 (1) 2976 (1) 2978 (1) 3104 (1)
		3118 (1) 3158 (1) 3343 (1) 3610 (1) 3976 (1) 3980 (1) 3995 (1)
		3999 (1) 4012 (1) 4028 (1) 4094 (1) 4101 (1)
CASEB	008F	2204 (1)
CHMK	00BC	2459 (1) 4019 (1)
CLRB	0094	2053 (1) 2544 (1) 2795 (1) 3137 (1) 3998 (1)

CLRL	00D4	2062	(1)	2103	(1)	2459	(1)	2477	(1)	2641	(1)	2653	(1)	2664	(1)
		2726	(1)	2837	(1)	2867	(1)	2878	(1)	2944	(1)	2945	(1)	2946	(1)
		2988	(1)	3076	(1)	3125	(1)	3143	(1)	3160	(1)	3258	(1)	3346	(1)
		3356	(1)	3501	(1)	3518	(1)	3566	(1)	3592	(1)	3770	(1)	3786	(1)
		3800	(1)	3814	(1)	3828	(1)	3845	(1)	3912	(1)	4019	(1)	4067	(1)
CLRQ	007C	2052	(1)	2066	(1)	2603	(1)	2654	(1)	2868	(1)				
CMPB	0091	2922	(1)	2927	(1)										
CMPC3	0029	2716	(1)	2735	(1)	2748	(1)	2754	(1)	2760	(1)				
CMPL	00D1	2730	(1)	2839	(1)	3109	(1)	3603	(1)	3894	(1)	3899	(1)		
CMPW	00B1	2105	(1)												
CVTBL	0098	2081	(1)	2485	(1)	2490	(1)	2816	(1)	2912	(1)	2976	(1)	3158	(1)
		3343	(1)	4012	(1)										
DECL	00D7	3111	(1)	3678	(1)	3688	(1)								
INCL	00D6	2740	(1)	2850	(1)	3112	(1)	3679	(1)	3687	(1)	3775	(1)	3789	(1)
		3803	(1)	3817	(1)	3831	(1)	3848	(1)						
JMP	0017	2606	(1)												
JSB	0016	2071	(1)	2072	(1)	2076	(1)	2092	(1)	2172	(1)	2459	(1)	2491	(1)
		2594	(1)	2595	(1)	2599	(1)	2605	(1)	2938	(1)	3946	(1)	4019	(1)
		4105	(1)												
LOCC	003A	2841	(1)	2843	(1)	2845	(1)	3091	(1)						
MCOML	00D2	3269	(1)												
MOVAB	009E	2430	(1)	2478	(1)	2534	(1)	2596	(1)	2613	(1)	2646	(1)	2673	(1)
		2810	(1)	3151	(1)	3153	(1)	3334	(1)	3338	(1)	3495	(1)	3498	(1)
		3512	(1)	3515	(1)	3558	(1)	3562	(1)	3583	(1)	3588	(1)	3867	(1)
		3881	(1)	3884	(1)	3893	(1)	3896	(1)	3898	(1)	3901	(1)	3934	(1)
		3937	(1)	3953	(1)	3960	(1)	3969	(1)	3988	(1)	4006	(1)	4023	(1)
		4037	(1)	4044	(1)	4051	(1)	4056	(1)	4063	(1)	4086	(1)		
MOVAL	00DE	2054	(1)	2075	(1)	2118	(1)	2437	(1)	2445	(1)	2466	(1)	2501	(1)
		2510	(1)	2515	(1)	2524	(1)	2542	(1)	2556	(1)	2574	(1)	2584	(1)
		2630	(1)	2660	(1)	2689	(1)	2838	(1)	2874	(1)	2885	(1)	2893	(1)
		2987	(1)	3075	(1)	3102	(1)	3171	(1)	3176	(1)	3179	(1)	3193	(1)
		3205	(1)	3208	(1)	3229	(1)	3234	(1)	3237	(1)	3275	(1)	3278	(1)
		3291	(1)	3295	(1)	3310	(1)	3911	(1)						
MOVB	0090	2548	(1)	2779	(1)	3117	(1)	3140	(1)	3142	(1)	3979	(1)		
MOVC3	0028	2777	(1)	2796	(1)	3100	(1)								
MOVL	00D0	2064	(1)	2170	(1)	2386	(1)	2428	(1)	2429	(1)	2597	(1)	2598	(1)
		2642	(1)	2662	(1)	2711	(1)	2733	(1)	2876	(1)	3029	(1)	3046	(1)
		3053	(1)	3060	(1)	3067	(1)	3087	(1)	3122	(1)	3216	(1)	3247	(1)
		3312	(1)	3351	(1)	3621	(1)	3643	(1)	3646	(1)	3649	(1)	3652	(1)
		3656	(1)	3695	(1)	3704	(1)	3859	(1)	3863	(1)	3944	(1)	4017	(1)
		4018	(1)	4070	(1)	4073	(1)	4076	(1)						
MOVPSL	00DC	2459	(1)	4019	(1)										
MOVQ	007D	2600	(1)												
MOVW	00B0	3099	(1)												
MOVZBL	009A	2081	(1)	2485	(1)	2490	(1)	2514	(1)	2715	(1)	2734	(1)	2747	(1)
		2753	(1)	2759	(1)	2816	(1)	2912	(1)	2976	(1)	3158	(1)	3343	(1)
		4012	(1)												
MOVZWL	003C	2634	(1)												
POPR	00BA	2800	(1)	2851	(1)	3106	(1)	3119	(1)	3902	(1)	4021	(1)		
PUSHAB	009F	2081	(1)	2097	(1)	2100	(1)	2148	(1)	2149	(1)	2159	(1)	2160	(1)
		2449	(1)	2485	(1)	2490	(1)	2560	(1)	2617	(1)	2705	(1)	2766	(1)
		2767	(1)	2768	(1)	2769	(1)	2787	(1)	2789	(1)	2816	(1)	2819	(1)
		2820	(1)	2912	(1)	2976	(1)	3158	(1)	3343	(1)	3607	(1)	3973	(1)
		3992	(1)	4012	(1)	4015	(1)	4025	(1)	4091	(1)	4098	(1)		
PUSHAL	00DF	2099	(1)	2126	(1)	2127	(1)	2353	(1)	2729	(1)				
PUSHAQ	007F	2128	(1)	2601	(1)	3104	(1)	3118	(1)						
PUSHL	00DD	2098	(1)	2107	(1)	2150	(1)	2151	(1)	2161	(1)	2162	(1)	2353	(1)

		2450	(1)	2451	(1)	2490	(1)	2561	(1)	2562	(1)	2566	(1)	2618	(1)
		2619	(1)	2706	(1)	2707	(1)	2729	(1)	2770	(1)	2771	(1)	2790	(1)
		2791	(1)	2821	(1)	2822	(1)	2963	(1)	2964	(1)	2965	(1)	2966	(1)
		2977	(1)	3158	(1)	3343	(1)	3608	(1)	3609	(1)	3974	(1)	3975	(1)
		3993	(1)	3994	(1)	4026	(1)	4027	(1)	4092	(1)	4093	(1)	4099	(1)
		4100	(1)												
PUSHR	00BB	2702	(1)	2836	(1)	3084	(1)	3892	(1)	4016	(1)				
RET	0004	2494	(1)												
RSB	0005	2354	(1)	2363	(1)	2390	(1)	2398	(1)	2405	(1)	2413	(1)	2421	(1)
		2431	(1)	2439	(1)	2446	(1)	2453	(1)	2460	(1)	2470	(1)	2479	(1)
		2503	(1)	2511	(1)	2517	(1)	2528	(1)	2535	(1)	2545	(1)	2549	(1)
		2557	(1)	2564	(1)	2568	(1)	2578	(1)	2586	(1)	2614	(1)	2621	(1)
		2624	(1)	2635	(1)	2647	(1)	2665	(1)	2676	(1)	2683	(1)	2693	(1)
		2802	(1)	2811	(1)	2817	(1)	2824	(1)	2852	(1)	2861	(1)	2879	(1)
		2887	(1)	2895	(1)	2979	(1)	2989	(1)	2997	(1)	3005	(1)	3013	(1)
		3023	(1)	3030	(1)	3038	(1)	3047	(1)	3054	(1)	3061	(1)	3068	(1)
		3077	(1)	3123	(1)	3126	(1)	3144	(1)	3161	(1)	3181	(1)	3195	(1)
		3207	(1)	3210	(1)	3219	(1)	3231	(1)	3236	(1)	3239	(1)	3250	(1)
		3261	(1)	3272	(1)	3280	(1)	3293	(1)	3297	(1)	3301	(1)	3313	(1)
		3347	(1)	3367	(1)	3375	(1)	3383	(1)	3391	(1)	3399	(1)	3407	(1)
		3415	(1)	3423	(1)	3433	(1)	3441	(1)	3449	(1)	3457	(1)	3467	(1)
		3475	(1)	3483	(1)	3497	(1)	3500	(1)	3502	(1)	3514	(1)	3517	(1)
		3519	(1)	3561	(1)	3565	(1)	3567	(1)	3586	(1)	3591	(1)	3593	(1)
		3615	(1)	3622	(1)	3630	(1)	3640	(1)	3644	(1)	3647	(1)	3650	(1)
		3663	(1)	3681	(1)	3690	(1)	3696	(1)	3705	(1)	3716	(1)	3724	(1)
		3732	(1)	3740	(1)	3748	(1)	3756	(1)	3776	(1)	3790	(1)	3804	(1)
		3818	(1)	3832	(1)	3835	(1)	3849	(1)	3864	(1)	3869	(1)	3871	(1)
		3882	(1)	3885	(1)	3903	(1)	3913	(1)	3924	(1)	3935	(1)	3938	(1)
		3947	(1)	3954	(1)	3961	(1)	3970	(1)	3977	(1)	3981	(1)	3989	(1)
		3996	(1)	4000	(1)	4007	(1)	4013	(1)	4029	(1)	4038	(1)	4045	(1)
		4052	(1)	4057	(1)	4064	(1)	4078	(1)	4087	(1)	4095	(1)	4102	(1)
		4106	(1)												
SOBGTR	00F5	2067	(1)												
SUBL2	00C2	3098	(1)												
SUBL3	00C3	2388	(1)	2995	(1)	3044	(1)	3088	(1)	3919	(1)				
TSTB	0095	4020	(1)												
TSTL	00D5	2073	(1)	2113	(1)	2481	(1)	3270	(1)	3601	(1)	3773	(1)	3801	(1)
		3815	(1)												
TSTW	00B5	3097	(1)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	1035	(1)	1040	(1)	1043	(1)	1047	(1)	1050	(1)	1053	(1)	1070	(1)	1073	(1)
	1076	(1)	1079	(1)	1084	(1)	1087	(1)	1090	(1)	1115	(1)	1118	(1)	1255	(1)
	1292	(1)	1297	(1)	1304	(1)	1311	(1)	1318	(1)	1327	(1)	1348	(1)	1363	(1)
	1368	(1)	1379	(1)	1385	(1)	1407	(1)	1411	(1)	1416	(1)	1419	(1)	1422	(1)
	1425	(1)	1429	(1)	1432	(1)	1437	(1)	1440	(1)	1453	(1)	1475	(1)	1490	(1)
	1496	(1)	1540	(1)	1545	(1)	1551	(1)	1557	(1)	1563	(1)	1575	(1)	1580	(1)
	1587	(1)	1592	(1)	1594	(1)	1599	(1)	1604	(1)	1614	(1)	1617	(1)	1620	(1)
	1623	(1)	1626	(1)	1631	(1)	1636	(1)	1643	(1)	1648	(1)	1653	(1)	1658	(1)
	1663	(1)	1668	(1)	1675	(1)	1680	(1)	1682	(1)	1687	(1)	1692	(1)	1697	(1)
	1702	(1)	1710	(1)	1713	(1)	1738	(1)	1743	(1)	1756	(1)	1758	(1)	1761	(1)
	1763	(1)	1769	(1)	1775	(1)	1808	(1)	1815	(1)	1827	(1)	1837	(1)	1839	(1)
	1842	(1)	1844	(1)	1849	(1)	1855	(1)	1861	(1)	1868	(1)	1879	(1)	1887	(1)
	1890	(1)	1893	(1)	1899	(1)	1916	(1)	1919	(1)	1922	(1)	1925	(1)	1928	(1)
	1931	(1)	1943	(1)	1970	(1)	1975	(1)	1980	(1)	1985	(1)	1990	(1)	1995	(1)
	2518	(1)	610	(1)	611	(1)	612	(1)	630	(1)	635	(1)	638	(1)	641	(1)
	644	(1)	647	(1)	650	(1)	653	(1)	656	(1)	659	(1)	663	(1)	666	(1)
	669	(1)	672	(1)	675	(1)	678	(1)	681	(1)	684	(1)	687	(1)	690	(1)
	708	(1)	713	(1)	719	(1)	735	(1)	740	(1)	745	(1)	763	(1)	765	(1)
	772	(1)	783	(1)	788	(1)	796	(1)	803	(1)	824	(1)	829	(1)	840	(1)
	848	(1)	853	(1)	860	(1)	867	(1)	894	(1)	899	(1)	904	(1)	910	(1)
	917	(1)														
.ASCII	608	(1)	621	(1)												
.BLKB	598	(1)	604	(1)												
.BLKL	579	(1)														
.BLKQ	601	(1)														
.BYTE	1000	(1)	1001	(1)	1002	(1)	1016	(1)	1017	(1)	1019	(1)	1020	(1)	1022	(1)
	1023	(1)	1024	(1)	1035	(1)	1036	(1)	1039	(1)	1040	(1)	1041	(1)	1043	(1)
	1044	(1)	1046	(1)	1047	(1)	1048	(1)	1050	(1)	1051	(1)	1053	(1)	1054	(1)
	1056	(1)	1057	(1)	1058	(1)	1059	(1)	1061	(1)	1062	(1)	1064	(1)	1065	(1)
	1067	(1)	1068	(1)	1070	(1)	1071	(1)	1073	(1)	1074	(1)	1076	(1)	1077	(1)
	1079	(1)	1080	(1)	1084	(1)	1085	(1)	1087	(1)	1088	(1)	1090	(1)	1093	(1)
	1094	(1)	1100	(1)	1101	(1)	1102	(1)	1103	(1)	1104	(1)	1105	(1)	1107	(1)
	1108	(1)	1114	(1)	1115	(1)	1116	(1)	1118	(1)	1119	(1)	1121	(1)	1122	(1)
	1123	(1)	1136	(1)	1137	(1)	1139	(1)	1140	(1)	1142	(1)	1143	(1)	1144	(1)
	1145	(1)	1146	(1)	1148	(1)	1149	(1)	1151	(1)	1152	(1)	1153	(1)	1155	(1)
	1157	(1)	1159	(1)	1160	(1)	1162	(1)	1163	(1)	1165	(1)	1166	(1)	1167	(1)
	1168	(1)	1172	(1)	1173	(1)	1174	(1)	1175	(1)	1176	(1)	1177	(1)	1178	(1)
	1179	(1)	1180	(1)	1181	(1)	1182	(1)	1183	(1)	1184	(1)	1185	(1)	1186	(1)
	1187	(1)	1188	(1)	1189	(1)	1190	(1)	1191	(1)	1192	(1)	1193	(1)	1194	(1)
	1195	(1)	1196	(1)	1197	(1)	1198	(1)	1199	(1)	1200	(1)	1201	(1)	1202	(1)
	1203	(1)	1204	(1)	1205	(1)	1206	(1)	1211	(1)	1212	(1)	1213	(1)	1214	(1)
	1215	(1)	1216	(1)	1217	(1)	1218	(1)	1219	(1)	1220	(1)	1221	(1)	1222	(1)
	1223	(1)	1224	(1)	1236	(1)	1237	(1)	1238	(1)	1243	(1)	1244	(1)	1245	(1)
	1249	(1)	1250	(1)	1251	(1)	1255	(1)	1256	(1)	1260	(1)	1261	(1)	1265	(1)
	1266	(1)	1267	(1)	1271	(1)	1272	(1)	1273	(1)	1277	(1)	1278	(1)	1279	(1)
	1288	(1)	1292	(1)	1293	(1)	1297	(1)	1298	(1)	1299	(1)	1300	(1)	1304	(1)
	1305	(1)	1309	(1)	1311	(1)	1312	(1)	1313	(1)	1314	(1)	1318	(1)	1319	(1)
	1320	(1)	1321	(1)	1325	(1)	1327	(1)	1328	(1)	1329	(1)	1331	(1)	1332	(1)
	1333	(1)	1334	(1)	1335	(1)	1336	(1)	1337	(1)	1338	(1)	1339	(1)	1340	(1)
	1342	(1)	1348	(1)	1349	(1)	1350	(1)	1351	(1)	1352	(1)	1353	(1)	1355	(1)

	1361	(1)	1363	(1)	1364	(1)	1365	(1)	1366	(1)	1368	(1)	1369	(1)	1371	(1)
	1377	(1)	1379	(1)	1380	(1)	1381	(1)	1382	(1)	1383	(1)	1385	(1)	1386	(1)
	1388	(1)	1405	(1)	1407	(1)	1408	(1)	1410	(1)	1411	(1)	1412	(1)	1413	(1)
	1414	(1)	1416	(1)	1417	(1)	1419	(1)	1420	(1)	1422	(1)	1423	(1)	1425	(1)
	1426	(1)	1428	(1)	1429	(1)	1430	(1)	1432	(1)	1433	(1)	1434	(1)	1435	(1)
	1437	(1)	1438	(1)	1440	(1)	1441	(1)	1443	(1)	1451	(1)	1453	(1)	1454	(1)
	1455	(1)	1456	(1)	1457	(1)	1459	(1)	1460	(1)	1461	(1)	1462	(1)	1463	(1)
	1465	(1)	1466	(1)	1467	(1)	1468	(1)	1469	(1)	1474	(1)	1475	(1)	1476	(1)
	1477	(1)	1478	(1)	1480	(1)	1488	(1)	1490	(1)	1491	(1)	1492	(1)	1493	(1)
	1494	(1)	1496	(1)	1497	(1)	1499	(1)	1535	(1)	1536	(1)	1540	(1)	1541	(1)
	1545	(1)	1546	(1)	1550	(1)	1551	(1)	1552	(1)	1557	(1)	1558	(1)	1563	(1)
	1564	(1)	1570	(1)	1571	(1)	1575	(1)	1576	(1)	1580	(1)	1581	(1)	1582	(1)
	1586	(1)	1587	(1)	1588	(1)	1592	(1)	1593	(1)	1594	(1)	1595	(1)	1599	(1)
	1600	(1)	1604	(1)	1605	(1)	1610	(1)	1614	(1)	1615	(1)	1617	(1)	1618	(1)
	1620	(1)	1621	(1)	1623	(1)	1624	(1)	1626	(1)	1627	(1)	1631	(1)	1632	(1)
	1636	(1)	1637	(1)	1641	(1)	1642	(1)	1643	(1)	1644	(1)	1648	(1)	1649	(1)
	1653	(1)	1654	(1)	1658	(1)	1659	(1)	1663	(1)	1664	(1)	1668	(1)	1669	(1)
	1673	(1)	1674	(1)	1675	(1)	1676	(1)	1680	(1)	1681	(1)	1682	(1)	1683	(1)
	1687	(1)	1688	(1)	1692	(1)	1693	(1)	1697	(1)	1698	(1)	1702	(1)	1703	(1)
	1707	(1)	1708	(1)	1710	(1)	1711	(1)	1713	(1)	1714	(1)	1732	(1)	1733	(1)
	1738	(1)	1739	(1)	1740	(1)	1742	(1)	1743	(1)	1744	(1)	1746	(1)	1747	(1)
	1749	(1)	1750	(1)	1752	(1)	1756	(1)	1757	(1)	1758	(1)	1759	(1)	1761	(1)
	1762	(1)	1763	(1)	1764	(1)	1768	(1)	1769	(1)	1770	(1)	1775	(1)	1776	(1)
	1780	(1)	1803	(1)	1804	(1)	1808	(1)	1809	(1)	1810	(1)	1811	(1)	1815	(1)
	1816	(1)	1817	(1)	1818	(1)	1819	(1)	1820	(1)	1822	(1)	1826	(1)	1827	(1)
	1828	(1)	1829	(1)	1830	(1)	1831	(1)	1832	(1)	1837	(1)	1838	(1)	1839	(1)
	1840	(1)	1842	(1)	1843	(1)	1844	(1)	1845	(1)	1849	(1)	1850	(1)	1854	(1)
	1855	(1)	1856	(1)	1861	(1)	1862	(1)	1866	(1)	1867	(1)	1868	(1)	1869	(1)
	1870	(1)	1871	(1)	1873	(1)	1874	(1)	1878	(1)	1879	(1)	1880	(1)	1881	(1)
	1882	(1)	1887	(1)	1888	(1)	1889	(1)	1890	(1)	1891	(1)	1893	(1)	1894	(1)
	1898	(1)	1899	(1)	1900	(1)	1901	(1)	1902	(1)	1904	(1)	1913	(1)	1914	(1)
	1916	(1)	1917	(1)	1919	(1)	1920	(1)	1922	(1)	1923	(1)	1925	(1)	1926	(1)
	1928	(1)	1929	(1)	1931	(1)	1932	(1)	1934	(1)	1935	(1)	1936	(1)	1938	(1)
	1943	(1)	1944	(1)	1945	(1)	1946	(1)	1950	(1)	1965	(1)	1970	(1)	1971	(1)
	1975	(1)	1976	(1)	1980	(1)	1981	(1)	1985	(1)	1986	(1)	1990	(1)	1991	(1)
	1995	(1)	1996	(1)	2001	(1)	2002	(1)	2006	(1)	616	(1)	620	(1)	702	(1)
	704	(1)	708	(1)	709	(1)	713	(1)	714	(1)	719	(1)	720	(1)	721	(1)
	722	(1)	731	(1)	735	(1)	736	(1)	740	(1)	741	(1)	745	(1)	746	(1)
	759	(1)	763	(1)	764	(1)	765	(1)	766	(1)	767	(1)	768	(1)	772	(1)
	773	(1)	774	(1)	775	(1)	776	(1)	777	(1)	779	(1)	783	(1)	784	(1)
	788	(1)	789	(1)	790	(1)	791	(1)	792	(1)	796	(1)	797	(1)	803	(1)
	804	(1)	805	(1)	806	(1)	807	(1)	808	(1)	809	(1)	810	(1)	811	(1)
	818	(1)	819	(1)	824	(1)	825	(1)	829	(1)	830	(1)	831	(1)	832	(1)
	833	(1)	834	(1)	840	(1)	841	(1)	842	(1)	843	(1)	844	(1)	848	(1)
	849	(1)	853	(1)	854	(1)	855	(1)	856	(1)	860	(1)	861	(1)	862	(1)
	863	(1)	867	(1)	868	(1)	869	(1)	870	(1)	871	(1)	872	(1)	884	(1)
	888	(1)	889	(1)	890	(1)	894	(1)	895	(1)	899	(1)	900	(1)	904	(1)
	905	(1)	906	(1)	910	(1)	911	(1)	916	(1)	917	(1)	918	(1)	922	(1)
	923	(1)	924	(1)	925	(1)	935	(1)	936	(1)	937	(1)	939	(1)	940	(1)
	941	(1)	964	(1)	965	(1)	971	(1)	972	(1)	973	(1)	981	(1)	994	(1)
	995	(1)	997	(1)	998	(1)										
.DISABLE	3145	(1)	4079	(1)												
.ENABLE	3135	(1)	4065	(1)												
.END	4108	(1)														
.ENDC	1000	(1)	1001	(1)	1002	(1)	1016	(1)	1017	(1)	1019	(1)	1020	(1)	1022	(1)
	1023	(1)	1024	(1)	1035	(1)	1036	(1)	1039	(1)	1040	(1)	1041	(1)	1043	(1)
	1044	(1)	1046	(1)	1047	(1)	1048	(1)	1050	(1)	1051	(1)	1053	(1)	1054	(1)

1056 (1) 1057 (1) 1058 (1) 1059 (1) 1061 (1) 1062 (1) 1064 (1) 1065 (1)
1067 (1) 1068 (1) 1070 (1) 1071 (1) 1073 (1) 1074 (1) 1076 (1) 1077 (1)
1079 (1) 1080 (1) 1084 (1) 1085 (1) 1087 (1) 1088 (1) 1090 (1) 1093 (1)
1094 (1) 1100 (1) 1101 (1) 1102 (1) 1103 (1) 1104 (1) 1105 (1) 1107 (1)
1108 (1) 1114 (1) 1115 (1) 1116 (1) 1118 (1) 1119 (1) 1121 (1) 1122 (1)
1123 (1) 1136 (1) 1137 (1) 1139 (1) 1140 (1) 1142 (1) 1143 (1) 1144 (1)
1145 (1) 1146 (1) 1148 (1) 1149 (1) 1151 (1) 1152 (1) 1153 (1) 1155 (1)
1157 (1) 1159 (1) 1160 (1) 1162 (1) 1163 (1) 1165 (1) 1166 (1) 1167 (1)
1168 (1) 1172 (1) 1173 (1) 1174 (1) 1175 (1) 1176 (1) 1177 (1) 1178 (1)
1179 (1) 1180 (1) 1181 (1) 1182 (1) 1183 (1) 1184 (1) 1185 (1) 1186 (1)
1187 (1) 1188 (1) 1189 (1) 1190 (1) 1191 (1) 1192 (1) 1193 (1) 1194 (1)
1195 (1) 1196 (1) 1197 (1) 1198 (1) 1199 (1) 1200 (1) 1201 (1) 1202 (1)
1203 (1) 1204 (1) 1205 (1) 1206 (1) 1211 (1) 1212 (1) 1213 (1) 1214 (1)
1215 (1) 1216 (1) 1217 (1) 1218 (1) 1219 (1) 1220 (1) 1221 (1) 1222 (1)
1223 (1) 1224 (1) 1236 (1) 1237 (1) 1238 (1) 1243 (1) 1244 (1) 1245 (1)
1249 (1) 1250 (1) 1251 (1) 1255 (1) 1256 (1) 1260 (1) 1261 (1) 1265 (1)
1266 (1) 1267 (1) 1271 (1) 1272 (1) 1273 (1) 1277 (1) 1278 (1) 1279 (1)
1288 (1) 1292 (1) 1293 (1) 1297 (1) 1298 (1) 1299 (1) 1300 (1) 1304 (1)
1305 (1) 1309 (1) 1311 (1) 1312 (1) 1313 (1) 1314 (1) 1318 (1) 1319 (1)
1320 (1) 1321 (1) 1325 (1) 1327 (1) 1328 (1) 1329 (1) 1331 (1) 1332 (1)
1333 (1) 1334 (1) 1335 (1) 1336 (1) 1337 (1) 1338 (1) 1339 (1) 1340 (1)
1342 (1) 1348 (1) 1349 (1) 1350 (1) 1351 (1) 1352 (1) 1353 (1) 1355 (1)
1361 (1) 1363 (1) 1364 (1) 1365 (1) 1366 (1) 1368 (1) 1369 (1) 1371 (1)
1377 (1) 1379 (1) 1380 (1) 1381 (1) 1382 (1) 1383 (1) 1385 (1) 1386 (1)
1388 (1) 1405 (1) 1407 (1) 1408 (1) 1410 (1) 1411 (1) 1412 (1) 1413 (1)
1414 (1) 1416 (1) 1417 (1) 1419 (1) 1420 (1) 1422 (1) 1423 (1) 1425 (1)
1426 (1) 1428 (1) 1429 (1) 1430 (1) 1432 (1) 1433 (1) 1434 (1) 1435 (1)
1437 (1) 1438 (1) 1440 (1) 1441 (1) 1443 (1) 1451 (1) 1453 (1) 1454 (1)
1455 (1) 1456 (1) 1457 (1) 1459 (1) 1460 (1) 1461 (1) 1462 (1) 1463 (1)
1465 (1) 1466 (1) 1467 (1) 1468 (1) 1469 (1) 1474 (1) 1475 (1) 1476 (1)
1477 (1) 1478 (1) 1480 (1) 1488 (1) 1490 (1) 1491 (1) 1492 (1) 1493 (1)
1494 (1) 1496 (1) 1497 (1) 1499 (1) 1535 (1) 1536 (1) 1540 (1) 1541 (1)
1545 (1) 1546 (1) 1550 (1) 1551 (1) 1552 (1) 1557 (1) 1558 (1) 1563 (1)
1564 (1) 1570 (1) 1571 (1) 1575 (1) 1576 (1) 1580 (1) 1581 (1) 1582 (1)
1586 (1) 1587 (1) 1588 (1) 1592 (1) 1593 (1) 1594 (1) 1595 (1) 1599 (1)
1600 (1) 1604 (1) 1605 (1) 1610 (1) 1614 (1) 1615 (1) 1617 (1) 1618 (1)
1620 (1) 1621 (1) 1623 (1) 1624 (1) 1626 (1) 1627 (1) 1631 (1) 1632 (1)
1636 (1) 1637 (1) 1641 (1) 1642 (1) 1643 (1) 1644 (1) 1648 (1) 1649 (1)
1653 (1) 1654 (1) 1658 (1) 1659 (1) 1663 (1) 1664 (1) 1668 (1) 1669 (1)
1673 (1) 1674 (1) 1675 (1) 1676 (1) 1680 (1) 1681 (1) 1682 (1) 1683 (1)
1687 (1) 1688 (1) 1692 (1) 1693 (1) 1697 (1) 1698 (1) 1702 (1) 1703 (1)
1707 (1) 1708 (1) 1710 (1) 1711 (1) 1713 (1) 1714 (1) 1732 (1) 1733 (1)
1738 (1) 1739 (1) 1740 (1) 1742 (1) 1743 (1) 1744 (1) 1746 (1) 1747 (1)
1749 (1) 1750 (1) 1752 (1) 1756 (1) 1757 (1) 1758 (1) 1759 (1) 1761 (1)
1762 (1) 1763 (1) 1764 (1) 1768 (1) 1769 (1) 1770 (1) 1775 (1) 1776 (1)
1780 (1) 1803 (1) 1804 (1) 1808 (1) 1809 (1) 1810 (1) 1811 (1) 1815 (1)
1816 (1) 1817 (1) 1818 (1) 1819 (1) 1820 (1) 1822 (1) 1826 (1) 1827 (1)
1828 (1) 1829 (1) 1830 (1) 1831 (1) 1832 (1) 1837 (1) 1838 (1) 1839 (1)
1840 (1) 1842 (1) 1843 (1) 1844 (1) 1845 (1) 1849 (1) 1850 (1) 1854 (1)
1855 (1) 1856 (1) 1861 (1) 1862 (1) 1866 (1) 1867 (1) 1868 (1) 1869 (1)
1870 (1) 1871 (1) 1873 (1) 1874 (1) 1878 (1) 1879 (1) 1880 (1) 1881 (1)
1882 (1) 1887 (1) 1888 (1) 1889 (1) 1890 (1) 1891 (1) 1893 (1) 1894 (1)
1898 (1) 1899 (1) 1900 (1) 1901 (1) 1902 (1) 1904 (1) 1913 (1) 1914 (1)
1916 (1) 1917 (1) 1919 (1) 1920 (1) 1922 (1) 1923 (1) 1925 (1) 1926 (1)
1928 (1) 1929 (1) 1931 (1) 1932 (1) 1934 (1) 1935 (1) 1936 (1) 1938 (1)
1943 (1) 1944 (1) 1945 (1) 1946 (1) 1950 (1) 1965 (1) 1970 (1) 1971 (1)
1975 (1) 1976 (1) 1980 (1) 1981 (1) 1985 (1) 1986 (1) 1990 (1) 1991 (1)

```
                1995  (1)   1996  (1)   2001  (1)   2002  (1)   2006  (1)   2057  (1)   2060  (1)   2063  (1)
                2081  (1)   2353  (1)   2485  (1)   2490  (1)   2816  (1)   2912  (1)   2976  (1)   3104  (1)
                3118  (1)   3158  (1)   3343  (1)   4012  (1)   575   (1)   578   (1)   580   (1)   584   (1)
                585   (1)   586   (1)   587   (1)   702   (1)   704   (1)   708   (1)   709   (1)   713   (1)
                714   (1)   719   (1)   720   (1)   721   (1)   722   (1)   731   (1)   735   (1)   736   (1)
                740   (1)   741   (1)   745   (1)   746   (1)   759   (1)   763   (1)   764   (1)   765   (1)
                766   (1)   767   (1)   768   (1)   772   (1)   773   (1)   774   (1)   775   (1)   776   (1)
                777   (1)   779   (1)   783   (1)   784   (1)   788   (1)   789   (1)   790   (1)   791   (1)
                792   (1)   796   (1)   797   (1)   803   (1)   804   (1)   805   (1)   806   (1)   807   (1)
                808   (1)   809   (1)   810   (1)   811   (1)   818   (1)   819   (1)   824   (1)   825   (1)
                829   (1)   830   (1)   831   (1)   832   (1)   833   (1)   834   (1)   840   (1)   841   (1)
                842   (1)   843   (1)   844   (1)   848   (1)   849   (1)   853   (1)   854   (1)   855   (1)
                856   (1)   860   (1)   861   (1)   862   (1)   863   (1)   867   (1)   868   (1)   869   (1)
                870   (1)   871   (1)   872   (1)   884   (1)   888   (1)   889   (1)   890   (1)   894   (1)
                895   (1)   899   (1)   900   (1)   904   (1)   905   (1)   906   (1)   910   (1)   911   (1)
                916   (1)   917   (1)   918   (1)   922   (1)   923   (1)   924   (1)   925   (1)   935   (1)
                936   (1)   937   (1)   939   (1)   940   (1)   941   (1)   964   (1)   965   (1)   971   (1)
                972   (1)   973   (1)   981   (1)   994   (1)   995   (1)   997   (1)   998   (1)
.ENDM           2069  (1)   2078  (1)   2079  (1)   2081  (1)   2087  (1)   2088  (1)   2089  (1)   2095  (1)
                2107  (1)   2353  (1)   2448  (1)   2459  (1)   2559  (1)   2728  (1)   2940  (1)   3104  (1)
                569   (1)   575   (1)   576   (1)   577   (1)   578   (1)   579   (1)   580   (1)   581   (1)
                582   (1)   583   (1)   584   (1)   585   (1)   586   (1)   587   (1)   588   (1)   630   (1)
                702   (1)
.ENDR           2081  (1)   2485  (1)   2490  (1)   2816  (1)   2912  (1)   2976  (1)   3158  (1)   3343  (1)
                4012  (1)   575   (1)   578   (1)   580   (1)   584   (1)   585   (1)   586   (1)   587   (1)
.ENTRY          2050  (1)
.EXTRN          557   (1)
.GLOBL          2107  (1)   3104  (1)   3118  (1)
.IDENT          33    (1)
.IF             1000  (1)   1001  (1)   1002  (1)   1016  (1)   1017  (1)   1019  (1)   1020  (1)   1022  (1)
                1023  (1)   1024  (1)   1035  (1)   1036  (1)   1039  (1)   1040  (1)   1041  (1)   1043  (1)
                1044  (1)   1046  (1)   1047  (1)   1048  (1)   1050  (1)   1051  (1)   1053  (1)   1054  (1)
                1056  (1)   1057  (1)   1058  (1)   1059  (1)   1061  (1)   1062  (1)   1064  (1)   1065  (1)
                1067  (1)   1068  (1)   1070  (1)   1071  (1)   1073  (1)   1074  (1)   1076  (1)   1077  (1)
                1079  (1)   1080  (1)   1084  (1)   1085  (1)   1087  (1)   1088  (1)   1090  (1)   1093  (1)
                1094  (1)   1100  (1)   1101  (1)   1102  (1)   1103  (1)   1104  (1)   1105  (1)   1107  (1)
                1108  (1)   1114  (1)   1115  (1)   1116  (1)   1118  (1)   1119  (1)   1121  (1)   1122  (1)
                1123  (1)   1136  (1)   1137  (1)   1139  (1)   1140  (1)   1142  (1)   1143  (1)   1144  (1)
                1145  (1)   1146  (1)   1148  (1)   1149  (1)   1151  (1)   1152  (1)   1153  (1)   1155  (1)
                1157  (1)   1159  (1)   1160  (1)   1162  (1)   1163  (1)   1165  (1)   1166  (1)   1167  (1)
                1168  (1)   1172  (1)   1173  (1)   1174  (1)   1175  (1)   1176  (1)   1177  (1)   1178  (1)
                1179  (1)   1180  (1)   1181  (1)   1182  (1)   1183  (1)   1184  (1)   1185  (1)   1186  (1)
                1187  (1)   1188  (1)   1189  (1)   1190  (1)   1191  (1)   1192  (1)   1193  (1)   1194  (1)
                1195  (1)   1196  (1)   1197  (1)   1198  (1)   1199  (1)   1200  (1)   1201  (1)   1202  (1)
                1203  (1)   1204  (1)   1205  (1)   1206  (1)   1211  (1)   1212  (1)   1213  (1)   1214  (1)
                1215  (1)   1216  (1)   1217  (1)   1218  (1)   1219  (1)   1220  (1)   1221  (1)   1222  (1)
                1223  (1)   1224  (1)   1236  (1)   1237  (1)   1238  (1)   1243  (1)   1244  (1)   1245  (1)
                1249  (1)   1250  (1)   1251  (1)   1255  (1)   1256  (1)   1260  (1)   1261  (1)   1265  (1)
                1266  (1)   1267  (1)   1271  (1)   1272  (1)   1273  (1)   1277  (1)   1278  (1)   1279  (1)
                1288  (1)   1292  (1)   1293  (1)   1297  (1)   1298  (1)   1299  (1)   1300  (1)   1304  (1)
                1305  (1)   1309  (1)   1311  (1)   1312  (1)   1313  (1)   1314  (1)   1318  (1)   1319  (1)
                1320  (1)   1321  (1)   1325  (1)   1327  (1)   1328  (1)   1329  (1)   1331  (1)   1332  (1)
                1333  (1)   1334  (1)   1335  (1)   1336  (1)   1337  (1)   1338  (1)   1339  (1)   1340  (1)
                1342  (1)   1348  (1)   1349  (1)   1350  (1)   1351  (1)   1352  (1)   1353  (1)   1355  (1)
                1361  (1)   1363  (1)   1364  (1)   1365  (1)   1366  (1)   1368  (1)   1369  (1)   1371  (1)
                1377  (1)   1379  (1)   1380  (1)   1381  (1)   1382  (1)   1383  (1)   1385  (1)   1386  (1)
                1388  (1)   1405  (1)   1407  (1)   1408  (1)   1410  (1)   1411  (1)   1412  (1)   1413  (1)
```

```
                1414  (1)     1416  (1)     1417  (1)     1419  (1)     1420  (1)     1422  (1)     1423  (1)     1425  (1)
                1426  (1)     1428  (1)     1429  (1)     1430  (1)     1432  (1)     1433  (1)     1434  (1)     1435  (1)
                1437  (1)     1438  (1)     1440  (1)     1441  (1)     1443  (1)     1451  (1)     1453  (1)     1454  (1)
                1455  (1)     1456  (1)     1457  (1)     1459  (1)     1460  (1)     1461  (1)     1462  (1)     1463  (1)
                1465  (1)     1466  (1)     1467  (1)     1468  (1)     1469  (1)     1474  (1)     1475  (1)     1476  (1)
                1477  (1)     1478  (1)     1480  (1)     1488  (1)     1490  (1)     1491  (1)     1492  (1)     1493  (1)
                1494  (1)     1496  (1)     1497  (1)     1499  (1)     1535  (1)     1536  (1)     1540  (1)     1541  (1)
                1545  (1)     1546  (1)     1550  (1)     1551  (1)     1552  (1)     1557  (1)     1558  (1)     1563  (1)
                1564  (1)     1570  (1)     1571  (1)     1575  (1)     1576  (1)     1580  (1)     1581  (1)     1582  (1)
                1586  (1)     1587  (1)     1588  (1)     1592  (1)     1593  (1)     1594  (1)     1595  (1)     1599  (1)
                1600  (1)     1604  (1)     1605  (1)     1610  (1)     1614  (1)     1615  (1)     1617  (1)     1618  (1)
                1620  (1)     1621  (1)     1623  (1)     1624  (1)     1626  (1)     1627  (1)     1631  (1)     1632  (1)
                1636  (1)     1637  (1)     1641  (1)     1642  (1)     1643  (1)     1644  (1)     1648  (1)     1649  (1)
                1653  (1)     1654  (1)     1658  (1)     1659  (1)     1663  (1)     1664  (1)     1668  (1)     1669  (1)
                1673  (1)     1674  (1)     1675  (1)     1676  (1)     1680  (1)     1681  (1)     1682  (1)     1683  (1)
                1687  (1)     1688  (1)     1692  (1)     1693  (1)     1697  (1)     1698  (1)     1702  (1)     1703  (1)
                1707  (1)     1708  (1)     1710  (1)     1711  (1)     1713  (1)     1714  (1)     1732  (1)     1733  (1)
                1738  (1)     1739  (1)     1740  (1)     1742  (1)     1743  (1)     1744  (1)     1746  (1)     1747  (1)
                1749  (1)     1750  (1)     1752  (1)     1756  (1)     1757  (1)     1758  (1)     1759  (1)     1761  (1)
                1762  (1)     1763  (1)     1764  (1)     1768  (1)     1769  (1)     1770  (1)     1775  (1)     1776  (1)
                1780  (1)     1803  (1)     1804  (1)     1808  (1)     1809  (1)     1810  (1)     1811  (1)     1815  (1)
                1816  (1)     1817  (1)     1818  (1)     1819  (1)     1820  (1)     1822  (1)     1826  (1)     1827  (1)
                1828  (1)     1829  (1)     1830  (1)     1831  (1)     1832  (1)     1837  (1)     1838  (1)     1839  (1)
                1840  (1)     1842  (1)     1843  (1)     1844  (1)     1845  (1)     1849  (1)     1850  (1)     1854  (1)
                1855  (1)     1856  (1)     1861  (1)     1862  (1)     1866  (1)     1867  (1)     1868  (1)     1869  (1)
                1870  (1)     1871  (1)     1873  (1)     1874  (1)     1878  (1)     1879  (1)     1880  (1)     1881  (1)
                1882  (1)     1887  (1)     1888  (1)     1889  (1)     1890  (1)     1891  (1)     1893  (1)     1894  (1)
                1898  (1)     1899  (1)     1900  (1)     1901  (1)     1902  (1)     1904  (1)     1913  (1)     1914  (1)
                1916  (1)     1917  (1)     1919  (1)     1920  (1)     1922  (1)     1923  (1)     1925  (1)     1926  (1)
                1928  (1)     1929  (1)     1931  (1)     1932  (1)     1934  (1)     1935  (1)     1936  (1)     1938  (1)
                1943  (1)     1944  (1)     1945  (1)     1946  (1)     1950  (1)     1965  (1)     1970  (1)     1971  (1)
                1975  (1)     1976  (1)     1980  (1)     1981  (1)     1985  (1)     1986  (1)     1990  (1)     1991  (1)
                1995  (1)     1996  (1)     2001  (1)     2002  (1)     2006  (1)     2055  (1)     2058  (1)     2061  (1)
                2081  (1)     2353  (1)     2485  (1)     2490  (1)     2816  (1)     2912  (1)     2976  (1)     3104  (1)
                3118  (1)     3158  (1)     3343  (1)     4012  (1)     575   (1)     578   (1)     580   (1)     584   (1)
                585   (1)     586   (1)     587   (1)     702   (1)     704   (1)     708   (1)     709   (1)     713   (1)
                714   (1)     719   (1)     720   (1)     721   (1)     722   (1)     731   (1)     735   (1)     736   (1)
                740   (1)     741   (1)     745   (1)     746   (1)     759   (1)     763   (1)     764   (1)     765   (1)
                766   (1)     767   (1)     768   (1)     772   (1)     773   (1)     774   (1)     775   (1)     776   (1)
                777   (1)     779   (1)     783   (1)     784   (1)     788   (1)     789   (1)     790   (1)     791   (1)
                792   (1)     796   (1)     797   (1)     803   (1)     804   (1)     805   (1)     806   (1)     807   (1)
                808   (1)     809   (1)     810   (1)     811   (1)     818   (1)     819   (1)     824   (1)     825   (1)
                829   (1)     830   (1)     831   (1)     832   (1)     833   (1)     834   (1)     840   (1)     841   (1)
                842   (1)     843   (1)     844   (1)     848   (1)     849   (1)     853   (1)     854   (1)     855   (1)
                856   (1)     860   (1)     861   (1)     862   (1)     863   (1)     867   (1)     868   (1)     869   (1)
                870   (1)     871   (1)     872   (1)     884   (1)     888   (1)     889   (1)     890   (1)     894   (1)
                895   (1)     899   (1)     900   (1)     904   (1)     905   (1)     906   (1)     910   (1)     911   (1)
                916   (1)     917   (1)     918   (1)     922   (1)     923   (1)     924   (1)     925   (1)     935   (1)
                936   (1)     937   (1)     939   (1)     940   (1)     941   (1)     964   (1)     965   (1)     971   (1)
                972   (1)     973   (1)     981   (1)     994   (1)     995   (1)     997   (1)     998   (1)
.IFF            1000  (1)     1001  (1)     1002  (1)     1016  (1)     1017  (1)     1019  (1)     1020  (1)     1022  (1)
                1023  (1)     1024  (1)     1035  (1)     1036  (1)     1039  (1)     1040  (1)     1041  (1)     1043  (1)
                1044  (1)     1046  (1)     1047  (1)     1048  (1)     1050  (1)     1051  (1)     1053  (1)     1054  (1)
                1056  (1)     1057  (1)     1058  (1)     1059  (1)     1061  (1)     1062  (1)     1064  (1)     1065  (1)
                1067  (1)     1068  (1)     1070  (1)     1071  (1)     1073  (1)     1074  (1)     1076  (1)     1077  (1)
                1079  (1)     1080  (1)     1084  (1)     1085  (1)     1087  (1)     1088  (1)     1090  (1)     1093  (1)
                1094  (1)     1100  (1)     1101  (1)     1102  (1)     1103  (1)     1104  (1)     1105  (1)     1107  (1)
```

1108	(1)	1114	(1)	1115	(1)	1116	(1)	1118	(1)	1119	(1)	1121	(1)	1122	(1)
1123	(1)	1136	(1)	1137	(1)	1139	(1)	1140	(1)	1142	(1)	1143	(1)	1144	(1)
1145	(1)	1146	(1)	1148	(1)	1149	(1)	1151	(1)	1152	(1)	1153	(1)	1155	(1)
1157	(1)	1159	(1)	1160	(1)	1162	(1)	1163	(1)	1165	(1)	1166	(1)	1167	(1)
1168	(1)	1172	(1)	1173	(1)	1174	(1)	1175	(1)	1176	(1)	1177	(1)	1178	(1)
1179	(1)	1180	(1)	1181	(1)	1182	(1)	1183	(1)	1184	(1)	1185	(1)	1186	(1)
1187	(1)	1188	(1)	1189	(1)	1190	(1)	1191	(1)	1192	(1)	1193	(1)	1194	(1)
1195	(1)	1196	(1)	1197	(1)	1198	(1)	1199	(1)	1200	(1)	1201	(1)	1202	(1)
1203	(1)	1204	(1)	1205	(1)	1206	(1)	1211	(1)	1212	(1)	1213	(1)	1214	(1)
1215	(1)	1216	(1)	1217	(1)	1218	(1)	1219	(1)	1220	(1)	1221	(1)	1222	(1)
1223	(1)	1224	(1)	1236	(1)	1237	(1)	1238	(1)	1243	(1)	1244	(1)	1245	(1)
1249	(1)	1250	(1)	1251	(1)	1255	(1)	1256	(1)	1260	(1)	1261	(1)	1265	(1)
1266	(1)	1267	(1)	1271	(1)	1272	(1)	1273	(1)	1277	(1)	1278	(1)	1279	(1)
1288	(1)	1292	(1)	1293	(1)	1297	(1)	1298	(1)	1299	(1)	1300	(1)	1304	(1)
1305	(1)	1309	(1)	1311	(1)	1312	(1)	1313	(1)	1314	(1)	1318	(1)	1319	(1)
1320	(1)	1321	(1)	1325	(1)	1327	(1)	1328	(1)	1329	(1)	1331	(1)	1332	(1)
1333	(1)	1334	(1)	1335	(1)	1336	(1)	1337	(1)	1338	(1)	1339	(1)	1340	(1)
1342	(1)	1348	(1)	1349	(1)	1350	(1)	1351	(1)	1352	(1)	1353	(1)	1355	(1)
1361	(1)	1363	(1)	1364	(1)	1365	(1)	1366	(1)	1368	(1)	1369	(1)	1371	(1)
1377	(1)	1379	(1)	1380	(1)	1381	(1)	1382	(1)	1383	(1)	1385	(1)	1386	(1)
1388	(1)	1405	(1)	1407	(1)	1408	(1)	1410	(1)	1411	(1)	1412	(1)	1413	(1)
1414	(1)	1416	(1)	1417	(1)	1419	(1)	1420	(1)	1422	(1)	1423	(1)	1425	(1)
1426	(1)	1428	(1)	1429	(1)	1430	(1)	1432	(1)	1433	(1)	1434	(1)	1435	(1)
1437	(1)	1438	(1)	1440	(1)	1441	(1)	1443	(1)	1451	(1)	1453	(1)	1454	(1)
1455	(1)	1456	(1)	1457	(1)	1459	(1)	1460	(1)	1461	(1)	1462	(1)	1463	(1)
1465	(1)	1466	(1)	1467	(1)	1468	(1)	1469	(1)	1474	(1)	1475	(1)	1476	(1)
1477	(1)	1478	(1)	1480	(1)	1488	(1)	1490	(1)	1491	(1)	1492	(1)	1493	(1)
1494	(1)	1496	(1)	1497	(1)	1499	(1)	1535	(1)	1536	(1)	1540	(1)	1541	(1)
1545	(1)	1546	(1)	1550	(1)	1551	(1)	1552	(1)	1557	(1)	1558	(1)	1563	(1)
1564	(1)	1570	(1)	1571	(1)	1575	(1)	1576	(1)	1580	(1)	1581	(1)	1582	(1)
1586	(1)	1587	(1)	1588	(1)	1592	(1)	1593	(1)	1594	(1)	1595	(1)	1599	(1)
1600	(1)	1604	(1)	1605	(1)	1610	(1)	1614	(1)	1615	(1)	1617	(1)	1618	(1)
1620	(1)	1621	(1)	1623	(1)	1624	(1)	1626	(1)	1627	(1)	1631	(1)	1632	(1)
1636	(1)	1637	(1)	1641	(1)	1642	(1)	1643	(1)	1644	(1)	1648	(1)	1649	(1)
1653	(1)	1654	(1)	1658	(1)	1659	(1)	1663	(1)	1664	(1)	1668	(1)	1669	(1)
1673	(1)	1674	(1)	1675	(1)	1676	(1)	1680	(1)	1681	(1)	1682	(1)	1683	(1)
1687	(1)	1688	(1)	1692	(1)	1693	(1)	1697	(1)	1698	(1)	1702	(1)	1703	(1)
1707	(1)	1708	(1)	1710	(1)	1711	(1)	1713	(1)	1714	(1)	1732	(1)	1733	(1)
1738	(1)	1739	(1)	1740	(1)	1742	(1)	1743	(1)	1744	(1)	1746	(1)	1747	(1)
1749	(1)	1750	(1)	1752	(1)	1756	(1)	1757	(1)	1758	(1)	1759	(1)	1761	(1)
1762	(1)	1763	(1)	1764	(1)	1768	(1)	1769	(1)	1770	(1)	1775	(1)	1776	(1)
1780	(1)	1803	(1)	1804	(1)	1808	(1)	1809	(1)	1810	(1)	1811	(1)	1815	(1)
1816	(1)	1817	(1)	1818	(1)	1819	(1)	1820	(1)	1822	(1)	1826	(1)	1827	(1)
1828	(1)	1829	(1)	1830	(1)	1831	(1)	1832	(1)	1837	(1)	1838	(1)	1839	(1)
1840	(1)	1842	(1)	1843	(1)	1844	(1)	1845	(1)	1849	(1)	1850	(1)	1854	(1)
1855	(1)	1856	(1)	1861	(1)	1862	(1)	1866	(1)	1867	(1)	1868	(1)	1869	(1)
1870	(1)	1871	(1)	1873	(1)	1874	(1)	1878	(1)	1879	(1)	1880	(1)	1881	(1)
1882	(1)	1887	(1)	1888	(1)	1889	(1)	1890	(1)	1891	(1)	1893	(1)	1894	(1)
1898	(1)	1899	(1)	1900	(1)	1901	(1)	1902	(1)	1904	(1)	1913	(1)	1914	(1)
1916	(1)	1917	(1)	1919	(1)	1920	(1)	1922	(1)	1923	(1)	1925	(1)	1926	(1)
1928	(1)	1929	(1)	1931	(1)	1932	(1)	1934	(1)	1935	(1)	1936	(1)	1938	(1)
1943	(1)	1944	(1)	1945	(1)	1946	(1)	1950	(1)	1965	(1)	1970	(1)	1971	(1)
1975	(1)	1976	(1)	1980	(1)	1981	(1)	1985	(1)	1986	(1)	1990	(1)	1991	(1)
1995	(1)	1996	(1)	2001	(1)	2002	(1)	2006	(1)	2353	(1)	3104	(1)	3118	(1)
575	(1)	578	(1)	580	(1)	584	(1)	585	(1)	586	(1)	587	(1)	702	(1)
704	(1)	708	(1)	709	(1)	713	(1)	714	(1)	719	(1)	720	(1)	721	(1)
722	(1)	731	(1)	735	(1)	736	(1)	740	(1)	741	(1)	745	(1)	746	(1)

```
                759  (1)  763  (1)  764  (1)  765  (1)  766  (1)  767  (1)  768  (1)  772  (1)
                773  (1)  774  (1)  775  (1)  776  (1)  777  (1)  779  (1)  783  (1)  784  (1)
                788  (1)  789  (1)  790  (1)  791  (1)  792  (1)  796  (1)  797  (1)  803  (1)
                804  (1)  805  (1)  806  (1)  807  (1)  808  (1)  809  (1)  810  (1)  811  (1)
                818  (1)  819  (1)  824  (1)  825  (1)  829  (1)  830  (1)  831  (1)  832  (1)
                833  (1)  834  (1)  840  (1)  841  (1)  842  (1)  843  (1)  844  (1)  848  (1)
                849  (1)  853  (1)  854  (1)  855  (1)  856  (1)  860  (1)  861  (1)  862  (1)
                863  (1)  867  (1)  868  (1)  869  (1)  870  (1)  871  (1)  872  (1)  884  (1)
                888  (1)  889  (1)  890  (1)  894  (1)  895  (1)  899  (1)  900  (1)  904  (1)
                905  (1)  906  (1)  910  (1)  911  (1)  916  (1)  917  (1)  918  (1)  922  (1)
                923  (1)  924  (1)  925  (1)  935  (1)  936  (1)  937  (1)  939  (1)  940  (1)
                941  (1)  964  (1)  965  (1)  971  (1)  972  (1)  973  (1)  981  (1)  994  (1)
                995  (1)  997  (1)  998  (1)
.IIF            2353 (1)  575  (1)  578  (1)  580  (1)  584  (1)  585  (1)  586  (1)  587  (1)
.IRP            2081 (1)  2485 (1)  2490 (1)  2816 (1)  2912 (1)  2976 (1)  3158 (1)  3343 (1)
                4012 (1)  575  (1)  578  (1)  580  (1)  584  (1)  585  (1)  586  (1)  587  (1)
.LIBRARY        550  (1)  551  (1)  552  (1)  553  (1)
.LIST           577  (1)  579  (1)
.LONG           614  (1)  618  (1)
.MACRO          565  (1)  575  (1)  578  (1)  580  (1)  584  (1)  585  (1)  586  (1)  587  (1)
.MDELETE        575  (1)  578  (1)  586  (1)  587  (1)
.NLIST          577  (1)  579  (1)
.NOCROSS        576  (1)  577  (1)  581  (1)  582  (1)  583  (1)  588  (1)
.NOSHOW         34   (1)
.PAGE           1025 (1)  1081 (1)  1124 (1)  1225 (1)  1280 (1)  1343 (1)  1389 (1)  1444 (1)
                1500 (1)  152  (1)  1553 (1)  1606 (1)  1718 (1)  1781 (1)  1883 (1)  1905 (1)
                1951 (1)  1966 (1)  2010 (1)  2049 (1)  205  (1)  2084 (1)  2120 (1)  2174 (1)
                2201 (1)  2355 (1)  2391 (1)  2422 (1)  2495 (1)  254  (1)  2587 (1)  2666 (1)
                2896 (1)  2981 (1)  3014 (1)  305  (1)  3127 (1)  3196 (1)  3240 (1)  3262 (1)
                3302 (1)  3314 (1)  3358 (1)  3424 (1)  3458 (1)  3484 (1)  3520 (1)  3568 (1)
                3594 (1)  3631 (1)  364  (1)  3671 (1)  3697 (1)  3707 (1)  3757 (1)  3819 (1)
                3850 (1)  3872 (1)  3925 (1)  3962 (1)  4030 (1)  420  (1)  545  (1)  590  (1)
                623  (1)  692  (1)  747  (1)  798  (1)  835  (1)  873  (1)  926  (1)  96   (1)
                974  (1)
.PSECT          2012 (1)  579  (1)  595  (1)  628  (1)
.RESTORE        576  (1)  577  (1)  579  (1)  581  (1)  582  (1)  583  (1)  588  (1)
.SAVE           576  (1)  577  (1)  579  (1)  581  (1)  582  (1)  583  (1)  588  (1)
.SBTTL          2011 (1)  2175 (1)  546  (1)
.SUBTITLE       1026 (1)  1082 (1)  1125 (1)  1226 (1)  1281 (1)  1344 (1)  1390 (1)  1445 (1)
                1501 (1)  1719 (1)  1782 (1)  1952 (1)  2356 (1)  2392 (1)  2423 (1)  2496 (1)
                2588 (1)  2667 (1)  2897 (1)  2982 (1)  3015 (1)  3128 (1)  3197 (1)  3241 (1)
                3263 (1)  3303 (1)  3315 (1)  3359 (1)  3425 (1)  3459 (1)  3485 (1)  3521 (1)
                3569 (1)  3595 (1)  3632 (1)  3672 (1)  3708 (1)  3758 (1)  3851 (1)  3873 (1)
                3926 (1)  3963 (1)  4031 (1)  591  (1)  624  (1)  632  (1)  693  (1)  748  (1)
                799  (1)  836  (1)  874  (1)  927  (1)  975  (1)
.TITLE          32   (1)
.WEAK           3857 (1)  558  (1)  561  (1)
.WORD           1000 (1)  1001 (1)  1002 (1)  1016 (1)  1017 (1)  1019 (1)  1020 (1)  1022 (1)
                1023 (1)  1024 (1)  1035 (1)  1036 (1)  1039 (1)  1040 (1)  1041 (1)  1043 (1)
                1044 (1)  1046 (1)  1047 (1)  1048 (1)  1050 (1)  1051 (1)  1053 (1)  1054 (1)
                1056 (1)  1057 (1)  1058 (1)  1059 (1)  1061 (1)  1062 (1)  1064 (1)  1065 (1)
                1067 (1)  1068 (1)  1070 (1)  1071 (1)  1073 (1)  1074 (1)  1076 (1)  1077 (1)
                1079 (1)  1080 (1)  1084 (1)  1085 (1)  1087 (1)  1088 (1)  1090 (1)  1093 (1)
                1094 (1)  1100 (1)  1101 (1)  1102 (1)  1103 (1)  1104 (1)  1105 (1)  1107 (1)
                1108 (1)  1114 (1)  1115 (1)  1116 (1)  1118 (1)  1119 (1)  1121 (1)  1122 (1)
                1123 (1)  1136 (1)  1137 (1)  1139 (1)  1140 (1)  1142 (1)  1143 (1)  1144 (1)
                1145 (1)  1146 (1)  1148 (1)  1149 (1)  1151 (1)  1152 (1)  1153 (1)  1155 (1)
```

1157	(1)	1159	(1)	1160	(1)	1162	(1)	1163	(1)	1165	(1)	1166	(1)	1167	(1)
1168	(1)	1172	(1)	1173	(1)	1174	(1)	1175	(1)	1176	(1)	1177	(1)	1178	(1)
1179	(1)	1180	(1)	1181	(1)	1182	(1)	1183	(1)	1184	(1)	1185	(1)	1186	(1)
1187	(1)	1188	(1)	1189	(1)	1190	(1)	1191	(1)	1192	(1)	1193	(1)	1194	(1)
1195	(1)	1196	(1)	1197	(1)	1198	(1)	1199	(1)	1200	(1)	1201	(1)	1202	(1)
1203	(1)	1204	(1)	1205	(1)	1206	(1)	1211	(1)	1212	(1)	1213	(1)	1214	(1)
1215	(1)	1216	(1)	1217	(1)	1218	(1)	1219	(1)	1220	(1)	1221	(1)	1222	(1)
1223	(1)	1224	(1)	1236	(1)	1237	(1)	1238	(1)	1243	(1)	1244	(1)	1245	(1)
1249	(1)	1250	(1)	1251	(1)	1255	(1)	1256	(1)	1260	(1)	1261	(1)	1265	(1)
1266	(1)	1267	(1)	1271	(1)	1272	(1)	1273	(1)	1277	(1)	1278	(1)	1279	(1)
1288	(1)	1292	(1)	1293	(1)	1297	(1)	1298	(1)	1299	(1)	1300	(1)	1304	(1)
1305	(1)	1309	(1)	1311	(1)	1312	(1)	1313	(1)	1314	(1)	1318	(1)	1319	(1)
1320	(1)	1321	(1)	1325	(1)	1327	(1)	1328	(1)	1329	(1)	1331	(1)	1332	(1)
1333	(1)	1334	(1)	1335	(1)	1336	(1)	1337	(1)	1338	(1)	1339	(1)	1340	(1)
1342	(1)	1348	(1)	1349	(1)	1350	(1)	1351	(1)	1352	(1)	1353	(1)	1355	(1)
1361	(1)	1363	(1)	1364	(1)	1365	(1)	1366	(1)	1368	(1)	1369	(1)	1371	(1)
1377	(1)	1379	(1)	1380	(1)	1381	(1)	1382	(1)	1383	(1)	1385	(1)	1386	(1)
1388	(1)	1405	(1)	1407	(1)	1408	(1)	1410	(1)	1411	(1)	1412	(1)	1413	(1)
1414	(1)	1416	(1)	1417	(1)	1419	(1)	1420	(1)	1422	(1)	1423	(1)	1425	(1)
1426	(1)	1428	(1)	1429	(1)	1430	(1)	1432	(1)	1433	(1)	1434	(1)	1435	(1)
1437	(1)	1438	(1)	1440	(1)	1441	(1)	1443	(1)	1451	(1)	1453	(1)	1454	(1)
1455	(1)	1456	(1)	1457	(1)	1459	(1)	1460	(1)	1461	(1)	1462	(1)	1463	(1)
1465	(1)	1466	(1)	1467	(1)	1468	(1)	1469	(1)	1474	(1)	1475	(1)	1476	(1)
1477	(1)	1478	(1)	1480	(1)	1488	(1)	1490	(1)	1491	(1)	1492	(1)	1493	(1)
1494	(1)	1496	(1)	1497	(1)	1499	(1)	1535	(1)	1536	(1)	1540	(1)	1541	(1)
1545	(1)	1546	(1)	1550	(1)	1551	(1)	1552	(1)	1557	(1)	1558	(1)	1563	(1)
1564	(1)	1570	(1)	1571	(1)	1575	(1)	1576	(1)	1580	(1)	1581	(1)	1582	(1)
1586	(1)	1587	(1)	1588	(1)	1592	(1)	1593	(1)	1594	(1)	1595	(1)	1599	(1)
1600	(1)	1604	(1)	1605	(1)	1610	(1)	1614	(1)	1615	(1)	1617	(1)	1618	(1)
1620	(1)	1621	(1)	1623	(1)	1624	(1)	1626	(1)	1627	(1)	1631	(1)	1632	(1)
1636	(1)	1637	(1)	1641	(1)	1642	(1)	1643	(1)	1644	(1)	1648	(1)	1649	(1)
1653	(1)	1654	(1)	1658	(1)	1659	(1)	1663	(1)	1664	(1)	1668	(1)	1669	(1)
1673	(1)	1674	(1)	1675	(1)	1676	(1)	1680	(1)	1681	(1)	1682	(1)	1683	(1)
1687	(1)	1688	(1)	1692	(1)	1693	(1)	1697	(1)	1698	(1)	1702	(1)	1703	(1)
1707	(1)	1708	(1)	1710	(1)	1711	(1)	1713	(1)	1714	(1)	1732	(1)	1733	(1)
1738	(1)	1739	(1)	1740	(1)	1742	(1)	1743	(1)	1744	(1)	1746	(1)	1747	(1)
1749	(1)	1750	(1)	1752	(1)	1756	(1)	1757	(1)	1758	(1)	1759	(1)	1761	(1)
1762	(1)	1763	(1)	1764	(1)	1768	(1)	1769	(1)	1770	(1)	1775	(1)	1776	(1)
1780	(1)	1803	(1)	1804	(1)	1808	(1)	1809	(1)	1810	(1)	1811	(1)	1815	(1)
1816	(1)	1817	(1)	1818	(1)	1819	(1)	1820	(1)	1822	(1)	1826	(1)	1827	(1)
1828	(1)	1829	(1)	1830	(1)	1831	(1)	1832	(1)	1837	(1)	1838	(1)	1839	(1)
1840	(1)	1842	(1)	1843	(1)	1844	(1)	1845	(1)	1849	(1)	1850	(1)	1854	(1)
1855	(1)	1856	(1)	1861	(1)	1862	(1)	1866	(1)	1867	(1)	1868	(1)	1869	(1)
1870	(1)	1871	(1)	1873	(1)	1874	(1)	1878	(1)	1879	(1)	1880	(1)	1881	(1)
1882	(1)	1887	(1)	1888	(1)	1889	(1)	1890	(1)	1891	(1)	1893	(1)	1894	(1)
1898	(1)	1899	(1)	1900	(1)	1901	(1)	1902	(1)	1904	(1)	1913	(1)	1914	(1)
1916	(1)	1917	(1)	1919	(1)	1920	(1)	1922	(1)	1923	(1)	1925	(1)	1926	(1)
1928	(1)	1929	(1)	1931	(1)	1932	(1)	1934	(1)	1935	(1)	1936	(1)	1938	(1)
1943	(1)	1944	(1)	1945	(1)	1946	(1)	1950	(1)	1965	(1)	1970	(1)	1971	(1)
1975	(1)	1976	(1)	1980	(1)	1981	(1)	1985	(1)	1986	(1)	1990	(1)	1991	(1)
1995	(1)	1996	(1)	2001	(1)	2002	(1)	2006	(1)	2207	(1)	2208	(1)	2209	(1)
2210	(1)	2211	(1)	2212	(1)	2213	(1)	2214	(1)	2215	(1)	2216	(1)	2217	(1)
2218	(1)	2219	(1)	2220	(1)	2221	(1)	2222	(1)	2223	(1)	2224	(1)	2225	(1)
2226	(1)	2227	(1)	2228	(1)	2229	(1)	2230	(1)	2231	(1)	2232	(1)	2233	(1)
2234	(1)	2235	(1)	2236	(1)	2237	(1)	2238	(1)	2239	(1)	2240	(1)	2241	(1)
2242	(1)	2243	(1)	2244	(1)	2245	(1)	2246	(1)	2247	(1)	2248	(1)	2249	(1)
2250	(1)	2251	(1)	2252	(1)	2253	(1)	2254	(1)	2255	(1)	2256	(1)	2257	(1)

2258	(1)	2259	(1)	2260	(1)	2261	(1)	2262	(1)	2263	(1)	2264	(1)	2265	(1)
2266	(1)	2267	(1)	2268	(1)	2269	(1)	2270	(1)	2271	(1)	2272	(1)	2273	(1)
2274	(1)	2275	(1)	2276	(1)	2277	(1)	2278	(1)	2279	(1)	2280	(1)	2281	(1)
2282	(1)	2283	(1)	2284	(1)	2285	(1)	2286	(1)	2287	(1)	2288	(1)	2289	(1)
2290	(1)	2291	(1)	2292	(1)	2293	(1)	2294	(1)	2295	(1)	2296	(1)	2297	(1)
2298	(1)	2299	(1)	2300	(1)	2301	(1)	2302	(1)	2303	(1)	2304	(1)	2305	(1)
2306	(1)	2307	(1)	2308	(1)	2309	(1)	2310	(1)	2311	(1)	2312	(1)	2313	(1)
2314	(1)	2315	(1)	2316	(1)	2317	(1)	2318	(1)	2319	(1)	2320	(1)	2321	(1)
2322	(1)	2323	(1)	2324	(1)	2325	(1)	2326	(1)	2327	(1)	2328	(1)	2329	(1)
2330	(1)	2331	(1)	2332	(1)	2333	(1)	2334	(1)	2335	(1)	2336	(1)	2337	(1)
2338	(1)	2339	(1)	2340	(1)	2341	(1)	2342	(1)	2343	(1)	2344	(1)	2345	(1)
2346	(1)	2347	(1)	2348	(1)	2349	(1)	702	(1)	704	(1)	708	(1)	709	(1)
713	(1)	714	(1)	719	(1)	720	(1)	721	(1)	722	(1)	731	(1)	735	(1)
736	(1)	740	(1)	741	(1)	745	(1)	746	(1)	759	(1)	763	(1)	764	(1)
765	(1)	766	(1)	767	(1)	768	(1)	772	(1)	773	(1)	774	(1)	775	(1)
776	(1)	777	(1)	779	(1)	783	(1)	784	(1)	788	(1)	789	(1)	790	(1)
791	(1)	792	(1)	796	(1)	797	(1)	803	(1)	804	(1)	805	(1)	806	(1)
807	(1)	808	(1)	809	(1)	810	(1)	811	(1)	818	(1)	819	(1)	824	(1)
825	(1)	829	(1)	830	(1)	831	(1)	832	(1)	833	(1)	834	(1)	840	(1)
841	(1)	842	(1)	843	(1)	844	(1)	848	(1)	849	(1)	853	(1)	854	(1)
855	(1)	856	(1)	860	(1)	861	(1)	862	(1)	863	(1)	867	(1)	868	(1)
869	(1)	870	(1)	871	(1)	872	(1)	884	(1)	888	(1)	889	(1)	890	(1)
894	(1)	895	(1)	899	(1)	900	(1)	904	(1)	905	(1)	906	(1)	910	(1)
911	(1)	916	(1)	917	(1)	918	(1)	922	(1)	923	(1)	924	(1)	925	(1)
935	(1)	936	(1)	937	(1)	939	(1)	940	(1)	941	(1)	964	(1)	965	(1)
971	(1)	972	(1)	973	(1)	981	(1)	994	(1)	995	(1)	997	(1)	998	(1)

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	1	#-2490 (1)
R0	59	#-2064 (1) #-2067 (1) #-2105 (1) #-2109 (1) #-2133 (1) #-2170 (1)
		2172 (1) #-2204 (1) #-2596 (1) #-2597 (1) #-2598 (1) #-2604 (1)
		#-2702 (1) #-2730 (1) #-2800 (1) #-2836 (1) #-2838 (1) #-2839 (1)
		#-2851 (1) #-2969 (1) #-3121 (1) #-3122 (1) #-3125 (1) #-3137 (1)
		#-3140 (1) #-3142 (1) #-3151 (1) #-3153 (1) #-3158 (1) #-3334 (1)
		#-3338 (1) #-3343 (1) #-3770 (1) #-3775 (1) #-3786 (1) #-3789 (1)
		#-3800 (1) #-3803 (1) #-3814 (1) #-3817 (1) #-3828 (1) #-3831 (1)
		#-3845 (1) #-3848 (1) #-3859 (1) #-3863 (1) #-3892 (1) #-3893 (1)
		#-3894 (1) #-3898 (1) #-3899 (1) #-3902 (1) #-3944 (1) 3946 (1)
		#-4067 (1) #-4070 (1) #-4073 (1) #-4076 (1) #-4077 (1)
R1	15	#-2113 (1) #-2702 (1) #-2800 (1) #-2836 (1) #-2851 (1) #-3044 (1)
		#-3046 (1) #-3084 (1) #-3106 (1) #-3109 (1) #-3117 (1) #-3119 (1)
		#-4016 (1) #-4020 (1) #-4021 (1)
R10	11	#-3029 (1) #-3053 (1) #-3060 (1) #-3067 (1) #-3216 (1) #-3247 (1)
		#-3601 (1) #-3603 (1) #-3605 (1) #-3621 (1) #-3695 (1)
R2	219	2050 (1) #-2054 (1) #-2062 (1) #-2066 (1) #-2075 (1) 2099 (1)
		2100 (1) 2118 (1) #-2119 (1) 2128 (1) 2142 (1) #-2170 (1)
		2370 (1) 2378 (1) #-2386 (1) 2412 (1) 2420 (1) #-2428 (1)
		#-2429 (1) #-2430 (1) #-2438 (1) #-2445 (1) #-2467 (1) 2469 (1)
		#-2478 (1) #-2502 (1) #-2510 (1) #-2514 (1) #-2516 (1) #-2525 (1)
		2527 (1) #-2534 (1) #-2543 (1) #-2556 (1) #-2566 (1) #-2575 (1)
		2577 (1) #-2585 (1) #-2613 (1) #-2631 (1) #-2633 (1) #-2634 (1)
		#-2642 (1) #-2644 (1) #-2646 (1) #-2657 (1) 2659 (1) #-2661 (1)
		#-2662 (1) #-2664 (1) #-2673 (1) 2675 (1) 2682 (1) #-2690 (1)
		2692 (1) #-2702 (1) #-2711 (1) #-2800 (1) #-2810 (1) #-2839 (1)
		2860 (1) #-2871 (1) 2873 (1) #-2875 (1) #-2876 (1) #-2878 (1)
		#-2886 (1) #-2894 (1) #-2987 (1) #-2988 (1) #-2995 (1) #-2996 (1)
		3004 (1) 3012 (1) #-3029 (1) 3037 (1) #-3045 (1) #-3046 (1)
		#-3053 (1) #-3060 (1) #-3067 (1) #-3075 (1) #-3076 (1) #-3084 (1)
		#-3087 (1) #-3106 (1) #-3119 (1) #-3143 (1) #-3160 (1) 3168 (1)
		3170 (1) #-3172 (1) 3175 (1) #-3177 (1) #-3180 (1) 3188 (1)
		3190 (1) 3192 (1) #-3194 (1) 3204 (1) #-3206 (1) #-3209 (1)
		#-3216 (1) 3218 (1) 3226 (1) 3228 (1) #-3230 (1) 3233 (1)
		#-3235 (1) #-3238 (1) #-3247 (1) 3249 (1) #-3258 (1) 3260 (1)
		#-3269 (1) #-3270 (1) 3274 (1) #-3276 (1) #-3279 (1) 3287 (1)
		3290 (1) #-3292 (1) #-3296 (1) 3300 (1) #-3311 (1) #-3312 (1)
		#-3346 (1) 3366 (1) 3374 (1) 3382 (1) 3390 (1) 3398 (1)
		3406 (1) 3414 (1) 3422 (1) 3432 (1) 3440 (1) 3448 (1)
		3456 (1) 3466 (1) 3474 (1) 3482 (1) 3492 (1) 3494 (1)
		#-3496 (1) #-3499 (1) #-3501 (1) 3509 (1) 3511 (1) #-3513 (1)
		#-3516 (1) #-3518 (1) 3529 (1) 3534 (1) 3539 (1) 3544 (1)
		3549 (1) 3554 (1) 3556 (1) #-3559 (1) #-3563 (1) #-3566 (1)
		3573 (1) 3579 (1) 3581 (1) #-3584 (1) #-3589 (1) #-3592 (1)
		3605 (1) #-3621 (1) 3629 (1) 3639 (1) #-3643 (1) #-3646 (1)
		#-3649 (1) #-3652 (1) #-3656 (1) 3659 (1) 3662 (1) 3669 (1)
		#-3695 (1) #-3704 (1) 3715 (1) 3723 (1) 3731 (1) 3739 (1)
		3747 (1) 3755 (1) 3772 (1) #-3773 (1) 3788 (1) #-3801 (1)
		3830 (1) 3834 (1) #-3868 (1) 3879 (1) #-3881 (1) #-3884 (1)
		3891 (1) #-3894 (1) #-3896 (1) #-3899 (1) #-3901 (1) 3923 (1)

```
                    3932  (1)    #-3934  (1)    #-3937  (1)    #-3944  (1)    #-3953  (1)    #-3960  (1)
                 #-3969  (1)    #-3988  (1)    #-4006  (1)    #-4037  (1)    #-4044  (1)    #-4051  (1)
                 #-4056  (1)    #-4063  (1)    #-4086  (1)
R3          9     2050  (1)    #-2702  (1)    #-2800  (1)    #-3084  (1)    #-3106  (1)    #-3119  (1)
                 #-4016  (1)    #-4018  (1)    #-4021  (1)
R4          6     2050  (1)    #-2702  (1)    #-2800  (1)    #-3084  (1)    #-3106  (1)    #-3119  (1)
R5         19     2050  (1)    #-2702  (1)    #-2715  (1)    #-2716  (1)    #-2734  (1)    #-2735  (1)
                 #-2747  (1)    #-2748  (1)    #-2753  (1)    #-2754  (1)    #-2759  (1)    #-2760  (1)
                 #-2800  (1)    #-3084  (1)    #-3106  (1)    #-3119  (1)    #-4016  (1)    #-4017  (1)
                 #-4021  (1)
R6         27    #-2702  (1)    #-2711  (1)    #-2715  (1)     2716  (1)    #-2734  (1)     2736  (1)
                 2787  (1)    #-2800  (1)    #-3084  (1)    #-3087  (1)    #-3088  (1)    #-3089  (1)
                 #-3091  (1)     3092  (1)    #-3098  (1)    #-3100  (1)     3101  (1)    #-3102  (1)
                 #-3103  (1)     3104  (1)    #-3105  (1)    #-3106  (1)    #-3109  (1)    #-3111  (1)
                 #-3112  (1)     3118  (1)    #-3119  (1)
R7         11    #-2702  (1)    #-2733  (1)     2736  (1)    #-2747  (1)     2749  (1)    #-2753  (1)
                 2755  (1)    #-2759  (1)     2761  (1)     2777  (1)    #-2800  (1)
R8         15    #-2389  (1)    #-2429  (1)    #-2597  (1)     2841  (1)     2843  (1)     2845  (1)
                 2987  (1)    #-2996  (1)    #-3044  (1)     3075  (1)    #-3088  (1)    #-3678  (1)
                 #-3687  (1)     3911  (1)    #-3919  (1)
R9          7    #-2428  (1)    #-2598  (1)    #-2841  (1)    #-2843  (1)    #-2845  (1)    #-3679  (1)
                 #-3688  (1)
SP         36    #-2081  (1)    #-2459  (1)    #-2485  (1)    #-2490  (1)    #-2600  (1)     2601  (1)
                 #-2603  (1)    #-2816  (1)    #-2837  (1)    #-2850  (1)    #-2912  (1)    #-2976  (1)
                 #-3097  (1)    #-3098  (1)    #-3099  (1)     3101  (1)     3102  (1)    #-3105  (1)
                 #-3158  (1)    #-3343  (1)    #-4012  (1)    #-4019  (1)    #-4023  (1)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.01	00:00:00.18
Command processing	892	00:00:00.25	00:00:00.92
Pass 1	2718	00:00:23.18	00:00:25.89
Symbol table sort	5	00:00:00.40	00:00:00.39
Pass 2	826	00:00:03.30	00:00:06.61
Symbol table output	2	00:00:00.11	00:00:00.45
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	155	00:00:03.55	00:00:05.09
Assembler run totals	4625	00:00:30.81	00:00:39.54

The working set limit was 4000 pages.
592008 bytes (1157 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1074 non-local and 441 local symbols.
4108 source lines were read in Pass 1, producing 114 object records in Pass 2.
141 pages of virtual memory were used to define 41 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              9
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6               17
DISK$DS:[DS.LOCAL.LIBRARY]CRD.MLB;4                    1
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                      11
TOTALS (all libraries)                                38

881 GETS were required to define 38 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CLI.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DIA
```

Table of contents

(1) 169 Macros and Equated Symbols
(1) 319 Work Psect Declarations
(1) 358 Data Psect Declarations
(1) 363 DS$CanWait - CANCEL DELAY ROUTINE
(1) 406 $WAKE - Wakeup from hibernate
(1) 440 DSX$WaitMS - MILLISECOND DELAY ROUTINE
(1) 521 DSX$WaitUC - MICROSECOND DELAY ROUTINE
(1) 613 EXE$HIBER - HIBERNATE SYSTEM SERVICE ROUTINE.
(1) 670 EXE$CANTIM - CANCEL TIMER SYSTEM SERVICE.
(1) 737 DSR$SETIMR_INI - SET TIMER QUEUE INITIALIZATION
(1) 815 EXE$Setimr - SET TIMER SYSTEM SERVICE.
(1) 905 DSI$TimSrv - INTERVAL TIMER INTERRUPT SERVICE ROUTINE.
(1) 976 DSI$TIMER - Level 7 timer interrupt service
(1) 1098 CLK$RE_INIT - SYSTEM TIMER INITIALIZATION SUBROUTINE

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CLOCK.MAR
0000     2 ; *2    15-MAY-1986 16:29:09 BERGAZZI "MADE SOME SYMBOLS GLOBAL"
0000     3 ; *1     6-FEB-1986 15:15:03 DS ""
0000     4 ; DEC/CMS REPLACEMENT HISTORY, Element CLOCK.MAR
0000     5         .TITLE  CLOCK   INTERVAL TIMER ROUTINES.
0000     6         .IDENT  /07-36/                                              ;G:2
0000     7         .NoShow Conditionals
0000     8
0000     9 ;
0000    10 ; Copyright (c) 1977, 1982, 1983, 1984, 1985, 1986
0000    11 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    12 ;
0000    13 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    14 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    15 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    16 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    17 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    18 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    19 ; REMAIN IN DEC.
0000    20 ;
0000    21 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    22 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    23 ; CORPORATION.
0000    24 ;
0000    25 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    26 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    27
0000    28 ;++
0000    29 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ; ENVIRONMENT:
0000    34 ;
0000    35 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    36 ;
0000    37 ; MODIFIED BY:
0000    38 ;                KEN CHAPMAN     30-NOV-77       VERSION 02
0000    39 ;       01       FIXED $DS_WAITMS_L TO WAKE IN USER MODE.  (DSPR #65)
0000    40 ;                CHANGED $SETIMR TO RETURN A WARNING IF IPL IS TOO HIGH.
0000    41 ;                  (DSPR #53).
0000    42 ;
0000    43 ;                TOM SOUTTER     22-FEB-78       VERSION 03
0000    44 ;       02       INSTALLED EXE$GL_ABSTIM (NON-FUNCTIONAL)
0000    45 ;                TOM SOUTTER      2-MAR-78
0000    46 ;       03       CONVERTED SETIMR & TIMSRV ROUTINES TO USE VMS ASTDEL
0000    47 ;                TOM SOUTTER     23-MAR-78
0000    48 ;       04       TURN ON SYSTEM REAL TIME USING INTERVAL TIMER & TODC
0000    49 ;                ROGER RIGGS     03-APR-78
0000    50 ;       05       MODIFIED SETIMR TO QUE LONG REQUESTS SO THAT
0000    51 ;                THE CLOCK CAN BE USED FOR QIO WATCH DOG TIMERS
0000    52 ;       06       Fixed DSR$SETIMR_INI to release buffers in timer queue.
0000    53 ;                NICK HOWGATE    06-DEC-78       VERSION 4 (ESSAA-5.01)
0000    54 ;       07       EMULATE 'QUEUE' INSTRUCTIONS
0000    55 ;                COPY REQIDT TO ASTPRM IN SETIMR
0000    56 ;                AND FIX CANTIM TO BE ABLE TO CANCEL SPECIFIC TIMERS
0000    57 ;       08       Fix residual time in WAITMS to return correct value
```

```
                     0000     58 ;          Roger Riggs     11-Jun-1979
                     0000     59 ;     09   Revamped timer interrupt service to fix bug caused
                     0000     60 ;          by lowering IPL from 18 to 7 in calling SCH$QAST
                     0000     61 ;          when delivering timer requests. Also EXE$SETIMR
                     0000     62 ;          was affected, since the queue is now sorted by completion
                     0000     63 ;          time
                     0000     64 ;          Roger Riggs     20-NOV-1979
                     0000     65 ;     10   Remove W^ from several references
                     0000     66 ;          Roger Riggs, 24-Jan-1980, Version 5.2
                     0000     67 ;     12   Removed Errsup after clock overflow in WAITUS, since
                     0000     68 ;          it is unnecessary and means that we loaded the clock
                     0000     69 ;          with a value that will overflow before we can check it.
                     0000     70 ;          Also added DS$GK_USOVR, used by WAITUS and defined
                     0000     71 ;          in ONLY7xx for each processor, Expermentially determined
                     0000     72 ;          to produce an accuracy of about 1%.
                     0000     73 ;          Also added clear of overflow bit to clock interrupt service
                     0000     74 ;          Roger Riggs, 27-Feb-1980
                     0000     75 ;     13   Changed DSBINT to SETIPL(#18) before removing entries from
                     0000     76 ;          timer queue, Also restore IPL before continuing.
                     0000     77 ;     14   Assured that Both SETIMR and CANTIM are executed in KERNEL mode
                     0000     78 ;          Roger Riggs, 22-july-1980, Version 5.5
                     0000     79 ;     15   Moved EXE$GL_ABSTIM to ENTRY,
                     0000     80 ;
                     0000     81 ;          Dave Butenhof, 16-oct-1980, version 6.1
                     0000     82 ;     16   Move EXE$GQ_SYSTIME to entry, to make it accessible to drivers.
                     0000     83 ;          Dave Butenhof, 18-feb-1981, version 6.3
                     0000     84 ;     17   Delete SEP psect.
                     0000     85 ;          Dave Butenhof, 25-feb-1981, version 6.3
                     0000     86 ;     18   Fix problem found in queue handling.  REMQUE/INSQUE instructs
                     0000     87 ;          had been emulated.  Restore the queue instructions.
                     0000     88 ;     19   Fix WAITUS bug (Tom Kraus generated fix).
                     0000     89 ;
                     0000     90 ;     20   - Dave Butenhof, 26-Feb-1982, version 6.6
                     0000     91 ;          Add special code in WAITUS for Nebula...use interrupt
                     0000     92 ;          instead of polling.
                     0000     93 ;
                     0000     94 ;     21   - Jack Stansbury, 26-Feb-1982, Version 6.6
                     0000     95 ;          Added .LIBRARY statements for $DS, $DIAG, and $LIB.
                     0000     96 ;
                     0000     97 ;     22   Jack Stansbury, 15-September-1982, Version 6.9
                     0000     98 ;          Changed ICCS$M_ERR from 31 to 15. Also added code in DSI$TimSrv
                     0000     99 ;          routine that will not reset the interval timer if the SetVec routine
                     0000    100 ;          is currently trying to stop the interval timer.
                     0000    101 ;
                     0000    102 ;     23   Jack Stansbury, 16-September-1982, Version 6.9
                     0000    103 ;          Changed the ICCS$M_ERR bit back to #15. The VAX Hardware Handbook
                     0000    104 ;          is wrong in its picture of the ICCS in the back of the book.
                     0000    105 ;
                     0000    106 ;     24   Marion Baggett, 24-Sept-1982,   Version 6.9
                     0000    107 ;          Changed the entry mask in DSX$WAITUS to not save R2, R3. These regis
                     0000    108 ;          are not changed, and the extra time for CALLx to save them should be
                     0000    109 ;     25   Bob Bergazzi    21-Oct-1982     Version 6.9
                     0000    110 ;          Changed DSI$TIMSRV to clear the interrupt when the
                     0000    111 ;          DS$GB_Stopping_The_Timer flag is set. Otherwise we get stuck
                     0000    112 ;          in a loop of continuous interrupts.
                     0000    113 ;
                     0000    114 ;     26   Bob Bergazzi    25-April-1983   Version 6.11
```

```
0000   115 ;              Added global symbol DSX$WAITUS_USOVR to the instruction in the
0000   116 ;              DSX$WAITUS routine which computes the interval clock count,
0000   117 ;              so that we can reference it from KERNEL and change the value
0000   118 ;              of the clock routine overhead factor (DS$GK_USOVR),
0000   119 ;              if we are running on a SUPERSTAR processor. This is done from
0000   120 ;              KERNEL during the BOOT procedure so that we don't add any more
0000   121 ;              instructions to the DSX$WAITUS routine, which would change the
0000   122 ;              value of the overhead factor for the existing processors.
0000   123 ;              I.e. The code in KERNEL is modifying the instruction at
0000   124 ;              DSX$WAITUS_USOVR.
0000   125 ;
0000   126 ;       27     Bob Bergazzi     May 16,1983     Version 6.11
0000   127 ;              Changed the order of the .LIB statements.
0000   128 ;
0000   129 ;       28     Richard Brown    August 4, 1983  Version 6.13
0000   130 ;              Corrected bug in DSX$CANWAIT.  If this service was called
0000   131 ;              to cancel a $DS_WAITMS, it did not remove the entry from the
0000   132 ;              timer queue, so the AST was still being delivered.  Therefore,
0000   133 ;              the $SETIMR call in DSX$WAITMS was changed so that a request
0000   134 ;              ID is specified (-1).  DSX$CANWAIT was changed so that it
0000   135 ;              issues a $CANTIM to kill all timer requests with an ID of -1.
0000   136 ;
0000   137 ;       29     R.E.Muse         10 January 1984 version 6.14
0000   138 ;              added PRDEF macro definition - Version 4 changes.
0000   139 ;
0000   140 ;       30     Bob Bergazzi     Jan. 26, 1984   Version 6.14
0000   141 ;              Changed CLK$RE_INIT routine to call ONLY_CLOCK_INIT which
0000   142 ;              does cpu-specific stuff to the TODR.
0000   143 ;
0000   144 ;       31     Bob Bergazzi     Aug. 24, 1984   Version 7.1
0000   145 ;              Changed CLK$RE_INIT to not call ONLY_CLOCK_INIT, and also to
0000   146 ;              write the bias value (10000000) into the TODR, if the TODR
0000   147 ;              read less than 10000000. This fixes the bug where the EXE$GQ_SYSTIME
0000   148 ;              was not being re-synchronized if the TODR read less than 10000000.
0000   149 ;
0000   150 ;       32     M. Baggett       23-May-1985     Version 8.2
0000   151 ;              Fixed truncation errors caused by adding module ULTRIX
0000   152 ;
0000   153 ;       33     Domenic Andella 4-Nov-1985       Version 9.1
0000   154 ;              Add code to DSI$TIMSRV to branch around if Ap.
0000   155 ;
0000   156 ;       34     Domenic Andella 23-Dec-1985      Version 9.1
0000   157 ;              Use stack address as argument when getting processor
0000   158 ;              id in routine DSI$TIMSRV, to protect from other processors
0000   159 ;              overwritting location in memory.
0000   160 ;
0000   161 ;       35     Domenic Andella 6-Jan-1986       Version 9.1
0000   162 ;              More mp mods required in routine DSI$TIMSRV to support
0000   163 ;              Scorpio Ap. The Ap should only reset the timer, and exit
0000   164 ;
0000   165 ;       36     Bob Bergazzi     15-May-1986     Version 10.1                    ;G:2
0000   166 ;              Made TODR_BIAS and TICK_NS global for use by KERNEL module.     ;G:2
0000   167 ;--
```

```
                        169          .SBTTL  Macros and Equated Symbols
                        170 ;
                        171 ; INCLUDE FILES:
                        172 ;
                        173
                        174          .LIBRARY        /SYS$LIBRARY:LIB/        ;                    [27]
                        175          .LIBRARY        /$DS/                    ;                    [27]
                        176          .LIBRARY        /$DIAG/                  ;                    [27]
                        177
                        178 ;
                        179 ; EXTERNAL SYMBOLS
                        180 ;
                        181          .EXTRN  MP$_GET_PROCESSOR_ID             ;                [33]
                        182
                        183
                        184
                        185
                        186 ;
                        187 ; MACROS
                        188 ;
                        189          .MACRO  $PRDEF,$GBL
                        190
                        191          $DEFINI PR,$GBL
                        192
                        193
                        194  $EQU    PR$_KSP 0
                        195  $EQU    PR$_ESP 1
                        196  $EQU    PR$_SSP 2
                        197  $EQU    PR$_USP 3
                        198  $EQU    PR$_ISP 4
                        199  $EQU    PR$_P0BR        8
                        200  $EQU    PR$_P0LR        9
                        201  $EQU    PR$_P1BR        10
                        202  $EQU    PR$_P1LR        11
                        203  $EQU    PR$_SBR 12
                        204  $EQU    PR$_SLR 13
                        205  $EQU    PR$_PCBB        16
                        206  $EQU    PR$_SCBB        17
                        207  $EQU    PR$_IPL 18
                        208  $EQU    PR$_ASTLVL      19
                        209  $EQU    PR$_SIRR        20
                        210  $EQU    PR$_SISR        21
                        211  $EQU    PR$_ICCS        24
                        212  $EQU    PR$_NICR        25
                        213  $EQU    PR$_ICR 26
                        214  $EQU    PR$_TODR        27
                        215  $EQU    PR$_RXCS        32
                        216  $EQU    PR$_RXDB        33
                        217  $EQU    PR$_TXCS        34
                        218  $EQU    PR$_TXDB        35
                        219  $EQU    PR$_ACCS        40
                        220  $EQU    PR$_ACCR        41
                        221  $EQU    PR$_MAPEN       56
                        222  $EQU    PR$_TBIA        57
                        223  $EQU    PR$_TBIS        58
                        224  $EQU    PR$_PME 61
                        225  $EQU    PR$_SID 62
```

```
0000   226  $EQU   PR$_TBCHK       63
0000   227  $EQU   PR$V_SID_SN     0
0000   228  $EQU   PR$S_SID_SN     12
0000   229  $EQU   PR$V_SID_PL     12
0000   230  $EQU   PR$S_SID_PL     3
0000   231  $EQU   PR$V_SID_ECO    15
0000   232  $EQU   PR$S_SID_ECO    9
0000   233  $EQU   PR$V_SID_TYPE   24
0000   234  $EQU   PR$S_SID_TYPE   8
0000   235  $EQU   PR$_SID_TYP780  1
0000   236  $EQU   PR$_SID_TYP750  2
0000   237  $EQU   PR$_SID_TYP730  3
0000   238  $EQU   PR$_SID_TYP7VV  4
0000   239  $EQU   PR$_SID_TYPMAX  4
0000   240  $EQU   PR$_WCSA        44
0000   241  $EQU   PR$_WCSD        45
0000   242  $EQU   PR$_SBIFS       48
0000   243  $EQU   PR$_SBIS        49
0000   244  $EQU   PR$_SBISC       50
0000   245  $EQU   PR$_SBIMT       51
0000   246  $EQU   PR$_SBIER       52
0000   247  $EQU   PR$_SBITA       53
0000   248  $EQU   PR$_SBIQC       54
0000   249  $EQU   PR$_CMIERR      23
0000   250  $EQU   PR$_CSRS        28
0000   251  $EQU   PR$_CSRD        29
0000   252  $EQU   PR$_CSTS        30
0000   253  $EQU   PR$_CSTD        31
0000   254  $EQU   PR$_TBDR        36
0000   255  $EQU   PR$_CADR        37
0000   256  $EQU   PR$_MCESR       38
0000   257  $EQU   PR$_CAER        39
0000   258  $EQU   PR$_UBRESET     55
0000   259  $EQU   PR$_PAMACC      64
0000   260  $EQU   PR$_PAMLOC      65
0000   261  $EQU   PR$_CSWP        66
0000   262  $EQU   PR$_CRBT        67
0000   263  $EQU   PR$_MCTL1       68
0000   264  $EQU   PR$_MCTL2       69
0000   265  $EQU   PR$_MGEN        70
0000   266  $EQU   PR$_MTBER       71
0000   267  $EQU   PR$_MEAR        72
0000   268  $EQU   PR$_MDCR1       73
0000   269  $EQU   PR$_MEDR        74
0000   270  $EQU   PR$_MECCR       75
0000   271
0000   272          $DEFEND PR,$GBL,DEF
0000   273
0000   274          .ENDM   $PRDEF
0000   275
0000   276 $PR8SSDEF                              ;[35]
0000               .SAVE   LOCAL_BLOCK
0000               .RESTORE
0000   277
0000   278 ;
0000   279 ; EQUATED SYMBOLS:
0000   280 ;
```

```
                        0000   281
                        0000   282          $CANTIMDEF
                        0000   283          $GETTIMDEF
                        0000   284          $DDBDEF
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
                00000004 0000                       .BLKL   1
                        0004                .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
                00000008 0004                       .BLKL   1
                        0008                .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
                0000000A 0008                       .BLKW   1
                        000A                .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
                0000000B 000A                       .BLKB   1
                0000000C 000B                       .BLKB   1
                        000C                .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
                00000010 000C                       .BLKL   1
                        0010                .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
                00000013 0010                       .BLKB   3
                        0013                .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
                00000014 0013                       .BLKB   1
                        0014                .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
                00000015 0014                       .BLKB   1
                00000024 0015                       .BLKB   15
                        0024                .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
                00000025 0024                       .BLKB   1
                00000034 0025                       .BLKB   15
                        0034                .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                        0034                .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                        0000                .RESTORE
                        0000   285          $DS_DSADEF
                        0000   286          $DS_DSDEF
                        0000   287          $DYNDEF
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000   288          $IPLDEF
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000   289          $PRDEF
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000   290          $PR780DEF
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000   291          $DS_SCBDEF
                        0000   292          $SSDEF
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
```

```
                    0000   293          $PSLDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
           00000000 FE70                .=0
                    0000                .RESTORE
                    0000   294          $SETIMRDEF
                    0000   295          $TQEDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
           00000000 FE70                .=0
                    0000                .RESTORE
                    0000   296          $UCBDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
           00000000 FE70                .=0
                    0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                    0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
           00000004 0000                        .BLKL   1
                    0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                    0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
           00000008 0004                        .BLKL   1
                    0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
           0000000A 0008                        .BLKW   1
                    000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
           0000000B 000A                        .BLKB   1
                    000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
           0000000C 000B                        .BLKB   1
                    000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
           0000000D 000C                        .BLKB   1
           00000010 000D                        .BLKB   3
                    0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
           00000014 0010                        .BLKL   1
                    0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
           00000016 0014                        .BLKW   1
                    0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
           00000018 0016                        .BLKW   1
                    0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
           0000001A 0018                        .BLKW   1
                    001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
           0000001C 001A                        .BLKW   1
                    001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
           00000020 001C                        .BLKL   1
                    0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
           00000024 0020                        .BLKL   1
                    0024                .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
           00000028 0024                        .BLKL   1
                    0028                .IIF    NB,UCB$L_PID,   UCB$L_PID:
           0000002C 0028                        .BLKL   1
                    002C                .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
           00000030 002C                        .BLKL   1
```

```
                     0030          .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034             0030                  .BLKL   1
                     0034          .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038             0034                  .BLKL   1
                     0038          .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039             0038                  .BLKB   1
                     0039          .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A             0039                  .BLKB   1
                     003A          .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C             003A                  .BLKW   1
                     003C          .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                     003C          .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                     003C          .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D             003C                  .BLKB   1
                     003D          .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                     003D          .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E             003D                  .BLKB   1
                     003E          .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                     003E          .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F             003E                  .BLKB   1
                     003F          .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040             003F                  .BLKB   1
                     0040          .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044             0040                  .BLKL   1
                     0044          .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048             0044                  .BLKL   1
                     0048          .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A             0048                  .BLKW   1
                     004A          .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                     004A          .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B             004A                  .BLKB   1
                     004B          .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C             004B                  .BLKB   1
                     004C          .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050             004C                  .BLKL   1
                     0050          .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052             0050                  .BLKW   1
                     0052          .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                     0052          .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053             0052                  .BLKB   1
                     0053          .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054             0053                  .BLKB   1
                     0054          .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058             0054                  .BLKL   1
                     0058          .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A             0058                  .BLKW   1
                     005A          .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C             005A                  .BLKW   1
                     005C          .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                     005C          .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060             005C                  .BLKL   1
                     0060          .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064             0060                  .BLKL   1
                     0064          .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                     0064          .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068             0064                  .BLKL   1
                     0068          .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
```

```
0000006C  0068                                  .BLKL   1
          006C                  .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                                  .BLKW   1
          006E                  .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                                  .BLKW   1
          0070                  .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                                  .BLKB   1
          0071                  .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                                  .BLKB   1
          0072                  .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                                  .BLKW   1
          0074                  .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
          0074                  .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074                  . = 0
          0000                  .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                                  .BLKW   1
00000074  0002                  . = 116
          0074                  .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                                  .BLKL   1
          0080                  .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                                  .BLKL   1
          0090                  .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
          0090                  .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                  . = 116
          0074                  .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                                  .BLKB   1
          0075                  .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                                  .BLKB   1
          0076                  .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                                  .BLKB   1
          0077                  .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                                  .BLKB   1
          0078                  .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                                  .BLKL   1
0000007E  007C                                  .BLKW   1
          007E                  .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                                  .BLKW   1
          0080                  .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                                  .BLKL   1
00000080  0084                  . = 128
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                                  .BLKW   1
          008A                  .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                                  .BLKW   1
```

```
                    008C            .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                    008C            .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E            008C                            .BLKW   1
                    008E            .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090            008E                            .BLKW   1
                    0090            .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092            0090                            .BLKW   1
                    0092            .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094            0092                            .BLKW   1
                    0094            .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095            0094                            .BLKB   1
                    0095            .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096            0095                            .BLKB   1
                    0096            .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098            0096                            .BLKW   1
00000096            0098            . = 150
00000098            0096                            .BLKW   1
                    0098            .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C            0098                            .BLKL   1
                    009C            .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0            009C                            .BLKL   1
                    00A0            .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4            00A0                            .BLKL   1
                    00A4            .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6            00A4                            .BLKW   1
                    00A6            .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7            00A6                            .BLKB   1
000000A8            00A7                            .BLKB   1
00000074            00A8            . = 116
00000078            0074                            .BLKL   1
0000007C            0078                            .BLKL   1
00000080            007C                            .BLKL   1
00000084            0080                            .BLKL   1
00000088            0084                            .BLKL   1
0000008C            0088                            .BLKL   1
                    008C            .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090            008C                            .BLKL   1
00000094            0090                            .BLKL   1
00000098            0094                            .BLKL   1
0000009C            0098                            .BLKL   1
                    009C            .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D            009C                            .BLKB   1
                    009D            .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E            009D                            .BLKB   1
                    009E            .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F            009E                            .BLKB   1
000000A0            009F                            .BLKB   1
                    00A0            .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1            00A0                            .BLKB   1
                    00A1            .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2            00A1                            .BLKB   1
                    00A2            .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3            00A2                            .BLKB   1
000000A4            00A3                            .BLKB   1
                    00A4            .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5            00A4                            .BLKB   1
                    00A5            .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
```

```
000000A7  00A5                          .BLKW   1
000000A8  00A7                          .BLKB   1
          00A8              .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                          .BLKL   1
000000B0  00AC                          .BLKL   1
000000B4  00B0                          .BLKL   1
000000B8  00B4                          .BLKL   1
          00B8              .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                          .BLKL   1
          00BC              .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC              .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC              . = 116
          0074              .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                          .BLKL   1
          0078              .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                          .BLKL   1
          007C              .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                          .BLKW   1
00000080  007E                          .BLKW   1
          0000              .RESTORE
          0000    297       $DS_WAITMS_DEF
          0000    298       $DS_WAITUS_DEF
          0000    299       CMKDEF
          0000    300       DSFDEF
          0000    301
00000001  0000    302 ICCS$M_RUN      = 1 @ 0
00000010  0000    303 ICCS$M_XFER     = 1 @ 4
00000040  0000    304 ICCS$M_IE       = 1 @ 6
00000080  0000    305 ICCS$M_INT      = 1 @ 7
80000000  0000    306 ICCS$M_ERR      = 1 @ 31                ;                       [23]  ;G:2
          0000    307
800000C1  0000    308 ICCS$M_RESET    = <ICCS$M_ERR ! ICCS$M_INT ! ICCS$M_IE ! ICCS$M_RUN>
          0000    309
10000000  0000    310 TODR_BIAS       == 1 @ 28               ; ^X10000000, approx. 1 month   [36]  ;G:2
          0000    311
00002710  0000    312 TICK_US         = 10000                 ; MICROSECONDS PER TICK
000186A0  0000    313 TICK_NS         == TICK_US*10           ; 100NS UNITS PER TICK          [36]  ;G:2
00000064  0000    314 SEC_TICK        == 1000000/TICK_US      ; TICKS PER SECOND
          0000    315
00000018  0000    316 CLOCK_IPL       = 24
```

```
                        00000000   318          .PSECT  Work, NoShr, NoExe, Wrt, Long   ;                          [17]  ;G:2
                             0000  319          .SBTTL  Work Psect Declarations
                             0000  320 ;
                             0000  321 ; OWN STORAGE:
                             0000  322 ;
                             0000  323
                             0000  324 DS$GL_TIMQFL:
                        00000000' 0000  325          .LONG   DS$GL_TIMQFL            ;QUEUE HEADER FOR LONG SETIMR REQUESTS
                        00000000' 0004  326          .LONG   DS$GL_TIMQFL
                             0008  327
                             0008  328 DS$GQ_CURYEAR::
                        00000010  0008  329          .BLKQ   1
                             0010  330
                             0010  331 DS$GT_NEWYEAR::
20 38 37 39 31 2D 4E 41 4A 2D 31 30  0010  332          .ASCII  \01-JAN-1978  0:0:0.0\
           30 2E 30 3A 30 3A 30 20   001C
                             0024  333
                        00000014  0024  334 DS$K_NYSIZ  ==  .-DS$GT_NEWYEAR
                             0024  335
                             0024  336 ONESECTIM:
                   00000000 00000000  0024  337          .LONG   0,0                     ; Flink and Blink
                            0000  002C  338          .WORD   0                       ; Length =0
                              0F  002E  339          .BYTE   DYN$C_TQE               ; Type of data struncture
                              05  002F  340          .BYTE   TQE$C_SSREPT            ; Request type=repeat
                        000003B4' 0030  341          .LONG   ONE_SECOND              ; Address of once/sec routine
                   00000000 00000000  0034  342          .LONG   0,0                     ; Unused longwords
                   00000000 00000000  003C  343          .LONG   0,0                     ; Initial expiration time
                            00989680  0044  344          .LONG   100000*100              ; Delta time=1 second
                            00000000  0048  345          .LONG   0                       ; Final unused
                                      004C  346
                                      004C  347 WAITMS_ARGS:
                                      004C  348          $SETIMR  DAYTIM=MSTIME, ASTADR=DS$CANWAIT, -
                                      004C  349                   REQIDT=-1                                        ;[28]
                        00000004' 004C                    .ADDRESS        4
                        00000000' 0050                    .ADDRESS        0
                        00000060' 0054                    .ADDRESS        MSTIME
                        00000000' 0058                    .ADDRESS        DS$CANWAIT
                        FFFFFFFF' 005C                    .ADDRESS        -1
                                  0060  350
                                  0060  351 MSTIME:
                        00000068  0060  352          .BLKL   2
                                  0068  353
                        00000000  0068  354 ID_TEMP:        .LONG 0                         ; Storage for cpu id [33]
                                  006C  355
                                  006C  356
```

```
                        006C   358          .SBTTL  Data Psect Declarations
                    00000000   359          .PSECT  Data, Shr, NoExe, NoWrt, Byte
                        0000   360
                        0000   361          MODNAM  CLOCK
 4B 43 4F 4C 43 00' 0000       $MODULE:     .ASCIC  \CLOCK\
                    05  0000
```

```
                                0006          .SBTTL  DS$CanWait -    CANCEL DELAY ROUTINE
                            00000000          .PSECT  Code, Shr, Exe, NoWrt, Long     ;                                [17]
                                0000  ;++
                                0000  ; FUNCTIONAL DESCRIPTION:
                                0000  ;
                                0000  ;       ROUTINE TO CANCEL PROGRAM DELAY
                                0000  ;
                                0000  ; CALLING SEQUENCE:
                                0000  ;
                                0000  ;       DS$CANWAIT()
                                0000  ;
                                0000  ; INPUT PARAMETERS:     NONE
                                0000  ;
                                0000  ; IMPLICIT INPUTS:      NONE
                                0000  ;
                                0000  ; OUTPUT PARAMETERS:    NONE
                                0000  ;
                                0000  ; IMPLICIT OUTPUTS:
                                0000  ;
                                0000  ;       DS$GL_FLAGS:
                                0000  ;               DS$V_ABRTFLG    = ABORT WAIT FLAG.
                                0000  ;
                                0000  ; COMPLETION CODES:
                                0000  ;
                                0000  ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                                0000  ;
                                0000  ; SIDE EFFECTS:
                                0000  ;
                                0000  ;       WAKES UP HIBERNATING PROCESS.
                                0000  ;--
                                0000
                          0000  0000  .ENTRY  DSX$CANWAIT,^M<>
                                0002
                                0002          $CANTIM_S       REQIDT=#-1      ; Dequeue the timer queue entry [28]
                    00    DD    0002                  PUSHL   #0
          FFFFFFFF  8F    DD    0004                  PUSHL   #-1
  00000000'GF       02    FB    000A                  CALLS   #2,G^SYS$CANTIM
                                0011                                          ; that was enqueued by the      [28]
                                0011                                          ; $SETIMR call in DSX$WAITMS    [28]
                                0011                                          ; (If we're not cancelling a    [28]
                                0011                                          ; $DS_WAITMS, this doesn't do   [28]
                                0011                                          ; anything.)                    [28]
                    7E    7C    0011          CLRQ    -(SP)                   ; Two null args
  00000000'EF       02    FB    0013          CALLS   #2,SYS$WAKE             ; wake the program
                          04    001A          RET                             ; RETURN TO THE DIAGNOSTIC
```

```
                                001B   406          .SBTTL  $WAKE - Wakeup from hibernate
                                001B   407 ;++
                                001B   408 ; FUNCTIONAL DESCRIPTION:
                                001B   409 ;
                                001B   410 ;       This routine releases the process blocked by $HIBER
                                001B   411 ;
                                001B   412 ; CALLING SEQUENCE:
                                001B   413 ;
                                001B   414 ;       SYS$WAKE()
                                001B   415 ;
                                001B   416 ; INPUT PARAMETERS:     NONE
                                001B   417 ;
                                001B   418 ; IMPLICIT INPUTS:      NONE
                                001B   419 ;
                                001B   420 ; OUTPUT PARAMETERS:    NONE
                                001B   421 ;
                                001B   422 ; IMPLICIT OUTPUTS:
                                001B   423 ;
                                001B   424 ;       DS$V_ABRTFLG is SET in DS$GL_FLAGS
                                001B   425 ;
                                001B   426 ; COMPLETION CODES:     SS$_NORMAL
                                001B   427 ;
                                001B   428 ; SIDE EFFECTS:
                                001B   429 ;
                                001B   430 ;       The next time or current time $HIBER is called it will exit.
                                001B   431 ;--
                                001B   432
                        0000    001B   433 .ENTRY  EXE$WAKE,^M<>
                                001D   434
00000000'EF     40 8F    88     001D   435          BISB    #DS$M_ABRTFLG, -
                                0025   436                  L^DS$GL_FLAGS           ; Set the flag
              50    01    D0    0025   437          MOVL    S^#SS$_NORMAL,R0        ; It worked
                                 04    0028   438          RET                     ; Return
```

```
0029   440          .SBTTL  DSX$WaitMS -    MILLISECOND DELAY ROUTINE
0029   441 ;++
0029   442 ; FUNCTIONAL DESCRIPTION:
0029   443 ;
0029   444 ;       THIS ROUTINE ALLOWS THE PROGRAMMER TO DELAY HIS PROGRAM SOME
0029   445 ;       NUMBER OF 10 MILLISECOND UNITS.
0029   446 ;
0029   447 ; CALLING SEQUENCE:
0029   448 ;
0029   449 ;       DIAGNOSTIC SUPERVISOR PROCEDURE CALL.
0029   450 ;
0029   451 ; INPUT PARAMETERS:
0029   452 ;
0029   453 ;       WAITMS$_TIME(AP)= NUMBER OF 10 MILLISECOND UNITS TO DELAY.
0029   454 ;       WAITMS$_TAG(AP) = ADDRESS TO RECEIVE REMAINING TIME ON CANWAIT.
0029   455 ;
0029   456 ; IMPLICIT INPUTS:
0029   457 ;
0029   458 ;       NONE
0029   459 ;
0029   460 ; OUTPUT PARAMETERS:
0029   461 ;
0029   462 ;       R0 = SUCCESS/FAILURE
0029   463 ;       R1 = NUMBER OF UNITS LEFT
0029   464 ;
0029   465 ; IMPLICIT OUTPUTS:
0029   466 ;
0029   467 ;       NONE
0029   468 ;
0029   469 ; COMPLETION CODES:
0029   470 ;
0029   471 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
0029   472 ;       DS$_ICERR       = INTERVAL CLOCK ERROR.
0029   473 ;       DS$_PROGERR     = NEGATIVE TIME INTERVAL SPECIFIED.
0029   474 ;                           (OR LESS THAN OVERHEAD)
0029   475 ;
0029   476 ; SIDE EFFECTS:
0029   477 ;
0029   478 ;       NONE
0029   479 ;--
```

```
                          0000  0029   481 .ENTRY  DSX$WAITMS,^M<>
                                002B   482
                                002B   483         $GETTIM_S  -(SP)                 ; GET START TIME
                          7E    7F     002B                   PUSHAQ  -(SP)
              00000000'GF 01    FB     002D                   CALLS   #1,G^SYS$GETTIM
                                0034   484
                          50    DC     0034   485         MOVPSL  R0                       ; READ PROCESSOR STATUS.
                       05 10    ED     0036   486         CMPZV   #PSL$V_IPL, #PSL$S_IPL,-; CHECK IPL.
                       02 50            0039   487                 R0, #2
                          19    19     003B   488         BLSS    10$
     50   000003E8 8F   04 AC  C5     003D   489         MULL3   WAITMS$_TIME(AP), -      ; CHANGE MILLSECONDS TO MICROSECONDS.
                                0046   490                 #1000, R0
                                0046   491         $DS_WAITUS_S  TIME=R0
                          00    DD     0046                   PUSHL   #0
                          50    DD     0048                   PUSHL   R0
              00000000'9F 02    FB     004A                   CALLS   #2, a#DS$WAITUS
                       5F 50    E9     0051   492         BLBC    R0,DSX$WAITMS_X          ; ERROR IN WAITUS, EXIT
                          30    11     0054   493         BRB     30$
                                0056   494
           50   00660020 8F    D0     0056   495 10$:     MOVL    #DS$_PROGERR, R0         ; ANTICIPATE NEG. TIME
         FFFE7960 8F   04 AC  7A     005D   496         EMUL    WAITMS$_TIME(AP), #-TICK_NS,- ; TRANSLATE TIME INTO 100NS INCS.
    00000060'EF   00001200 8F       0065   497                 #4608, MSTIME            ;  PLUS OVERHEAD
                          42    18     006F   498         BGEQ    DSX$WAITMS_X             ; IF GEQ, NEGATIVE OR TOO SMALL
                                0071   499
                                0071   500         $SETIMR_G  L^WAITMS_ARGS
    00000000'GF   0000004C'EF   FA     0071                   CALLG   L^WAITMS_ARGS,G^SYS$SETIMR
                       34 50    E9     007C   501         BLBC    R0, DSX$WAITMS_X         ; SKIP IF ERROR.
                                007F   502
                                007F   503         $HIBER_S
              00000000'GF 00    FB     007F                   CALLS   #0,G^SYS$HIBER
                                0086   504
                                0086   505 30$:     $GETTIM_S  -(SP)                 ; GET END TIME
                          7E    7F     0086                   PUSHAQ  -(SP)
              00000000'GF 01    FB     0088                   CALLS   #1,G^SYS$GETTIM
                       50 8E    7D     008F   506         MOVQ    (SP)+,R0                 ; GET END TIME OFF STACK
                       50 8E    C2     0092   507         SUBL2   (SP)+,R0                 ; MAKE LOW ORDER DIFFERENCE
                       51 8E    D9     0095   508         SBWC    (SP)+,R1                 ; AND HIGH ORDER DIFFERENCE
 50   51   50   000186A0 8F    7B     0098   509         EDIV    #100000,R0,R1,R0         ; MAKE TRASH(R1) AND 10'S OF MILLSECONDS
                50   08 AC    D0     00A1   510         MOVL    WAITMS$_TAG(AP),R0       ; CHECK FOR REMAINING TIME ADR.
                          09    13     00A5   511         BEQL    40$
                60   04 AC    51     C3     00A7   512         SUBL3   R1,WAITMS$_TIME(AP),(R0); Store unused time
                          02    18     00AC   513         BGEQ    40$                      ; Branch if positive
                          60    D4     00AE   514         CLRL    (R0)                     ; Was negative reduce to zero
                                00B0   515
                       50 01    D0     00B0   516 40$:     MOVL    S^#SS$_NORMAL,R0         ; NORMAL RETURN
                                00B3   517
                                00B3   518 DSX$WAITMS_X:
                          04    00B3   519         RET                              ; RETURN TO PROGRAM
```

```
                        00B4    521          .SBTTL  DSX$WaitUC -    MICROSECOND DELAY ROUTINE
                        00B4    522 ;++
                        00B4    523 ; FUNCTIONAL DESCRIPTION:
                        00B4    524 ;
                        00B4    525 ;       THIS ROUTINE ALLOWS THE PROGRAMMER TO DELAY HIS PROGRAM SOME
                        00B4    526 ;       NUMBER OF 10 MICROSECOND UNITS.
                        00B4    527 ;
                        00B4    528 ; CALLING SEQUENCE:
                        00B4    529 ;
                        00B4    530 ;       DIAGNOSTIC SUPERVISOR PROCEDURE CALL.
                        00B4    531 ;
                        00B4    532 ; INPUT PARAMETERS:
                        00B4    533 ;
                        00B4    534 ;       WAITUS$_TIME(AP)= NUMBER OF 10 MICROSECOND UNITS TO DELAY.
                        00B4    535 ;       WAITUS$_TAG(AP) = ADDRESS TO RECEIVE REMAINING TIME ON CANWAIT.
                        00B4    536 ;
                        00B4    537 ; IMPLICIT INPUTS:
                        00B4    538 ;
                        00B4    539 ;       NONE
                        00B4    540 ;
                        00B4    541 ; OUTPUT PARAMETERS:
                        00B4    542 ;
                        00B4    543 ;       R0 = SUCCESS/FAILURE
                        00B4    544 ;       R1 = NUMBER OF UNITS LEFT
                        00B4    545 ;
                        00B4    546 ; IMPLICIT OUTPUTS:
                        00B4    547 ;
                        00B4    548 ;       NONE
                        00B4    549 ;
                        00B4    550 ; COMPLETION CODES:
                        00B4    551 ;
                        00B4    552 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                        00B4    553 ;       DS$_ICERR       = INTERVAL CLOCK ERROR.
                        00B4    554 ;       DS$_PROGERR     = NEGATIVE TIME INTERVAL SPECIFIED.
                        00B4    555 ;                           (OR LESS THAN OVERHEAD)
                        00B4    556 ;
                        00B4    557 ; SIDE EFFECTS:
                        00B4    558 ;
                        00B4    559 ;       NONE
                        00B4    560 ;--
```

```
                                00B4          .ENABLE LOCAL_BLOCK                ;                                    [26]
                        0000    00B4  .ENTRY  DSX$WAITUS,^M<>                    ;                                    [24]
                                00B6          DsbInt  #31                        ; Set to IPL 31                      [20]
                7E   12   DB    00B6                  MFPR    S^#PR$_IPL,-(SP)
                12   1F   DA    00B9                  MTPR    #31,S^#PR$_IPL
                50   1C   D0    00BC          MOVL    S^#SS$_EXQUOTA, R0         ; ASSUME MULTIPLE TIME FUNCTION ERROR
     00000000'EF     11   E0    00BF          BBS     #DS$V_TIMRON, -            ; CHECK FOR TIMER ALREADY IN USE.
                     1F         00C6                  DS$GL_FLAGS,25$            ; LEAP FROG TO DSX$WAITUS_X
                                00C7
  00000000'EF   40 8F   8A      00C7          BICB2   #DS$M_ABRTFLG, -           ; CLEAR ABORT WAIT FLAG.
                                00CF                  DS$GL_FLAGS
                                00CF  DSX$WAITUS_USOVR::                         ;                                    [26]
  FFFFFFF6 8F   04 AC   7A      00CF          EMUL    WAITUS$_TIME(AP), -        ; COMPUTE INTERVAL CLOCK COUNT.
     50   00000000'8F          00D7
                                00DD                  #-10,#DS$GK_USOVR,R0       ; PLUS OVERHEAD
                     0C   19    00DD          BLSS    30$
     50   00660020 8F   D0      00DF          MOVL    #DS$_PROGERR, R0           ; IF GEQ, NEGATIVE OR TOO SMALL
                                00E6  25$:    EnbInt                             ; Restore IPL on error path          [20]
                12   8E   DA    00E6                  MTPR    (SP)+,S^#PR$_IPL
                     49   11    00E9          BRB     DSX$WAITUS_X
                                00EB
     18   80000080 8F   DA      00EB  30$:    MTPR    #ICCS$M_INT + -            ; STOP THE CLOCK.
                                00F2                   ICCS$M_ERR, #PR$_ICCS
                     51   D4    00F2          CLRL    R1                         ; ANTICIPATE NO RESIDUAL TIME.
                19   50   DA    00F4          MTPR    R0, #PR$_NICR              ; SET NEXT INTERVAL.
     18   80000091 8F   DA      00F7          MTPR    #ICCS$M_XFER ! ICCS$M_RUN ! -   ;                            [19]
                                00FE                   ICCS$M_INT ! ICCS$M_ERR, -     ;                            [19]
                                00FE                     #PR$_ICCS                ; LOAD AND START THE INTERVAL CLOCK.
                                00FE          EnbInt                             ; Enable interrupts again            [20]
                12   8E   DA    00FE                  MTPR    (SP)+,S^#PR$_IPL
                                0101
     00000000'EF     06   E4    0101  40$:    BBSC    #DS$V_ABRTFLG, -           ; CHECK FOR CANWAIT.
                     0E         0108                  DS$GL_FLAGS, 50$
                50   18   DB    0109          MFPR    #PR$_ICCS, R0              ; READ THE CLOCK STATUS.
     50   80000080 8F   D3      010C          BITL    #ICCS$M_INT ! -            ; CHECK FOR DONE
                                0113                   ICCS$M_ERR, R0            ;  OR ERROR.
                     EC   13    0113          BEQL    40$                        ; BR IF NEITHER.
                     0A   11    0115          BRB     55$                        ; Normal completion, no residual     [20]
                                0117
                51   1A   DB    0117  50$:    MFPR    #PR$_ICR, R1               ; SAVE COUNT REMAINING.
     51   FFFFFFF6 8F   C6      011A          DIVL    #-10, R1                   ; COMPUTE 10US UNITS LEFT.
                                0121
                08 AC   D5      0121  55$:    TSTL    WAITUS$_TAG(AP)            ; IF RETURN TAG NOT SPECIFIED        [20]
                     04   13    0124          BEQL    60$                        ; SKIP THE STORE
           08 BC     51   D0    0126          MOVL    R1,@WAITUS$_TAG(AP)        ; STORE THE REMAINING (10US)
                                012A
                50   01   D0    012A  60$:    MOVL    S^#SS$_NORMAL, R0                  ; NORMAL RETURN.
                                012D
                     03   BB    012D  70$:    PUSHR   #^M<R0,R1>                 ; SAVE RETURN INFORMATION
                  02DA   30     012F          BSBW    CLK$RE_INIT                ; RESTART AND RESYNC THE TOD AND ICCS
                     03   BA    0132          POPR    #^M<R0,R1>                 ; RESTORE RETURN INFORMATION
                                0134          .DISABLE LOCAL_BLOCK               ;                                    [26]
                                0134
                                0134  DSX$WAITUS_X:
                          04    0134          RET                                ; RETURN TO PROGRAM
```

```
0135   613          .SBTTL  EXE$HIBER -     HIBERNATE SYSTEM SERVICE ROUTINE.
0135   614 ;++
0135   615 ; FUNCTIONAL DESCRIPTION:
0135   616 ;
0135   617 ;       EMULATION OF STARLET SYSTEM SERVICE "HIBER" (HIBERNATE).
0135   618 ;       THE HIBERNATE SYSTEM SERVICE ALLOWS A PROCESS TO MAKE ITSELF INACTIVE
0135   619 ;       BUT TO REMAIN KNOWN TO THE SYSTEM SO THAT IT CAN BE INTERRUPTED, FOR
0135   620 ;       EXAMPLE TO RECEIVE ASTS.  A HIBERNATE REQUEST IS A WAIT-FOR-WAKE-EVENT
0135   621 ;       REQUEST.  WHEN A WAKE IS ISSUED FOR A HIBERNATING PROCESS WITH THE
0135   622 ;       $WAKE SYSTEM SERVICE OR A RESULT OF A SCHEDULE WAKEUP ($SCHDWK)
0135   623 ;       SYSTEM SERVICE, THE PROCESS CONTINUES EXECUTION AT THE INSTRUCTION
0135   624 ;       FOLLOWING THE HIBERNATE CALL.
0135   625 ;
0135   626 ; CALLING SEQUENCE:
0135   627 ;
0135   628 ;       STARLET PROCEDURE CALL.
0135   629 ;
0135   630 ; INPUT PARAMETERS:     NONE
0135   631 ;
0135   632 ; IMPLICIT INPUTS:      NONE
0135   633 ;
0135   634 ; OUTPUT PARAMETERS:    NONE
0135   635 ;
0135   636 ; IMPLICIT OUTPUTS:     NONE
0135   637 ;
0135   638 ; COMPLETION CODES:
0135   639 ;
0135   640 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
0135   641 ;
0135   642 ; SIDE EFFECTS:         NONE
0135   643 ;--
```

```
                     0000  0135   645 .ENTRY  EXE$HIBER,^M<>
                           0137   646
                FEC6'  30  0137   647 10$:    BSBW    KB_CHECK                ; CHECK FOR ^C.
00000000'EF       06  E5  013A   648          BBCC    #DS$V_ABRTFLG, -        ; CHECK FOR ABORTWAIT FLAG.
                  F5       0141   649                  DS$GL_FLAGS, 10$
                           0142   650          CMK     HIBER                   ; CLEAR CLOCK.
                  7E  DC   0142                MOVPSL  -(SP)
              06 6E   1A  E0 0144                BBS     #PSL$V_IS, (SP), 30000$
                  8E  D4   0148                CLRL    (SP)+
                  07  BC   014A                CHMK    #CMK$_HIBER
                  06  11   014C                BRB     30001$
         00000158'EF  16   014E        30000$: JSB     CMK$HIBER
                           0154        30001$:
              50  01  D0   0154   651          MOVL    S^#SS$_NORMAL, R0       ; NORMAL RETURN STATUS.
                           0157   652 EXE$HIBER_X:
                      04   0157   653          RET
                           0158   654
                           0158   655 ;
                           0158   656 ; KERNEL MODE HIBERNATE ROUTINE.
                           0158   657 ;
                           0158   658
                           0158   659 CMK$HIBER::
00000000'EF       11  E5  0158   660          BBCC    #DS$V_TIMRON, -         ; RELEASE THE TIMER.
                  0D       015F   661                  DS$GL_FLAGS, CMK$HIBER_X
18   80000080 8F      DA  0160   662          MTPR    #ICCS$M_ERR + -         ; IF TIMED, STOP & CLEAR CLOCK.
                           0167   663                   ICCS$M_INT, #PR$_ICCS
              51  1A   DB  0167   664          MFPR    #PR$_ICR, R1            ; SAVE COUNT.
                 029F  30  016A   665          BSBW    CLK$RE_INIT
                           016D   666
                           016D   667 CMK$HIBER_X:
                      02   016D   668          REI                             ; RETURN.
```

```
                     016E    670          .SBTTL  EXE$CANTIM -    CANCEL TIMER SYSTEM SERVICE.
                     016E    671 ;++
                     016E    672 ; FUNCTIONAL DESCRIPTION:
                     016E    673 ;
                     016E    674 ;       THE CANCEL TIMER REQUEST SYSTEM SERVICE CANCELS ALL OR A SELECTED
                     016E    675 ;       SUBSET OF THE SET TIMER REQUESTS PREVIOUSLY ISSUED BY A PROCESS.
                     016E    676 ;
                     016E    677 ; CALLING SEQUENCE:
                     016E    678 ;
                     016E    679 ;       STARLET PROCEDURE CALL.
                     016E    680 ;
                     016E    681 ; INPUT PARAMETERS:
                     016E    682 ;
                     016E    683 ;       REQIDT  = If zero cancel all, else match with ASTPRM.
                     016E    684 ;       ACMODE  = NOT IMPLEMENTED.
                     016E    685 ;
                     016E    686 ; IMPLICIT INPUTS:      NONE
                     016E    687 ;
                     016E    688 ; OUTPUT PARAMETERS:    NONE
                     016E    689 ;
                     016E    690 ; IMPLICIT OUTPUTS:     NONE
                     016E    691 ;
                     016E    692 ; COMPLETION CODES:
                     016E    693 ;
                     016E    694 ;       SS$_NORMAL      = SERVICE SUCCESSFULLY COMPLETED.
                     016E    695 ;
                     016E    696 ; SIDE EFFECTS:         NONE
                     016E    697 ;--
```

```
                     003C  016E   699 .ENTRY  EXE$CANTIM,^M<R2,R3,R4,R5>
                           0170   700
                           0170   701         CMK     CANTIM                  ; Change to KERNEL mode
                  7E   DC  0170               MOVPSL  -(SP)
            06 6E   1A   E0  0172               BBS     #PSL$V_IS, (SP), 30002$
                    8E   D4  0176               CLRL    (SP)+
                    0B   BC  0178               CHMK    #CMK$_CANTIM
                    06   11  017A               BRB     30003$
          00000186'EF    16  017C    30002$: JSB     CMK$CANTIM
                           0182    30003$:
               50   01   D0  0182   702         MOVL    S^#SS$_NORMAL,R0        ; Success
                    04   0185   703         RET
                           0186   704
                           0186   705 ;+
                           0186   706 ; KERNEL mode routine to cancel timer queues
                           0186   707 ;-
                           0186   708 CMK$CANTIM::
                           0186   709         DSBINT                          ; Prevent interruptions
               7E   12   DB  0186               MFPR    S^#PR$_IPL,-(SP)
               12   1F   DA  0189               MTPR    #31,S^#PR$_IPL
         55 00000000'EF    DE  018C   710         MOVAL   DS$GL_TIMQFL,R5         ; Point to queue head
             54   04 AC   D0  0193   711         MOVL    CANTIM$_REQIDT(AP),R4   ; Get request identification
                           0197   712
                    65   DD  0197   713         PUSHL   (R5)                    ; Push address of first FLINK
                           0199   714
                  50 8ED0  0199   715 10$:     POPL    R0                      ; Get address of entry
               55   50   D1  019C   716         CMPL    R0,R5                   ; FLINK to queue head?
                    19   13  019F   717         BEQL    30$                     ; All done
                    60   DD  01A1   718         PUSHL   TQE$L_TQFL(R0)          ; Push link to next TQE
                           01A3   719
             0B A0   95   01A3   720         TSTB    TQE$B_RQTYPE(R0)        ; Is it ast request?
                F1   12   01A6   721         BNEQ    10$                     ; No, leave it in the queue
                54   D5   01A8   722         TSTL    R4                      ; Cancel all?
                06   13   01AA   723         BEQL    20$                     ; Branch if all
         14 A0   54   D1   01AC   724         CMPL    R4,TQE$L_ASTPRM(R0)     ; Match on REQIDT?
                E7   12   01B0   725         BNEQ    10$                     ; Branch if not
                           01B2   726 ;+
                           01B2   727 ; REMOVE ENTRY IN R0 FROM QUEUE AND DEALLOCATE
                           01B2   728 ;-
            50   60   0F  01B2   729 20$:     REMQUE  (R0), R0                ; Remove queue entry                    [18]
                FE48'   30  01B5   730         BSBW    EXE$DEANONPAGED         ; DE-ALLOCATE THE SPACE
                  DF   11   01B8   731         BRB     10$                     ; Check next
                           01BA   732
                           01BA   733 30$:
                           01BA   734         ENBINT                          ; Restore IPL
            12   8E   DA  01BA               MTPR    (SP)+,S^#PR$_IPL
                    02   01BD   735         REI                             ; Return to previous mode
```

```
                                01BE   737          .SBTTL  DSR$SETIMR_INI -        SET TIMER QUEUE INITIALIZATION
                                01BE   738 ;++
                                01BE   739 ; FUNCTIONAL DESCRIPTION:
                                01BE   740 ;
                                01BE   741 ;       THIS ROUTINE INITIALIZES THE SETIMR QUEUES
                                01BE   742 ;       AND IS CALLED FROM KERNEL ON A START 10000
                                01BE   743 ;
                                01BE   744 ; CALLING SEQUENCE:
                                01BE   745 ;
                                01BE   746 ;       BSBW    DSR$SETIMR_INI
                                01BE   747 ;
                                01BE   748 ; INPUT PARAMETERS:     NONE
                                01BE   749 ;
                                01BE   750 ; IMPLICIT INPUTS:      NONE
                                01BE   751 ;
                                01BE   752 ; OUTPUT PARAMETERS:    NONE
                                01BE   753 ;
                                01BE   754 ; IMPLICIT OUTPUTS:
                                01BE   755 ;
                                01BE   756 ;       THE DS$GL_TIMQFL QUEUE IS EMPTY
                                01BE   757 ;
                                01BE   758 ; SIDE EFFECTS:         NONE
                                01BE   759 ;
                                01BE   760 ;--
                                01BE   761
                                01BE   762 DSR$SETIMR_INI::
          50   00000000'EF   DE   01BE   763          MOVAL   L^DS$GL_TIMQFL,R0       ; GET QUEUE ADDRESS
                     60   50   D0   01C5   764          MOVL    R0,(R0)                 ; Flink to self
                   04 A0   50   D0   01C8   765          MOVL    R0,4(R0)                ; Blink to self
          50   00000000'EF   7D   01CC   766          MOVQ    EXE$GQ_SYSTIME,R0       ; Expected completion time
          55   00000024'EF   DE   01D3   767          MOVAL   ONESECTIM,R5            ; Point to one/sec timer
                        01   10   01DA   768          BSBB    EXE$INSTIMQ             ; Insert in timer queue
                             05   01DC   769          RSB                             ; RETURN
```

```
                        01DD    771 ;++
                        01DD    772 ; FUNCTIONAL DESCRIPTION:
                        01DD    773 ;
                        01DD    774 ;       This routine is used to place entries in the time queue
                        01DD    775 ;       The queue is ordered by completion time. the soonest request
                        01DD    776 ;       being the first in the queue.
                        01DD    777 ;
                        01DD    778 ; CALLING SEQUCENE:
                        01DD    779 ;
                        01DD    780 ;       BSBW    EXE$INSTIMQ
                        01DD    781 ;
                        01DD    782 ;
                        01DD    783 ; INPUT PARAMETERS:
                        01DD    784 ;
                        01DD    785 ;       R0/R1           64 bit completion time
                        01DD    786 ;       R5              Address of TQE
                        01DD    787 ;
                        01DD    788 ;
                        01DD    789 ; IMPLICIT INPUTS:      NONE
                        01DD    790 ;
                        01DD    791 ; OUTPUT PARAMETERS:    NONE
                        01DD    792 ;
                        01DD    793 ; IMPLICIT OUTPUTS:     NONE
                        01DD    794 ;
                        01DD    795 ;--
                        01DD    796
                        01DD    797 EXE$INSTIMQ::
                        01DD    798
      18 A5   50   7D   01DD    799         MOVQ    R0,TQE$Q_TIME(R5)       ; Insert completion time in TQE
                        01E1    800         DSBINT  #IPL$_TIMER             ; Disable to timer level
         7E   12   DB   01E1                MFPR    S^#PR$_IPL,-(SP)
         12   07   DA   01E4                MTPR    #IPL$_TIMER,S^#PR$_IPL
54 00000000'EF   DE   01E7    801         MOVAL   DS$GL_TIMQFL,R4         ; Point to queue head
         53   54   D0   01EE    802         MOVL    R4,R3                   ; Copy start of queue
      53   04 A3   D0   01F1    803 50$:    MOVL    TQE$L_TQBL(R3),R3       ; Link to next entry
         54   53   D1   01F5    804         CMPL    R3,R4                   ; End of queue?
              0E   13   01F8    805         BEQL    60$                     ; Yes insert it here
      1C A3   51   D1   01FA    806         CMPL    R1,TQE$Q_TIME+4(R3)     ; Compare High order parts of time
              F1   1F   01FE    807         BLSSU   50$                     ; If LSSU new entry more imminent
              06   1A   0200    808         BGTRU   60$                     ; If GTRU new entry less imminent
      18 A3   50   D1   0202    809         CMPL    R0,TQE$Q_TIME(R3)       ; Compare low order part of time
              E9   1F   0206    810         BLSSU   50$                     ; If LSSU new entry more imminent
         63   65   0E   0208    811 60$:    INSQUE  (R5), (R3)              ; Insert new entry                      [18]
                        020B    812         ENBINT
         12   8E   DA   020B                MTPR    (SP)+,S^#PR$_IPL
                   05   020E    813         RSB
```

```
                        .SBTTL  EXE$Setimr -    SET TIMER SYSTEM SERVICE.
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THE SET TIMER SYSTEM SERVICE ALLOWS A PROCESS TO CAUSE THE SETTING OF
                ;       AN EVENT FLAG AND THE ISSUANCE OF AN AST AT SOME FUTURE TIME.  THE
                ;       TIME FOR THE EVENT CAN BE SPECIFIED AS AN OFFSET FROM THE CURRENT
                ;       TIME, THAT IS, DELTA TIME.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       STARLET PROCEDURE CALL.
                ;
                ; INPUT PARAMETERS:
                ;
                ;       EFN     = EVENT FLAG NUMBER OF THE EVENT FLAG TO SET WHEN THE
                ;                 TIME INTERVAL EXPIRES.  IF NOT SPECIFIED, IT DEFAULTS
                ;                 TO 0.
                ;       DAYTIM  = ADDRESS OF THE QUADWORD EXPIRATION TIME.  IF THE TIME
                ;                 VALUE IS NEGATIVE, IT INDICATES AN OFFSET (DELTA TIME)
                ;                 FROM THE CURRENT TIME.
                ;       ASTADR  = ADDRESS OF THE ENTRY MASK OF AN AST ROUTINE TO BE
                ;                 CALLED WHEN THE TIME INTERVAL EXPIRES.  IF 0, IT
                ;                 INDICATES NO AST IS TO BE QUEUED.
                ;       REQIDT  = COPIED TO ASTPRM ON AST DELIVERY.
                ;
                ; IMPLICIT INPUTS:      NONE
                ;
                ; OUTPUT PARAMETERS:    NONE
                ;
                ; IMPLICIT OUTPUTS:     NONE
                ;
                ; COMPLETION CODES:     NONE
                ;
                ;       SS$_EXQUOTA     = QUOTA FOR TIMER REQUESTS EXCEEDED.
                ;       DS$_NOTIMP      = FUNCTION NOT IMPLEMENTED IN STAND-ALONE.
                ;       SS$_NORMAL      = NORMAL RETURN.
                ;
                ; SIDE EFFECTS:         NONE
                ;--
```

```
                     003C  020F    856 .ENTRY  EXE$SETIMR,^M<R2,R3,R4,R5>
                           0211    857
                           0211    858         CMK     SETIMR                  ; Execute code in KERNEL mode
                        7E   DC    0211         MOVPSL  -(SP)
                     06 6E   1A    E0  0213         BBS     #PSL$V_IS, (SP), 30004$
                        8E   D4    0217         CLRL    (SP)+
                        0C   BC    0219         CHMK    #CMK$_SETIMR
                        06   11    021B         BRB     30005$
                00000224'EF   16    021D     30004$: JSB     CMK$SETIMR
                           0223     30005$:
                        04   0223    859         RET                             ; All done, return
                           0224    860
                           0224    861 ;+
                           0224    862 ; KERNEL mode routine to SETIMR
                           0224    863 ;-
                           0224    864 CMK$SETIMR::
                     50    1C   D0    0224    865         MOVL    S^#SS$_EXQUOTA,R0       ; PRESUME QUOTA EXCEEDED
                        FDD6'   30    0227    866         BSBW    W^KB_CHECK
                00000000'EF   11    E0  022A    867         BBS     #DS$V_TIMRON, DS$GL_FLAGS -
                        64          0231    868                 ,50$                    ; Can't SETIMR if TIMER owned
                           0232    869
                           0232    870         $CLREF_G  (AP)                  ; CLEAR THE EVENT FLAG.
                00000000'GF   6C    FA  0232                CALLG   (AP),G^SYS$CLREF
                     5A 50    E9    0239    871         BLBC    R0,50$                  ; Exit if CLREF failed
                           023C    872
                00000000'EF   40 8F   8A  023C    873         BICB2   #DS$M_ABRTFLG, -        ; CLEAR ABORTWAIT FLAG.
                           0244    874                 DS$GL_FLAGS
                           0244    875
                50   006600B8 8F   D0    0244    876         MOVL    #DS$_IPL2HI, R0         ; ANTICIPATE IPL TOO HIGH
                        51   DC    024B    877         MOVPSL  R1                      ; GET PROCESSOR STATUS.
                     05   10   ED    024D    878         CMPZV   #PSL$V_IPL,#PSL$S_IPL,- ; CHECK IPL.
                     02   51          0250    879                   R1, #2
                        42   18    0252    880         BGEQ    50$                     ; If GEQ, IPL is too high
                50   006600B0 8F   D0    0254    881         MOVL    #DS$_NOTIMP,R0          ; NOT IMPLEMENTED FOR ABSOLUTE TIME
                     51   08 BC   7D    025B    882         MOVQ    @SETIMR$_DAYTIM(AP),R1  ; TEST SIGN BIT OF QUADWORD
                        35   14    025F    883         BGTR    50$                     ; Branch to exit if invalid delta time
                           0261    884 ;
                           0261    885 ; ALLOCATE SPACE AND BUILD A TIMER QUEUE ENTRY FOR THIS TIME REQUEST
                           0261    886 ;
                        FD9C'   30    0261    887         BSBW    EXE$ALLOCTQE            ; RETURNS R2=ADDR OF BLOCK IF SUCCEEDS
                     2F 50    E9    0264    888         BLBC    R0,50$                  ; Exit if no space for TQE
                     55   52   D0    0267    889         MOVL    R2,R5                   ; Copy address of TQE
                        0B A5   94    026A    890         CLRB    TQE$B_RQTYPE(R5)        ; CLEAR AST CONTROL BLOCK FLAGS
             0C A5   00000000'8F   D0    026D    891         MOVL    #IOC$K_DIAGPID,TQE$L_PID(R5) ; SET PID
                10 A5   0C AC   7D    0275    892         MOVQ    SETIMR$_ASTADR(AP), -
                           027A    893                   TQE$L_AST(R5)           ; COPY AST ADDRESS AND PARAMETER
                29 A5   04 AC   90    027A    894         MOVB    SETIMR$_EFN(AP), -      ; COPY EVENT FLAG NUMBER TO TQE
                           027F    895                   TQE$B_EFN(R5)
                   7E   08 BC   7D    027F    896         MOVQ    @SETIMR$_DAYTIM(AP),-(SP) ; GET DURATION TIME
                50   00000000'EF   7D    0283    897         MOVQ    EXE$GQ_SYSTIME,R0       ; GET CURRENT TIME
                     50   8E   C2    028A    898         SUBL2   (SP)+,R0                ; Current -(-DELTA)=Current+DELTA
                     51   8E   D9    028D    899         SBWC    (SP)+,R1                ; Same with high order
                        FF4A   30    0290    900         BSBW    EXE$INSTIMQ             ; Insert it in the timer queue
                     50   01   D0    0293    901         MOVL    S^#SS$_NORMAL,R0        ; RETURN SUCCESS
                           0296    902                                             ; FALL IN TO EXIT
                        02   0296    903 50$:    REI                             ; Return to calling mode
```

```
0297   905          .SBTTL  DSI$TimSrv -    INTERVAL TIMER INTERRUPT SERVICE ROUTINE.
0297   906 ;++
0297   907 ; FUNCTIONAL DESCRIPTION:
0297   908 ;
0297   909 ;       INTERVAL TIMER INTERRUPT SERVICE ROUTINE
0297   910 ;
0297   911 ; CALLING SEQUENCE:
0297   912 ;
0297   913 ;       INTERRUPT
0297   914 ;
0297   915 ; IMPLICIT INPUTS:
0297   916 ;
0297   917 ;       NONE
0297   918 ;
0297   919 ; IMPLICIT OUTPUTS:
0297   920 ;
0297   921 ;       NONE
0297   922 ;
0297   923 ; SIDE EFFECTS:
0297   924 ;
0297   925 ;       NONE
0297   926 ;--
```

```
                                0297   928          .ALIGN  LONG
                                0298   929 DSI$TIMSRV::
          09 00000000'EF    E8  0298   930          Blbs    L^DS$GB_Stopping_The_Timer, -; Don't start the interval [35]
                                029F   931            4$                              ; timer up again if we are in the [35]
                                029F   932                                            ; . . . the process of stopping it - in [22]
                                029F   933                                            ; . . . the DSX$SETVEC routine, because of
                                029F   934                                            ; . . . the asynchronous nature of the  [22]
                                029F   935                                            ; . . . VAX 11/730 console instructions [22]
                                029F   936
        18  800000C1 8F    DA  029F   937          MTPR    #ICCS$M_RESET,#PR$_ICCS ; Resetting the timer
                          07    11  02A6   938          BRB     5$                      ; Normal flow                           [25]
                                02A8   939
        18  80000080 8F    DA  02A8   940 4$:       MTPR    #ICCS$M_ERR+ICCS$M_INT, -;                                      [35]
                                02AF   941                  #PR$_ICCS               ; Clear the interrupt to avoid race     [25]
                                02AF   942                                          ; with DSX$SETVEC                       [25]
             00000000'EF    D5  02AF   943 5$:       TSTL    MP$GR_BOOTED_PROCESSORS ; Are we an mp system? [33]
                          1B    13  02B5   944          BEQL    7$                      ; If not, then branch  [33]
                          52    DD  02B7   945          PUSHL   R2                      ; Get register         [35]
        52  0000005E 8F    DB  02B9   946          MFPR    #PR$SS$_BINID,R2        ; Read node id for current cpu [35]
        52  00000000'EF    91  02C0   947          CMPB    DS$GB_PRIMARY_CPU_IDENTIFIER,R2; Are we the primary?[35]
                          06    13  02C7   948          BEQL    6$                      ; Branch if Pp  [35]
                          52 8ED0  02C9   949          POPL    R2                      ; Release storage       [35]
                       004B   31  02CC   950          BRW     30$                     ; Not the primary, skip incr time [35]
                                02CF   951
                          52 8ED0  02CF   952 6$:      POPL    R2                      ; Release storage       [35]
  00000000'EF   000186A0 8F    C0  02D2   953 7$:      ADDL2   #TICK_NS, EXE$GQ_SYSTIME ; INCREMENT SYSTIME    [35]
             00000004'EF    00    D8  02DD   954          ADWC    #0, EXE$GQ_SYSTIME+4    ; AND HIGH HALF
                                02E4   955
                                02E4   956 ;+
                                02E4   957 ;       CHECK TIMER QUEUE FOR DUE REQUESTS
                                02E4   958 ;-
                                02E4   959
                          50    DD  02E4   960          PUSHL   R0                      ; Save a register               [35]
     50   00000000'EF    D0  02E6   961          MOVL    DS$GL_TIMQFL,R0         ; Get address of timer queue entry      [22]
     00000000'8F    50    D1  02ED   962          CMPL    R0,#DS$GL_TIMQFL        ; Anything in queue?
                          21    13  02F4   963          BEQL    20$                     ; Branch if not
  00000004'EF    1C A0    D1  02F6   964          CMPL    TQE$Q_TIME+4(R0),EXE$GQ_SYSTIME+4 ; Compare high order
                          0C    1F  02FE   965          BLSSU   10$                     ; If LSSU entry due
                          15    1A  0300   966          BGTRU   20$                     ; If GTRU entry not due
  00000000'EF    18 A0    D1  0302   967          CMPL    TQE$Q_TIME(R0),EXE$GQ_SYSTIME ; Compare low order
                          0B    1A  030A   968          BGTRU   20$                     ; IF GTRU entry not due
                                030C   969
                                030C   970 10$:      SOFTINT #IPL$_TIMER             ; Entry is due dispatch on timer level
                    14    07    DA  030C                       MTPR    #IPL$_TIMER,S^#PR$_SIRR
  00 00000000'EF    07    E2  030F   971          BBSS    #IPL$_TIMER,DS$GA_SOFTINT,20$; Set level 7 interrupt expected
                                0317   972
                          50 8ED0  0317   973 20$:      POPL    R0                      ; Restore R0
                             02    031A   974 30$:      REI
```

```
                        976          .SBTTL  DSI$TIMER -     Level 7 timer interrupt service
                        977 ;++
                        978 ; FUNCTIONAL DESCRIPTION:
                        979 ;
                        980 ;       This interrupt service routine is called from a software level 7
                        981 ;       timer interrupt.  It is used to process timer queue entries that
                        982 ;       are due and to check for hung qio devices
                        983 ;
                        984 ; CALLING SEQUENCE:
                        985 ;
                        986 ;       Interrupt service entry
                        987 ;
                        988 ;--
```

```
                                031B   990          .ALIGN  LONG
                                031C   991 DSI$TIMER::
                     3F   BB    031C   992          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save a few
                                031E   993
                                031E   994 ;+
                                031E   995 ;       CHECK TIMER QUEUE FOR DUE REQUESTS
                                031E   996 ;-
                                031E   997
                                031E   998 10$:     SETIPL  #IPL$_HWCLK             ; Prevent interruptions by clock
                12   18   DA    031E                MTPR    #IPL$_HWCLK,S^#PR$_IPL
        55   00000000'EF   D0    0321   999          MOVL    DS$GL_TIMQFL,R5         ; Get address of timer queue entry
        00000000'8F   55   D1    0328  1000          CMPL    R5,#DS$GL_TIMQFL        ; Anything in queue?
                      03   12    032F  1001          BNEQ    15$                     ; Branch if something in queue
                   007A    31    0331  1002          BRW     40$                     ; Exit if empty
                                0334  1003 15$:
    00000004'EF    1C A5   D1    0334  1004          CMPL    TQE$Q_TIME+4(R5),EXE$GQ_SYSTIME+4 ; Compare high order
                      0C   1F    033C  1005          BLSSU   20$                     ; If LSSU entry due
                      6E   1A    033E  1006          BGTRU   40$                     ; If GTRU entry not due
    00000000'EF    18 A5   D1    0340  1007          CMPL    TQE$Q_TIME(R5),EXE$GQ_SYSTIME ; Compare low order
                      64   1A    0348  1008          BGTRU   40$                     ; IF GTRU entry not due
                                034A  1009 ;+
                                034A  1010 ; Remove entry from queue
                                034A  1011 ;-
                                034A  1012
                50   65   0F    034A  1013 20$:     REMQUE  (R5), R0                ; Remove queue entry                [18]
                                034D  1014          SETIPL  #IPL$_TIMER             ; Back to our level
                12   07   DA    034D                MTPR    #IPL$_TIMER,S^#PR$_IPL
    50   0B A5   02   00   EF    0350  1015          EXTZV   #0,#2,TQE$B_RQTYPE(R5),R0; Get request type
                                0356  1016          CASE    R0,<25$,35$>            ; Dispatch on type
           01'   00   50   AF    0356                CASEW   R0,#0,S^#<<30007$-30006$>/2>-1
                                035A          30006$:
                    0017'  035A                    .SIGNED_WORD    25$-30006$
                    0036'  035C                    .SIGNED_WORD    35$-30006$
                                035E          30007$:
                                035E  1017          ERRSUP_S
     00000000'EF    DF     035E                        PUSHAL  $MODULE
                   00      DD    0364                  PUSHL   #0
                   01      DD    0366                  PUSHL   #$ER
    00000000'EF    03      FB    0368                  CALLS   #3, DS_ERRSUP
                   AD   11   036F  1018          BRB     10$                     ; Check again
```

```
                        0371  1020 ;+
                        0371  1021 ; Process timer entry that queues an AST or sets an event flag
                        0371  1022 ;-
                        0371  1023
        7E   29 A5   9A 0371  1024 25$:    MOVZBL  TQE$B_EFN(R5),-(SP)     ; SPECIFY EVENT FLAG NUMBER
00000000'9F    01   FB 0375  1025          CALLS   #1,a#SYS$SETEF          ; AND GO SET IT
          10 A5   D5 037C  1026          TSTL    TQE$L_AST(R5)           ; AST ADDRESS SPECIFIED ?
             07   13 037F  1027          BEQL    30$                     ; IF EQL, NO AST SPECIFIED
                        0381  1028
             52   D4 0381  1029          CLRL    R2                      ; PRIORITY INCREMENT CLASS
          FC7A'   30 0383  1030          BSBW    SCH$QAST                ; GO SCHEDULE AN AST FOR THIS TIMER
             96   11 0386  1031          BRB     10$                     ; GO BACK FOR NEXT ENTRY
                        0388  1032
        50   55   D0 0388  1033 30$:    MOVL    R5,R0                   ; POINT TO CURRENT ENTRY
          FC72'   30 038B  1034          BSBW    EXE$DEANONPAGED         ;  AND RETURN IT TO FREE MEMORY POOL
             8E   11 038E  1035          BRB     10$                     ; BACK TO LOOK AT NEXT ENTRY
                        0390  1036
                        0390  1037 ;+
                        0390  1038 ; call timer driven system routine
                        0390  1039 ;-
                        0390  1040
     53   10 A5   7D 0390  1041 35$:    MOVQ    TQE$L_FR3(R5),R3        ; Restore subroutine context
          0C B5   16 0394  1042          JSB     aTQE$L_FPC(R5)          ; Call routine
  0F 0B A5    02   E1 0397  1043          BBC     #TQE$V_REPEAT,TQE$B_RQTYPE(R5),38$; if not repeatable
     50   20 A5   7D 039C  1044          MOVQ    TQE$Q_DELTA(R5),R0      ; Get delta time
     50   18 A5   C0 03A0  1045          ADDL    TQE$Q_TIME(R5),R0       ; Add low order parts
     51   1C A5   D8 03A4  1046          ADWC    TQE$Q_TIME+4(R5),R1     ; Add high order parts w/ carry
          FE32   30 03A8  1047          BSBW    EXE$INSTIMQ             ; re-insert in the timer queue
                  03AB  1048 38$:
          FF70   31 03AB  1049          BRW     10$                     ; Look in queue again
                  03AE  1050
                  03AE  1051 40$:
                  03AE  1052          SETIPL  #IPL$_TIMER             ; Back to our level
12   07   DA 03AE                      MTPR    #IPL$_TIMER,S^#PR$_IPL
                  03B1  1053 DSI$TIMER_X:
     3F   BA 03B1  1054          POPR    #^M<R0,R1,R2,R3,R4,R5>
             02   03B3  1055          REI                             ; Exit from timer service
```

```
                                03B4  1057 ;+
                                03B4  1058 ; This routine is called each second to update the time
                                03B4  1059 ; and check for timed out UCB's
                                03B4  1060 ;-
                                03B4  1061
                                03B4  1062 ONE_SECOND:
          00000000'EF    D6    03B4  1063          INCL    EXE$GL_ABSTIM                ; Update time
                                03BA  1064
              7E    55    7D    03BA  1065          MOVQ    R5,-(SP)                     ; Save registers
      56  00000000'EF    DE    03BD  1066          MOVAL   L^IOC$GL_DEVLIST-DDB$L_LINK, R6 ; Locate first DDB          [17]
          56    66    D0    03C4  1067 10$:     MOVL    DDB$L_LINK(R6),R6            ; LOCATE NEXT DDB
                3F    13    03C7  1068          BEQL    60$                          ; IF EQL, NO MORE DDB'S
                            03C9  1069                                               ;
          55    D8 A6   DE    03C9  1070 20$:     MOVAL   DDB$L_UCB-UCB$L_LINK(R6),R5  ; LOCATE FIRST UCB
                            03CD  1071                                               ;
          55    2C A5   D0    03CD  1072 30$:     MOVL    UCB$L_LINK(R5),R5            ; LOCATE NEXT UCB
                F1    13    03D1  1073          BEQL    10$                          ; IF EQL, NO MORE UCB'S THIS DDB
                            03D3  1074 ;
                            03D3  1075 ; AT THIS POINT A UCB HAS BEEN LOCATED
                            03D3  1076 ;
                            03D3  1077
                            03D3  1078          SETIPL  #IPL$_POWER                  ; Raise to Powerfail level
          12    1F    DA    03D3                MTPR    #IPL$_POWER,S^#PR$_IPL
      28 58 A5    00    E1    03D6  1079          BBC     #UCB$V_TIM,UCB$W_STS(R5),50$ ; IF NOT WAITING, GET NEXT UCB
00000000'EF    5C A5   D1    03DB  1080          CMPL    UCB$L_DUETIM(R5),EXE$GL_ABSTIM ; CHECK DUE TIME FOR TIMED OUT
                1E    14    03E3  1081          BGTR    50$                          ; IF GTR, LET IT WAIT LONGER
                            03E5  1082
       58 A5    03    AA    03E5  1083          BICW    #UCB$M_INT!UCB$M_TIM, -
                            03E9  1084                  UCB$W_STS(R5)                ; Clear interrupt and timer
 58 A5    0040  8F    A8    03E9  1085          BISW    #UCB$M_TIMOUT,UCB$W_STS(R5)  ; SET TIMED OUT STATUS BIT
                            03EF  1086          SETIPL  UCB$B_DIPL(R5)               ; Set IPL to device level
          12    52 A5   DA    03EF                MTPR    UCB$B_DIPL(R5),S^#PR$_IPL
          53    10 A5   7D    03F3  1087          MOVQ    UCB$L_FR3(R5),R3             ; RETREIVE FORK CONTEXT
          52    0C A5   D0    03F7  1088          MOVL    UCB$L_FPC(R5),R2             ; Get sved pc
              7E    72    32    03FB  1089          CVTWL   -(R2),-(SP)                  ; Get offset to exception routine
                52    8E    C0    03FE  1090          ADDL    (SP)+,R2                     ; Address of sxception routine
                      62    16    0401  1091          JSB     (R2)                         ; RE-ACTIVATE I/O REQUEST
                            0403  1092 50$:     SETIPL  #IPL$_TIMER                  ; Restore to timer IPL
          12    07    DA    0403                MTPR    #IPL$_TIMER,S^#PR$_IPL
                C5    11    0406  1093          BRB     30$                          ; GO LOCATE NEXT UCB
                            0408  1094
              55    8E    7D    0408  1095 60$:     MOVQ    (SP)+,R5                     ; Restore
                      05    040B  1096          RSB                                  ; Return from whence we came
```

```
                     040C  1098          .SBTTL  CLK$RE_INIT -   SYSTEM TIMER INITIALIZATION SUBROUTINE
                     040C  1099 ;++
                     040C  1100 ; FUNCTIONAL DESCRIPTION:
                     040C  1101 ;
                     040C  1102 ;       INITIALIZES THE INTERVAL TIMER FOR 10 MILLISECOND TICKS
                     040C  1103 ;       AND SYNCHRONIZES THE SYSTEM TIME TO THE TIME OF YEAR CLOCK
                     040C  1104 ;
                     040C  1105 ; CALLING SEQUENCE:
                     040C  1106 ;
                     040C  1107 ;       BSBW    CLK$RE_INIT
                     040C  1108 ;
                     040C  1109 ; IMPLICIT INPUTS:
                     040C  1110 ;
                     040C  1111 ;       NONE
                     040C  1112 ;
                     040C  1113 ; IMPLICIT OUTPUTS:
                     040C  1114 ;
                     040C  1115 ;       NONE
                     040C  1116 ;
                     040C  1117 ; COMPLETION CODES:
                     040C  1118 ;
                     040C  1119 ;       NONE
                     040C  1120 ;
                     040C  1121 ; SIDE EFFECTS:
                     040C  1122 ;
                     040C  1123 ;       INTERVAL TIMER IS LEFT RUNNING (INTERRUPT ENABLED)
                     040C  1124 ;       NEXT INTERVAL REGISTER IS SET TO 10 MILLISECONDS
                     040C  1125 ;
                     040C  1126 ;       IF PR$_TODR VALID,
                     040C  1127 ;          EXE$GQ_SYSTIME      =  CURRENT DATE/TIME (64-BIT FORMAT)
                     040C  1128 ;--
```

```
                                040C  1130 CLK$RE_INIT::
                                040C  1131
                                040C  1132 ;+
                                040C  1133 ; SETUP AND START INTERVAL TIMER
                                040C  1134 ;-
                                040C  1135
        18   80000080 8F    DA  040C  1136          MTPR    #ICCS$M_ERR + -          ; CLEAR ERROR,
                                0413  1137                   ICCS$M_INT, -            ; CLEAR DONE,
                                0413  1138                         #PR$_ICCS          ; AND STOP INTERVAL TIMER
        19   FFFFD8F0 8F    DA  0413  1139          MTPR    #-TICK_US, #PR$_NICR     ; SET NEXT INTERVAL.
        18   00000051 8F    DA  041A  1140          MTPR    #ICCS$M_IE + -           ; ENABLE INTERRUPT,
                                0421  1141                   ICCS$M_XFER + -          ; LOAD,
                                0421  1142                   ICCS$M_RUN, -            ; AND RESTART
                                0421  1143                         #PR$_ICCS          ; THE INTERVAL CLOCK.
                                0421  1144
                                0421  1145 ;+
                                0421  1146 ; SET SYSTIME FROM TOD IF TOD IS VALID
                                0421  1147 ;-
            50    1B    DB    0421  1148          MFPR    #PR780$_TODR, R0         ; GET TIME OF YEAR                    [31]
        50   10000000 8F    C2  0424  1149          SUBL2   #TODR_BIAS, R0           ; REMOVE VALIDATION BIAS
                     09    1E  042B  1150          BGEQU   10$                      ; BRANCH IF TODR OK                   [31]
        1B   10000000 8F    DA  042D  1151          MTPR    #TODR_BIAS, #PR780$_TODR; TODR BAD, RESET TO 0 (+ BIAS)       [31]
                     50    D4  0434  1152          CLRL    R0                       ; BIAS MINUS BIAS = 0                 [31]
                                0436  1153
            50    01    CA    0436  1154 10$:     BICL    #1,R0                    ; CLEAR LOW BIT
         50   50    FF 8F    9C  0439  1155          ROTL    #-1,R0,R0                ; DIVIDE BY 2 UNSIGNED
   50    00   50   00030D40 8F    7A  043E  1156          EMUL    #2*TICK_NS, R0, -        ; CONVERT TIME OF YEAR TO
                                0447  1157                    #0, R0                  ;   SYSTIME FORMAT (10MS TO 100NS)
        50   00000008'EF    C0  0447  1158          ADDL2   L^DS$GQ_CURYEAR, R0      ; ADJUST TO CURRENT TIME/DATE
        51   0000000C'EF    D8  044E  1159          ADWC    L^DS$GQ_CURYEAR+4, R1    ;   (ALL 64 BITS)
   00000000'EF    50    7D  0455  1160          MOVQ    R0, EXE$GQ_SYSTIME       ; Set system date/time
   0000003C'EF    50    7D  045C  1161          MOVQ    R0, ONESECTIM+TQE$Q_TIME ; Reset 1/second timer expiration time
                                0463  1162
                          05    0463  1163          RSB
                                0464  1164
                                0464  1165
                                0464  1166          .END
```

```
$$ARGS                          = 00000004     D
$$T1                            = 00000000     D
$ER                             = 00000002     D
$MODULE                           00000000 R   D  03
BIT...                          = 0000001A     D
CANTIM$_ACMODE                  = 00000008     D
CANTIM$_NARGS                   = 00000002     D
CANTIM$_REQIDT                  = 00000004     D
CLK$RE_INIT                       0000040C RG  D  04
CLOCK_IPL                       = 00000018     D
CMK$CANTIM                        00000186 RG  D  04
CMK$HIBER                         00000158 RG  D  04
CMK$HIBER_X                       0000016D R   D  04
CMK$SETIMR                        00000224 RG  D  04
CMK$_APBOOT                     = 00000004     D
CMK$_ASTEXIT                    = 00000000     D
CMK$_CANTIM                     = 0000000B     D
CMK$_HIBER                      = 00000007     D
CMK$_INITSCB                    = 00000008     D
CMK$_LAST                       = 0000000E     D
CMK$_MMENABLE                   = 00000003     D
CMK$_RCONSOLE                   = 00000001     D
CMK$_REFRESH                    = 00000005     D
CMK$_SCHDWK                     = 00000006     D
CMK$_SETIMR                     = 0000000C     D
CMK$_SETIPL                     = 00000009     D
CMK$_SETPRT                     = 0000000D     D
CMK$_SGIPR                      = 0000000A     D
CMK$_TCONSOLE                   = 00000002     D
DDB$B_ACPCLASS                    00000013     D
DDB$B_TYPE                        0000000A     D
DDB$C_LENGTH                      00000034     D
DDB$K_LENGTH                      00000034     D
DDB$L_ACPD                        00000010     D
DDB$L_DDT                         0000000C     D
DDB$L_LINK                        00000000     D
DDB$L_UCB                         00000004     D
DDB$T_DRVNAME                     00000024     D
DDB$T_NAME                        00000014     D
DDB$W_SIZE                        00000008     D
DS$CANWAIT                        ********   X   02
DS$GA_SOFTINT                     ********   X   04
DS$GB_PRIMARY_CPU_IDENTIFIER      ********   X   04
DS$GB_STOPPING_THE_TIMER          ********   X   04
DS$GK_USOVR                       ********   X   04
DS$GL_FLAGS                       ********   X   04
DS$GL_TIMQFL                      00000000 R   D  02
DS$GQ_CURYEAR                     00000008 RG  D  02
DS$GT_NEWYEAR                     00000010 RG  D  02
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_NYSIZ                      = 00000014   G D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_NI                         = 04000000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMED                      = 02000000     D
DS$M_TIMED_OUT                  = 08000000     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_NI                         = 0000001A     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMED                      = 00000019     D
DS$V_TIMED_OUT                  = 0000001B     D
DS$V_TIMRON                     = 00000011     D
DS$WAITUS                         ********   X   04
```

```
DS$_AP_NORMAL_BREAK             = 00660150   D
DS$_ARITH                       = 006600D0   D
DS$_ASBE                        = 00660118   D
DS$_BADLINK                     = 006600F0   D
DS$_BADTYPE                     = 006600E8   D
DS$_BIIC                        = 00660120   D
DS$_CHME                        = 006600A8   D
DS$_CHMK                        = 006600E0   D
DS$_DEVNAME                     = 00660108   D
DS$_ERROR                       = 00660002   D
DS$_FHWE                        = 00660068   D
DS$_FRAGBUF                     = 00660080   D
DS$_ICBUSY                      = 006600C8   D
DS$_ICERR                       = 006600C0   D
DS$_IHWE                        = 00660060   D
DS$_ILLCHAR                     = 00660018   D
DS$_ILLPAGCNT                   = 00660078   D
DS$_ILLUNIT                     = 00660100   D
DS$_INIT_FAIL                   = 00660140   D
DS$_INSFMEM                     = 00660050   D
DS$_INVCPU                      = 00660130   D
DS$_IPL2HI                      = 006600B8   D
DS$_IVADDR                      = 00660040   D
DS$_IVVECT                      = 00660038   D
DS$_KRNLSTK                     = 00660090   D
DS$_LOGIC                       = 00660070   D
DS$_MCHK                        = 00660088   D
DS$_MEM_ALLOC_ERR               = 00660138   D
DS$_MMOFF                       = 00660058   D
DS$_NEEDUNIT                    = 006600F8   D
DS$_NODE                        = 00660128   D
DS$_NOPCS                       = 00660110   D
DS$_NORMAL                      = 00660001   D
DS$_NOSUPPORT                   = 006600B1   D
DS$_NOTALLOWMP                  = 00660148   D
DS$_NOTDON                      = 00660030   D
DS$_NOTIMP                      = 006600B0   D
DS$_NULLSTR                     = 00660010   D
DS$_OVERFLOW                    = 00660008   D
DS$_POWER                       = 00660098   D
DS$_PROGERR                     = 00660020   D
DS$_SEVERE                      = 00660004   D
DS$_TRANSL                      = 006600A0   D
DS$_TRUNCATE                    = 00660028   D
DS$_UNEXPINT                    = 006600D8   D
DS$_UNSUPPDEV                   = 00660158   D
DS$_VASFULL                     = 00660048   D
DS$_WARNING                     = 00660000   D
DSA$AL_APTMAIL                    0000FE00   D
DSA$AT_APTTXT                     0000FA00   D
DSA$GL_APTCOM                     0000FE04   D
DSA$GL_DEVLEN                     0000FE58   D
DSA$GL_ERRNO                      0000FE44   D
DSA$GL_EVENT                      0000FE48   D
DSA$GL_FLAGS                      0000FE00   D
DSA$GL_MSGTYP                     0000FE40   D
DSA$GL_PASSES                     0000FE08   D
DSA$GL_PASSNO                     0000FE54   D
DSA$GL_SECTNO                     0000FE10   D
DSA$GL_SID                        0000FE14   D
DSA$GL_SUBTNO                     0000FE4C   D
DSA$GL_TESTNO                     0000FE50   D
DSA$GL_UNITS                      0000FE0C   D
DSA$GQ_MSGPTR                     0000FE68   D
DSA$GT_DEVNAM                     0000FE5C   D
DSI$TIMER                         0000031C RG D  04
DSI$TIMER_X                       000003B1 R   D  04
DSI$TIMSRV                        00000298 RG  D  04
DSR$SETIMR_INI                    000001BE RG  D  04
DSX$CANWAIT                       00000000 RG  D  04
DSX$WAITMS                        00000029 RG  D  04
DSX$WAITMS_X                      000000B3 R   D  04
DSX$WAITUS                        000000B4 RG  D  04
DSX$WAITUS_USOVR                  000000CF RG  D  04
DSX$WAITUS_X                      00000134 R   D  04
DS_ERRSUP                         ********     X  04
DYN$C_TQE                       = 0000000F     D
EXE$ALLOCTQE                      ********     X  04
EXE$CANTIM                        0000016E RG  D  04
EXE$DEANONPAGED                   ********     X  04
EXE$GL_ABSTIM                     ********     X  04
EXE$GQ_SYSTIME                    ********     X  04
EXE$HIBER                         00000135 RG  D  04
EXE$HIBER_X                       00000157 R   D  04
EXE$INSTIMQ                       000001DD RG  D  04
EXE$SETIMR                        0000020F RG  D  04
EXE$WAKE                          0000001B RG  D  04
GETTIM$_NARGS                   = 00000001     D
GETTIM$_TIMADR                  = 00000004     D
ICCS$M_ERR                      = 80000000     D
ICCS$M_IE                       = 00000040     D
ICCS$M_INT                      = 00000080     D
ICCS$M_RESET                    = 800000C1     D
ICCS$M_RUN                      = 00000001     D
ICCS$M_XFER                     = 00000010     D
ID_TEMP                           00000068 R   D  02
IOC$GL_DEVLIST                    ********     X  04
IOC$K_DIAGPID                     ********     X  04
IPL$_HWCLK                      = 00000018     D
IPL$_POWER                      = 0000001F     D
IPL$_TIMER                      = 00000007     D
KB_CHECK                          *********    X  04
MP$GR_BOOTED_PROCESSORS           ********     X  04
MP$_GET_PROCESSOR_ID              ********     X  00
MSTIME                            00000060 R   D  02
ONESECTIM                         00000024 R   D  02
ONE_SECOND                        000003B4 R   D  04
PR$_ICCS                        = 00000018     D
PR$_ICR                         = 0000001A     D
PR$_IPL                         = 00000012     D
PR$_NICR                        = 00000019     D
PR$_SIRR                        = 00000014     D
PR780$_TODR                     = 0000001B     D
PR8SS$_BINID                    = 0000005E     D
```

```
PSL$S_IPL                 = 00000005    D        TICK_NS                  = 000186A0  G D
PSL$V_IPL                 = 00000010    D        TICK_US                  = 00002710    D
PSL$V_IS                  = 0000001A    D        TODR_BIAS                = 10000000  G D
SCB$L_ACCESS                00000020    D        TQE$B_EFN                = 00000029    D
SCB$L_ARITH                 00000034    D        TQE$B_RQTYPE             = 0000000B    D
SCB$L_BREAK                 0000002C    D        TQE$C_SSREPT             = 00000005    D
SCB$L_CHME                  00000044    D        TQE$L_AST                = 00000010    D
SCB$L_CHMK                  00000040    D        TQE$L_ASTPRM             = 00000014    D
SCB$L_CHMS                  00000048    D        TQE$L_FPC                = 0000000C    D
SCB$L_CHMU                  0000004C    D        TQE$L_FR3                = 00000010    D
SCB$L_COMPAT                00000030    D        TQE$L_PID                = 0000000C    D
SCB$L_KNLSTK                00000008    D        TQE$L_TQBL               = 00000004    D
SCB$L_MACHCHK               00000004    D        TQE$L_TQFL               = 00000000    D
SCB$L_OPCCUS                00000014    D        TQE$Q_DELTA              = 00000020    D
SCB$L_OPCDEC                00000010    D        TQE$Q_TIME               = 00000018    D
SCB$L_POWER                 0000000C    D        TQE$V_REPEAT             = 00000002    D
SCB$L_RADRMOD               0000001C    D        UCB$B_AMOD                 00000053    D
SCB$L_ROPRAND               00000018    D        UCB$B_CEX                  00000077    D
SCB$L_RXDB                  000000F8    D        UCB$B_CM1                  0000004A    D
SCB$L_SFTLVL1               00000084    D        UCB$B_CM2                  0000004B    D
SCB$L_SFTLVL10              000000A8    D        UCB$B_DEVCLASS             00000038    D
SCB$L_SFTLVL11              000000AC    D        UCB$B_DEVTYPE              00000039    D
SCB$L_SFTLVL12              000000B0    D        UCB$B_DIPL                 00000052    D
SCB$L_SFTLVL13              000000B4    D        UCB$B_DX_SCTCNT            000000A6    D
SCB$L_SFTLVL14              000000B8    D        UCB$B_ERTCNT               00000070    D
SCB$L_SFTLVL15              000000BC    D        UCB$B_ERTMAX               00000071    D
SCB$L_SFTLVL2               00000088    D        UCB$B_FEX                  00000076    D
SCB$L_SFTLVL3               0000008C    D        UCB$B_FIPL                 0000000B    D
SCB$L_SFTLVL4               00000090    D        UCB$B_LOCSRV               0000003C    D
SCB$L_SFTLVL5               00000094    D        UCB$B_OFFNDX               00000094    D
SCB$L_SFTLVL6               00000098    D        UCB$B_OFFRTC               00000095    D
SCB$L_SFTLVL7               0000009C    D        UCB$B_REMSRV               0000003D    D
SCB$L_SFTLVL8               000000A0    D        UCB$B_SECTORS              0000003C    D
SCB$L_SFTLVL9               000000A4    D        UCB$B_SLAVE                00000074    D
SCB$L_TBIT                  00000028    D        UCB$B_SPR                  00000075    D
SCB$L_TIMER                 000000C0    D        UCB$B_STATE                00000052    D
SCB$L_TRANSL                00000024    D        UCB$B_TRACKS               0000003D    D
SCB$L_TXDB                  000000FC    D        UCB$B_TT_CRFILL            0000009D    D
SCB$L_VM                    00000038    D        UCB$B_TT_DECRF             000000A1    D
SCB$L_ZERO                  00000000    D        UCB$B_TT_DELFF             000000A2    D
SCH$QAST                    ********  X   04     UCB$B_TT_DESPEE            000000A0    D
SEC_TICK                  = 00000064  G D        UCB$B_TT_DETYPE            000000A4    D
SETIMR$_ASTADR            = 0000000C    D        UCB$B_TT_LFFILL            0000009E    D
SETIMR$_DAYTIM            = 00000008    D        UCB$B_TT_SPEED             0000009C    D
SETIMR$_EFN               = 00000004    D        UCB$B_TYPE                 0000000A    D
SETIMR$_NARGS             = 00000004    D        UCB$B_VERTSZ               0000003F    D
SETIMR$_REQIDT            = 00000010    D        UCB$C_LENGTH               00000074    D
SIZ...                    = 00000001    D        UCB$C_MB_LENGTH            00000090    D
SS$_EXQUOTA               = 0000001C    D        UCB$C_TT_LENGTH            000000BC    D
SS$_NORMAL                = 00000001    D        UCB$K_LENGTH               00000074    D
SYS$CANTIM                  ********  GX   04     UCB$K_MB_LENGTH            00000090    D
SYS$CLREF                   ********  GX   04     UCB$K_TT_LENGTH            000000BC    D
SYS$GETTIM                  ********  GX   04     UCB$L_AMB                  00000054    D
SYS$HIBER                   ********  GX   04     UCB$L_ASTQBL               00000010    D
SYS$SETEF                   ********   X   04     UCB$L_ASTQFL               0000000C    D
SYS$SETIMR                  ********  GX   04     UCB$L_CPID                 0000005C    D
SYS$WAKE                    ********   X   04     UCB$L_CRB                  00000020    D
```

```
UCB$L_DDB                       00000024    D
UCB$L_DEVCHAR                   00000034    D
UCB$L_DEVDEPEND                 0000003C    D
UCB$L_DPC                       00000080    D
UCB$L_DUETIM                    0000005C    D
UCB$L_DX_BFPNT                  0000009C    D
UCB$L_DX_BUF                    00000098    D
UCB$L_DX_RXDB                   000000A0    D
UCB$L_EMB                       00000078    D
UCB$L_FIRST                     00000014    D
UCB$L_FPC                       0000000C    D
UCB$L_FQBL                      00000004    D
UCB$L_FQFL                      00000000    D
UCB$L_FR3                       00000010    D
UCB$L_FR4                       00000014    D
UCB$L_IOQBL                     00000044    D
UCB$L_IOQFL                     00000040    D
UCB$L_IRP                       0000004C    D
UCB$L_LINK                      0000002C    D
UCB$L_LOGADR                    00000064    D
UCB$L_MAXBLOCK                  00000084    D
UCB$L_MB_MBX                    0000007C    D
UCB$L_MB_PORT                   0000008C    D
UCB$L_MB_RAST                   00000078    D
UCB$L_MB_SHB                    00000080    D
UCB$L_MB_WAST                   00000074    D
UCB$L_MB_WIOQBL                 00000088    D
UCB$L_MB_WIOQFL                 00000084    D
UCB$L_MEDIA                     0000008C    D
UCB$L_NT_DATSSB                 00000074    D
UCB$L_NT_INTSSB                 00000078    D
UCB$L_OPCNT                     00000060    D
UCB$L_OWNUIC                    0000001C    D
UCB$L_PID                       00000028    D
UCB$L_RQBL                      00000004    D
UCB$L_RQFL                      00000000    D
UCB$L_SVAPTE                    00000068    D
UCB$L_SVPN                      00000064    D
UCB$L_TT_DECHAR                 000000A8    D
UCB$L_TT_RDUE                   0000008C    D
UCB$L_TT_RTIMOU                 000000B8    D
UCB$L_VCB                       00000030    D
UCB$M_INT                     = 00000002    D
UCB$M_TIM                     = 00000001    D
UCB$M_TIMOUT                  = 00000040    D
UCB$T_PARTNER                   0000000C    D
UCB$V_TIM                     = 00000000    D
UCB$W_BCNT                      0000006E    D
UCB$W_BCR                       00000096    D
UCB$W_BOFF                      0000006C    D
UCB$W_BUFQUO                    00000018    D
UCB$W_BYTESTOGO                 0000003E    D
UCB$W_CHARGE                    0000004A    D
UCB$W_CYLINDERS                 0000003E    D
UCB$W_DA                        0000008C    D
UCB$W_DC                        0000008E    D
UCB$W_DEVBUFSIZ                 0000003A    D
UCB$W_DEVSTS                    0000005A    D
UCB$W_DIRSEQ                    00000088    D
UCB$W_DSTADDR                   00000018    D
UCB$W_DX_BCR                    000000A4    D
UCB$W_EC1                       00000090    D
UCB$W_EC2                       00000092    D
UCB$W_ERRCNT                    00000072    D
UCB$W_FUNC                      0000007E    D
UCB$W_MB_SEED                   00000000    D
UCB$W_MSGCNT                    00000016    D
UCB$W_MSGMAX                    00000014    D
UCB$W_NT_CHAN                   0000007C    D
UCB$W_OFFSET                    0000008A    D
UCB$W_REFC                      00000050    D
UCB$W_SIZE                      00000008    D
UCB$W_SRCADDR                   0000001A    D
UCB$W_STS                       00000058    D
UCB$W_TT_DESIZE                 000000A5    D
UCB$W_UNIT                      00000048    D
UCB$W_VPROT                     0000001A    D
WAITMS$_NARGS                 = 00000002    D
WAITMS$_TAG                   = 00000008    D
WAITMS$_TIME                  = 00000004    D
WAITMS_ARGS                     0000004C R  D  02
WAITUS$_NARGS                 = 00000002    D
WAITUS$_TAG                   = 00000008    D
WAITUS$_TIME                  = 00000004    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000006C (108.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000006 (6.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000464 (1124.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000004	349 (1)	282 (1) 283 (1) 294 (1) 297 (1) 298 (1) 349 (1)
$$T1	=00000000	396 (1)	282 (1) 283 (1) 294 (1) 297 (1) 298 (1) 349 (1) 396 (1)
$ER	=00000002	1017 (1)	#-1017 (1)
$MODULE	00000000-R	361 (1)	1017 (1)
BIT...	=0000001A	300 (1)	286 (1) 299 (1) 300 (1)
CANTIM$_ACMODE	=00000008	282 (1)	
CANTIM$_NARGS	=00000002	282 (1)	
CANTIM$_REQIDT	=00000004	282 (1)	#-711 (1)
CLK$RE_INIT	0000040C-R	1130 (1)	#-606 (1) #-665 (1)
CLOCK_IPL	=00000018	316 (1)	
CMK$CANTIM	00000186-R	708 (1)	701 (1)
CMK$HIBER	00000158-R	659 (1)	650 (1)
CMK$HIBER_X	0000016D-R	667 (1)	#-661 (1)
CMK$SETIMR	00000224-R	864 (1)	858 (1)
CMK$_APBOOT	=00000004	299 (1)	
CMK$_ASTEXIT	=00000000	299 (1)	
CMK$_CANTIM	=0000000B	299 (1)	#-701 (1)
CMK$_HIBER	=00000007	299 (1)	#-650 (1)
CMK$_INITSCB	=00000008	299 (1)	
CMK$_LAST	=0000000E	299 (1)	
CMK$_MMENABLE	=00000003	299 (1)	
CMK$_RCONSOLE	=00000001	299 (1)	
CMK$_REFRESH	=00000005	299 (1)	
CMK$_SCHDWK	=00000006	299 (1)	
CMK$_SETIMR	=0000000C	299 (1)	#-858 (1)
CMK$_SETIPL	=00000009	299 (1)	
CMK$_SETPRT	=0000000D	299 (1)	
CMK$_SGIPR	=0000000A	299 (1)	
CMK$_TCONSOLE	=00000002	299 (1)	
DDB$L_LINK	00000000		1066 (1) #-1067 (1)
DDB$L_UCB	00000004		1070 (1)
DS$CANWAIT	00000000-XR		349 (1)
DS$GA_SOFTINT	00000000-XR		971 (1)
DS$GB_PRIMARY_CPU_IDENTIFIER	00000000-XR		#-947 (1)
DS$GB_STOPPING_THE_TIMER	00000000-XR		#-930 (1)
DS$GK_USOVR	00000000-XR		#-573 (1)
DS$GL_FLAGS	00000000-XR		#-436 (1) 567 (1) #-570 (1) 589 (1) 649 (1) 661 (1) 867 (1) #-874 (1)
DS$GL_TIMQFL	00000000-R	324 (1)	#-1000 (1) 325 (1) 326 (1) 710 (1) 763 (1) 801 (1) #-961 (1) #-962 (1) #-999 (1)
DS$GQ_CURYEAR	00000008-R	328 (1)	#-1158 (1) #-1159 (1)
DS$GT_NEWYEAR	00000010-R	331 (1)	334 (1)
DS$K_ERROR	=00000002	286 (1)	
DS$K_NORMAL	=00000001	286 (1)	
DS$K_NYSIZ	=00000014	334 (1)	
DS$K_SEVERE	=00000004	286 (1)	
DS$K_SUBSYS	=00000066	286 (1)	286 (1)

```
DS$K_WARNING                =00000000     286    (1)
DS$M_ABRTFLG                =00000040     300    (1)    #-435    (1)    #-569    (1)    #-873    (1)
DS$M_BADTIME                =00100000     300    (1)
DS$M_BATCH                  =00400000     300    (1)
DS$M_BRKCLR                 =00001000     300    (1)
DS$M_BRKPT                  =00000800     300    (1)
DS$M_CHARFLG                =00000100     300    (1)
DS$M_CMDFLG                 =00000080     300    (1)
DS$M_CTRLC                  =00000001     300    (1)
DS$M_CTRLO                  =00010000     300    (1)
DS$M_DEVFLG                 =00000200     300    (1)
DS$M_DISABLCC               =01000000     300    (1)
DS$M_DONFLG                 =00002000     300    (1)
DS$M_ERRFLG                 =00000010     300    (1)
DS$M_EXCEPT                 =00080000     300    (1)
DS$M_EXETST                 =00040000     300    (1)
DS$M_HLTFLG                 =00000008     300    (1)
DS$M_LODFLG                 =00000002     300    (1)
DS$M_MEMMGT                 =00008000     300    (1)
DS$M_NI                     =04000000     300    (1)
DS$M_OUTPUT                 =00800000     300    (1)
DS$M_RUBFLG                 =00000020     300    (1)
DS$M_SCRIPT                 =00200000     300    (1)
DS$M_SETIMR                 =02000000     300    (1)
DS$M_STRFLG                 =00000004     300    (1)
DS$M_SUBT                   =00004000     300    (1)
DS$M_SYSFLG                 =00000400     300    (1)
DS$M_TIMED                  =02000000     300    (1)
DS$M_TIMED_OUT              =08000000     300    (1)
DS$M_TIMRON                 =00020000     300    (1)
DS$V_ABRTFLG                =00000006     300    (1)    #-588    (1)    #-648    (1)
DS$V_BADTIME                =00000014     300    (1)
DS$V_BATCH                  =00000016     300    (1)
DS$V_BRKCLR                 =0000000C     300    (1)
DS$V_BRKPT                  =0000000B     300    (1)
DS$V_CHARFLG                =00000008     300    (1)
DS$V_CMDFLG                 =00000007     300    (1)
DS$V_CTRLC                  =00000000     300    (1)
DS$V_CTRLO                  =00000010     300    (1)
DS$V_DEVFLG                 =00000009     300    (1)
DS$V_DISABLCC               =00000018     300    (1)
DS$V_DONFLG                 =0000000D     300    (1)
DS$V_ERRFLG                 =00000004     300    (1)
DS$V_EXCEPT                 =00000013     300    (1)
DS$V_EXETST                 =00000012     300    (1)
DS$V_HLTFLG                 =00000003     300    (1)
DS$V_LODFLG                 =00000001     300    (1)
DS$V_MEMMGT                 =0000000F     300    (1)
DS$V_NI                     =0000001A     300    (1)
DS$V_OUTPUT                 =00000017     300    (1)
DS$V_RUBFLG                 =00000005     300    (1)
DS$V_SCRIPT                 =00000015     300    (1)
DS$V_SETIMR                 =00000019     300    (1)
DS$V_STRFLG                 =00000002     300    (1)
DS$V_SUBT                   =0000000E     300    (1)
DS$V_SYSFLG                 =0000000A     300    (1)
DS$V_TIMED                  =00000019     300    (1)
```

```
DS$V_TIMED_OUT                  =0000001B      300     (1)
DS$V_TIMRON                     =00000011      300     (1)    #-566    (1)    #-660    (1)    #-867    (1)
DS$WAITUS                        00000000-XR                    491    (1)
DS$_AP_NORMAL_BREAK             =00660150      286     (1)
DS$_ARITH                       =006600D0      286     (1)
DS$_ASBE                        =00660118      286     (1)
DS$_BADLINK                     =006600F0      286     (1)
DS$_BADTYPE                     =006600E8      286     (1)
DS$_BIIC                        =00660120      286     (1)
DS$_CHME                        =006600A8      286     (1)
DS$_CHMK                        =006600E0      286     (1)
DS$_DEVNAME                     =00660108      286     (1)
DS$_ERROR                       =00660002      286     (1)
DS$_FHWE                        =00660068      286     (1)
DS$_FRAGBUF                     =00660080      286     (1)
DS$_ICBUSY                      =006600C8      286     (1)
DS$_ICERR                       =006600C0      286     (1)
DS$_IHWE                        =00660060      286     (1)
DS$_ILLCHAR                     =00660018      286     (1)
DS$_ILLPAGCNT                   =00660078      286     (1)
DS$_ILLUNIT                     =00660100      286     (1)
DS$_INIT_FAIL                   =00660140      286     (1)
DS$_INSFMEM                     =00660050      286     (1)
DS$_INVCPU                      =00660130      286     (1)
DS$_IPL2HI                      =006600B8      286     (1)    #-876    (1)
DS$_IVADDR                      =00660040      286     (1)
DS$_IVVECT                      =00660038      286     (1)
DS$_KRNLSTK                     =00660090      286     (1)
DS$_LOGIC                       =00660070      286     (1)
DS$_MCHK                        =00660088      286     (1)
DS$_MEM_ALLOC_ERR               =00660138      286     (1)
DS$_MMOFF                       =00660058      286     (1)
DS$_NEEDUNIT                    =006600F8      286     (1)
DS$_NODE                        =00660128      286     (1)
DS$_NOPCS                       =00660110      286     (1)
DS$_NORMAL                      =00660001      286     (1)
DS$_NOSUPPORT                   =006600B1      286     (1)
DS$_NOTALLOWMP                  =00660148      286     (1)
DS$_NOTDON                      =00660030      286     (1)
DS$_NOTIMP                      =006600B0      286     (1)      286    (1)    #-881    (1)
DS$_NULLSTR                     =00660010      286     (1)
DS$_OVERFLOW                    =00660008      286     (1)
DS$_POWER                       =00660098      286     (1)
DS$_PROGERR                     =00660020      286     (1)    #-495    (1)    #-575    (1)
DS$_SEVERE                      =00660004      286     (1)
DS$_TRANSL                      =006600A0      286     (1)
DS$_TRUNCATE                    =00660028      286     (1)
DS$_UNEXPINT                    =006600D8      286     (1)
DS$_UNSUPPDEV                   =00660158      286     (1)
DS$_VASFULL                     =00660048      286     (1)
DS$_WARNING                     =00660000      286     (1)      286    (1)
DSI$TIMER                        0000031C-R    991     (1)
DSI$TIMER_X                      000003B1-R    1053    (1)
DSI$TIMSRV                       00000298-R    929     (1)
DSR$SETIMR_INI                   000001BE-R    762     (1)
DSX$CANWAIT                      00000000-R    394     (1)
DSX$WAITMS                       00000029-R    481     (1)
```

```
DSX$WAITMS_X                    000000B3-R   518   (1)    #-492    (1)    #-498    (1)    #-501    (1)
DSX$WAITUS                      000000B4-R   563   (1)
DSX$WAITUS_USOVR                000000CF-R   571   (1)
DSX$WAITUS_X                    00000134-R   610   (1)    #-577    (1)
DS_ERRSUP                       00000000-XR                   1017    (1)
DYN$C_TQE                      =0000000F                      339    (1)
EXE$ALLOCTQE                    00000000-XR                 #-887    (1)
EXE$CANTIM                      0000016E-R   699   (1)
EXE$DEANONPAGED                 00000000-XR                 #-1034   (1)    #-730    (1)
EXE$GL_ABSTIM                   00000000-XR                 #-1063   (1)    #-1080   (1)
EXE$GQ_SYSTIME                  00000000-XR                 #-1004   (1)    #-1007   (1)    #-1160   (1)    #-766    (1)
                                                            #-897    (1)    #-953    (1)    #-954    (1)    #-964    (1)
                                                            #-967    (1)
EXE$HIBER                       00000135-R   645   (1)
EXE$HIBER_X                     00000157-R   652   (1)
EXE$INSTIMQ                     000001DD-R   797   (1)    #-1047   (1)    #-768    (1)    #-900    (1)
EXE$SETIMR                      0000020F-R   856   (1)
EXE$WAKE                        0000001B-R   433   (1)
GETTIM$_NARGS                  =00000001     283   (1)
GETTIM$_TIMADR                 =00000004     283   (1)
ICCS$M_ERR                     =80000000     306   (1)    #-1136   (1)     308     (1)    #-580    (1)    #-584    (1)
                                                            #-592    (1)    #-662    (1)    #-940    (1)
ICCS$M_IE                      =00000040     304   (1)    #-1140   (1)     308     (1)
ICCS$M_INT                     =00000080     305   (1)    #-1137   (1)     308     (1)    #-579    (1)    #-584    (1)
                                                            #-591    (1)    #-663    (1)    #-940    (1)
ICCS$M_RESET                   =800000C1     308   (1)    #-937    (1)
ICCS$M_RUN                     =00000001     302   (1)    #-1142   (1)     308     (1)    #-583    (1)
ICCS$M_XFER                    =00000010     303   (1)    #-1141   (1)    #-583    (1)
ID_TEMP                         00000068-R   354   (1)
IOC$GL_DEVLIST                  00000000-XR                   1066    (1)
IOC$K_DIAGPID                   00000000-XR                 #-891    (1)
IPL$_HWCLK                     =00000018                    #-998    (1)
IPL$_POWER                     =0000001F                    #-1078   (1)
IPL$_TIMER                     =00000007                    #-1014   (1)    #-1052   (1)    #-1092   (1)    #-800    (1)
                                                            #-970    (1)    #-971    (1)
KB_CHECK                        00000000-XR                 #-647    (1)    #-866    (1)
MP$GR_BOOTED_PROCESSORS         00000000-XR                 #-943    (1)
MP$_GET_PROCESSOR_ID            00000000-XR                   181    (1)
MSTIME                          00000060-R   351   (1)      349    (1)    #-497    (1)
ONESECTIM                       00000024-R   336   (1)    #-1161   (1)     767     (1)
ONE_SECOND                      000003B4-R   1062  (1)      341    (1)
PR$_ICCS                       =00000018                    #-1138   (1)    #-1143   (1)    #-580    (1)    #-585    (1)
                                                            #-590    (1)    #-663    (1)    #-937    (1)    #-941    (1)
PR$_ICR                        =0000001A                    #-596    (1)    #-664    (1)
PR$_IPL                        =00000012                    #-1014   (1)    #-1052   (1)    #-1078   (1)    #-1086   (1)
                                                            #-1092   (1)    #-564    (1)    #-576    (1)    #-586    (1)
                                                            #-709    (1)    #-734    (1)    #-800    (1)    #-812    (1)
                                                            #-998    (1)
PR$_NICR                       =00000019                    #-1139   (1)    #-582    (1)
PR$_SIRR                       =00000014                    #-970    (1)
PR780$_TODR                    =0000001B                    #-1148   (1)    #-1151   (1)
PR8SS$_BINID                   =0000005E                    #-946    (1)
PSL$S_IPL                      =00000005                    #-486    (1)    #-878    (1)
PSL$V_IPL                      =00000010                    #-486    (1)    #-878    (1)
PSL$V_IS                       =0000001A     299   (1)    #-650    (1)    #-701    (1)    #-858    (1)
SCH$QAST                        00000000-XR                 #-1030   (1)
SEC_TICK                       =00000064     314   (1)
```

```
SETIMR$_ASTADR                  =0000000C     349     (1)     #-892     (1)
SETIMR$_DAYTIM                  =00000008     349     (1)     #-882     (1)     #-896     (1)
SETIMR$_EFN                     =00000004     349     (1)     #-894     (1)
SETIMR$_NARGS                   =00000004     349     (1)
SETIMR$_REQIDT                  =00000010     349     (1)
SIZ...                          =00000001     300     (1)       300     (1)
SS$_EXQUOTA                     =0000001C                     #-565     (1)     #-865     (1)
SS$_NORMAL                      =00000001                     #-437     (1)     #-516     (1)     #-603     (1)     #-651     (1)
                                                              #-702     (1)     #-901     (1)
SYS$CANTIM                       00000000-XR                    396     (1)
SYS$CLREF                        00000000-XR                    870     (1)
SYS$GETTIM                       00000000-XR                    483     (1)       505     (1)
SYS$HIBER                        00000000-XR                    503     (1)
SYS$SETEF                        00000000-XR                   1025     (1)
SYS$SETIMR                       00000000-XR                    500     (1)
SYS$WAKE                         00000000-XR                    403     (1)
TICK_NS                         =000186A0     313     (1)     #-1156    (1)     #-496     (1)     #-953     (1)
TICK_US                         =00002710     312     (1)     #-1139    (1)       313     (1)       314     (1)
TODR_BIAS                       =10000000     310     (1)     #-1149    (1)     #-1151    (1)
TQE$B_EFN                       =00000029                     #-1024    (1)     #-895     (1)
TQE$B_RQTYPE                    =0000000B                      1015     (1)      1043     (1)     #-720     (1)     #-890     (1)
TQE$C_SSREPT                    =00000005                      340     (1)
TQE$L_AST                       =00000010                     #-1026    (1)     #-893     (1)
TQE$L_ASTPRM                    =00000014                     #-724     (1)
TQE$L_FPC                       =0000000C                      1042     (1)
TQE$L_FR3                       =00000010                     #-1041    (1)
TQE$L_PID                       =0000000C                     #-891     (1)
TQE$L_TQBL                      =00000004                     #-803     (1)
TQE$L_TQFL                      =00000000                     #-718     (1)
TQE$Q_DELTA                     =00000020                     #-1044    (1)
TQE$Q_TIME                      =00000018                     #-1004    (1)     #-1007    (1)     #-1045    (1)     #-1046    (1)
                                                              #-1161    (1)     #-799     (1)     #-806     (1)     #-809     (1)
                                                              #-964     (1)     #-967     (1)
TQE$V_REPEAT                    =00000002                     #-1043    (1)
UCB$B_DIPL                       00000052                     #-1086    (1)
UCB$L_DUETIM                     0000005C                     #-1080    (1)
UCB$L_FPC                        0000000C                     #-1088    (1)
UCB$L_FR3                        00000010                     #-1087    (1)
UCB$L_LINK                       0000002C                      1070     (1)     #-1072    (1)
UCB$M_INT                       =00000002                     #-1083    (1)
UCB$M_TIM                       =00000001                     #-1083    (1)
UCB$M_TIMOUT                    =00000040                     #-1085    (1)
UCB$V_TIM                       =00000000                     #-1079    (1)
UCB$W_STS                        00000058                      1079     (1)     #-1084    (1)     #-1085    (1)
WAITMS$_NARGS                   =00000002     297     (1)
WAITMS$_TAG                     =00000008     297     (1)     #-510     (1)
WAITMS$_TIME                    =00000004     297     (1)     #-489     (1)     #-496     (1)     #-512     (1)
WAITMS_ARGS                      0000004C-R   347     (1)       500     (1)
WAITUS$_NARGS                   =00000002     298     (1)
WAITUS$_TAG                     =00000008     298     (1)     #-599     (1)     #-601     (1)
WAITUS$_TIME                    =00000004     298     (1)     #-572     (1)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO            SIZE        DEFINITION      REFERENCES...
-----            ----        ----------      -------------
$CANTIMDEF        1           282    (1)      282    (1)
$CANTIM_S         1           396    (1)      396    (1)
$CLREF_G          1           870    (1)      870    (1)
$DDBDEF           1           284    (1)      284    (1)
$DEF              1           300    (1)
$DEFINI           1           276    (1)      276    (1)      284    (1)      285    (1)      287    (1)      288    (1)
                                              289    (1)      290    (1)      291    (1)      292    (1)      293    (1)
                                              295    (1)      296    (1)
$DS_DSADEF        5           285    (1)      285    (1)
$DS_DSDEF         3           286    (1)      286    (1)
$DS_SCBDEF        3           291    (1)      291    (1)
$DS_WAITMS_DEF    1           297    (1)      297    (1)
$DS_WAITUS_DEF    1           298    (1)      298    (1)
$DS_WAITUS_S      1           491    (1)      491    (1)
$DYNDEF           2           287    (1)      287    (1)
$EQU              1           300    (1)      286    (1)      299    (1)
$EQULS1           1           299    (1)      286    (1)      299    (1)
$EQULST           1           286    (1)      286    (1)      299    (1)
$GBLINI           2                           286    (1)      299    (1)      300    (1)
$GETTIMDEF        1           283    (1)      283    (1)
$GETTIM_S         1           483    (1)      483    (1)      505    (1)
$HIBER_S          1           503    (1)      503    (1)
$IPLDEF           1           288    (1)      288    (1)
$OFFDEF           1           282    (1)      282    (1)      283    (1)      294    (1)      297    (1)      298    (1)
                                              349    (1)
$PR780DEF         1           290    (1)      290    (1)
$PR8SSDEF         5           276    (1)      276    (1)
$PRDEF            4           189    (1)      289    (1)
$PSLDEF           2           293    (1)      293    (1)
$PUSHADR          1           483    (1)      1017   (1)      483    (1)      491    (1)      505    (1)
$PUSHTWO          1           396    (1)      396    (1)
$SETIMR           1           348    (1)      348    (1)
$SETIMRDEF        1           294    (1)      294    (1)      349    (1)
$SETIMR_G         1           500    (1)      500    (1)
$SSDEF            21          292    (1)      292    (1)
$TQEDEF           2           295    (1)      295    (1)
$UCBDEF           10          296    (1)      296    (1)
$VIELD            1                           300    (1)
$VIELD1           1           300    (1)      300    (1)
CASE              1           1016   (1)      1016   (1)
CMK               1           650    (1)      650    (1)      701    (1)      858    (1)
CMKDEF            1           299    (1)      299    (1)
DSBINT            1           564    (1)      564    (1)      709    (1)      800    (1)
DSFDEF            3           300    (1)      300    (1)
ENBINT            1           576    (1)      576    (1)      586    (1)      734    (1)      812    (1)
ERRSUP_S          1           1017   (1)      1017   (1)
MODNAM            1           361    (1)      361    (1)
SETIPL            1           998    (1)      1014   (1)      1052   (1)      1078   (1)      1086   (1)      1092   (1)
                                              998    (1)
SOFTINT           1           970    (1)      970    (1)
```

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDL            00C0      1045   (1)    1090   (1)
ADDL2           00C0      1158   (1)    953    (1)
ADWC            00D8      1046   (1)    1159   (1)    954    (1)
BBC             00E1      1043   (1)    1079   (1)
BBCC            00E5      648    (1)    660    (1)
BBS             00E0      566    (1)    650    (1)    701    (1)    858    (1)    867    (1)
BBSC            00E4      588    (1)
BBSS            00E2      971    (1)
BEQL            0013      1027   (1)    1068   (1)    1073   (1)    511    (1)    593    (1)    600    (1)    717    (1)
                          723    (1)    805    (1)    944    (1)    948    (1)    963    (1)
BGEQ            0018      498    (1)    513    (1)    880    (1)
BGEQU           001E      1150   (1)
BGTR            0014      1081   (1)    883    (1)
BGTRU           001A      1006   (1)    1008   (1)    808    (1)    966    (1)    968    (1)
BICB2           008A      569    (1)    873    (1)
BICL            00CA      1154   (1)
BICW            00AA      1083   (1)
BISB            0088      435    (1)
BISW            00A8      1085   (1)
BITL            00D3      591    (1)
BLBC            00E9      492    (1)    501    (1)    871    (1)    888    (1)
BLBS            00E8      930    (1)
BLSS            0019      488    (1)    574    (1)
BLSSU           001F      1005   (1)    807    (1)    810    (1)    965    (1)
BNEQ            0012      1001   (1)    721    (1)    725    (1)
BRB             0011      1018   (1)    1031   (1)    1035   (1)    1093   (1)    493    (1)    577    (1)    594    (1)
                          650    (1)    701    (1)    731    (1)    858    (1)    938    (1)
BRW             0031      1002   (1)    1049   (1)    950    (1)
BSBB            0010      768    (1)
BSBW            0030      1030   (1)    1034   (1)    1047   (1)    606    (1)    647    (1)    665    (1)    730    (1)
                          866    (1)    887    (1)    900    (1)
CALLG           00FA      500    (1)    870    (1)
CALLS           00FB      1017   (1)    1025   (1)    396    (1)    403    (1)    483    (1)    491    (1)    503    (1)
                          505    (1)
CASEW           00AF      1016   (1)
CHMK            00BC      650    (1)    701    (1)    858    (1)
CLRB            0094      890    (1)
CLRL            00D4      1029   (1)    1152   (1)    514    (1)    581    (1)    650    (1)    701    (1)    858    (1)
CLRQ            007C      402    (1)
CMPB            0091      947    (1)
CMPL            00D1      1000   (1)    1004   (1)    1007   (1)    1080   (1)    716    (1)    724    (1)    804    (1)
                          806    (1)    809    (1)    962    (1)    964    (1)    967    (1)
CMPZV           00ED      486    (1)    878    (1)
CVTWL           0032      1089   (1)
DIVL            00C6      597    (1)
EDIV            007B      509    (1)
EMUL            007A      1156   (1)    496    (1)    572    (1)
EXTZV           00EF      1015   (1)
INCL            00D6      1063   (1)
INSQUE          000E      811    (1)
```

```
JSB             0016    1042 (1)  1091 (1)  650 (1)  701 (1)  858 (1)
MFPR            00DB    1148 (1)  564 (1)  590 (1)  596 (1)  664 (1)  709 (1)  800 (1)
                        946 (1)
MOVAL           00DE    1066 (1)  1070 (1)  710 (1)  763 (1)  767 (1)  801 (1)
MOVB            0090    894 (1)
MOVL            00D0    1033 (1)  1067 (1)  1072 (1)  1088 (1)  437 (1)  495 (1)  510 (1)
                        516 (1)  565 (1)  575 (1)  601 (1)  603 (1)  651 (1)  702 (1)
                        711 (1)  764 (1)  765 (1)  802 (1)  803 (1)  865 (1)  876 (1)
                        881 (1)  889 (1)  891 (1)  901 (1)  961 (1)  999 (1)
MOVPSL          00DC    485 (1)  650 (1)  701 (1)  858 (1)  877 (1)
MOVQ            007D    1041 (1)  1044 (1)  1065 (1)  1087 (1)  1095 (1)  1160 (1)  1161 (1)
                        506 (1)  766 (1)  799 (1)  882 (1)  892 (1)  896 (1)  897 (1)
MOVZBL          009A    1024 (1)
MTPR            00DA    1014 (1)  1052 (1)  1078 (1)  1086 (1)  1092 (1)  1136 (1)  1139 (1)
                        1140 (1)  1151 (1)  564 (1)  576 (1)  579 (1)  582 (1)  583 (1)
                        586 (1)  662 (1)  709 (1)  734 (1)  800 (1)  812 (1)  937 (1)
                        940 (1)  970 (1)  998 (1)
MULL3           00C5    489 (1)
POPL            8ED0    715 (1)  949 (1)  952 (1)  973 (1)
POPR            00BA    1054 (1)  607 (1)
PUSHAL          00DF    1017 (1)
PUSHAQ          007F    483 (1)  505 (1)
PUSHL           00DD    1017 (1)  396 (1)  491 (1)  713 (1)  718 (1)  945 (1)  960 (1)
PUSHR           00BB    605 (1)  992 (1)
REI             0002    1055 (1)  668 (1)  735 (1)  903 (1)  974 (1)
REMQUE          000F    1013 (1)  729 (1)
RET             0004    404 (1)  438 (1)  519 (1)  611 (1)  653 (1)  703 (1)  859 (1)
ROTL            009C    1155 (1)
RSB             0005    1096 (1)  1163 (1)  769 (1)  813 (1)
SBWC            00D9    508 (1)  899 (1)
SUBL2           00C2    1149 (1)  507 (1)  898 (1)
SUBL3           00C3    512 (1)
TSTB            0095    720 (1)
TSTL            00D5    1026 (1)  599 (1)  722 (1)  943 (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	349	(1)														
.ALIGN	928	(1)	990	(1)												
.ASCIC	361	(1)														
.ASCII	332	(1)														
.BLKL	352	(1)														
.BLKQ	329	(1)														
.BYTE	339	(1)	340	(1)												
.DISABLE	608	(1)														
.ENABLE	562	(1)														
.END	1166	(1)														
.ENDC	1014	(1)	1017	(1)	1052	(1)	1078	(1)	1086	(1)	1092	(1)	286	(1)	299	(1)
	300	(1)	396	(1)	483	(1)	491	(1)	505	(1)	564	(1)	576	(1)	586	(1)
	709	(1)	734	(1)	800	(1)	812	(1)	998	(1)						
.ENDM	1016	(1)	1017	(1)	274	(1)	276	(1)	282	(1)	283	(1)	284	(1)	285	(1)
	286	(1)	287	(1)	288	(1)	290	(1)	291	(1)	292	(1)	293	(1)	294	(1)
	295	(1)	296	(1)	297	(1)	298	(1)	299	(1)	300	(1)	348	(1)	349	(1)
	361	(1)	396	(1)	483	(1)	491	(1)	500	(1)	503	(1)	564	(1)	576	(1)
	650	(1)	870	(1)	970	(1)	998	(1)								
.ENDR	1016	(1)	286	(1)	299	(1)	300	(1)								
.ENTRY	394	(1)	433	(1)	481	(1)	563	(1)	645	(1)	699	(1)	856	(1)		
.EXTRN	181	(1)														
.GLOBL	396	(1)	483	(1)	500	(1)	503	(1)	505	(1)	870	(1)				
.IDENT	6	(1)														
.IF	1014	(1)	1017	(1)	1052	(1)	1078	(1)	1086	(1)	1092	(1)	286	(1)	299	(1)
	300	(1)	396	(1)	483	(1)	491	(1)	505	(1)	564	(1)	576	(1)	586	(1)
	709	(1)	734	(1)	800	(1)	812	(1)	998	(1)						
.IFF	1014	(1)	1017	(1)	1052	(1)	1078	(1)	1086	(1)	1092	(1)	286	(1)	299	(1)
	300	(1)	396	(1)	483	(1)	491	(1)	505	(1)	564	(1)	576	(1)	586	(1)
	709	(1)	734	(1)	800	(1)	812	(1)	998	(1)						
.IIF	1017	(1)	286	(1)	299	(1)	300	(1)								
.IRP	1016	(1)	282	(1)	283	(1)	286	(1)	294	(1)	297	(1)	298	(1)	299	(1)
	300	(1)	349	(1)												
.LIBRARY	174	(1)	175	(1)	176	(1)										
.LIST	282	(1)	283	(1)	285	(1)	291	(1)	294	(1)	297	(1)	298	(1)	349	(1)
.LONG	325	(1)	326	(1)	337	(1)	341	(1)	342	(1)	343	(1)	344	(1)	345	(1)
	354	(1)														
.MACRO	189	(1)	286	(1)	299	(1)	300	(1)								
.MDELETE	286	(1)	297	(1)	298	(1)	299	(1)								
.NLIST	282	(1)	283	(1)	285	(1)	291	(1)	294	(1)	297	(1)	298	(1)	349	(1)
.NOCROSS	276	(1)	284	(1)	285	(1)	287	(1)	288	(1)	289	(1)	290	(1)	291	(1)
	292	(1)	293	(1)	295	(1)	296	(1)								
.NOSHOW	7	(1)														
.PAGE	1019	(1)	1056	(1)	1097	(1)	1129	(1)	168	(1)	317	(1)	357	(1)	362	(1)
	405	(1)	439	(1)	480	(1)	520	(1)	561	(1)	612	(1)	644	(1)	669	(1)
	698	(1)	736	(1)	770	(1)	814	(1)	855	(1)	904	(1)	927	(1)	975	(1)
	989	(1)														
.PSECT	318	(1)	359	(1)	364	(1)										
.RESTORE	276	(1)	284	(1)	285	(1)	287	(1)	288	(1)	289	(1)	290	(1)	291	(1)
	292	(1)	293	(1)	295	(1)	296	(1)								
.SAVE	276	(1)	284	(1)	285	(1)	287	(1)	288	(1)	289	(1)	290	(1)	291	(1)

```
                 292    (1)    293    (1)    295    (1)    296    (1)
.SBTTL           1098   (1)    169    (1)    319    (1)    358    (1)    363    (1)    406    (1)    440    (1)    521    (1)
                 613    (1)    670    (1)    737    (1)    815    (1)    905    (1)    976    (1)
.SIGNED_WORD     1016   (1)
.TITLE           5      (1)
.WORD            338    (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	13	#-489	(1)	#-496	(1)	#-510	(1)	#-512	(1)	#-572	(1)	#-599	(1)
		#-601	(1)	#-711	(1)	870	(1)	#-882	(1)	#-892	(1)	#-894	(1)
		#-896	(1)										
R0	75	#-1013	(1)	#-1015	(1)	#-1016	(1)	#-1033	(1)	#-1044	(1)	#-1045	(1)
		#-1054	(1)	#-1148	(1)	#-1149	(1)	#-1152	(1)	#-1154	(1)	#-1155	(1)
		#-1156	(1)	#-1157	(1)	#-1158	(1)	#-1160	(1)	#-1161	(1)	#-437	(1)
		#-485	(1)	487	(1)	#-490	(1)	#-491	(1)	#-492	(1)	#-495	(1)
		#-501	(1)	#-506	(1)	#-507	(1)	#-509	(1)	#-510	(1)	#-512	(1)
		#-514	(1)	#-516	(1)	#-565	(1)	#-573	(1)	#-575	(1)	#-582	(1)
		#-590	(1)	#-592	(1)	#-603	(1)	#-605	(1)	#-607	(1)	#-651	(1)
		#-702	(1)	#-715	(1)	#-716	(1)	#-718	(1)	#-720	(1)	#-724	(1)
		729	(1)	#-763	(1)	#-764	(1)	#-765	(1)	#-766	(1)	#-799	(1)
		#-809	(1)	#-865	(1)	#-871	(1)	#-876	(1)	#-881	(1)	#-888	(1)
		#-897	(1)	#-898	(1)	#-901	(1)	#-960	(1)	#-961	(1)	#-962	(1)
		#-964	(1)	#-967	(1)	#-973	(1)	#-992	(1)				
R1	19	#-1046	(1)	#-1054	(1)	#-1159	(1)	#-508	(1)	#-509	(1)	#-512	(1)
		#-581	(1)	#-596	(1)	#-597	(1)	#-601	(1)	#-605	(1)	#-607	(1)
		#-664	(1)	#-806	(1)	#-877	(1)	879	(1)	#-882	(1)	#-899	(1)
		#-992	(1)										
R2	15	#-1029	(1)	#-1054	(1)	#-1088	(1)	#-1089	(1)	#-1090	(1)	1091	(1)
		699	(1)	856	(1)	#-889	(1)	#-945	(1)	#-946	(1)	#-947	(1)
		#-949	(1)	#-952	(1)	#-992	(1)						
R3	13	#-1041	(1)	#-1054	(1)	#-1087	(1)	699	(1)	#-802	(1)	#-803	(1)
		#-804	(1)	#-806	(1)	#-809	(1)	811	(1)	856	(1)	#-992	(1)
R4	10	#-1054	(1)	699	(1)	#-711	(1)	#-722	(1)	#-724	(1)	#-801	(1)
		#-802	(1)	#-804	(1)	856	(1)	#-992	(1)				
R5	42	#-1000	(1)	#-1004	(1)	#-1007	(1)	1013	(1)	1015	(1)	#-1024	(1)
		#-1026	(1)	#-1033	(1)	#-1041	(1)	1042	(1)	1043	(1)	#-1044	(1)
		#-1045	(1)	#-1046	(1)	#-1054	(1)	#-1065	(1)	#-1070	(1)	#-1072	(1)
		1079	(1)	#-1080	(1)	#-1084	(1)	#-1085	(1)	#-1086	(1)	#-1087	(1)
		#-1088	(1)	#-1095	(1)	699	(1)	#-710	(1)	#-713	(1)	#-716	(1)
		#-767	(1)	#-799	(1)	811	(1)	856	(1)	#-889	(1)	#-890	(1)
		#-891	(1)	#-893	(1)	#-895	(1)	#-992	(1)	#-999	(1)		
R6	4	#-1066	(1)	#-1067	(1)	1070	(1)						
SP	30	#-1024	(1)	#-1065	(1)	#-1089	(1)	#-1090	(1)	#-1095	(1)	#-402	(1)
		483	(1)	505	(1)	#-506	(1)	#-507	(1)	#-508	(1)	#-564	(1)
		#-576	(1)	#-586	(1)	#-650	(1)	#-701	(1)	#-709	(1)	#-734	(1)
		#-800	(1)	#-812	(1)	#-858	(1)	#-896	(1)	#-898	(1)	#-899	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.01	00:00:00.19
Command processing	889	00:00:00.21	00:00:00.82
Pass 1	1134	00:00:04.22	00:00:06.80
Symbol table sort	0	00:00:00.36	00:00:00.35
Pass 2	67	00:00:01.17	00:00:02.10

Symbol table output	2	00:00:00.08	00:00:00.23
Psect synopsis output	2	00:00:00.01	00:00:00.04
Cross-reference output	8	00:00:00.36	00:00:00.61
Assembler run totals	2127	00:00:06.42	00:00:11.14

The working set limit was 3400 pages.
239143 bytes (468 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1241 non-local and 46 local symbols.
1166 source lines were read in Pass 1, producing 104 object records in Pass 2.
151 pages of virtual memory were used to define 44 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	11
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	5
SYS$COMMON:[SYSLIB]LIB.MLB;2	5
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	21
TOTALS (all libraries)	42

1464 GETS were required to define 42 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CLOCK.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

Table of contents

(2) 79 COM$DELATTNAST - DELIVER ATTENTION ASTS
(3) 117 COM$FLUSHATTNS - FLUSH ATTENTION AST LIST
(4) 159 COM$POST - POST I.O COMPLETION INDEPENDENT OF UNIT STATUS
(5) 190 COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
(6) 235 COM$SETATTNAST - SET UP ATTENTION AST

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element COMDRVSUB.MAR
0000     2 ; *1     6-FEB-1986 15:15:12 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element COMDRVSUB.MAR
0000     4         .TITLE  COMDRVSUB       *** COMDRVSUB driver support routines
0000     5         .IDENT  /06.01/
0000     6 ;
0000     7 ;*****************************************************************************
0000     8 ;*                                                                           *
0000     9 ;*  COPYRIGHT (c) 1978, 1979, 1980                                           *
0000    10 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    11 ;*                                                                           *
0000    12 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    13 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    14 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    15 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    16 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    17 ;*  TRANSFERRED.                                                             *
0000    18 ;*                                                                           *
0000    19 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    20 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    21 ;*  CORPORATION.                                                             *
0000    22 ;*                                                                           *
0000    23 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    24 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;++
0000    29 ; FACILITY:
0000    30 ;
0000    31 ;       VAX/VMS I/O DRIVERS
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;
0000    35 ;       THIS MODULE CONTAINS SUBROUTINES FOR THE TERMINAL,MAILBOX AND DMC11 DRIVERS.
0000    36 ;
0000    37 ; AUTHOR:
0000    38 ;
0000    39 ;       R.HEINEN 8-SEPT-1977
0000    40 ;
0000    41 ; REVISION HISTORY:
0000    42 ;
0000    43 ;       V07     TCM0001         Trudy C Matthews        2-Jan-1980
0000    44 ;               Changed COM$SETATTNAST so that it only extracts the
0000    45 ;               two bits of the access mode in the P3 parameter,
0000    46 ;               instead of taking the whole byte.
0000    47 ;
0000    48 ;
0000    49 ;               Dave Butenhof
0000    50 ;       01      Update VMS module for Supervisor; delete quota checks
0000    51 ;--
0000    52 ;
0000    53 ; EXTERNAL SYMBOLS
0000    54 ;
0000    55         $ACBDEF                         ; DEFINE AST CONTROL BLOCK
0000               .SAVE   LOCAL_BLOCK
0000               .IIF    NB,ACB$L_ASTQFL,        ACB$L_ASTQFL:
```

```
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,ACB$L_ASTQBL,        ACB$L_ASTQBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,ACB$W_SIZE,  ACB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,ACB$B_TYPE,  ACB$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,ACB$B_RMOD,  ACB$B_RMOD:
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,ACB$L_PID,   ACB$L_PID:
00000010  000C                                  .BLKL   1
          0010                  .IIF    NB,ACB$L_AST,   ACB$L_AST:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,ACB$L_ASTPRM,        ACB$L_ASTPRM:
00000018  0014                                  .BLKL   1
          0018                  .IIF    NB,ACB$C_LENGTH,        ACB$C_LENGTH:
          0018                  .IIF    NB,ACB$K_LENGTH,        ACB$K_LENGTH:
          0018                  .IIF    NB,ACB$L_KAST,  ACB$L_KAST:
0000001C  0018                                  .BLKL   1
          0000                  .RESTORE
          0000     56           $CCBDEF                         ; DEFINE CCB
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  001C                  .=0
          0000                  .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,CCB$B_STS,   CCB$B_STS:
00000009  0008                                  .BLKB   1
          0009                  .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
0000000A  0009                                  .BLKB   1
          000A                  .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
0000000C  000A                                  .BLKW   1
          000C                  .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
00000010  000C                                  .BLKL   1
          0010                  .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
          0010                  .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
          0000                  .RESTORE
          0000     57           $DYNDEF                         ; DEFINE DYNAMIC MEMORY BLOCKS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  001C                  .=0
          0000                  .RESTORE
          0000     58           $IPLDEF                         ; DEFINE IPL LEVELS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  001C                  .=0
          0000                  .RESTORE
          0000     59           $IRPDEF                         ; DEFINE I/O PACKET
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  001C                  .=0
          0000                  .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
00000008  0004                                  .BLKL   1
```

```
                     0008          .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
0000000A  0008                             .BLKW   1
                     000A          .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
0000000B  000A                             .BLKB   1
                     000B          .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
0000000C  000B                             .BLKB   1
                     000C          .IIF    NB,IRP$L_PID,   IRP$L_PID:
00000010  000C                             .BLKL   1
                     0010          .IIF    NB,IRP$L_AST,   IRP$L_AST:
00000014  0010                             .BLKL   1
                     0014          .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
00000018  0014                             .BLKL   1
                     0018          .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
0000001C  0018                             .BLKL   1
                     001C          .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
00000020  001C                             .BLKL   1
                     0020          .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
00000022  0020                             .BLKW   1
                     0022          .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
00000023  0022                             .BLKB   1
                     0023          .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
00000024  0023                             .BLKB   1
                     0024          .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
00000028  0024                             .BLKL   1
                     0028          .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
0000002A  0028                             .BLKW   1
                     002A          .IIF    NB,IRP$W_STS,   IRP$W_STS:
0000002C  002A                             .BLKW   1
                     002C          .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
00000030  002C                             .BLKL   1
                     0030          .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
00000032  0030                             .BLKW   1
                     0032          .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
00000034  0032                             .BLKW   1
                     0034          .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
                     0034          .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
00000038  0034                             .BLKL   1
                     0038          .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
                     0038          .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
                     0038          .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
00000039  0038                             .BLKB   1
0000003C  0039                             .BLKB   3
                     003C          .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
                     003C          .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
                     003C          .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                             .BLKW   1
                     003E          .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                             .BLKW   1
                     0040          .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                             .BLKL   1
                     0044          .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                             .BLKL   1
                     0048          .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                             .BLKL   1
                     004C          .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                             .BLKL   1
                     0050          .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
```

```
00000054  0050                                .BLKL   1
00000058  0054                                .BLKL   1
0000005C  0058                                .BLKL   1
          005C                .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
          005C                .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
          0000                .RESTORE
          0000    60          $PCBDEF                         ; DEFINE PCB
          0000                .SAVE   LOCAL_BLOCK
          0000                .PSECT  $ABS$,ABS
00000000  005C                .=0
          0000                .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004  0000                        .BLKL   1
          0004                .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008  0004                        .BLKL   1
          0008                .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A  0008                        .BLKW   1
          000A                .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B  000A                        .BLKB   1
          000B                .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C  000B                        .BLKB   1
          000C                .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D  000C                        .BLKB   1
          000D                .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E  000D                        .BLKB   1
          000E                .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010  000E                        .BLKW   1
          0010                .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014  0010                        .BLKL   1
          0014                .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018  0014                        .BLKL   1
          0018                .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C  0018                        .BLKL   1
          001C                .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020  001C                        .BLKL   1
          0020                .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024  0020                        .BLKL   1
          0024                .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028  0024                        .BLKL   1
          0028                .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C  0028                        .BLKL   1
          002C                .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E  002C                        .BLKW   1
          002E                .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                        .BLKB   1
          002F                .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                        .BLKB   1
          0030                .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                        .BLKW   1
          0032                .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                        .BLKW   1
          0034                .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                        .BLKW   1
          0036                .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                        .BLKW   1
          0038                .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                        .BLKW   1
          003A                .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
```

```
0000003C  003A                          .BLKW   1
                003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                          .BLKW   1
                003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                          .BLKW   1
                0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                          .BLKW   1
                0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                          .BLKW   1
                0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                          .BLKB   8
                004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                004C            .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                          .BLKL   1
                0050            .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                          .BLKL   1
                0054            .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                          .BLKL   1
                0058            .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                          .BLKL   1
                005C            .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                          .BLKL   1
                0060            .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                          .BLKL   1
                0064            .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                          .BLKL   1
                0068            .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                          .BLKB   16
                0078            .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                          .BLKL   1
                007C            .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084  007C                          .BLKQ   1
                0084            .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                          .BLKL   1
                0088            .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                0088            .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                          .BLKW   1
                008A            .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                          .BLKW   1
                008C            .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                008C            .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                0000            .RESTORE
                0000    61      $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                0000            .SAVE   LOCAL_BLOCK
                0000            .PSECT  $ABS$,ABS
00000000  008C                  .=0
                0000            .RESTORE
                0000    62      $PRIDEF                         ; DEFINE NEW PRIORITIES
                0000            .SAVE   LOCAL_BLOCK
                0000            .PSECT  $ABS$,ABS
00000000  008C                  .=0
                0000            .RESTORE
                0000    63      $PRVDEF                         ; DEFINE PRIVELEGE BITS
                0000            .SAVE   LOCAL_BLOCK
                0000            .PSECT  $ABS$,ABS
00000000  008C                  .=0
                0000            .RESTORE
```

```
                        0000     64          $PSLDEF                                 ; DEFINE PSL
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 008C                .=0
                        0000                 .RESTORE
                        0000     65          $RSNDEF                                 ; DEFINE RESOURCES
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 008C                .=0
                        0000                 .RESTORE
                        0000     66          $UCBDEF                                 ; DEFINE UCB
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 008C                .=0
                        0000                 .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                        0000                 .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                        0004                 .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                0000000B 000A                        .BLKB   1
                        000B                 .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                0000000C 000B                        .BLKB   1
                        000C                 .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                        000C                 .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                        000C                 .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                0000000D 000C                        .BLKB   1
                00000010 000D                        .BLKB   3
                        0010                 .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                        0010                 .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
                00000014 0010                        .BLKL   1
                        0014                 .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                        0014                 .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                        0014                 .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
                00000016 0014                        .BLKW   1
                        0016                 .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
                00000018 0016                        .BLKW   1
                        0018                 .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                        0018                 .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
                0000001A 0018                        .BLKW   1
                        001A                 .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                        001A                 .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                0000001C 001A                        .BLKW   1
                        001C                 .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                00000020 001C                        .BLKL   1
                        0020                 .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                00000024 0020                        .BLKL   1
                        0024                 .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                00000028 0024                        .BLKL   1
                        0028                 .IIF    NB,UCB$L_PID,   UCB$L_PID:
                0000002C 0028                        .BLKL   1
                        002C                 .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
                00000030 002C                        .BLKL   1
                        0030                 .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
```

```
00000034  0030                                  .BLKL   1
          0034                  .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039  0038                                  .BLKB   1
          0039                  .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A  0039                                  .BLKB   1
          003A                  .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C  003A                                  .BLKW   1
          003C                  .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
          003C                  .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
          003C                  .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D  003C                                  .BLKB   1
          003D                  .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
          003D                  .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E  003D                                  .BLKB   1
          003E                  .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
          003E                  .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F  003E                                  .BLKB   1
          003F                  .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040  003F                                  .BLKB   1
          0040                  .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                                  .BLKW   1
          004A                  .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
          004A                  .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                                  .BLKB   1
          004B                  .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                                  .BLKB   1
          004C                  .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                  .BLKW   1
          0052                  .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
          0052                  .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                  .BLKB   1
          0053                  .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                  .BLKB   1
          0054                  .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
          005A                  .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
          005C                  .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C                  .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                  .BLKL   1
          0060                  .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                                  .BLKL   1
          0064                  .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
          0064                  .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                                  .BLKL   1
          0068                  .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                                  .BLKL   1
```

```
                    006C            .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                              .BLKW   1
                    006E            .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                              .BLKW   1
                    0070            .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                              .BLKB   1
                    0071            .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                              .BLKB   1
                    0072            .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                              .BLKW   1
                    0074            .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                    0074            .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074                  . = 0
                    0000            .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                              .BLKW   1
00000074  0002                  . = 116
                    0074            .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                              .BLKL   1
                    0078            .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                              .BLKL   1
                    007C            .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                              .BLKL   1
                    0080            .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                              .BLKL   1
                    0084            .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                              .BLKL   1
                    0088            .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                              .BLKL   1
                    008C            .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                              .BLKL   1
                    0090            .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                    0090            .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                  . = 116
                    0074            .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                              .BLKB   1
                    0075            .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                              .BLKB   1
                    0076            .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                              .BLKB   1
                    0077            .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                              .BLKB   1
                    0078            .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                              .BLKL   1
0000007E  007C                              .BLKW   1
                    007E            .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                              .BLKW   1
                    0080            .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                              .BLKL   1
00000080  0084                  . = 128
00000084  0080                              .BLKL   1
                    0084            .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                              .BLKL   1
                    0088            .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                              .BLKW   1
                    008A            .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                              .BLKW   1
                    008C            .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
```

```
                     008C                .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                   .BLKW   1
                     008E                .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                   .BLKW   1
                     0090                .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                   .BLKW   1
                     0092                .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                   .BLKW   1
                     0094                .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                   .BLKB   1
                     0095                .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                   .BLKB   1
                     0096                .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                   .BLKW   1
00000096  0098                           . = 150
00000098  0096                                   .BLKW   1
                     0098                .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                   .BLKL   1
                     009C                .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                   .BLKL   1
                     00A0                .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                                   .BLKL   1
                     00A4                .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                                   .BLKW   1
                     00A6                .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                                   .BLKB   1
000000A8  00A7                                   .BLKB   1
00000074  00A8                           . = 116
00000078  0074                                   .BLKL   1
0000007C  0078                                   .BLKL   1
00000080  007C                                   .BLKL   1
00000084  0080                                   .BLKL   1
00000088  0084                                   .BLKL   1
0000008C  0088                                   .BLKL   1
                     008C                .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                   .BLKL   1
00000094  0090                                   .BLKL   1
00000098  0094                                   .BLKL   1
0000009C  0098                                   .BLKL   1
                     009C                .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                   .BLKB   1
                     009D                .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                   .BLKB   1
                     009E                .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                   .BLKB   1
000000A0  009F                                   .BLKB   1
                     00A0                .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                   .BLKB   1
                     00A1                .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                   .BLKB   1
                     00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                   .BLKB   1
000000A4  00A3                                   .BLKB   1
                     00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                   .BLKB   1
                     00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                   .BLKW   1
```

```
000000A8  00A7                          .BLKB   1
          00A8              .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                          .BLKL   1
000000B0  00AC                          .BLKL   1
000000B4  00B0                          .BLKL   1
000000B8  00B4                          .BLKL   1
          00B8              .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                          .BLKL   1
          00BC              .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC              .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC              . = 116
          0074              .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                          .BLKL   1
          0078              .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                          .BLKL   1
          007C              .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                          .BLKW   1
00000080  007E                          .BLKW   1
          0000              .RESTORE
          0000    67 ;
          0000    68 ; LOCAL DEFINITIONS
          0000    69 ;
00000000  0000    70 P1= 0
00000004  0000    71 P2= 4
00000008  0000    72 P3= 8
0000000C  0000    73 P4= 12
00000010  0000    74 P5= 16
00000014  0000    75 P6= 20
          0000    76
        00000000  77          .PSECT  SEP, SHR,EXE,RD,WRT,LONG
```

```
                                0000     79          .SBTTL  COM$DELATTNAST - DELIVER ATTENTION ASTS
                                0000     80 ;++
                                0000     81 ; COM$DELATTNAST - DELIVER ATTENION ASTS
                                0000     82 ;
                                0000     83 ; FUNCTIONAL DESCRIPTION:
                                0000     84 ;
                                0000     85 ; THIS ROUTINE IS USED BY THE TERMINAL AND MAILBOX DRIVERS TO DELIVER
                                0000     86 ; ALL OF THE ASTS AWAITING ATTENTION. THE CONTROL BLOCKS ARE USED AS FORK BLOCKS
                                0000     87 ; TO IPL$_QUEUEAST.
                                0000     88 ;
                                0000     89 ; INPUTS:
                                0000     90 ;
                                0000     91 ;       R4 = ADDRESS OF LIST HEAD OF AST CONTROL BLOCKS
                                0000     92 ;       R5 = UCB OF UNIT
                                0000     93 ;
                                0000     94 ; OUTPUTS:
                                0000     95 ;
                                0000     96 ;       R2,R3,R4,R5 ARE PRESERVED.
                                0000     97 ;--
                                0000     98 COM$DELATTNAST::                             ; DELIVER ATTENTION ASTS
                     38   BB    0000     99          PUSHR   #^M<R3,R4,R5>               ;
          55    04 BE   D0    0002    100 10$:     MOVL    @4(SP),R5                   ; GET NEXT ENTRY
                     28   13    0006    101          BEQL    50$                         ; IF EQL THEN NONE
          04 BE      65   D0    0008    102          MOVL    (R5),@4(SP)                 ; CLOSE LIST
               F3 AF    9F    000C    103          PUSHAB  B^10$                       ; SET UP RETURN ADDRESS
                                000F    104          FORK                                ; CREATE FORK PROCESS
                00000000'GF    16    000F                       JSB     G^EXE$FORK
                                0015    105 ;
                                0015    106 ; AST QUEUE FORK PROCESS
                                0015    107 ;
     10 A5    18 A5    7D    0015    108          MOVQ    ACB$L_KAST(R5),ACB$L_AST(R5); REARRANGE ENTRIES
     0B A5    20 A5    90    001A    109          MOVB    ACB$L_KAST+8(R5),ACB$B_RMOD(R5);
     0C A5    24 A5    D0    001F    110          MOVL    ACB$L_KAST+12(R5),ACB$L_PID(R5);
              18 A5    D4    0024    111          CLRL    ACB$L_KAST(R5)              ;
                 52    01    9A    0027    112          MOVZBL  #PRI$_IOCOM,R2              ; SET UP PRIORITY INCREMENT
        00000000'GF    17    002A    113          JMP     G^SCH$QAST                  ; QUEUE THE AST
                     38   BA    0030    114 50$:     POPR    #^M<R3,R4,R5>               ;
                          05    0032    115          RSB
```

```
                                0033   117          .SBTTL  COM$FLUSHATTNS - FLUSH ATTENTION AST LIST
                                0033   118 ;++
                                0033   119 ; COM$FLUSHATTNS - FLUNS ATTENTION AST LIST
                                0033   120 ;
                                0033   121 ; THIS ROUTINE IS USED BY THE TERMINAL AND MAILBOX DRIVERS TO FLUSH
                                0033   122 ; AN ATTENTION AST LIST. THIS IS DONE AT CANCEL I/O TIME AND WHEN A
                                0033   123 ; QIO SPECIFIES A 0 AST ADDRESS ON A SET ATTENTION AST FUNCTION.
                                0033   124 ; IF THE AST CONTROL BLOCK OWNER IS NO LONGER IN THE SYSTEM THE AST IS ALSO
                                0033   125 ; FLUSHED.
                                0033   126 ;
                                0033   127 ;
                                0033   128 ; INPUTS:
                                0033   129 ;
                                0033   130 ;       R4 = PCB ADDRESS
                                0033   131 ;       R5 = UCB ADDRESS OF RELATED UNIT
                                0033   132 ;       R6 = CHANNEL NUMBER
                                0033   133 ;       R7 = LIST HEAD
                                0033   134 ;
                                0033   135 ; OUTPUTS:
                                0033   136 ;
                                0033   137 ;       R0 = SS$_NORMAL
                                0033   138 ;       R1,R2,R7 ARE DISTROYED.
                                0033   139 ;
                                0033   140 ;--
                                0033   141 COM$FLUSHATTNS::                           ; FLUSH ATTENTION AST LIST
                                0033   142          DSBINT  UCB$B_DIPL(R5)          ; DISABLE INTERRUPTS
          7E    12    DB      0033                        MFPR    S^#PR$_IPL,-(SP)
       12   52  A5    DA      0036                        MTPR    UCB$B_DIPL(R5),S^#PR$_IPL
          50    67    D0      003A   143 10$:     MOVL    (R7),R0                 ; GET LIST ENTRY
                1C    13      003D   144          BEQL    50$                     ; IF EQL THEN DONE
    24 A0    60  A4    D1      003F   145          CMPL    PCB$L_PID(R4),ACB$L_KAST+12(R0); PID MATCH?
                10    12      0044   146          BNEQ    40$                     ; IF NEQ THEN NO
       22 A0    56    B1      0046   147          CMPW    R6,ACB$L_KAST+10(R0)    ; CHANNEL MATCH?
                0A    12      004A   148          BNEQ    40$                     ; IF NEQ THEN NO
          67    60    D0      004C   149          MOVL    (R0),(R7)               ; CLOSE UP LIST TO REMOVE ENTRY
                      004F   150          ENBINT                          ; REENABLE INTERRUPTS
          12    8E    DA      004F                        MTPR    (SP)+,S^#PR$_IPL
                1F    10      0052   151          BSBB    COM$DRVDEALMEM          ; DEALLOCATE THE BLOCK
                DD    11      0054   152          BRB     COM$FLUSHATTNS          ; CONTINUE
          57    50    D0      0056   153 40$:     MOVL    R0,R7                   ; LOOK TO NEXT ENTRY
                DF    11      0059   154          BRB     10$                     ; CONTINUE
                      005B   155 50$:     ENBINT                          ; REENABLE INTERRUPTS
          12    8E    DA      005B                        MTPR    (SP)+,S^#PR$_IPL
       50    00'8F    9A      005E   156          MOVZBL  #SS$_NORMAL,R0          ; SET NORMAL RETURN
                05    05      0062   157          RSB
```

```
                                0063   159          .SBTTL  COM$POST - POST I.O COMPLETION INDEPENDENT OF UNIT STATUS
                                0063   160 ;++
                                0063   161 ; COM$POST - POST I/O COMPLETION INDEPENDENT OF UNIT STATUS
                                0063   162 ;
                                0063   163 ; FUNCTIONAL DESCRIPTION:
                                0063   164 ;
                                0063   165 ; THIS ROUTINE IS USED BY THE TERMINAL, MAILBOX AND DMC DRIVER TO COMPLETE
                                0063   166 ; I/O OPERATIONS INDEPENDENT OF THE STATUS OF THE UNIT. NO ATTEMPT IS MADE
                                0063   167 ; TO DE-QUEUE ANOTHER PACKET OR CHANGE THE BUSY STATUS OF THE UNIT.
                                0063   168 ;
                                0063   169 ; INPUTS:
                                0063   170 ;
                                0063   171 ;       R3 = I/O PACKET ADDRESS
                                0063   172 ;       R5 = UCB ADDRESS
                                0063   173 ;
                                0063   174 ; IMPLICIT INPUTS:
                                0063   175 ;
                                0063   176 ;       CALLER AT DRIVER FORK IPL OR GREATER.
                                0063   177 ;       IRP$L_MEDIA AND IRP$L_MEDIA+4 ARE THE IOSB QUAD WORD.
                                0063   178 ;
                                0063   179 ; OUTPUTS:
                                0063   180 ;
                                0063   181 ;       R0,R1 ARE DISTROYED.
                                0063   182 ;--
                                0063   183 COM$POST::                                ; COMPLETE I/O
          60 A5   D6            0063   184          INCL    UCB$L_OPCNT(R5)         ; INCREMENT OPERATION COUNT
00000000'FF   63   0E           0066   185          INSQUE  (R3),@IOC$GL_PSBL       ; INSERT PACKET ON QUEUE
             03   12            006D   186          BNEQ    10$                     ; IF NEQ THEN NOT FIRST
                                006F   187          SOFTINT #IPL$_IOPOST            ; REQUEST FORK
       14    04   DA            006F                  MTPR    #IPL$_IOPOST,S^#PR$_SIRR
             05   0072   188 10$:    RSB                                             ; RETURN
```

```
                                0073          .SBTTL  COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
                                0073    ;++
                                0073    ; COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
                                0073    ;
                                0073    ; FUNCTIONAL DESCRIPTION:
                                0073    ;
                                0073    ; THIS ROUTINE IS USED BY DRIVERS TO DEALLOCATE SYSTEM DYNAMIC MEMORY.
                                0073    ;
                                0073    ; IT CAN BE CALLED AT ANY IPL.
                                0073    ;
                                0073    ; INPUTS:
                                0073    ;
                                0073    ;       R0 = ADDRESS OF THE BLOCK TO DEALLOCATE
                                0073    ;
                                0073    ;*******************************************************
                                0073    ;
                                0073    ; THE BUFFER MUST BE AT LEAST 24 BYTES LONG!
                                0073    ;
                                0073    ;*******************************************************
                                0073    ;
                                0073    ; OUTPUTS:
                                0073    ;
                                0073    ;       R0-R5 ARE PRESERVED.
                                0073    ;--
                                0073    COM$DRVDEALMEM::                                ; DEALLOCATE DRIVER MEMORY
        08 A0    18    B1      0073            CMPW    #24,8(R0)                       ; BIG ENOUGH BUFFER TO DEALLOCATE?
                 1E    1A      0077            BGTRU   30$                             ; IF GTRU THEN NO - ERROR
                 38    BB      0079            PUSHR   #^M<R3,R4,R5>                   ; SAVE FORKING REGS
        0B A0    06    90      007B            MOVB    #IPL$_QUEUEAST,11(R0)           ; INSERT PROPER IPL
           55    50    D0      007F            MOVL    R0,R5                           ; COPY ADDRESS
              94'AF    9F      0082            PUSHAB  B^20$                           ; SET UP RETURN ADDRESS
      00000000'GF      16      0085            JSB     G^EXE$FORK                      ; CREATE FORK
                                008B    ;
                                008B    ; IPL$_QUEUEAST FORK ROUTINE
                                008B    ;
           50    55    D0      008B            MOVL    R5,R0                           ; DEALLOCATE THE BLOCK
      00000000'GF      17      008E            JMP     G^EXE$DEANONPAGED               ;
                 38    BA      0094    20$:    POPR    #^M<R3,R4,R5>                   ;
                       05      0096            RSB                                     ;
                                0097    ;
                                0097    ; BUGCHECK ON TOO SMALL A BUFFER
                                0097    ;
                                0097    30$::   BUG_CHECK BADDALRQSZ                    ; BUGCHECK
                       05      0097            RSB                                     ; CONTINUE
```

```
                     0098   235          .SBTTL  COM$SETATTNAST - SET UP ATTENTION AST
                     0098   236 ;++
                     0098   237 ; COM$SETATTNAST - SET UP ATTENTION AST
                     0098   238 ;
                     0098   239 ; FUNCTIONAL DESCRIPTION:
                     0098   240 ;
                     0098   241 ; THIS ROUTINE IS A SUBROUTINE USED BY THE TERMINAL AND MAILBOX DRIVERS
                     0098   242 ; TO PROCESS REQUESTS FOR ENABLE OR DISABLE OF ATTENTION ASTS.
                     0098   243 ; P1 IS THE ADDRESS OF THE AST SERVICE FOR ENABLES. P1 = 0 FOR DISABLE.
                     0098   244 ; FOR DISABLES, THE SPECIFIED LIST IS SEARCHED AND THE ENTRY EXTRACTED AND
                     0098   245 ; DEALLOCATED.
                     0098   246 ; FOR ENABLES, A CONTROL BLOCK IS SET UP THAT WILL DOUBLE AS THE AST CONTROL
                     0098   247 ; BLOCK WHEN THE AST IS DELIVERED. THE BLOCK IS FORMATTED AS FOLLOWS:
                     0098   248 ;
                     0098   249 ;       ACB$B_RMOD = IPR$_QUEUEAST
                     0098   250 ;       ACB$L_KAST = AST PC
                     0098   251 ;       ACB$L_KAST+4 = AST PARAMETER (P2)
                     0098   252 ;       ACB$L_KAST+8 = ACCESS MODE OF REQUEST
                     0098   253 ;       ACB$L_KAST+10 = CHANNEL NUMBER
                     0098   254 ;       ACB$L_KAST+12 = PID OF REQUEST
                     0098   255 ;
                     0098   256 ; THE NEW BLOCK IS PLACED AT THE HEAD OF THE CURRENT LIST.
                     0098   257 ;
                     0098   258 ; IN BOTH CASES THE I/O IS COMPLETED.
                     0098   259 ;
                     0098   260 ; INPUTS:
                     0098   261 ;
                     0098   262 ;       R3 = I/O PACKET ADDRESS
                     0098   263 ;       R4 = CURRENT PCB
                     0098   264 ;       R5 = UCB ADDRESS
                     0098   265 ;       R6 = ASSIGNED CCB
                     0098   266 ;       R7 = ADDRESS OF THE CONTROL AST LIST HEAD TO CHANGE
                     0098   267 ;       AP = ADDRESS OF THE QIO ARGLIST
                     0098   268 ;
                     0098   269 ; OUTPUTS:
                     0098   270 ;
                     0098   271 ;       R0 = STATUS OF THE I/O
                     0098   272 ;       R3 = PACKET ADDRESS
                     0098   273 ;       R5 = UCB ADDRESS
                     0098   274 ;
                     0098   275 ;       NO OTHER REGISTERS ARE PRESERVED.
                     0098   276 ;
                     0098   277 ; COMPLETION CODES:
                     0098   278 ;
                     0098   279 ;       SS$_NORMAL
                     0098   280 ;       SS$_EXQUOTA  -- BUFFERED I/O OR AST QUOTA FAILURE
                     0098   281 ;       SS$_INSUFMEM -- DYNAMIC MEMORY FAILURE
                     0098   282 ;--
                     0098   283 COM$SETATTNAST::                          ; SET UP ATTENTION AST
 56   28 A3   3C     0098   284          MOVZWL  IRP$W_CHAN(R3),R6        ; GET PACKET CHANNEL NUMBER
    58   6C   D0     009C   285          MOVL    P1(AP),R8                ; GET USER AST ADDRESS
         92   13     009F   286          BEQL    COM$FLUSHATTNS           ; IF EQL THEN DISABLE FUNCTION
                     00A1   287 ;
                     00A1   288 ; REQUEST TO ENABLE AST
                     00A1   289 ;
                     00A1   290 ; SET UP AST BLOCK
                     00A1   291 ;
```

```
                51   28   9A   00A1   292          MOVZBL  #ACB$L_KAST+16,R1       ; SET SIZE OF NEEDED BLOCK
                     53   DD   00A4   293          PUSHL   R3                      ; SAVE PACKET ADDRESS
          00000000'GF   16   00A6   294          JSB     G^EXE$ALLOCBUF          ; ALLOCATE THE BUFFERED BLOCK
                     53 8ED0   00AC   295          POPL    R3                      ; RESTORE PACKET ADDRESS
                  40 50   E9   00AF   296          BLBC    R0,20$                  ; IF LOW SET THEN ALLOCATED
              0B A2   06   90   00B2   297          MOVB    #IPL$_QUEUEAST,ACB$B_RMOD(R2); INSERT FORK IPL
              18 A2   58   D0   00B6   298          MOVL    R8,ACB$L_KAST(R2)       ; INSERT AST ROUTINE ADDRESS
           1C A2   04 AC   D0   00BA   299          MOVL    P2(AP),ACB$L_KAST+4(R2) ; INSERT PARAMETER FOR AST
     50    08 AC   02   00   EF   00BF   300          EXTZV   #0,#2,P3(AP),R0         ; GET REQUEST ACCESS MODE
          00000000'GF   16   00C5   301          JSB     G^EXE$MAXACMODE         ; MAXIMIZE ACCESS MODE
              20 A2   50   9A   00CB   302          MOVZBL  R0,ACB$L_KAST+8(R2)     ; INSERT IN CONTROL BLOCK
           20 A2   40 8F   88   00CF   303          BISB    #ACB$M_QUOTA,ACB$L_KAST+8(R2); INSERT TARGET ACCESS MODE
              22 A2   56   B0   00D4   304          MOVW    R6,ACB$L_KAST+10(R2)    ; SAVE CHANNEL
           24 A2   60 A4   D0   00D8   305          MOVL    PCB$L_PID(R4),ACB$L_KAST+12(R2); INSERT PID ADDRESS OF REQUESTOR
                           00DD   306 ;
                           00DD   307 ; LOCK OUT INTERRUPTS TO ENTER BLOCK ON UCB
                           00DD   308 ;
                           00DD   309          DSBINT  UCB$B_DIPL(R5)          ; LOCK OUT INTERRUPTS
                7E   12   DB   00DD                  MFPR    S^#PR$_IPL,-(SP)
             12   52 A5   DA   00E0                  MTPR    UCB$B_DIPL(R5),S^#PR$_IPL
                62   67   D0   00E4   310          MOVL    (R7),(R2)               ;  MERGE WITH CURRENT ENTRY
                67   52   D0   00E7   311          MOVL    R2,(R7)                 ; INSERT NEW ENTRY VALUE
                           00EA   312          ENBINT                          ; LOWER IPL
                12   8E   DA   00EA                  MTPR    (SP)+,S^#PR$_IPL
             50   00'8F   9A   00ED   313          MOVZBL  #SS$_NORMAL,R0          ; SET NORMAL RETURN
                     05   00F1   314          RSB                             ; RETURN VIA CALLER
                   FF0B'  31   00F2   315 20$:     BRW     EXE$ABORTIO             ; ABORT THE I/O
                           00F5   316 ;
                           00F5   317          .END
```

ACB$B_RMOD		0000000B	D		IRP$L_WIND		00000018	D		PCB$W_MTXCNT		0000000E	D				
ACB$B_TYPE		0000000A	D		IRP$Q_NT_PRVMSK		0000003C	D		PCB$W_PPGCNT		00000036	D				
ACB$C_LENGTH		00000018	D		IRP$W_ABCNT		0000003C	D		PCB$W_PRCCNT		00000042	D				
ACB$K_LENGTH		00000018	D		IRP$W_BCNT		00000032	D		PCB$W_SIZE		00000008	D				
ACB$L_AST		00000010	D		IRP$W_BOFF		00000030	D		PCB$W_STATE		0000002C	D				
ACB$L_ASTPRM		00000014	D		IRP$W_CHAN		00000028	D		PCB$W_TMBU		00000032	D				
ACB$L_ASTQBL		00000004	D		IRP$W_FUNC		00000020	D		PR$_IPL	=	00000012	D				
ACB$L_ASTQFL		00000000	D		IRP$W_OBCNT		0000003E	D		PR$_SIRR	=	00000014	D				
ACB$L_KAST		00000018	D		IRP$W_SIZE		00000008	D		PRI$_IOCOM	=	00000001	D				
ACB$L_PID		0000000C	D		IRP$W_STS		0000002A	D		SCH$QAST		********	X	02			
ACB$M_QUOTA	=	00000040	D		IRP$W_TT_PRMPT		0000003C	D		SIZ...	=	00000002	D				
ACB$W_SIZE		00000008	D		P1	=	00000000	D		SS$_NORMAL		********	X	02			
CCB$B_AMOD		00000009	D		P2	=	00000004	D		UCB$B_AMOD		00000053	D				
CCB$B_STS		00000008	D		P3	=	00000008	D		UCB$B_CEX		00000077	D				
CCB$C_LENGTH		00000010	D		P4	=	0000000C	D		UCB$B_CM1		0000004A	D				
CCB$K_LENGTH		00000010	D		P5	=	00000010	D		UCB$B_CM2		0000004B	D				
CCB$L_DIRP		0000000C	D		P6	=	00000014	D		UCB$B_DEVCLASS		00000038	D				
CCB$L_UCB		00000000	D		PCB$B_ASTACT		0000000C	D		UCB$B_DEVTYPE		00000039	D				
CCB$L_WIND		00000004	D		PCB$B_ASTEN		0000000D	D		UCB$B_DIPL		00000052	D				
CCB$W_IOC		0000000A	D		PCB$B_PRI		0000000B	D		UCB$B_DX_SCTCNT		000000A6	D				
COM$DELATTNAST		00000000	RG	D	02	PCB$B_PRIB		0000002F	D		UCB$B_ERTCNT		00000070	D			
COM$DRVDEALMEM		00000073	RG	D	02	PCB$B_TYPE		0000000A	D		UCB$B_ERTMAX		00000071	D			
COM$FLUSHATTNS		00000033	RG	D	02	PCB$B_WEFC		0000002E	D		UCB$B_FEX		00000076	D			
COM$POST		00000063	RG	D	02	PCB$C_LENGTH		0000008C	D		UCB$B_FIPL		0000000B	D			
COM$SETATTNAST		00000098	RG	D	02	PCB$K_LENGTH		0000008C	D		UCB$B_LOCSRV		0000003C	D			
EXE$ABORTIO		********		X	02	PCB$L_ARB		00000084	D		UCB$B_OFFNDX		00000094	D			
EXE$ALLOCBUF		********		X	02	PCB$L_ASTQBL		00000014	D		UCB$B_OFFRTC		00000095	D			
EXE$DEANONPAGED		********		X	02	PCB$L_ASTQFL		00000010	D		UCB$B_REMSRV		0000003D	D			
EXE$FORK		********		X	02	PCB$L_EFC2P		00000058	D		UCB$B_SECTORS		0000003C	D			
EXE$MAXACMODE		********		X	02	PCB$L_EFC3P		0000005C	D		UCB$B_SLAVE		00000074	D			
IOC$GL_PSBL		********		X	02	PCB$L_EFCS		00000050	D		UCB$B_SPR		00000075	D			
IPL$_IOPOST	=	00000004		D		PCB$L_EFCU		00000054	D		UCB$B_STATE		00000052	D			
IPL$_QUEUEAST	=	00000006		D		PCB$L_EFWM		0000004C	D		UCB$B_TRACKS		0000003D	D			
IRP$B_CARCON		00000038	D		PCB$L_JIB		00000078	D		UCB$B_TT_CRFILL		0000009D	D				
IRP$B_EFN		00000022	D		PCB$L_OWNER		0000001C	D		UCB$B_TT_DECRF		000000A1	D				
IRP$B_PRI		00000023	D		PCB$L_PHD		00000064	D		UCB$B_TT_DELFF		000000A2	D				
IRP$B_RMOD		0000000B	D		PCB$L_PHYPCB		00000018	D		UCB$B_TT_DESPEE		000000A0	D				
IRP$B_TYPE		0000000A	D		PCB$L_PID		00000060	D		UCB$B_TT_DETYPE		000000A4	D				
IRP$C_LENGTH		0000005C	D		PCB$L_PQB		0000004C	D		UCB$B_TT_LFFILL		0000009E	D				
IRP$K_LENGTH		0000005C	D		PCB$L_SQBL		00000004	D		UCB$B_TT_SPEED		0000009C	D				
IRP$L_ARB		00000050	D		PCB$L_SQFL		00000000	D		UCB$B_TYPE		0000000A	D				
IRP$L_AST		00000010	D		PCB$L_STS		00000024	D		UCB$B_VERTSZ		0000003F	D				
IRP$L_ASTPRM		00000014	D		PCB$L_UIC		00000088	D		UCB$C_LENGTH		00000074	D				
IRP$L_DIAGBUF		00000044	D		PCB$L_WSSWP		00000020	D		UCB$C_MB_LENGTH		00000090	D				
IRP$L_EXTEND		0000004C	D		PCB$L_WTIME		00000028	D		UCB$C_TT_LENGTH		000000BC	D				
IRP$L_IOQBL		00000004	D		PCB$Q_PRIV		0000007C	D		UCB$K_LENGTH		00000074	D				
IRP$L_IOQFL		00000000	D		PCB$T_LNAME		00000068	D		UCB$K_MB_LENGTH		00000090	D				
IRP$L_IOSB		00000024	D		PCB$T_TERMINAL		00000044	D		UCB$K_TT_LENGTH		000000BC	D				
IRP$L_IOST1		00000034	D		PCB$W_APTCNT		00000030	D		UCB$L_AMB		00000054	D				
IRP$L_IOST2		00000038	D		PCB$W_ASTCNT		00000038	D		UCB$L_ASTQBL		00000010	D				
IRP$L_MEDIA		00000034	D		PCB$W_BIOCNT		0000003A	D		UCB$L_ASTQFL		0000000C	D				
IRP$L_PID		0000000C	D		PCB$W_BIOLM		0000003C	D		UCB$L_CPID		0000005C	D				
IRP$L_SEGVBN		00000040	D		PCB$W_DIOCNT		0000003E	D		UCB$L_CRB		00000020	D				
IRP$L_SEQNUM		00000048	D		PCB$W_DIOLM		00000040	D		UCB$L_DDB		00000024	D				
IRP$L_SVAPTE		0000002C	D		PCB$W_GPGCNT		00000034	D		UCB$L_DEVCHAR		00000034	D				
IRP$L_TT_TERM		00000038	D		PCB$W_GRP		0000008A	D		UCB$L_DEVDEPEND		0000003C	D				
IRP$L_UCB		0000001C	D		PCB$W_MEM		00000088	D		UCB$L_DPC		00000080	D				

UCB$L_DUETIM	0000005C	D	UCB$W_MB_SEED	00000000	D
UCB$L_DX_BFPNT	0000009C	D	UCB$W_MSGCNT	00000016	D
UCB$L_DX_BUF	00000098	D	UCB$W_MSGMAX	00000014	D
UCB$L_DX_RXDB	000000A0	D	UCB$W_NT_CHAN	0000007C	D
UCB$L_EMB	00000078	D	UCB$W_OFFSET	0000008A	D
UCB$L_FIRST	00000014	D	UCB$W_REFC	00000050	D
UCB$L_FPC	0000000C	D	UCB$W_SIZE	00000008	D
UCB$L_FQBL	00000004	D	UCB$W_SRCADDR	0000001A	D
UCB$L_FQFL	00000000	D	UCB$W_STS	00000058	D
UCB$L_FR3	00000010	D	UCB$W_TT_DESIZE	000000A5	D
UCB$L_FR4	00000014	D	UCB$W_UNIT	00000048	D
UCB$L_IOQBL	00000044	D	UCB$W_VPROT	0000001A	D
UCB$L_IOQFL	00000040	D			
UCB$L_IRP	0000004C	D			
UCB$L_LINK	0000002C	D			
UCB$L_LOGADR	00000064	D			
UCB$L_MAXBLOCK	00000084	D			
UCB$L_MB_MBX	0000007C	D			
UCB$L_MB_PORT	0000008C	D			
UCB$L_MB_RAST	00000078	D			
UCB$L_MB_SHB	00000080	D			
UCB$L_MB_WAST	00000074	D			
UCB$L_MB_WIOQBL	00000088	D			
UCB$L_MB_WIOQFL	00000084	D			
UCB$L_MEDIA	0000008C	D			
UCB$L_NT_DATSSB	00000074	D			
UCB$L_NT_INTSSB	00000078	D			
UCB$L_OPCNT	00000060	D			
UCB$L_OWNUIC	0000001C	D			
UCB$L_PID	00000028	D			
UCB$L_RQBL	00000004	D			
UCB$L_RQFL	00000000	D			
UCB$L_SVAPTE	00000068	D			
UCB$L_SVPN	00000064	D			
UCB$L_TT_DECHAR	000000A8	D			
UCB$L_TT_RDUE	0000008C	D			
UCB$L_TT_RTIMOU	000000B8	D			
UCB$L_VCB	00000030	D			
UCB$T_PARTNER	0000000C	D			
UCB$W_BCNT	0000006E	D			
UCB$W_BCR	00000096	D			
UCB$W_BOFF	0000006C	D			
UCB$W_BUFQUO	00000018	D			
UCB$W_BYTESTOGO	0000003E	D			
UCB$W_CHARGE	0000004A	D			
UCB$W_CYLINDERS	0000003E	D			
UCB$W_DA	0000008C	D			
UCB$W_DC	0000008E	D			
UCB$W_DEVBUFSIZ	0000003A	D			
UCB$W_DEVSTS	0000005A	D			
UCB$W_DIRSEQ	00000088	D			
UCB$W_DSTADDR	00000018	D			
UCB$W_DX_BCR	000000A4	D			
UCB$W_EC1	00000090	D			
UCB$W_EC2	00000092	D			
UCB$W_ERRCNT	00000072	D			
UCB$W_FUNC	0000007E	D			

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	000000F5 (245.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACB$B_RMOD	0000000B		#-109 (2) #-297 (6)
ACB$L_AST	00000010		#-108 (2)
ACB$L_KAST	00000018		#-108 (2) #-109 (2) #-110 (2) #-111 (2) #-145 (3) #-147 (3) #-292 (6) #-298 (6) #-299 (6) #-302 (6) #-303 (6) #-304 (6) #-305 (6)
ACB$L_PID	0000000C		#-110 (2)
ACB$M_QUOTA	=00000040		#-303 (6)
COM$DELATTNAST	00000000-R	98 (2)	
COM$DRVDEALMEM	00000073-R	214 (5)	#-151 (3)
COM$FLUSHATTNS	00000033-R	141 (3)	#-152 (3) #-286 (6)
COM$POST	00000063-R	183 (4)	
COM$SETATTNAST	00000098-R	283 (6)	
EXE$ABORTIO	00000000-XR		#-315 (6)
EXE$ALLOCBUF	00000000-XR		294 (6)
EXE$DEANONPAGED	00000000-XR		226 (5)
EXE$FORK	00000000-XR		104 (2) 221 (5)
EXE$MAXACMODE	00000000-XR		301 (6)
IOC$GL_PSBL	00000000-XR		185 (4)
IPL$_IOPOST	=00000004		#-187 (4)
IPL$_QUEUEAST	=00000006		#-218 (5) #-297 (6)
IRP$W_CHAN	00000028		#-284 (6)
P1	=00000000	70 (1)	#-285 (6)
P2	=00000004	71 (1)	#-299 (6)
P3	=00000008	72 (1)	300 (6)
P4	=0000000C	73 (1)	
P5	=00000010	74 (1)	
P6	=00000014	75 (1)	
PCB$L_PID	00000060		#-145 (3) #-305 (6)
PR$_IPL	=00000012		#-142 (3) #-150 (3) #-155 (3) #-309 (6) #-312 (6)
PR$_SIRR	=00000014		#-187 (4)
PRI$_IOCOM	=00000001		#-112 (2)
SCH$QAST	00000000-XR		113 (2)
SS$_NORMAL	00000000-XR		#-156 (3) #-313 (6)
UCB$B_DIPL	00000052		#-142 (3) #-309 (6)
UCB$L_OPCNT	00000060		#-184 (4)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	55 (1)	55 (1)
$CCBDEF	1	56 (1)	56 (1)
$DEFINI	1	55 (1)	55 (1) 56 (1) 57 (1) 58 (1) 59 (1) 60 (1) 61 (1) 62 (1) 63 (1) 64 (1) 65 (1) 66 (1)
$DYNDEF	2	57 (1)	57 (1)
$IPLDEF	1	58 (1)	58 (1)
$IRPDEF	4	59 (1)	59 (1)
$PCBDEF	4	60 (1)	60 (1)
$PRDEF	5	61 (1)	61 (1)
$PRIDEF	1	62 (1)	62 (1)
$PRVDEF	4	63 (1)	63 (1)
$PSLDEF	2	64 (1)	64 (1)
$RSNDEF	1	65 (1)	65 (1)
$UCBDEF	10	66 (1)	66 (1)
DSBINT	1	142 (3)	142 (3) 309 (6)
ENBINT	1	150 (3)	150 (3) 155 (3) 312 (6)
FORK	1	104 (2)	104 (2)
SOFTINT	1	187 (4)	187 (4)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
BEQL	0013	101	(2)	144	(3)	286	(6)								
BGTRU	001A	216	(5)												
BISB	0088	303	(6)												
BLBC	00E9	296	(6)												
BNEQ	0012	146	(3)	148	(3)	186	(4)								
BRB	0011	152	(3)	154	(3)										
BRW	0031	315	(6)												
BSBB	0010	151	(3)												
CLRL	00D4	111	(2)												
CMPL	00D1	145	(3)												
CMPW	00B1	147	(3)	215	(5)										
EXTZV	00EF	300	(6)												
INCL	00D6	184	(4)												
INSQUE	000E	185	(4)												
JMP	0017	113	(2)	226	(5)										
JSB	0016	104	(2)	221	(5)	294	(6)	301	(6)						
MFPR	00DB	142	(3)	309	(6)										
MOVB	0090	109	(2)	218	(5)	297	(6)								
MOVL	00D0	100	(2)	102	(2)	110	(2)	143	(3)	149	(3)	153	(3)	219	(5)
		225	(5)	285	(6)	298	(6)	299	(6)	305	(6)	310	(6)	311	(6)
MOVQ	007D	108	(2)												
MOVW	00B0	304	(6)												
MOVZBL	009A	112	(2)	156	(3)	292	(6)	302	(6)	313	(6)				
MOVZWL	003C	284	(6)												
MTPR	00DA	142	(3)	150	(3)	155	(3)	187	(4)	309	(6)	312	(6)		
POPL	8ED0	295	(6)												
POPR	00BA	114	(2)	227	(5)										
PUSHAB	009F	103	(2)	220	(5)										
PUSHL	00DD	293	(6)												
PUSHR	00BB	217	(5)	99	(2)										
RSB	0005	115	(2)	157	(3)	188	(4)	228	(5)	233	(5)	314	(6)		

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.END	317	(6)														
.ENDC	142	(3)	150	(3)	155	(3)	309	(6)	312	(6)						
.ENDM	104	(2)	142	(3)	150	(3)	187	(4)	55	(1)	56	(1)	57	(1)	58	(1)
	59	(1)	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)
.IDENT	5	(1)														
.IF	142	(3)	150	(3)	155	(3)	309	(6)	312	(6)						
.IFF	142	(3)	150	(3)	155	(3)	309	(6)	312	(6)						
.NOCROSS	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)	65	(1)	66	(1)								
.PSECT	77	(1)														
.RESTORE	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)	65	(1)	66	(1)								
.SAVE	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)	65	(1)	66	(1)								
.SBTTL	117	(3)	159	(4)	190	(5)	235	(6)	79	(2)						
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	3	#-285	(6)	#-299	(6)	300	(6)						
R0	14	#-143	(3)	#-145	(3)	#-147	(3)	#-149	(3)	#-153	(3)	#-156	(3)
		#-215	(5)	#-218	(5)	#-219	(5)	#-225	(5)	#-296	(6)	#-300	(6)
		#-302	(6)	#-313	(6)								
R1	1	#-292	(6)										
R2	10	#-112	(2)	#-297	(6)	#-298	(6)	#-299	(6)	#-302	(6)	#-303	(6)
		#-304	(6)	#-305	(6)	#-310	(6)	#-311	(6)				
R3	8	#-114	(2)	185	(4)	#-217	(5)	#-227	(5)	#-284	(6)	#-293	(6)
		#-295	(6)	#-99	(2)								
R4	6	#-114	(2)	#-145	(3)	#-217	(5)	#-227	(5)	#-305	(6)	#-99	(2)
R5	18	#-100	(2)	#-102	(2)	#-108	(2)	#-109	(2)	#-110	(2)	#-111	(2)
		#-114	(2)	#-142	(3)	#-184	(4)	#-217	(5)	#-219	(5)	#-225	(5)
		#-227	(5)	#-309	(6)	#-99	(2)						
R6	3	#-147	(3)	#-284	(6)	#-304	(6)						
R7	5	#-143	(3)	#-149	(3)	#-153	(3)	#-310	(6)	#-311	(6)		
R8	2	#-285	(6)	#-298	(6)								
SP	7	#-100	(2)	#-102	(2)	#-142	(3)	#-150	(3)	#-155	(3)	#-309	(6)
		#-312	(6)										

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.42
Command processing	894	00:00:00.26	00:00:01.18
Pass 1	708	00:00:02.39	00:00:05.29
Symbol table sort	0	00:00:00.18	00:00:00.19
Pass 2	41	00:00:00.62	00:00:01.45
Symbol table output	2	00:00:00.04	00:00:00.08
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	5	00:00:00.08	00:00:00.35
Assembler run totals	1676	00:00:03.60	00:00:08.97

The working set limit was 2400 pages.
130678 bytes (256 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 666 non-local and 9 local symbols.
317 source lines were read in Pass 1, producing 46 object records in Pass 2.
98 pages of virtual memory were used to define 25 macros.

```
                                    +----------------------------+
                                    ! Macro library statistics !
                                    +----------------------------+

Macro library name                   Macros defined
------------------                   --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5          7
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6            0
SYS$COMMON:[SYSLIB]LIB.MLB;2                       6
SYS$COMMON:[SYSLIB]STARLET.MLB;2                   8
TOTALS (all libraries)                            21

1010 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:COMDRVSUB.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

Table of contents


```
(3)      440      INI$LOAD_DEVICE         Make ptables for console and boot devices
(4)      494      DSP$CONFIG_LOAD Configure load device
(9)      989      Common routines
(10)    1118      configure RB730
(10)    1252      Attach UDA-50/LESI/KDB50/KFBTA/DEBNT load path        ;G:9
```

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CONFIG.MAR
0000     2 ; *14   16-DEC-1987 16:20:39 GEARIN "fixed problem with exclusion of zeroes in devic
0000     3 ; *13    4-DEC-1987 16:38:37 GEARIN "ADDED SUPPORT FOR 5 DIGIT UNIT #"
0000     4 ; *12   11-NOV-1987 08:36:20 GEARIN "ADDED SUPPORT FOR EXTENDED P-TABLE"
0000     5 ; *11   17-JUN-1987 16:08:40 BAGGETT "CHANGES COMPLETED"
0000     6 ; *10    8-MAY-1987 14:43:54 BAGGETT "PLACE .OBJ IN GOOD LIBRARY"
0000     7 ; *9     8-MAY-1987 13:05:53 BAGGETT "TK50 CODE ADDED"
0000     8 ; *8     5-MAY-1987 11:15:42 MI "REPLACED WITHOUT CHANGE SINCE SELF TEST ISNSHOW DAY
0000     9 ; *7    23-FEB-1987 11:03:34 BAGGETT "ALL VDS WILL BOOT WITH VMB->DIAGBOOT->VDS FROM
0000    10 ; *6    13-FEB-1987 15:50:14 SALEM "DONE - MARION NEEDS MODULE"
0000    11 ; *5     4-DEC-1986 16:32:28 SILVER "FIX FOR VOLUME SHADOWING"
0000    12 ; *4    30-MAY-1986 16:24:22 BAGGETT "SYMBOLS REMOVED"
0000    13 ; *3    27-MAY-1986 15:12:13 SALEM "ALL DONE"
0000    14 ; *2     1-APR-1986 09:43:51 BAGGETT "<no reason given>"
0000    15 ; *1     6-FEB-1986 15:15:16 DS ""
0000    16 ; DEC/CMS REPLACEMENT HISTORY, Element CONFIG.MAR
0000    17         .TITLE  CONFIG  *** CONFIG Configure load device
0000    18         .IDENT /10.10-45/                                              ;G:12
0000    19         .NoShow Conditionals
0000    20         .DSABL  GBL
0000    21
0000    22 ;
0000    23 ; Copyright (c) 1977, 1982, 1984, 1985, 1986, 1987                     ;G:7
0000    24 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    25 ;
0000    26 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    27 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    28 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    29 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    30 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    31 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    32 ; REMAIN IN DEC.
0000    33 ;
0000    34 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    35 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    36 ; CORPORATION.
0000    37 ;
0000    38 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    39 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    40
0000    41 ;++
0000    42 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    43 ;
0000    44 ; ABSTRACT:
0000    45 ;
0000    46 ; ENVIRONMENT:
0000    47 ;
0000    48 ; AUTHOR:       Roger Riggs     20-FEB-79       VERSION 01.
0000    49 ;
0000    50 ; MODIFICATIONS:
0000    51 ;
0000    52 ;                N. Howgate      26-Jun-79
0000    53 ;       01       References to I/O space are changed from
0000    54 ;                60000000 or 4000000 to IOC$K_IOSPACE
0000    55 ;                and are CPU dependant as defined in ONLYnnn.
0000    56 ;
0000    57 ;       02       John Ciukaj     1-25-80
```

```
0000     58 ;              Allow for RL01/RL02 to be configured: Add CON_UBA_RL
0000     59 ;
0000     60 ;       03     dave butenhof, 02-Sep-1981, version 6.5
0000     61 ;              Support RB730 (Nebula IDC) as load path.
0000     62 ;
0000     63 ;       04     - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     64 ;              Changed SEP Psect to CODE, DATA, and WORK.
0000     65 ;              Also added .LIBRARY statements for $DS,
0000     66 ;              $DIAG and SYS$LIBRARY:LIB.
0000     67 ;
0000     68 ;       05     - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     69 ;              Fixed truncation errors.
0000     70 ;
0000     71 ;       06     - Dave Butenhof, 24-Mar-1982, Version 6.6
0000     72 ;              Fix bug in CONFIG RB730 routine.  IDC can generate double
0000     73 ;              interrupt...make sure to disable device interrupts in handler
0000     74 ;              routine to avoid UNEXPECTED INTERRUPT error.
0000     75 ;       [07] Dave Butenhof, 15-Apr-1982, Version 6.7
0000     76 ;              Support UDA-50 load device.
0000     77 ;       [08] Dave Butenhof, 28-May-1982, version 6.8
0000     78 ;              Fix IDC code - use correct address for IDC, not the
0000     79 ;              one VMB gives us.
0000     80 ;
0000     81 ;       09     M. Baggett      24-Feb-1983     Version 6.11
0000     82 ;              Fixed document typo, support RC25.
0000     83 ;
0000     84 ;       10     M. Baggett      3-March-1983    Version 6.11
0000     85 ;              Added suport for HSC50.
0000     86 ;
0000     87 ;       11     M. Baggett      9-may-1983      Version 6.11
0000     88 ;              Booting from RP07 will have device name of DR and not DB.
0000     89 ;
0000     90 ;       12     M. Baggett      22-May-1983     Version 6.11
0000     91 ;              Allow any length name to be boot device (eg, CIAx, CIAxx, CIAxxx).
0000     92 ;
0000     93 ;       13     M. Baggett      21-Sep-1983     Version 6.13
0000     94 ;              Change HSC50 boot to a disk on the HSC50 to allow selections
0000     95 ;              among the different disks. Changed the name of LESI to RC11, and
0000     96 ;              temporarily changed the RC25 generic to DU.
0000     97 ;
0000     98 ;       14     M. Baggett      7-Sep-1983      Version 6.13
0000     99 ;              Change HSC50 controller naming to match standards.
0000    100 ;              Generic name is  HSC50$DUAnnn for DISK p-table.
0000    101 ;
0000    102 ;       15     Bob Bergazzi    2-14-84         Version 6.14
0000    103 ;              Moved code from KERNEL to here which builds ptables for the
0000    104 ;              console and boot devices. Subroutine is called INI$LOAD_DEVICE.
0000    105 ;              Done so that SET MEM can call this routine.
0000    106 ;
0000    107 ;       16     M. Baggett      15-Feb-1984     Version 6.14
0000    108 ;              Changed the default name for RC25/RCF25 device to DUA temporary ****
0000    109 ;
0000    110 ;       17     M. Baggett      16-May-1984     Version 7.0
0000    111 ;              Fixed HSC50 disk name to show proper unit number.
0000    112 ;
0000    113 ;       18     M. Baggett      6-Jun-1984      Version 7.0
0000    114 ;              Change device name for RC11/RC25/RCF25 back to DA.
```

```
0000   115 ;
0000   116 ;       19      Richard Brown   10-July-1984    Version 7.0
0000   117 ;               Change all references of "RC11" to "LESI".
0000   118 ;
0000   119 ;       20      Jim Shelhamer   1-Oct-1984      Version 7.1
0000   120 ;               Copied HSC50 code ([13] stuff) for multi-digit unit numbers to
0000   121 ;               RA disks section.
0000   122 ;
0000   123 ;       21      Jim Shelhamer   17-Oct-1984     Version 7.1
0000   124 ;               Added support (modelled after D Muse's changes in Attach) to handle
0000   125 ;               long device names for HSC devices.
0000   126 ;
0000   127 ;       22      Domenic Andella 25-Oct-1984     Version 7.1
0000   128 ;               Changed BLSS instruction in routine CON_CI_HSCCI
0000   129 ;               to unsigned BLSSU for comparison of drive # >= 10.
0000   130 ;
0000   131 ;       23      Jim Shelhamer   31-Oct-1984     Version 7.1
0000   132 ;               Changed BLSS instructio in routine CON_UBA_UDA
0000   133 ;               to unsigned BLSSU for comparison of drive # >= 10.
0000   134 ;
0000   135 ;       24      Bob Bergazzi    17-Dec-1984     Version 8.0
0000   136 ;               Fixed CI_NODE ptable to inherit controller letter from
0000   137 ;               CI780 ptable.
0000   138 ;
0000   139 ;       25      Bob Bergazzi    12-June-1985    Version 8.3
0000   140 ;               Add support for auto-config of BDA device ptables.
0000   141 ;
0000   142 ;       26      M. Baggett      12-Aug-1985     Version 8.3
0000   143 ;               Added support for Aurora booting on RDBX1.
0000   144 ;
0000   145 ;       27      Ted Salem       20-Aug-1985     Version 8.3
0000   146 ;               Added code in order to get the HSC port number being used
0000   147 ;
0000   148 ;       28      M. Baggett      3-Sep-1985      VDS     8.3
0000   149 ;               Made symbol AU$K_TYPE global.
0000   150 ;
0000   151 ;       29      Bob Bergazzi    4-Sep-1985      Version 8.3
0000   152 ;               Fix edit 26 (setup R0 with UD_INIT).
0000   153 ;
0000   154 ;       30      M. Baggett      4-Sep-1985      VDS 9.0
0000   155 ;               Set up registers for calling init routines.
0000   156 ;
0000   157 ;       31      Bob Bergazzi    24-Sep-1985     Version 9.0
0000   158 ;               Fixed auto config of UDA50 disk ptable, so that
0000   159 ;               the disk (e.g. DUA0) inherits the controller letter
0000   160 ;               from the DUA ptable. Symptom was reported for KDB50.
0000   161 ;
0000   162 ;       32      Jim Shelhamer   1-OCT-1985      Version 9.0
0000   163 ;               The values returned by the alloc for the CI ptable
0000   164 ;               were being clobbered by the routines to get the
0000   165 ;               pseudo-RPB. See edit 27.
0000   166 ;
0000   167 ;       33      M. Baggett      7-Oct-1985      Version 9.0
0000   168 ;               Changes for aurora booting AIO devices.
0000   169 ;
0000   170 ;       34      M. Baggett      14-Nov-1985     Version 9.2
0000   171 ;               Changes name RDBX1 to KFBTA.
```

```
0000    172 ;
0000    173 ;       35      Ted Salem       21-Nov-1985     Version 9.2                                     ;G:2
0000    174 ;               Moved code to configure the console device(s)                                   ;G:2
0000    175 ;               in the ONLYxxx modules.                                                         ;G:2
0000    176 ;                                                                                               ;G:2
0000    177 ;       36      M. Baggett      17-Dec-1985     Version 9.2                                     ;G:2
0000    178 ;               Correct value for KFBTA boot code and add                                       ;G:2
0000    179 ;               MAYA boot code.                                                                 ;G:2
0000    180 ;                                                                                               ;G:2
0000    181 ;       37      M. Baggett      11-Feb-1986     Version 9.2                                     ;G:2
0000    182 ;               Change for booting AIO on SCORPIO.                                              ;G:2
0000    183 ;                                                                                               ;G:3
0000    184 ;       38      T. Salem        6-May-1986      Version 10.0                                    ;G:3
0000    185 ;               Took out symbol for Aurora sid.  No longer needed.                              ;G:3
0000    186 ;                                                                                               ;G:5
0000    187 ;       39      M. Baggett      30-May-1986     Version 10.0                                    ;G:4
0000    188 ;               Remove symbols defined in library                                               ;G:4
0000    189 ;                                                                                               ;G:5
0000    190 ;       40      Doug Silver     1-Dec-1986      Version 10.3                                    ;G:5
0000    191 ;               Create proper device name for a shadow set.                                     ;G:5
0000    192 ;               Also fixed bug which determines proper unit                                     ;G:5
0000    193 ;               number.                                                                         ;G:5
0000    194 ;                                                                                               ;G:13
0000    195 ;       41      Ted Salem       29-Jan-1987     Version 10.5                                    ;G:6
0000    196 ;               Add code to configure an NI boot path.                                          ;G:6
0000    197 ;                                                                                               ;G:7
0000    198 ;       42      M. Baggett      13-Feb-1987     Version 10.5                                    ;G:7
0000    199 ;               Allow VDS to boot from console media with VMB/DIAGBOOT.                         ;G:13
0000    200 ;                                                                                               ;G:9
0000    201 ;       43      M. Baggett      24-Apr-1987     Version 10.8                                    ;G:9
0000    202 ;               Complete TK50 boot support.                                                     ;G:9
0000    203 ;                                                                                               ;G:11
0000    204 ;       44      M. Baggett      17-Jun-1987     Version 10.8                                    ;G:11
0000    205 ;               Fixed bug to allow booting from UDA50 and KDB50.                                ;G:11
0000    206 ;                                                                                               ;G:12
0000    207 ;               David Gearin    20-Oct-1987     Version 10.10                                   ;G:12
0000    208 ;               Add support for new extended p-table format.                                    ;G:12
0000    209 ;                                                                                               ;G:13
0000    210 ;               David Gearin    3-Dec-1987      Version 10.10                                   ;G:13
0000    211 ;               Allow for unit numbers of up to 5 digits.                                       ;G:13
0000    212 ;                                                                                               ;G:14
0000    213 ;               David Gearin    16-Dec-1987     Version 10.10                                   ;G:14
0000    214 ;               Fixed bug that excluded zeroes that are encountered                             ;G:14
0000    215 ;               in unit #.  These zeroes were left out of some                                  ;G:14
0000    216 ;               device names.                                                                   ;G:14
0000    217 ;--
```

```
                    0000   ;                                                                                    [04]
                    0000   ; Include Files                                                                       [04]
                    0000   ;                                                                                    [04]
                    0000           .LIBRARY        /SYS$LIBRARY:LIB/        ;                                  [04]
                    0000           .LIBRARY        /$DS/                    ;                                  [04]
                    0000           .LIBRARY        /$DIAG/                  ;                                  [04]
                    0000   ;                                                                                    [04]
                    0000   ; Macros                                                                             [04]
                    0000   ;                                                                                    [04]
                    0000
                    0000   ;                                                                                    [04]
                    0000   ; Equated Symbols                                                                    [04]
                    0000   ;                                                                                    [04]
                    0000
                    0000           CLIDEF
                    0000           $PRDEF
                    0000           .SAVE   LOCAL_BLOCK
                    0000           .PSECT  $ABS$,ABS
          00000000  044C           .=0
                    0000           .RESTORE
                    0000           $BTDDEF                                  ; VMS boot device codes            [07]
                    0000           .SAVE   LOCAL_BLOCK
                    0000           .PSECT  $ABS$,ABS
          00000000  044C           .=0
                    0000           .RESTORE
                    0000           $MSCPDef                                 ; Define disk protocol             [07]
                    0000           .SAVE   LOCAL_BLOCK
                    0000           .PSECT  $ABS$,ABS
          00000000  044C           .=0
                    0000           .RESTORE
                    0000
                    0000   ;
                    0000   ; Define Ptable data structures
                    0000   ;
                    0000           $DS_HPEODEF                                                                  ;G:12
                    0000           .SAVE   LOCAL_BLOCK
                    0000           .PSECT  $ABS$,ABS
          00000000  044C           .=0
                    0000           .IIF    NB,HPE$T_DEVICE,        HPE$T_DEVICE:
          00000014  0000           .IIF    NB,.BLKL,       .BLKL   5
                    0014           .IIF    NB,HPE$W_EXT_DRIVE,     HPE$W_EXT_DRIVE:
          00000016  0014           .IIF    NB,.BLKW,       .BLKW   1
                    0016           .IIF    NB,HPE$A_EPB,   HPE$A_EPB:
          0000001A  0016           .IIF    NB,.BLKL,       .BLKL   1
                    0000           .RESTORE
                    0000           $DS_HPODEF
                    0000           .SAVE   LOCAL_BLOCK
                    0000           .PSECT  $ABS$,ABS
          00000000  044C           .=0
                    0000           .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
          00000008  0000           .IIF    NB,.BLKQ,       .BLKQ   1
                    0008           .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
          0000000A  0008           .IIF    NB,.BLKW,       .BLKW   1
                    000A           .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
          0000000B  000A           .IIF    NB,.BLKB,       .BLKB   1
                    000B           .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
          0000000C  000B           .IIF    NB,.BLKB,       .BLKB   1
```

```
                    000C                .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
          00000018  000C                .IIF    NB,.BLKB,       .BLKB   12
                    0018                .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
          0000001C  0018                .IIF    NB,.BLKL,       .BLKL   1
                    001C                .IIF    NB,HP$A_DVA,    HP$A_DVA:
          00000020  001C                .IIF    NB,.BLKL,       .BLKL   1
                    0020                .IIF    NB,HP$A_LINK,   HP$A_LINK:
          00000024  0020                .IIF    NB,.BLKL,       .BLKL   1
                    0024                .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
          00000026  0024                .IIF    NB,.BLKW,       .BLKW   1
                    0026                .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
          00000032  0026                .IIF    NB,.BLKB,       .BLKB   12
                    0032                .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                    0000                .RESTORE
                    0000    243         $DS_RP04_DEF                                    ;G:13
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000032  044C                .=HP$A_DEPENDENT
                    0032                .IIF    NB,HP$K_RP04_LEN,       HP$K_RP04_LEN:
                    0032                .IIF    NB,RP04$K_LEN,  RP04$K_LEN:
                    0000                .RESTORE
                    0000    244         $DS_RK611_DEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000032  044C                .=HP$A_DEPENDENT
                    0032                .IIF    NB,HP$L_RK611_CSR,      HP$L_RK611_CSR:
                    0032                .IIF    NB,RK611$L_CSR, RK611$L_CSR:
          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                    0036                .IIF    NB,HP$B_RK611_BR,       HP$B_RK611_BR:
                    0036                .IIF    NB,RK611$B_BR,  RK611$B_BR:
          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                    0037                .IIF    NB,HP$K_RK611_LEN,      HP$K_RK611_LEN:
                    0037                .IIF    NB,RK611$K_LEN, RK611$K_LEN:
                    0000                .RESTORE
                    0000    245         $ds_rb730_def
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000032  044C                .=HP$A_DEPENDENT
                    0032                .IIF    NB,HP$L_RB730_CSR,      HP$L_RB730_CSR:
                    0032                .IIF    NB,RB730$L_CSR, RB730$L_CSR:
          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                    0036                .IIF    NB,HP$B_RB730_BR,       HP$B_RB730_BR:
                    0036                .IIF    NB,RB730$B_BR,  RB730$B_BR:
          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                    0037                .IIF    NB,HP$K_RB730_LEN,      HP$K_RB730_LEN:
                    0037                .IIF    NB,RB730$K_LEN, RB730$K_LEN:
                    0000                .RESTORE
                    0000    246         $ds_r80_def
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000032  044C                .=HP$A_DEPENDENT
                    0032                .IIF    NB,R80$K_LEN,   R80$K_LEN:
                    0000                .RESTORE
                    0000    247         $ds_r102_def
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000032  044C                .=HP$A_DEPENDENT
```

```
                     0032              .IIF    NB,HP$K_RL02_LEN,       HP$K_RL02_LEN:
                     0032              .IIF    NB,RL02$K_LEN,  RL02$K_LEN:
                     0000              .RESTORE
                     0000   248        $DS_RK06_DEF
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
            00000032 044C              .=HP$A_DEPENDENT
                     0032              .IIF    NB,HP$K_RK06_LEN,       HP$K_RK06_LEN:
                     0032              .IIF    NB,RK06$K_LEN,  RK06$K_LEN:
                     0000              .RESTORE
                     0000   249        $DS_RL11_DEF
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
            00000032 044C              .=HP$A_DEPENDENT
                     0032              .IIF    NB,HP$L_RL11_CSR,       HP$L_RL11_CSR:
                     0032              .IIF    NB,RL11$L_CSR,  RL11$L_CSR:
            00000036 0032              .IIF    NB,.BLKL,       .BLKL   1
                     0036              .IIF    NB,HP$B_RL11_BR,        HP$B_RL11_BR:
                     0036              .IIF    NB,RL11$B_BR,   RL11$B_BR:
            00000037 0036              .IIF    NB,.BLKB,       .BLKB   1
                     0037              .IIF    NB,HP$K_RL11_LEN,       HP$K_RL11_LEN:
                     0037              .IIF    NB,RL11$K_LEN,  RL11$K_LEN:
                     0000              .RESTORE
                     0000   250        $DS_RL01_DEF
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
            00000032 044C              .=HP$A_DEPENDENT
                     0032              .IIF    NB,HP$K_RL01_LEN,       HP$K_RL01_LEN:
                     0032              .IIF    NB,RL01$K_LEN,  RL01$K_LEN:
                     0000              .RESTORE
                     0000   251        $Ds_UDA50_Def                           ;                                       [07]
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
            00000032 044C              .=HP$A_DEPENDENT
                     0032              .IIF    NB,HP$L_UDA50_UDAIP,    HP$L_UDA50_UDAIP:
                     0032              .IIF    NB,UDA50$L_UDAIP,       UDA50$L_UDAIP:
            00000036 0032              .IIF    NB,.BLKL,       .BLKL   1
                     0036              .IIF    NB,HP$B_UDA50_BR,       HP$B_UDA50_BR:
                     0036              .IIF    NB,UDA50$B_BR,  UDA50$B_BR:
            00000037 0036              .IIF    NB,.BLKB,       .BLKB   1
                     0037              .IIF    NB,HP$B_UDA50_BURST,    HP$B_UDA50_BURST:
                     0037              .IIF    NB,UDA50$B_BURST,       UDA50$B_BURST:
            00000038 0037              .IIF    NB,.BLKB,       .BLKB   1
                     0038              .IIF    NB,HP$K_UDA50_LEN,      HP$K_UDA50_LEN:
                     0038              .IIF    NB,UDA50$K_LEN, UDA50$K_LEN:
                     0000              .RESTORE
                     0000   252        $Ds_RA80_Def                            ;                                       [07]
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
            00000032 044C              .=HP$A_DEPENDENT
                     0032              .IIF    NB,HP$K_RA80_LEN,       HP$K_RA80_LEN:
                     0032              .IIF    NB,RA80$K_LEN,  RA80$K_LEN:
                     0000              .RESTORE
                     0000   253        $ds_ra90_def                                                                    ;G:12
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
            00000032 044C              .=HP$A_DEPENDENT
```

```
                        0032                .IIF    NB,HP$K_RA90_LEN,       HP$K_RA90_LEN:
                        0032                .IIF    NB,RA90$K_LEN,  RA90$K_LEN:
                        0000                .RESTORE
                        0000    254         $DS_CI_DEF                                      ;                               [10]
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000032 044C               .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$B_CI_TR,  HP$B_CI_TR:
                        0032                .IIF    NB,CI$B_BI_ID,  CI$B_BI_ID:
                        0032                .IIF    NB,CI$B_TR,     CI$B_TR:
                00000033 0032               .IIF    NB,,BLKB        .BLKB   1
                        0033                .IIF    NB,HP$B_CI_BR,  HP$B_CI_BR:
                        0033                .IIF    NB,CI$B_BR,     CI$B_BR:
                00000034 0033               .IIF    NB,,BLKB        .BLKB   1
                        0034                .IIF    NB,HP$B_CI_NODE,        HP$B_CI_NODE:
                        0034                .IIF    NB,CI$B_NODE,   CI$B_NODE:
                00000035 0034               .IIF    NB,,BLKB        .BLKB   1
                        0035                .IIF    NB,HP$L_CI_FUNC,        HP$L_CI_FUNC:
                        0035                .IIF    NB,Ci$L_Func,   Ci$L_Func:
                00000039 0035               .IIF    NB,,BlkL        .BlkL   1
                        0039                .IIF    NB,HP$L_CI_MAINT_ID,    HP$L_CI_MAINT_ID:
                        0039                .IIF    NB,Ci$L_Maint_Id,       Ci$L_Maint_Id:
                0000003D 0039               .IIF    NB,,BlkL        .BlkL   1
                        003D                .IIF    NB,HP$L_CI_INDEX,       HP$L_CI_INDEX:
                        003D                .IIF    NB,Ci$L_Index,  Ci$L_Index:
                00000041 003D               .IIF    NB,,BlkL        .BlkL   1
                        0041                .IIF    NB,HP$K_CI_LEN, HP$K_CI_LEN:
                        0041                .IIF    NB,CI$K_LEN,    CI$K_LEN:
                        0000                .RESTORE
                        0000    255         $DS_DISK_DEF                                    ;                               [13]
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000032 044C               .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_DISK_LEN,       HP$K_DISK_LEN:
                        0032                .IIF    NB,DISK$K_LEN,  DISK$K_LEN:
                        0000                .RESTORE
                        0000    256
                        0000    257 ;
                        0000    258 ; Define miscellaneous structures/constants
                        0000    259 ;
                        0000    260
                        0000    261         $DS_DSADEF
                        0000    262         $DS_DSDEF
                        0000    263         $SSDef                                          ; Define SS status codes        [07]
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000    264
                        0000    265         $DS_CHCDEF
                        0000    266         $DS_CHIDEF
                        0000    267
                00000000 0000   268 RK_CS1=0                    ; Offset to control/status reg 1
                00000040 0000   269 RK_CS1_M_IE=1a6             ; Interrupt bit
                00000080 0000   270 RK_CS1_M_RDY=1a7            ; Controller ready
                00008000 0000   271 RK_CS1_M_CERR=1a15          ; Clear controller errors
                00000008 0000   272 RK_CS2=8                    ; Offset to control/status reg 2
```

```
0000000A  0000   273 RK_DS=10                    ; Offset to Drive status register
00000008  0000   274 RK_DS_V_DDT=8               ; Zero if RK06 drive, 1 if RK07
00000004  0000   275 F_DRVCLR=4                  ; Function to clear indicated drive
          0000   276
          0000   277
00000000  0000   278 RL_CS=0                     ; offset to control/status register
00000080  0000   279 RL_CS_M_CRDY=1a7            ; controller ready
00000040  0000   280 RL_CS_M_IE=1a6              ; interrupt bit
00000004  0000   281 RL_DA=4                     ; offset to disk address register
00000001  0000   282 RL_DA_M_MRK=1a0             ; bit 0 in disk address register
00000002  0000   283 RL_DA_M_STS=1a1             ; get status bit
00000008  0000   284 RL_DA_M_RST=1a3             ; reset bit in disk adress register
00000006  0000   285 RL_MP=6                     ; offset to multipurpose register
00000007  0000   286 RL_MP_V_TYP=7               ; 0=RL01,1=RL02
00000004  0000   287 F_STAT=4                    ; get status function
          0000   288
00000000  0000   289 RP_CS1=0                    ; Offset to control/status reg 1
00000000  0000   290 F_NOP=0                     ; Function for drive NOP
00000018  0000   291 RP_DT=24                    ; Offset to drive type register
00000000  0000   292 RP_DT_V_DTN=0               ; Low bit of drive type
00000009  0000   293 RP_DT_S_DTN=9               ; Size of drive type
00000010  0000   294 DT_RP04=^X10                ; RP04 drive type
00000011  0000   295 DT_RP05=^X11                ; RP05
00000012  0000   296 DT_RP06=^X12                ; RP06
00000021  0000   297 DT_RP07_HPT=^X21            ; RP07(with head per track option)
00000022  0000   298 DT_RP07=^X22                ; RP07(w/o head per track option)
00000015  0000   299 DT_RM03=^X15                ; RM03
00000016  0000   300 DT_RM80=^X16                ; RM80
00000017  0000   301 DT_RM05=^X17                ; RM05
00000014  0000   302 HP$C_NAMELEN    == 20       ;                                                    [21]
00000040  0000   303 CONSOLE_CODE    = 64        ; Boot code for console boot                         [42]
          0000   304
          0000   305 ;
          0000   306 ; RB730:/RB02/RB80 REGISTER OFFSETS FROM CSR ADDRESS
          0000   307 ;
          0000   308         $DEFINI RB                          ; START OF REGISTER DEFINITIONS
          0000                 .SAVE   LOCAL_BLOCK
          0000                 .PSECT  $ABS$,ABS
00000000  FE70                 .=0
          0000   309
          0000   310 $DEF      RB_CS               .BLKL   1      ;CONTROL STATUS REGISTER (CSR)
          0000                 .IIF    NB,RB_CS,           RB_CS:
00000004  0000                 .IIF    NB,,BLKL,           .BLKL   1
          0004   311           _VIELD  RB_CS,0,<-                 ;START OF CSR BIT DEFINITIONS
          0004   312                   <DRDY,,M>,-                ; DRIVE READY
          0004   313                   <FCODE,3>,-                ; FUNCTION CODE
          0004   314                   <,2>,-                     ; RESERVED BITS
          0004   315                   <IE,,M>,-                  ; INTERRUPT ENABLE
          0004   316                   <CRDY,,M>,-                ; CONTROLLER READY
          0004   317                   <DS,2>,-                   ; DRIVE SELECT
          0004   318                   <OPI,,M>,-                 ; OPERATION INCOMPLETE
          0004   319                   <CRC,,M>,-                 ; DATA CRC OR HEADER CRC OR DATA ECC
          0004   320                   <DLT,,M>,-                 ; DATA LATE OR HEADER NOT FOUND
          0004   321                   <NXM,,M>,-                 ; NON-EXISTENT MEMORY
          0004   322                   <DE,,M>,-                  ; DRIVE ERROR
          0004   323                   <CE,,M>,-                  ; COMPOSITE ERROR
          0004   324                   <ATN,4>,-                  ; DRIVE ATTENTION BITS
```

```
                 0004                    <ECC,2>,-                  ; ECC STATUS
                 0004                    <SSEI,,M>,-                ; SKIP SECTOR ERROR INHIBIT
                 0004                    <SSE,,M>,-                 ; SKIP SECTOR ERROR
                 0004                    <IR,,M>,-                  ; IDC INTERRUPT REQUEST
                 0004                    <MTN,,M>,-                 ; MAINTENANCE MODE
                 0004                    <TYP,,M>,-                 ; DRIVE TYPE 1=RB80, 0=RB02
                 0004                    <ASSI,,M>,-                ; AUTOMATIC SKIP SECTOR INHIBIT
                 0004                    <TOI,,M>,-                 ; TIME OUT INHIBIT (U-DIAG'S)
                 0004                    <FMT,,M>,-                 ; R80 FORMAT CONTROL
                 0004                    <,2>-                      ; RESERVED BITS
                 0004            >                                  ;END CSR BIT DEFINITIONS
                 0004
                 0004
                 0004    $DEF    RB_BA            .BLKL   1         ;BUS ADDRESS REGISTER (BAR)
                 0004            .IIF    NB,RB_BA,        RB_BA:
        00000008 0004            .IIF    NB,.BLKL,        .BLKL   1
                 0008
                 0008    $DEF    RB_BC            .BLKL   1         ;BYTE COUNT REGISTER (BCR)
                 0008            .IIF    NB,RB_BC,        RB_BC:
        0000000C 0008            .IIF    NB,.BLKL,        .BLKL   1
                 000C
                 000C    $DEF    RB_DA            .BLKL   1         ;DISK ADDRESS REGISTER (DAR)
                 000C            .IIF    NB,RB_DA,        RB_DA:
        00000010 000C            .IIF    NB,.BLKL,        .BLKL   1
                 0010            _VIELD  RB_DA,0,<-                 ;START OF DAR BIT DEFINITIONS
                 0010                    <SEC,8>,-                  ; SECTOR
                 0010                    <TRK,8>,-                  ; TRACK
                 0010                    <CYL,16>-                  ; CYLINDER
                 0010            >                                  ;END OF DAR BIT DEFINITIONS
                 0010
                 0010    $DEF    RB_MP            .BLKL   1         ;MULTIPURPOSE REGISTER (MPR)
                 0010            .IIF    NB,RB_MP,        RB_MP:
        00000014 0010            .IIF    NB,.BLKL,        .BLKL   1
                 0014            _VIELD  RB_MP,0,<-                 ;RB02 STATUS WORD DEFINITIONS
                 0014                    <STA,3>,-                  ; DRIVE STATE
                 0014                    <BH,,M>,-                  ; BRUSH HOME
                 0014                    <HO,,M>,-                  ; HEADS OUT
                 0014                    <CO,,M>,-                  ; COVER OPEN
                 0014                    <HS,,M>,-                  ; HEAD SELECT
                 0014                    <,1>,-                     ; RESERVED
                 0014                    <DSE,,M>,-                 ; DRIVE SELECT ERROR
                 0014                    <VC,,M>,-                  ; VOLUME CHECK
                 0014                    <WGE,,M>,-                 ; WRITE GATE ERROR
                 0014                    <SPE,,M>,-                 ; SPIN ERROR
                 0014                    <SKTO,,M>,-                ; SEEK TIME OUT
                 0014                    <WL,,M>,-                  ; WRITE LOCK
                 0014                    <CHE,,M>,-                 ; CURRENT HEAD ERROR
                 0014                    <WDE,,M>-                  ; WRITE DATA ERROR
                 0014            >                                  ;
                 0014            _VIELD  RB_MP,0,<-                 ;GET STATUS COMMAND DEFINITIONS
                 0014                    <MRK,,M>,-                 ; MARK (ALWAYS 1)
                 0014                    <STS,,M>,-                 ; GET STATUS
                 0014                    <,1>,-                     ; RESERVED
                 0014                    <RST,,M>,-                 ; RESET
                 0014            >
                 0014            _VIELD  RB_MP,0,<-                 ;RB80 STATUS WORD DEFINITIONS
                 0014                    <SEC,5>,-                  ; CURRENT RB80 SECTOR
```

```
                                  0014                <,3>,-                    ; RESERVED
                                  0014                <FLT,,M>,-                ; DRIVE FAULT
                                  0014                <PLGV,,M>,-               ; PLUG VALID
                                  0014                <SKE,,M>,-                ; SEEK ERROR
                                  0014                <ONCY,,M>,-               ; ON CYLINDER
                                  0014                <DRDY,,M>,-               ; DRIVE READY
                                  0014                <WTP,,M>,-                ; WRITE PROTECT
                                  0014                <,2>,-                    ; RESERVED
                                  0014          >                               ;END MPR BIT DEFINITIONS
                                  0014
                                  0014          $DEFEND RB                      ;END RB730:RB80/RB02 REGISTER DEFS
                                  0000          .RESTORE
                                  0000
                                  0000  ;
                                  0000  ; HARDWARE FUNCTION CODES
                                  0000  ;
                                  0000
                        00000004  0000  F_GETSTATUS=2*2                         ;GET STATUS
                                  0000
                                  0000  ;                                                                                [04]
                                  0000  ; External Symbols                                                               [04]
                                  0000  ;                                                                                [04]
                                  0000          .EXTRN  SCB_BASE, SCB_IMAGE, IOC$K_IOSPACE
                                  0000          .EXTRN  EXE$ALONONPAGED, EXE$DEANONPAGED, EXE$GB_CPUTYPE
                                  0000          .EXTRN  DS$WAITUS, DS_ERRSUP, INS$PTABLE, DSP$CONFIG_ADP, DS$CHANNEL
                                  0000          .EXTRN  DSV$SETLOAD
                                  0000          .EXTRN  SYS$UNWIND
                                  0000          .EXTRN  DS$GA_SELECTED, DS$GL_CLIBASE
                                  0000          .EXTRN  DSR$ONLY_CONSOLE_CONFIG         ; Routine in only module   [35]
                                  0000          .weak   Uda_Response, UD_Init           ; UDA-50 data block/init code   [07]
                                  0000          .WEAK   NI$GN_RPB_BUFFER                ; RPB buffer for the NI    [41] ;G:6
                                  0000          .WEAK   PA_INIT                         ; INIT SECTION FOR HSC50   [27]
                                  0000          .Extrn  Dsr$Allocate_Pseudo_RPB         ;                                [07]
                                  0000          .Extrn  Dsr$DeAllocate_Pseudo_RPB       ;                                [07]
                                  0000          .EXTRN  BOOT$L_ADP,BOOT$L_R2,BOOT$L_R1,BOOT$L_DEVTYP,BOOT$L_UNIT,BOOT$L_CSR
                                  0000          .EXTRN  HPE$C_EPB_SIZE                                                  ;G:12
                                  0000                                                                                  ;G:12
                                  0000                                                                                  ;G:13
                                  0000          .WEAK   KDB50_RESPONSE, BD_INIT         ; KDB50 data block/init code    [25]
                                  0000          .WEAK   PB$INIT,ET$INIT                 ; KFBTA and NIinit entry point  [26]
                                  0000                                                                                  ;G:12
                                  0000                                                                                  ;G:12
                              00000000          .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                                [04]
                                  0000          ModNam  CONFIG
          47 49 46 4E 4F 43 00'   0000  $MODULE:        .ASCIC  \CONFIG\
                              06  0000
                                  0007  T_TYPES:
       41 42 44 5F 34 30 50 52    0007          .ASCII  "RP04_DBA"      ; ^X10  RP04
       41 42 44 5F 35 30 50 52    000F          .ASCII  "RP05_DBA"      ; ^X11  RP05
       41 42 44 5F 36 30 50 52    0017          .ASCII  "RP06_DBA"      ; ^X12  RP06
            00000000 00000000     001F          .QUAD   0               ; ^X13
       41 52 44 5F 33 30 4D 52    0027          .ASCII  "RM03_DRA"      ; ^X14  RM03 drive
            00000000 00000000     002F          .QUAD   0
       41 52 44 5F 30 38 4D 52    0037          .ASCII  "RM80_DRA"      ; ^X16 RM80
       41 52 44 5F 35 30 4D 52    003F          .ASCII  "RM05_DRA"      ; ^X17 RM05
00000000'00000000'00000000'00000000'  0047          .LONG   0[18]           ; 9 Unused Quadwords
00000000'00000000'00000000'00000000'  0057
```

```
00000000'00000000'00000000'00000000' 0067
00000000'00000000'00000000'00000000' 0077
                00000000'00000000' 0087
          41 52 44 5F 37 30 50 52  008F   427          .ASCII  "RP07_DRA"      ; ^X21 RP07(with head per track option)
          41 52 44 5F 37 30 50 52  0097   428          .ASCII  "RP07_DRA"      ; ^X22 RP07(w/o head  per track option)
                                   009F   429
                         00000013  009F   430 K_TYPES=<.-T_TYPES>/8           ; number of entries
                                   009F   431 HSC50_BOOT::                                                    ;G:9
                               00  009F   432          .BYTE   0               ; Flag for BOOTING/SET MEMORY      [43]
                                   00A0   433                                                                 ;G:14
                                   00A0   434                                                                 ;G:14
                         00000000  00A0   435 .PSECT  WORK,NOSHR,NOEXE,WRT,LONG                              ;G:14
                                   0000   436 DEVICE_UNIT_FLAG:                                              ;G:14
                               00  0000   437          .BYTE   0               ; Flag for constructing unit number ;G:14
                                   0001   438                                                                 ;G:14
```

```
                     0001   440          .SBTTL  INI$LOAD_DEVICE          Make ptables for console and boot devices
                 00000000   441          .PSECT  Code, Exe, Shr, NoWrt, Long
                     0000   442 ;++                                                                      begin   [15]
                     0000   443 ; FUNCTIONAL DESCRIPTION:
                     0000   444 ;
                     0000   445 ;       This routine sets up the necessary P-tables
                     0000   446 ;       for the console load device and the boot device.
                     0000   447 ;
                     0000   448 ; CALLING SEQUENCE:
                     0000   449 ;
                     0000   450 ;       JSB     INI$LOAD_DEVICE
                     0000   451 ;
                     0000   452 ; INPUT PARAMETERS:
                     0000   453 ;
                     0000   454 ;       None
                     0000   455 ;
                     0000   456 ; IMPLICIT INPUTS:      NONE
                     0000   457 ;
                     0000   458 ; OUTPUT PARAMETERS:    NONE
                     0000   459 ;
                     0000   460 ; IMPLICIT OUTPUTS:
                     0000   461 ;
                     0000   462 ;       P-Tables for each device in the boot path and the
                     0000   463 ;       console load device.
                     0000   464 ;
                     0000   465 ; COMPLETION CODES:
                     0000   466 ;
                     0000   467 ;       None
                     0000   468 ;
                     0000   469 ; SIDE EFFECTS:
                     0000   470 ;
                     0000   471 ;       None
                     0000   472 ;
                     0000   473 ;--
                     0000   474
                     0000   475 INI$LOAD_DEVICE::
                     0000   476
    00000000'EF   16 0000   477          JSB     DSR$ONLY_CONSOLE_CONFIG         ; Config. Console device(s) [35]
                     0006   478
 50 00000000'EF   D0 0006   479          MOVL    L^BOOT$L_ADP,R0                 ; Get boot adapter address
                31 13 000D   480          BEQL    20$                             ; Branch if not there
00000000'EF   40 8F 91 000F   481          CMPB    #CONSOLE_CODE,BOOT$L_DEVTYP     ; Check if console boot via VMB/DIAG
                27 13 0017   482          BEQL    20$                             ; Skip.  Bus load not used.
    00000000'EF   DD 0019   483          pushl   l^boot$l_R1                     ; BI number                     [33]
    00000000'EF   DD 001F   484          PUSHL   L^BOOT$L_R2                     ; CI station number
    00000000'EF   DD 0025   485          PUSHL   L^BOOT$L_DEVTYP                 ; Push boot device type
    00000000'EF   DD 002B   486          PUSHL   L^BOOT$L_UNIT                   ; Push BOOT device unit number
    00000000'EF   DD 0031   487          PUSHL   L^BOOT$L_CSR                    ; Push BOOT device CSR address
                50 DD 0037   488          PUSHL   R0                              ; Push BOOT adapter address
 00000041'EF   06 FB 0039   489          CALLS   #6, L^DSP$CONFIG_LOAD           ; Auto-config load device
                     0040   490 20$:
                05   0040   491          RSB                                     ;                       end     [15]
```

```
                     0041   493          .List   MEB
                     0041   494          .SBTTL  DSP$CONFIG_LOAD Configure load device
                     0041   495 ;++
                     0041   496 ; FUNCTIONAL DESCRIPTION:
                     0041   497 ;
                     0041   498 ;       This routine determines sets up the necessary P-tables
                     0041   499 ;       for the load device given the adapter configuration register,
                     0041   500 ;       the device register and unit number.
                     0041   501 ;
                     0041   502 ; CALLING SEQUENCE:
                     0041   503 ;
                     0041   504 ;       DSP$CONFIG_LOAD(Adp adr, Dev adr, Unit, dev type, ci-port, bi num)
                     0041   505 ;
                     0041   506 ; INPUT PARAMETERS:
                     0041   507 ;
                     0041   508 ;       4(AP)   Address of configuration status register of adapter
                     0041   509 ;       8(AP)   Address of CSR for device.
                     0041   510 ;       12(AP)  Unit number of drive
                     0041   511 ;       16(AP)  device type: 0=massbus,1=RK06/07,2=RL01/02,32=HSC50          ;G:7
                     0041   512 ;       20(ap)  CI-HSC50 port station number
                     0041   513 ;       24(ap)  BI number
                     0041   514 ;
                     0041   515 ; IMPLICIT INPUTS:      NONE
                     0041   516 ;
                     0041   517 ; OUTPUT PARAMETERS:    NONE
                     0041   518 ;
                     0041   519 ; IMPLICIT OUTPUTS:
                     0041   520 ;
                     0041   521 ;       P-Tables for each device in the load control and data path
                     0041   522 ;
                     0041   523 ; COMPLETION CODES:
                     0041   524 ;
                     0041   525 ;       SS$_NORMAL
                     0041   526 ;
                     0041   527 ; SIDE EFFECTS:
                     0041   528 ;
                     0041   529 ;       The device and the adapter specified are pinged to determine
                     0041   530 ;       interrupt level and interrupt vectors.
                     0041   531 ;
                     0041   532 ; REGISTER USAGE:
                     0041   533 ;
                     0041   534 ;       R4      Address of P-table being created
                     0041   535 ;       R5      Address of channel or device
                     0041   536 ;--                                                                          ;G:13
                     0041   537          .NLIST  ME,MEB
                     0041   538          $OFFSET 0,NEGATIVE, < -
                     0041   539                  <L_STATUS,8>,-
                     0041   540                  <L_LINK,4>,-
                     0041   541                  <L_TYPE,4>,-
                     0041   542                  <L_LOCAL,0>>
                     0041                .SAVE   LOCAL_BLOCK
                     0041                .PSECT $ABS$ ABS
00000000  FE70                           .=0
FFFFFFF8  0000                           .BLKB   -8
          FFF8              L_STATUS:
FFFFFFF4  FFF8                           .BLKB   -4
          FFF4              L_LINK:
```

```
                FFFFFFF0  FFF4                .BLKB   -4
                          FFF0        L_TYPE:
                          FFF0        L_LOCAL:
                      00000041                .RESTORE
                          0041   543          .LIST   MEB
```

```
                        0A3C  0041   545 .ENTRY  DSP$CONFIG_LOAD,^M<R2,R3,R4,R5,R9,R11>  ;                                [07]
                              0043   546
          5E   F0 AD    9E    0043   547         MOVAB   L_LOCAL(FP),SP          ; Allocate space for local storage
               F0 AD    DF    0047   548         PUSHAL  L_TYPE(FP)              ; Address to store type
               F4 AD    DF    004A   549         PUSHAL  L_LINK(FP)              ; Address of P-table created
               04 AC    DD    004D   550         PUSHL   4(AP)                   ; Address of Adapter config reg
   00000000'EF    03    FB    0050   551         CALLS   #3,L^DSP$CONFIG_ADP     ; Configure the adaptor                [05]
               53 50    E9    0057   552         BLBC    R0,100$                 ; Return on error
          22   10 AC    D1    005A   553         CMPL    16(AP),#BTD$K_BVPSSP    ; Is it KFBTA ?                        [36]
                  1C    13    005E   554         Beql    50$                     ; Treat as MSCP device                 [36]
          23   10 AC    D1    0060   555         CMPL    16(AP),#BTD$K_AIE_TK50  ; Is it MAYA                           [36]
                  16    13    0064   556         Beql    50$                     ; Treat as MSCP device                 [36]
   00000062 8F  10 AC    D1    0066   557         CMPL    16(AP),#BTD$K_AIE_NI    ; Is it NI                             [41]
                  0C    13    006E   558         Beql    50$                     ; Treat as MSCP device                 [41]
          11   10 AC    D1    0070   559         CmpL    16(Ap), #Btd$K_UDA      ; Is it a UDA (code 17)                [07]
                  06    13    0074   560         BeqL    50$                     ; If not, do CASE for contiguous codes [25]
          21   10 AC    D1    0076   561         CMPL    16(AP),#BTD$K_BDA       ; Is it a KDB50 ?                      [25]
                  03    12    007A   562         BNEQ    60$                     ; Branch if not                        [25]
                03E4    31    007C   563 50$:    BrW     Con_Uba                 ; UDA50 sits on UNIBUS, BDA does not but[25]
                              007F   564                                         ; CON_UBA works for it.                [25]
          20   10 AC    D1    007F   565 60$:    CMPL    16(AP),#BTD$K_HSCCI     ; IS IT A HSC50 ?                      [10]
                  03    12    0083   566         BNEQ    70$                     ; BRANCH IF NOT.                       [10]
                0026    31    0085   567         BRW     CON_CI_HSCCI            ; BRANCH FOR HSC50                     [10]
                              0088   568 70$:    CASE    16(AP),TYPE=L,LIMIT=#0, -       ;                              [07]
                              0088   569                 displist=<con_mba,con_uba,con_uba,con_uba>
     03'  00   10 AC    CF    0088               CASEL   16(AP),#0,S^#<<30001$-30000$>/2>-1
                              008D       30000$:
                        0337' 008D               .SIGNED_WORD    con_mba-30000$
                        03D6' 008F               .SIGNED_WORD    con_uba-30000$
                        03D6' 0091               .SIGNED_WORD    con_uba-30000$
                        03D6' 0093               .SIGNED_WORD    con_uba-30000$
                              0095       30001$:
                              0095   570         ERRSUP_S                        ; Should never happen!
        00000000'EF    DF    0095                         PUSHAL  $MODULE
                  00    DD    009B                         PUSHL   #0
                  01    DD    009D                         PUSHL   #$ER
   00000000'EF    03    FB    009F                         CALLS   #3, DS_ERRSUP
   50   00660002 8F    D0    00A6   571         MOVL    #DS$_ERROR,R0           ; Flag error
                              00AD   572 100$:
                        04    00AD   573         RET
```

```
                                00AE   575 CON_CI_HSCCI:
                                00AE   576
                                00AE   577 ;
                                00AE   578 ; Now, build a pseudo-RPB to make the bootdriver init code                    [27]
                                00AE   579 ; happy, and then call said routine.  After it completes, the                [27]
                                00AE   580 ; global response buffer is accessible: it has the drive type, etc.          [27]
                                00AE   581 ;
                    0C AC   DD  00AE   582         PUSHL   12(AP)                  ; ci port number                     [27]
                       00   DD  00B1   583         PUSHL   #0                      ; No slave number                    [27]
                    0C AC   DD  00B3   584         PushL   12(Ap)                  ; Unit number                        [27]
                50  F4 BD   DE  00B6   585         MovAL   aL_Link(Fp), R0         ; Get address of link Ptable         [27]
                    18 A0   DD  00BA   586         PushL   Hp$A_Device(R0)         ; Address of link device             [27]
                       55   DD  00BD   587         PushL   R5                      ; Address of device CSR              [27]
                       7E   7C  00BF   588         ClrQ    -(Sp)                   ; Don't bother with adap./dev. codes [27]
          00000000'EF   07   FB  00C1   589         CallS   #7, Dsr$Allocate_Pseudo_RPB     ; Allocate the RPB           [27]
                       50   D5  00C8   590         TSTL    R0                      ; IF R0 is 0 there is an error       [27]
                       01   12  00CA   591         BNEQ    15$                     ; IF not 0 then continue             [27]
                            04  00CC   592         RET                             ; Else return                        [27]
                    59 50   D0  00CD   593 15$:     MovL    R0, R9                  ; Put RPB where PA_INIT wants it     [27]
               0000009F'EF   96  00D0   594         INCB    HSC50_BOOT              ; Set indicate driver called from here [43]
          00000000'EF   00   FB  00D6   595         CALLS   #0,PA_INIT              ; HSC50 inti routine                 [27]
               0000009F'EF   94  00DD   596         CLRB    HSC50_BOOT              ; Clear byte                         [43]
                    0A 50   E8  00E3   597         BlbS    R0, 17$                 ; BRANCH if SUCCESS                  [27]
                       69   9F  00E6   598         PushAB  (R9)                    ; Address of RPB                     [27]
          00000000'EF   01   FB  00E8   599         CallS   #1, Dsr$DeAllocate_Pseudo_RPB   ; Deallocate the RPB         [27]
                            04  00EF   600         RET                             ; ...                                [27]
                 14 AC   51   9A  00F0   601 17$:     MOVZBL  R1,20(AP)               ; REWRITE OVER OLD HSC PORT NUMBER   [27]
                       69   9F  00F4   602         PushAB  (R9)                    ; Address of RPB                     [27]
          00000000'EF   01   FB  00F6   603         CallS   #1, Dsr$DeAllocate_Pseudo_RPB   ; Deallocate the RPB         [27]
                                00FD   604 ;+                                                                  start [32]
                                00FD   605 ; BUILD PT FOR HSC50 DEVICE
                                00FD   606 ;-
               51   00000041 8F   D0  00FD   607         MOVL    #HP$K_CI_LEN, R1 ; LENGTH OF PT FOR CI_NODE.           [10]
                                0104   608                                                                        ;G:12
                                0104   609                                                                        ;G:12
                                0104   610                                                                        ;G:12
                         050B   30  0104   611         BSBW    ALLOCATE_PTABLE ; ALLOCATE THE SPACE FOR THIS DRIVE.   [10]
                      01 50   E8  0107   612         BLBS    R0,18$                  ; BRANCH IF SUCCESS                  [10]
                              04  010A   613         RET                             ; RETURN ON ERROR                    [10]
                                010B   614                                                                  ;    end [32]
                    08 A2   51   B0  010B   615 18$:     MOVW    R1,HP$W_SIZE(R2) ; SET SIZE OF STRUCTURE.              [10]
                    26 A2   07   90  010F   616         MOVB    #7,HP$T_TYPE(R2) ; SET LENGTH OF TYPE NAME             [10]
27 A2   2045444F 4E5F4943 8F   7D  0113   617         MOVQ    #^A"CI_NODE ",HP$T_TYPE+1(R2) ; SET DEVICE TYPE        [10]
                       62   05  D0  011F   618         MOVL    #5,HP$Q_DEVICE(R2) ; SET LENGTH OF NAME                [10]
                    50  0C A2   9E  0122   619         MOVAB   HP$T_DEVICE(R2),R0 ; ADDRESS OF DEVICE NAME            [10]
                    04 A2   60   9E  0126   620         MOVAB   (R0),HP$Q_DEVICE+4(R2) ; SET ADDRESS OF DEVICE NAME    [10]
               80   4149435F 8F   D0  012A   621         MOVL    #^A"_CIA",(R0)+ ; INSERT "_CIA".                       [10]
                    53   F4 AD   D0  0131   622         MOVL    L_LINK(FP),R3   ; Fetch address of link ptable         [24]
                 FF A0   0F A3   90  0135   623         MOVB    HP$T_DEVICE+3(R3),-1(R0) ; Copy controller letter      [24]
                    53   14 AC   D0  013A   624         MOVL    20(AP),R3       ; GET PORT NUMBER                      [11]
                                013E   625                                                                        ;G:13
               00000000'EF   00   90  013E   626         MOVB    #0,DEVICE_UNIT_FLAG     ; Clear flag                 ;G:14
                            54   D4  0145   627         CLRL    R4                      ; CLEAR HIGH ORDER FIELD     ;G:13
  54   53   53   00002710 8F   7B  0147   628         EDIV    #10000,R3,R3,R4 ; GET FIRST DIGIT                      ;G:13
                            0D   13  0150   629         BEQL    181$                    ; BRANCH IF NOT THIS LARGE   ;G:13
                 80   53   30   81  0152   630         ADDB3   #^A"0",R3,(R0)+ ; SET THIS ASCII DIGIT                 ;G:13
               00000000'EF   01   88  0156   631         BISB2   #1,DEVICE_UNIT_FLAG     ; Non-zero digit has been found ;G:14
```

```
                    62   D6   015D   632          INCL   HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                    ;G:13
          03E8 8F   54   B1   015F   633 181$:    CMPW   R4,#1000        ; DIGIT EQUAL TO ZERO?                        ;G:14
                    1C   1F   0164   634          BLSSU  182$            ; BRANCH IF YES                               ;G:14
                    55   D4   0166   635          CLRL   R5              ; CLEAR HIGH ORDER FIELD                      ;G:14
54  53  54  000003E8 8F   7B   0168   636          EDIV   #1000,R4,R3,R4  ; GET NEXT DIGIT                              ;G:13
                    1B   13   0171   637          BEQL   19$             ; BRANCH IF NOT THIS LARGE                    ;G:13
          80  53    30   81   0173   638          ADDB3  #^A"0",R3,(R0)+ ; SET THIS ASCII DIGIT                        ;G:13
                    62   D6   0177   639          INCL   HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                    ;G:13
      00000000'EF   01   88   0179   640          BISB2  #1,DEVICE_UNIT_FLAG     ; Non-zero digit has been found       ;G:14
                    0C   11   0180   641          BRB    19$             ; CONTINUE                                    ;G:14
        05 00000000'EF   E9   0182   642 182$:    BLBC   DEVICE_UNIT_FLAG,19$    ; Non zero digit hasn't been found    ;G:14
                80  30   90   0189   643          MOVB   #^A"0",(R0)+    ; SET THIS ASCII DIGIT                        ;G:14
                    62   D6   018C   644          INCL   HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                    ;G:14
             64 8F  54   91   018E   645 19$:     CMPB   R4,#100         ; DIGIT EQUAL TO ZERO?                        ;G:14
                    1C   1F   0192   646          BLSSU  195$            ; BRANCH IF YES                               ;G:14
                    55   D4   0194   647          CLRL   R5              ; CLEAR HIGH ORDER FIELD                        [11]
54  53  54  00000064 8F   7B   0196   648          EDIV   #100,R4,R3,R4   ; GET NEXT DIGIT                                [10]
                    1B   13   019F   649          BEQL   20$             ; BRANCH IF NOT THIS LARGE                      [10]
          80  53    30   81   01A1   650          ADDB3  #^A"0",R3,(R0)+ ; SET THIS ASCII DIGIT                          [10]
                    62   D6   01A5   651          INCL   HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                      [10]
      00000000'EF   01   88   01A7   652          BISB2  #1,DEVICE_UNIT_FLAG     ; Non-zero digit has been found       ;G:14
                    0C   11   01AE   653          BRB    20$             ; CONTINUE                                    ;G:14
        05 00000000'EF   E9   01B0   654 195$:    BLBC   DEVICE_UNIT_FLAG,20$    ; Non zero digit hasn't been found    ;G:14
                80  30   90   01B7   655          MOVB   #^A"0",(R0)+    ; SET THIS ASCII DIGIT                        ;G:14
                    62   D6   01BA   656          INCL   HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                    ;G:14
                0A  54   91   01BC   657 20$:     CMPB   R4,#10          ; DIGIT EQUAL TO ZERO?                        ;G:14
                    18   1F   01BF   658          BLSSU  25$             ; BRANCH IF YES                               ;G:14
                    55   D4   01C1   659          CLRL   R5              ; CLEAR HIGH ORDER FIELD                        [11]
    54  53  54     0A   7B   01C3   660          EDIV   #10,R4,R3,R4    ; GET NEXT DIGIT                                [10]
                    1B   13   01C8   661          BEQL   30$             ; BRANCH IF NOT THIS LARGE                      [10]
          80  53    30   81   01CA   662          ADDB3  #^A"0",R3,(R0)+ ; SET THIS ASCII DIGIT                          [10]
      00000000'EF   01   88   01CE   663          BISB2  #1,DEVICE_UNIT_FLAG     ; Non-zero digit has been found       ;G:14
                    62   D6   01D5   664          INCL   HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                      [10]
                    0C   11   01D7   665          BRB    30$             ; CONTINUE                                    ;G:14
        05 00000000'EF   E9   01D9   666 25$:     BLBC   DEVICE_UNIT_FLAG,30$    ; Non zero digit hasn't been found    ;G:14
                80  30   90   01E0   667          MOVB   #^A"0",(R0)+    ; SET THIS ASCII DIGIT                        ;G:14
                    62   D6   01E3   668          INCL   HP$Q_DEVICE(R2) ; INCREASE LENGTH OF STRING                    ;G:14
          60  54    30   81   01E5   669 30$:     ADDB3  #^A"0",R4,(R0)  ; SET LAST DIGIT                                [10]
        14 AB   14  AC   B0   01E9   670          MOVW   20(AP),HPE$W_EXT_DRIVE(R11)     ;port # in EPB                ;G:12
  000000FF'EF   14  AC   B1   01EE   671          CMPW   20(AP),^Xff                                                   ;G:12
                    05   14   01F6   672          BGTR   35$                                                           ;G:12
        0B A2   14  AC   90   01F8   673          MOVB   20(AP),HP$B_DRIVE(R2)           ; SET port NUMBER
18 A2   04 AC   00000000'8F   C9   01FD   674 35$:     BISL3  #<IOC$K_IOSPACE>,4(AP),HP$A_DEVICE(R2) ; SET DEVICE ADDR  [10]
        1C A2   18  A2   D0   0207   675          MOVL   HP$A_DEVICE(R2),HP$A_DVA(R2)    ; Set direct address          [13]
        20 A2   F4  AD   D0   020C   676          MOVL   L_LINK(FP),HP$A_LINK(R2)        ; SET LINK ADDRESS.
           3D A2    04   9A   0211   677          MOVZBL #^X4,HP$L_CI_INDEX(R2)          ; SET INDEX USED BY $DS_CASE
     39 A2   80000004 8F   D0   0215   678          MOVL   #^X80000004,HP$L_CI_MAINT_ID(R2); SET HSC50 IDENTIFICATION   [10]
     35 A2   4F710200 8F   D0   021D   679          MOVL   #^X4F710200,HP$L_CI_FUNC(R2)    ; FUNCTIONALITY OF HSC50
        34 A2   14  AC   90   0225   680          MOVB   20(AP), HP$B_CI_NODE(R2)        ; SET NODE NUMBER.
                    52   DD   022A   681          PUSHL  R2                              ; Save address of CI_NODE p-table
          0000'CF   62   FA   022C   682          CALLG  (R2), W^INS$PTABLE              ; Insert it in list
                 03 50   E8   0231   683          BLBS   R0,100$                         ; Continue if no errors.
                   021E  31   0234   684          BRW    CON_RELEASE_X                   ; Go release buffer
    51   00000046 8F   D0   0237   685 100$:     MOVL   #<HP$K_DISK_LEN+HP$C_NAMELEN>,R1; Length of ptable for DISK   [13]
                         023E   686                                                                                    ;G:12
                         023E   687                                                                                    ;G:12
                         023E   688                                                                                    ;G:12
```

```
                    03D1   30   023E   689          BSBW    ALLOCATE_PTABLE          ; Allocate the space for this drive     [13]
                  01 50    E8   0241   690          BLBS    R0,110$                  ; Branch if success                     [13]
                           04   0244   691          RET                              ; Return on error.                      [13]
              08 A2   51    B0   0245   692 110$:    MOVW    R1,HP$W_SIZE(R2)         ; Set size of structure                 [13]
              26 A2   04    90   0249   693          MOVB    #4,HP$T_TYPE(R2)         ; Set length of type name               [13]
        27 A2   4B534944 8F  D0   024D   694          MOVL    #^A"DISK",HP$T_TYPE+1(R2) ; Set device type                      [13]
                    62   0B   D0   0255   695          MOVL    #11,HP$Q_DEVICE(R2)      ; Set  length of name                  [13]
              0A A2   04    88   0258   696          BISB2   #HP$M_NAME,HP$B_FLAGS(R2)       ; set long name flag              [21]
              50   08 A2   3C   025C   697          MOVZWL  HP$W_SIZE(R2),R0         ; Get size of ptable                    [21]
           50   50   14    C3   0260   698          SUBL3   #HP$C_NAMELEN,R0,R0      ; Determine offset                      [21]
           50   52   50    C1   0264   699          ADDL3   R0,R2,R0                 ; Address to store long device name     [21]
             04 A2   60    9E   0268   700          MOVAB   (R0),HP$Q_DEVICE+4(R2)   ; Set address of device name              [13]
        80   4353485F 8F    D0   026C   701          MOVL    #^A"_HSC",(R0)+          ; Insert "_HSC".                        [14]
           80   3035 8F    B0   0273   702          MOVW    #^A"50",(R0)+            ; Insert "50".                          [14]
             56   0C A2   DE   0278   703          MOVAL   HP$T_DEVICE(R2),R6       ; P-table entry for device name         [40]
                  0C AC    B5   027C   704          TSTW    12(AP)                   ; Shadow set?                           [40]
                           19   18   027F   705          BGEQ    115$                     ; No                                    [40]
        80   53554424 8F    D0   0281   706          MOVL    #^A"$DUS",(R0)+          ; Insert "$DUS".                        [40]
        86   5355445F 8F    D0   0288   707          MOVL    #^A"_DUS",(R6)+          ; Put "_DUS" in usual dev. name area    [40]
                           53   D4   028F   708          CLRL    R3                       ; Get rid of any junk                   [40]
     53   0C AC   8000 8F    AB   0291   709          BICW3   #^X8000,12(AP),R3        ; Clear shadow set bit.  R3 = drive #   [40]
                           19   11   0298   710          BRB     1155$                    ; Don't do non-shadow set stuff         [40]
                                029A   711          ; Not a shadow set                                                       ;G:5
              53   0C AC   D0   029A   712 115$:    MOVL    12(AP),R3                ; Get drive #                           [17]
        80   41554424 8F    D0   029E   713          MOVL    #^A"$DUA",(R0)+          ; Insert "$DUA".                        [13]
        86   4155445F 8F    D0   02A5   714          MOVL    #^A"_DUA",(R6)+          ; Put "_DUA" in usual dev. name area    [21]
        00000000'EF   00    90   02AC   715          MOVB    #0,DEVICE_UNIT_FLAG                                              ;G:14
                                02B3   716                                                                                    ;G:14
                                02B3   717          ; Get unit number for a shadowed or non-shadowed set.                    ;G:14
                                02B3   718                                                                                    ;G:14
                           54   D4   02B3   719 1155$:    CLRL    R4                       ; Clear high order field                ;G:14
  54   53   53   00002710 8F    7B   02B5   720          EDIV    #10000,R3,R3,R4          ; Set first Digit                       ;G:13
                           11   13   02BE   721          BEQL    116$                     ; Branch if number less than 5 digits   ;G:13
              80   53   30    81   02C0   722          ADDB3   #^A"0",R3,(R0)+          ; Set Ascii digit                       ;G:13
              86   53   30    81   02C4   723          ADDB3   #^A"0",R3,(R6)+          ; Put digit in old area                 ;G:13
                           62   D6   02C8   724          INCL    HP$Q_DEVICE(R2)          ; Increase length of string             ;G:13
        00000000'EF   01    88   02CA   725          BISB2   #1,DEVICE_UNIT_FLAG      ; Non-zero digit has been found         ;G:14
           03E8 8F    54   B1   02D1   726 116$:    CMPW    R4,#1000                 ; Digit equal to zero?                  ;G:14
                           20   1F   02D6   727          BLSSU   1165$                    ; Branch if yes                         ;G:14
                           55   D4   02D8   728          CLRL    R5                       ; Clear high order field                ;G:14
  54   53   54   000003E8 8F    7B   02DA   729          EDIV    #1000,R4,R3,R4           ; get next digit                        ;G:13
                           22   13   02E3   730          BEQL    117$                     ; Branch if number less than 4 digits   ;G:13
              80   53   30    81   02E5   731          ADDB3   #^A"0",R3,(R0)+          ; Set Ascii digit                       ;G:13
              86   53   30    81   02E9   732          ADDB3   #^A"0",R3,(R6)+          ; Put digit in old area                 ;G:13
                           62   D6   02ED   733          INCL    HP$Q_DEVICE(R2)          ; Increase length of string             ;G:13
        00000000'EF   01    88   02EF   734          BISB2   #1,DEVICE_UNIT_FLAG      ; Non-zero digit has been found         ;G:14
                           0F   11   02F6   735          BRB     117$                     ; Continue                              ;G:14
           08 00000000'EF   E9   02F8   736 1165$:    BLBC    DEVICE_UNIT_FLAG,117$    ; Non zero digit hasn't been found      ;G:14
                    80   30    90   02FF   737          MOVB    #^A"0",(R0)+             ; Set Ascii digit                       ;G:14
                    86   30    90   0302   738          MOVB    #^A"0",(R6)+             ; Put digit in old area                 ;G:14
                           62   D6   0305   739          INCL    HP$Q_DEVICE(R2)          ; Increase length of string             ;G:14
                 64 8F    54   91   0307   740 117$:    CMPB    R4,#100                  ; Digit equal to zero?                  ;G:14
                           20   1F   030B   741          BLSSU   118$                     ; Branch if yes                         ;G:14
                           55   D4   030D   742          CLRL    R5                       ; Clear high order field                [13]
  54   53   54   00000064 8F    7B   030F   743          EDIV    #100,R4,R3,R4            ; Set first digit                       [13]
                           22   13   0318   744          BEQL    120$                     ; Branch if number is <100              [13]
              80   53   30    81   031A   745          ADDB3   #^A"0",R3,(R0)+          ; Set ascii digit                       [13]
```

```
          86   53   30   81  031E          ADDB3   #^A"0",R3,(R6)+          ; Put digit in old area                [21]
   00000000'EF       01   88  0322          BISB2   #1,DEVICE_UNIT_FLAG      ; Non-zero digit has been found       ;G:14
                     62   D6  0329          INCL    HP$Q_DEVICE(R2)          ; Increase length of string            [13]
                     0F   11  032B          BRB     120$                     ; Make sure you get 2nd digit          [40]
     08 00000000'EF       E9  032D  118$:   BLBC    DEVICE_UNIT_FLAG,120$    ; Non zero digit hasn't been found    ;G:14
             80   30   90  0334          MOVB    #^A"0",(R0)+             ; Set ascii digit                     ;G:14
             86   30   90  0337          MOVB    #^A"0",(R6)+             ; Put digit in old area               ;G:14
                     62   D6  033A          INCL    HP$Q_DEVICE(R2)          ; Increase length of string           ;G:14
             0A   54   91  033C  120$:   CMPB    R4,#10                   ; Is it >=10    ?                      [17]
                     1A   1F  033F          BLSSU   130$                     ; Branch if not.                       [22]
                     55   D4  0341          CLRL    R5                       ; Clear high order field               [13]
   54   53   54   0A   7B  0343          EDIV    #10,R4,R3,R4             ; Get next digit                       [13]
          80   53   30   81  0348          ADDB3   #^A"0",R3,(R0)+          ; Set this ascii digit                 [13]
          86   53   30   81  034C          ADDB3   #^A"0",R3,(R6)+          ; Put 2nd digit in old area            [21]
                     62   D6  0350          INCL    HP$Q_DEVICE(R2)          ; Increase length of string            [13]
   00000000'EF       01   88  0352          BISB2   #1,DEVICE_UNIT_FLAG      ; Non-zero digit has been found       ;G:14
                     0F   11  0359          BRB     133$                     ; Continue                            ;G:14
     08 00000000'EF       E9  035B  130$:   BLBC    DEVICE_UNIT_FLAG,133$    ; Non zero digit hasn't been found    ;G:14
             80   30   90  0362          MOVB    #^A"0",(R0)+             ; Set last digit                       [13]
             86   30   90  0365          MOVB    #^A"0",(R6)+             ; Put 3rd digit in old area            [21]
                     62   D6  0368          INCL    HP$Q_DEVICE(R2)          ; Increase length of string           ;G:14
                              036A                                                                                ;G:14
          60   54   30   81  036A  133$:   ADDB3   #^A"0",R4,(R0)           ; Set this ascii digit                 [13]
          66   54   30   81  036E          ADDB3   #^A"0",R4,(R6)           ; Put 2nd digit in old area            [21]
                              0372                                                                                ;G:14
       14 AB    0C AC   B0  0372          MOVW    12(AP),HPE$W_EXT_DRIVE(R11)  ; Drive # in EPB                  ;G:14
 000000FF'EF    0C AC   B1  0377          CMPW    12(AP),^XFF                                                    ;G:12
                     05   14  037F          BGTR    135$                                                           ;G:12
       0B A2    0C AC   90  0381          MOVB    12(AP),HP$B_DRIVE(R2)    ; Set drive number                     [13]
                1C A2   D4  0386  135$:   CLRL    HP$A_DVA(R2)             ; Clear direct address                 [13]
          20 A2    8E   D0  0389          MOVL    (SP)+,HP$A_LINK(R2)      ; Set link address.                    [13]
        0000'CF    62   FA  038D          callg   (r2), w^ins$ptable       ; Insert it in list                    [21]
                              0392          blbcw   r0, CON_RELEASE_X        ; fail if error                        [21]
                03 50   E8  0392          BLBS    r0,.+4
                 00BD   31  0395          BRW     CON_RELEASE_X
             50   52   D0  0398          MOVL    R2,R0                    ; Copy address of P-table of load dev  [21]
   52   00000000'EF   DE  039B          MOVAL   DS$GL_CLIBASE,R2         ; Set R2 to base of CLI data area      [21]
          08 A2    60   D0  03A2          MOVL    HP$Q_DEVICE(R0), -
                              03A6                  CLI$Q_FILE(R2)           ; Set length of load device name       [21]
             7E   3A   90  03A6          MOVB    #^A":",-(SP)             ; Tack a ":" to the end                [21]
             53   60   D0  03A9          MOVL    HP$Q_DEVICE(R0),R3       ; Length of name. Any size.            [12]
                     53   D7  03AC          DECL    R3                       ; "_" not included                     [12]
          50    04 A0   D0  03AE          MOVL    HP$Q_DEVICE+4(R0),R0     ; Get address of long device name      [21]
           7E    6043   90  03B2  140$:   MOVB    (R0)[R3],-(SP)           ; Place name on stack                  [12]
                F9 53   F5  03B6          SOBGTR  R3,140$                  ; Loop to move all of name             [12]
          0C A2    6E   9E  03B9          MOVAB   (SP),CLI$Q_FILE+4(R2)    ; Set address of load device name      [21]
                 FC40'  30  03BD          BSBW    DSV$SETLOAD              ; Set default load device              [21]
             50   01   D0  03C0          MOVL    #SS$_NORMAL,R0           ; Success                              [21]
                     04  03C3          RET                              ;                                      [21]
                              03C4
```

```
                                03C4    795 CON_MBA:
                                03C4    796 ;+
                                03C4    797 ; BUILD PT FOR MASSBUS DRIVE
                                03C4    798 ;-
    55   04 AC   00000400'8F    C9  03C4    799         BISL3   #<IOC$K_IOSPACE>+<^X400>,4(AP),R5       ; Base adapter
         55   03   07   0C AC   F0  03CD    800         INSV    12(AP),#7,#3,R5         ; Insert unit number to make drive csr
                   65   01      D0  03D3    801         MOVL    #F_NOP!1,RP_CS1(R5)     ; Issue NOP to drive
                  50   18 A5    D0  03D6    802         MOVL    RP_DT(R5),R0            ; Get drive type register
                 50   09   00    EF  03DA    803         EXTZV   #RP_DT_V_DTN,#RP_DT_S_DTN, -
                           7E        03DE    804                 R0,-(SP)                ; Isolate drive type
                      51   32    D0  03DF    805         MOVL    #RP04$K_LEN,R1          ; Length of PT for RP04,RP05,RP06,RM03
                                03E2    806                                                                             ;G:12
                                03E2    807                                                                             ;G:12
                                03E2    808                                                                             ;G:12
                                03E2    809                                                                             ;G:12
                         022D   30  03E2    810         BSBW    Allocate_Ptable         ; Allocate the space for this device    [07]
                           53 8ED0  03E5    811         POPL    R3                      ; Get drive type back
                        01 50   E8  03E8    812         BLBS    R0,10$                  ; Branch if success
                               04   03EB    813         RET                             ; Return on error
                                03EC    814 10$:
                    08 A2   51    B0  03EC    815         MOVW    R1,HP$W_SIZE(R2)        ; Set size of structure
                                03F0    816
                       53   10    C2  03F0    817         SUBL    #DT_RP04,R3             ; Make zero correspond to RP04 drivetype
                            0F    19  03F3    818         BLSS    30$                     ; Branch if less than that
                       53   13    D1  03F5    819         CMPL    #K_TYPES,R3             ; Range check drive type
                            0A    15  03F8    820         BLEQ    30$                     ; Branch if unfamilar drive type
          50   00000007'EF43    7D  03FA    821         MOVQ    L^T_TYPES[R3],R0        ; Get device type and name              [05]
                            13    12  0402    822         BNEQ    40$                     ; Branch if valid
                                0404    823 30$:    ERRSUP_S                            ; Unknown drive type
            00000000'EF         DF  0404                        PUSHAL  $MODULE
                            00  DD  040A                        PUSHL   #0
                            02  DD  040C                        PUSHL   #$ER
       00000000'EF          03  FB  040E                        CALLS   #3, DS_ERRSUP
                            3E  11  0415    824         BRB     CON_RELEASE_X           ; Release buffer and exit
                                0417    825
            26 A2       04  90  0417    826 40$:    MOVB    #4,HP$T_TYPE(R2)        ; Set length of type name
            27 A2       50  D0  041B    827         MOVL    R0,HP$T_TYPE+1(R2)      ; Set drive type
                                041F    828
                62      05  D0  041F    829         MOVL    #5,HP$Q_DEVICE(R2)      ; Set length of name
               50   0C A2  9E  0422    830         MOVAB   HP$T_DEVICE(R2),R0      ; Address of device name
               04 A2    60  9E  0426    831         MOVAB   (R0),HP$Q_DEVICE+4(R2)  ; Set address of device name
                80      51  D0  042A    832         MOVL    R1,(R0)+                ; Insert "_xxx" xxx is DBA or DRA
         80    0C AC     30  81  042D    833         ADDB3   #^A"0",12(AP),(R0)+     ; Set unit number
           14 AB    0C AC   B0  0432    834         MOVW    12(AP),HPE$W_EXT_DRIVE(R11) ; Put the drive # in EPB           ;G:12
    000000FF'EF     0C AC   B1  0437    835         CMPW    12(AP),^XFF                                                     ;G:12
                        05  14  043F    836         BGTR    50$                                                             ;G:12
          0B A2     0C AC   90  0441    837         MOVB    12(AP),HP$B_DRIVE(R2)   ; Set drive number
             18 A2     55   D0  0446    838 50$:    MOVL    R5,HP$A_DEVICE(R2)      ; Set device address                    ;G:12
                1C A2       D4  044A    839         CLRL    HP$A_DVA(R2)            ; Clear direct address
          20 A2    F4 AD     D0  044D    840         MOVL    L_LINK(FP),HP$A_LINK(R2); Set link address
                    018C    31  0452    841         BRW     CON_NORMAL_X            ; Set Default load device
                                0455    842
                                0455    843 CON_RELEASE_X:
                50   52     D0  0455    844         MOVL    R2,R0                   ; Copy buffer address
                   FBA5'    30  0458    845         BSBW    EXE$DEANONPAGED         ; Release the buffer
                                045B    846 CON_ERROR_X:
     50    00660002 8F       D0  045B    847         MOVL    #DS$_ERROR,R0           ; Error
```

```
                0462   848 CON_X:
 04             0462   849          RET
```

```
                                0463    851 CON_UBA:
                                0463    852 ;+
                                0463    853 ; initialize adaptor and unibus
                                0463    854 ;-
            23    10 AC    D1  0463    855         CMPL    16(AP),#BTD$K_AIE_TK50  ; Is it MAYA                                [43]
                     7A    13  0467    856         BEQL    30$                     ; NO INIT FOR DEBNT                         [43]
            22    10 AC    D1  0469    857         CMPL    16(AP),#BTD$K_BVPSSP    ; IS IT KFBTA ?                             [37]
                     74    13  046D    858         BEQL    30$                     ; NO INIT FOR KFBTA                         [37]
00000062    8F    10 AC    D1  046F    859         CMPL    16(AP),#BTD$K_AIE_NI    ; Is it NI                                  [41]
                     6A    13  0477    860         Beql    30$                     ; Treat as MSCP device                      [41]
   0000FE0C'EF       01    D0  0479    861         MOVL    #1,DSA$GL_UNITS         ; Set 1 UUT                                 ;G:6
00000004'EF    F4 AD       D0  0480    862         MOVL    L_LINK(FP),DS$GA_SELECTED+4 ; Set address of P-table for channel
                           0488    863
   53   00000000'EF       9E  0488    864         MOVAB   DS$CHANNEL,R3           ; Point to channel services
             F8 AD       7F  048F    865         PUSHAQ  L_STATUS(FP)            ; Address of status quadword
         00000645'EF     9F  0492    866         PUSHAB  L^COMMON_INTERRUPT      ; interrupt address for RK0X & RL0X         [05]
          03    10 AC    D1  0498    867         CmpL    16(Ap), #3              ; Is the device an RB730?                   [06]
                     07    12  049C    868         BNeq    1$                      ; Branch if not                             [06]
   6E   00000767'EF     9E  049E    869         MovAB   L^RB730_INTERRUPT, (Sp) ; Replace interrupt address                 [06]
                     00    DD  04A5    870 1$:     PUSHL   #CHC$_INITA             ; Use INITA                                 [06]
               7E    04    7D  04A7    871         MOVQ    #4,-(SP)                ; Unit zero and arg count=4
               63    6E    FA  04AA    872         CALLG   (SP),(R3)               ; Init adapter
                  B2 50    E9  04AD    873         BLBC    R0,CON_X                ; Exit if error
            08 AE    01    D0  04B0    874         MOVL    #CHC$_INITB,8(SP)       ; Function INITB
               63    6E    FA  04B4    875         CALLG   (SP),(R3)               ; Init Unibus
                  A8 50    E9  04B7    876 2$:     BLBC    R0,CON_X                ; Exit if error
            08 AE    06    D0  04BA    877         MOVL    #CHC$_CLEAR,8(SP)       ; Function Clear adapter
               63    6E    FA  04BE    878         CALLG   (SP),(R3)               ; Clear status
                  9E 50    E9  04C1    879         BLBC    R0,CON_X                ; Exit if error
            08 AE    04    D0  04C4    880         MOVL    #CHC$_ENINT,8(SP)       ; Function enable interrupts
               63    6E    FA  04C8    881         CALLG   (SP),(R3)               ; Init unibus
                  94 50    E9  04CB    882         BLBC    R0,CON_X                ; Return on error
55  08 AC    00000000'8F     C9  04CE    883         BISL3   #IOC$K_IOSPACE,8(AP),R5 ; Get real address (VA)                 [07]
                           04D7    884
            11    10 AC    D1  04D7    885         CmpL    16(Ap), #Btd$K_UDA      ; Is it a UDA (code 17)                     [07]
                     06    13  04DB    886         BeqL    30$                     ; If not, do CASE for contiguous codes      [25]
            21    10 AC    D1  04DD    887         CMPL    16(AP),#BTD$K_BDA       ; Is it a KDB50?                            [25]
                     03    12  04E1    888         BNEQ    40$                     ; Branch if not                             [25]
                   039E    31  04E3    889 30$:    BrW     Con_Uba_Uda             ; UDA50                                     [07]
                           04E6    890 40$:    CASE    16(AP),TYPE=L,LIMIT=#1,-
                           04E6    891                 displist=<con_uba_rk,con_uba_rl,con_uba_rb>
      02'   01    10 AC    CF  04E6            CASEL   16(AP),#1,S^#<<30003$-30002$>/2>-1
                           04EB        30002$:
                     0006' 04EB                .SIGNED_WORD    con_uba_rk-30002$
                     016C' 04ED                .SIGNED_WORD    con_uba_rl-30002$
                     0295' 04EF                .SIGNED_WORD    con_uba_rb-30002$
                           04F1        30003$:
                           04F1    892
```

```
                                04F1    894 .enabl  lsb
                                04F1    895 CON_UBA_RK:
                                04F1    896 ;+
                                04F1    897 ; now build PT for RK611
                                04F1    898 ;-
          65   8000 8F    B0   04F1    899          MOVW    #RK_CS1_M_CERR,RK_CS1(R5); Clear controller errors
        08 A5    0C AC    B0   04F6    900          MOVW    12(AP),RK_CS2(R5)        ; Set unit number of drive in question
          65   0045 8F    B0   04FB    901          MOVW    #RK_CS1_M_IE!F_DRVCLR!1, -
                                0500    902                  RK_CS1(R5)               ; Enable interrupts and clear drive
                                0500    903          $DS_WAITUS_S    #1000            ; Wait for it to finish
                          00    DD    0500                   PUSHL   #0
                000003E8  8F    DD    0502                   PUSHL   #1000
          00000000'9F     02    FB    0508                   CALLS   #2, a#DS$WAITUS
             54    0A A5    3C    050F    904          MOVZWL  RK_DS(R5),R4             ; Get drive status
          65   0040 8F    AA    0513    905          BICW    #RK_CS1_M_IE,RK_CS1(R5)  ; Clear interrupt enable
            08 AE    05    D0    0518    906          MOVL    #CHC$_DSINT,8(SP)        ; Change function to disable
               63    6E    FA    051C    907          CALLG   (SP),(R3)                ; Disable interrupts
                  12 50    E9    051F    908          BLBC    R0,10$                   ; Return if error
                     54    B5    0522    909          TSTW    R4                       ; Status valid?
                     0F    18    0524    910          BGEQ    20$                      ; Error if not
     54   54    01    08    EF    0526    911          EXTZV   #RK_DS_V_DDT,#1,R4,R4    ; Isolate drive type, 0=RK06, 1=Rk07
                                052B    912
                     51    37    3C    052B    913          MOVZWL  #RK611$K_LEN,R1          ; Length of RK611 PT
                                052E    914                                                                          ;G:12
                                052E    915                                                                          ;G:12
                                052E    916                                                                          ;G:12
                   00E1    30    052E    917          BSBW    Allocate_Ptable          ; Allocate space for same                [07]
                  04 50    E8    0531    918          BLBS    R0,25$                   ; Branch if succeeded
                                0534    919 10$:
                     04    0534    920          RET                              ; Return error
                   FF23    31    0535    921 20$:     BRW     CON_ERROR_X              ; Leap to error exit
                                0538    922 25$:
               08 A2    51    B0    0538    923          MOVW    R1,HP$W_SIZE(R2)         ; Set size of structure
                                053C    924
                  62    04    D0    053C    925          MOVL    #4,HP$Q_DEVICE(R2)       ; Set length of "_DMA"
               50    0C A2    9E    053F    926          MOVAB   HP$T_DEVICE(R2),R0       ; Set address of "_DMA"
               04 A2    50    D0    0543    927          MOVL    R0,HP$Q_DEVICE+4(R2)     ; Set address of device name
          60   414D445F 8F    D0    0547    928          MOVL    #^A"_DMA",(R0)           ; Set "_DMA"
               18 A2    55    D0    054E    929          MOVL    R5,HP$A_DEVICE(R2)       ; Set device address
                  1C A2    D4    0552    930          CLRL    HP$A_DVA(R2)             ; Clear virtural address
             20 A2    F4 AD    D0    0555    931          MOVL    L_LINK(FP),HP$A_LINK(R2); Set link to channel
          26 A2    5205 8F    B0    055A    932          MOVW    #^X5205,HP$T_TYPE(R2)    ; Set length and "R" of RK611
       28 A2    3131364B 8F    D0    0560    933          MOVL    #^A"K611",HP$T_TYPE+2(R2) ; Set "K611"
32 A2  18 A2    12    00    EF    0568    934          EXTZV   #0,#18, HP$A_DEVICE(R2), -
                                056F    935                  RK611$L_CSR(R2)          ; Set CSR address
                                056F    936
             24 A2    FE AD    B0    056F    937          MOVW    L_STATUS+6(FP), -
                                0574    938                  HP$W_VECTOR(R2)          ; Set current vector
                     14    13    0574    939          BEQL    30$                      ; Branch if invalid vector
       50   FC AD    04    02    EF    0576    940          EXTZV   #CHI$V_IPL,#4, -
                                057C    941                  L_STATUS+4(FP),R0        ; Extract BR level of device
               36 A2    50    90    057C    942          MOVB    R0,RK611$B_BR(R2)        ; Save BR level
                                0580    943
          00000000'EF     62    FA    0580    944          CALLG   (R2),L^INS$PTABLE        ; Add this PT                    [05]
                  03 50    E8    0587    945          BLBS    R0,40$                   ; Branch if success
                   FEC8    31    058A    946 30$:     BRW     CON_RELEASE_X            ; Release buffer and exit
                                058D    947 40$:
```

```
                F4 AD    52   D0  058D   948          MOVL    R2,L_LINK(FP)           ; Save PT of RK611
                   51    32   3C  0591   949          MOVZWL  #RK06$K_LEN,R1          ; Length of RK06/RK07 PT
                                    0594   950                                                                  ;G:12
                                    0594   951                                                                  ;G:12
                      007B   30  0594   952          BSBW    Allocate_Ptable         ; Allocate space for PT      [07]
                   9A 50     E9  0597   953          BLBC    R0,10$                  ; Take error exit
                08 A2    51   B0  059A   954          MOVW    R1,HP$W_SIZE(R2)        ; Set size of structure
                                    059E   955
                   62    05   D0  059E   956          MOVL    #5,HP$Q_DEVICE(R2)      ; Insert length of "_DMA0"
                50    0C A2    9E  05A1   957          MOVAB   HP$T_DEVICE(R2),R0      ; Insert address of "_DMA0"
                04 A2    50   D0  05A5   958          MOVL    R0,HP$Q_DEVICE+4(R2)    ; Set address of device name
             80    414D445F 8F   D0  05A9   959          MOVL    #^A"_DMA",(R0)+         ; Insert "_DMA"
              80    0C AC    30   81  05B0   960          ADDB3   #^A"0",12(AP),(R0)+     ; Insert unit number
                14 AB    0C AC    B0  05B5   961          MOVW    12(AP),HPE$W_EXT_DRIVE(R11) ; Put drive # in EPB     ;G:12
           000000FF'EF   0C AC    B1  05BA   962          CMPW    12(AP),^XFF                                          ;G:12
                         05       14  05C2   963          BGTR    45$                                                  ;G:12
                0B A2    0C AC    90  05C4   964          MOVB    12(AP),HP$B_DRIVE(R2)   ; Set drive number           ;G:12
                         18 A2    D4  05C9   965 45$:     CLRL    HP$A_DEVICE(R2)         ; Clear device address       ;G:12
                         1C A2    D4  05CC   966          CLRL    HP$A_DVA(R2)            ; Clear virtural address
                20 A2    F4 AD    D0  05CF   967          MOVL    L_LINK(FP),HP$A_LINK(R2); Link to RK611
           26 A2    304B5204 8F   D0  05D4   968          MOVL    #<^A"@RK0"-60>, -
                                    05DC   969                  HP$T_TYPE(R2)           ; Set length=4 and "RK0"
              2A A2    54   36   81  05DC   970          ADDB3   #^A"6",R4,HP$T_TYPE+4(R2); Set "6" or "7" if R4=1
                                    05E1   971 CON_NORMAL_X:
                   0000'CF   62   FA  05E1   972          calls   (r2), w^ins$ptable      ; Insert it in list
                         A1 50     E9  05E6   973          blbc    r0, 30$                 ; fail if error
                      50    52    D0  05E9   974          MOVL    R2,R0                   ; Copy address of P-table of load dev
          52    00000000'EF   DE  05EC   975          MOVAL   DS$GL_CLIBASE,R2        ; Set R2 to base of CLI data area
                08 A2    60   D0  05F3   976          MOVL    HP$Q_DEVICE(R0), -
                                    05F7   977                  CLI$Q_FILE(R2)          ; Set length of load device name
                   7E    3A   90  05F7   978          MOVB    #^A":",-(SP)            ; Tack a ":" to the end
                   53    60   D0  05FA   979          MOVL    HP$Q_DEVICE(R0),R3      ; Length of name. Any size.  [12]
                         53   D7  05FD   980          DECL    R3                      ; "-" not included           [12]
             7E    0C A043    90  05FF   981 50$:     MOVB    HP$T_DEVICE(R0)[R3],-(SP)       ; Place name on stack [12]
                      F8 53    F5  0604   982          SOBGTR  R3,50$                  ; Loop to move all of name   [12]
                0C A2    6E   9E  0607   983          MOVAB   (SP),CLI$Q_FILE+4(R2)   ; Set address of load device name
                      F9F2'   30  060B   984          BSBW    DSV$SETLOAD             ; Set default load device
                   50    01   D0  060E   985          MOVL    #SS$_NORMAL,R0          ; Success
                                04  0611   986          RET
                                    0612   987 .dsabl  lsb
```

```
                                  0612   989 .Subtitle       Common routines
                                  0612   990
                                  0612   991 ;
                                  0612   992 ; This little routine allocates a Ptable and pre-clears it.
                                  0612   993 ;
                                  0612   994 ; Inputs:
                                  0612   995 ;  R1 => Size of Ptable to allocate
                                  0612   996 ;
                                  0612   997 ; Outputs:
                                  0612   998 ;  R0 => Status code
                                  0612   999 ;  R1 => Size of Ptable
                                  0612  1000 ;  R2 => Address of Ptable
                                  0612  1001 ;  R11 => Address of EPB                                                    ;G:12
                                  0612  1002 ;
                                  0612  1003
                                  0612  1004 Allocate_Ptable:                          ; Use to allocate/clear Ptable          [07]
                0400 8F     BB    0612  1005         PUSHR   #^M<R10>                                                    ;G:12
        51   00000000'8F    C0    0616  1006         ADDL2   #HPE$C_EPB_SIZE,R1                                          ;G:12
             00000000'EF    16    061D  1007         Jsb     L^Exe$AloNonPaged        ; Allocate the structure               [07]
                      3F    BB    0623  1008         PushR   #^M<R0,R1,R2,R3,R4,R5>   ; Save volatile registers              [07]
  62   51   00   6E   00    2C    0625  1009         MovC5   #0, (Sp), #0, R1, (R2)   ; Clear the Ptable                     [07]
                      3F    BA    062B  1010         PopR    #^M<R0,R1,R2,R3,R4,R5>   ; Restore the registers                [07]
             5B   52   51    C1    062D  1011         ADDL3   R1,R2,R11                ; End of ptable in R11                ;G:12
             5A   5B   04    C3    0631  1012         SUBL3   #4,R11,R10                ; Address to store address of EPB    ;G:12
   6A   5B   00000000'8F    C3    0635  1013         SUBL3   #HPE$C_EPB_SIZE,R11,(R10); Store EPB address in               ;G:13
                5B   6A    D0    063D  1014         MOVL    (R10),R11                ; EPB address in R11                  ;G:12
                0400 8F     BA    0640  1015         POPR    #^M<R10>                                                    ;G:12
                            05    0644  1016         Rsb                              ;                                      [07]
                                  0645  1017
                                  0645  1018 COMMON_INTERRUPT:
                      03    BB    0645  1019         PUSHR   #^M<R0,R1>               ; Save volatile regs
                      7E    7C    0647  1020         CLRQ    -(SP)                    ; no status or interrupt routine
                      05    DD    0649  1021         PUSHL   #CHC$_DSINT              ; Disable further interrupts
                      00    DD    064B  1022         PUSHL   #0                       ; unit 0
        00000000'EF   04    FB    064D  1023         CALLS   #4,DS$CHANNEL            ; disable
                      03    BA    0654  1024         POPR    #^M<R0,R1>               ; Restore
                            02    0656  1025         REI                              ; All status is in L_STATUS(FP)
                                  0657  1026
```

```
                          0657  1028 CON_UBA_RL:
                          0657  1029 ;+
                          0657  1030 ; now build PT for RL11
                          0657  1031 ;-
                          0657  1032
                     0B   B0  0657  1033          MOVW    #RL_DA_M_STS!RL_DA_M_RST!1,-
                   04 A5      0659  1034                  RL_DA(R5)                   ; Setup disk adress register to
                              065B  1035                                              ; clear errors and get status.
                              065B  1036
                 52   04   D0  065B  1037          MOVL    #F_STAT,R2                  ; setup get status function
       52   02   08   0C AC   F0  065E  1038          INSV    12(AP),#8,#2,R2             ; setup unit # of drive
                 65   52   B0  0664  1039          MOVW    R2,RL_CS(R5)                ; initiate get status to clear errors
                              0667  1040          $DS_WAITUS_S    #1000               ; Wait for it to finish
                     00   DD  0667                          PUSHL   #0
                 000003E8 8F   DD  0669                      PUSHL   #1000
            00000000'9F   02   FB  066F                      CALLS   #2, a#DS$WAITUS
                              0676  1041
                65   0080 8F   B3  0676  1042          BITW    #RL_CS_M_CRDY,RL_CS(R5) ; Check if controller ready
                     16   13  067B  1043          BEQL    5$                          ; Branch if not ready, Throw it away
                              067D  1044
                  52   40 8F   88  067D  1045          BISB2   #RL_CS_M_IE,R2          ; Setup interrupt enable
                     65   52   B0  0681  1046          MOVW    R2,RL_CS(R5)            ; Now do get status to cause interrupt
                              0684  1047
                              0684  1048          $DS_WAITUS_S    #1000               ; Wait for it to finish
                     00   DD  0684                          PUSHL   #0
                 000003E8 8F   DD  0686                      PUSHL   #1000
            00000000'9F   02   FB  068C                      CALLS   #2, a#DS$WAITUS
                     54   65   3C  0693  1049 5$:      MOVZWL  RL_CS(R5),R4            ; Get status reg. contents
                65   0040 8F   AA  0696  1050          BICW    #RL_CS_M_IE,RL_CS(R5)   ; Clear interrupt enable
                  08 AE   05   D0  069B  1051          MOVL    #CHC$_DSINT,8(SP)       ; Change function to disable
                     63   6E   FA  069F  1052          CALLG   (SP),(R3)               ; Disable interrupts
                        12 50   E9  06A2  1053          BLBC    R0,10$                  ; Return if error
                     54   B5  06A5  1054          TSTW    R4                          ; Check composite error
                     0F   19  06A7  1055          BLSS    20$                         ; Error if set
          54   54   01   07   EF  06A9  1056          EXTZV   #RL_MP_V_TYP,#1,R4,R4   ; Isolate drive type, 0=RL01,1=RL02
                              06AE  1057
                     51   37   3C  06AE  1058          MOVZWL  #RL11$K_LEN,R1          ; Length of RL11 PT
                              06B1  1059                                                                      ;G:12
                              06B1  1060                                                                      ;G:12
                        FF5E   30  06B1  1061          BSBW    Allocate_Ptable         ; Allocate space for same          [07]
                      04 50   E8  06B4  1062          BLBS    R0,25$                  ; Branch if succeeded
                              06B7  1063 10$:
                              04  06B7  1064          RET                             ; Return error
                        FDA0   31  06B8  1065 20$:      BRW     CON_ERROR_X             ; Leap to error exit
                              06BB  1066 25$:
                  08 A2   51   B0  06BB  1067          MOVW    R1,HP$W_SIZE(R2)        ; Set size of structure
                              06BF  1068
                     62   04   D0  06BF  1069          MOVL    #4,HP$Q_DEVICE(R2)      ; Set length of "_DLA"
                  50   0C A2   9E  06C2  1070          MOVAB   HP$T_DEVICE(R2),R0      ; Set address of "_DLA"
                  04 A2   50   D0  06C6  1071          MOVL    R0,HP$Q_DEVICE+4(R2)    ; Set address of device name
           60   414C445F 8F   D0  06CA  1072          MOVL    #^A"_DLA",(R0)          ; Set "_DLA"
                  18 A2   55   D0  06D1  1073          MOVL    R5,HP$A_DEVICE(R2)      ; Set device address
                     1C A2   D4  06D5  1074          CLRL    HP$A_DVA(R2)            ; Clear virtural address
                20 A2   F4 AD   D0  06D8  1075          MOVL    L_LINK(FP),HP$A_LINK(R2); Set link to channel
              26 A2   5204 8F   B0  06DD  1076          MOVW    #^X5204,HP$T_TYPE(R2)   ; Set length and "R" of RL11
           28 A2   0031314C 8F   D0  06E3  1077          MOVL    #^A"L11",HP$T_TYPE+2(R2) ; Set "L11"
        32 A2   18 A2   12   00   EF  06EB  1078          EXTZV   #0,#18, HP$A_DEVICE(R2), -
```

```
                          06F2  1079                RL11$L_CSR(R2)          ; Set CSR address
                          06F2  1080
          24 A2   FE AD   B0    06F2  1081          MOVW    L_STATUS+6(FP), -
                          06F7  1082                HP$W_VECTOR(R2)         ; Set current vector
                       14 13    06F7  1083          BEQL    30$             ; Branch if invalid vector
     50   FC AD   04    02    EF    06F9  1084          EXTZV   #CHI$V_IPL,#4, -
                          06FF  1085                L_STATUS+4(FP),R0       ; Extract BR level of device
          36 A2   50    90    06FF  1086          MOVB    R0,RL11$B_BR(R2)        ; Save BR level
                          0703  1087
    00000000'EF   62    FA    0703  1088          CALLG   (R2),L^INS$PTABLE       ; Add this PT                   [05]
                 03 50  E8    070A  1089          BLBS    R0,40$                  ; Branch if success
                   FD45 31    070D  1090 30$:     BRW     CON_RELEASE_X           ; Release buffer and exit
                          0710  1091 40$:
          F4 AD   52    D0    0710  1092          MOVL    R2,L_LINK(FP)           ; Save PT of RL11
             51   32    3C    0714  1093          MOVZWL  #RL01$K_LEN,R1          ; Length of RL01/RL02 PT
                          0717  1094                                                                  ;G:12
                          0717  1095                                                                  ;G:12
                    FEF8  30    0717  1096          BSBW    Allocate_ptable         ; Allocate space for PT         [07]
                  9A 50   E9    071A  1097          BLBC    R0,10$                  ; Take error exit
          08 A2   51    B0    071D  1098          MOVW    R1,HP$W_SIZE(R2)        ; Set size of structure
                          0721  1099
             62   05    D0    0721  1100          MOVL    #5,HP$Q_DEVICE(R2)      ; Insert length of "_DLA0"
          50   0C A2   9E    0724  1101          MOVAB   HP$T_DEVICE(R2),R0      ; Insert address of "_DLA0"
          04 A2   50    D0    0728  1102          MOVL    R0,HP$Q_DEVICE+4(R2)    ; Set address of device name
    80   414C445F 8F    D0    072C  1103          MOVL    #^A"_DLA",(R0)+         ; Insert "_DLA"
     80   0C AC   30    81    0733  1104          ADDB3   #^A"0",12(AP),(R0)+     ; Insert unit number
       14 AB   0C AC   B0    0738  1105          MOVW    12(AP),HPE$W_EXT_DRIVE(R11) ; Put the drive # in R11   ;G:12
 000000FF'EF   0C AC   B1    073D  1106          CMPW    12(AP),^XFF                                         ;G:12
                    05   14    0745  1107          BGTR    45$                                                 ;G:12
       0B A2   0C AC   90    0747  1108          MOVB    12(AP),HP$B_DRIVE(R2)   ; Set drive number
                 18 A2  D4    074C  1109 45$:     CLRL    HP$A_DEVICE(R2)         ; Clear device address          ;G:12
                 1C A2  D4    074F  1110          CLRL    HP$A_DVA(R2)            ; Clear virtural address
          20 A2   F4 AD   D0    0752  1111          MOVL    L_LINK(FP),HP$A_LINK(R2); Link to RL11
 26 A2    304C5204 8F    D0    0757  1112          MOVL    #<^A"@RL0"-60>, -
                          075F  1113                HP$T_TYPE(R2)           ; Set length=4 and "RL0"
     2A A2   54   31    81    075F  1114          ADDB3   #^A"1",R4,HP$T_TYPE+4(R2); Set "1" or "2" if R4=1
                   FE7A 31    0764  1115          BRW     CON_NORMAL_X            ; now set default load device
                          0767  1116
```

```
                                  0767  .sbttl  configure RB730
                                  0767  ;
                                  0767  ; Device interrupt handler for RB730
                                  0767  ;
                                  0767  ; [warning:  this code really should not expect R5 to be valid.  It
                                  0767  ;  works because DS$WAITUS doesn't trash R5; however, there is no
                                  0767  ;  guarantee on this in the future.  THIS SHOULD BE CHANGED IN A FUTURE
                                  0767  ;  RELEASE, but I am making these alterations 3 working days prior to
                                  0767  ;  release cut-off, and I just am not up to fancies.  If it works, I'll
                                  0767  ;  be ecstatic.]
                                  0767  ;
                                  0767  RB730_Interrupt:                          ;                                    [06]
                      03   BB     0767          PushR   #^M<R0,R1>                ; Save volatile regs                 [06]
                      7E   7C     0769          ClrQ    -(Sp)                     ; no status or interrupt routine     [06]
                      05   DD     076B          PushL   #CHC$_DSInt               ; Disable further interrupts         [06]
                      00   DD     076D          PushL   #0                        ; unit 0                             [06]
        00000000'EF   04   FB     076F          CallS   #4, DS$Channel            ; disable                            [06]
        65   00000040 8F   CA     0776          BicL2   #Rb_Cs_M_Ie, Rb_Cs(R5)    ; Disable device interrupts          [06]
                      03   BA     077D          PopR    #^M<R0,R1>                ; Restore                            [06]
                           02     077F          Rei                               ; All status is in L_STATUS(FP)      [06]
                                  0780
                                  0780  con_uba_rb:
                                  0780
                                  0780  ;
                                  0780  ; Build Ptable for RB730 (Nebula IDC)
                                  0780  ;
        55   40F26200'EF   9E     0780          MovAB   ^X40F26200, R5            ; Get proper address for IDC         [08]
                      52   D4     0787          clrl    r2                        ; r2 will be shifted unit number
   52   02   08   0C AC   F0     0789          insv    12(ap), #8, #2, r2        ; set unit number in right field
                                  078F
                                  078F  ;
                                  078F  ; reset drive, with interrupt on completion
                                  078F  ;
            10 A5   0B   D0       078F          movl    #rb_mp_m_sts-             ; put get status
                                  0793                  !rb_mp_m_rst-             ;  and reset
                                  0793                  !rb_mp_m_mrk, -           ;  and mark bit
                                  0793                  rb_mp(r5)                 ;  into register
   65   00000044 8F   52   C9     0793          bisl3   r2, -                     ; put unit number
                                  079B                  #f_getstatus -            ;  and command
                                  079B                  !rb_cs_m_ie, -            ;  and interrupt enable bit
                                  079B                  rb_cs(r5)                 ;  into cs register
                                  079B          $ds_waitus_s    #1000             ; wait for it to finish
                      00   DD     079B                  PUSHL   #0
            000003E8 8F   DD     079D                  PUSHL   #1000
        00000000'9F   02   FB     07A3                  CALLS   #2, @#DS$WAITUS
                 54   65   D0     07AA          movl    rb_cs(r5), r4             ; fetch status register
        65   00000040 8F   CA     07AD          bicl2   #rb_cs_m_ie, rb_cs(r5)    ; clear interrupt enable
            08 AE   05   D0       07B4          movl    #chc$_dsint, 8(sp)        ; change CHANNEL func to disable
                 63   6E   FA     07B8          callg   (sp), (r3)                ; disable interrupts.
                   12 50   E9     07BB          blbc    r0, 10$                   ; return if error
                      54   B5     07BE          tstw    r4                        ; is good status?
                      0F   19     07C0          blss    20$                       ; composite error set
   54   54   01   1A   EF       07C2          extzv   #rb_cs_v_typ, #1, r4, r4; get type bit (0 = rl02, 1 = r80)
                 51   37   3C     07C7          movzwl  #rb730$k_len, r1          ; length of RB730 Ptable
                                  07CA                                                                                  ;G:12
                                  07CA                                                                                  ;G:12
                                  07CA                                                                                  ;G:12
```

```
                FE45   30  07CA          bsbw    Allocate_ptable          ; allocate space for it                [07]
               04 50   E8  07CD          blbs    r0, 30$                  ; check status
                           07D0  10$:
                       04  07D0          ret                              ; return error
                           07D1  20$:
                FC87   31  07D1          brw     con_error_x              ; error exit (create status)
                           07D4  30$:
                           07D4  ;
                           07D4  ; device responded: make Ptable
                           07D4  ;
                           07D4  ; Register usage:
                           07D4  ;
                           07D4  ;       R1 = size of Ptable
                           07D4  ;       R2 = pointer to Ptable
                           07D4  ;       R3 = address of DSX$CHANNEL routine
                           07D4  ;       R4 = 0 if device is RL02, 1 if R80
                           07D4  ;       R5 = address of device CS
                           07D4  ;
                           07D4
          08 A2   51   B0  07D4          movw    r1, hp$w_size(r2)        ; Set size of Ptable
             62   04   D0  07D8          movl    #4, hp$q_device(r2)      ; set length of "_DQA"
          50   0C A2   9E  07DB          movab   hp$t_device(r2), r0      ; set address of "_DQA"
          04 A2   50   D0  07DF          movl    r0, hp$q_device+4(r2)    ; set address of device name
     60   4151445F 8F  D0  07E3          movl    #^a"_DQA", (r0)          ; set "_DQA" as name
          18 A2   55   D0  07EA          movl    r5, hp$a_device(r2)      ; set device address
               1C A2   D4  07EE          clrl    hp$a_dva(r2)             ; clear virtual address
       20 A2   F4 AD   D0  07F1          movl    l_link(fp), -
                           07F6                  hp$a_link(r2)            ; set link field
     26 A2   5205 8F   B0  07F6          movw    #^x5205, hp$t_type(r2)   ; set length and 'R' of RB730
  28 A2   30333742 8F  D0  07FC          movl    #^a"B730", -
                           0804                  hp$t_type+2(r2)          ; set rest of type field
       24 A2   FE AD   B0  0804          movw    l_status+6(fp), -
                           0809                  hp$w_vector(r2)          ; set vector it interrupted thru
                   12  13  0809          beql    40$                      ; branch if invalid
 50   FC AD   04   02  EF  080B          extzv   #chi$v_ipl, #4, -
                           0811                  l_status+4(fp), r0       ; get BR level
          36 A2   50   90  0811          movb    r0, rb730$b_br(r2)       ; save BR level
       0000'CF   62   FA  0815          callg   (r2), w^ins$ptable       ; add this Ptable to list
               03 50   E8  081A          blbs    r0, 50$                  ; branch if OK
                           081D  40$:
                FC35   31  081D          brw     con_release_x            ; release buffer and exit
                           0820  50$:
          F4 AD   52   D0  0820          movl    r2, l_link(fp)           ; save link of RB730
             51   32   3C  0824          movzwl  #rl02$k_len, r1          ; length of RL02/R80 Ptable
                           0827                                                                                  ;G:12
                           0827                                                                                  ;G:12
                FDE8   30  0827          bsbw    Allocate_Ptable          ; allocate Ptable                      [07]
               A3 50   E9  082A          blbc    r0, 10$                  ; exit if error
          08 A2   51   B0  082D          movw    r1, hp$w_size(r2)        ; set size
             62   05   D0  0831          movl    #5, hp$q_device(r2)      ; set length of "_DQA0"
          50   0C A2   9E  0834          movab   hp$t_device(r2), r0      ; address of "_DQA0"
          04 A2   50   D0  0838          movl    r0, hp$q_device+4(r2)    ; set into descriptor
     80   4151445F 8F  D0  083C          movl    #^a"_DQA", (r0)+         ; set first part of name
     60   0C AC   30   81  0843          addb3   #^a"0", 12(ap), (r0)     ; insert correct unit number
       14 AB   0C AC   B0  0848          MOVW    12(AP),HPE$W_EXT_DRIVE(R11)  ; Put the drive # in EPB          ;G:12
 000000FF'EF   0C AC   B1  084D          CMPW    12(AP),^XFF                                                    ;G:12
                   05  14  0855          BGTR    55$                                                            ;G:12
```

```
                0B A2    0C AC    90  0857  1229          movb    12(ap), hp$b_drive(r2)  ; set binary unit number
                         18 A2    D4  085C  1230 55$:     clrl    hp$a_device(r2)         ; no address                    ;G:12
                         1C A2    D4  085F  1231          clrl    hp$a_dva(r2)            ; no pass-thru address
                20 A2    F4 AD    D0  0862  1232          movl    l_link(fp), -
                                      0867  1233                   hp$a_link(r2)           ; set link address to RB730
                   50    26 A2    9E  0867  1234          movab   hp$t_type(r2), r0       ; set address of type field
                         09 54    E9  086B  1235          blbc    r4, 60$                 ; branch if RL02
                                      086E  1236 ;
                                      086E  1237 ; Set device type to R80
                                      086E  1238 ;
             60   30385203 8F    D0  086E  1239          movl    #3+<^a"R80"a8>, (r0)    ; set type to counted "R80"
                         0A    11  0875  1240          brb     70$                     ; join common code
                                      0877  1241 ;
                                      0877  1242 ; Set device type to RL02
                                      0877  1243 ;
             80   304C5204 8F    D0  0877  1244 60$:     movl    #4+<^a"RL0"a8>, (r0)+   ; set type to counted "RL02"
                     60   32    90  087E  1245          movb    #^a"2", (r0)            ;
                                      0881  1246 ;
                                      0881  1247 ; Insert Ptable, return
                                      0881  1248 ;
                       FD5D   31  0881  1249 70$:     brw     con_normal_x            ; set load device, exit OK
                                      0884  1250
```

```
                                0884  1252          .Subtitle       Attach UDA-50/LESI/KDB50/KFBTA/DEBNT load path          ;G:9
                                0884  1253
                                0884  1254 ;                                                                                [07]
                                0884  1255 ; Since there seems to be no particular purpose to using interrupts
                                0884  1256 ; when initializing the device, this code uses the perfectly good
                                0884  1257 ; PUBTDRIVR/BDABTDRIVR/PBBTDRIVR code for the purpose, and merely checks the status
                                0884  1258 ;
                                0884  1259
                                0884  1260 con_uba_uda:
                                0884  1261
                                0884  1262 ;
                                0884  1263 ; First, turn off interrupts (which were set up in common
                                0884  1264 ; UBA code)
                                0884  1265 ;
                                0884  1266
                23   10 AC   D1 0884  1267          CMPL    16(AP),#BTD$K_AIE_TK50  ; Is it MAYA                          [43]
                       1D   13 0888  1268          BEQL    2$                      ; NO INIT FOR DEBNT                   [43]
                22   10 AC   D1 088A  1269          CMPL    16(AP),#BTD$K_BVPSSP    ; IS IT KFBTA ?                       [37]
                       17   13 088E  1270          BEQL    2$                      ; NO INIT FOR KFBTA                   [37]
          00000062 8F   10 AC   D1 0890  1271          CMPL    16(AP),#BTD$K_AIE_NI    ; Is it NI ?                          [41]
                       0D   13 0898  1272          BEQL    2$                      ; No init for NI                      [41]
                08 AE    05   D0 089A  1273          MovL    #Chc$_DsInt, 8(Sp)      ; change CHANNEL func to disable
                   63    6E   FA 089E  1274          CallG   (Sp), (R3)              ; disable interrupts.
                      03 50   E8 08A1  1275          Blbs    R0, 2$                  ; return if error                     ;G:6
                       00D8   31 08A4  1276          BRW     10$
                                08A7  1277 ;
                                08A7  1278 ; Now, build a pseudo-RPB to make the bootdriver init code                    ;G:13
                                08A7  1279 ; happy, and then call said routine.  After it completes, the
                                08A7  1280 ; global response buffer is accessible: it has the drive type, etc.
                                08A7  1281 ;
                   18 AC   DD 08A7  1282 2$:     pushL   24(AP)                  ; BI number if applicable             [33]
                      7E   7C 08AA  1283          CLRQ    -(SP)                   ; No slave number/No ci port          [07]
                   0C AC   DD 08AC  1284          PushL   12(Ap)                  ; Unit number                         [07]
                50 F4 BD   DE 08AF  1285          MovAL   aL_Link(Fp), R0         ; Get address of link Ptable          [07]
                   18 A0   DD 08B3  1286          PushL   Hp$A_Device(R0)         ; Address of link device              [07]
                      55   DD 08B6  1287          PushL   R5                      ; Address of device CSR               [07]
                      7E   7C 08B8  1288          ClrQ    -(Sp)                   ; Don't bother with adap./dev. codes  [07]
    00000000'EF       08   FB 08BA  1289          CallS   #8, Dsr$Allocate_Pseudo_RPB     ; Allocate the RPB            [07]
                      50   D5 08C1  1290          TstL    R0                      ; If R0 is 0,                         [07]
                      03   12 08C3  1291          BNEQ    3$                      ;                                     ;G:2
                    00B8   31 08C5  1292          BRW     20$                     ; .. there was an error               [36]
                59    50   D0 08C8  1293 3$:     MovL    R0, R9                  ; Put RPB where UD_INIT wants it      [07]
                                08CB  1294
    50  00000000'EF   DE 08CB  1295          MOVAL   UD_INIT,R0              ; UDA50 init routine                  [29]
    52  00000000'EF   9E 08D2  1296          MOVAB   UDA_RESPONSE,R2         ; UDA50 response buffer               [29]
          11   10 AC   D1 08D9  1297          CMPL    16(AP),#BTD$K_UDA       ; Is it UDA50 ?                       [44]
                      48   13 08DD  1298          BEQL    5$                      ; BRANCH IF UDA50                     [44]
    50  00000000'EF   DE 08DF  1299          MOVAL   PA_INIT,R0              ; DEBN* initi routine                 [43]
          23   10 AC   D1 08E6  1300          CMPL    16(AP),#BTD$K_AIE_TK50  ; Is it MAYA ?                        [36]
                      3B   13 08EA  1301          Beql    5$                      ; Branch if MAYA                      [43]
    50  00000000'EF   DE 08EC  1302          moval   pb$init,r0              ; KFBTA init routine                  [44]
          22   10 AC   D1 08F3  1303          CMPL    16(AP),#BTD$K_BVPSSP    ; Is it KFBTA ?                       [44]
                      2E   13 08F7  1304          BEQL    5$                      ; BRANCH IF KFBTA                     [36]
  00000062 8F   10 AC   D1 08F9  1305          CMPL    16(AP),#BTD$K_AIE_NI    ; Is it NI ?                          [41]
                      10   12 0901  1306          Bneq    57$                     ; Branch if not NI                    [41]
    59  00000000'EF   DE 0903  1307          moval   NI$GN_RPB_BUFFER,r9     ; use the real rpb for now            [41]
    50  00000000'EF   DE 090A  1308          moval   et$init,r0              ; DEBNT init routine                  [41]
```

```
                    14   11  0911  1309              brb     5$                              ;                                       [41]
            21   10 AC   D1  0913  1310 57$:         CMPL    16(AP),#BTD$K_BDA               ; Is it a KDB50?                        [25]
                    0E   12  0917  1311              BNEQ    5$                              ; Branch if not                         [25]
        50  00000000'EF  DE  0919  1312              MOVAL   BD_INIT,R0                      ; KDB50 init routine                    [25]
        52  00000000'EF  9E  0920  1313              MOVAB   KDB50_RESPONSE,R2               ; KDB50 response buffer                 [25]
                         0927  1314
                    50   D5  0927  1315 5$:         TSTL    R0                              ; Check if zero                         [37]
                    13   12  0929  1316              BNEQ    501$                            ; Branch if good                        [37]
                         092B  1317                                                                                              ;G:2
                         092B  1318              ERRSUP_S                                    ; Should never happen!                  [37]
             00000000'EF   DF  092B                          PUSHAL  $MODULE
                    00   DD  0931                          PUSHL   #0
                    03   DD  0933                          PUSHL   #$ER
        00000000'EF 03   FB  0935                          CALLS   #3, DS_ERRSUP
                    41   11  093C  1319              brb     10$                             ; return on error                       [37]
                         093E  1320                                                                                              ;G:2
                60  00   FB  093E  1321 501$:        CallS   #0,(R0)                         ; Call init routine                     [25]
                  3B 50   E9  0941  1322              BlbC    R0, 10$                         ; Return if error                       [07]
                    69   9F  0944  1323              PushAB  (R9)                            ; Address of RPB                        [07]
        00000000'EF 01   FB  0946  1324              CallS   #1, Dsr$DeAllocate_Pseudo_RPB   ; Deallocate the RPB                    [07]
                59  52   D0  094D  1325              Movl    R2,R9                           ; Get address of response buffer        [25]
               59   1C A9   D0  0950  1326              MovL    MSCP$L_Media_ID(R9), R9 ; Replace R9 with media ID codes        [07]
               22   10 AC   D1  0954  1327              CMPL    16(AP),#BTD$K_BVPSSP            ; Is it KFBTA ?                 [36]
                    10   13  0958  1328              beql    601$                            ; Branch if KFBTA                       [36]
               23   10 AC   D1  095A  1329              CMPL    16(AP),#BTD$K_AIE_TK50  ; Is it MAYA ?                          [36]
                    0A   13  095E  1330              Beql    601$                            ;                                       [36]
    00000062 8F     10 AC   D1  0960  1331              CMPL    16(AP),#BTD$K_AIE_NI    ; Is it NI ?                            [41]
                    03   12  0968  1332              Bneq    67$                             ;                                       [41]
                  0093   31  096A  1333 601$:         BRW     51$                             ; KFBTA, skip UDA50/LESI PTABLE build   [36]
                         096D  1334
          21    10 AC   D1  096D  1335 67$:          CMPL    16(AP),#BTD$K_BDA               ; Is it a KDB50?                        [25]
                    03   12  0971  1336              BNEQ    7$                              ; Branch if not                         [25]
                  008A   31  0973  1337              BRW     51$                             ; KDB50, skip UDA50/LESI ptable build   [25]
                         0976  1338 ;
                         0976  1339 ; Build Ptable for UDA-50 (UNIBUS => SDI adapter)
                         0976  1340 ; Build Ptable for LESI                                                                      [19]
                         0976  1341 ;
                         0976  1342
            51   38   3C  0976  1343 7$:          MovZWL  #Hp$K_UDA50_Len, r1             ; length of UDA-50 Ptable               [25]
                         0979  1344                                                                                              ;G:12
                         0979  1345                                                                                              ;G:12
              FC96   30  0979  1346              BsbW    Allocate_Ptable                 ; allocate space for it                 [07]
             04 50   E8  097C  1347              BlbS    R0, 30$                         ; check status
                         097F  1348 10$:
                    04   097F  1349              Ret                                     ; return error
                         0980  1350 20$:
              FAD8   31  0980  1351              BrW     Con_Error_X                     ; error exit (create status)
                         0983  1352 30$:
                         0983  1353 ;
                         0983  1354 ; device responded: make Ptable
                         0983  1355 ;
                         0983  1356 ; Register usage:
                         0983  1357 ;
                         0983  1358 ;       R1 = size of Ptable
                         0983  1359 ;       R2 = pointer to Ptable
                         0983  1360 ;       R3 = scratch
                         0983  1361 ;       R4 = scratch
```

```
                                  0983  ;       R5 = address of device CS
                                  0983  ;       R9 = UDA-50 drive media ID code
                                  0983  ;
                                  0983  ;               31   26   21   16   11  7 6    0
                                  0983  ;               +----+----+----+----+----+------+
                                  0983  ;               | D0 | D1 | A0 | A1 | A2 |   N  |
                                  0983  ;               +----+----+----+----+----+------+                                  ;G:14
                                  0983  ;                                                                                    ;G:13
                                  0983  ;                       D0, D1 : 5-bit ASCII preferred device
                                  0983  ;                                generic prefix for unit. 1='A', 2='B'.
                                  0983  ;                                D0 is left letter, D1 is right.
                                  0983  ;                       A0,A1,A2,N : Media name of unit.  Up to 3 alpha
                                  0983  ;                                characters (left-justified in A0-2), and
                                  0983  ;                                N (two decimal digits).  Ex: A0=17,
                                  0983  ;                                A1=1, A2=0, N=80; name is 'RA80'.
                                  0983  ;                                Name can also be RA81,RA60,RC25,RCF25,RX50,  ;G:12
                                  0983  ;                                RD52,RD53,RA90                                ;G:12
                                  0983  ;
                                  0983
          08 A2    51    B0      0983          MovW    R1, Hp$W_Size(R2)          ; Set size of Ptable
             62    04    D0      0987          MovL    #4, Hp$Q_Device(R2)        ; set length of "_DUA"
          50    0C A2    9E      098A          MovAB   Hp$T_Device(R2), R0        ; set address of "_DUA"
          04 A2    50    D0      098E          MovL    R0, Hp$Q_Device+4(R2)      ; set address of device name
     60   4155445F 8F    D0      0992          MovL    #^A"_DUA", (R0)            ; set "_DUA" as name/DEFAULT                [10]
          18 A2    55    D0      0999          MovL    R5, Hp$A_Device(R2)        ; set device address
  32 A2   55   12   00    EF      099D          ExtZV   #0, #18, R5, -             ; Set UNIBUS address into
                                  09A3                   Hp$L_Uda50_UdaIp(R2)      ; .. correct field of Ptable
                1C A2    D4      09A3          ClrL    Hp$A_Dva(R2)               ; clear sibling address
       20 A2    F4 AD    D0      09A6          MovL    L_Link(Fp), -
                                  09AB                   Hp$A_Link(R2)             ; set link field
     20643019 8F    59    D1      09AB          CMPL    R9,#^X20643019             ; Check for RC25                            [10]
                    09    13      09B2          BEQL    33$                        ; Branch if found.                          [10]
     20643319 8F    59    D1      09B4          CMPL    R9,#^X20643319             ; Check for RCF25                           [10]
                    18    12      09BB          BNEQ    35$                        ; Branch if found.                          [10]
                                  09BD
      26 A2   4C04 8F    B0      09BD  33$:    MOVW    #4!<^A"L"a8>, -            ; Set length and 'L'                        [13]
                                  09C3                   HP$T_TYPE(R2)             ; of LESI                                   [10]
  28 A2   20495345 8F    D0      09C3          MOVL    #^A"ESI ", -               ; and set                                   [13]
                                  09CB                   HP$T_TYPE+2(R2)           ; rest of type field                        [10]
  0C A2   4141445F 8F    D0      09CB          MOVL    #^A"_DAA",HP$T_DEVICE(R2); RESET "_DAA" AS NAME                      [18]
                    0E    11      09D3          BRB     38$                        ; Branch to continue                        [10]
                                  09D5
      26 A2   5505 8F    B0      09D5  35$:    MovW    #5!<^A"U"a8>, -            ; Set length and 'U'
                                  09DB                   Hp$T_Type(R2)             ; .. of UDA50
  28 A2   30354144 8F    D0      09DB          MovL    #^A"DA50", -               ; And set
                                  09E3                   Hp$T_Type+2(R2)           ; .. rest of type field
                                  09E3
      24 A2   006C 8F    B0      09E3  38$:    MovW    #^O154, Hp$W_Vector(R2)    ; Set vector of fst UDA
          36 A2    05    90      09E9          MovB    #5, Hp$B_UDA50_Br(R2)      ; save BR level
          37 A2    02    90      09ED          MovB    #2, Hp$B_UDA50_Burst(R2); Assume default burst rate                  [07]
       0000'CF     62    FA      09F1          CallG   (R2), W^Ins$Ptable         ; add this Ptable to list
                03 50    E8      09F6          BlbS    R0, 50$                    ; branch if OK
                                  09F9  40$:
                FA59     31      09F9          BrW     Con_Release_X              ; release buffer and exit
                                  09FC  50$:
          F4 AD    52    D0      09FC          MovL    R2, L_Link(Fp)             ; save link of UDA50/LESI
             51    32    3C      0A00  51$:    MovZWL  #RA80$K_Len, R1            ; length of RA80/81/60,RC25,RCF25,          [09]
```

```
                                0A03  1419                                    ; RD53,RD52,RX50, TK50,NI Ptable        [26]
                                0A03  1420                                                                      ;G:12
                                0A03  1421                                                                      ;G:12
                     FC0C   30  0A03  1422          BsbW    Allocate_Ptable  ; allocate Ptable                      [07]
                   03 50    E8  0A06  1423          BlbS    R0, 52$          ; continue if no error                [10]
                     FF73   31  0A09  1424          BRW     10$              ; exit if error                       [10]
               08 A2    51   B0  0A0C  1425 52$:    MovW    R1, Hp$W_Size(R2)  ; set size
                  62    05   D0  0A10  1426          MovL    #5, Hp$Q_Device(R2)  ; set length of "_DUA0"
               50   0C A2   9E  0A13  1427          MovAB   Hp$T_Device(R2), R0  ; address of "_DUA0"
               04 A2    50   D0  0A17  1428          MovL    R0, Hp$Q_Device+4(R2)  ; set into descriptor
         60    4155445F 8F   D0  0A1B  1429          MovL    #^A"_DUA", (R0)  ; set first part of name,DEFAULT      [10]
               23   10 AC   D1  0A22  1430          CMPL    16(AP),#BTD$K_AIE_TK50  ; Is it MAYA ?                 [43]
                     07    12  0A26  1431          bneq    53$              ; Branch if not                       [43]
         60    41554D5F 8F   D0  0A28  1432          MovL    #^A"_MUA", (R0)  ; set first part of name,DEFAULT      [43]
      53 59    0A   16   EF  0A2F  1433 53$:    ExtZV   #22, #10, R9, R3        ; Get preferred name                  ;G:9
                     16    13  0A34  1434          BEql    55$              ; Branch if null
      54 53    05   00   EF  0A36  1435          ExtZV   #0, #5, R3, R4          ; Get second character
      53 53    05   05   EF  0A3B  1436          ExtZV   #5, #5, R3, R3          ; Get first character
   01 A0  53    40 8F   81  0A40  1437          AddB3   #<^A"A"-1>, R3, 1(R0)   ; Replace first character
   02 A0  54    40 8F   81  0A46  1438          AddB3   #<^A"A"-1>, R4, 2(R0)   ; .. and second as well
            53   F4 AD   D0  0A4C  1439 55$:    MOVL    L_LINK(FP),R3           ; Fetch address of link ptable        [31]
         03 A0   0F A3   90  0A50  1440          MOVB    HP$T_DEVICE+3(R3),3(R0) ; Inherit controller letter   [31]
               50   04   C0  0A55  1441          AddL2   #4, R0                  ; Increment pointer
            53   0C AC   D0  0A58  1442          MOVL    12(AP),R3               ; Get drive #                         [20]
                          0A5C  1443                                                                            ;G:13
   00000000'EF   00   90  0A5C  1444          MOVB    #0,DEVICE_UNIT_FLAG     ; Clea unit number flag               ;G:14
                 54   D4  0A63  1445          CLRL    R4                      ; Clear High order field              ;G:13
54 53 53 00002710 8F   7B  0A65  1446          EDIV    #10000,R3,R3,R4         ; Set first digit                     ;G:13
                     0D   13  0A6E  1447          BEQL    90$                     ; Branch if less than 5 digits        ;G:13
         80   53    30   81  0A70  1448          ADDB3   #^A"0",R3,(R0)+         ; Set first ascii digit               ;G:13
                 62   D6  0A74  1449          INCL    HP$Q_DEVICE(R2)         ; Increase length of string           ;G:13
   00000000'EF   01   88  0A76  1450          BISB2   #1,DEVICE_UNIT_FLAG     ; Non-zero digit has been found       ;G:14
                          0A7D  1451                                                                            ;G:13
      03E8 8F    54   B1  0A7D  1452 90$:    CMPW    R4,#1000                ; Digit equal to zero?                ;G:14
                     1C   1F  0A82  1453          BLSSU   95$                     ; Branch if yes                       ;G:14
                 55   D4  0A84  1454          CLRL    R5                      ; Clear high order field              ;G:14
54 53 54 000003E8 8F   7B  0A86  1455          EDIV    #1000,R4,R3,R4          ; Get next digit                      ;G:13
                     1B   13  0A8F  1456          BEQL    100$                    ; Branch if less than 4 digits        ;G:13
         80   53    30   81  0A91  1457          ADDB3   #^A"0",R3,(R0)+         ; Set next ascii digit                ;G:13
                 62   D6  0A95  1458          INCL    HP$Q_DEVICE(R2)         ; Increase length of string           ;G:13
   00000000'EF   01   88  0A97  1459          BISB2   #1,DEVICE_UNIT_FLAG     ; Non-zero digit has been found       ;G:14
                     0C   11  0A9E  1460          BRB     100$                    ; Continue                            ;G:14
   05 00000000'EF    E9   0AA0  1461 95$:    BLBC    DEVICE_UNIT_FLAG,100$   ; Non zero digit hasn't been found    ;G:14
               80    30   90  0AA7  1462          MOVB    #^A"0",(R0)+            ; Set Ascii Digit                     ;G:14
                 62   D6  0AAA  1463          INCL    HP$Q_DEVICE(R2)         ; Increase length of string           ;G:14
            64 8F    54   91  0AAC  1464 100$:   CMPB    R4,#100                 ; Digit equal to zero?                ;G:14
                     1C   1F  0AB0  1465          BLSSU   110$                    ; Branch if yes                       ;G:14
                 55   D4  0AB2  1466          CLRL    R5                      ; Clear high order field              [20]
54 53 54 00000064 8F   7B  0AB4  1467          EDIV    #100,R4,R3,R4           ; Set first digit                     [20]
                     1B   13  0ABD  1468          BEQL    120$                    ; Branch if number is <100            [20]
         80   53    30   81  0ABF  1469          ADDB3   #^A"0",R3,(R0)+         ; Set first ascii digit               [20]
                 62   D6  0AC3  1470          INCL    HP$Q_DEVICE(R2)         ; Increase length of string           [20]
   00000000'EF   01   88  0AC5  1471          BISB2   #1,DEVICE_UNIT_FLAG     ; Non-zero digit has been found       ;G:14
                     0C   11  0ACC  1472          BRB     120$                    ; Continue                            ;G:14
   05 00000000'EF    E9   0ACE  1473 110$:   BLBC    DEVICE_UNIT_FLAG,120$   ; Non zero digit hasn't been found    ;G:14
               80    30   90  0AD5  1474          MOVB    #^A"0",(R0)+            ; Set Ascii Digit                     ;G:14
                 62   D6  0AD8  1475          INCL    HP$Q_DEVICE(R2)         ; Increase length of string           ;G:14
```

```
                0A    0C AC   91  0ADA  1476 120$:   CMPB    12(AP),#10                  ; Is it >=10    ?                          [20]
                         16   1F  0ADE  1477         BLSSU   125$                        ; Branch if not.                           [20]
                         55   D4  0AE0  1478         CLRL    R5                          ; Clear high order field                   [20]
          54    53    54  0A   7B  0AE2  1479         EDIV    #10,R4,R3,R4                ; Get next digit                           [20]
                80    53  30   81  0AE7  1480         ADDB3   #^A"0",R3,(R0)+             ; Set this ascii digit                     [20]
                         62   D6  0AEB  1481         INCL    HP$Q_DEVICE(R2)             ; Increase length of string                [20]
          00000000'EF    01   88  0AED  1482         BISB2   #1,DEVICE_UNIT_FLAG         ; Non-zero digit has been found          ;G:14
                         0C   11  0AF4  1483         BRB     130$                        ; Continue                               ;G:14
             05 00000000'EF   E9  0AF6  1484 125$:   BLBC    DEVICE_UNIT_FLAG,130$       ; Non zero digit hasn't been found       ;G:14
                    80   30   90  0AFD  1485         MOVB    #^A"0",(R0)+                ; Set digit                              ;G:14
                         62   D6  0B00  1486         INCL    HP$Q_DEVICE(R2)             ; Increase length of string                [20]
                60    54  30   81  0B02  1487 130$:   ADDB3   #^A"0",R4,(R0)              ; Set last digit                           [20]
             14 AB    0C AC   B0  0B06  1488         MOVW    12(AP),HPE$W_EXT_DRIVE(R11) ; Put the drive # in EPB                 ;G:12
       000000FF'EF    0C AC   B1  0B0B  1489         CMPW    12(AP),^XFF                                                          ;G:12
                         05   14  0B13  1490         BGTR    131$                                                                 ;G:12
             0B A2    0C AC   90  0B15  1491         MOVB    12(AP),HP$B_DRIVE(R2)       ; Set drive number                         [20]
                      18 A2   D4  0B1A  1492 131$:   ClrL    Hp$A_Device(R2)             ; no address                             ;G:12
                      1C A2   D4  0B1D  1493         ClrL    Hp$A_Dva(R2)                ; no pass-thru address
             20 A2    F4 AD   D0  0B20  1494         MovL    L_Link(Fp), -               ;                                        ;G:12
                                  0B25  1495                  Hp$A_Link(R2)              ; set link address to UDA50/LESI/KDB50     [25]
                23    10 AC   D1  0B25  1496         CMPL    16(AP),#BTD$K_AIE_TK50      ; Is it MAYA ?                             [43]
                         14   12  0B29  1497         bneq    135$                        ; Branch if not                            [43]
                         59   D5  0B2B  1498         TSTL    R9                          ; Did a response come back ?               [43]
                         10   12  0B2D  1499         BNEQ    135$                        ; Branch if YES                            [43]
          26 A2    5404 8F   B0  0B2F  1500         MOVW    #4!<^A"T"a8>, -             ; Set length and 'T'.  DEFAULT             [43]
                                  0B35  1501                  HP$T_TYPE(R2)              ; of TK50                                  [43]
       28 A2    2030354B 8F   D0  0B35  1502         MOVL    #^A"K50 ", -                ; and set                                  [43]
                                  0B3D  1503                  HP$T_TYPE+2(R2)            ; rest of type field                       [43]
                         3C   11  0B3D  1504         BRB     70$                         ; EXIT                                     [43]
                50    26 A2   9E  0B3F  1505 135$:   MovAB   Hp$T_Type(R2), R0           ; set address of type field              ;G:9
                         50   D6  0B43  1506         IncL    R0                          ; Skip count byte
                    51   50   D0  0B45  1507         MovL    R0, R1                      ; Copy start of length
                    53   11   D0  0B48  1508         MovL    #17, R3                     ; Bit of first character
          54    59    05  53   EF  0B4B  1509 60$:    ExtZV   R3, #5, R9, R4              ; Get next character
                         0F   13  0B50  1510         BEql    65$                         ; Exit if null
            80    54    40 8F   81  0B52  1511         AddB3   #<^A"A"-1>, R4, (R0)+       ; Otherwise, store it
  FFEA 53   FFFFFFFB 8F   06   F1  0B57  1512         AcbL    #6, #-5, R3, 60$            ; Loop through characters
          53    59    07  00   EF  0B61  1513 65$:    ExtZV   #0, #7, R9, R3              ; Get number
                         54   D4  0B66  1514         ClrL    R4                          ; Clear high order
          54    53    53  0A   7B  0B68  1515         EDiv    #10, R3, R3, R4             ; Divide by 10
                80    53  30   81  0B6D  1516         AddB3   #^A"0", R3, (R0)+           ; First digit
                80    54  30   81  0B71  1517         AddB3   #^A"0", R4, (R0)+           ; Second digit
                    50   51   C2  0B75  1518         SubL2   R1, R0                      ; Get length
                    71   50   90  0B78  1519         MovB    R0, -(R1)                   ; Store byte length
                                  0B7B  1520
                                  0B7B  1521 ;
                                  0B7B  1522 ; Insert Ptable, return
                                  0B7B  1523 ;
                                  0B7B  1524
                      FA63   31  0B7B  1525 70$:    BrW     Con_Normal_X                ; set load device, exit OK
                                  0B7E  1526                                             ;                                  -      [07]
                                  0B7E  1527
                                  0B7E  1528         .END
```

```
$ER                             = 00000004    D
$MODULE                           00000000 R  D  02
ALLOCATE_PTABLE                   00000612 R  D  04
BD_INIT                           ********W G    00
BOOT$L_ADP                        ********   X   00
BOOT$L_CSR                        ********   X   00
BOOT$L_DEVTYP                     ********   X   00
BOOT$L_R1                         ********   X   00
BOOT$L_R2                         ********   X   00
BOOT$L_UNIT                       ********   X   00
BTD$K_AIE_NI                    = 00000062    D
BTD$K_AIE_TK50                  = 00000023    D
BTD$K_BDA                       = 00000021    D
BTD$K_BVPSSP                    = 00000022    D
BTD$K_HSCCI                     = 00000020    D
BTD$K_UDA                       = 00000011    D
CHC$_ABORT                      = 00000002    D
CHC$_CLEAR                      = 00000006    D
CHC$_CLRDFT                     = 00000009    D
CHC$_DSINT                      = 00000005    D
CHC$_ENINT                      = 00000004    D
CHC$_INITA                      = 00000000    D
CHC$_INITB                      = 00000001    D
CHC$_PURGE                      = 00000003    D
CHC$_SELF_TEST                  = 0000000B    D
CHC$_SETDFT                     = 00000008    D
CHC$_STATUS                     = 00000007    D
CHC$_STOP                       = 0000000A    D
CHI$M_CHNINT                    = 00000001    D
CHI$M_DEVINT                    = 00000002    D
CHI$M_IPL                       = 0000007C    D
CHI$S_IPL                       = 00000005    D
CHI$V_CHNINT                    = 00000000    D
CHI$V_DEVINT                    = 00000001    D
CHI$V_IPL                       = 00000002    D
CI$B_BI_ID                        00000032    D
CI$B_BR                           00000033    D
CI$B_NODE                         00000034    D
CI$B_TR                           00000032    D
CI$K_LEN                          00000041    D
CI$L_FUNC                         00000035    D
CI$L_INDEX                        0000003D    D
CI$L_MAINT_ID                     00000039    D
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        0000044C    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_FLAGS2                      00000444    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEMP_BASE                   00000448    D
CLI$L_TEST                        00000020    D
CLI$M_START_OR_RUN              = 00000008    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BASE_QUAL                 = 00000002    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NEXT                      = 00000001    D
CLI$V_NOALLOC                   = 00000000    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_START_OR_RUN              = 00000003    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
COMMON_INTERRUPT                  00000645 R  D  04
CONSOLE_CODE                    = 00000040    D
CON_CI_HSCCI                      000000AE R  D  04
CON_ERROR_X                       0000045B R  D  04
CON_MBA                           000003C4 R  D  04
CON_NORMAL_X                      000005E1 R  D  04
CON_RELEASE_X                     00000455 R  D  04
CON_UBA                           00000463 R  D  04
CON_UBA_RB                        00000780 R  D  04
CON_UBA_RK                        000004F1 R  D  04
CON_UBA_RL                        00000657 R  D  04
CON_UBA_UDA                       00000884 R  D  04
CON_X                             00000462 R  D  04
DEVICE_UNIT_FLAG                  00000000 R  D  03
DIR...                          = FFFFFFFF    D
```

```
DISK$K_LEN                      00000032    D
DS$CHANNEL                      ********    X   00
DS$GA_SELECTED                  ********    X   00
DS$GL_CLIBASE                   ********    X   00
DS$K_ERROR                    = 00000002    D
DS$K_NORMAL                   = 00000001    D
DS$K_SEVERE                   = 00000004    D
DS$K_SUBSYS                   = 00000066    D
DS$K_WARNING                  = 00000000    D
DS$WAITUS                       ********    X   00
DS$_AP_NORMAL_BREAK           = 00660150    D
DS$_ARITH                     = 006600D0    D
DS$_ASBE                      = 00660118    D
DS$_BADLINK                   = 006600F0    D
DS$_BADTYPE                   = 006600E8    D
DS$_BIIC                      = 00660120    D
DS$_CHME                      = 006600A8    D
DS$_CHMK                      = 006600E0    D
DS$_DEVNAME                   = 00660108    D
DS$_ERROR                     = 00660002    D
DS$_FHWE                      = 00660068    D
DS$_FRAGBUF                   = 00660080    D
DS$_ICBUSY                    = 006600C8    D
DS$_ICERR                     = 006600C0    D
DS$_IHWE                      = 00660060    D
DS$_ILLCHAR                   = 00660018    D
DS$_ILLPAGCNT                 = 00660078    D
DS$_ILLUNIT                   = 00660100    D
DS$_INIT_FAIL                 = 00660140    D
DS$_INSFMEM                   = 00660050    D
DS$_INVCPU                    = 00660130    D
DS$_IPL2HI                    = 006600B8    D
DS$_IVADDR                    = 00660040    D
DS$_IVVECT                    = 00660038    D
DS$_KRNLSTK                   = 00660090    D
DS$_LOGIC                     = 00660070    D
DS$_MCHK                      = 00660088    D
DS$_MEM_ALLOC_ERR             = 00660138    D
DS$_MMOFF                     = 00660058    D
DS$_NEEDUNIT                  = 006600F8    D
DS$_NODE                      = 00660128    D
DS$_NOPCS                     = 00660110    D
DS$_NORMAL                    = 00660001    D
DS$_NOSUPPORT                 = 006600B1    D
DS$_NOTALLOWMP                = 00660148    D
DS$_NOTDON                    = 00660030    D
DS$_NOTIMP                    = 006600B0    D
DS$_NULLSTR                   = 00660010    D
DS$_OVERFLOW                  = 00660008    D
DS$_POWER                     = 00660098    D
DS$_PROGERR                   = 00660020    D
DS$_SEVERE                    = 00660004    D
DS$_TRANSL                    = 006600A0    D
DS$_TRUNCATE                  = 00660028    D
DS$_UNEXPINT                  = 006600D8    D
DS$_UNSUPPDEV                 = 00660158    D
DS$_VASFULL                   = 00660048    D
DS$_WARNING                   = 00660000    D
DSA$AL_APTMAIL                  0000FE00    D
DSA$AT_APTTXT                   0000FA00    D
DSA$GL_APTCOM                   0000FE04    D
DSA$GL_DEVLEN                   0000FE58    D
DSA$GL_ERRNO                    0000FE44    D
DSA$GL_EVENT                    0000FE48    D
DSA$GL_FLAGS                    0000FE00    D
DSA$GL_MSGTYP                   0000FE40    D
DSA$GL_PASSES                   0000FE08    D
DSA$GL_PASSNO                   0000FE54    D
DSA$GL_SECTNO                   0000FE10    D
DSA$GL_SID                      0000FE14    D
DSA$GL_SUBTNO                   0000FE4C    D
DSA$GL_TESTNO                   0000FE50    D
DSA$GL_UNITS                    0000FE0C    D
DSA$GQ_MSGPTR                   0000FE68    D
DSA$GT_DEVNAM                   0000FE5C    D
DSP$CONFIG_ADP                  ********    X   00
DSP$CONFIG_LOAD                 00000041 RG D   04
DSR$ALLOCATE_PSEUDO_RPB         ********    X   00
DSR$DEALLOCATE_PSEUDO_RPB       ********    X   00
DSR$ONLY_CONSOLE_CONFIG         ********    X   00
DSV$SETLOAD                     ********    X   00
DS_ERRSUP                       ********    X   00
DT_RM03                       = 00000015    D
DT_RM05                       = 00000017    D
DT_RM80                       = 00000016    D
DT_RP04                       = 00000010    D
DT_RP05                       = 00000011    D
DT_RP06                       = 00000012    D
DT_RP07                       = 00000022    D
DT_RP07_HPT                   = 00000021    D
ET$INIT                         ********W G     00
EXE$ALONONPAGED                 ********    X   00
EXE$DEANONPAGED                 ********    X   00
EXE$GB_CPUTYPE                  ********    X   00
F_DRVCLR                      = 00000004    D
F_GETSTATUS                   = 00000004    D
F_NOP                         = 00000000    D
F_STAT                        = 00000004    D
HP$A_DEPENDENT                  00000032    D
HP$A_DEVICE                     00000018    D
HP$A_DVA                        0000001C    D
HP$A_LINK                       00000020    D
HP$B_CI_BR                      00000033    D
HP$B_CI_NODE                    00000034    D
HP$B_CI_TR                      00000032    D
HP$B_DRIVE                      0000000B    D
HP$B_FLAGS                      0000000A    D
HP$B_RB730_BR                   00000036    D
HP$B_RK611_BR                   00000036    D
HP$B_RL11_BR                    00000036    D
HP$B_UDA50_BR                   00000036    D
HP$B_UDA50_BURST                00000037    D
HP$C_NAMELEN                  = 00000014  G D
HP$K_CI_LEN                     00000041    D
```

```
HP$K_DISK_LEN                   00000032    D
HP$K_RA80_LEN                   00000032    D
HP$K_RA90_LEN                   00000032    D
HP$K_RB730_LEN                  00000037    D
HP$K_RK06_LEN                   00000032    D
HP$K_RK611_LEN                  00000037    D
HP$K_RL01_LEN                   00000032    D
HP$K_RL02_LEN                   00000032    D
HP$K_RL11_LEN                   00000037    D
HP$K_RP04_LEN                   00000032    D
HP$K_UDA50_LEN                  00000038    D
HP$L_CI_FUNC                    00000035    D
HP$L_CI_INDEX                   0000003D    D
HP$L_CI_MAINT_ID                00000039    D
HP$L_RB730_CSR                  00000032    D
HP$L_RK611_CSR                  00000032    D
HP$L_RL11_CSR                   00000032    D
HP$L_UDA50_UDAIP                00000032    D
HP$M_NAME                     = 00000004    D
HP$Q_DEVICE                     00000000    D
HP$T_DEVICE                     0000000C    D
HP$T_TYPE                       00000026    D
HP$W_SIZE                       00000008    D
HP$W_VECTOR                     00000024    D
HPE$A_EPB                       00000016    D
HPE$C_EPB_SIZE                  ********      X   00
HPE$T_DEVICE                    00000000      D
HPE$W_EXT_DRIVE                 00000014      D
HSC50_BOOT                      0000009F RG D   02
INI$LOAD_DEVICE                 00000000 RG D   04
INS$PTABLE                      ********      X   00
IOC$K_IOSPACE                   ********      X   00
KDB50_RESPONSE                  ********W G       00
K_TYPES                       = 00000013      D
L_LINK                          FFFFFFF4      D
L_LOCAL                         FFFFFFF0      D
L_STATUS                        FFFFFFF8      D
L_TYPE                          FFFFFFF0      D
MSCP$L_MEDIA_ID               = 0000001C      D
NI$GN_RPB_BUFFER                ********W G       00
PA_INIT                         ********W G       00
PB$INIT                         ********W G       00
RB0$K_LEN                       00000032      D
RA80$K_LEN                      00000032      D
RA90$K_LEN                      00000032      D
RB730$B_BR                      00000036      D
RB730$K_LEN                     00000037      D
RB730$L_CSR                     00000032      D
RB730_INTERRUPT                 00000767 R    D   04
RB_BA                           00000004      D
RB_BC                           00000008      D
RB_CS                           00000000      D
RB_CS_M_IE                    = 00000040      D
RB_CS_V_TYP                   = 0000001A      D
RB_DA                           0000000C      D
RB_MP                           00000010      D
RB_MP_M_MRK                   = 00000001      D
RB_MP_M_RST                   = 00000008      D
RB_MP_M_STS                   = 00000002      D
RK06$K_LEN                      00000032      D
RK611$B_BR                      00000036      D
RK611$K_LEN                     00000037      D
RK611$L_CSR                     00000032      D
RK_CS1                        = 00000000      D
RK_CS1_M_CERR                 = 00008000      D
RK_CS1_M_IE                   = 00000040      D
RK_CS1_M_RDY                  = 00000080      D
RK_CS2                        = 00000008      D
RK_DS                         = 0000000A      D
RK_DS_V_DDT                   = 00000008      D
RL01$K_LEN                      00000032      D
RL02$K_LEN                      00000032      D
RL11$B_BR                       00000036      D
RL11$K_LEN                      00000037      D
RL11$L_CSR                      00000032      D
RL_CS                         = 00000000      D
RL_CS_M_CRDY                  = 00000080      D
RL_CS_M_IE                    = 00000040      D
RL_DA                         = 00000004      D
RL_DA_M_MRK                   = 00000001      D
RL_DA_M_RST                   = 00000008      D
RL_DA_M_STS                   = 00000002      D
RL_MP                         = 00000006      D
RL_MP_V_TYP                   = 00000007      D
RP04$K_LEN                      00000032      D
RP_CS1                        = 00000000      D
RP_DT                         = 00000018      D
RP_DT_S_DTN                   = 00000009      D
RP_DT_V_DTN                   = 00000000      D
SAVABS...                     = FFFFFFF0      D
SCB_BASE                        ********      X   00
SCB_IMAGE                       ********      X   00
SIZ...                        = 00000002      D
SS$_NORMAL                    = 00000001      D
SYS$UNWIND                      ********      X   00
T_TYPES                         00000007 R    D   02
UDA50$B_BR                      00000036      D
UDA50$B_BURST                   00000037      D
UDA50$K_LEN                     00000038      D
UDA50$L_UDAIP                   00000032      D
UDA_RESPONSE                    ********W G       00
UD_INIT                         ********W G       00
```

```
                                    +----------------+
                                    ! Psect synopsis !
                                    +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           FFFFFFF8  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
DATA                            000000A0  (  160.)  02 (  2.)  NOPIC   USR   CON   REL   LCL   SHR NOEXE   RD  NOWRT NOVEC LONG
WORK                            00000001  (    1.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
CODE                            00000B7E  ( 2942.)  04 (  4.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000004	1318 (10)	#-1318 (10) #-570 (5) #-823 (7)
$MODULE	00000000-R	416 (2)	1318 (10) 570 (5) 823 (7)
ALLOCATE_PTABLE	00000612-R	1004 (9)	#-1061 (10) #-1096 (10) #-1172 (10) #-1218 (10)
			#-1346 (10) #-1422 (10) #-611 (6) #-689 (6)
			#-810 (7) #-917 (9) #-952 (9)
BD_INIT	00000000-XR		1312 (10) 411 (2)
BIT...	=00000010	266 (2)	262 (2) 265 (2) 266 (2)
BOOT$L_ADP	00000000-XR		407 (2) #-479 (3)
BOOT$L_CSR	00000000-XR		407 (2) #-487 (3)
BOOT$L_DEVTYP	00000000-XR		407 (2) #-481 (3) #-485 (3)
BOOT$L_R1	00000000-XR		407 (2) #-483 (3)
BOOT$L_R2	00000000-XR		407 (2) #-484 (3)
BOOT$L_UNIT	00000000-XR		407 (2) #-486 (3)
BTD$K_AIE_NI	=00000062		#-1271 (10) #-1305 (10) #-1331 (10) #-557 (5)
			#-859 (8)
BTD$K_AIE_TK50	=00000023		#-1267 (10) #-1300 (10) #-1329 (10) #-1430 (10)
			#-1496 (10) #-555 (5) #-855 (8)
BTD$K_BDA	=00000021		#-1310 (10) #-1335 (10) #-561 (5) #-887 (8)
BTD$K_BVPSSP	=00000022		#-1269 (10) #-1303 (10) #-1327 (10) #-553 (5)
			#-857 (8)
BTD$K_HSCCI	=00000020		#-565 (5)
BTD$K_UDA	=00000011		#-1297 (10) #-559 (5) #-885 (8)
CHC$_ABORT	=00000002	265 (2)	
CHC$_CLEAR	=00000006	265 (2)	#-877 (8)
CHC$_CLRDFT	=00000009	265 (2)	
CHC$_DSINT	=00000005	265 (2)	#-1021 (9) #-1051 (10) #-1132 (10) #-1162 (10)
			#-1273 (10) #-906 (9)
CHC$_ENINT	=00000004	265 (2)	#-880 (8)
CHC$_INITA	=00000000	265 (2)	#-870 (8)
CHC$_INITB	=00000001	265 (2)	#-874 (8)
CHC$_PURGE	=00000003	265 (2)	
CHC$_SELF_TEST	=0000000B	265 (2)	
CHC$_SETDFT	=00000008	265 (2)	
CHC$_STATUS	=00000007	265 (2)	
CHC$_STOP	=0000000A	265 (2)	
CHI$M_CHNINT	=00000001	266 (2)	
CHI$M_DEVINT	=00000002	266 (2)	
CHI$M_IPL	=0000007C	266 (2)	
CHI$S_IPL	=00000005	266 (2)	
CHI$V_CHNINT	=00000000	266 (2)	
CHI$V_DEVINT	=00000001	266 (2)	
CHI$V_IPL	=00000002	266 (2)	#-1084 (10) #-1206 (10) #-940 (9)
CLI$K_BUFSIZ	=00000100	233 (2)	233 (2)
CLI$K_SIZE	0000044C	233 (2)	
CLI$L_ADDRESS	00000018	233 (2)	
CLI$L_COMMAND	00000004	233 (2)	
CLI$L_DATA	0000001C	233 (2)	
CLI$L_FLAGS	00000000	233 (2)	
CLI$L_FLAGS2	00000444	233 (2)	
CLI$L_LAST	00000024	233 (2)	

```
CLI$L_NEXT                      00000030     233    (2)
CLI$L_PASS                      0000002C     233    (2)
CLI$L_SUBT                      00000028     233    (2)
CLI$L_TEMP_BASE                 00000448     233    (2)
CLI$L_TEST                      00000020     233    (2)
CLI$M_START_OR_RUN             =00000008     233    (2)
CLI$Q_BUFQWD                    00000034     233    (2)
CLI$Q_FILE                      00000008     233    (2)    #-782    (6)    #-789    (6)    #-977    (9)    #-983    (9)
CLI$Q_SECTION                   00000010     233    (2)
CLI$Q_TIME                      0000043C     233    (2)
CLI$T_BUFFER                    0000003C     233    (2)
CLI$V_ADAPTER                  =00000018     233    (2)
CLI$V_ADR                      =0000000B     233    (2)
CLI$V_ASCII                    =00000013     233    (2)
CLI$V_BASE_QUAL                =00000002     233    (2)
CLI$V_BREAK                    =0000000A     233    (2)
CLI$V_BRIEF                    =0000001B     233    (2)
CLI$V_BYTE                     =0000000D     233    (2)
CLI$V_CLEAR                    =00000002     233    (2)
CLI$V_DEC                      =00000010     233    (2)
CLI$V_DEFAULT                  =0000000C     233    (2)
CLI$V_DEPOSIT                  =00000019     233    (2)
CLI$V_EVENT                    =00000008     233    (2)
CLI$V_EXAM                     =00000005     233    (2)
CLI$V_FLAGS                    =00000009     233    (2)
CLI$V_HEX                      =00000012     233    (2)
CLI$V_KERNEL                   =00000017     233    (2)
CLI$V_LOAD                     =00000006     233    (2)
CLI$V_LONG                     =0000000F     233    (2)
CLI$V_NEXT                     =00000001     233    (2)
CLI$V_NOALLOC                  =00000000     233    (2)
CLI$V_NOTNUF                   =00000001     233    (2)
CLI$V_OCT                      =00000011     233    (2)
CLI$V_PREG                     =0000001A     233    (2)
CLI$V_QA                       =00000007     233    (2)
CLI$V_QACKLOOPLOOPS            =0000001C     233    (2)
CLI$V_QAERRORPRINTS            =0000001B     233    (2)
CLI$V_QAMULTIPLEPASS           =0000001F     233    (2)
CLI$V_QASUBTESTLOOPS           =0000001E     233    (2)
CLI$V_QATESTLOOPS              =0000001D     233    (2)
CLI$V_REG                      =00000014     233    (2)
CLI$V_REQUIRED                 =00000000     233    (2)
CLI$V_RUN                      =00000015     233    (2)
CLI$V_SET                      =00000003     233    (2)
CLI$V_SHOW                     =00000004     233    (2)
CLI$V_START_OR_RUN             =00000003     233    (2)      233    (2)
CLI$V_VALSEC                   =00000016     233    (2)
CLI$V_WORD                     =0000000E     233    (2)
COMMON_INTERRUPT                00000645-R   1018   (9)      866    (8)
CONSOLE_CODE                   =00000040     303    (2)    #-481    (3)
CON_CI_HSCCI                    000000AE-R   575    (6)    #-567    (5)
CON_ERROR_X                     0000045B-R   846    (7)    #-1065   (10)   #-1177   (10)   #-1351   (10)   #-921    (9)
CON_MBA                         000003C4-R   795    (7)      569    (5)
CON_NORMAL_X                    000005E1-R   971    (9)    #-1115   (10)   #-1249   (10)   #-1525   (10)   #-841    (7)
CON_RELEASE_X                   00000455-R   843    (7)    #-1090   (10)   #-1212   (10)   #-1415   (10)   #-684    (6)
                                                           #-778    (6)    #-824    (7)    #-946    (9)
CON_UBA                         00000463-R   851    (8)    #-563    (5)      569    (5)
```

```
CON_UBA_RB                      00000780-R    1139    (10)      891    (8)
CON_UBA_RK                      000004F1-R    895     (9)       891    (8)
CON_UBA_RL                      00000657-R    1028    (10)      891    (8)
CON_UBA_UDA                     00000884-R    1260    (10)    #-889    (8)
CON_X                           00000462-R    848     (7)     #-873    (8)     #-876    (8)     #-879    (8)     #-882    (8)
DEVICE_UNIT_FLAG                00000000-R    436     (2)     #-1444   (10)    #-1450   (10)    #-1459   (10)    #-1461   (10)
                                                              #-1471   (10)    #-1473   (10)    #-1482   (10)    #-1484   (10)
                                                              #-626    (6)     #-631    (6)     #-640    (6)     #-642    (6)
                                                              #-652    (6)     #-654    (6)     #-663    (6)     #-666    (6)
                                                              #-715    (6)     #-725    (6)     #-734    (6)     #-736    (6)
                                                              #-747    (6)     #-750    (6)     #-761    (6)     #-763    (6)
DIR...                          =FFFFFFFF     542     (4)       542    (4)
DS$CHANNEL                       00000000-XR                    1023   (9)       1134   (10)      397    (2)       864    (8)
DS$GA_SELECTED                   00000000-XR                    400    (2)     #-862    (8)
DS$GL_CLIBASE                    00000000-XR                    400    (2)       780    (6)      975    (9)
DS$K_ERROR                      =00000002     262     (2)
DS$K_NORMAL                     =00000001     262     (2)
DS$K_SEVERE                     =00000004     262     (2)
DS$K_SUBSYS                     =00000066     262     (2)       262    (2)
DS$K_WARNING                    =00000000     262     (2)
DS$WAITUS                        00000000-XR                    1040   (10)      1048   (10)      1159   (10)      397    (2)
                                                               903    (9)
DS$_AP_NORMAL_BREAK             =00660150     262     (2)
DS$_ARITH                       =006600D0     262     (2)
DS$_ASBE                        =00660118     262     (2)
DS$_BADLINK                     =006600F0     262     (2)
DS$_BADTYPE                     =006600E8     262     (2)
DS$_BIIC                        =00660120     262     (2)
DS$_CHME                        =006600A8     262     (2)
DS$_CHMK                        =006600E0     262     (2)
DS$_DEVNAME                     =00660108     262     (2)
DS$_ERROR                       =00660002     262     (2)     #-571    (5)     #-847    (7)
DS$_FHWE                        =00660068     262     (2)
DS$_FRAGBUF                     =00660080     262     (2)
DS$_ICBUSY                      =006600C8     262     (2)
DS$_ICERR                       =006600C0     262     (2)
DS$_IHWE                        =00660060     262     (2)
DS$_ILLCHAR                     =00660018     262     (2)
DS$_ILLPAGCNT                   =00660078     262     (2)
DS$_ILLUNIT                     =00660100     262     (2)
DS$_INIT_FAIL                   =00660140     262     (2)
DS$_INSFMEM                     =00660050     262     (2)
DS$_INVCPU                      =00660130     262     (2)
DS$_IPL2HI                      =006600B8     262     (2)
DS$_IVADDR                      =00660040     262     (2)
DS$_IVVECT                      =00660038     262     (2)
DS$_KRNLSTK                     =00660090     262     (2)
DS$_LOGIC                       =00660070     262     (2)
DS$_MCHK                        =00660088     262     (2)
DS$_MEM_ALLOC_ERR               =00660138     262     (2)
DS$_MMOFF                       =00660058     262     (2)
DS$_NEEDUNIT                    =006600F8     262     (2)
DS$_NODE                        =00660128     262     (2)
DS$_NOPCS                       =00660110     262     (2)
DS$_NORMAL                      =00660001     262     (2)
DS$_NOSUPPORT                   =006600B1     262     (2)
DS$_NOTALLOWMP                  =00660148     262     (2)
```

```
DS$_NOTDON                      =00660030     262     (2)
DS$_NOTIMP                      =006600B0     262     (2)       262     (2)
DS$_NULLSTR                     =00660010     262     (2)
DS$_OVERFLOW                    =00660008     262     (2)
DS$_POWER                       =00660098     262     (2)
DS$_PROGERR                     =00660020     262     (2)
DS$_SEVERE                      =00660004     262     (2)
DS$_TRANSL                      =006600A0     262     (2)
DS$_TRUNCATE                    =00660028     262     (2)
DS$_UNEXPINT                    =006600D8     262     (2)
DS$_UNSUPPDEV                   =00660158     262     (2)
DS$_VASFULL                     =00660048     262     (2)
DS$_WARNING                     =00660000     262     (2)       262     (2)
DSA$GL_UNITS                     0000FE0C                      #-861     (8)
DSP$CONFIG_ADP                   00000000-XR                    397     (2)       551     (5)
DSP$CONFIG_LOAD                  00000041-R    545     (5)      489     (3)
DSR$ALLOCATE_PSEUDO_RPB          00000000-XR                   1289     (10)      405     (2)       589     (6)
DSR$DEALLOCATE_PSEUDO_RPB        00000000-XR                   1324     (10)      406     (2)       599     (6)       603     (6)
DSR$ONLY_CONSOLE_CONFIG          00000000-XR                    401     (2)       477     (3)
DSV$SETLOAD                      00000000-XR                    398     (2)     #-790     (6)     #-984     (9)
DS_ERRSUP                        00000000-XR                   1318     (10)      397     (2)       570     (5)       823     (7)
DT_RM03                         =00000015     299     (2)
DT_RM05                         =00000017     301     (2)
DT_RM80                         =00000016     300     (2)
DT_RP04                         =00000010     294     (2)     #-817     (7)
DT_RP05                         =00000011     295     (2)
DT_RP06                         =00000012     296     (2)
DT_RP07                         =00000022     298     (2)
DT_RP07_HPT                     =00000021     297     (2)
ET$INIT                          00000000-XR                   1308     (10)      412     (2)
EXE$ALONONPAGED                  00000000-XR                   1007     (9)       396     (2)
EXE$DEANONPAGED                  00000000-XR                    396     (2)     #-845     (7)
EXE$GB_CPUTYPE                   00000000-XR                    396     (2)
F_DRVCLR                        =00000004     275     (2)     #-901     (9)
F_GETSTATUS                     =00000004     390     (2)     #-1156    (10)
F_NOP                           =00000000     290     (2)     #-801     (7)
F_STAT                          =00000004     287     (2)     #-1037    (10)
HP$A_DEVICE                      00000018                     #-1073    (10)      1078    (10)      #-1109    (10)      #-1196    (10)
                                                              #-1230    (10)    #-1286    (10)      #-1386    (10)      #-1492    (10)
                                                              #-586     (6)     #-674     (6)       #-675     (6)       #-838     (7)
                                                              #-929     (9)       934     (9)       #-965     (9)
HP$A_DVA                         0000001C                     #-1074    (10)    #-1110    (10)      #-1197    (10)      #-1231    (10)
                                                              #-1389    (10)    #-1493    (10)      #-675     (6)       #-775     (6)
                                                              #-839     (7)     #-930     (9)       #-966     (9)
HP$A_LINK                        00000020                     #-1075    (10)    #-1111    (10)      #-1199    (10)      #-1233    (10)
                                                              #-1391    (10)    #-1495    (10)      #-676     (6)       #-776     (6)
                                                              #-840     (7)     #-931     (9)       #-967     (9)
HP$B_CI_NODE                     00000034                     #-680     (6)
HP$B_DRIVE                       0000000B                     #-1108    (10)    #-1229    (10)      #-1491    (10)      #-673     (6)
                                                              #-774     (6)     #-837     (7)       #-964     (9)
HP$B_FLAGS                       0000000A                     #-696     (6)
HP$B_UDA50_BR                    00000036                     #-1410    (10)
HP$B_UDA50_BURST                 00000037                     #-1411    (10)
HP$C_NAMELEN                    =00000014     302     (2)     #-685     (6)     #-698     (6)
HP$K_CI_LEN                      00000041                     #-607     (6)
HP$K_DISK_LEN                    00000032                     #-685     (6)
HP$K_UDA50_LEN                   00000038                     #-1343    (10)
```

HP$L_CI_FUNC	00000035			#-679	(6)					
HP$L_CI_INDEX	0000003D			#-677	(6)					
HP$L_CI_MAINT_ID	00000039			#-678	(6)					
HP$L_UDA50_UDAIP	00000032			#-1388	(10)					
HP$M_NAME	=00000004			#-696	(6)					
HP$Q_DEVICE	00000000			#-1069	(10)	#-1071	(10)	#-1100	(10)	#-1102	(10)
				#-1192	(10)	#-1194	(10)	#-1221	(10)	#-1223	(10)
				#-1382	(10)	#-1384	(10)	#-1426	(10)	#-1428	(10)
				#-1449	(10)	#-1458	(10)	#-1463	(10)	#-1470	(10)
				#-1475	(10)	#-1481	(10)	#-1486	(10)	#-618	(6)
				#-620	(6)	#-632	(6)	#-639	(6)	#-644	(6)
				#-651	(6)	#-656	(6)	#-664	(6)	#-668	(6)
				#-695	(6)	#-700	(6)	#-724	(6)	#-733	(6)
				#-739	(6)	#-748	(6)	#-753	(6)	#-760	(6)
				#-766	(6)	#-781	(6)	#-784	(6)	#-786	(6)
				#-829	(7)	#-831	(7)	#-925	(9)	#-927	(9)
				#-956	(9)	#-958	(9)	#-976	(9)	#-979	(9)
HP$T_DEVICE	0000000C			1070	(10)	1101	(10)	1193	(10)	1222	(10)
				1383	(10)	#-1401	(10)	1427	(10)	#-1440	(10)
				619	(6)	#-623	(6)	703	(6)	830	(7)
				926	(9)	957	(9)	#-981	(9)		
HP$T_TYPE	00000026			#-1076	(10)	#-1077	(10)	#-1113	(10)	#-1114	(10)
				#-1200	(10)	#-1202	(10)	1234	(10)	#-1398	(10)
				#-1400	(10)	#-1405	(10)	#-1407	(10)	#-1501	(10)
				#-1503	(10)	1505	(10)	#-616	(6)	#-617	(6)
				#-693	(6)	#-694	(6)	#-826	(7)	#-827	(7)
				#-932	(9)	#-933	(9)	#-969	(9)	#-970	(9)
HP$W_SIZE	00000008			#-1067	(10)	#-1098	(10)	#-1191	(10)	#-1220	(10)
				#-1381	(10)	#-1425	(10)	#-615	(6)	#-692	(6)
				#-697	(6)	#-815	(7)	#-923	(9)	#-954	(9)
HP$W_VECTOR	00000024			#-1082	(10)	#-1204	(10)	#-1409	(10)	#-938	(9)
HPE$C_EPB_SIZE	00000000-XR			#-1006	(9)	#-1013	(9)	408	(2)		
HPE$W_EXT_DRIVE	00000014			#-1105	(10)	#-1226	(10)	#-1488	(10)	#-670	(6)
				#-771	(6)	#-834	(7)	#-961	(9)		
HSC50_BOOT	0000009F-R	431	(2)	#-594	(6)	#-596	(6)				
INI$LOAD_DEVICE	00000000-R	475	(3)								
INS$PTABLE	00000000-XR			1088	(10)	1209	(10)	1412	(10)	397	(2)
				682	(6)	777	(6)	944	(9)	972	(9)
IOC$K_IOSPACE	00000000-XR			395	(2)	#-674	(6)	#-799	(7)	#-883	(8)
KDB50_RESPONSE	00000000-XR			1313	(10)	411	(2)				
K_TYPES	=00000013	430	(2)	#-819	(7)						
L_LINK	FFFFFFF4	542	(4)	#-1075	(10)	#-1092	(10)	#-1111	(10)	#-1198	(10)
				#-1214	(10)	#-1232	(10)	1285	(10)	#-1390	(10)
				#-1417	(10)	#-1439	(10)	#-1494	(10)	549	(5)
				585	(6)	#-622	(6)	#-676	(6)	#-840	(7)
				#-862	(8)	#-931	(9)	#-948	(9)	#-967	(9)
L_LOCAL	FFFFFFF0	542	(4)	547	(5)						
L_STATUS	FFFFFFF8	542	(4)	#-1081	(10)	1085	(10)	#-1203	(10)	1207	(10)
				865	(8)	#-937	(9)	941	(9)		
L_TYPE	FFFFFFF0	542	(4)	548	(5)						
MSCP$L_MEDIA_ID	=0000001C			#-1326	(10)						
NI$GN_RPB_BUFFER	00000000-XR			1307	(10)	403	(2)				
PA_INIT	00000000-XR			1299	(10)	404	(2)	595	(6)		
PB$INIT	00000000-XR			1302	(10)	412	(2)				
RA80$K_LEN	00000032			#-1418	(10)						
RB730$B_BR	00000036			#-1208	(10)						
RB730$K_LEN	00000037			#-1168	(10)						

```
RB730_INTERRUPT                 00000767-R   1129  (10)     869  (8)
RB_CS                           00000000                 #-1135  (10)   #-1158  (10)   #-1160  (10)   #-1161  (10)
RB_CS_M_IE                     =00000040                 #-1135  (10)   #-1157  (10)   #-1161  (10)
RB_CS_V_TYP                    =0000001A                 #-1167  (10)
RB_MP                           00000010                 #-1154  (10)
RB_MP_M_MRK                    =00000001                 #-1153  (10)
RB_MP_M_RST                    =00000008                 #-1153  (10)
RB_MP_M_STS                    =00000002                 #-1152  (10)
RK06$K_LEN                      00000032                 #-949   (9)
RK611$B_BR                      00000036                 #-942   (9)
RK611$K_LEN                     00000037                 #-913   (9)
RK611$L_CSR                     00000032                 #-935   (9)
RK_CS1                         =00000000     268  (2)   #-899   (9)    #-902   (9)    #-905   (9)
RK_CS1_M_CERR                  =00008000     271  (2)   #-899   (9)
RK_CS1_M_IE                    =00000040     269  (2)   #-901   (9)    #-905   (9)
RK_CS1_M_RDY                   =00000080     270  (2)
RK_CS2                         =00000008     272  (2)   #-900   (9)
RK_DS                          =0000000A     273  (2)   #-904   (9)
RK_DS_V_DDT                    =00000008     274  (2)   #-911   (9)
RL01$K_LEN                      00000032                 #-1093  (10)
RL02$K_LEN                      00000032                 #-1215  (10)
RL11$B_BR                       00000036                 #-1086  (10)
RL11$K_LEN                      00000037                 #-1058  (10)
RL11$L_CSR                      00000032                 #-1079  (10)
RL_CS                          =00000000     278  (2)   #-1039  (10)   #-1042  (10)   #-1046  (10)   #-1049  (10)
                                                        #-1050  (10)
RL_CS_M_CRDY                   =00000080     279  (2)   #-1042  (10)
RL_CS_M_IE                     =00000040     280  (2)   #-1045  (10)   #-1050  (10)
RL_DA                          =00000004     281  (2)   #-1034  (10)
RL_DA_M_MRK                    =00000001     282  (2)
RL_DA_M_RST                    =00000008     284  (2)   #-1033  (10)
RL_DA_M_STS                    =00000002     283  (2)   #-1033  (10)
RL_MP                          =00000006     285  (2)
RL_MP_V_TYP                    =00000007     286  (2)   #-1056  (10)
RP04$K_LEN                      00000032                 #-805   (7)
RP_CS1                         =00000000     289  (2)   #-801   (7)
RP_DT                          =00000018     291  (2)   #-802   (7)
RP_DT_S_DTN                    =00000009     293  (2)   #-803   (7)
RP_DT_V_DTN                    =00000000     292  (2)   #-803   (7)
SAVABS...                      =FFFFFFF0     542  (4)
SCB_BASE                        00000000-XR              395  (2)
SCB_IMAGE                       00000000-XR              395  (2)
SIZ...                         =00000002     266  (2)   266  (2)
SS$_NORMAL                     =00000001                 #-791  (6)    #-985  (9)
SYS$UNWIND                      00000000-XR              399  (2)
T_TYPES                         00000007-R   417  (2)   430  (2)    #-821  (7)
UDA_RESPONSE                    00000000-XR              1296  (10)    402  (2)
UD_INIT                         00000000-XR              1295  (10)    402  (2)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BTDDEF	2	235 (2)	235 (2)
$DEF	1	266 (2)	
$DEFINI	1	234 (2)	234 (2) 235 (2) 236 (2) 241 (2) 242 (2)
			243 (2) 244 (2) 245 (2) 246 (2) 247 (2)
			248 (2) 249 (2) 250 (2) 251 (2) 252 (2)
			253 (2) 254 (2) 255 (2) 261 (2) 263 (2)
			308 (2)
$DS_CHCDEF	1	265 (2)	265 (2)
$DS_CHIDEF	1	266 (2)	266 (2)
$DS_CI_DEF	2	254 (2)	254 (2)
$DS_DISK_DEF	1	255 (2)	255 (2)
$DS_DSADEF	5	261 (2)	261 (2)
$DS_DSDEF	3	262 (2)	262 (2)
$DS_HPEODEF	1	241 (2)	241 (2)
$DS_HPODEF	2	242 (2)	242 (2)
$DS_R80_DEF	1	246 (2)	246 (2)
$DS_RA80_DEF	1	252 (2)	252 (2)
$DS_RA90_DEF	1	253 (2)	253 (2)
$DS_RB730_DEF	1	245 (2)	245 (2)
$DS_RK06_DEF	1	248 (2)	248 (2)
$DS_RK611_DEF	1	244 (2)	244 (2)
$DS_RL01_DEF	1	250 (2)	250 (2)
$DS_RL02_DEF	1	247 (2)	247 (2)
$DS_RL11_DEF	1	249 (2)	249 (2)
$DS_RP04_DEF	1	243 (2)	243 (2)
$DS_UDA50_DEF	1	251 (2)	251 (2)
$DS_WAITUS_S	1	903 (9)	1040 (10) 1048 (10) 1159 (10) 903 (9)
$EQU	1	266 (2)	262 (2) 265 (2)
$EQULS1	1	265 (2)	262 (2) 265 (2)
$EQULST	1	262 (2)	262 (2) 265 (2)
$GBLINI	2		262 (2) 265 (2) 266 (2)
$MSCPDEF	25	236 (2)	236 (2)
$OFFSET	2	538 (4)	538 (4)
$OFFST1	1	542 (4)	542 (4)
$PRDEF	5	234 (2)	234 (2)
$PUSHADR	1	570 (5)	1040 (10) 1048 (10) 1159 (10) 1318 (10) 570 (5)
			823 (7) 903 (9)
$SSDEF	21	263 (2)	263 (2)
$VIELD	1		266 (2)
$VIELD1	1	266 (2)	266 (2)
BLBCW	1	778 (6)	778 (6)
CASE	1	568 (5)	568 (5) 890 (8)
CLIDEF	4	233 (2)	233 (2)
ERRSUP_S	1	570 (5)	1318 (10) 570 (5) 823 (7)
MODNAM	1	416 (2)	416 (2)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ACBL	00F1	1512 (10)
ADDB3	0081	1104 (10) 1114 (10) 1225 (10) 1437 (10) 1438 (10) 1448 (10) 1457 (10)
		1469 (10) 1480 (10) 1487 (10) 1511 (10) 1516 (10) 1517 (10) 630 (6)
		638 (6) 650 (6) 662 (6) 669 (6) 722 (6) 723 (6) 731 (6)
		732 (6) 745 (6) 746 (6) 758 (6) 759 (6) 768 (6) 769 (6)
		833 (7) 960 (9) 970 (9)
ADDL2	00C0	1006 (9) 1441 (10)
ADDL3	00C1	1011 (9) 699 (6)
BEQL	0013	1043 (10) 1083 (10) 1205 (10) 1268 (10) 1270 (10) 1272 (10) 1298 (10)
		1301 (10) 1304 (10) 1328 (10) 1330 (10) 1393 (10) 1434 (10) 1447 (10)
		1456 (10) 1468 (10) 1510 (10) 480 (3) 482 (3) 554 (5) 556 (5)
		558 (5) 560 (5) 629 (6) 637 (6) 649 (6) 661 (6) 721 (6)
		730 (6) 744 (6) 856 (8) 858 (8) 860 (8) 886 (8) 939 (9)
BGEQ	0018	705 (6) 910 (9)
BGTR	0014	1107 (10) 1228 (10) 1490 (10) 672 (6) 773 (6) 836 (7) 963 (9)
BICL2	00CA	1135 (10) 1161 (10)
BICW	00AA	1050 (10) 905 (9)
BICW3	00AB	709 (6)
BISB2	0088	1045 (10) 1450 (10) 1459 (10) 1471 (10) 1482 (10) 631 (6) 640 (6)
		652 (6) 663 (6) 696 (6) 725 (6) 734 (6) 747 (6) 761 (6)
BISL3	00C9	1155 (10) 674 (6) 799 (7) 883 (8)
BITW	00B3	1042 (10)
BLBC	00E9	1053 (10) 1097 (10) 1164 (10) 1219 (10) 1235 (10) 1322 (10) 1461 (10)
		1473 (10) 1484 (10) 552 (5) 642 (6) 654 (6) 666 (6) 736 (6)
		750 (6) 763 (6) 873 (8) 876 (8) 879 (8) 882 (8) 908 (9)
		953 (9) 973 (9)
BLBS	00E8	1062 (10) 1089 (10) 1173 (10) 1210 (10) 1275 (10) 1347 (10) 1413 (10)
		1423 (10) 597 (6) 612 (6) 683 (6) 690 (6) 778 (6) 812 (7)
		918 (9) 945 (9)
BLEQ	0015	820 (7)
BLSS	0019	1055 (10) 1166 (10) 818 (7)
BLSSU	001F	1453 (10) 1465 (10) 1477 (10) 634 (6) 646 (6) 658 (6) 727 (6)
		741 (6) 755 (6)
BNEQ	0012	1291 (10) 1306 (10) 1311 (10) 1316 (10) 1332 (10) 1336 (10) 1395 (10)
		1431 (10) 1497 (10) 1499 (10) 562 (5) 566 (5) 591 (6) 822 (7)
		868 (8) 888 (8)
BRB	0011	1240 (10) 1309 (10) 1319 (10) 1402 (10) 1460 (10) 1472 (10) 1483 (10)
		1504 (10) 641 (6) 653 (6) 665 (6) 710 (6) 735 (6) 749 (6)
		762 (6) 824 (7)
BRW	0031	1065 (10) 1090 (10) 1115 (10) 1177 (10) 1212 (10) 1249 (10) 1276 (10)
		1292 (10) 1333 (10) 1337 (10) 1351 (10) 1415 (10) 1424 (10) 1525 (10)
		563 (5) 567 (5) 684 (6) 778 (6) 841 (7) 889 (8) 921 (9)
		946 (9)
BSBW	0030	1061 (10) 1096 (10) 1172 (10) 1218 (10) 1346 (10) 1422 (10) 611 (6)
		689 (6) 790 (6) 810 (7) 845 (7) 917 (9) 952 (9) 984 (9)
CALLG	00FA	1052 (10) 1088 (10) 1163 (10) 1209 (10) 1274 (10) 1412 (10) 682 (6)
		777 (6) 872 (8) 875 (8) 878 (8) 881 (8) 907 (9) 944 (9)
		972 (9)
CALLS	00FB	1023 (9) 1040 (10) 1048 (10) 1134 (10) 1159 (10) 1289 (10) 1318 (10)
		1321 (10) 1324 (10) 489 (3) 551 (5) 570 (5) 589 (6) 595 (6)

```
                 599  (6)   603  (6)   823  (7)   903  (9)
CASEL    00CF    569  (5)   891  (8)
CLRB     0094    596  (6)
CLRL     00D4   1074 (10)  1109 (10)  1110 (10)  1145 (10)  1197 (10)  1230 (10)  1231 (10)
                1389 (10)  1445 (10)  1454 (10)  1466 (10)  1478 (10)  1492 (10)  1493 (10)
                1514 (10)   627  (6)   635  (6)   647  (6)   659  (6)   708  (6)   719  (6)
                 728  (6)   742  (6)   756  (6)   775  (6)   839  (7)   930  (9)   965  (9)
                 966  (9)
CLRQ     007C   1020  (9)  1131 (10)  1283 (10)  1288 (10)   588  (6)
CMPB     0091   1464 (10)  1476 (10)   481  (3)   645  (6)   657  (6)   740  (6)   754  (6)
CMPL     00D1   1267 (10)  1269 (10)  1271 (10)  1297 (10)  1300 (10)  1303 (10)  1305 (10)
                1310 (10)  1327 (10)  1329 (10)  1331 (10)  1335 (10)  1392 (10)  1394 (10)
                1430 (10)  1496 (10)   553  (5)   555  (5)   557  (5)   559  (5)   561  (5)
                 565  (5)   819  (7)   855  (8)   857  (8)   859  (8)   867  (8)   885  (8)
                 887  (8)
CMPW     00B1   1106 (10)  1227 (10)  1452 (10)  1489 (10)   633  (6)   671  (6)   726  (6)
                 772  (6)   835  (7)   962  (9)
DECL     00D7    785  (6)   980  (9)
EDIV     007B   1446 (10)  1455 (10)  1467 (10)  1479 (10)  1515 (10)   628  (6)   636  (6)
                 648  (6)   660  (6)   720  (6)   729  (6)   743  (6)   757  (6)
EXTZV    00EF   1056 (10)  1078 (10)  1084 (10)  1167 (10)  1206 (10)  1387 (10)  1433 (10)
                1435 (10)  1436 (10)  1509 (10)  1513 (10)   803  (7)   911  (9)   934  (9)
                 940  (9)
INCB     0096    594  (6)
INCL     00D6   1449 (10)  1458 (10)  1463 (10)  1470 (10)  1475 (10)  1481 (10)  1486 (10)
                1506 (10)   632  (6)   639  (6)   644  (6)   651  (6)   656  (6)   664  (6)
                 668  (6)   724  (6)   733  (6)   739  (6)   748  (6)   753  (6)   760  (6)
                 766  (6)
INSV     00F0   1038 (10)  1146 (10)   800  (7)
JSB      0016   1007  (9)   477  (3)
MOVAB    009E   1070 (10)  1101 (10)  1144 (10)  1193 (10)  1222 (10)  1234 (10)  1296 (10)
                1313 (10)  1383 (10)  1427 (10)  1505 (10)   547  (5)   619  (6)   620  (6)
                 700  (6)   789  (6)   830  (7)   831  (7)   864  (8)   869  (8)   926  (9)
                 957  (9)   983  (9)
MOVAL    00DE   1285 (10)  1295 (10)  1299 (10)  1302 (10)  1307 (10)  1308 (10)  1312 (10)
                 585  (6)   703  (6)   780  (6)   975  (9)
MOVB     0090   1086 (10)  1108 (10)  1208 (10)  1229 (10)  1245 (10)  1410 (10)  1411 (10)
                1440 (10)  1444 (10)  1462 (10)  1474 (10)  1485 (10)  1491 (10)  1519 (10)
                 616  (6)   623  (6)   626  (6)   643  (6)   655  (6)   667  (6)   673  (6)
                 680  (6)   693  (6)   715  (6)   737  (6)   738  (6)   751  (6)   752  (6)
                 764  (6)   765  (6)   774  (6)   783  (6)   787  (6)   826  (7)   837  (7)
                 942  (9)   964  (9)   978  (9)   981  (9)
MOVC5    002C   1009  (9)
MOVL     00D0   1014  (9)  1037 (10)  1051 (10)  1069 (10)  1071 (10)  1072 (10)  1073 (10)
                1075 (10)  1077 (10)  1092 (10)  1100 (10)  1102 (10)  1103 (10)  1111 (10)
                1112 (10)  1151 (10)  1160 (10)  1162 (10)  1192 (10)  1194 (10)  1195 (10)
                1196 (10)  1198 (10)  1201 (10)  1214 (10)  1221 (10)  1223 (10)  1224 (10)
                1232 (10)  1239 (10)  1244 (10)  1273 (10)  1293 (10)  1325 (10)  1326 (10)
                1382 (10)  1384 (10)  1385 (10)  1386 (10)  1390 (10)  1399 (10)  1401 (10)
                1406 (10)  1417 (10)  1426 (10)  1428 (10)  1429 (10)  1432 (10)  1439 (10)
                1442 (10)  1494 (10)  1502 (10)  1507 (10)  1508 (10)   479  (3)   571  (5)
                 593  (6)   607  (6)   618  (6)   621  (6)   622  (6)   624  (6)   675  (6)
                 676  (6)   678  (6)   679  (6)   685  (6)   694  (6)   695  (6)   701  (6)
                 706  (6)   707  (6)   712  (6)   713  (6)   714  (6)   776  (6)   779  (6)
                 781  (6)   784  (6)   786  (6)   791  (6)   801  (7)   802  (7)   805  (7)
                 827  (7)   829  (7)   832  (7)   838  (7)   840  (7)   844  (7)   847  (7)
                 861  (8)   862  (8)   874  (8)   877  (8)   880  (8)   906  (9)   925  (9)
```

```
                    927  (9)   928  (9)   929  (9)   931  (9)   933  (9)   948  (9)   956  (9)
                    958  (9)   959  (9)   967  (9)   968  (9)   974  (9)   976  (9)   979  (9)
                    985  (9)
MOVQ        007D    617  (6)   821  (7)   871  (8)
MOVW        00B0   1033 (10)  1039 (10)  1046 (10)  1067 (10)  1076 (10)  1081 (10)  1098 (10)
                   1105 (10)  1191 (10)  1200 (10)  1203 (10)  1220 (10)  1226 (10)  1381 (10)
                   1397 (10)  1404 (10)  1409 (10)  1425 (10)  1488 (10)  1500 (10)   615  (6)
                    670  (6)   692  (6)   702  (6)   771  (6)   815  (7)   834  (7)   899  (9)
                    900  (9)   901  (9)   923  (9)   932  (9)   937  (9)   954  (9)   961  (9)
MOVZBL      009A    601  (6)   677  (6)
MOVZWL      003C   1049 (10)  1058 (10)  1093 (10)  1168 (10)  1215 (10)  1343 (10)  1418 (10)
                    697  (6)   904  (9)   913  (9)   949  (9)
POPL        8ED0    811  (7)
POPR        00BA   1010  (9)  1015  (9)  1024  (9)  1136 (10)
PUSHAB      009F   1323 (10)   598  (6)   602  (6)   866  (8)
PUSHAL      00DF   1318 (10)   548  (5)   549  (5)   570  (5)   823  (7)
PUSHAQ      007F    865  (8)
PUSHL       00DD   1021  (9)  1022  (9)  1040 (10)  1048 (10)  1132 (10)  1133 (10)  1159 (10)
                   1282 (10)  1284 (10)  1286 (10)  1287 (10)  1318 (10)   483  (3)   484  (3)
                    485  (3)   486  (3)   487  (3)   488  (3)   550  (5)   570  (5)   582  (6)
                    583  (6)   584  (6)   586  (6)   587  (6)   681  (6)   823  (7)   870  (8)
                    903  (9)
PUSHR       00BB   1005  (9)  1008  (9)  1019  (9)  1130 (10)
REI         0002   1025  (9)  1137 (10)
RET         0004   1064 (10)  1175 (10)  1349 (10)   573  (5)   592  (6)   600  (6)   613  (6)
                    691  (6)   792  (6)   813  (7)   849  (7)   920  (9)   986  (9)
RSB         0005   1016  (9)   491  (3)
SOBGTR      00F5    788  (6)   982  (9)
SUBL        00C2    817  (7)
SUBL2       00C2   1518 (10)
SUBL3       00C3   1012  (9)  1013  (9)   698  (6)
TSTL        00D5   1290 (10)  1315 (10)  1498 (10)   590  (6)
TSTW        00B5   1054 (10)  1165 (10)   704  (6)   909  (9)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	416	(2)														
.ASCII	418	(2)	419	(2)	420	(2)	422	(2)	424	(2)	425	(2)	427	(2)	428	(2)
.BLKB	542	(4)														
.BLKL	233	(2)														
.BYTE	432	(2)	437	(2)												
.DSABL	20	(1)	987	(9)												
.ENABL	894	(9)														
.END	1528	(10)														
.ENDC	1040	(10)	1048	(10)	1159	(10)	1318	(10)	262	(2)	265	(2)	266	(2)	542	(4)
	570	(5)	823	(7)	903	(9)										
.ENDM	233	(2)	234	(2)	235	(2)	236	(2)	241	(2)	242	(2)	243	(2)	244	(2)
	245	(2)	246	(2)	247	(2)	248	(2)	249	(2)	250	(2)	251	(2)	252	(2)
	253	(2)	254	(2)	255	(2)	261	(2)	262	(2)	263	(2)	265	(2)	266	(2)
	416	(2)	538	(4)	542	(4)	568	(5)	570	(5)	778	(6)	903	(9)		
.ENDR	262	(2)	265	(2)	266	(2)	542	(4)	569	(5)	891	(8)				
.ENTRY	545	(5)														
.EXTRN	395	(2)	396	(2)	397	(2)	398	(2)	399	(2)	400	(2)	401	(2)	405	(2)
	406	(2)	407	(2)	408	(2)										
.IDENT	18	(1)														
.IF	1040	(10)	1048	(10)	1159	(10)	1318	(10)	262	(2)	265	(2)	266	(2)	542	(4)
	570	(5)	823	(7)	903	(9)										
.IFF	1040	(10)	1048	(10)	1159	(10)	1318	(10)	262	(2)	265	(2)	266	(2)	542	(4)
	570	(5)	823	(7)	903	(9)										
.IIF	1318	(10)	262	(2)	265	(2)	266	(2)	570	(5)	823	(7)				
.IRP	262	(2)	265	(2)	266	(2)	542	(4)	569	(5)	891	(8)				
.LIBRARY	222	(2)	223	(2)	224	(2)										
.LIST	233	(2)	261	(2)	493	(4)	542	(4)	543	(4)						
.LONG	426	(2)														
.MACRO	262	(2)	265	(2)	266	(2)										
.MDELETE	262	(2)	265	(2)	266	(2)										
.NLIST	233	(2)	261	(2)	537	(4)	542	(4)								
.NOCROSS	234	(2)	235	(2)	236	(2)	241	(2)	242	(2)	243	(2)	244	(2)	245	(2)
	246	(2)	247	(2)	248	(2)	249	(2)	250	(2)	251	(2)	252	(2)	253	(2)
	254	(2)	255	(2)	261	(2)	263	(2)	308	(2)						
.NOSHOW	19	(1)														
.PAGE	1117	(10)	1251	(10)	988	(9)										
.PSECT	233	(2)	415	(2)	435	(2)	441	(3)	542	(4)						
.QUAD	421	(2)	423	(2)												
.RESTORE	233	(2)	234	(2)	235	(2)	236	(2)	241	(2)	242	(2)	243	(2)	244	(2)
	245	(2)	246	(2)	247	(2)	248	(2)	249	(2)	250	(2)	251	(2)	252	(2)
	253	(2)	254	(2)	255	(2)	261	(2)	263	(2)	384	(2)	542	(4)		
.SAVE	233	(2)	234	(2)	235	(2)	236	(2)	241	(2)	242	(2)	243	(2)	244	(2)
	245	(2)	246	(2)	247	(2)	248	(2)	249	(2)	250	(2)	251	(2)	252	(2)
	253	(2)	254	(2)	255	(2)	261	(2)	263	(2)	308	(2)	542	(4)		
.SBTTL	1118	(10)	440	(3)	494	(4)										
.SIGNED_WORD	569	(5)	891	(8)												
.SUBTITLE	1252	(10)	989	(9)												
.TITLE	17	(1)														
.WEAK	402	(2)	403	(2)	404	(2)	411	(2)	412	(2)						

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	73	#-1038 (10) #-1104 (10) #-1105 (10) #-1106 (10) #-1108 (10) #-1146 (10)
		#-1225 (10) #-1226 (10) #-1227 (10) #-1229 (10) #-1267 (10) #-1269 (10)
		#-1271 (10) #-1282 (10) #-1284 (10) #-1297 (10) #-1300 (10) #-1303 (10)
		#-1305 (10) #-1310 (10) #-1327 (10) #-1329 (10) #-1331 (10) #-1335 (10)
		#-1430 (10) #-1442 (10) #-1476 (10) #-1488 (10) #-1489 (10) #-1491 (10)
		#-1496 (10) #-550 (5) #-553 (5) #-555 (5) #-557 (5) #-559 (5)
		#-561 (5) #-565 (5) #-569 (5) #-582 (6) #-584 (6) #-601 (6)
		#-624 (6) #-670 (6) #-671 (6) #-673 (6) #-674 (6) #-680 (6)
		#-704 (6) #-709 (6) #-712 (6) #-771 (6) #-772 (6) #-774 (6)
		#-799 (7) #-800 (7) #-833 (7) #-834 (7) #-835 (7) #-837 (7)
		#-855 (8) #-857 (8) #-859 (8) #-867 (8) #-883 (8) #-885 (8)
		#-887 (8) #-891 (8) #-900 (9) #-960 (9) #-961 (9) #-962 (9)
		#-964 (9)
FP	29	#-1075 (10) #-1081 (10) 1085 (10) #-1092 (10) #-1111 (10) #-1198 (10)
		#-1203 (10) 1207 (10) #-1214 (10) #-1232 (10) 1285 (10) #-1390 (10)
		#-1417 (10) #-1439 (10) #-1494 (10) 547 (5) 548 (5) 549 (5)
		585 (6) #-622 (6) #-676 (6) #-840 (7) #-862 (8) 865 (8)
		#-931 (9) #-937 (9) 941 (9) #-948 (9) #-967 (9)
R0	163	#-1008 (9) #-1010 (9) #-1019 (9) #-1024 (9) #-1053 (10) #-1062 (10)
		#-1070 (10) #-1071 (10) #-1072 (10) #-1085 (10) #-1086 (10) #-1089 (10)
		#-1097 (10) #-1101 (10) #-1102 (10) #-1103 (10) #-1104 (10) #-1130 (10)
		#-1136 (10) #-1164 (10) #-1173 (10) #-1193 (10) #-1194 (10) #-1195 (10)
		#-1207 (10) #-1208 (10) #-1210 (10) #-1219 (10) #-1222 (10) #-1223 (10)
		#-1224 (10) #-1225 (10) #-1234 (10) #-1239 (10) #-1244 (10) #-1245 (10)
		#-1275 (10) #-1285 (10) #-1286 (10) #-1290 (10) #-1293 (10) #-1295 (10)
		#-1299 (10) #-1302 (10) #-1308 (10) #-1312 (10) #-1315 (10) 1321 (10)
		#-1322 (10) #-1347 (10) #-1383 (10) #-1384 (10) #-1385 (10) #-1413 (10)
		#-1423 (10) #-1427 (10) #-1428 (10) #-1429 (10) #-1432 (10) #-1437 (10)
		#-1438 (10) #-1440 (10) #-1441 (10) #-1448 (10) #-1457 (10) #-1462 (10)
		#-1469 (10) #-1474 (10) #-1480 (10) #-1485 (10) #-1487 (10) #-1505 (10)
		#-1506 (10) #-1507 (10) #-1511 (10) #-1516 (10) #-1517 (10) #-1518 (10)
		#-1519 (10) #-479 (3) #-488 (3) #-552 (5) #-571 (5) #-585 (6)
		#-586 (6) #-590 (6) #-593 (6) #-597 (6) #-612 (6) #-619 (6)
		620 (6) #-621 (6) #-623 (6) #-630 (6) #-638 (6) #-643 (6)
		#-650 (6) #-655 (6) #-662 (6) #-667 (6) #-669 (6) #-683 (6)
		#-690 (6) #-697 (6) #-698 (6) #-699 (6) 700 (6) #-701 (6)
		#-702 (6) #-706 (6) #-713 (6) #-722 (6) #-731 (6) #-737 (6)
		#-745 (6) #-751 (6) #-758 (6) #-764 (6) #-768 (6) #-778 (6)
		#-779 (6) #-781 (6) #-784 (6) #-786 (6) #-787 (6) #-791 (6)
		#-802 (7) 804 (7) #-812 (7) #-821 (7) #-827 (7) #-830 (7)
		831 (7) #-832 (7) #-833 (7) #-844 (7) #-847 (7) #-873 (8)
		#-876 (8) #-879 (8) #-882 (8) #-908 (9) #-918 (9) #-926 (9)
		#-927 (9) #-928 (9) #-941 (9) #-942 (9) #-945 (9) #-953 (9)
		#-957 (9) #-958 (9) #-959 (9) #-960 (9) #-973 (9) #-974 (9)
		#-976 (9) #-979 (9) #-981 (9) #-985 (9)
R1	36	#-1006 (9) #-1008 (9) #-1009 (9) #-1010 (9) #-1011 (9) #-1019 (9)
		#-1024 (9) #-1058 (10) #-1067 (10) #-1093 (10) #-1098 (10) #-1130 (10)
		#-1136 (10) #-1168 (10) #-1191 (10) #-1215 (10) #-1220 (10) #-1343 (10)
		#-1381 (10) #-1418 (10) #-1425 (10) #-1507 (10) #-1518 (10) #-1519 (10)
		#-601 (6) #-607 (6) #-615 (6) #-685 (6) #-692 (6) #-805 (7)

```
                #-815   (7)     #-832   (7)     #-913   (9)     #-923   (9)     #-949   (9)     #-954   (9)
R10          5  #-1005  (9)     #-1012  (9)     #-1013  (9)     #-1014  (9)     #-1015  (9)
R11         12  #-1011  (9)     #-1012  (9)     #-1013  (9)     #-1014  (9)     #-1105  (10)    #-1226  (10)
                #-1488  (10)      545   (5)     #-670   (6)     #-771   (6)     #-834   (7)     #-961   (9)
R2         188  #-1008  (9)       1009  (9)     #-1010  (9)     #-1011  (9)     #-1037  (10)      1038  (10)
                #-1039  (10)    #-1045  (10)    #-1046  (10)    #-1067  (10)    #-1069  (10)      1070  (10)
                #-1071  (10)    #-1073  (10)    #-1074  (10)    #-1075  (10)    #-1076  (10)    #-1077  (10)
                  1078  (10)    #-1079  (10)    #-1082  (10)    #-1086  (10)      1088  (10)    #-1092  (10)
                #-1098  (10)    #-1100  (10)      1101  (10)    #-1102  (10)    #-1108  (10)    #-1109  (10)
                #-1110  (10)    #-1111  (10)    #-1113  (10)    #-1114  (10)    #-1145  (10)      1146  (10)
                #-1155  (10)    #-1191  (10)    #-1192  (10)      1193  (10)    #-1194  (10)    #-1196  (10)
                #-1197  (10)    #-1199  (10)    #-1200  (10)    #-1202  (10)    #-1204  (10)    #-1208  (10)
                  1209  (10)    #-1214  (10)    #-1220  (10)    #-1221  (10)      1222  (10)    #-1223  (10)
                #-1229  (10)    #-1230  (10)    #-1231  (10)    #-1233  (10)      1234  (10)    #-1296  (10)
                #-1313  (10)    #-1325  (10)    #-1381  (10)    #-1382  (10)      1383  (10)    #-1384  (10)
                #-1386  (10)    #-1388  (10)    #-1389  (10)    #-1391  (10)    #-1398  (10)    #-1400  (10)
                #-1401  (10)    #-1405  (10)    #-1407  (10)    #-1409  (10)    #-1410  (10)    #-1411  (10)
                  1412  (10)    #-1417  (10)    #-1425  (10)    #-1426  (10)      1427  (10)    #-1428  (10)
                #-1449  (10)    #-1458  (10)    #-1463  (10)    #-1470  (10)    #-1475  (10)    #-1481  (10)
                #-1486  (10)    #-1491  (10)    #-1492  (10)    #-1493  (10)    #-1495  (10)    #-1501  (10)
                #-1503  (10)      1505  (10)      545   (5)     #-615   (6)     #-616   (6)     #-617   (6)
                #-618   (6)       619   (6)     #-620   (6)     #-632   (6)     #-639   (6)     #-644   (6)
                #-651   (6)     #-656   (6)     #-664   (6)     #-668   (6)     #-673   (6)     #-674   (6)
                #-675   (6)     #-676   (6)     #-677   (6)     #-678   (6)     #-679   (6)     #-680   (6)
                #-681   (6)       682   (6)     #-692   (6)     #-693   (6)     #-694   (6)     #-695   (6)
                #-696   (6)     #-697   (6)     #-699   (6)     #-700   (6)       703   (6)     #-724   (6)
                #-733   (6)     #-739   (6)     #-748   (6)     #-753   (6)     #-760   (6)     #-766   (6)
                #-774   (6)     #-775   (6)     #-776   (6)       777   (6)     #-779   (6)     #-780   (6)
                #-782   (6)     #-789   (6)     #-815   (7)     #-826   (7)     #-827   (7)     #-829   (7)
                  830   (7)     #-831   (7)     #-837   (7)     #-838   (7)     #-839   (7)     #-840   (7)
                #-844   (7)     #-923   (9)     #-925   (9)       926   (9)     #-927   (9)     #-929   (9)
                #-930   (9)     #-931   (9)     #-932   (9)     #-933   (9)       934   (9)     #-935   (9)
                #-938   (9)     #-942   (9)       944   (9)     #-948   (9)     #-954   (9)     #-956   (9)
                  957   (9)     #-958   (9)     #-964   (9)     #-965   (9)     #-966   (9)     #-967   (9)
                #-969   (9)     #-970   (9)       972   (9)     #-974   (9)     #-975   (9)     #-977   (9)
                #-983   (9)
R3         76   #-1008  (9)     #-1010  (9)       1052  (10)      1163  (10)      1274  (10)    #-1433  (10)
                  1435  (10)      1436  (10)    #-1437  (10)    #-1439  (10)    #-1440  (10)    #-1442  (10)
                #-1446  (10)    #-1448  (10)    #-1455  (10)    #-1457  (10)    #-1467  (10)    #-1469  (10)
                #-1479  (10)    #-1480  (10)    #-1508  (10)    #-1509  (10)    #-1512  (10)    #-1513  (10)
                #-1515  (10)    #-1516  (10)      545   (5)     #-622   (6)     #-623   (6)     #-624   (6)
                #-628   (6)     #-630   (6)     #-636   (6)     #-638   (6)     #-648   (6)     #-650   (6)
                #-660   (6)     #-662   (6)     #-708   (6)     #-709   (6)     #-712   (6)     #-720   (6)
                #-722   (6)     #-723   (6)     #-729   (6)     #-731   (6)     #-732   (6)     #-743   (6)
                #-745   (6)     #-746   (6)     #-757   (6)     #-758   (6)     #-759   (6)     #-784   (6)
                #-785   (6)     #-787   (6)     #-788   (6)     #-811   (7)     #-817   (7)     #-819   (7)
                #-821   (7)     #-864   (8)       872   (8)       875   (8)       878   (8)       881   (8)
                  907   (9)     #-979   (9)     #-980   (9)     #-981   (9)     #-982   (9)
R4         61   #-1008  (9)     #-1010  (9)     #-1049  (10)    #-1054  (10)      1056  (10)    #-1114  (10)
                #-1160  (10)    #-1165  (10)      1167  (10)    #-1235  (10)    #-1435  (10)    #-1438  (10)
                #-1445  (10)    #-1446  (10)    #-1452  (10)    #-1455  (10)    #-1464  (10)    #-1467  (10)
                #-1479  (10)    #-1487  (10)    #-1509  (10)    #-1511  (10)    #-1514  (10)    #-1515  (10)
                #-1517  (10)      545   (5)     #-627   (6)     #-628   (6)     #-633   (6)     #-636   (6)
                #-645   (6)     #-648   (6)     #-657   (6)     #-660   (6)     #-669   (6)     #-719   (6)
                #-720   (6)     #-726   (6)     #-729   (6)     #-740   (6)     #-743   (6)     #-754   (6)
                #-757   (6)     #-768   (6)     #-769   (6)     #-904   (9)     #-909   (9)       911   (9)
                #-970   (9)
```

```
R5              42      #-1008  (9)     #-1010  (9)     #-1034  (10)    #-1039  (10)    #-1042  (10)    #-1046  (10)
                        #-1049  (10)    #-1050  (10)    #-1073  (10)    #-1135  (10)    #-1144  (10)    #-1154  (10)
                        #-1158  (10)    #-1160  (10)    #-1161  (10)    #-1196  (10)    #-1287  (10)    #-1386  (10)
                          1387  (10)    #-1454  (10)    #-1466  (10)    #-1478  (10)      545   (5)     #-587   (6)
                        #-635   (6)     #-647   (6)     #-659   (6)     #-728   (6)     #-742   (6)     #-756   (6)
                        #-799   (7)       800   (7)     #-801   (7)     #-802   (7)     #-838   (7)     #-883   (8)
                        #-899   (9)     #-900   (9)     #-902   (9)     #-904   (9)     #-905   (9)     #-929   (9)
R6              11      #-703   (6)     #-707   (6)     #-714   (6)     #-723   (6)     #-732   (6)     #-738   (6)
                        #-746   (6)     #-752   (6)     #-759   (6)     #-765   (6)     #-769   (6)
R9              16      #-1293  (10)    #-1307  (10)      1323  (10)    #-1325  (10)    #-1326  (10)    #-1392  (10)
                        #-1394  (10)      1433  (10)    #-1498  (10)      1509  (10)      1513  (10)      545   (5)
                        #-593   (6)       598   (6)       602   (6)
SP              32        1009  (9)     #-1020  (9)     #-1051  (10)      1052  (10)    #-1131  (10)    #-1162  (10)
                          1163  (10)    #-1273  (10)      1274  (10)    #-1283  (10)    #-1288  (10)    #-547   (5)
                        #-588   (6)     #-776   (6)     #-783   (6)     #-787   (6)       789   (6)     #-804   (7)
                        #-869   (8)     #-871   (8)       872   (8)     #-874   (8)       875   (8)     #-877   (8)
                          878   (8)     #-880   (8)       881   (8)     #-906   (9)       907   (9)     #-978   (9)
                        #-981   (9)       983   (9)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	51	00:00:00.04	00:00:00.30
Command processing	900	00:00:00.27	00:00:01.11
Pass 1	1101	00:00:05.12	00:00:09.00
Symbol table sort	0	00:00:00.42	00:00:00.51
Pass 2	28	00:00:01.33	00:00:02.70
Symbol table output	0	00:00:00.05	00:00:00.13
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	6	00:00:00.54	00:00:00.83
Assembler run totals	2086	00:00:07.77	00:00:14.58

The working set limit was 4012 pages.
255385 bytes (499 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1498 non-local and 95 local symbols.
1528 source lines were read in Pass 1, producing 106 object records in Pass 2.
158 pages of virtual memory were used to define 56 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	21
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	4
SYS$COMMON:[SYSLIB]LIB.MLB;2	2
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;9	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	38

1564 GETS were required to define 38 macros.

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 5 Frame L16          Sequence 1030
CONFIG                        *** CONFIG Configure load device   17-DEC-1987 14:09:38  VAX/VMS Macro V04-00        Page  55
VAX-11 Macro Run Statistics                                      16-DEC-1987 09:23:27  CONFIG.MAR;3                      (10)

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CONFIG.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 219 Libraries, Equated Symbols, Macros
(1) 266 Work Psect Declarations ; [13]
(1) 282 Data Psect Declarations ; [13]
(1) 467 VRSetWidth Routine - Set terminal width ; [12]
(1) 557 VRShowWidth Routine - Show terminal width ; [12]
(1) 606 KB_Poll Routine - Keyboard ready bit check
(1) 675 RInput Routine - Flag driven terminal input
(1) 790 RType Routine - Retype input line ('^R' function)
(1) 831 Out Routines - Flag driven terminal output
(1) 949 WriteVBlk Routine - Console write driver
(1) 997 ReadVBlk Routine - Console read driver
(1) 1225 SIZE_TERM Routine to size the console terminal

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CONSOLE.MAR
0000     2 ; *6    12-SEP-1986 12:40:34 SILVER "<no reason given>"
0000     3 ; *5    11-SEP-1986 10:56:45 SILVER "<no reason given>"
0000     4 ; *4    30-MAY-1986 16:47:26 BAGGETT "NO CHANGES"
000C     5 ; *3    28-MAY-1986 10:51:53 SALEM "DONE"
0000     6 ; *2    27-MAY-1986 15:12:51 SALEM "ALL DONE"
0000     7 ; *1     6-FEB-1986 15:15:19 DS ""
0000     8 ; DEC/CMS REPLACEMENT HISTORY, Element CONSOLE.MAR
0000     9         .TITLE  CONSOLE *** CONSOLE Console I/O handler
0000    10         .IDENT  /09-35/                                                ;G:2
0000    11         .NoShow Conditionals    ;                                        [16]
0000    12
0000    13 ;++
0000    14 ; COPYRIGHT (C) 1977, 1983, 1985
0000    15 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    16 ;
0000    17 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    18 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    19 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    20 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    21 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    22 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    23 ; REMAIN IN DEC.
0000    24 ;
0000    25 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    26 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    27 ; CORPORATION.
0000    28 ;
0000    29 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    30 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    31 ;--
```

```
0000     33 ;++
0000     34 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000     35 ;
0000     36 ; ABSTRACT:
0000     37 ;
0000     38 ;       This routine contains the interface with the LSI-11 console.
0000     39 ;       It provides the routines necessary to read and write characters
0000     40 ;       and strings on the operators console.  It contains the limited
0000     41 ;       functionality to interface with QIO.
0000     42 ;
0000     43 ; ENVIRONMENT:
0000     44 ;
0000     45 ;       Operates as part of the Supervisor only in stand-alone.
0000     46 ;       Portions hereof must operate in KERNEL mode.
0000     47 ;
0000     48 ; AUTHOR:
0000     49 ;
0000     50 ;       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000     51 ;
0000     52 ; MODIFICATIONS:
0000     53 ;
0000     54 ;       01      NICK HOWGATE    30-DEC-77       VERSION 02.
0000     55 ;               CMK$TCONSOLE - TURN OFF ALL OUTPUT IF APT REQUESTS
0000     56 ;               NO REPORT (DSA$GL_FLAGS)=DSA$V_NORPT
0000     57 ;
0000     58 ;       02      Roger Riggs     25-JUL-78
0000     59 ;               CMK$TCONSOLE looks also at DS$V_CTRLO in DS$GL_FLAGS
0000     60 ;               MAKE ^O WORK.
0000     61 ;       03      Made unsolicited input is echoed as bells instead of the
0000     62 ;               characters typed.
0000     63 ;
0000     64 ;       04      NICK HOWGATE    5-FEB-79        (ESSAA-5.00)
0000     65 ;               CONVERT TABS TO SPACES
0000     66 ;
0000     67 ;       05      Implement P4 buffer in WRITEBLK
0000     68 ;
0000     69 ;       06      Fix IO$_READPROMPT to always prompt on new line
0000     70 ;
0000     71 ;       07      Defer echo of ^C until KB_CHECK but suppress rest of output
0000     72 ;               until ^C is serviced
0000     73 ;
0000     74 ;       08      MARION BAGGETT  NOV 12,1979
0000     75 ;               ADD THE CONTROL S AND CONTROL Q FUNCTION FOR STARTING AND
0000     76 ;               STOPPING THE PRINTOUT ON THE TERMINAL.
0000     77 ;
0000     78 ;       09      Roger Riggs, 14-Jan-1980, Version 5.2
0000     79 ;               Added checking of IO$M_CANCTRLO  on write to terminal.
0000     80 ;               Used by exception to make sure exceptions always print.
0000     81 ;               Also fixed RINPUT to ignore character if bit 15 (ERR) is
0000     82 ;               Set in the data register.
0000     83 ;
0000     84 ;       10      Dave Butenhof, 17-oct-1980, version 6.1
0000     85 ;               Cause re-type of input line ('^R') after asynchronous output
0000     86 ;               occurs, in the fashion of VMS.  Also, remove hyphen line
0000     87 ;               continuation code, since it doesn't belong here.
0000     88 ;
0000     89 ;       11      To fully support ^C 'disable' in tight time-critical code,
```

```
0000     90 ;                cut down on instruction stream for control characters.
0000     91 ;                Specifically, test for ^C first in RINPUT instead of last.
0000     92 ;                Also, make new routine OUT_CNTRL to output ^C and ^O echo;
0000     93 ;                without all the overhead of doing extra KB_POLL each
0000     94 ;                character (small messages), or checking for tabs, etc.
0000     95 ;
0000     96 ;        12      Dave Butenhof, 17-mar-1981, version 6.3
0000     97 ;                Add VRSETWIDTH routine, and hooks for SET WIDTH command.
0000     98 ;                This allows users (FS in particular) to avoid destroying
0000     99 ;                80 column printer paper with output from DS. Also add
0000    100 ;                VRSHOWWIDTH routine.
0000    101 ;
0000    102 ;        13      - Jack Stansbury, 22-Oct-1981, Version 6.-
0000    103 ;                Changed SEP Psect to CODE, DATA, and WORK.
0000    104 ;                Also added .LIBRARY statements for $DS and $DIAG.
0000    105 ;
0000    106 ;        14      - Jack Stansbury, 26-Oct-1981, Version 6.-
0000    107 ;                Fixed more truncation errors....
0000    108 ;
0000    109 ;        15      - Jack Stansbury, 25-Jan-1982, Version 6.6
0000    110 ;                Changed the WIDTH variable to make it global, so that QA can
0000    111 ;                access it.
0000    112 ;
0000    113 ;        16      - Jack Stansbury, 23-Mar-1982, Version 6.?
0000    114 ;                Added typecoding to the output messages.
0000    115 ;
0000    116 ;        17      - Richard Brown, 8-June-82, Version 6.8
0000    117 ;                Addition of support for specification of
0000    118 ;                terminators other than carriage-return.  We will
0000    119 ;                now be functionally equivalent to VMS QIO calls
0000    120 ;                with respect to the optional terminator specification
0000    121 ;                (argument P4) for the terminal driver.
0000    122 ;
0000    123 ;        18      Bob Bergazzi    6-Aug-82        Version 6.8
0000    124 ;                Added code to change a ^Z into an EXIT command
0000    125 ;
0000    126 ;        19      Jack Stansbury, 26-Oct-1982, Version 6.9
0000    127 ;                Added code to echo the ^C character only if the
0000    128 ;                DS$V_Ctrl_C_No_Echo bit is clear. If it is set,
0000    129 ;                then ^C's will not be echoed.
0000    130 ;
0000    131 ;        20      Bob Bergazzi    5-March-1983    Version 6.11
0000    132 ;                Added code to make SET WIDTH work online.
0000    133 ;
0000    134 ;        21      Bob Bergazzi    8-Apr-1983      Version 6.11
0000    135 ;                Removed incorrectly implemented edit 18.
0000    136 ;
0000    137 ;        22      Domenic Andella 25-Apr-83       Version 6.11
0000    138 ;                Added code in CMK$TCONSOLE routine to activate
0000    139 ;                11/XXX console terminal line.
0000    140 ;
0000    141 ;        23      Richard Brown   22-June-83      Version 6.12
0000    142 ;                Added routine SIZE_TERM, which determines
0000    143 ;                console terminal type.
0000    144 ;
0000    145 ;                Also removed code in READVBLK that treated a
0000    146 ;                question mark as a terminator, since many
```

```
                                0000    147 ;               terminal IDs contain question marks, and we
                                0000    148 ;               couldn't figure why it was a terminator anyway.
                                0000    149 ;
                                0000    150 ;       24      Bob Bergazzi    Nov. 29,1984    Version 8.0
                                0000    151 ;               Removed edit 22.
                                0000    152 ;
                                0000    153 ;       25      Jim Shelhamer   30 Nov 1984     Version 8.0
                                0000    154 ;               Added code for 11/ZZZ to handle console I/O over
                                0000    155 ;               the BI via the RXCD register.
                                0000    156 ;
                                0000    157 ;       26      Bob Bergazzi    Dec. 14, 1984   Version 8.0
                                0000    158 ;               Added support for DELETE character for video terminals.
                                0000    159 ;               Does a backspace, space, backspace sequence.
                                0000    160 ;
                                0000    161 ;       27      M. Baggett      18-Dec-1984     Version 8.0
                                0000    162 ;               Added terminals ID's for VT101, VT102, VT125 and VT200 family.
                                0000    163 ;               Undefined types are set to VT100.
                                0000    164 ;
                                0000    165 ;       28      Jim Shelhamer   6-Aug-1985      Version 8.3
                                0000    166 ;               Added check for sys$output being mailbox on
                                0000    167 ;               SET WIDTH command processing.
                                0000    168 ;
                                0000    169 ;       29      M. Baggett      14-Aug-1985     Version 8.3
                                0000    170 ;               Ignore console data bits for AURORA only.
                                0000    171 ;
                                0000    172 ;       30      Ted Salem       1-Oct-1985      Version 9.0
                                0000    173 ;               Changed RSB to RET in order to do a proper return
                                0000    174 ;               from an MP entry QIO call.
                                0000    175 ;
                                0000    176 ;       31      Jim Shelhamer   7-Oct-1985      Version 9.0
                                0000    177 ;               Added setting of bit 7 in R0 when data is recv'd
                                0000    178 ;               from RXCD to indicate data gotten.
                                0000    179 ;
                                0000    180 ;       32      M. Baggett      22-Oct-1985     Version 9.0
                                0000    181 ;               Restore changes so Ctrl z, etc work on AURORA.
                                0000    182 ;               Renumber some local labels.
                                0000    183 ;
                                0000    184 ;       33      Jim Shelhamer   29-Oct-1985     Version 9.1
                                0000    185 ;               Skip checking if char is console data for AP's.
                                0000    186 ;
                                0000    187 ;       34      Jim Shelhamer   30-Jan-1986     Version 9.1
                                0000    188 ;               Remove scorpio specific code.
                                0000    189 ;
                                0000    190 ;       35      Ted Salem       4- May-1986     Version 10.0                    ;G:2
                                0000    191 ;               Made changes to check SYSTYPE instead of SID                    ;G:2
                                0000    192 ;               for the Aurora system.                                          ;G:2
                                0000    193 ;                                                                               ;G:2
                                0000    194 ;       36      Doug Silver     8- Sep-1986     Version 10.3                    ;G:6
                                0000    195 ;               In READVBLK, decremented the buffer pointer                     ;G:5
                                0000    196 ;               by a longword displacement instead of a byte                    ;G:5
                                0000    197 ;               displacement when using a video terminal.  If                   ;G:6
                                0000    198 ;               the adress happened to end in 00, decrementing                  ;G:6
                                0000    199 ;               by 1 with a byte displacement actually                          ;G:6
                                0000    200 ;               increments by FF.                                               ;G:6
                                0000    201 ;                                                                               ;G:6
                                0000    202 ;       37      Doug Silver     12-Sep-1986     Version 10.3                    ;G:6
                                0000    203 ;               To delete a character on a video terminal you                   ;G:6
```

```
0000    204 ;               type a backspace, space, backspace.  When a                 ;G:6
0000    205 ;               character is at the very end of a line on a                 ;G:6
0000    206 ;               video terminal, the cursor remains under it                 ;G:6
0000    207 ;               instead of to the right of it.  In this                     ;G:6
0000    208 ;               situation, only a space should be typed over                ;G:6
0000    209 ;               the character.                                              ;G:6
0000    210 ;                                                                           ;G:6
0000    211 ;                                                                           ;G:5
0000    212 ;*************************************************************************
0000    213 ; NOTE : If you make any changes to this module, you should check to make
0000    214 ;        sure if a similar change should be made for NAUTILUS, VENUS, or
0000    215 ;        SCORPIO which have their own console module.
0000    216 ;*************************************************************************
0000    217 ;--
```

```
                        0000   219          .SBTTL  Libraries, Equated Symbols, Macros
                        0000   220 ;
                        0000   221 ; INCLUDE FILES:
                        0000   222 ;
                        0000   223
                        0000   224          .Library        /Sys$Library:Lib/        ;                       [16]
                        0000   225          .LIBRARY        /$DS/                   ;                       [13]
                        0000   226          .LIBRARY        /$DIAG/                 ;                       [13]
                        0000   227
                        0000   228 ;
                        0000   229 ; MACROS:
                        0000   230 ;
                        0000   231
                        0000   232          $DS_DSADEF
                        0000   233          CLIDEF                                  ; CLI data structure    [12]
                        0000   234          $PRDEF
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                00000000 FE70              .=0
                        0000               .RESTORE
                        0000   235          $QIODEF
                        0000   236          CMKDEF
                        0000   237          DSFDEF
                        0000   238          $DS_TypeDef                             ;                       [16]
                        0000   239          CRDDef                                  ; CRD Bit definitions   [19]
                        0000   240          $TTDEF                                  ; Terminal characteristics  [23]
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                00000000 FE70              .=0
                        0000               .RESTORE
                        0000   241
                        0000   242 ;
                        0000   243 ; EQUATED SYMBOLS:
                        0000   244 ;
                00000080 0000  245          PARITY  = 1@7
                00000020 0000  246          LOWCASE = 1@5
                00000003 0000  247          CNTRLC  = 3
                00000007 0000  248          BELL    = 7
                00000008 0000  249          BS      = 8                             ;                       [17]
                00000009 0000  250          TAB     = 9
                0000000A 0000  251          LF      = 10
                0000000B 0000  252          VT      = 11                            ;                       [17]
                0000000C 0000  253          FF      = 12                            ;                       [17]
                0000000D 0000  254          CR      = 13
                0000000F 0000  255          CNTRLO  = 15
                00000011 0000  256          CNTRLQ  = 17
                00000012 0000  257          CNTRLR  = 18
                00000013 0000  258          CNTRLS  = 19
                00000015 0000  259          CNTRLU  = 21
                00000020 0000  260          SPACE   = 32
                0000002D 0000  261          HYPHEN  = 45
                0000003F 0000  262          QUESTN  = 63
                0000005C 0000  263          BSLASH  = 92
                0000007F 0000  264          DELETE  = 127
```

```
                 0000          .SUBTITLE       Work Psect Declarations          ;                    [13]
             00000000          .PSECT  Work, NoExe, NoShr, Wrt, Long            ;                    [13]
                 0000  ;
                 0000  ; OWN STORAGE:
                 0000  ;
                 0000
             00  0000  COLUMN:          .BYTE   0                        ; COLUMN COUNTER
                 0001
             00  0001  LASTCHAR:        .BYTE   0                        ; Last character output to console
                 0002
             50  0002  DS$GB_WIDTH::    .BYTE   80                       ; Initial width is 80          [15]
                 0003
             00  0003  CONSFLG:         .BYTE   0                        ; CONTROL S AND CONTROL Q FLAG
                 0004
       00000000  0004  EVENT_FLAGS:     .LONG   0                        ; Receives event flags         [23]
```

```
                                    0008          .SUBTITLE       Data Psect Declarations          ;                    [13]
                                00000000          .PSECT  Data, NoExe, Shr, NoWrt, Long            ;                    [13]
                                    0000
                                    0000  T_CNTRLC:
              0A 0D 43 5E 0A 0D 00' 0000          .ASCIC  <CR><LF>'^C'<CR><LF>
                                06  0000
                                    0007
                                    0007  T_CNTRLU:
                    0A 0D 55 5E 00' 0007          .ASCIC  '^U'<CR><LF>
                                04  0007
                                    000C
                                    000C  T_CNTRLR:
                          52 5E 00' 000C          .ASCIC  '^R'
                                02  000C
                                    000F
                                    000F  T_CNTRLO:
                    0A 0D 4F 5E 00' 000F          .ASCIC  '^O'<CR><LF>
                                04  000F
                                    0014
                                    0014  T_CRLF:
                          0A 0D 00' 0014          .ASCIC  <CR><LF>
                                02  0014
                                    0017
                                    0017  T_WIDTH:                                                          ;       [12]
20 74 73 75 6D 20 68 74 64 69 57 00' 0017          .ASCIC  "Width must be greater than 0 and less than 133!/"      ;       [12]
74 20 72 65 74 61 65 72 67 20 65 62  0023
65 6C 20 64 6E 61 20 30 20 6E 61 68  002F
21 33 33 31 20 6E 61 68 74 20 73 73  003B
                                2F  0047
                                30  0017
                                    0048
                                    0048  T_SHOW_WIDTH:                                                     ;       [12]
69 77 20 6C 61 6E 69 6D 72 65 54 00' 0048          .ASCIC  "Terminal width is !ZB character!%S!/"                  ;       [12]
63 20 42 5A 21 20 73 69 20 68 74 64  0054
21 53 25 21 72 65 74 63 61 72 61 68  0060
                                2F  006C
                                24  0048
                                    006D
                                    006D  T_OUT_IS_MBX:                                                     ;[28]
64 20 74 6F 6E 6E 61 43 20 3F 3F 00' 006D          .ASCIC  "?? Cannot do SET WIDTH for mailbox output.!/"          ;[28]
20 48 54 44 49 57 20 54 45 53 20 6F  0079
20 78 6F 62 6C 69 61 6D 20 72 6F 66  0085
         2F 21 2E 74 75 70 74 75 6F  0091
                                2C  006D
                                    009A
                                    009A  ;                                                                 [23]
                                    009A  ;TABLE OF VIDEO TERMINAL TYPES                                    [23]
                                    009A  ;                                                                 [23]
                                    009A
                                    009A  VIDEO_TYPES:
                                    009A
                                    009A  T_VT50_A:
                       41 2F 1B 00' 009A          .ASCIC  <27>"/A"
                                03  009A
                                40  009E          .BYTE   TT$_VT5X
                                    009F  T_VT50_B:
                       42 2F 1B 00' 009F          .ASCIC  <27>"/B"
```

```
                     03  009F
                     40  00A3    320          .BYTE   TT$_VT5X
                         00A4    321 T_VT50_C:
                43 2F 1B 00' 00A4    322          .ASCIC  <27>"/C"
                     03  00A4
                     40  00A8    323          .BYTE   TT$_VT5X
                         00A9    324 T_VT55_E:
                45 2F 1B 00' 00A9    325          .ASCIC  <27>"/E"
                     03  00A9
                     41  00AD    326          .BYTE   TT$_VT55
                         00AE    327 T_VT50_H:
                48 2F 1B 00' 00AE    328          .ASCIC  <27>"/H"
                     03  00AE
                     40  00B2    329          .BYTE   TT$_VT5X
                         00B3    330 T_VT50_J:
                4A 2F 1B 00' 00B3    331          .ASCIC  <27>"/J"
                     03  00B3
                     40  00B7    332          .BYTE   TT$_VT5X
                         00B8    333 T_VT52_K:
                4B 2F 1B 00' 00B8    334          .ASCIC  <27>"/K"
                     03  00B8
                     40  00BC    335          .BYTE   TT$_VT52
                         00BD    336 T_VT52_L:
                4C 2F 1B 00' 00BD    337          .ASCIC  <27>"/L"
                     03  00BD
                     40  00C1    338          .BYTE   TT$_VT52
                         00C2    339 T_VT52_M:
                4D 2F 1B 00' 00C2    340          .ASCIC  <27>"/M"
                     03  00C2
                     40  00C6    341          .BYTE   TT$_VT52
                         00C7    342 T_VT100A:
                5A 2F 1B 00' 00C7    343          .ASCIC  <27>"/Z"
                     03  00C7
                     60  00CB    344          .BYTE   TT$_VT100
                         00CC    345 T_VT101:
          63 30 3B 31 3F 5B 1B 00' 00CC    346          .ASCIC  <27>'[?1;0'<99>                          ;VT101 (STRIPPED VT100)
                     07  00CC
                     61  00D4    347          .BYTE   TT$_VT101
                         00D5    348 T_VT102:
             63 36 3F 5B 1B 00' 00D5    349          .ASCIC  <27>'[?6'<99>                            ;VT102 STP, AVO, GO, PRINTER
                     05  00D5
                     62  00DB    350          .BYTE   TT$_VT102
                         00DC    351 T_VT125:
01 3B 30 3B 35 3B 32 31 3F 5B 1B 00' 00DC    352          .ASCIC  <27>'[?12;5;0;'<1><1><1><99><0>          ;VT125 WITHOUT AVO
                00 63 01 01  00E8
                     0F  00DC
                     64  00EC    353          .BYTE   TT$_VT125
                         00ED    354 T_VT125A:
01 3B 31 3B 35 3B 32 31 3F 5B 1B 00' 00ED    355          .ASCIC  <27>'[?12;5;1;'<1><1><1><99><0>          ;VT125 WITHOUT AVO
                00 63 01 01  00F9
                     0F  00ED
                     64  00FD    356          .BYTE   TT$_VT125
                         00FE    357 T_VT125B:
01 3B 30 3B 37 3B 32 31 3F 5B 1B 00' 00FE    358          .ASCIC  <27>'[?12;7;0;'<1><1><1><99><0>          ;VT125 WITH AVO
                00 63 01 01  010A
                     0F  00FE
                     64  010E    359          .BYTE   TT$_VT125
```

```
                                    010F   360 T_VT125C:
01 3B 31 3B 37 3B 32 31 3F 5B 1B 00' 010F   361          .ASCIC  <27>'[?12;7;1;'<1><1><1><99><0>          ;VT125 WITH AVO
                        00 63 01 01  011B
                                 0F  010F
                                 64  011F   362          .BYTE   TT$_VT125
                                     0120   363 T_VT61:
                        27 2F 1B 00' 0120   364          .ASCIC  <27>a/'a
                                 03  0120
                                 60  0124   365          .BYTE   TT$_VT100                               ; ASSIGNED UNTIL DEFINED
                                     0125   366 T_VT71:
                        7E 2F 1B 00' 0125   367          .ASCIC  <27><47><126>
                                 03  0125
                                 60  0129   368          .BYTE   TT$_VT100                               ; ASSIGNED UNTIL DEFINED
                                     012A   369 T_VT100:
           63 30 3B 31 3F 5B 1B 00' 012A   370          .ASCIC  <27>"[?1;0"<99>
                                 07  012A
                                 60  0132   371          .BYTE   TT$_VT100
                                     0133   372 T_VT100B:
           63 31 3B 31 3F 5B 1B 00' 0133   373          .ASCIC  <27>"[?1;1"<99>
                                 07  0133
                                 60  013B   374          .BYTE   TT$_VT100
                                     013C   375 T_VT100C:
           63 32 3B 31 3F 5B 1B 00' 013C   376          .ASCIC  <27>"[?1;2"<99>
                                 07  013C
                                 60  0144   377          .BYTE   TT$_VT100
                                     0145   378 T_VT100D:
           63 33 3B 31 3F 5B 1B 00' 0145   379          .ASCIC  <27>"[?1;3"<99>
                                 07  0145
                                 60  014D   380          .BYTE   TT$_VT100
                                     014E   381 T_VT100E:
           63 34 3B 31 3F 5B 1B 00' 014E   382          .ASCIC  <27>"[?1;4"<99>
                                 07  014E
                                 60  0156   383          .BYTE   TT$_VT100
                                     0157   384 T_VT100F:
           63 35 3B 31 3F 5B 1B 00' 0157   385          .ASCIC  <27>"[?1;5"<99>
                                 07  0157
                                 60  015F   386          .BYTE   TT$_VT100
                                     0160   387 T_VT100G:
           63 36 3B 31 3F 5B 1B 00' 0160   388          .ASCIC  <27>"[?1;6"<99>
                                 07  0160
                                 60  0168   389          .BYTE   TT$_VT100
                                     0169   390 T_VT100H:
           63 37 3B 31 3F 5B 1B 00' 0169   391          .ASCIC  <27>"[?1;7"<99>
                                 07  0169
                                 60  0171   392          .BYTE   TT$_VT100
                                     0172   393 T_VT220:
37 3B 32 3B 31 3B 32 36 3F 5B 1B 00' 0172   394          .ASCIC  <27>"[?62;1;2;7;8"<99>
                        63 38 3B     017E
                                 0E  0172
                                 6E  0181   395          .BYTE   TT$_VT200_SERIES
                                     0182   396 T_VT220A:
                 32 36 3F 5B 1B 00' 0182   397          .ASCIC  <27>'[?62'                                  ;VT200 FAMILY
                                 05  0182
                                 6E  0188   398          .BYTE   TT$_VT200_SERIES
                                     0189   399 T_VT240:
33 3B 32 3B 31 3B 32 35 3F 5B 1B 00' 0189   400          .ASCIC  <27>"[?52;1;2;3;4;7;8"<99>
                  63 38 3B 37 3B 34 3B  0195
```

```
                            12  0189
                            6E  019C   401          .BYTE   TT$_VT200_SERIES
                                019D   402 T_VT132:
        63 30 3B 34 3F 5B 1B 00' 019D   403          .ASCIC  <27>"[?4;0"<99>
                            07  019D
                            66  01A5   404          .BYTE   TT$_VT132
                                01A6   405 T_VT132A:
        63 31 3B 34 3F 5B 1B 00' 01A6   406          .ASCIC  <27>"[?4;1"<99>
                            07  01A6
                            66  01AE   407          .BYTE   TT$_VT132
                                01AF   408 T_VT132B:
        63 32 3B 34 3F 5B 1B 00' 01AF   409          .ASCIC  <27>"[?4;2"<99>
                            07  01AF
                            66  01B7   410          .BYTE   TT$_VT132
                                01B8   411 T_VT132C:
        63 33 3B 34 3F 5B 1B 00' 01B8   412          .ASCIC  <27>"[?4;3"<99>
                            07  01B8
                            66  01C0   413          .BYTE   TT$_VT132
                                01C1   414 T_VT132D:
        63 34 3B 34 3F 5B 1B 00' 01C1   415          .ASCIC  <27>"[?4;4"<99>
                            07  01C1
                            66  01C9   416          .BYTE   TT$_VT132
                                01CA   417 T_VT132E:
        63 35 3B 34 3F 5B 1B 00' 01CA   418          .ASCIC  <27>"[?4;5"<99>
                            07  01CA
                            66  01D2   419          .BYTE   TT$_VT132
                                01D3   420 T_VT132F:
        63 36 3B 34 3F 5B 1B 00' 01D3   421          .ASCIC  <27>"[?4;6"<99>
                            07  01D3
                            66  01DB   422          .BYTE   TT$_VT132
                                01DC   423 T_VT132G:
        63 37 3B 34 3F 5B 1B 00' 01DC   424          .ASCIC  <27>"[?4;7"<99>
                            07  01DC
                            66  01E4   425          .BYTE   TT$_VT132
                                01E5   426 T_VT61A:
                61 2F 1B 00' 01E5   427          .ASCIC  <27><47><97>
                            03  01E5
                            60  01E9   428          .BYTE   TT$_VT100                ; ASSIGNED UNTIL DEFINED
                                01EA   429 T_VT61B:
                62 2F 1B 00' 01EA   430          .ASCIC  <27><47><98>
                            03  01EA
                            60  01EE   431          .BYTE   TT$_VT100                ; ASSIGNED UNTIL DEFINED
                                01EF   432 T_VT61C:
                63 2F 1B 00' 01EF   433          .ASCIC  <27><47><99>
                            03  01EF
                            60  01F3   434          .BYTE   TT$_VT100                ; ASSIGNED UNTIL DEFINED
                                01F4   435 END_VIDEO_TYPES:
                   FFFFFFFF     01F4   436          .LONG -1
                                01F8   437
                                01F8   438 ;                                                              [23]
                                01F8   439 ; TABLE OF HARDCOPY TERMINAL TYPES                             [23]
                                01F8   440 ;                                                              [23]
                                01F8   441
                                01F8   442 HARDCOPY_TYPES:
                                01F8   443
                                01F8   444 T_LA12:
            63 35 31 3F 5B 1B 00' 01F8   445          .ASCIC  <27>"[?15"<99>
```

```
                          06  01F8
                          24  01FF    446          .BYTE   TT$_LA12
                              0200    447 T_LA100:
          63 30 31 3F 5B 1B 00' 0200    448          .ASCIC  <27>"[?10"<99>
                          06  0200
                          25  0207    449          .BYTE   TT$_LA100
                              0208    450 T_LA120:
             63 32 3F 5B 1B 00' 0208    451          .ASCIC  <27>"[?2"<99>
                          05  0208
                          21  020E    452          .BYTE   TT$_LA120
                              020F    453 T_LA34:
       63 30 3B 33 3F 5B 1B 00' 020F    454          .ASCIC  <27>"[?3;0"<99>
                          07  020F
                          22  0217    455          .BYTE   TT$_LA34
                              0218    456 T_LA34A:
       63 31 3B 33 3F 5B 1B 00' 0218    457          .ASCIC  <27>"[?3;1"<99>
                          07  0218
                          22  0220    458          .BYTE   TT$_LA34
                              0221    459 END_HARDCOPY_TYPES:
                    FFFFFFFF  0221    460          .LONG -1
                              0225    461
                              0225    462 T_LA36:                          ; LA36 has no response sequence.       [23]
                          20  0225    463          .BYTE   TT$_LA36
                              0226    464 ;
                              0226    465
```

```
                     0226   467          .SBTTL  VRSetWidth Routine - Set terminal width                 ;       [12]
                 00000000   468          .PSECT  Code, Exe, Shr, NoWrt, Long                             ;       [13]
                     0000   469 ;++                                                                               [12]
                     0000   470 ; VRSETWIDTH                                                                      [12]
                     0000   471 ;                                                                                 [12]
                     0000   472 ; Functional description:                                                         [12]
                     0000   473 ;                                                                                 [12]
                     0000   474 ;       This is a CLI command routine.  It loads the global DS$GB_WIDTH with    [15]
                     0000   475 ;       a value specified to CLI.  If the width is out of range, an error       [12]
                     0000   476 ;       message is typed and it returns a failure status code (though CLI       [12]
                     0000   477 ;       does not currently recognize return status).                            [12]
                     0000   478 ;                                                                                 [12]
                     0000   479 ;       In online mode, a QIO SENSE mode operation is done in order to get      [20]
                     0000   480 ;       the terminal's characteristics, and then a SET is done with the width   [20]
                     0000   481 ;       parameter changed.                                                      [20]
                     0000   482 ;                                                                                 [12]
                     0000   483 ; Calling sequence:                                                               [12]
                     0000   484 ;                                                                                 [12]
                     0000   485 ;       JSB     VRSETWIDTH                                                      [12]
                     0000   486 ;                                                                                 [12]
                     0000   487 ; Input Parameters:                                                               [12]
                     0000   488 ;                                                                                 [12]
                     0000   489 ;       none                                                                    [12]
                     0000   490 ;                                                                                 [12]
                     0000   491 ; Implicit Inputs:                                                               [12]
                     0000   492 ;                                                                                 [12]
                     0000   493 ;       R2      points to CLI data table                                        [12]
                     0000   494 ;                                                                                 [12]
                     0000   495 ; Output parameters:                                                              [12]
                     0000   496 ;                                                                                 [12]
                     0000   497 ;       none                                                                    [12]
                     0000   498 ;                                                                                 [12]
                     0000   499 ; Implicit outputs:                                                              [12]
                     0000   500 ;                                                                                 [12]
                     0000   501 ;       none                                                                    [12]
                     0000   502 ;                                                                                 [12]
                     0000   503 ; Side effects:                                                                  [12]
                     0000   504 ;                                                                                 [12]
                     0000   505 ;       Changes value of DS$GB_WIDTH variable                                   [15]
                     0000   506 ;                                                                                 [12]
                     0000   507 ; Completion codes:                                                              [12]
                     0000   508 ;                                                                                 [12]
                     0000   509 ;       0       failure (value out of range)                                    [12]
                     0000   510 ;       1       success                                                         [12]
                     0000   511 ;--                                                                               [12]
```

```
                                0000   513 VRSETWIDTH::                                ;                                       [12]
                     50   1C A2   D0  0000   514          MOVL    CLI$L_DATA(R2), R0      ; Get width                             [12]
                          71   15  0004   515          BLEQ    10$                     ; Branch if value too small             [12]
              00000084 8F   50   D1  0006   516          CMPL    R0, #132                ; Also branch if                        [12]
                          68   14  000D   517          BGTR    10$                     ;  value too large                      [12]
                                0000F  518
                                000F   519          Br_If_Not_User  5$                  ; Thats all for standalone              [20]
              0000FE00'EF   1C   E1  000F             BBC     #DSA$V_User, -          ; If bit not set, branch
                          54         0016                       L^DSA$GL_Flags, 5$      ;
                                0017   520
                          7E   7C  0017   521          ClrQ    -(SP)                   ; Space for characteristics buffer      [20]
                     50   6E   7E  0019   522          MovAQ   (SP), R0                ; on stack                              [20]
                                001C   523          $QIOW_S , -                     ; First, read the terminal characteristics
                                001C   524                  L^DS$GW_TTOUT, -        ; Channel number                        [20]
                                001C   525                  #IO$_SENSEMODE, -       ; Function                              [20]
                                001C   526                  P1 = (R0)               ; Address of characteristics buffer     [20]
                          7E   7C  001C                      CLRQ    -(SP)
                          7E   7C  001E                      CLRQ    -(SP)
                          00   DD  0020                      PUSHL   #0
                          60   DF  0022                      PUSHAL  (R0)
                          7E   7C  0024                      CLRQ    -(SP)
                          00   DD  0026                      PUSHL   #0
                   7E   0000'8F   3C  0028                   MOVZWL  #IO$_SENSEMODE,-(SP)
              7E   00000000'EF   3C  002D                   MOVZWL  L^DS$GW_TTOUT,-(SP)
                          00   DD  0034                      PUSHL   #0
              00000000'GF   0C   FB  0036                   CALLS   #12,G^SYS$QIOW
                     4D   50   E9  003D   527          BLBC    R0, 20$                 ; Error, output is not terminal, ie mbx [28]
                                0040   528
                02 AE   1C A2   9B  0040   529          MovZBW  CLI$L_DATA(R2), 2(SP)   ; Set the desired page width          [20]
                     50   6E   7E  0045   530          MovAQ   (SP), R0                ; Buffer on the stack                   [20]
                                0048   531          $QIOW_S , -                     ; no efn                                [20]
                                0048   532                  L^DS$GW_TTOUT, -        ; Channel number                        [20]
                                0048   533                  #IO$_SETMODE, -         ; Function                              [20]
                                0048   534                  P1 = (R0)               ; Address of characteristics buffer     [20]
                          7E   7C  0048                      CLRQ    -(SP)
                          7E   7C  004A                      CLRQ    -(SP)
                          00   DD  004C                      PUSHL   #0
                          60   DF  004E                      PUSHAL  (R0)
                          7E   7C  0050                      CLRQ    -(SP)
                          00   DD  0052                      PUSHL   #0
                   7E   0000'8F   3C  0054                   MOVZWL  #IO$_SETMODE,-(SP)
              7E   00000000'EF   3C  0059                   MOVZWL  L^DS$GW_TTOUT,-(SP)
                          00   DD  0060                      PUSHL   #0
              00000000'GF   0C   FB  0062                   CALLS   #12,G^SYS$QIOW
                          8E   7C  0069   535          ClrQ    (SP)+                   ; Release the buffer on the stack       [20]
                                006B   536
        00000002'EF   1C A2   90  006B   537 5$:       MOVB    CLI$L_DATA(R2), DS$GB_WIDTH     ; Store the value               [15]
                     50   01   D0  0073   538          MOVL    #1, R0                  ; Return success                        [12]
                          05   0076   539          RSB                                 ;                                       [12]
                                0077   540
                                0077   541 10$:      $Print  -                       ; Print with a typecode                 [16]
                                0077   542                    #DS$K_Type_Command_Err, -     ; ... the typecode                [16]
                                0077   543                    #DS$K_PrintF, -         ; ... use PrintF to print it            [16]
                                0077   544                    T_Width                 ; ... the message                       [16]
                   00000017'EF   9F  0077                   PUSHAB  T_Width
                       7E   01   9A  007D                   movzbl  #DS$K_PrintF, -(sp)
                       7E   15   98  0080                   cvtbl   #DS$K_Type_Command_Err, -(sp)
```

```
00000000'GF   03   FB  0083                          CALLS   #$$N+2, G^DSX$PRINT
              50   D4  008A   545          CLRL     R0                      ; Return failure                    [12]
                   05  008C   546          RSB                              ;                                   [12]
                       008D   547
              8E   7C  008D   548 20$:     ClrQ     (SP)+                   ; Release the buffer on the stack   [28]
                       008F   549          $Print   -                       ; Error, attempt to SET WIDTH       [28]
                       008F   550                     #DS$K_Type_Command_Err, -     ; ...for mbx output         [28]
                       008F   551                     #DS$K_PrintF, -        ; ... use PrintF to print it        [28]
                       008F   552                     T_OUT_IS_MBX           ; ... the message                   [28]
     0000006D'EF   9F  008F                          PUSHAB  T_OUT_IS_MBX
          7E   01  9A  0095                          movzbl  #DS$K_PrintF, -(sp)
          7E   15  98  0098                          cvtbl   #DS$K_Type_Command_Err, -(sp)
00000000'GF   03   FB  009B                          CALLS   #$$N+2, G^DSX$PRINT
              50   D4  00A2   553          CLRL     R0                      ; Return failure                    [28]
                   05  00A4   554          RSB                              ;                                   [28]
                       00A5   555
```

```
00A5   557          .SBTTL  VRShowWidth Routine - Show terminal width       ;                [12]
00A5   558
00A5   559 ;++                                                                              [12]
00A5   560 ; VRSHOWWIDTH                                                                   [12]
00A5   561 ;                                                                               [12]
00A5   562 ; Functional description:                                                       [12]
00A5   563 ;                                                                               [12]
00A5   564 ;       This is a CLI command routine.  It prints the current DS$GB_WIDTH.     [12]
00A5   565 ;                                                                               [12]
00A5   566 ; Calling sequence:                                                             [12]
00A5   567 ;                                                                               [12]
00A5   568 ;       JSB     VRSHOWWIDTH                                                     [12]
00A5   569 ;                                                                               [12]
00A5   570 ; Input Parameters:                                                             [12]
00A5   571 ;                                                                               [12]
00A5   572 ;       none                                                                    [12]
00A5   573 ;                                                                               [12]
00A5   574 ; Implicit Inputs:                                                              [12]
00A5   575 ;                                                                               [12]
00A5   576 ;       none                                                                    [12]
00A5   577 ;                                                                               [12]
00A5   578 ; Output parameters:                                                            [12]
00A5   579 ;                                                                               [12]
00A5   580 ;       none                                                                    [12]
00A5   581 ;                                                                               [12]
00A5   582 ; Implicit outputs:                                                             [12]
00A5   583 ;                                                                               [12]
00A5   584 ;       none                                                                    [12]
00A5   585 ;                                                                               [12]
00A5   586 ; Side effects:                                                                 [12]
00A5   587 ;                                                                               [12]
00A5   588 ;       none                                                                    [12]
00A5   589 ;                                                                               [12]
00A5   590 ; Completion codes:                                                             [12]
00A5   591 ;                                                                               [12]
00A5   592 ;       1       success                                                         [12]
00A5   593 ;--                                                                             [12]
```

```
                                00A5   595 VRSHOWWIDTH::                                  ;                                        [12]
      50   00000002'EF    9A   00A5   596         MovZBL  L^DS$GB_Width, R0               ; Put the current width in R0            [16]
                                00AC   597         $Print  -                               ; Print with a typecode                  [16]
                                00AC   598                   #DS$K_Type_Command_Out, -      ; ... the typecode number          [16]
                                00AC   599                   #DS$K_PrintF, -                ; ... the type of print routine          [16]
                                00AC   600                   T_Show_Width, -                ; ... the format message                 [16]
                                00AC   601                   R0                             ; ... and the width                      [16]
                     50    DD   00AC                       PUSHL   R0
           00000048'EF    9F   00AE                       PUSHAB  T_Show_Width
                7E   01    9A   00B4                       movzbl  #DS$K_PrintF, -(sp)
                7E   16    98   00B7                       cvtbl   #DS$K_Type_Command_Out, -(sp)
      00000000'GF    04    FB   00BA                       CALLS   #$$N+2, G^DSX$PRINT
                                00C1   602
                50   01    D0   00C1   603         MOVL    #1, R0                          ; Set success                            [12]
                           05   00C4   604         RSB                                     ;                                        [12]
```

```
00C5    606          .SBTTL  KB_Poll Routine - Keyboard ready bit check
00C5    607 ;**
00C5    608 ; FUNCTIONAL DESCRIPTION:
00C5    609 ;
00C5    610 ;       THIS ROUTINE CHECKS THE 'READY' BIT IN THE CONSOLE'S STATUS
00C5    611 ;       REGISTER AND, IF CLEAR, RETURNS TO THE CALLING ROUTINE.
00C5    612 ;       If no character is ready this returns immediately.
00C5    613 ;       Select one of the following based on the character typed:
00C5    614 ;       ^C      clear local flags and set the global ^C flag
00C5    615 ;               set ^O flag to suppress output until ^C is serviced by KB_CHECK
00C5    616 ;       ^O      toggle the ^O bit (output may be flushed based on this bit)
00C5    617 ;       ^S      SET CONSFLG FLAG TO SUSPEND TERMINAL OUTPUT
00C5    618 ;       ^Q      CLEAR CONSFLG FLAG TO RESUME TERMINAL OUTPUT IF ANY
00C5    619 ;       ^C      WILL ALSO CLEAR CONSFLG FLAG.
00C5    620 ;       Else,   echo the character as a <BELL>
00C5    621 ;
00C5    622 ; CALLING SEQUENCE:
00C5    623 ;
00C5    624 ;       SUBROUTINE CALL.
00C5    625 ;
00C5    626 ; INPUT PARAMETERS:
00C5    627 ;
00C5    628 ;       NONE
00C5    629 ;
00C5    630 ; IMPLICIT INPUTS:
00C5    631 ;
00C5    632 ; OUTPUT PARAMETERS:
00C5    633 ;
00C5    634 ;       DS$GL_FLAGS<DS$V_CTRLC>         = ^C FLAG.
00C5    635 ;
00C5    636 ; IMPLICIT OUTPUTS:
00C5    637 ;
00C5    638 ;       NONE
00C5    639 ;
00C5    640 ; COMPLETION CODES:
00C5    641 ;
00C5    642 ;       NONE
00C5    643 ;
00C5    644 ; SIDE EFFECTS:
00C5    645 ;
00C5    646 ;       NONE
00C5    647 ;--
```

```
                        00C5   KB_POLL::
          03    BB      00C5            PUSHR   #^M<R0,R1>                  ; SAVE R0 & R1.
                        00C7            CMK     RCONSOLE
          7E    DC      00C7            MOVPSL  -(SP)
   06 6E  1A    E0      00C9            BBS     #PSL$V_IS, (SP), 30000$
          8E    D4      00CD            CLRL    (SP)+
          01    BC      00CF            CHMK    #CMK$_RCONSOLE
          06    11      00D1            BRB     30001$
 00000132'EF    16      00D3    30000$: JSB     CMK$RCONSOLE
                        00D9    30001$:
                        00D9
   1C 50  07    E1      00D9            BBC     #7,R0,KB_POLL_X             ; EXIT IF NO CHARACTER
                        00DD
      51  11    91      00DD            CMPB    #CNTRLQ,R1
          17    13      00E0            BEQL    KB_POLL_X                   ; SKIP ECHO IF CONTROL Q
          0F    19      00E2            BLSS    10$                         ; SKIP FOR SPEED OPTIMIZATION
                        00E4
      51  03    91      00E4            CMPB    #CNTRLC,R1
          10    13      00E7            BEQL    KB_POLL_X                   ; SKIP ECHO IF CONTROL-C
                        00E9
      51  0F    91      00E9            CMPB    #CNTRLO,R1
          0B    13      00EC            BEQL    KB_POLL_X                   ; SKIP ECHO IF CONTROL-O
                        00EE
      51  13    91      00EE            CMPB    #CNTRLS,R1
          06    13      00F1            BEQL    KB_POLL_X                   ; SKIP ECHO IF CONTROL S
                        00F3
      51  07    D0      00F3    10$:    MOVL    #BELL,R1                    ; ECHO A BELL.
        017A    30      00F6            BSBW    OUT_ECHO                    ; TYPE IT OUT
                        00F9
                        00F9    KB_POLL_X:
          03    BA      00F9            POPR    #^M<R0,R1>                  ; RESTORE R0 & R1.
                05      00FB            RSB                                 ; RETURN.
```

```
                        00FC    675          .SBTTL  RInput Routine - Flag driven terminal input
                        00FC    676 ;++
                        00FC    677 ; FUNCTIONAL DESCRIPTION:
                        00FC    678 ;
                        00FC    679 ;       This routine returns a character from the console terminal.
                        00FC    680 ;       The associated interrupt routine CMK$RCONSOLE checks the
                        00FC    681 ;       the character and sets or cleas flags for ^S, ^Q, ^O.
                        00FC    682 ;
                        00FC    683 ; CALLING SEQUENCE:
                        00FC    684 ;
                        00FC    685 ;       BSBW    RINPUT
                        00FC    686 ;
                        00FC    687 ; INPUT PARAMETERS:
                        00FC    688 ;
                        00FC    689 ;       NONE
                        00FC    690 ;
                        00FC    691 ; IMPLICIT INPUTS:
                        00FC    692 ;
                        00FC    693 ;       NONE
                        00FC    694 ;
                        00FC    695 ; OUTPUT PARAMETERS:
                        00FC    696 ;
                        00FC    697 ;       NONE
                        00FC    698 ;
                        00FC    699 ; IMPLICIT OUTPUTS:
                        00FC    700 ;
                        00FC    701 ;       NONE
                        00FC    702 ;
                        00FC    703 ; COMPLETION CODES:
                        00FC    704 ;
                        00FC    705 ;       NONE
                        00FC    706 ;
                        00FC    707 ; SIDE EFFECTS:
                        00FC    708 ;
                        00FC    709 ;       NONE
                        00FC    710 ;--
```

```
                                00FC  712 RINPUT:                                  ; KEYBOARD INPUT ROUTINE.
      00000000'EF    17    E5   00FC  713          BBCC    S^#DS$V_OUTPUT, -
                     03         0103  714                   L^DS$GL_FLAGS, 10$     ; Branch if no asynch output, clear flag
                   00ED    30   0104  715          BSBW    RTYPE                    ; Re-type input line
                                0107  716
                                0107  717 10$:     CMK     RCONSOLE                 ; READ KEYBOARD.
                     7E    DC   0107                MOVPSL  -(SP)
            06 6E    1A    E0   0109                BBS     #PSL$V_IS, (SP), 30002$
                     8E    D4   010D                CLRL    (SP)+
                     01    BC   010F                CHMK    #CMK$_RCONSOLE
                     06    11   0111                BRB     30003$
         00000132'EF       16   0113        30002$: JSB     CMK$RCONSOLE
                                0119        30003$:
            DF 50    07    E1   0119  718          BBC     #7,R0,RINPUT             ; TRY AGAIN IF NOTHING THERE
   00'8F   00000000'EF    91   011D  719          CMPB    DS$GB_SYSTYPE,#PR$_SYS_TYP800 ; Is it AURORA cpu ? [29][35]   ;G:3
                     07    13   0125  720          BEQL    20$                      ; Don't check if AURORA. Prevent hangs. [29]
      00   51   04    08    ED   0127  721          CMPZV   #8,#4,R1,#0              ; CONSOLE DATA?
                     CE    12   012C  722          BNEQ    RINPUT                   ; NO, IGNORE IT
               50    51    D0   012E  723 20$:     MOVL    R1,R0                    ; COPY CHARACTER
                                0131  724
                                0131  725 RINPUTX:                                 ; ROUTINE EXIT POINT.
                           05   0131  726          RSB                              ; RETURN TO CALLING ROUTINE
                                0132  727
                                0132  728 CMK$RCONSOLE::
                     51    D4   0132  729 10$:     CLRL    R1                       ; INITIALIZE R1.                    [19]
                                0134  730
               50    20    DB   0134  731 20$:     MFPR    #PR$_RXCS, R0            ; GET KEYBOARD STATUS.
            03 50    07    E0   0137  732          BBS     #7, R0, 30$              ;
                   00B5    31   013B  733          BRW     140$                     ; No character, exit
                                013E  734
               51    21    DB   013E  735 30$:     MFPR    #PR$_RXDB, R1            ; INPUT CHARACTER.
            EF 51    0F    E0   0141  736          BBS     #15,R1,20$               ; Error on character, try again.
                                0145  737
   00'8F   00000000'EF    91   0145  738 60$:     CMPB    DS$GB_SYSTYPE,#PR$_SYS_TYP800 ; Is it AURORA cpu ? [29][35]   ;G:3
                     0A    13   014D  739          BEQL    70$                      ; DON'T check if aurora. Allow ^C etc   [32]
         51   0F00 8F    B3   014F  740          BITW    #^XF00,R1                ; Is this a console character?
                     03    13   0154  741          BEQL    70$                      ; Yes
                   009A    31   0156  742          BRW     140$                     ; No - just REI
                                0159  743
      51   51   07    00    EF   0159  744 70$:     EXTZV   #0,#7,R1,R1              ; Remove parity and garbage
               51    03    91   015E  745          CMPB    #CNTRLC,R1               ; CONTROL-C?
                     4B    12   0161  746          BNEQ    100$                     ; Check next
         00000003'EF       94   0163  747          CLRB    CONSFLG                  ; RESUME PRINTING ON ^C
                                0169  748
                                0169  749          Br_If_Ctrl_C_No_Echo    80$     ; If not echoing ^C's, skip around      [19]
      00000000'EF    01    E0   0169                BBS     #CRD$V_Ctrl_C_No_Echo, -         ; If bit set, branch
                     20         0170                          L^DS$GL_CRD_Flags, 80$         ;
                     03    BB   0171  750          PUSHR   #^M<R0,R1>               ; Save flag and character
      50   00000000'EF    9E   0173  751          MOVAB   T_CNTRLC,R0              ; ASCIC <NL>^C<NL>
                   00AF    30   017A  752          BSBW    OUT_CNTRL                ; Type it
                     03    BA   017D  753          POPR    #^M<R0,R1>               ; Restore
                                017F  754          Set_CtrlO                        ; Set ^O flag                       [16]
      00000000'EF    10    E2   017F                BBSS    #DS$V_CtrlO, -           ; Branch if CTRLO bit is set to
                     00         0186                          L^DS$GL_Flags, 30004$ ; ...
                                0187        30004$:                                  ; ... here. Set bit regardless.
                                0187  755          Set_CtrlC                        ; Set ^C flag                       [16]
      00000000'EF    00    E2   0187                BBSS    #DS$V_CtrlC, -           ; Branch if CTRLC bit is set to
```

```
                    00        018E                          L^DS$GL_Flags, 30005$ ; ...
                              018F        30005$:                                 ; ... here. Set bit regardless.
                    62    11  018F   756          BRB     140$                    ; Return Control C character
                              0191   757
                              0191   758 80$:     ; We are not echoing ^C's. Now check to see if we are allowing ^C as a  [19]
                              0191   759          ; valid character. If we are not, then go back up and see if there is   [19]
                              0191   760          ; another character there.                                              [19]
                              0191   761
                              0191   762          Br_If_Not_Flush_Ctrl_C  90$
        00000000'EF   02    E1  0191               BBC     #CRD$V_Flush_Ctrl_C, -          ; If bit not set, branch        [35]
                    03        0198                          L^DS$GL_CRD_Flags, 90$        ;                               [35]
                  FF96    31  0199   763          BRW     10$                     ; If flushing, return with no character [19]
                              019C   764 90$:     Set_Flush_Ctrl_C                ; Not currently flushing -> start now,  [19]
        00000000'EF   02    E2  019C               BBSS    #CRD$V_Flush_Ctrl_C, -  ; Branch if FLUSH_CTRL_C bit is set to
                    00        01A3                          L^DS$GL_CRD_Flags, 30006$     ; ...
                              01A4        30006$:                                 ; ... here. Set bit regardless.
                              01A4   765          Set_CtrlC                       ; . . . set ^C flag, and                [19]
        00000000'EF   00    E2  01A4               BBSS    #DS$V_CtrlC, -          ; Branch if CTRLC bit is set to
                    00        01AB                          L^DS$GL_Flags, 30007$ ; ...
                              01AC        30007$:                                 ; ... here. Set bit regardless.
                    45    11  01AC   766          BRB     140$                    ; . . . return with the ^C character    [19]
                              01AE   767
                              01AE   768 100$:    Clear_Flush_Ctrl_C              ; This is not a ^C - stop flushing now  [19]
        00000000'EF   02    E4  01AE               BBSC    #CRD$V_Flush_Ctrl_C, -  ; Branch if FLUSH_CTRL_C bit is set to
                    00        01B5                          L^DS$GL_CRD_Flags, 30008$     ; ...
                              01B6        30008$:                                 ; ... here. Clear bit regardless.
                 51   13    91  01B6   769          CMPB    #CNTRLS,R1              ; CONTROL-S ?
                    09    12  01B9   770          BNEQ    110$                    ; NO, NEXT
        00000003'EF   01    88  01BB   771          BISB    #1,CONSFLG              ; STOP PRINTING
                    2D    11  01C2   772          BRB     130$                    ; Flush and exit
                              01C4   773
                 51   11    91  01C4   774 110$:    CMPB    #CNTRLQ,R1              ; CONTROL-Q ?
                    08    12  01C7   775          BNEQ    120$                    ; NO EXIT
            00000003'EF   94  01C9   776          CLRB    CONSFLG                 ; RESUME PRINTING
                    20    11  01CF   777          BRB     130$                    ; Flush and exit
                              01D1   778
                 51   0F    B1  01D1   779 120$:    CMPW    #CNTRLO,R1              ; CONTROL O?
                    1D    12  01D4   780          BNEQ    140$                    ; NO
            00000003'EF   94  01D6   781          CLRB    CONSFLG                 ; RESUME PRINTING ON CONTROL O
        50  0000000F'EF   9E  01DC   782          MOVAB   T_CNTRLO,R0             ; ECHO STRING FOR CNTRLO
                  0046    30  01E3   783          BSBW    OUT_CNTRL               ; TYPE IT
00000000'EF 00010000 8F   CC  01E6   784          XORL2   #DS$M_CTRLO,DS$GL_FLAGS ; TOGGLE ^O FLAG
                              01F1   785
                    50    7C  01F1   786 130$:    CLRQ    R0                      ; Set to no character found
                              01F3   787
                        02    01F3   788 140$:    REI                             ; Return from interrupt
```

```
                        .SBTTL  RType Routine - Retype input line ('^R' function)
                ;++
                ; Functional description:
                ;
                ;       Cause a linefeed-carriage return, echo prompt and whatever characters
                ;       have been read so far from the terminal on the current line.
                ;
                ; Implicit inputs:
                ;
                ;       R2              count of characters input
                ;       QIO$_P5(AP)     prompt length/pointer from $QIO argument list
                ;       QIO$_P1(AP)     pointer to input buffer
                ;
                ; Side effects:
                ;
                ;       If this routine is interrupted, and output to terminal occurs, the
                ;       routine will restart itself--repeatedly if necessary, until the line
                ;       is output completely.
                ;--
```

```
                    38   BB   01F4   810 RTYPE:  PUSHR   #^M<R3,R4,R5>           ; Save registers
          55  00000000'EF   9E   01F6   811        MOVAB   L^DS$GL_FLAGS, R5      ; Save pointer to flags longword
                          01FD   812
          50  00000014'EF   9E   01FD   813 10$:   MOVAB   T_CRLF, R0             ; Type <CR><LF>
                    4D   10   0204   814        BSBB    OUT_ASCIC
              53    2C AC   7D   0206   815        MOVQ    QIO$_P5(AP), R3        ; Get prompt descriptor (ADDRESS/LENGTH)
                          020A   816
                 51   83   9A   020A   817 20$:   MOVZBL  (R3)+, R1              ; Get next character of prompt
              EC 65   17   E4   020D   818        BBSC    S^#DS$V_OUTPUT,(R5),10$ ; Re-start if asynch typeout
                    55   10   0211   819        BSBB    OUT_CHAR               ; Type it out
                    F4 54   F5   0213   820        SOBGTR  R4, 20$                ; Loop to end of prompt
              54    1C BC   9E   0216   821        MOVAB   @QIO$_P1(AP), R4       ; Point to buffer
                 53   52   D0   021A   822        MOVL    R2, R3                 ; Copy length
                          021D   823
                 51   84   9A   021D   824 30$:   MOVZBL  (R4)+, R1              ; Get next character
              D9 65   17   E4   0220   825        BBSC    S^#DS$V_OUTPUT,(R5),10$ ; Re-start if asynch typeout
                    42   10   0224   826        BSBB    OUT_CHAR               ; Type it
                    F4 53   F5   0226   827        SOBGTR  R3, 30$                ; Loop to end of line
                    38   BA   0229   828        POPR    #^M<R3,R4,R5>          ; Restore registers
                         05   022B   829        RSB                            ; Return
```

```
                        831          .SBTTL  Out Routines - Flag driven terminal output
                        832 ;++
                        833 ; FUNCTIONAL DESCRIPTION:
                        834 ;
                        835 ;       OUT_CNTRL       Print out control character echo with low overhead
                        836 ;       OUT_ASCIC       OUTPUT ASCIC STRING R0=ADDRESS
                        837 ;       OUT_STRING      OUTPUT STRING R0=ADDRESS, R1=LENGTH
                        838 ;       CHAR_OUTPUT     OUTPUT CHARACTER R1=CHARACTER
                        839 ;       CHAR_ECHO       OUTPUT CHARACTER R1=CHARACTER, IGNORE ^O
                        840 ;
                        841 ; CALLING SEQUENCE:
                        842 ;
                        843 ;       NONE
                        844 ;
                        845 ; INPUT PARAMETERS:
                        846 ;
                        847 ;       NONE
                        848 ;
                        849 ; IMPLICIT INPUTS:
                        850 ;
                        851 ;       NONE
                        852 ;
                        853 ; OUTPUT PARAMETERS:
                        854 ;
                        855 ;       NONE
                        856 ;
                        857 ; IMPLICIT OUTPUTS:
                        858 ;
                        859 ;       NONE
                        860 ;
                        861 ; COMPLETION CODES:
                        862 ;
                        863 ;       NONE
                        864 ;
                        865 ; SIDE EFFECTS:         NONE
                        866 ;--
```

```
                                 022C   OUT_CNTRL:
                      52   DD    022C           PUSHL   R2                          ; Save register 2
                 52   80   9A    022E           MOVZBL  (R0)+,R2                    ; Get length
                                 0231
                 51   80   9A    0231   10$:    MOVZBL  (R0)+,R1                    ; Get character
                                 0234           CMK     TCONSOLE                    ; Type character
                      7E   DC    0234           MOVPSL  -(SP)
              06 6E   1A   E0    0236           BBS     #PSL$V_IS, (SP), 30009$
                      8E   D4    023A           CLRL    (SP)+
                      02   BC    023C           CHMK    #CMK$_TCONSOLE
                      06   11    023E           BRB     30010$
           000002EE'EF     16    0240   30009$: JSB     CMK$TCONSOLE
                                 0246   30010$:
                 E8 52   F5      0246           SOBGTR  R2,10$                      ; Loop
           00000000'EF     94    0249           CLRB    COLUMN                      ; Control char. echo ends with <CR><LF>
                      52 8ED0    024F           POPL    R2                          ; Restore register
                         05      0252           RSB                                 ; Return
                                 0253
                                 0253   OUT_ASCIC:
                 51   80   9A    0253           MOVZBL  (R0)+,R1                    ; GET LENGTH
                                 0256
                                 0256   OUT_STRING:
                      04   BB    0256           PUSHR   #^MR2                       ; SAVE
                 52   51   D0    0258           MOVL    R1,R2                       ; COPY LENGTH
                      08   15    025B           BLEQ    20$                         ; START WITH COUNT
                                 025D
                 51   80   9A    025D   10$:    MOVZBL  (R0)+,R1                    ; GET CHARACTER
                      06   10    0260           BSBB    OUT_CHAR                    ; OUTPUT A CHARACTER
                 F8 52   F5      0262           SOBGTR  R2,10$                      ; NEXT
                                 0265
                      04   BA    0265   20$:    POPR    #^MR2                       ; RESTORE
                         05      0267           RSB
                                 0268
                                 0268   OUT_CHAR:                                   ; TELETYPE OUTPUT ROUTINE.
                 FE5A     30      0268           BSBW    KB_POLL                     ; Check for ^C and ^O
                                 026B           Br_If_Ctrl0     Out_CharX           ; Exit if ^O in effect                [16]
      00000000'EF   10   E0      026B           BBS     #DS$V_Ctrl0, -                      ; If bit set, branch
                      7A         0272                     L^DS$GL_Flags, Out_CharX                ;
                                 0273
                                 0273   OUT_ECHO:
                                 0273           Br_If_Not_APT   5$                  ; Branch around if not in APT mode    [16]
      0000FE00'EF   1F   E1      0273           BBC     #DSA$V_Apt, -                       ; If bit not set, branch
                      08         027A                     L^DSA$GL_Flags, 5$                  ;
      0000FE00'EF   1B   E0      027B           BBS     #DSA$V_NORPT, -
                      6A         0282                     DSA$GL_FLAGS,OUT_CHARX; BRANCH IF REPORT DISABLED ?
                                 0283
        05 00000003'EF   E9      0283   5$:     BLBC    CONSFLG,10$                 ; BRANCH IF CONTROL S NOT ON
                 FE38     30      028A           BSBW    KB_POLL                     ; WAIT FOR CONTROL Q OR CONTROL C
                      F4   11    028D           BRB     5$
                                 028F
                 51   0D   91    028F   10$:    CMPB    #CR,R1                      ; CARRIAGE RETURN ?
                      06   12    0292           BNEQ    20$                         ; NO -
           00000000'EF     94    0294           CLRB    COLUMN                      ; INIT COLUMN COUNT
                                 029A
00000002'EF  00000000'EF   91    029A   20$:    CMPB    COLUMN, DS$GB_WIDTH         ; Have we passed width limit?         [15]
                      0E   1F    02A5           BLSSU   25$                         ; (Must be unsigned) No, continue     [12]
                      07   BB    02A7           PUSHR   #^M<R0,R1,R2>               ; Save registers                      [12]
```

```
                50   00000014'EF   9E   02A9   914          MOVAB   T_CRLF, R0                ; Yes, Type CRLF, reset COLUMN          [12]
                          FFA0   30   02B0   915          BSBW    OUT_ASCIC                   ;                                       [12]
                            07   BA   02B3   916          POPR    #^M<R0,R1,R2>               ; Restore                               [12]
                                 02B5   917                                                                                          [12]
                                 02B5   918 25$:                                             ;
                            51   09   91   02B5   919          CMPB    #TAB,R1                     ; CHARACTER = TAB
                                 0F   12   02B8   920          BNEQ    50$                         ; NO -
                            51   20   9A   02BA   921          MOVZBL  #SPACE,R1                   ; SPACE
                                 02BD   922
                                 0A   10   02BD   923 30$:     BSBB    50$                         ; OUTPUT A SPACE
                00000000'EF      07   93   02BF   924          BITB    #7,COLUMN                   ; ALIGNED ON MULT OF 8 BOUNDRY?
                                 F5   12   02C6   925          BNEQ    30$                         ; NO, OUTPUT ANOTHER SPACE
                                      05   02C8   926          RSB                                 ; YES, RETURN
                                      02C9   927
                            51   1F   91   02C9   928 50$:     CMPB    #31,R1                      ; Control character?
                                 06   18   02CC   929          BGEQ    60$                         ; Not Horizontal motion if control char
                      00000000'EF   96   02CE   930          INCB    COLUMN                      ; Inc position
                                      02D4   931
                00000001'EF      51   90   02D4   932 60$:     MOVB    R1,LASTCHAR                 ; Save character to be saved
                                      02DB   933          CMK     TCONSOLE
                                 7E   DC   02DB              MOVPSL  -(SP)
                          06 6E      1A   E0   02DD              BBS     #PSL$V_IS, (SP), 30011$
                                     8E   D4   02E1              CLRL    (SP)+
                                     02   BC   02E3              CHMK    #CMK$_TCONSOLE
                                     06   11   02E5              BRB     30012$
                      000002EE'EF    16   02E7         30011$: JSB     CMK$TCONSOLE
                                      02ED         30012$:
                                      02ED   934
                                      02ED   935 OUT_CHARX:
                                 05   02ED   936          RSB                                 ; RETURN TO CALLING ROUTINE.
                                      02EE   937
```

```
                        02EE   939 CMK$TCONSOLE::                          ; TELETYPE OUTPUT ROUTINE.
            01    BB    02EE   940         PUSHR   #^M<R0>                 ; Save registers                [25][34]
                        02F0   941
        50  22    DB    02F0   942 10$:    MFPR    #PR$_TXCS, R0           ; GET PRINTER STATUS.
     F9 50  07    E1    02F3   943         BBC     #7, R0, 10$             ; WAIT FOR PRINTER READY BIT.
        23  51    DA    02F7   944         MTPR    R1, #PR$_TXDB           ; OUTPUT CHARACTER.
            01    BA    02FA   945         POPR    #^M<R0>                 ; Restore                       [25][34]
                  02    02FC   946         REI                             ; RETURN TO CALLING ROUTINE.
                        02FD   947
```

```
02FD    949          .SBTTL  WriteVBlk Routine - Console write driver
02FD    950 ;++
02FD    951 ; FUNCTIONAL DESCRIPTION:
02FD    952 ;
02FD    953 ;       PERFORMS AN OUTPUT TO THE CONSOLE OF AN ASCII STRING.
02FD    954 ;
02FD    955 ; CALLING SEQUENCE:
02FD    956 ;
02FD    957 ;       CASE FROM QIO.
02FD    958 ;
02FD    959 ; INPUT PARAMETERS:
02FD    960 ;
02FD    961 ;       QIO$_EFN(AP)    = QIO EVENT FLAG NUMBER.
02FD    962 ;       QIO$_P1(AP)     = ADDRESS OF ASCII STRING.
02FD    963 ;       QIO$_P2(AP)     = LENGTH OF ASCII STRING.
02FD    964 ;
02FD    965 ; IMPLICIT INPUTS:
02FD    966 ;
02FD    967 ;       NONE
02FD    968 ;
02FD    969 ; OUTPUT PARAMETERS:
02FD    970 ;
02FD    971 ;       NONE
02FD    972 ;
02FD    973 ; IMPLICIT OUTPUTS:
02FD    974 ;
02FD    975 ;       NONE
02FD    976 ;
02FD    977 ; COMPLETION CODES:
02FD    978 ;
02FD    979 ;       NONE
02FD    980 ;
02FD    981 ; SIDE EFFECTS:
02FD    982 ;
02FD    983 ;       NONE
02FD    984 ;--
```

```
                                02FD   986 WRITEVBLK::
      08 0C AC    00'   E1      02FD   987          BBC     S^#IO$V_CANCTRLO, -
                                0302   988                  QIO$_FUNC(AP),10$       ; Branch if no Cancel Control O
                                0302   989          Clear_CtrlO                     ; Clear the ^O flag                [16]
  00000000'EF     10    E4      0302                BBSC    #DS$V_CtrlO, -          ; Branch if CTRLO bit is set to
                  00            0309                          L^DS$GL_Flags, 30013$ ; ...
                                030A        30013$:                                 ; ... here. Clear bit regardless.
                                030A   990
         50    1C AC    7D      030A   991 10$:     MOVQ    QIO$_P1(AP), R0         ; GET ADDRESS/LENGTH OF ASCII STRING.
                FF45    30      030E   992          BSBW    OUT_STRING              ; OUTPUT THE STRING
                                0311   993          $SETEF_G        (AP)
  00000000'GF     6C    FA      0311                        CALLG   (AP),G^SYS$SETEF
                        05      0318   994          RSB
                                0319   995 ;        RET                             ; RETURN TO ENTRY VECTOR.       [30]
```

```
0319   997          .SBTTL  ReadVBlk Routine - Console read driver
0319   998 ;++
0319   999 ; FUNCTIONAL DESCRIPTION:
0319  1000 ;
0319  1001 ;       THE ROUTINE INPUTS CHARACTERS FROM THE TELETYPE AND STORES THEM
0319  1002 ;       IN THE BUFFER.  IT HANDLES "DELETE", "^U", "^C", "^R", AND
0319  1003 ;       "HYPHEN_CARRTN".
0319  1004 ;
0319  1005 ; CALLING SEQUENCE:
0319  1006 ;
0319  1007 ;       CASE FROM QIO.
0319  1008 ;
0319  1009 ; INPUT PARAMETERS:
0319  1010 ;
0319  1011 ;       QIO$_FUNC(AP)    = FUNCTION EITHER IO$_READVBLK OR IO$_READPROMPT
0319  1012 ;       QIO$_P1(AP)      = ADDRESS OF THE BEGINNING OF THE BUFFER.
0319  1013 ;       QIO$_IOSB(AP)    = ADDRESS TO HOLD LENGTH OF INPUT LINE
0319  1014 ;       QIO$_P2(AP)      = LENGTH IN BYTES ALLOWED FOR THE BUFFER.
0319  1015 ;       QIO$_P4(AP)      = OPTIONAL ADDR. OF TERMINATOR DESCRIPTOR.      [17]
0319  1016 ;
0319  1017 ; IMPLICIT INPUTS:
0319  1018 ;
0319  1019 ;       NONE
0319  1020 ;
0319  1021 ; OUTPUT PARAMETERS:
0319  1022 ;
0319  1023 ;       R1 = LENGTH OF LINE INPUT
0319  1024 ;       SUCCESS INDICATOR IN R0, R0 = 0 IF '?' TYPED ELSE R0 = 1
0319  1025 ;
0319  1026 ; IMPLICIT OUTPUTS:
0319  1027 ;
0319  1028 ;       NONE
0319  1029 ;
0319  1030 ; COMPLETION CODES:
0319  1031 ;
0319  1032 ;       NONE
0319  1033 ;
0319  1034 ; SIDE EFFECTS:
0319  1035 ;
0319  1036 ;       NONE
0319  1037 ;--
```

```
                                0319  1039 PREADVBLK::
                                0319  1040 READVBLK::
00000000'EF   00810000 8F   CA  0319  1041          BICL2   #DS$M_CTRLO ! DS$M_OUTPUT, -
                                0324  1042                    DS$GL_FLAGS             ; Clear ^O and output flag
            0C AC   00'8F   91  0324  1043          CMPB    #IO$_READPROMPT,QIO$_FUNC(AP) ; READ W/PROMPT?
                          22  12  0329  1044          BNEQ    10$                     ; Branch if not
              00000000'EF   95  032B  1045          TSTB    COLUMN                  ; At left margin?
                          13  12  0331  1046          BNEQ    6$                      ; Branch if not
          00000001'EF     0A  91  0333  1047          CMPB    #LF,LASTCHAR            ; Was last char a LF?
                          0A  13  033A  1048          BEQL    6$                      ; Branch if so
                              033C  1049
          50    00000014'EF   9E  033C  1050 3$:      MOVAB   T_CRLF,R0               ; Output CRLF
                        FF0D  30  0343  1051          BSBW    OUT_ASCIC               ; Type it
                              0346  1052
             50   2C AC   7D  0346  1053 6$:      MOVQ    QIO$_P5(AP),R0          ; GET ADDRESS,LENGTH OF PROMPT
                        FF09  30  034A  1054          BSBW    OUT_STRING              ; AND TYPE IT
                              034D  1055
             53   1C BC   9E  034D  1056 10$:     MOVAB   @QIO$_P1(AP), R3        ; SET BUFFER POINTER IN R3
                           52  D4  0351  1057          CLRL    R2                      ; INITIALIZE BYTE COUNTER IN R2
                              0353  1058
                        FDA6  30  0353  1059 20$:     BSBW    RINPUT                  ; GO GET A CHARACTER
                              0356  1060          Clear_CtrlO             ; Clear the ^O flag                     [16]
          00000000'EF     10  E4  0356                  BBSC    #DS$V_CtrlO, -          ; Branch if CTRLO bit is set to
                          00      035D                    L^DS$GL_Flags, 30014$ ; ...
                              035E          30014$:                         ; ... here. Clear bit regardless.
             50   80 8F   8A  035E  1061          BICB2   #PARITY, R0             ; CLEAR PARITY BIT
          00000004'EF     0C  E1  0362  1062          BBC     #TT$V_SCOPE, -          ; Scope or hardcopy terminal?   [26]
                          4D      0369  1063                  DS$GL_TERMCHAR+4,25$
                              036A  1064                                          ; Scope                         [26]
             50   7F 8F   91  036A  1065          CMPB    #DELETE, R0             ; IS THIS "DELETE"              [26]
                          7D  12  036E  1066          BNEQU   60$                     ; BRANCH IF NOT                 [26]
                          52  D5  0370  1067          TSTL    R2                      ; IS THIS BEGINNING OF BUFFER   [26]
                          03  12  0372  1068          BNEQ    23$                     ; BRANCH IF NOT BEGINNING       [26]
                        0132  31  0374  1069          BRW     140$                    ; GO TYPE BELL IF BEGINNING     [26]
                              0377  1070
00000002'EF   00000000'EF   91  0377  1071 23$:      CMPB    COLUMN,DS$GB_WIDTH      ; At the end of the line?       ;G:6
                          13  1F  0382  1072          BLSSU   24$                     ; Branch if not                 ;G:6
          00000000'EF     02  82  0384  1073          SUBB2   #2,COLUMN               ; SP char increments COLUMN, so subtract[26]
                              038B  1074                                          ; 1 for that, another 1 for the char    [26]
                              038B  1075                                          ; we are deleting               [26]
                          53  D7  038B  1076          DECL    R3                      ; Backup buffer pointer         [37]
                    51    20  9A  038D  1077          MOVZBL  #SPACE,R1               ; Load space char               [37]
                        FED5  30  0390  1078          BSBW    OUT_CHAR                ; Type space                    [37]
                          52  D7  0393  1079          DECL    R2                      ; Adjust character counter      [37]
                          BC  11  0395  1080          BRB     20$                     ; Get another character         [37]
          00000000'EF     02  82  0397  1081 24$:      SUBB2   #2,COLUMN               ; SP char increments COLUMN, so subtract[26]
                              039E  1082                                          ; 1 for that, another 1 for the char    [26]
                              039E  1083                                          ; we are deleting               [26]
                          53  D7  039E  1084          DECL    R3                      ; Backup buffer pointer         [36]
                    51    08  9A  03A0  1085          MOVZBL  #BS,R1                  ; Load backspace char           [26]
                        FEC2  30  03A3  1086          BSBW    OUT_CHAR                ; Type backspace                [26]
                    51    20  9A  03A6  1087          MOVZBL  #SPACE,R1               ; Load space char               [26]
                        FEBC  30  03A9  1088          BSBW    OUT_CHAR                ; Type space                    [26]
                    51    08  9A  03AC  1089          MOVZBL  #BS,R1                  ; Load backspace char           [26]
                        FEB6  30  03AF  1090          BSBW    OUT_CHAR                ; Type backspace                [26]
                          52  D7  03B2  1091          DECL    R2                      ; Adjust character counter      [26]
                        FF9C  31  03B4  1092          BRW     20$                     ; Get another character         [26]
```

```
                                03B7
          50    7F 8F    91    03B7  25$:    CMPB    #DELETE, R0                ; IS THIS "DELETE"                          [26]
                   21    12    03BB          BNEQU   50$                        ; BRANCH IF NOT                             [26]
                   52    D5    03BD          TSTL    R2                         ; IS THIS BEGINNING OF BUFFER               [26]
                   03    12    03BF          BNEQ    30$                        ; BRANCH IF NOT BEGINNING                   [26]
                 00E5    31    03C1          BRW     140$                       ; GO TYPE BELL IF BEGINNING                 [26]
                                03C4
   00000000'EF    05    E2    03C4  30$:    BBSS    #DS$V_RUBFLG, -            ; BRANCH IF BSLASH ALREADY TYPED
                   07          03CB                    DS$GL_FLAGS, 40$
          51    5C 8F    9A    03CC          MOVZBL  #BSLASH, R1                ; LOAD BACK SLASH INTO R1
                 FE95    30    03D0          BSBW    OUT_CHAR                   ; TYPE BACK SLASH
                                03D3
             51    73    9A    03D3  40$:    MOVZBL  -(R3), R1                  ; LOAD DELETED CHARACTER INTO R1
                 FE8F    30    03D6          BSBW    OUT_CHAR                   ; TYPE DELETED CHARACTER
                   52    D7    03D9          DECL    R2                         ; ADJUST CHARACTER COUNTER
                 FF75    31    03DB          BRW     20$                        ; GO GET ANOTHER CHARACTER                  ;G:6
                                03DE
   00000000'EF    05    E5    03DE  50$:    BBCC    #DS$V_RUBFLG, -            ; Branch if last char not "DEL"
                   07          03E5                    DS$GL_FLAGS, 60$
          51    5C 8F    9A    03E6          MOVZBL  #BSLASH, R1                ; LOAD BACK SLASH INTO R1
                 FE7B    30    03EA          BSBW    OUT_CHAR                   ; TYPE BACK SLASH
                                03ED
             50    15    91    03ED  60$:    CMPB    #CNTRLU, R0                ; IS THIS CONTROL U
                   0D    12    03F0          BNEQU   70$                        ; BRANCH IF NOT
   50   00000007'EF    9E    03F2          MOVAB   L^T_CNTRLU,R0              ; ADDRESS OF TEXT                           [14]
                 FE57    30    03F9          BSBW    OUT_ASCIC                  ; TYPE IT
                 FF1A    31    03FC          BRW     PREADVBLK                  ; GO INITIALIZE BUFFER
                                03FF
             50    03    91    03FF  70$:    CMPB    #CNTRLC, R0                ; IS THIS CONTROL C
                   0D    12    0402          BNEQU   90$                        ; BRANCH IF NOT
             53    50    9A    0404          MOVZBL  R0, R3                     ; Set terminator character
                   52    D4    0407          CLRL    R2                         ; FLUSH BUFFER.
      50   0000'8F    3C    0409          MOVZWL  #SS$_CONTROLC,R0           ; SET RETURN CODE
                 00AF    31    040E          BRW     170$
                                0411
             50    12    91    0411  90$:    CMPB    #CNTRLR, R0                ; IS THIS CONTROL R
                   17    12    0414          BNEQU   110$                       ; BRANCH IF NOT
                   52    D5    0416          TSTL    R2                         ; IS THIS BEGINNING OF BUFFER
                   03    12    0418          BNEQU   100$                       ; NO                                    [17]
                 008C    31    041A          BRW     140$                       ; GO TYPE BELL IF BEGINNING             [17]
                                041D  100$:                                      ;                                       [17]
   50   0000000C'EF    9E    041D          MOVAB   L^T_CNTRLR,R0              ; OUTPUT "^R<CR>"                           [14]
                 FE2C    30    0424          BSBW    OUT_ASCIC
                 FDCA    30    0427          BSBW    RTYPE                      ; Re-type input line
                 FF26    31    042A          BRW     20$                        ; BRANCH TO GET NEXT CHAR
                                042D
                                042D  110$:
                                042D  ;[23]   CMPB    #QUESTN, R0                ; IS THIS A QUESTION MARK
                                042D  ;[23]   BNEQU   120$                       ; BRANCH IF NOT
                                042D  ;[23]   MOVZBL  R0, R1                     ; OUTPUT QUESTION MARK
                                042D  ;[23]   BSBW    OUT_CHAR                   ; TYPE QUESTION MARk
                                042D  ;[23]   MOVZBL  R0, R3                     ; Set terminator character
                                042D  ;[23]   MOVL    #CR,R1                     ; RETURN THE CARRIAGE
                                042D  ;[23]   BSBW    OUT_CHAR
                                042D  ;[23]   CLRL    R0                         ; SET ERROR RETURN
                                042D  ;[23]   BRW     170$                       ; BRANCH TO EXIT                        [17]
                                042D
```

```
                50    0D   91   042D  1150 120$:   CMPB    #CR, R0                 ; IS THIS CARRIAGE RETURN
                      17   12   0430  1151         BNEQU   130$                    ; BRANCH IF NOT
                63    50   90   0432  1152         MOVB    R0,(R3)                 ; STORE THE CHARACTER... MAY BE IGNORED
                51    50   D0   0435  1153         MOVL    R0,R1                   ; COPY FOR ECHO
                   FE2D    30   0438  1154         BSBW    OUT_CHAR                ; ECHO <CR>
                51    0A   D0   043B  1155         MOVL    #LF,R1
                   FE27    30   043E  1156         BSBW    OUT_CHAR                ; TYPE <LF>
                50    01   D0   0441  1157         MOVL    #1,R0                   ; ASSUME IT'S <CR> AND SET SUCCESS
                53    0D   9A   0444  1158         MOVZBL  #CR, R3                 ; Set terminator character
                      77   11   0447  1159         BRB     170$                    ; Exit
                                0449  1160
                                0449  1161 130$:                                  ;                                       [17]
             54   28 AC   D0   0449  1162         MOVL    QIO$_P4(AP),R4          ;IF TERMINATOR DESCRIPTOR PASSED        [17]
                      28   13   044D  1163         BEQL    4000$                   ;THEN                                   [17]
                      64   D5   044F  1164         TSTL    (R4)                    ;   IF SHORT FORM                       [17]
                      08   12   0451  1165         BNEQ    1000$                   ;   THEN (SEE VMS I/O USERS GDE)        [17]
                56    1F   D0   0453  1166         MOVL    #31,R6                  ;      INIT BIT POINTER TO MAX          [17]
                54    04   C0   0456  1167         ADDL2   #4,R4                   ;      GET ADDR. OF BIT PATTERN         [17]
                      0C   11   0459  1168         BRB     2000$                   ;   ELSE                                [17]
                56    64   D0   045B  1169 1000$:  MOVL    (R4),R6                 ;      GET BYTE COUNT OF BIT PAT        [17]
                56    08   C4   045E  1170         MULL2   #8,R6                   ;      CONVERT TO BIT POINTER           [17]
                54    04   C0   0461  1171         ADDL2   #4,R4                   ;      GET ADDR. OF BIT PAT. PTR        [17]
                54    64   D0   0464  1172         MOVL    (R4),R4                 ;      GET ADDR. OF BIT PATTERN         [17]
                                0467  1173 2000$:                                  ;   REPEAT
                64    56   E1   0467  1174         BBC     R6,(R4),-               ;      IF THIS BIT IS SET               [17]
                      05        046A  1175                 3000$                   ;      THEN                             [17]
                56    50   91   046B  1176         CMPB    R0,R6                   ;         IF BIT NO. = INP. CHR ASCII   [17]
                      25   13   046E  1177         BEQL    5000$                   ;         THEN LEAVE LOOP               [17]
                                0470  1178 3000$:                                  ;      ELSE                             [17]
                      56   D7   0470  1179         DECL    R6                      ;         DECR. BIT POINTER             [17]
                      F3   18   0472  1180         BGEQ    2000$                   ;   UNTIL ENTIRE BIT PAT. DONE          [17]
                   002C    31   0474  1181         BRW     6000$                   ;   CHR NOT TERMINATOR, SO CONT.         [17]
                                0477  1182 4000$:                                  ;ELSE                                   [17]
             50    1F   91   0477  1183         CMPB    #31,R0                  ;   IF INP. CHR ASCII LSS 31            [17]
                   27   1B   047A  1184         BLEQU   6000$                   ;   THEN NOT TERMINATOR                 [17]
             50    0A   91   047C  1185         CMPB    #LF,R0                  ;   IF INP. CHR IS LINEFEED             [17]
                   22   13   047F  1186         BEQL    6000$                   ;   THEN NOT TERMINATOR                 [17]
             50    0B   91   0481  1187         CMPB    #VT,R0                  ;   IF INP. CHR IS VT                   [17]
                   1D   13   0484  1188         BEQL    6000$                   ;   THEN NOT TERMINATOR                 [17]
             50    0C   91   0486  1189         CMPB    #FF,R0                  ;   IF INP. CHR IS FORMFEED             [17]
                   18   13   0489  1190         BEQL    6000$                   ;   THEN NOT TERMINATOR                 [17]
             50    09   91   048B  1191         CMPB    #TAB,R0                 ;   IF INP. CHR IS TAB                  [17]
                   13   13   048E  1192         BEQL    6000$                   ;   THEN NOT TERMINATOR                 [17]
             50    08   91   0490  1193         CMPB    #BS,R0                  ;   IF INP. CHR IS BACKSPACE            [17]
                   0E   13   0493  1194         BEQL    6000$                   ;   THEN NOT TERMINATOR                 [17]
                                0495  1195 5000$:                                  ;TERMINATOR HAS BEEN FOUND              [17]
             51    50   D0   0495  1196         MOVL    R0,R1                   ;PUT CHAR INTO R1                       [17]
                FDCD    30   0498  1197         BSBW    OUT_CHAR                ;PRINT IT                               [17]
             83    50   D0   049B  1198         MOVL    R0,(R3)+                ;STORE IT                               [17]
                   52   D6   049E  1199         INCL    R2                      ;COUNT IT                               [17]
                001D    31   04A0  1200         BRW     170$                    ;EXIT                                   [17]
                                04A3  1201 6000$:                                  ;                                       [17]
          20 AC    52   D1   04A3  1202         CMPL    R2, QIO$_P2(AP)         ; IS THIS LAST BYTE IN BUFFER
                   09   19   04A7  1203         BLSS    150$                    ; BRANCH IF NOT LAST CHARACTER
                                04A9  1204
             51    07   9A   04A9  1205 140$:   MOVZBL  #BELL, R1               ; LOAD "BELL" CODE INTO R1
                FDB9    30   04AC  1206         BSBW    OUT_CHAR                ; TYPE "BELL"
```

```
                FEA1   31  04AF  1207          BRW     20$                      ; TRY FOR SOMETHING ELSE
                           04B2  1208
            51    50   9A  04B2  1209 150$:    MOVZBL  R0, R1                   ; LOAD CHARACTER INTO R1
                FDB0   30  04B5  1210          BSBW    OUT_CHAR                 ; ECHO CHARACTER
            83    50   90  04B8  1211          MOVB    R0, (R3)+                ; STORE CHARACTER IN BUFFER
                  52   D6  04BB  1212          INCL    R2                       ; COUNT THIS CHARACTER
                FE93   31  04BD  1213          BRW     20$                      ; GO GET ANOTHER CHARACTER
                           04C0  1214
        51    10 BC   DE  04C0  1215 170$:    MOVAL   @QIO$_IOSB(AP), R1       ; GET ADDRESS OF I/O STATUS BLOCK.
            81    50   B0  04C4  1216          MOVW    R0, (R1)+                ; STORE OPERATION STATUS
            81    52   B0  04C7  1217          MOVW    R2, (R1)+                ; Store character count.
            81    53   B0  04CA  1218          MOVW    R3, (R1)+                ; Store terminator char.
            61    01   B0  04CD  1219          MOVW    S^#1, (R1)               ; Store terminator size
                           04D0  1220          $SETEF_G (AP)                    ; SET EVENT FLAG.
  00000000'GF     6C   FA  04D0                        CALLG   (AP),G^SYS$SETEF
                       05  04D7  1221          RSB
                           04D8  1222 ;        RET                              ; RETURN TO QIO ENTRY VECTOR.   [30]
                           04D8  1223
```

```
                     1225 .sbttl SIZE_TERM Routine to size the console terminal
                     1226
                     1227 ;++
                     1228 ; FUNCTIONAL DESCRIPTION:                                                    [23]
                     1229 ;                                                                           [23]
                     1230 ;       This routine will size the console terminal to determine its        [23]
                     1231 ;       characteristics.  The routine is meant for standalone use,          [23]
                     1232 ;       and attempts to provide some of the same information that           [23]
                     1233 ;       can be obtained for timesharing terminals in user mode              [23]
                     1234 ;       using the $GETCHN service.  The routine sets the "type"             [23]
                     1235 ;       and "terminal characteristics" fields of DS$GL_TERMCHAR.            [23]
                     1236 ;       The fields are defined the the $TTDEF macro of VMS.                 [23]
                     1237 ;                                                                           [23]
                     1238 ;       The format of DS$GL_TERMCHAR is as follows:                         [23]
                     1239 ;                                                                           [23]
                     1240 ;       32                                  0                               [23]
                     1241 ;       .....................................                               [23]
                     1242 ;       : page          :       :           :                               [23]
                     1243 ;       :   width       : type  : class     :                               [23]
                     1244 ;       :...............:.......:...........:                               [23]
                     1245 ;       :page   :    terminal               :                               [23]
                     1246 ;       : length:     characteristics       :                               [23]
                     1247 ;       :.......:.............................:                               [23]
                     1248 ;                                                                           [23]
                     1249 ;       NOTE: Currently, the routine only sets the TT$M_SCOPE               [23]
                     1250 ;       characteristic.  Other characteristics defined by $TTDEF,           [23]
                     1251 ;       or other fields in DS$GL_TERMCHAR, could be also be sized for.      [23]
                     1252 ;                                                                           [23]
                     1253 ;                                                                           [23]
                     1254 ; CALLING SEQUENCE:                                                         [23]
                     1255 ;                                                                           [23]
                     1256 ;       BSBW    SIZE_TERM                                                   [23]
                     1257 ;                                                                           [23]
                     1258 ; INPUT PARAMETERS:                                                         [23]
                     1259 ;                                                                           [23]
                     1260 ;       NONE                                                                [23]
                     1261 ;                                                                           [23]
                     1262 ; IMPLICIT INPUTS:                                                          [23]
                     1263 ;                                                                           [23]
                     1264 ;       DS$GL_TERMCHAR                                                      [23]
                     1265 ;                                                                           [23]
                     1266 ; OUTPUT PARAMETERS:                                                        [23]
                     1267 ;                                                                           [23]
                     1268 ;       NONE                                                                [23]
                     1269 ;                                                                           [23]
                     1270 ; IMPLICIT OUTPUTS:                                                         [23]
                     1271 ;                                                                           [23]
                     1272 ;       DS$GL_TERMCHAR                                                      [23]
                     1273 ;                                                                           [23]
                     1274 ; COMPLETION CODES:                                                         [23]
                     1275 ;                                                                           [23]
                     1276 ;                                                                           [23]
                     1277 ;                                                                           [23]
                     1278 ; SIDE EFFECTS:                                                             [23]
                     1279 ;                                                                           [23]
                     1280 ;                                                                           [23]
                     1281 ;                                                                           [23]
```

```
                                  04D8  1282 ;--                                                                  [23]
                                  04D8  1283
                     5A 1B 00'    04D8  1284 DEC_PROMPT:     .ASCIC  <27><90>        ;DEC prompt                  [23]
                           02     04D8
                  63 5B 1B 00'    04DB  1285 ANSI_PROMPT:    .ASCIC  <27>'[c'        ;ANSI prompt                 [23]
                           03     04DB
                        05 00'    04DF  1286 LA36_PROMPT:    .ASCIC  <5>             ;LA36 prompt                 [23]
                           01     04DF
                                  04E1  1287
                                  04E1  1288 PROMPTS:                                ; Pointers to prompt strings   [23]
                     000004D8'    04E1  1289         .ADDRESS DEC_PROMPT
                     000004DB'    04E5  1290         .ADDRESS ANSI_PROMPT
                     000004DF'    04E9  1291         .ADDRESS LA36_PROMPT
                     00000000     04ED  1292         .LONG   0
                                  04F1  1293
                     00000014     04F1  1294 RECVD_BUF_LENGTH = 20                   ; (Longest response string will [23]
                                  04F1  1295                                         ;  probably be from LA36, which [23]
                                  04F1  1296                                         ;  can be up to 20 chars. But we[23]
                                  04F1  1297                                         ;  don't know what LA36 string  [23]
                                  04F1  1298                                         ;  is anyway;it is user-defined)[23]
                     00000505     04F1  1299 RECVD_BUF:      .BLKB RECVD_BUF_LENGTH  ; Buffer to receive response   [23]
2E 31 3A 3A 20 30 0000050D'010E0000' 0505  1300 MAX_WAIT:       .ASCID  /0 ::1.5/       ; Max. time to wait for response[23]
                           35     0513
                     0000051C     0514  1301 QUAD_TIME:      .BLKQ   1               ; Converted binary time        [23]
                                  051C  1302
                        01FC      051C  1303 .ENTRY SIZE_TERM, ^M<R2,R3,R4,R5,R6,R7,R8>;                          [23]
                                  051E  1304
                                  051E  1305         $BINTIM_S MAX_WAIT, QUAD_TIME   ; Convert ASCII time to binary  [23]
                  F3 AF    7F     051E                     PUSHAQ  QUAD_TIME
                  E1 AF    7F     0521                     PUSHAQ  MAX_WAIT
          00000000'GF   02   FB   0524                     CALLS   #2,G^SYS$BINTIM
               00000000'EF   7C   052B  1306         CLRQ    DS$GL_TERMCHAR          ; Initialize DS$GL_TERMCHAR    [23]
                57   AD AF   9E   0531  1307         MOVAB   PROMPTS, R7             ; Get list of prompts          [23]
                   58   87   D0   0535  1308 10$:    MOVL    (R7)+,R8                ; Get address of next prompt   [23]
                        03   12   0538  1309         BNEQ    15$                     ; Any more remaining?          [23]
                      0085   31   053A  1310         BRW     40$                     ; Branch if no more prompts    [23]
                                  053D  1311 15$:    $SETIMR_S       #1,QUAD_TIME    ; Start timer.  (This MUST be  [23]
                        7E   7C   053D                     CLRQ    -(SP)
                  D2 AF    7F     053F                     PUSHAQ  QUAD_TIME
                        01   DD   0542                     PUSHL   #1
          00000000'GF   04   FB   0544                     CALLS   #4,G^SYS$SETIMR
                                  054B  1312                                         ;  done before the prompt is   [23]
                                  054B  1313                                         ;  sent, because the $SETIMR   [23]
                                  054B  1314                                         ;  service calls KB_CHECK, which[23]
                                  054B  1315                                         ;  will read KB input and throw [23]
                                  054B  1316                                         ;  it away.)                   [23]
                   50   58   D0   054B  1317         MOVL    R8,R0                   ; Put prompt addr. in R0       [23]
                     FCDB   30   054E  1318         BSBW    OUT_CNTRL               ; Send prompt to terminal      [23]
                  9D AF    94     0551  1319         CLRB    RECVD_BUF               ; Clear 1st byte of response buf[23]
                58   9A AF   9E   0554  1320         MOVAB   RECVD_BUF,R8            ; Init response buffer pointer [23]
                                  0558  1321 20$:                                    ; REPEAT
                                  0558  1322 30$:    $READEF_S       #1,EVENT_FLAGS  ; See if time was used up      [23]
               00000004'EF   DF   0558                     PUSHAL  EVENT_FLAGS
                        01   DD   055E                     PUSHL   #1
          00000000'GF   02   FB   0560                     CALLS   #2,G^SYS$READEF
          50   00000000'8F   D1   0567  1323         CMPL    #SS$_WASSET,R0          ; Has time run out?            [23]
                        19   13   056E  1324         BEQL    35$                     ; Yes.  No response.           [23]
```

```
                                0570  1325          CMK     RCONSOLE                ; Get a character of response   [23]
                          7E    DC  0570            MOVPSL  -(SP)
                    06 6E 1A    E0  0572            BBS     #PSL$V_IS, (SP), 30015$
                          8E    D4  0576            CLRL    (SP)+
                          01    BC  0578            CHMK    #CMK$_RCONSOLE
                          04    11  057A            BRB     30016$
                       FBB2 CF  16  057C      30015$: JSB     CMK$RCONSOLE
                                0580      30016$:
                    D4 50    07  E1  0580  1326      BBC     #7,R0,30$               ; If no character, try again    [23]
                       88    51  90  0584  1327      MOVB    R1,(R8)+                ; Store char. in response buffer[23]
                          CF    11  0587  1328      BRB     30$                     ; Get another character         [23]
                       FF64 CF  95  0589  1329 35$:  TSTB    RECVD_BUF               ; IF no characters were received[23]
                          A6    13  058D  1330      BEQL    10$                     ; THEN try another prompt       [23]
                                058F  1331
                                058F  1332 ;
                                058F  1333 ; A response was received.  Now match the response to a terminal type.  [23]
                                058F  1334 ;
                                058F  1335
                58   00001000 8F    D0  058F  1336      MOVL    #TT$M_SCOPE, R8         ; Assume video terminal         [23]
                54   0000009A'EF    9E  0596  1337      MOVAB   VIDEO_TYPES, R4         ; Point to video response list  [23]
                           0023    30  059D  1338      BSBW    100$                    ; Search list                   [23]
                         1F 50     E8  05A0  1339      BLBS    R0,40$                  ; Found a match                 [23]
                            58     D4  05A3  1340      CLRL    R8                      ; Not a video terminal          [23]
                54   000001F8'EF    9E  05A5  1341      MOVAB   HARDCOPY_TYPES, R4      ; Point to hardcopy response list[23]
                           0014    30  05AC  1342      BSBW    100$                    ; Search list                   [23]
                        10 50      E8  05AF  1343      BLBS    R0,40$                  ; Found a match                 [23]
00000000'EF   08   08    20      F0  05B2  1344      INSV    #TT$_LA36,#8,#8, -      ; No match, so chances are that [23]
                                05BB  1345              DS$GL_TERMCHAR          ;  the terminal is an LA36, since[23]
          00000004'EF    58      D0  05BB  1346      MOVL    R8,DS$GL_TERMCHAR+4     ;  they do not return a standard[23]
                                05C2  1347                                      ;  response.                    [23]
                            04  05C2  1348 40$:     RET                             ; BYE.                          [23]
                                05C3  1349
                                05C3  1350 100$:    ;This code searches a list of terminal responses and compares   [23]
                                05C3  1351          ;them to the received response, looking for a match.  If a   [23]
                                05C3  1352          ;match is found, DS$GL_TERMCHAR is set up appropriately.     [23]
                                05C3  1353
                          50    D4  05C3  1354      CLRL    R0                      ; Clear return status           [23]
                       FF 8F    64  91  05C5  1355      CMPB    (R4), #-1               ; End of list?                  [23]
                          28    13  05C9  1356      BEQL    300$                    ; Yes                           [23]
                       55       84  9A  05CB  1357      MOVZBL  (R4)+, R5               ; Get length                    [23]
                FF1D CF 64      55  29  05CE  1358      CMPC3   R5,(R4),RECVD_BUF       ; Compare strings               [23]
                          07    13  05D4  1359      BEQL    200$                    ; Found a match                 [23]
                       54       55  C0  05D6  1360      ADDL    R5,R4                   ; Go to end of string           [23]
                          54    D6  05D9  1361      INCL    R4                      ; Skip over type code           [23]
                          E6    11  05DB  1362      BRB     100$                    ; Try next response in list     [23]
                       54       55  C0  05DD  1363 200$: ADDL    R5,R4                   ; Point to type code            [23]
00000000'EF   08   08    64      F0  05E0  1364      INSV    (R4),#8,#8,DS$GL_TERMCHAR; Store type code              [23]
          00000004'EF    58      D0  05E9  1365      MOVL    R8,DS$GL_TERMCHAR+4     ; Store characteristics         [23]
                       50       01  D0  05F0  1366      MOVL    #1,R0                   ; Set good return status        [23]
                                05  05F3  1367 300$:    RSB                             ; Leave                         [23]
                                05F4  1368
                                05F4  1369          .END
```

```
$$ARGS                          = 0000000C    D
$$N                             = 00000002    D
$$T1                            = 00000000    D
ANSI_PROMPT                       000004DB R  D  04
BELL                            = 00000007    D
BIT...                          = 00000004    D
BS                              = 00000008    D
BSLASH                          = 0000005C    D
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        0000044C    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_FLAGS2                      00000444    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEMP_BASE                   00000448    D
CLI$L_TEST                        00000020    D
CLI$M_START_OR_RUN              = 00000008    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BASE_QUAL                 = 00000002    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NEXT                      = 00000001    D
CLI$V_NOALLOC                   = 00000000    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_START_OR_RUN              = 00000003    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
CMK$RCONSOLE                      00000132 RG D  04
CMK$TCONSOLE                      000002EE RG D  04
CMK$_APBOOT                     = 00000004    D
CMK$_ASTEXIT                    = 00000000    D
CMK$_CANTIM                     = 0000000B    D
CMK$_HIBER                      = 00000007    D
CMK$_INITSCB                    = 00000008    D
CMK$_LAST                       = 0000000E    D
CMK$_MMENABLE                   = 00000003    D
CMK$_RCONSOLE                   = 00000001    D
CMK$_REFRESH                    = 00000005    D
CMK$_SCHDWK                     = 00000006    D
CMK$_SETIMR                     = 0000000C    D
CMK$_SETIPL                     = 00000009    D
CMK$_SETPRT                     = 0000000D    D
CMK$_SGIPR                      = 0000000A    D
CMK$_TCONSOLE                   = 00000002    D
CNTRLC                          = 00000003    D
CNTRLO                          = 0000000F    D
CNTRLQ                          = 00000011    D
CNTRLR                          = 00000012    D
CNTRLS                          = 00000013    D
CNTRLU                          = 00000015    D
COLUMN                            00000000 R  D  02
CONSFLG                           00000003 R  D  02
CR                              = 0000000D    D
CRD$M_CRD_DEBUG                 = 00000001    D
CRD$M_CTRL_C_NO_ECHO            = 00000002    D
CRD$M_FLUSH_CTRL_C              = 00000004    D
CRD$M_INHIBIT_ALLOCATE          = 00000008    D
CRD$V_CRD_DEBUG                 = 00000000    D
CRD$V_CTRL_C_NO_ECHO            = 00000001    D
CRD$V_FLUSH_CTRL_C              = 00000002    D
CRD$V_INHIBIT_ALLOCATE          = 00000003    D
DEC_PROMPT                        000004D8 R  D  04
DELETE                          = 0000007F    D
DS$GB_SYSTYPE                     ********    X  04
DS$GB_WIDTH                       00000002 RG D  02
DS$GL_CRD_FLAGS                   ********    X  04
DS$GL_FLAGS                       ********    X  04
DS$GL_TERMCHAR                    ********    X  04
DS$GW_TTOUT                       ********    X  04
DS$K_PRINTB                     = 00000002    D
DS$K_PRINTF                     = 00000001    D
DS$K_PRINTI                     = 00000000    D
DS$K_PRINTX                     = 00000003    D
DS$K_TYPE_ABORT_PROGRAM         = 00000014    D
DS$K_TYPE_ABORT_TEST            = 00000013    D
DS$K_TYPE_COMMAND_ERR           = 00000015    D
DS$K_TYPE_COMMAND_OUT           = 00000016    D
```

```
DS$K_TYPE_CRD_AUTOTEST          = 0000001A    D
DS$K_TYPE_DS_PROMPT             = 00000001    D
DS$K_TYPE_DS_START              = 0000001D    D
DS$K_TYPE_ERRDEV                = 00000008    D
DS$K_TYPE_ERRHARD               = 00000006    D
DS$K_TYPE_ERROR_BODY            = 00000009    D
DS$K_TYPE_ERROR_END             = 0000000A    D
DS$K_TYPE_ERRPREP               = 0000001B    D
DS$K_TYPE_ERRSOFT               = 00000007    D
DS$K_TYPE_ERRSUP                = 00000004    D
DS$K_TYPE_ERRSYS                = 00000005    D
DS$K_TYPE_ERR_HALT              = 0000000D    D
DS$K_TYPE_EXCEPTION             = 0000000C    D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D
DS$K_TYPE_FIRST_PASS            = 00000011    D
DS$K_TYPE_GENERAL               = 00000000    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D
DS$K_TYPE_NO_TESTS              = 00000012    D
DS$K_TYPE_PARAM_ERROR           = 0000001C    D
DS$K_TYPE_PROGRAM_END           = 00000010    D
DS$K_TYPE_PROGRAM_INFO          = 00000017    D
DS$K_TYPE_PROGRAM_START         = 0000000F    D
DS$K_TYPE_QIO_INVADP            = 00000024    D
DS$K_TYPE_QIO_NODRIVER          = 00000022    D
DS$K_TYPE_QIO_WRONGVER          = 00000023    D
DS$K_TYPE_SCRIPT_ECHO           = 00000021    D
DS$K_TYPE_SCRIPT_PNF            = 0000001E    D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020    D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F    D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D
DS$K_TYPE_START_ERR             = 00000018    D
DS$K_TYPE_START_LIST            = 00000025    D
DS$K_TYPE_SUMMARY               = 0000000E    D
DS$K_TYPE_USER_PROMPT           = 00000002    D
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_NI                         = 04000000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSA$V_APT                       = 0000001F    D
DSA$V_NORPT                     = 0000001B    D
DSA$V_USER                      = 0000001C    D
DSX$PRINT                         ********    X   04
END_HARDCOPY_TYPES                00000221 R  D   03
```

```
END_VIDEO_TYPES                   000001F4 R   D  03
EVENT_FLAGS                       00000004 R   D  02
FF                              = 0000000C     D
HARDCOPY_TYPES                    000001F8 R   D  03
HYPHEN                          = 0000002D     D
IO$V_CANCTRLO                     ********     X  04
IO$_READPROMPT                    ********     X  04
IO$_SENSEMODE                     ********     X  04
IO$_SETMODE                       ********     X  04
KB_POLL                           000000C5 RG  D  04
KB_POLL_X                         000000F9 R   D  04
LA36_PROMPT                       000004DF R   D  04
LASTCHAR                          00000001 R   D  02
LF                              = 0000000A     D
LOWCASE                         = 00000020     D
MAX_WAIT                          00000505 R   D  04
OUT_ASCIC                         00000253 R   D  04
OUT_CHAR                          00000268 R   D  04
OUT_CHARX                         000002ED R   D  04
OUT_CNTRL                         0000022C R   D  04
OUT_ECHO                          00000273 R   D  04
OUT_STRING                        00000256 R   D  04
PARITY                          = 00000080     D
PR$_RXCS                        = 00000020     D
PR$_RXDB                        = 00000021     D
PR$_SYS_TYP800                    ********     X  04
PR$_TXCS                        = 00000022     D
PR$_TXDB                        = 00000023     D
PREADVBLK                         00000319 RG  D  04
PROMPTS                           000004E1 R   D  04
PSL$V_IS                        = 0000001A     D
QIO$_ASTADR                     = 00000014     D
QIO$_ASTPRM                     = 00000018     D
QIO$_CHAN                       = 00000008     D
QIO$_EFN                        = 00000004     D
QIO$_FUNC                       = 0000000C     D
QIO$_IOSB                       = 00000010     D
QIO$_NARGS                      = 0000000C     D
QIO$_P1                         = 0000001C     D
QIO$_P2                         = 00000020     D
QIO$_P3                         = 00000024     D
QIO$_P4                         = 00000028     D
QIO$_P5                         = 0000002C     D
QIO$_P6                         = 00000030     D
QUAD_TIME                         00000514 R   D  04
QUESTN                          = 0000003F     D
READVBLK                          00000319 RG  D  04
RECVD_BUF                         000004F1 R   D  04
RECVD_BUF_LENGTH                = 00000014     D
RINPUT                            000000FC R   D  04
RINPUTX                           00000131 R   D  04
RTYPE                             000001F4 R   D  04
SIZ...                          = 00000001     D
SIZE_TERM                         0000051C RG  D  04
SPACE                           = 00000020     D
SS$_CONTROLC                      ********     X  04
SS$_WASSET                        ********     X  04
SYS$BINTIM                        ********  GX   04
SYS$QIOW                          ********  GX   04
SYS$READEF                        ********  GX   04
SYS$SETEF                         ********  GX   04
SYS$SETIMR                        ********  GX   04
TAB                             = 00000009     D
TT$M_SCOPE                      = 00001000     D
TT$V_SCOPE                      = 0000000C     D
TT$_LA100                       = 00000025     D
TT$_LA12                        = 00000024     D
TT$_LA120                       = 00000021     D
TT$_LA34                        = 00000022     D
TT$_LA36                        = 00000020     D
TT$_VT100                       = 00000060     D
TT$_VT101                       = 00000061     D
TT$_VT102                       = 00000062     D
TT$_VT125                       = 00000064     D
TT$_VT132                       = 00000066     D
TT$_VT200_SERIES                = 0000006E     D
TT$_VT52                        = 00000040     D
TT$_VT55                        = 00000041     D
TT$_VT5X                        = 00000040     D
T_CNTRLC                          00000000 R   D  03
T_CNTRLO                          0000000F R   D  03
T_CNTRLR                          0000000C R   D  03
T_CNTRLU                          00000007 R   D  03
T_CRLF                            00000014 R   D  03
T_LA100                           00000200 R   D  03
T_LA12                            000001F8 R   D  03
T_LA120                           00000208 R   D  03
T_LA34                            0000020F R   D  03
T_LA34A                           00000218 R   D  03
T_LA36                            00000225 R   D  03
T_OUT_IS_MBX                      0000006D R   D  03
T_SHOW_WIDTH                      00000048 R   D  03
T_VT100                           0000012A R   D  03
T_VT100A                          000000C7 R   D  03
T_VT100B                          00000133 R   D  03
T_VT100C                          0000013C R   D  03
T_VT100D                          00000145 R   D  03
T_VT100E                          0000014E R   D  03
T_VT100F                          00000157 R   D  03
T_VT100G                          00000160 R   D  03
T_VT100H                          00000169 R   D  03
T_VT101                           000000CC R   D  03
T_VT102                           000000D5 R   D  03
T_VT125                           000000DC R   D  03
T_VT125A                          000000ED R   D  03
T_VT125B                          000000FE R   D  03
T_VT125C                          0000010F R   D  03
T_VT132                           0000019D R   D  03
T_VT132A                          000001A6 R   D  03
T_VT132B                          000001AF R   D  03
T_VT132C                          000001B8 R   D  03
T_VT132D                          000001C1 R   D  03
T_VT132E                          000001CA R   D  03
T_VT132F                          000001D3 R   D  03
```

```
T_VT132G                        000001DC R   D   03
T_VT220                         00000172 R   D   03
T_VT220A                        00000182 R   D   03
T_VT240                         00000189 R   D   03
T_VT50_A                        0000009A R   D   03
T_VT50_B                        0000009F R   D   03
T_VT50_C                        000000A4 R   D   03
T_VT50_H                        000000AE R   D   03
T_VT50_J                        000000B3 R   D   03
T_VT52_K                        000000B8 R   D   03
T_VT52_L                        000000BD R   D   03
T_VT52_M                        000000C2 R   D   03
T_VT55_E                        000000A9 R   D   03
T_VT61                          00000120 R   D   03
T_VT61A                         000001E5 R   D   03
T_VT61B                         000001EA R   D   03
T_VT61C                         000001EF R   D   03
T_VT71                          00000125 R   D   03
T_WIDTH                         00000017 R   D   03
VIDEO_TYPES                     0000009A R   D   03
VRSETWIDTH                      00000000 RG  D   04
VRSHOWWIDTH                     000000A5 RG  D   04
VT                            = 0000000B     D
WRITEVBLK                       000002FD RG  D   04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000226 (550.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	000005F4 (1524.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=0000000C	235 (1)	235 (1)
$$N	=00000002	601 (1)	#-544 (1) #-552 (1) 601 (1)
$$T1	=00000000	1311 (1)	1311 (1) 235 (1) 526 (1) 534 (1)
ANSI_PROMPT	000004DB-R	1285 (1)	1290 (1)
BELL	=00000007	248 (1)	#-1205 (1) #-668 (1)
BIT...	=00000004	239 (1)	236 (1) 237 (1) 238 (1) 239 (1)
BS	=00000008	249 (1)	#-1085 (1) #-1089 (1) #-1193 (1)
BSLASH	=0000005C	263 (1)	#-1102 (1) #-1112 (1)
CLI$K_BUFSIZ	=00000100	233 (1)	233 (1)
CLI$K_SIZE	0000044C	233 (1)	
CLI$L_ADDRESS	00000018	233 (1)	
CLI$L_COMMAND	00000004	233 (1)	
CLI$L_DATA	0000001C	233 (1)	#-514 (1) #-529 (1) #-537 (1)
CLI$L_FLAGS	00000000	233 (1)	
CLI$L_FLAGS2	00000444	233 (1)	
CLI$L_LAST	00000024	233 (1)	
CLI$L_NEXT	00000030	233 (1)	
CLI$L_PASS	0000002C	233 (1)	
CLI$L_SUBT	00000028	233 (1)	
CLI$L_TEMP_BASE	00000448	233 (1)	
CLI$L_TEST	00000020	233 (1)	
CLI$M_START_OR_RUN	=00000008	233 (1)	
CLI$Q_BUFQWD	00000034	233 (1)	
CLI$Q_FILE	00000008	233 (1)	
CLI$Q_SECTION	00000010	233 (1)	
CLI$Q_TIME	0000043C	233 (1)	
CLI$T_BUFFER	0000003C	233 (1)	
CLI$V_ADAPTER	=00000018	233 (1)	
CLI$V_ADR	=0000000B	233 (1)	
CLI$V_ASCII	=00000013	233 (1)	
CLI$V_BASE_QUAL	=00000002	233 (1)	
CLI$V_BREAK	=0000000A	233 (1)	
CLI$V_BRIEF	=0000001B	233 (1)	
CLI$V_BYTE	=0000000D	233 (1)	
CLI$V_CLEAR	=00000002	233 (1)	
CLI$V_DEC	=00000010	233 (1)	
CLI$V_DEFAULT	=0000000C	233 (1)	
CLI$V_DEPOSIT	=00000019	233 (1)	
CLI$V_EVENT	=00000008	233 (1)	
CLI$V_EXAM	=00000005	233 (1)	
CLI$V_FLAGS	=00000009	233 (1)	
CLI$V_HEX	=00000012	233 (1)	
CLI$V_KERNEL	=00000017	233 (1)	
CLI$V_LOAD	=00000006	233 (1)	
CLI$V_LONG	=0000000F	233 (1)	
CLI$V_NEXT	=00000001	233 (1)	
CLI$V_NOALLOC	=00000000	233 (1)	
CLI$V_NOTNUF	=00000001	233 (1)	
CLI$V_OCT	=00000011	233 (1)	
CLI$V_PREG	=0000001A	233 (1)	

```
CLI$V_QA                        =00000007     233     (1)
CLI$V_QACKLOOPLOOPS             =0000001C     233     (1)
CLI$V_QAERRORPRINTS             =0000001B     233     (1)
CLI$V_QAMULTIPLEPASS            =0000001F     233     (1)
CLI$V_QASUBTESTLOOPS            =0000001E     233     (1)
CLI$V_QATESTLOOPS               =0000001D     233     (1)
CLI$V_REG                       =00000014     233     (1)
CLI$V_REQUIRED                  =00000000     233     (1)
CLI$V_RUN                       =00000015     233     (1)
CLI$V_SET                       =00000003     233     (1)
CLI$V_SHOW                      =00000004     233     (1)
CLI$V_START_OR_RUN              =00000003     233     (1)       233     (1)
CLI$V_VALSEC                    =00000016     233     (1)
CLI$V_WORD                      =0000000E     233     (1)
CMK$RCONSOLE                     00000132-R   728     (1)       1325    (1)       651     (1)       717     (1)
CMK$TCONSOLE                     000002EE-R   939     (1)       873     (1)       933     (1)
CMK$_APBOOT                     =00000004     236     (1)
CMK$_ASTEXIT                    =00000000     236     (1)
CMK$_CANTIM                     =0000000B     236     (1)
CMK$_HIBER                      =00000007     236     (1)
CMK$_INITSCB                    =00000008     236     (1)
CMK$_LAST                       =0000000E     236     (1)
CMK$_MMENABLE                   =00000003     236     (1)
CMK$_RCONSOLE                   =00000001     236     (1)     #-1325    (1)     #-651     (1)     #-717     (1)
CMK$_REFRESH                    =00000005     236     (1)
CMK$_SCHDWK                     =00000006     236     (1)
CMK$_SETIMR                     =0000000C     236     (1)
CMK$_SETIPL                     =00000009     236     (1)
CMK$_SETPRT                     =0000000D     236     (1)
CMK$_SGIPR                      =0000000A     236     (1)
CMK$_TCONSOLE                   =00000002     236     (1)     #-873     (1)     #-933     (1)
CNTRLC                          =00000003     247     (1)     #-1121    (1)     #-659     (1)     #-745     (1)
CNTRLO                          =0000000F     255     (1)     #-662     (1)     #-779     (1)
CNTRLQ                          =00000011     256     (1)     #-655     (1)     #-774     (1)
CNTRLR                          =00000012     257     (1)     #-1128    (1)
CNTRLS                          =00000013     258     (1)     #-665     (1)     #-769     (1)
CNTRLU                          =00000015     259     (1)     #-1115    (1)
COLUMN                           00000000-R   272     (1)     #-1045    (1)     #-1071    (1)     #-1073    (1)     #-1081    (1)
                                                               #-875     (1)     #-909     (1)     #-911     (1)     #-924     (1)
                                                               #-930     (1)
CONSFLG                          00000003-R   278     (1)     #-747     (1)     #-771     (1)     #-776     (1)     #-781     (1)
                                                               #-903     (1)
CR                              =0000000D     254     (1)     #-1150    (1)     #-1158    (1)       286     (1)       289     (1)
                                                                 295     (1)       298     (1)     #-907     (1)
CRD$M_CRD_DEBUG                 =00000001     239     (1)
CRD$M_CTRL_C_NO_ECHO            =00000002     239     (1)
CRD$M_FLUSH_CTRL_C              =00000004     239     (1)
CRD$M_INHIBIT_ALLOCATE          =00000008     239     (1)
CRD$V_CRD_DEBUG                 =00000000     239     (1)
CRD$V_CTRL_C_NO_ECHO            =00000001     239     (1)     #-749     (1)
CRD$V_FLUSH_CTRL_C              =00000002     239     (1)     #-762     (1)     #-764     (1)     #-768     (1)
CRD$V_INHIBIT_ALLOCATE          =00000003     239     (1)
DEC_PROMPT                       000004D8-R   1284    (1)       1289    (1)
DELETE                          =0000007F     264     (1)     #-1065    (1)     #-1094    (1)
DS$GB_SYSTYPE                    00000000-XR                   #-719     (1)     #-738     (1)
DS$GB_WIDTH                      00000002-R   276     (1)     #-1071    (1)     #-537     (1)     #-596     (1)     #-911     (1)
DS$GL_CRD_FLAGS                  00000000-XR                     749     (1)       762     (1)       764     (1)       768     (1)
```

```
DS$GL_FLAGS                     00000000-XR                  #-1042  (1)    1060  (1)    1101  (1)    1111  (1)
                                                               714  (1)    754  (1)    755  (1)    765  (1)
                                                             #-784  (1)    811  (1)    896  (1)    989  (1)
DS$GL_TERMCHAR                  00000000-XR                  1063  (1)  #-1306  (1)    1345  (1)  #-1346  (1)
                                                            1364  (1)  #-1365  (1)
DS$GW_TTOUT                     00000000-XR                  #-526  (1)  #-534  (1)
DS$K_PRINTB                     =00000002     238    (1)
DS$K_PRINTF                     =00000001     238    (1)  #-544  (1)  #-552  (1)  #-601  (1)
DS$K_PRINTI                     =00000000     238    (1)
DS$K_PRINTX                     =00000003     238    (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     238    (1)
DS$K_TYPE_ABORT_TEST            =00000013     238    (1)
DS$K_TYPE_COMMAND_ERR           =00000015     238    (1)  #-544  (1)  #-552  (1)
DS$K_TYPE_COMMAND_OUT           =00000016     238    (1)  #-601  (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     238    (1)
DS$K_TYPE_DS_PROMPT             =00000001     238    (1)
DS$K_TYPE_DS_START              =0000001D     238    (1)
DS$K_TYPE_ERRDEV                =00000008     238    (1)
DS$K_TYPE_ERRHARD               =00000006     238    (1)
DS$K_TYPE_ERROR_BODY            =00000009     238    (1)
DS$K_TYPE_ERROR_END             =0000000A     238    (1)
DS$K_TYPE_ERRPREP               =0000001B     238    (1)
DS$K_TYPE_ERRSOFT               =00000007     238    (1)
DS$K_TYPE_ERRSUP                =00000004     238    (1)
DS$K_TYPE_ERRSYS                =00000005     238    (1)
DS$K_TYPE_ERR_HALT              =0000000D     238    (1)
DS$K_TYPE_EXCEPTION             =0000000C     238    (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     238    (1)
DS$K_TYPE_FIRST_PASS            =00000011     238    (1)
DS$K_TYPE_GENERAL               =00000000     238    (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     238    (1)
DS$K_TYPE_NO_TESTS              =00000012     238    (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     238    (1)
DS$K_TYPE_PROGRAM_END           =00000010     238    (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     238    (1)
DS$K_TYPE_PROGRAM_START         =0000000F     238    (1)
DS$K_TYPE_QIO_INVADP            =00000024     238    (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     238    (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     238    (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     238    (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     238    (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     238    (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     238    (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     238    (1)
DS$K_TYPE_START_ERR             =00000018     238    (1)
DS$K_TYPE_START_LIST            =00000025     238    (1)
DS$K_TYPE_SUMMARY               =0000000E     238    (1)
DS$K_TYPE_USER_PROMPT           =00000002     238    (1)
DS$M_ABRTFLG                    =00000040     237    (1)
DS$M_BADTIME                    =00100000     237    (1)
DS$M_BATCH                      =00400000     237    (1)
DS$M_BRKCLR                     =00001000     237    (1)
DS$M_BRKPT                      =00000800     237    (1)
DS$M_CHARFLG                    =00000100     237    (1)
DS$M_CMDFLG                     =00000080     237    (1)
DS$M_CTRLC                      =00000001     237    (1)
DS$M_CTRLO                      =00010000     237    (1)  #-1041  (1)  #-784  (1)
```

```
DS$M_DEVFLG                     =00000200      237     (1)
DS$M_DISABLCC                   =01000000      237     (1)
DS$M_DONFLG                     =00002000      237     (1)
DS$M_ERRFLG                     =00000010      237     (1)
DS$M_EXCEPT                     =00080000      237     (1)
DS$M_EXETST                     =00040000      237     (1)
DS$M_HLTFLG                     =00000008      237     (1)
DS$M_LODFLG                     =00000002      237     (1)
DS$M_MEMMGT                     =00008000      237     (1)
DS$M_NI                         =04000000      237     (1)
DS$M_OUTPUT                     =00800000      237     (1)     #-1041    (1)
DS$M_RUBFLG                     =00000020      237     (1)
DS$M_SCRIPT                     =00200000      237     (1)
DS$M_SETIMR                     =02000000      237     (1)
DS$M_STRFLG                     =00000004      237     (1)
DS$M_SUBT                       =00004000      237     (1)
DS$M_SYSFLG                     =00000400      237     (1)
DS$M_TIMED                      =02000000      237     (1)
DS$M_TIMED_OUT                  =08000000      237     (1)
DS$M_TIMRON                     =00020000      237     (1)
DS$V_ABRTFLG                    =00000006      237     (1)
DS$V_BADTIME                    =00000014      237     (1)
DS$V_BATCH                      =00000016      237     (1)
DS$V_BRKCLR                     =0000000C      237     (1)
DS$V_BRKPT                      =0000000B      237     (1)
DS$V_CHARFLG                    =00000008      237     (1)
DS$V_CMDFLG                     =00000007      237     (1)
DS$V_CTRLC                      =00000000      237     (1)     #-755     (1)     #-765     (1)
DS$V_CTRLO                      =00000010      237     (1)     #-1060    (1)     #-754     (1)     #-896     (1)     #-989     (1)
DS$V_DEVFLG                     =00000009      237     (1)
DS$V_DISABLCC                   =00000018      237     (1)
DS$V_DONFLG                     =0000000D      237     (1)
DS$V_ERRFLG                     =00000004      237     (1)
DS$V_EXCEPT                     =00000013      237     (1)
DS$V_EXETST                     =00000012      237     (1)
DS$V_HLTFLG                     =00000003      237     (1)
DS$V_LODFLG                     =00000001      237     (1)
DS$V_MEMMGT                     =0000000F      237     (1)
DS$V_NI                         =0000001A      237     (1)
DS$V_OUTPUT                     =00000017      237     (1)     #-713     (1)     #-818     (1)     #-825     (1)
DS$V_RUBFLG                     =00000005      237     (1)     #-1100    (1)     #-1110    (1)
DS$V_SCRIPT                     =00000015      237     (1)
DS$V_SETIMR                     =00000019      237     (1)
DS$V_STRFLG                     =00000002      237     (1)
DS$V_SUBT                       =0000000E      237     (1)
DS$V_SYSFLG                     =0000000A      237     (1)
DS$V_TIMED                      =00000019      237     (1)
DS$V_TIMED_OUT                  =0000001B      237     (1)
DS$V_TIMRON                     =00000011      237     (1)
DSA$GL_FLAGS                     0000FE00                       519     (1)       899     (1)       901     (1)
DSA$V_APT                       =0000001F                     #-899     (1)
DSA$V_NORPT                     =0000001B                     #-900     (1)
DSA$V_USER                      =0000001C                     #-519     (1)
DSX$PRINT                        00000000-XR                    544     (1)       552     (1)       601     (1)
END_HARDCOPY_TYPES               00000221-R     459     (1)
END_VIDEO_TYPES                  000001F4-R     435     (1)
EVENT_FLAGS                      00000004-R     280     (1)      1322     (1)
```

```
FF                              =0000000C     253   (1)     #-1189   (1)
HARDCOPY_TYPES                   000001F8-R   442   (1)       1341   (1)
HYPHEN                          =0000002D     261   (1)
IO$V_CANCTRLO                    00000000-XR                 #-987    (1)
IO$_READPROMPT                   00000000-XR                 #-1043   (1)
IO$_SENSEMODE                    00000000-XR                 #-526    (1)
IO$_SETMODE                      00000000-XR                 #-534    (1)
KB_POLL                          000000C5-R   649   (1)     #-895    (1)     #-904    (1)
KB_POLL_X                        000000F9-R   671   (1)     #-653    (1)     #-656    (1)     #-660    (1)     #-663    (1)
                                                           #-666    (1)
LA36_PROMPT                      000004DF-R   1286  (1)       1291   (1)
LASTCHAR                         00000001-R   274   (1)     #-1047   (1)     #-932    (1)
LF                              =0000000A     251   (1)     #-1047   (1)     #-1155   (1)     #-1185   (1)       286    (1)
                                                             289    (1)       295    (1)       298    (1)
LOWCASE                         =00000020     246   (1)
MAX_WAIT                         00000505-R   1300  (1)       1305   (1)
OUT_ASCIC                        00000253-R   879   (1)     #-1051   (1)     #-1118   (1)     #-1135   (1)     #-814    (1)
                                                           #-915    (1)
OUT_CHAR                         00000268-R   894   (1)     #-1078   (1)     #-1086   (1)     #-1088   (1)     #-1090   (1)
                                                           #-1103   (1)     #-1106   (1)     #-1113   (1)     #-1154   (1)
                                                           #-1156   (1)     #-1197   (1)     #-1206   (1)     #-1210   (1)
                                                           #-819    (1)     #-826    (1)     #-888    (1)
OUT_CHARX                        000002ED-R   935   (1)     #-896    (1)     #-901    (1)
OUT_CNTRL                        0000022C-R   868   (1)     #-1318   (1)     #-752    (1)     #-783    (1)
OUT_ECHO                         00000273-R   898   (1)     #-669    (1)
OUT_STRING                       00000256-R   882   (1)     #-1054   (1)     #-992    (1)
PARITY                          =00000080     245   (1)     #-1061   (1)
PR$_RXCS                        =00000020                   #-731    (1)
PR$_RXDB                        =00000021                   #-735    (1)
PR$_SYS_TYP800                   00000000-XR                 #-719    (1)     #-738    (1)
PR$_TXCS                        =00000022                   #-942    (1)
PR$_TXDB                        =00000023                   #-944    (1)
PREADVBLK                        00000319-R   1039  (1)     #-1119   (1)
PROMPTS                          000004E1-R   1288  (1)       1307   (1)
PSL$V_IS                        =0000001A     236   (1)     #-1325   (1)     #-651    (1)     #-717    (1)     #-873    (1)
                                                           #-933    (1)
QIO$_ASTADR                     =00000014     235   (1)
QIO$_ASTPRM                     =00000018     235   (1)
QIO$_CHAN                       =00000008     235   (1)
QIO$_EFN                        =00000004     235   (1)
QIO$_FUNC                       =0000000C     235   (1)     #-1043   (1)       988    (1)
QIO$_IOSB                       =00000010     235   (1)       1215   (1)
QIO$_NARGS                      =0000000C     235   (1)
QIO$_P1                         =0000001C     235   (1)       1056   (1)       821    (1)     #-991    (1)
QIO$_P2                         =00000020     235   (1)     #-1202   (1)
QIO$_P3                         =00000024     235   (1)
QIO$_P4                         =00000028     235   (1)     #-1162   (1)
QIO$_P5                         =0000002C     235   (1)     #-1053   (1)     #-815    (1)
QIO$_P6                         =00000030     235   (1)
QUAD_TIME                        00000514-R   1301  (1)       1305   (1)       1311   (1)
QUESTN                          =0000003F     262   (1)
READVBLK                         00000319-R   1040  (1)
RECVD_BUF                        000004F1-R   1299  (1)     #-1319   (1)       1320   (1)     #-1329   (1)       1358   (1)
RECVD_BUF_LENGTH                =00000014     1294  (1)       1299   (1)
RINPUT                           000000FC-R   712   (1)     #-1059   (1)     #-718    (1)     #-722    (1)
RINPUTX                          00000131-R   725   (1)
RTYPE                            000001F4-R   810   (1)     #-1136   (1)     #-715    (1)
```

```
SIZ...                          =00000001     239   (1)      237   (1)      239   (1)
SIZE_TERM                        0000051C-R   1303   (1)
SPACE                           =00000020     260   (1)    #-1077  (1)    #-1087  (1)    #-921   (1)
SS$_CONTROLC                     00000000-XR          #-1125  (1)
SS$_WASSET                       00000000-XR          #-1323  (1)
SYS$BINTIM                       00000000-XR            1305  (1)
SYS$QIOW                         00000000-XR             526  (1)      534   (1)
SYS$READEF                       00000000-XR            1322  (1)
SYS$SETEF                        00000000-XR            1220  (1)      993   (1)
SYS$SETIMR                       00000000-XR            1311  (1)
TAB                             =00000009     250   (1)    #-1191  (1)    #-919   (1)
TT$M_SCOPE                      =00001000            #-1336  (1)
TT$V_SCOPE                      =0000000C            #-1062  (1)
TT$_LA100                       =00000025              449   (1)
TT$_LA12                        =00000024              446   (1)
TT$_LA120                       =00000021              452   (1)
TT$_LA34                        =00000022              455   (1)      458   (1)
TT$_LA36                        =00000020            #-1344  (1)      463   (1)
TT$_VT100                       =00000060              344   (1)      365   (1)      368   (1)      371   (1)
                                                       374   (1)      377   (1)      380   (1)      383   (1)
                                                       386   (1)      389   (1)      392   (1)      428   (1)
                                                       431   (1)      434   (1)
TT$_VT101                       =00000061              347   (1)
TT$_VT102                       =00000062              350   (1)
TT$_VT125                       =00000064              353   (1)      356   (1)      359   (1)      362   (1)
TT$_VT132                       =00000066              404   (1)      407   (1)      410   (1)      413   (1)
                                                       416   (1)      419   (1)      422   (1)      425   (1)
TT$_VT200_SERIES                =0000006E              395   (1)      398   (1)      401   (1)
TT$_VT52                        =00000040              335   (1)      338   (1)      341   (1)
TT$_VT55                        =00000041              326   (1)
TT$_VT5X                        =00000040              317   (1)      320   (1)      323   (1)      329   (1)
                                                       332   (1)
T_CNTRLC                         00000000-R   285   (1)  751   (1)
T_CNTRLO                         0000000F-R   294   (1)  782   (1)
T_CNTRLR                         0000000C-R   291   (1)  1134  (1)
T_CNTRLU                         00000007-R   288   (1)  1117  (1)
T_CRLF                           00000014-R   297   (1)  1050  (1)      813   (1)      914   (1)
T_LA100                          00000200-R   447   (1)
T_LA12                           000001F8-R   444   (1)
T_LA120                          00000208-R   450   (1)
T_LA34                           0000020F-R   453   (1)
T_LA34A                          00000218-R   456   (1)
T_LA36                           00000225-R   462   (1)
T_OUT_IS_MBX                     0000006D-R   306   (1)  552   (1)
T_SHOW_WIDTH                     00000048-R   303   (1)  601   (1)
T_VT100                          0000012A-R   369   (1)
T_VT100A                         000000C7-R   342   (1)
T_VT100B                         00000133-R   372   (1)
T_VT100C                         0000013C-R   375   (1)
T_VT100D                         00000145-R   378   (1)
T_VT100E                         0000014E-R   381   (1)
T_VT100F                         00000157-R   384   (1)
T_VT100G                         00000160-R   387   (1)
T_VT100H                         00000169-R   390   (1)
T_VT101                          000000CC-R   345   (1)
T_VT102                          000000D5-R   348   (1)
T_VT125                          000000DC-R   351   (1)
```

```
T_VT125A                    000000ED-R   354    (1)
T_VT125B                    000000FE-R   357    (1)
T_VT125C                    0000010F-R   360    (1)
T_VT132                     0000019D-R   402    (1)
T_VT132A                    000001A6-R   405    (1)
T_VT132B                    000001AF-R   408    (1)
T_VT132C                    000001B8-R   411    (1)
T_VT132D                    000001C1-R   414    (1)
T_VT132E                    000001CA-R   417    (1)
T_VT132F                    000001D3-R   420    (1)
T_VT132G                    000001DC-R   423    (1)
T_VT220                     00000172-R   393    (1)
T_VT220A                    00000182-R   396    (1)
T_VT240                     00000189-R   399    (1)
T_VT50_A                    0000009A-R   315    (1)
T_VT50_B                    0000009F-R   318    (1)
T_VT50_C                    000000A4-R   321    (1)
T_VT50_H                    000000AE-R   327    (1)
T_VT50_J                    000000B3-R   330    (1)
T_VT52_K                    000000B8-R   333    (1)
T_VT52_L                    000000BD-R   336    (1)
T_VT52_M                    000000C2-R   339    (1)
T_VT55_E                    000000A9-R   324    (1)
T_VT61                      00000120-R   363    (1)
T_VT61A                     000001E5-R   426    (1)
T_VT61B                     000001EA-R   429    (1)
T_VT61C                     000001EF-R   432    (1)
T_VT71                      00000125-R   366    (1)
T_WIDTH                     00000017-R   300    (1)      544    (1)
VIDEO_TYPES                 0000009A-R   313    (1)      1337   (1)
VRSETWIDTH                  00000000-R   513    (1)
VRSHOWWIDTH                 000000A5-R   595    (1)
VT                         =0000000B     252    (1)    #-1187   (1)
WRITEVBLK                   000002FD-R   986    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ASNPUSH	1	1311 (1)	1311 (1)
$BINTIM_S	1	1305 (1)	1305 (1)
$D1_PRINT_S	1	544 (1)	544 (1) 552 (1) 601 (1)
$DEF	1	239 (1)	
$DEFINI	1	232 (1)	232 (1) 234 (1) 240 (1)
$DS_DSADEF	5	232 (1)	232 (1)
$DS_TYPEDEF	4	238 (1)	238 (1)
$EQU	1	239 (1)	236 (1) 238 (1)
$EQULS1	1	238 (1)	236 (1) 238 (1)
$EQULST	1	236 (1)	236 (1) 238 (1)
$GBLINI	2		236 (1) 237 (1) 238 (1) 239 (1)
$OFFDEF	1	235 (1)	235 (1)
$PRDEF	5	234 (1)	234 (1)
$PRINT	2	542 (1)	542 (1) 550 (1) 598 (1)
$PUSHADR	1	526 (1)	1305 (1) 1311 (1) 1322 (1) 526 (1) 534 (1)
$PUSHTWO	1	526 (1)	526 (1) 534 (1)
$QIODEF	1	235 (1)	235 (1)
$QIOPUSH	1	526 (1)	1311 (1) 526 (1) 534 (1)
$QIOW_S	1	523 (1)	523 (1) 531 (1)
$READEF_S	1	1322 (1)	1322 (1)
$SETEF_G	1	993 (1)	1220 (1) 993 (1)
$SETIMR_S	1	1311 (1)	1311 (1)
$TTDEF	7	240 (1)	240 (1)
$VIELD	1		237 (1) 239 (1)
$VIELD1	1	239 (1)	237 (1) 239 (1)
BR_IF_CTRLO	1	896 (1)	896 (1)
BR_IF_CTRL_C_NO_ECHO	1	749 (1)	749 (1)
BR_IF_NOT_APT	1	899 (1)	899 (1)
BR_IF_NOT_FLUSH_CTRL_C	1	762 (1)	762 (1)
BR_IF_NOT_USER	1	519 (1)	519 (1)
CLEAR_CTRLO	1	989 (1)	1060 (1) 989 (1)
CLEAR_FLUSH_CTRL_C	1	768 (1)	768 (1)
CLIDEF	4	233 (1)	233 (1)
CMK	1	651 (1)	1325 (1) 651 (1) 717 (1) 873 (1) 933 (1)
CMKDEF	1	236 (1)	236 (1)
CRDDEF	1	239 (1)	239 (1)
DSFDEF	3	237 (1)	237 (1)
SET_CTRLC	1	755 (1)	755 (1) 765 (1)
SET_CTRLO	1	754 (1)	754 (1)
SET_FLUSH_CTRL_C	1	764 (1)	764 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	1360 (1) 1363 (1)
ADDL2	00C0	1167 (1) 1171 (1)
BBC	00E1	1062 (1) 1174 (1) 1326 (1) 519 (1) 653 (1) 718 (1) 762 (1)
		899 (1) 943 (1) 987 (1)
BBCC	00E5	1110 (1) 713 (1)
BBS	00E0	1325 (1) 651 (1) 717 (1) 732 (1) 736 (1) 749 (1) 873 (1)
		896 (1) 900 (1) 933 (1)
BBSC	00E4	1060 (1) 768 (1) 818 (1) 825 (1) 989 (1)
BBSS	00E2	1100 (1) 754 (1) 755 (1) 764 (1) 765 (1)
BEQL	0013	1048 (1) 1163 (1) 1177 (1) 1186 (1) 1188 (1) 1190 (1) 1192 (1)
		1194 (1) 1324 (1) 1330 (1) 1356 (1) 1359 (1) 656 (1) 660 (1)
		663 (1) 666 (1) 720 (1) 739 (1) 741 (1)
BGEQ	0018	1180 (1) 929 (1)
BGTR	0014	517 (1)
BICB2	008A	1061 (1)
BICL2	00CA	1041 (1)
BISB	0088	771 (1)
BITB	0093	924 (1)
BITW	00B3	740 (1)
BLBC	00E9	527 (1) 903 (1)
BLBS	00E8	1339 (1) 1343 (1)
BLEQ	0015	515 (1) 885 (1)
BLEQU	001B	1184 (1)
BLSS	0019	1203 (1) 657 (1)
BLSSU	001F	1072 (1) 912 (1)
BNEQ	0012	1044 (1) 1046 (1) 1068 (1) 1097 (1) 1165 (1) 1309 (1) 722 (1)
		746 (1) 770 (1) 775 (1) 780 (1) 908 (1) 920 (1) 925 (1)
BNEQU	0012	1066 (1) 1095 (1) 1116 (1) 1122 (1) 1129 (1) 1131 (1) 1151 (1)
BRB	0011	1080 (1) 1159 (1) 1168 (1) 1325 (1) 1328 (1) 1362 (1) 651 (1)
		717 (1) 756 (1) 766 (1) 772 (1) 777 (1) 873 (1) 905 (1)
		933 (1)
BRW	0031	1069 (1) 1092 (1) 1098 (1) 1108 (1) 1119 (1) 1126 (1) 1132 (1)
		1137 (1) 1181 (1) 1200 (1) 1207 (1) 1213 (1) 1310 (1) 733 (1)
		742 (1) 763 (1)
BSBB	0010	814 (1) 819 (1) 826 (1) 888 (1) 923 (1)
BSBW	0030	1051 (1) 1054 (1) 1059 (1) 1078 (1) 1086 (1) 1088 (1) 1090 (1)
		1103 (1) 1106 (1) 1113 (1) 1118 (1) 1135 (1) 1136 (1) 1154 (1)
		1156 (1) 1197 (1) 1206 (1) 1210 (1) 1318 (1) 1338 (1) 1342 (1)
		669 (1) 715 (1) 752 (1) 783 (1) 895 (1) 904 (1) 915 (1)
		992 (1)
CALLG	00FA	1220 (1) 993 (1)
CALLS	00FB	1305 (1) 1311 (1) 1322 (1) 526 (1) 534 (1) 544 (1) 552 (1)
		601 (1)
CHMK	00BC	1325 (1) 651 (1) 717 (1) 873 (1) 933 (1)
CLRB	0094	1319 (1) 747 (1) 776 (1) 781 (1) 875 (1) 909 (1)
CLRL	00D4	1057 (1) 1124 (1) 1325 (1) 1340 (1) 1354 (1) 545 (1) 553 (1)
		651 (1) 717 (1) 729 (1) 873 (1) 933 (1)
CLRQ	007C	1306 (1) 1311 (1) 521 (1) 526 (1) 534 (1) 535 (1) 548 (1)
		786 (1)
CMPB	0091	1043 (1) 1047 (1) 1065 (1) 1071 (1) 1094 (1) 1115 (1) 1121 (1)

		1128	(1)	1150	(1)	1176	(1)	1183	(1)	1185	(1)	1187	(1)	1189	(1)
		1191	(1)	1193	(1)	1355	(1)	655	(1)	659	(1)	662	(1)	665	(1)
		719	(1)	738	(1)	745	(1)	769	(1)	774	(1)	907	(1)	911	(1)
		919	(1)	928	(1)										
CMPC3	0029	1358	(1)												
CMPL	00D1	1202	(1)	1323	(1)	516	(1)								
CMPW	00B1	779	(1)												
CMPZV	00ED	721	(1)												
CVTBL	0098	544	(1)	552	(1)	601	(1)								
DECL	00D7	1076	(1)	1079	(1)	1084	(1)	1091	(1)	1107	(1)	1179	(1)		
EXTZV	00EF	744	(1)												
INCB	0096	930	(1)												
INCL	00D6	1199	(1)	1212	(1)	1361	(1)								
INSV	00F0	1344	(1)	1364	(1)										
JSB	0016	1325	(1)	651	(1)	717	(1)	873	(1)	933	(1)				
MFPR	00DB	731	(1)	735	(1)	942	(1)								
MOVAB	009E	1050	(1)	1056	(1)	1117	(1)	1134	(1)	1307	(1)	1320	(1)	1337	(1)
		1341	(1)	751	(1)	782	(1)	811	(1)	813	(1)	821	(1)	914	(1)
MOVAL	00DE	1215	(1)												
MOVAQ	007E	522	(1)	530	(1)										
MOVB	0090	1152	(1)	1211	(1)	1327	(1)	537	(1)	932	(1)				
MOVL	00D0	1153	(1)	1155	(1)	1157	(1)	1162	(1)	1166	(1)	1169	(1)	1172	(1)
		1196	(1)	1198	(1)	1308	(1)	1317	(1)	1336	(1)	1346	(1)	1365	(1)
		1366	(1)	514	(1)	538	(1)	603	(1)	668	(1)	723	(1)	822	(1)
		884	(1)												
MOVPSL	00DC	1325	(1)	651	(1)	717	(1)	873	(1)	933	(1)				
MOVQ	007D	1053	(1)	815	(1)	991	(1)								
MOVW	00B0	1216	(1)	1217	(1)	1218	(1)	1219	(1)						
MOVZBL	009A	1077	(1)	1085	(1)	1087	(1)	1089	(1)	1102	(1)	1105	(1)	1112	(1)
		1123	(1)	1158	(1)	1205	(1)	1209	(1)	1357	(1)	544	(1)	552	(1)
		596	(1)	601	(1)	817	(1)	824	(1)	870	(1)	872	(1)	880	(1)
		887	(1)	921	(1)										
MOVZBW	009B	529	(1)												
MOVZWL	003C	1125	(1)	526	(1)	534	(1)								
MTPR	00DA	944	(1)												
MULL2	00C4	1170	(1)												
POPL	8ED0	876	(1)												
POPR	00BA	672	(1)	753	(1)	828	(1)	891	(1)	916	(1)	945	(1)		
PUSHAB	009F	544	(1)	552	(1)	601	(1)								
PUSHAL	00DF	1322	(1)	526	(1)	534	(1)								
PUSHAQ	007F	1305	(1)	1311	(1)										
PUSHL	00DD	1311	(1)	1322	(1)	526	(1)	534	(1)	601	(1)	869	(1)		
PUSHR	00BB	650	(1)	750	(1)	810	(1)	883	(1)	913	(1)	940	(1)		
REI	0002	788	(1)	946	(1)										
RET	0004	1348	(1)												
RSB	0005	1221	(1)	1367	(1)	539	(1)	546	(1)	554	(1)	604	(1)	673	(1)
		726	(1)	829	(1)	877	(1)	892	(1)	926	(1)	936	(1)	994	(1)
SOBGTR	00F5	820	(1)	827	(1)	874	(1)	889	(1)						
SUBB2	0082	1073	(1)	1081	(1)										
TSTB	0095	1045	(1)	1329	(1)										
TSTL	00D5	1067	(1)	1096	(1)	1130	(1)	1164	(1)						
XORL2	00CC	784	(1)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	1289	(1)	1290	(1)	1291	(1)										
.ASCIC	1284	(1)	1285	(1)	1286	(1)	286	(1)	289	(1)	292	(1)	295	(1)	298	(1)
	301	(1)	304	(1)	307	(1)	316	(1)	319	(1)	322	(1)	325	(1)	328	(1)
	331	(1)	334	(1)	337	(1)	340	(1)	343	(1)	346	(1)	349	(1)	352	(1)
	355	(1)	358	(1)	361	(1)	364	(1)	367	(1)	370	(1)	373	(1)	376	(1)
	379	(1)	382	(1)	385	(1)	388	(1)	391	(1)	394	(1)	397	(1)	400	(1)
	403	(1)	406	(1)	409	(1)	412	(1)	415	(1)	418	(1)	421	(1)	424	(1)
	427	(1)	430	(1)	433	(1)	445	(1)	448	(1)	451	(1)	454	(1)	457	(1)
.ASCID	1300	(1)														
.BLKB	1299	(1)														
.BLKL	233	(1)														
.BLKQ	1301	(1)														
.BYTE	272	(1)	274	(1)	276	(1)	278	(1)	317	(1)	320	(1)	323	(1)	326	(1)
	329	(1)	332	(1)	335	(1)	338	(1)	341	(1)	344	(1)	347	(1)	350	(1)
	353	(1)	356	(1)	359	(1)	362	(1)	365	(1)	368	(1)	371	(1)	374	(1)
	377	(1)	380	(1)	383	(1)	386	(1)	389	(1)	392	(1)	395	(1)	398	(1)
	401	(1)	404	(1)	407	(1)	410	(1)	413	(1)	416	(1)	419	(1)	422	(1)
	425	(1)	428	(1)	431	(1)	434	(1)	446	(1)	449	(1)	452	(1)	455	(1)
	458	(1)	463	(1)												
.END	1369	(1)														
.ENDC	1305	(1)	1311	(1)	1322	(1)	236	(1)	237	(1)	238	(1)	239	(1)	526	(1)
	534	(1)	544	(1)	552	(1)	601	(1)								
.ENDM	1305	(1)	1311	(1)	1322	(1)	232	(1)	233	(1)	234	(1)	235	(1)	236	(1)
	237	(1)	238	(1)	239	(1)	240	(1)	519	(1)	523	(1)	526	(1)	542	(1)
	544	(1)	651	(1)	749	(1)	754	(1)	755	(1)	762	(1)	764	(1)	768	(1)
	896	(1)	899	(1)	989	(1)	993	(1)								
.ENDR	236	(1)	237	(1)	238	(1)	239	(1)	544	(1)	552	(1)	601	(1)		
.ENTRY	1303	(1)														
.GLOBL	1220	(1)	1305	(1)	1311	(1)	1322	(1)	526	(1)	534	(1)	993	(1)		
.IDENT	10	(1)														
.IF	1305	(1)	1311	(1)	1322	(1)	236	(1)	237	(1)	238	(1)	239	(1)	526	(1)
	534	(1)	544	(1)	552	(1)	601	(1)								
.IFF	1305	(1)	1311	(1)	1322	(1)	236	(1)	237	(1)	238	(1)	239	(1)	526	(1)
	534	(1)														
.IIF	236	(1)	237	(1)	238	(1)	239	(1)								
.IRP	235	(1)	236	(1)	237	(1)	238	(1)	239	(1)	544	(1)	552	(1)	601	(1)
.LIBRARY	224	(1)	225	(1)	226	(1)										
.LIST	232	(1)	233	(1)	235	(1)										
.LONG	1292	(1)	280	(1)	436	(1)	460	(1)								
.MACRO	236	(1)	237	(1)	238	(1)	239	(1)								
.MDELETE	236	(1)														
.NLIST	232	(1)	233	(1)	235	(1)										
.NOCROSS	232	(1)	234	(1)	240	(1)										
.NOSHOW	11	(1)														
.PAGE	1038	(1)	1224	(1)	218	(1)	265	(1)	281	(1)	32	(1)	466	(1)	512	(1)
	556	(1)	594	(1)	605	(1)	648	(1)	674	(1)	711	(1)	789	(1)	809	(1)
	830	(1)	867	(1)	938	(1)	948	(1)	985	(1)	996	(1)				
.PSECT	233	(1)	267	(1)	283	(1)	468	(1)								
.RESTORE	232	(1)	233	(1)	234	(1)	240	(1)								
.SAVE	232	(1)	233	(1)	234	(1)	240	(1)								

.SBTTL	1225	(1)	219	(1)	467	(1)	557	(1)	606	(1)	675	(1)	790	(1)	831	(1)	
	949	(1)	997	(1)													
.SUBTITLE	266	(1)	282	(1)													
.TITLE	9	(1)															

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	12	#-1043 (1) #-1053 (1) 1056 (1) #-1162 (1) #-1202 (1) 1215 (1)
		1220 (1) #-815 (1) 821 (1) 988 (1) #-991 (1) 993 (1)
R0	73	#-1050 (1) #-1053 (1) #-1061 (1) #-1065 (1) #-1094 (1) #-1115 (1)
		#-1117 (1) #-1121 (1) #-1123 (1) #-1125 (1) #-1128 (1) #-1134 (1)
		#-1150 (1) #-1152 (1) #-1153 (1) #-1157 (1) #-1176 (1) #-1183 (1)
		#-1185 (1) #-1187 (1) #-1189 (1) #-1191 (1) #-1193 (1) #-1196 (1)
		#-1198 (1) #-1209 (1) #-1211 (1) #-1216 (1) #-1317 (1) #-1323 (1)
		1326 (1) #-1339 (1) #-1343 (1) #-1354 (1) #-1366 (1) #-514 (1)
		#-516 (1) #-522 (1) 526 (1) #-527 (1) #-530 (1) 534 (1)
		#-538 (1) #-545 (1) #-553 (1) #-596 (1) #-601 (1) #-603 (1)
		#-650 (1) 653 (1) #-672 (1) 718 (1) #-723 (1) #-731 (1)
		732 (1) #-750 (1) #-751 (1) #-753 (1) #-782 (1) #-786 (1)
		#-813 (1) #-870 (1) #-872 (1) #-880 (1) #-887 (1) #-913 (1)
		#-914 (1) #-916 (1) #-940 (1) #-942 (1) 943 (1) #-945 (1)
		#-991 (1)
R1	53	#-1077 (1) #-1085 (1) #-1087 (1) #-1089 (1) #-1102 (1) #-1105 (1)
		#-1112 (1) #-1153 (1) #-1155 (1) #-1196 (1) #-1205 (1) #-1209 (1)
		#-1215 (1) #-1216 (1) #-1217 (1) #-1218 (1) #-1219 (1) #-1327 (1)
		#-650 (1) #-655 (1) #-659 (1) #-662 (1) #-665 (1) #-668 (1)
		#-672 (1) 721 (1) #-723 (1) #-729 (1) #-735 (1) 736 (1)
		#-740 (1) 744 (1) #-745 (1) #-750 (1) #-753 (1) #-769 (1)
		#-774 (1) #-779 (1) #-817 (1) #-824 (1) #-872 (1) #-880 (1)
		#-884 (1) #-887 (1) #-907 (1) #-913 (1) #-916 (1) #-919 (1)
		#-921 (1) #-928 (1) #-932 (1) #-944 (1)
R2	27	#-1057 (1) #-1067 (1) #-1079 (1) #-1091 (1) #-1096 (1) #-1107 (1)
		#-1124 (1) #-1130 (1) #-1199 (1) #-1202 (1) #-1212 (1) #-1217 (1)
		1303 (1) #-514 (1) #-529 (1) #-537 (1) #-822 (1) #-869 (1)
		#-870 (1) #-874 (1) #-876 (1) #-883 (1) #-884 (1) #-889 (1)
		#-891 (1) #-913 (1) #-916 (1)
R3	17	#-1056 (1) #-1076 (1) #-1084 (1) #-1105 (1) #-1123 (1) #-1152 (1)
		#-1158 (1) #-1198 (1) #-1211 (1) #-1218 (1) 1303 (1) #-810 (1)
		#-815 (1) #-817 (1) #-822 (1) #-827 (1) #-828 (1)
R4	23	#-1162 (1) #-1164 (1) #-1167 (1) #-1169 (1) #-1171 (1) #-1172 (1)
		1174 (1) 1303 (1) #-1337 (1) #-1341 (1) #-1355 (1) #-1357 (1)
		1358 (1) #-1360 (1) #-1361 (1) #-1363 (1) #-1364 (1) #-810 (1)
		#-820 (1) #-821 (1) #-824 (1) #-828 (1)
R5	10	1303 (1) #-1357 (1) #-1358 (1) #-1360 (1) #-1363 (1) #-810 (1)
		#-811 (1) 818 (1) 825 (1) #-828 (1)
R6	7	#-1166 (1) #-1169 (1) #-1170 (1) #-1174 (1) #-1176 (1) #-1179 (1)
		1303 (1)
R7	3	1303 (1) #-1307 (1) #-1308 (1)
R8	9	1303 (1) #-1308 (1) #-1317 (1) #-1320 (1) #-1327 (1) #-1336 (1)
		#-1340 (1) #-1346 (1) #-1365 (1)
SP	38	#-1311 (1) #-1325 (1) #-521 (1) 522 (1) #-526 (1) #-529 (1)
		530 (1) #-534 (1) #-535 (1) #-544 (1) #-548 (1) #-552 (1)
		#-601 (1) #-651 (1) #-717 (1) #-873 (1) #-933 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.01	00:00:00.19
Command processing	887	00:00:00.21	00:00:01.60
Pass 1	798	00:00:02.82	00:00:05.53
Symbol table sort	0	00:00:00.20	00:00:00.19
Pass 2	47	00:00:01.00	00:00:01.83
Symbol table output	2	00:00:00.07	00:00:00.22
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	10	00:00:00.49	00:00:00.79
Assembler run totals	1769	00:00:04.80	00:00:10.35

The working set limit was 2900 pages.
134484 bytes (263 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 646 non-local and 85 local symbols.
1369 source lines were read in Pass 1, producing 59 object records in Pass 2.
84 pages of virtual memory were used to define 41 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	16
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	18
TOTALS (all libraries)	37

755 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CONSOLE.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element CRDIMAGE.B32
;    0002  0      ! *2    13-FEB-1986 10:18:54 BERGAZZI "<no reason given>"
;    0003  0      ! *1     6-FEB-1986 15:16:02 DS ""
;    0004  0      ! DEC/CMS REPLACEMENT HISTORY, Element CRDIMAGE.B32
;    0005  0      %TITLE '*** Loading and Unloading the CRD Image'
;    0006  0      MODULE CRDIMAGE (                       ! Routines to handle loading and unloading the Loadable-CRD image
;    0007  0                      IDENT = '08-19'
;    0008  0                      ) =
;    0009  0      !++
;    0010  0      ! COPYRIGHT (c) 1980, 1981, 1985
;    0011  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0012  0      !
;    0013  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0014  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0015  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0016  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0017  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0018  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0019  0      ! REMAIN IN DEC.
;    0020  0      !
;    0021  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0022  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0023  0      ! CORPORATION.
;    0024  0      !
;    0025  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0026  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0027  0      !--
;    0028  0      !++
;    0029  0      ! Facility:
;    0030  0      !
;    0031  0      !       Diagnostic Supervisor (part of the Resident-DS image).
;    0032  0      !
;    0033  0      ! Abstract:
;    0034  0      !
;    0035  0      !       This module contains the routines necessary to load the CRD-Loadable image file and integrate
;    0036  0      !       it with the Resident-DS, thus putting the Supervisor under control of CRD. It also now contains
;    0037  0      !       the routines necessary to unload the CRD image.
;    0038  0      !
;    0039  0      ! Author:
;    0040  0      !
;    0041  0      !       Jack Stansbury
;    0042  0      !
;    0043  0      ! Creation Date:
;    0044  0      !
;    0045  0      !       29-March-1982
;    0046  0      !
;    0047  0      ! Version
;    0048  0      !
;    0049  0      !       Version 6.9
;    0050  0      !
;    0051  0      ! Modified By:
;    0052  0      !
;    0053  0      !       01      Jack Stansbury, 29-June-1982, Version 6.8
;    0054  0      !               Took out checking the DS verification bytes, and the check to see if
;    0055  0      !               the DS vector count EQL'ed the CRD vector count (why was this in there
;    0056  0      !               anyway???). Assume that the DS verification bytes are okay. It is CRD
;    0057  0      !               that I am verifying anyway!
```

```
;    0058  0      !
;    0059  0      !       02      Jack Stansbury, 1-July-1982, Version 6.8
;    0060  0      !               You dummy! I put the check of the vector count in there for a good reason!
;    0061  0      !               If the vector counts are not the same, one or the other may get some other
;    0062  0      !               data overwritten (i.e., data following the dispatch vectors). This must
;    0063  0      !               NOT happen. So, if the vector counts are not equal, it is an incompat-
;    0064  0      !               ibility error (not a corrupted error).
;    0065  0      !
;    0066  0      !       03      Jack Stansbury, 15-July-1982, Version 6.9
;    0067  0      !               Added support for loading in the CRD Menu Test file. Which file to load in
;    0068  0      !               depends on which DSA bit is set.
;    0069  0      !
;    0070  0      !       04      Jack Stansbury, 25-July-1982, Version 6.9
;    0071  0      !               Added code to check to see if this was running in User-Mode, and if so, then
;    0072  0      !               use the CRD$ExpReg routine; else use $ExpReg to expand the VDS's memory.
;    0073  0      !               Note that although CRD Menu Test will NOT be normally run in User Mode,
;    0074  0      !               it is run this way when it is being debugged. Whether or not it should run
;    0075  0      !               is determined by the CLI routines.
;    0076  0      !
;    0077  0      !       05      Jack Stansbury, 7-September-1982, Version 6.9
;    0078  0      !               Made major modifications to all the routines in this module. I renamed it from
;    0079  0      !               LoadCRD to CRDImage. Added many routines.
;    0080  0      !
;    0081  0      !       06      Jack Stansbury, 26-Oct-1982, Version 6.9
;    0082  0      !               Added code to disable printing ^C's during running of CRD.
;    0083  0      !
;    0084  0      !       07      Jack Stansbury, -2-Mar-1983, Version 6.11
;    0085  0      !               Improved the debugging mode of CRD. Added SET CRD/DEBUG to the CLI.
;    0086  0      !               Made printout of debug vectors much better and much more informative.
;    0087  0      !               Put the actual debug vector printout in this module rather than in a
;    0088  0      !               CRD module. Also took out references to the DS$K_Numb_Dispatch_Vectors.
;    0089  0      !               Also added the DSR$Print_Typecode_Name routine, for use by all the VDS.
;    0090  0      !               Also changed the check to see if the debug flag was set (in CRD_Error).
;    0091  0      !
;    0092  0      !       08      Jack Stansbury, 7-Mar-1983, Version 6.11
;    0093  0      !               Added the DSR$Show_Calls routine - this is a global routine (JSB linkage)
;    0094  0      !               that will print the contents of the last 'n' call frames on the stack.
;    0095  0      !
;    0096  0      !       09      Jack Stansbury, 8-Mar-1983, Version 6.11
;    0097  0      !               Took out the DSR$Show_Calls routine. I created a new module for this routine
;    0098  0      !               called CallFrame.B32. That routine didn't realy belong in here.
;    0099  0      !
;    0100  0      !       10      John Ciukaj     4-April-1983    Version 6.11
;    0101  0      !               - Changed references to CRD Bits in DSA Flags
;    0102  0      !                 to distinguish between online and offline
;    0103  0      !
;    0104  0      !       11      Jack Stansbury, April 4, 1983, Version 6.11
;    0105  0      !               Made Exchange_Dispatch_Vectors into a global routine so it can
;    0106  0      !               be accessed through the debugger (DEBUG-32).
;    0107  0      !
;    0108  0      !       12      Ron Parker      5-NOV-1984
;    0109  0      !               Remove the test for the 730 CPU type. The DSR$LOAD_CRD
;    0110  0      !               routine will now load on any cpu type.
;    0111  0      !
;    0112  0      !       13      Ron Parker      20-FEB-1985
;    0113  0      !               - Clean up message text, remove specific CPU references
;    0114  0      !
```

```
;    0115  0      !       14      Ron Parker      18-APR-1985
;    0116  0      !               - Change references of EN* to EV*
;    0117  0      !
;    0118  0      !       15      Amy Iorio       24-JULY-1985
;    0119  0      !               Substitute $ds_load with $open and xab, rms.
;    0120  0      !
;    0121  0      !       16      Amy Iorio       06-AUG-1985
;    0122  0      !               Fix to $ds_load substitution (15).
;    0123  0      !
;    0124  0      !       17      Jim Shelhamer   19-SEP-1985     Version 8.3
;    0125  0      !               Reversed edit 14, ES* -> EN*.
;    0126  0      !
;    0127  0      !       18      Domenic Andella 8-Nov-1985      Version 9.0
;    0128  0      !               Add argument to CRD$EXPREG and replace $CNTREG
;    0129  0      !               with $DS_RELBUF To be compatible with new args
;    0130  0      !               generated with GETBUF, and RELBUF service.
;    0131  0      !
;    0132  0      !       19      Domenic Andella 25-Nov-1985     Version 9.2
;    0133  0      !               Cannot use RELBUF since this service depends upon
;    0134  0      !               GETBUF service to load DS$GL_BUFCNT. Therefore will
;    0135  0      !               make direct call to EXE$CNTREG, and push dummy arg
;    0136  0      !               to account for CNTREG$_VA.
;    0137  0      !
;    0138  0      !--
;    0139  0      %SBTTL  'Libraries'
;    0140  1      BEGIN                                   ! Begin the CRDIMAGE module
;    0141  1
;    0142  1      LIBRARY
;    0143  1              '$DS';
;    0144  1
;    0145  1      LIBRARY
;    0146  1              '$Diag';
;    0147  1
;    0148  1      LIBRARY
;    0149  1              '$CRD';
;    0150  1
;    0151  1      LIBRARY
;    0152  1              'Sys$Library:Lib';
;    0153  1      %SBTTL  'Table of Contents'
;    0154  1
;    0155  1      FORWARD ROUTINE
;    0156  1              Exchange_Dispatch_Vectors       : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0157  1                                              ! Exchange the dispatch vectors between the DS and CRD
;    0158  1
;    0159  1              DSR$Load_CRD                    : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0160  1                                              ! Load the CRD Loadable image file
;    0161  1
;    0162  1              DSR$Unload_CRD                  : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0163  1                                              ! Unload the CRD Loadable image file
;    0164  1
;    0165  1              DSR$Restore_CRD_Vectors         : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0166  1                                              ! Restore the CRD dispatch vectors - this is called by the BEGIN0 label
;    0167  1                                              ! in the KERNEL module. The Unload_CRD routine cannot be called because
;    0168  1                                              ! memory has already been re-mapped by the time BEGIN0 executes.
;    0169  1
;    0170  1              CRD_Exit                        : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0171  1                                              ! Either exit from the VDS image (back to the console), or to the CLI.
```

```
;    0172  1
;    0173  1              CRD_Error                       : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),
;    0174  1                                               ! Print an error message and take appropriate action(s).
;    0175  1
;    0176  1              Internal_Error                  : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),                    ![07
;    0177  1                                               ! Print the extended debugging information about CRD.         ![07
;    0178  1
;    0179  1              DSR$Print_TypeCode_Name         : NOVALUE ADDRESSING_MODE (LONG_RELATIVE);                    ![07
;    0180  1                                               ! Print the message                                           ![07
;    0181  1      %SBTTL  'Included Macros and Literals'
;    0182  1
;    0183  1              $CRD_Literals;                  ! Define the CRD literals
;    0184  1              $DS_DSSDef;                     ! Define the DS service routine vector addresses
;    0185  1              $DS_TypeDef;                    ! Define the typecode literals
;    0186  1              $DS_DSADef;                     ! Define the DSA flags.
;    0187  1      %SBTTL  'Other Literals'
;    0188  1
;    0189  1      LITERAL
;    0190  1              !+
;    0191  1              ! These indicate to the Error routine whether or not the space has been allocated
;    0192  1              ! for the CRD image.
;    0193  1              !-
;    0194  1
; P  0195  1              $EquLst (K_, LOCAL, 0, 1
; P  0196  1                      ,(Region_Expanded)
; P  0197  1                      ,(Region_Not_Expanded)
;    0198  1                      ),
;    0199  1
;    0200  1              !+
;    0201  1              ! These are the types of errors that are detected and reported.
;    0202  1              !-
;    0203  1
; P  0204  1              $EquLst (K_, LOCAL, 0, 1
; P  0205  1                      ,(No_DSA_Bit_Set)
; P  0206  1                      ,(Determine_Size)
; P  0207  1                      ,(Expand_Region)
; P  0208  1                      ,(Load_File)
; P  0209  1                      ,(CRD_Internal)
; P  0210  1                      ,(Corrupted)
; P  0211  1                      ,(Incompatible)
; P  0212  1                      ,(Not_Contracted)
; P  0213  1                      ,(MM_Is_On)
; P  0214  1                      ,(Last_Error_Type)
;    0215  1                      ),
;    0216  1
;    0217  1              !+
;    0218  1              ! These are used to reference the first longword in both the DS$AL_DS_Dispatch_Vectors
;    0219  1              ! and the CRD$AL_CRD_Dispatch_Vectors longwords.
;    0220  1              !-
;    0221  1
; P  0222  1              $EquLst (K_, LOCAL, 0, 1
; P  0223  1                      ,(Number_Of_Vectors)
; P  0224  1                      ,(Positive_Version)
; P  0225  1                      ,(Negative_Version)
; P  0226  1                      ,(Null_Byte)
;    0227  1                      ),
;    0228  1
```

```
;    0229  1              !+
;    0230  1              ! Use these to reference the CRD_Image_Location array.
;    0231  1              !-
;    0232  1
; P  0233  1              $EquLst (K_, LOCAL, 0, 1
; P  0234  1                      ,(Starting_Adr)
; P  0235  1                      ,(Ending_Adr)
;    0236  1                      ),
;    0237  1
;    0238  1              !+
;    0239  1              ! Use these to reference the DS$Q_CRD_Image_Desc descriptor. Note that this is not a normal
;    0240  1              ! descriptor, since the size field is a longword, rather than a word. This is so because the
;    0241  1              ! size of the CRD image may be bigger than 65536 (or whatever that number - 2^16 - is).
;    0242  1              !-
;    0243  1
; P  0244  1              $EquLst (K_, LOCAL, 0, 1
; P  0245  1                      ,(Image_Size)
; P  0246  1                      ,(Image_Ptr)
;    0247  1                      );
;    0248  1      %SBTTL  'External Routines'
;    0249  1
;    0250  1      EXTERNAL ROUTINE
;    0251  1              DS_Cleanup                      : JSB_None ADDRESSING_MODE (LONG_RELATIVE),
;    0252  1                                              ! Cleanup the diagnostic before going back to CLI command mode
;    0253  1
;    0254  1              MapFree                         : ADDRESSING_MODE (LONG_RELATIVE),
;    0255  1                                              ! Map all of free memory
;    0256  1
;    0257  1              CRD$ExpReg                      : ADDRESSING_MODE (LONG_RELATIVE),
;    0258  1                                              ! Expand Region for CRD.
;    0259  1
;    0260  1              EXE$CNTREG                      : ADDRESSING_MODE (LONG_RELATIVE),          ![19]
;    0261  1                                              ! Contract Region for CRD.                  ![19]
;    0262  1
;    0263  1              DSR$Check_MenuTest_Off_Set      : JSB_None ADDRESSING_MODE (LONG_RELATIVE),
;    0264  1                                              !                                  [10]
;    0265  1                                              ! Return 1 if the MenuTest bits are set in the R5 passed to the VDS
;    0266  1
;    0267  1              DSR$Check_AutoTest_Off_Set      : JSB_None ADDRESSING_MODE (LONG_RELATIVE),
;    0268  1                                              !                                  [10]
;    0269  1                                              ! Return 1 if the AutoTest bits are set in the R5 passed to the VDS
;    0270  1
;    0271  1              Exe$AloNonPaged                 : JSB_Alloc ADDRESSING_MODE (LONG_RELATIVE),
;    0272  1                                              ! Allocate a block of memory from non-paged pool
;    0273  1
;    0274  1              Exe$DeaNonPaged                 : JSB_Deall ADDRESSING_MODE (LONG_RELATIVE),
;    0275  1                                              ! Deallocate a block of memory from non-paged pool
;    0276  1
;    0277  1              DSV$Exit                        : JSB_None ADDRESSING_MODE (LONG_RELATIVE);
;    0278  1                                              ! Use a JSB instruction to get to this routine. Use this routine to
;    0279  1                                              ! cause the Resident-DS to exit.
;    0280  1      %SBTTL  'External Own Storage'
;    0281  1
;    0282  1      EXTERNAL
;    0283  1              Begin_Bliss                     : ADDRESSING_MODE (LONG_RELATIVE),
;    0284  1                                              ! The location of the Begin_Bliss label (in the KERNEL module).
;    0285  1
```

```
;    0286  1              DS$GA_DS_Ctrl_C_First          : ADDRESSING_MODE (LONG_RELATIVE),
;    0287  1                                              ! The first ^C handler for the VDS
;    0288  1
;    0289  1              DS$GA_DS_Ctrl_C_Second         : ADDRESSING_MODE (LONG_RELATIVE),
;    0290  1                                              ! The second ^C handler for the VDS
;    0291  1
;    0292  1              DS$L_UserCntrlC                : ADDRESSING_MODE (LONG_RELATIVE),
;    0293  1                                              ! The user's ^C handler
;    0294  1
;    0295  1              DS$GL_Flags                    : ADDRESSING_MODE (LONG_RELATIVE),
;    0296  1                                              ! The DS flags longword
;    0297  1
;    0298  1              DS$AL_DS_Dispatch_Vectors      : ADDRESSING_MODE (LONG_RELATIVE),
;    0299  1                                              ! The start of the Resident-DS dispatch vectors
;    0300  1
;    0301  1              DS$GL_CRD_Flags                : ADDRESSING_MODE (LONG_RELATIVE),
;    0302  1                                              ! The DS/CRD flags longword in the DSVECS module
;    0303  1
;    0304  1              DS$GA_LastAdr                  : ADDRESSING_MODE (LONG_RELATIVE),
;    0305  1                                              ! The last address at the top of the Supervisor image (after the memory
;    0306  1                                              ! has been allocated for the QIO database and the SCRIPT junk).
;    0307  1
;    0308  1              DS$GL_CRD_Debug                : ADDRESSING_MODE (LONG_RELATIVE);                                  ![07
;    0309  1                                              ! This longword contains either 0 or 1. 1 implies CRD will print CRD ![07
;    0310  1                                              ! debugging messages. 0 implies it will not.                        ![07
;    0311  1
;    0312  1          %SBTTL  'Psect Definitions'
;    0313  1
;    0314  1          PSECT
;    0315  1              PLIT   = Data (NOEXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0316  1
;    0317  1          PSECT
;    0318  1              CODE   = Code (  EXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0319  1
;    0320  1          PSECT
;    0321  1              OWN    = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0322  1
;    0323  1          PSECT
;    0324  1              GLOBAL = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0325  1          %SBTTL  'Module-Global Static Storage'
;    0326  1
;    0327  1              ModNam ('CRDIMAGE');           ! Module name for ErrSup macro
;    0328  1          %SBTTL  'Local ASCIC Bindings'
;    0329  1
;    0330  1          BIND
;    0331  1              T_CRD_Image_Name               = $ASCID ('ENSAB.EXE;0'),                                           ![03
;    0332  1                                             ! The name of the CRD Loadable Image file                           ![03
;    0333  1
;    0334  1              T_CRD_Image_Default            = $ASCID (''),                                                      ![03
;    0335  1                                             ! The default name for the CRD Loadable Image file                  ![03
;    0336  1
;    0337  1
;    0338  1              T_Null                         = $ASCIC (''),
;    0339  1              T_Menu                         = $ASCIC ('MENU TEST'),                                             ![03
;    0340  1              T_Auto                         = $ASCIC ('AUTO TEST'),                                             ![03
;    0341  1
;    0342  1
```

```
;    0343  1              T_File_Not_Found                = $ASCIC ('!/UNABLE TO FIND FILE ENSAB, THE LOADABLE PORTION OF CRD !AC!/'),
; P  0344  1              T_Unknown_Error                 = $ASCIC ('!/ERROR LOADING FILE ENSAB, THE LOADABLE PORTION ',          ![03
;    0345  1                                                       'OF CRD !AC!/'),                                              ![03
;    0346  1              T_Contraction_Error             = $ASCIC ('!/ERROR WHILE ATTEMPTING TO EXIT CRD !AC!/'),
;    0347  1              T_Incompatibility_Error         = $ASCIC ('!/INCORRECT VERSION OF CRD !AC!/'),                          ![03
;    0348  1              T_Corrupted_Error               = $ASCIC ('!/THE LOADABLE PORTION OF CRD !AC IS CORRUPTED!/'),          ![03
; P  0349  1              T_MM_On                         = $ASCIC ('!/CANNOT CONTINUE CRD - MEMORY MANAGEMENT MUST BE OFF',
; P  0350  1                                                       '!/TO TURN OFF MEMORY MANAGEMENT, TYPE "SET MM OFF"',
;    0351  1                                                       '!/RESTART CRD BY TYPING "CRD"!/'),
;    0352  1
;    0353  1              T_AUTO_Abort_CRD                = $ASCIC ('!/!/AUTO TEST ABORTED - CALL FIELD SERVICE'),
;    0354  1              T_AUTO_End_Message              = $ASCIC ('!/**** END OF VAX/CRD AUTO TEST ****'),
;    0355  1              T_AUTO_Exit_To_Console          = $ASCIC ('!/!/RETURNING TO VAX CONSOLE, WAIT FOR ">>>" PROMPT!/'),
;    0356  1              T_No_DSA_Bit_Set                = $ASCIC ('*** No DSA bit is set - DSA$GL_Flags = !XL(X)!/'),
;    0357  1
; P  0358  1              T_Determine_Size                = $ASCIC ('*** Error while determining size of the ENSAB file ',
;    0359  1                                                       '- Status = !XL(X)!/'),
;    0360  1              T_Expand_Region                 = $ASCIC ('*** Error while expanding the region - Status = !XL(X)!/'),
; P  0361  1              T_Load_File                     = $ASCIC ('*** Error while loading the ENSAB file into memory ',
;    0362  1                                                       '- Status = !XL(X)!/'),
;    0363  1
;    0364  1              T_CRD_Internal                  = $ASCIC ('*** A CRD Internal Error occurred!/'),
;    0365  1              T_CRD_Internal_Header           = $ASCIC ('*** The following are the CRD debug vectors:!/'),
;    0366  1              T_CRD_Internal_Debug            = $ASCIC ('*** Debug vector [!UL]: !XL(X)   !-!SL(D)!/'),
;    0367  1
;    0368  1              T_Not_Contracted                = $ASCIC ('*** The region was not contracted - Status = !XL(X)!/'),
; P  0369  1              T_Corrupted                     = $ASCIC ('*** Either the Null byte is not null, ',
;    0370  1                                                       'or the version numbers are wrong!/'),
;    0371  1              T_Incompatible                  = $ASCIC ('*** These versions of ENSAB and E*SAA are incompatible!/'),
;    0372  1
;    0373  1
; P  0374  1              T_Trace_Header                  = $ASCIC ('!/*** Beginning the CRD Tracing Facility',
; P  0375  1                                                       '!/*** All statements that begin with "***" ',
;    0376  1                                                       'are CRD trace statements!/!/'),
;    0377  1
;    0378  1
;    0379  1              T_Trace_1                       = $ASCIC ('*** DSR$Load_CRD: Starting location of CRD: !XL(X)!/'),
;    0380  1              T_Trace_2                       = $ASCIC ('*** DSR$Load_CRD: Size of the CRD image:    !XL(X)!/!/'),
;    0381  1
;    0382  1              T_Trace_4                       = $ASCIC ('*** DSR$Load_CRD: Vector count:                DS = !4SB  CRD = !4SB!
;    0383  1              T_Trace_5                       = $ASCIC ('*** DSR$Load_CRD: Positive version number: DS = !4SB  CRD = !4SB!
;    0384  1              T_Trace_6                       = $ASCIC ('*** DSR$Load_CRD: Negative version number: DS = !4SB  CRD = !4SB!
;    0385  1              T_Trace_7                       = $ASCIC ('*** DSR$Load_CRD: Null byte:                   DS = !4SB  CRD = !4SB!
;    0386  1      %SBTTL  'Own Storage'
;    0387  1
;    0388  1      OWN
;    0389  1              !+
;    0390  1              ! An array containing the original values in the DSVECS dispatch vectors. Anytime the vectors starting at
;    0391  1              ! DS$AL_DS_Dispatch_Vectors (DSVECS module) do not contain what they originally had, this array does contain
;    0392  1              ! their original values. Otherwise, this array contains %X'FFFFFFFF' so as to provide a debugging check.
;    0393  1              !-
;    0394  1
;    0395  1              DS$A_CRD_Vectors                : VECTOR [CRD$K_Total_Numb_Vectors, LONG]                              ![07
;    0396  1                                                INITIAL (REP CRD$K_Total_Numb_Vectors OF (%X'FFFFFFFF')),           ![07
;    0397  1
;    0398  1              !+
;    0399  1              ! A descriptor pointing to the DS$A_CRD_Vectors array. This descriptor is used to access the array.
```

```
;    0400  1              !-
;    0401  1
;    0402  1              DS$Q_CRD_Vector_Desc            : BLOCK [8, BYTE]
;    0403  1                                                PRESET (
;    0404  1                                                      [DSC$W_Length]  = 0,
;    0405  1                                                      [DSC$A_Pointer] = 0,
;    0406  1                                                      [DSC$B_DType]   = 0,
;    0407  1                                                      [DSC$B_Class]   = 0
;    0408  1                                                ),
;    0409  1
;    0410  1              !+
;    0411  1              ! A 'descriptor' for the CRD image itself.
;    0412  1              !-
;    0413  1
;    0414  1              DS$Q_CRD_Image_Desc             : VECTOR [2, LONG]
;    0415  1                                                PRESET (
;    0416  1                                                      [K_Image_Size] = 0,
;    0417  1                                                      [K_Image_Ptr]  = 0
;    0418  1                                                ),
;    0419  1
;    0420  1              !+
;    0421  1              ! Save the value of the DS$GA_LastAdr before it is changed in the Load_CRD routine.
;    0422  1              !-
;    0423  1
;    0424  1              DS$A_Saved_LastAdr              : INITIAL (0),
;    0425  1
;    0426  1              !+
;    0427  1              ! Save the contents of the DSA$GL_Flags longword before starting CRD - for restoring it to it's
;    0428  1              ! original value. Do this for the DS flags also.
;    0429  1              !-
;    0430  1
;    0431  1              DS$L_Saved_DSA_Flags            : INITIAL (0),
;    0432  1              DS$L_Saved_DS_Flags             : INITIAL (0),
;    0433  1
;    0434  1              !+
;    0435  1              ! Points to either T_Menu, T_Auto, or T_Null.
;    0436  1              !-
;    0437  1
;    0438  1              DS$A_Auto_Or_Menu;
;    0439  1
;    0440  1      MACRO                                                                                          ![07
; M  0441  1              Module_Debug (Debug_String) =                                                          ! V
; M  0442  1                      %IF %VARIANT
; M  0443  1                      %THEN
; M  0444  1                              !+
; M  0445  1                              ! If this module has been compiled with the /VARIANT:n qualifier where n <> 0, then the foll
; M  0446  1                              ! debugging statements will be compiled also. This can be invoked with the following:
; M  0447  1                              !       Module_Debug ('A debug string');
; M  0448  1                              !-
; M  0449  1                              BEGIN
; M  0450  1                              MAP
; M  0451  1                                      DS$A_CRD_Vectors     : VECTOR [, LONG],
; M  0452  1                                      DS$Q_CRD_Image_Desc  : VECTOR [, LONG],
; M  0453  1                                      DS$Q_CRD_Vector_Desc : VECTOR [, LONG];
; M  0454  1
; M  0455  1                              BIND
; M  0456  1                                      Debug_Out_1 = $ASCIC ('**** ', Debug_String, '!/'),
```

```
;  M 0457  1                    Debug_Out_2 = $ASCIC ('****     DSA Flags       = !XL, DS Flags         = !XL!/'),
;  M 0458  1                    Debug_Out_3 = $ASCIC ('****     Image_Desc  [0] = !XL, Image_Desc  [1] = !XL!/'),
;  M 0459  1                    Debug_Out_4 = $ASCIC ('****     Vector_Desc [0] = !XL, Vector_Desc [1] = !XL!/'),
;  M 0460  1                    Debug_Out_5 = $ASCIC ('****     Saved_Vector[0] = !XL, Saved_Vector[1] = !XL!/'),
;  M 0461  1                    Debug_Out_6 = $ASCIC ('****     Saved_LastAdr   = !XL, DS$GA__LastAdr  = !XL!/');
;  M 0462  1
;  M 0463  1                IF (.DSA$V_CRD_MenuTest_Off OR
;  M 0464  1                    .DSA$V_CRD_AutoTest_Off OR
;  M 0465  1                    .DSA$V_CRD_MenuTest_On) THEN
;  M 0466  1                                            !                                  [10]
;  M 0467  1                    BEGIN
;  M 0468  1                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_1);
;  M 0469  1                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_2,
;  M 0470  1                        .DSA$GL_Flags, .DS$GL_Flags);
;  M 0471  1                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_3,
;  M 0472  1                        .DS$Q_CRD_Image_Desc [0], .DS$Q_CRD_Image_Desc [1]);
;  M 0473  1                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_4,
;  M 0474  1                        .DS$Q_CRD_Vector_Desc [0], .DS$Q_CRD_Vector_Desc [1]);
;  M 0475  1                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_5,
;  M 0476  1                        .DS$A_CRD_Vectors [0], .DS$A_CRD_Vectors [1]);
;  M 0477  1                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, Debug_Out_6,
;  M 0478  1                        .DS$A_Saved_LastAdr, .DS$GA_LastAdr);
;  M 0479  1                    END
;  M 0480  1                ELSE
;  M 0481  1                    BEGIN
;  M 0482  1                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_1);
;  M 0483  1                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_2,
;  M 0484  1                        .DSA$GL_Flags, .DS$GL_Flags);
;  M 0485  1                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_3,
;  M 0486  1                        .DS$Q_CRD_Image_Desc [0], .DS$Q_CRD_Image_Desc [1]);
;  M 0487  1                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_4,
;  M 0488  1                        .DS$Q_CRD_Vector_Desc [0], .DS$Q_CRD_Vector_Desc [1]);
;  M 0489  1                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_5,
;  M 0490  1                        .DS$A_CRD_Vectors [0], .DS$A_CRD_Vectors [1]);
;  M 0491  1                    $Print (DS$K_Type_General, DS$K_PrintF, Debug_Out_6,
;  M 0492  1                        .DS$A_Saved_LastAdr, .DS$GA_LastAdr);
;  M 0493  1                    END
;  M 0494  1                END
;  M 0495  1            %FI                                                                ! ^
;     0496  1            %;                                                                 ![07
;     0497  1        %SBTTL 'DSR$Load_CRD Routine'
;     0498  1
;     0499  1        GLOBAL ROUTINE DSR$Load_CRD : NOVALUE =
;     0500  1
;     0501  1        !++
;     0502  1        ! Functional Description:
;     0503  1        !
;     0504  1        !       Load the CRD Image file into memory starting at the last address in the Supervisor. This address
;     0505  1        !       is contained in the variable DS$GA_LastAdr. Reset this variable after CRD has been loaded in to point
;     0506  1        !       to the last address used by CRD. Expand the region occupied by the Supervisor.
;     0507  1        !
;     0508  1        ! Formal Parameters:
;     0509  1        !
;     0510  1        !       None
;     0511  1        !
;     0512  1        ! Side Effects:
;     0513  1        !
```

```
;    0514  1      !       Changes the address contained in DS$GA_LastAdr.
;    0515  1      !       Loads the CRD image file into memory starting at the last address used by the Resident-DS.
;    0516  1      !       Exchange dispatch vectors with CRD and DS for the CRD routines.
;    0517  1      !
;    0518  1      ! Completion Codes:
;    0519  1      !
;    0520  1      !       None
;    0521  1      !--
;    0522  2      BEGIN                                   ! Routine DSR$Load_CRD
;    0523  2              LOCAL
;    0524  2
;    0525  2              xab_size,                  ! size of the image loaded                                  [15]
;    0526  2              CRD_Actual_Length,         ! The actual length of the file in bytes
;    0527  2              CRD_File_Attributes,       ! The attributes of the CRD image file
;    0528  2              file,
;    0529  2              default,
;    0530  2              retlen,
;    0531  2              retrec,
;    0532  2
;    0533  2              CRD_Image_Location : VECTOR [2, LONG],
;    0534  2              fab : $fab_decl,                        ! FAB for file access                       [15]
;    0535  2              xab : $xabfhc_decl;                     ! XAB for file access                       [15]
;    0536  2
;    0537  2              MAP
;    0538  2                      DSA$GL_SID : VECTOR [, BYTE],
;    0539  2                      file : ref vector [2, long],
;    0540  2                      default : ref vector [2, long];
;    0541  2
;    0542  2              REGISTER
;    0543  2                      Page_Count,             ! The number of pages of memory required for CRD image
;    0544  2                      Status;                 ! A status variable
;    0545  2
;    0546  2              Module_Debug ('DSR$Load_CRD - Start:');                                             ![07
;    0547  2
;    0548  2              IF (NOT $DS_MMOff) THEN
;    0549  2                      CRD_Error (K_MM_Is_On, 0, K_Region_Not_Expanded);
;    0550  2
;    0551  2              DS$L_Saved_DSA_Flags = .DSA$GL_Flags;
;    0552  2              DS$L_Saved_DS_Flags  = .DS$GL_Flags;
;    0553  2
;    0554  2              !+
;    0555  2              ! First, make sure that the Binary flag is set so that any subsequent $Print's with the
;    0556  2              ! CRD_AutoTest typecode won't get the first byte of the output string cut off (with Binary
;    0557  2              ! set, the TypeCode is stuck onto the front of the output string, and then if the TypeCode
;    0558  2              ! is CRD_AutoTest, then this first byte gets chopped off - by DSX$TypeOut in the PRINT module).
;    0559  2              !-
;    0560  2
;    0561  2              DSA$V_Binary = On;
;    0562  2
;    0563  2              IF (.DSA$V_CRD_Trace) THEN
;    0564  2                      $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_Header);
;    0565  2
;    0566  2              !+
;    0567  2              ! If the Debug longword is set (via the SET CRD/DEBUG/TRACE command), set the Debug flag bit.   ![07]
;    0568  2              ! This has to be done because the CRD_Error routine has to check a debug flag to see if the    ![07]
;    0569  2              ! Internal_Error routine should be called. However, at the time this check is done, the        ![07]
;    0570  2              ! CRD_Debug longword has the address of the start of the CRD image (due to the transfer of     ![07]
```

```
;    0571  2              ! vectors). So, we use another flag to say that debug is on for this one case only.                ![07]
;    0572  2              !-
;    0573  2
;    0574  2              IF (.DS$GL_CRD_Debug) THEN                                                                        ![07]
;    0575  2                      CRD$V_CRD_Debug = True;                                                                   ![07]
;    0576  2
;    0577  2              IF (.DSA$V_CRD_AutoTest_Off) THEN
;    0578  2                      DS$A_Auto_Or_Menu = T_Auto
;    0579  2              ELSE IF (.DSA$V_CRD_MenuTest_Off OR .DSA$V_CRD_MenuTest_On) THEN
;    0580  2                                      !                                                          [10]
;    0581  3                      BEGIN                                                                                     ![06]
;    0582  3                      DS$A_Auto_Or_Menu = T_Menu;                                                               ![06]
;    0583  3                      CRD$V_Ctrl_C_No_Echo = True;                                                              ![07]
;    0584  3                      CRD$V_Flush_Ctrl_C = False;                                                               ![07]
;    0585  3                      END                                                                                       ![06]
;    0586  2              ELSE
;    0587  3                      BEGIN
;    0588  3                      DS$A_Auto_Or_Menu = T_Null;
;    0589  3                      CRD_Error (K_No_DSA_Bit_Set, .DSA$GL_Flags, K_Region_Not_Expanded);
;    0590  2                      END;
;    0591  2
;    0592  2              IF (.DS$Q_CRD_Vector_Desc [DSC$W_Length] EQL 0) THEN
;    0593  3                      BEGIN
;    0594  3                      MAP
;    0595  3                              DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE, SIGNED];
;    0596  3
;    0597  3                      !+
;    0598  3                      ! These stored values will later be put back into the dispatch area when CRD exits. Store all
;    0599  3                      ! the vectors plus the 4 bytes at the beginning of the dispatch vectors. Store a pointer to the
;    0600  3                      ! block, and the ACTUAL size of the block in a descriptor.
;    0601  3                      !-
;    0602  3
;    0603  3                      DS$Q_CRD_Vector_Desc [DSC$W_Length] = CRD$K_Total_Numb_Vectors * 4;                       ![07
;    0604  3                      DS$Q_CRD_Vector_Desc [DSC$A_Pointer] = DS$A_CRD_Vectors;
;    0605  3
;    0606  3                      !+
;    0607  3                      ! Now transfer the four bytes at the beginning (version numbers, etc.) and the actual
;    0608  3                      ! dispatch vectors to the temporary storage.
;    0609  3                      !-
;    0610  3
;    0611  3                      INCR Vector_Index FROM 0 TO (.DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors] - 1) DO     ![07
;    0612  4                              BEGIN
;    0613  4                              MAP
;    0614  4                                      DS$AL_DS_Dispatch_Vectors : VECTOR [, LONG];
;    0615  4
;    0616  4                              BIND
;    0617  4                                      Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, LONG];
;    0618  4
;    0619  4                              Temp_Vectors [.Vector_Index] = .DS$AL_DS_Dispatch_Vectors [.Vector_Index];
;    0620  4                              END
;    0621  3                      END
;    0622  2              ELSE
;    0623  2                      Module_Debug ('DSR$Load_CRD: ***** ERROR *****');
;    0624  2
;    0625  2              !+
;    0626  2              ! Now do a 'fake' Load of the CRD image. This will return the size of the image, in
;    0627  2              ! preparation for allocating the necessary memory and loading the image into memory.
```

```
;    0628  2              !-
;    0629  2
;    0630  2              file    = T_CRD_Image_Name;
;    0631  2              default = T_CRD_Image_Default;
;    0632  2              retlen  = CRD_Actual_Length;
;    0633  2              retrec  = CRD_File_Attributes;
;    0634  2
; P  0635  2          $fab_init (fab = fab, fna = .file [1], fns = .file [0], dna = .default [1], dns = .default [0],
;    0636  2               xab = xab);                                                                          ![15]
;    0637  2          $xabfhc_init (xab = xab);                                                                 ![15]
;    0638  2
;    0639  2          if not (status = $open (fab = fab)) then
;    0640  2              CRD_Error (K_Load_File, .Status, K_Region_Not_Expanded);                              ![15]
;    0641  2
;    0642  2          XAB_SIZE = .xab [xab$l_ebk]^9 + .xab [xab$w_ffb] - 512 ;                                  ![15]
;    0643  2
;    0644  2
;    0645  3              begin
;    0646  3
;    0647  3              map
;    0648  3                  retrec : ref block [, byte];
;    0649  3
;    0650  3                  retrec [0, 0, 16, 0] = .fab [fab$w_mrs];
;    0651  3                  retrec [2, 0, 8, 0] = .fab [fab$b_rfm];
;    0652  3                  retrec [3, 0, 8, 0] = .fab [fab$b_fsz];
;    0653  2              end;
;    0654  2
;    0655  2           $close (fab = fab);                                                                      ![15]
;    0656  2
;    0657  2          .retlen =  .XAB_SIZE;
;    0658  2                    ! Bigger of XAB size and actually loaded size                                  ![15]
;    0659  2
;    0660  2
;    0661  3              IF (NOT (.Status))                                                                    ![15]
;    0662  2              THEN
;    0663  2                      CRD_Error (K_Determine_Size, .Status, K_Region_Not_Expanded);
;    0664  2
;    0665  2
;    0666  2              DS$A_Saved_LastAdr = .DS$GA_LastAdr;
;    0667  2                                   ! Save the old value.
;    0668  2
;    0669  2              Page_Count = (.CRD_Actual_Length + 511) / 512;
;    0670  2                                   ! Number of pages required to load CRD
;    0671  2
;    0672  2              IF (.DSA$V_User) THEN                                                                 ![04
;    0673  3                      BEGIN                                                                         ![04
;    0674  3                      !+
;    0675  3                      ! Use this one for running on-line.
;    0676  3                      !-
; P  0677  3                      Status = $ExpReg
; P  0678  3                                   (
; P  0679  3                                   PagCnt = .Page_Count,
; P  0680  3                                   RetAdr = CRD_Image_Location,
; P  0681  3                                   Region = 0
;    0682  4                                   )       ! Expand the Supervisor region by the number of pages
;    0683  4                                           ! required to hold the CRD image, set the access mode to
;    0684  4                                           ! User-Write, expand P0 space.
```

```
;   0685  3                 END                                                                                    ![04
;   0686  2             ELSE                                                                                       ![04
;   0687  3                 BEGIN                                                                                  ![04
;   0688  3                 !+
;   0689  3                 ! Use this one for running stand-alone.
;   0690  3                 !-
;   0691  3                 Status = CRD$ExpReg
;   0692  3                                 (
;   0693  3                                 .Page_Count,
;   0694  3                                 CRD_Image_Location,
;   0695  3                                 0,
;   0696  3                                 3,
;   0697  3                                 0       ! Specify PTE ownership code.                          [18]
;   0698  3                                 );      ! Expand the Supervisor region by the number of pages
;   0699  3                                         ! required to hold the CRD image, use no access mode,
;   0700  3                                         ! expand DS space (3).
;   0701  2                 END;                                                                                   ![04
;   0702  2
;   0703  2             DS$Q_CRD_Image_Desc [K_Image_Size] =
;   0704  2                 .CRD_Image_Location [K_Ending_Adr] - .CRD_Image_Location [K_Starting_Adr] + 1;
;   0705  2
;   0706  2             DS$Q_CRD_Image_Desc [K_Image_Ptr] = .CRD_Image_Location [K_Starting_Adr];
;   0707  2
;   0708  2             IF (NOT .Status) THEN
;   0709  2                 !+
;   0710  2                 ! If an error occurred, check the return address array. If both addresses are -1, then
;   0711  2                 ! no memory was expanded. If they are not both zero, some was expanded.
;   0712  2                 !-
;   0713  3                 IF ((.CRD_Image_Location [K_Starting_Adr] EQL -1) AND
;   0714  3                     (.CRD_Image_Location [K_Ending_Adr]   EQL -1))
;   0715  2                 THEN
;   0716  3                         BEGIN
;   0717  3                         DS$Q_CRD_Image_Desc [K_Image_Size] = 0;
;   0718  3                         DS$Q_CRD_Image_Desc [K_Image_Ptr]  = 0;
;   0719  3
;   0720  3                         CRD_Error (K_Expand_Region, .Status, K_Region_Not_Expanded);
;   0721  3                         END
;   0722  2                 ELSE
;   0723  2                         CRD_Error (K_Expand_Region, .Status, K_Region_Expanded);
;   0724  2
; P 0725  2             Status = $DS_LOAD (
; P 0726  2                             File    = T_CRD_Image_Name,                                                ![03
; P 0727  2                             Default = T_CRD_Image_Default,                                             ![03
; P 0728  2                             Length  = .CRD_Actual_Length,
; P 0729  2                             Address = .CRD_Image_Location [K_Starting_Adr],
; P 0730  2                             RetLen  = CRD_Actual_Length,
; P 0731  2                             RetRec  = CRD_File_Attributes,
; P 0732  2                             VBN     = 1
;   0733  2                            );           ! Actually load the file into the expanded region
;   0734  2
;   0735  2             IF (NOT .Status) THEN
;   0736  2                 CRD_Error (K_Load_File, .Status, K_Region_Expanded);
;   0737  2
;   0738  2             DS$GA_LastAdr = .DS$GA_LastAdr + .CRD_Actual_Length;
;   0739  2                                         ! Reset the LastAdr pointer
;   0740  2
;   0741  2             IF (.DSA$V_CRD_Trace) THEN
```

```
;    0742  3                    BEGIN
;    0743  3                    MAP
;    0744  3                            DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE, SIGNED];
;    0745  3
;    0746  3                    BIND
;    0747  3                            CRD_Bytes = (.CRD_Image_Location [K_Starting_Adr]) : VECTOR [, BYTE, SIGNED];
;    0748  3
;    0749  3                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_1, .DS$Q_CRD_Image_Desc [K_Image_Ptr]);
;    0750  3                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_2, .DS$Q_CRD_Image_Desc [K_Image_Size]);
;    0751  3
; P  0752  3                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_4,
; P  0753  3                            .DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors],
;    0754  3                            .CRD_Bytes [K_Number_Of_Vectors]);
;    0755  3
; P  0756  3                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_5,
; P  0757  3                            .DS$AL_DS_Dispatch_Vectors [K_Positive_Version],
;    0758  3                            .CRD_Bytes [K_Positive_Version]);
;    0759  3
; P  0760  3                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_6,
; P  0761  3                            .DS$AL_DS_Dispatch_Vectors [K_Negative_Version],
;    0762  3                            .CRD_Bytes [K_Negative_Version]);
;    0763  3
; P  0764  3                    $Print (DS$K_Type_CRD_AutoTest, DS$K_PrintF, T_Trace_7,
; P  0765  3                            .DS$AL_DS_Dispatch_Vectors [K_Null_Byte],
;    0766  3                            .CRD_Bytes [K_Null_Byte]);
;    0767  2                    END;
;    0768  2
;    0769  2            Exchange_Dispatch_Vectors (.CRD_Image_Location [K_Starting_Adr]);                    ![03
;    0770  2                                       ! Exchange all the dispatch vectors for CRD and DS
;    0771  2
;    0772  2            Module_Debug ('DSR$Load_CRD - End:');                                               ![07
;    0773  1        END;                           ! Routine DSR$Load_CRD


                                                                .TITLE  CRDIMAGE *** Loading and Unloading the CRD Imag
;                                                                       e
                                                                .IDENT  \08-19\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                    45  47  41  4D  49  44  52  43  08  00000 P.AAA:  .ASCII  <8>\CRDIMAGE\                    ;
                30  3B  45  58  45  2E  42  41  53  4E  45  00009 P.AAC:  .ASCII  \ENSAB.EXE;0\                 ;
                                                0000000B  00014 P.AAB:  .LONG   11                              ;
                                                00000000' 00018         .ADDRESS P.AAC                      ;
                                                          0001C P.AAE:  .BLKB   0                           ;
                                                00000000  0001C P.AAD:  .LONG   0                             ;
                                                00000000' 00020         .ADDRESS P.AAE                      ;
                                                      00  00024 P.AAF:  .ASCII  <0>                         ;
                    54  53  45  54  20  55  4E  45  4D  09  00025 P.AAG:  .ASCII  <9>\MENU TEST\              ;
                    54  53  45  54  20  4F  54  55  41  09  0002F P.AAH:  .ASCII  <9>\AUTO TEST\              ;
49  46  20  4F  54  20  45  4C  42  41  4E  55  2F  21  3E  00039 P.AAI:  .ASCII  \>!/UNABLE TO FIND FILE ENSAB, THE LOADAB\ ;
20  2C  42  41  53  4E  45  20  45  4C  49  46  20  44  4E  00048                                             ;
                    42  41  44  41  4F  4C  20  45  48  54  00057                                             ;
43  20  46  4F  20  4E  4F  49  54  52  4F  50  20  45  4C  00061         .ASCII  \LE PORTION OF CRD !AC!/\       ;
                        2F  21  43  41  21  20  44  52  00070                                                 ;
4E  49  44  41  4F  4C  20  52  4F  52  52  45  2F  21  3D  00078 P.AAJ:  .ASCII  \=!/ERROR LOADING FILE ENSAB, THE LOADABL\ ;
54  20  2C  42  41  53  4E  45  20  45  4C  49  46  20  47  00087                                             ;
```

```
                        4C 42 41 44 41 4F 4C 20 45 48  00096
52 43 20 46 4F 20 4E 4F 49 54 52 4F 50 20 45 48  000A0          .ASCII  \E PORTION OF CRD !AC!/\
                        2F 21 43 41 21 20 44 48  000AF
20 45 4C 49 48 57 20 52 4F 52 52 45 2F 21 2A 48  000B6 P.AAK:   .ASCII  \*!/ERROR WHILE ATTEMPTING TO EXIT CRD !A\
45 20 4F 54 20 47 4E 49 54 50 4D 45 54 54 41 48  000C5
                        41 21 20 44 52 43 20 54 49 58  000D4
                                    2F 21 43 48  000DE          .ASCII  \C!/\
45 56 20 54 43 45 52 52 4F 43 4E 49 2F 21 20 48  000E1 P.AAL:   .ASCII  \ !/INCORRECT VERSION OF CRD !AC!/\
41 21 20 44 52 43 20 46 4F 20 4E 4F 49 53 52 48  000F0
                                    2F 21 43 48  000FF
45 4C 42 41 44 41 4F 4C 20 45 48 54 2F 21 30 48  00102 P.AAM:   .ASCII  \0!/THE LOADABLE PORTION OF CRD !AC IS CO\
44 52 43 20 46 4F 20 4E 4F 49 54 52 4F 50 20 48  00111
                        4F 43 20 53 49 20 43 41 21 20 48  00120
                              2F 21 44 45 54 50 55 52 52 48  0012A          .ASCII  \RRUPTED!/\
49 54 4E 4F 43 20 54 4F 4E 4E 41 43 2F 21 86 48  00133 P.AAN:   .ASCII  <134>\!/CANNOT CONTINUE CRD - MEMORY MAN\
52 4F 4D 45 4D 20 2D 20 44 52 43 20 45 55 4E 48  00142
                              4E 41 4D 20 59 48  00151
45 42 20 54 53 55 4D 20 54 4E 45 4D 45 47 41 48  00156          .ASCII  \AGEMENT MUST BE OFF!/TO TURN OFF MEMORY \
4F 20 4E 52 55 54 20 4F 54 2F 21 46 46 4F 20 48  00165
                        20 59 52 4F 4D 45 4D 20 46 46 48  00174
50 59 54 20 2C 54 4E 45 4D 45 47 41 4E 41 4D 48  0017E          .ASCII  \MANAGEMENT, TYPE "SET MM OFF"!/RESTART C\
21 22 46 46 4F 20 4D 4D 20 54 45 53 22 20 45 48  0018D
                        43 20 54 52 41 54 53 45 52 2F 48  0019C
43 22 20 47 4E 49 50 59 54 20 59 42 20 44 52 48  001A6          .ASCII  \RD BY TYPING "CRD"!/\
                              2F 21 22 44 52 48  001B5
20 54 53 45 54 20 4F 54 55 41 2F 21 2F 21 2A 48  001BA P.AAO:   .ASCII  \*!/!/AUTO TEST ABORTED - CALL FIELD SERV\
20 4C 4C 41 43 20 2D 20 44 45 54 52 4F 42 41 48  001C9
                        56 52 45 53 20 44 4C 45 49 46 48  001D8
                                    45 43 49 48  001E2          .ASCII  \ICE\
20 46 4F 20 44 4E 45 20 2A 2A 2A 2A 2F 21 24 48  001E5 P.AAP:   .ASCII  \$!/**** END OF VAX/CRD AUTO TEST ****\
45 54 20 4F 54 55 41 20 44 52 43 2F 58 41 56 48  001F4
                              2A 2A 2A 2A 20 54 53 48  00203
20 47 4E 49 4E 52 55 54 45 52 2F 21 2F 21 35 48  0020A P.AAQ:   .ASCII  \5!/!/RETURNING TO VAX CONSOLE, WAIT FOR \
2C 45 4C 4F 53 4E 4F 43 20 58 41 56 20 4F 54 48  00219
                        20 52 4F 46 20 54 49 41 57 20 48  00228
      2F 21 54 50 4D 4F 52 50 20 22 3E 3E 3E 22 48  00232          .ASCII  \">>>" PROMPT!/\
74 69 62 20 41 53 44 20 6F 4E 20 2A 2A 2A 2F 48  00240 P.AAR:   .ASCII  \/*** No DSA bit is set - DSA$GL_Flags = \
47 24 41 53 44 20 2D 20 74 65 73 20 73 69 20 48  0024F
                        20 3D 20 73 67 61 6C 46 5F 4C 48  0025E
                              2F 21 29 58 28 4C 58 21 48  00268          .ASCII  \!XL(X)!/\
6C 69 68 77 20 72 6F 72 72 45 20 2A 2A 2A 46 48  00270 P.AAS:   .ASCII  \F*** Error while determining size of the\
73 20 67 6E 69 6E 69 6D 72 65 74 65 64 20 65 48  0027F
                        65 68 74 20 66 6F 20 65 7A 69 48  0028E
53 20 2D 20 65 6C 69 66 20 42 41 53 4E 45 20 48  00298          .ASCII  \ ENSAB file - Status = !XL(X)!/\
21 29 58 28 4C 58 21 20 3D 20 73 75 74 61 74 48  002A7
                                    2F 48  002B6
6C 69 68 77 20 72 6F 72 72 45 20 2A 2A 2A 38 48  002B7 P.AAT:   .ASCII  \8*** Error while expanding the region - \
65 68 74 20 67 6E 69 64 6E 61 70 78 65 20 65 48  002C6
                        20 2D 20 6E 6F 69 67 65 72 20 48  002D5
29 58 28 4C 58 21 20 3D 20 73 75 74 61 74 53 48  002DF          .ASCII  \Status = !XL(X)!/\
                                    2F 21 48  002EE
6C 69 68 77 20 72 6F 72 72 45 20 2A 2A 2A 46 48  002F0 P.AAU:   .ASCII  \F*** Error while loading the ENSAB file \
45 20 65 68 74 20 67 6E 69 64 61 6F 6C 20 65 48  002FF
                        20 65 6C 69 66 20 42 41 53 4E 48  0030E
53 20 2D 20 79 72 6F 6D 65 6D 20 6F 74 6E 69 48  00318          .ASCII  \into memory - Status = !XL(X)!/\
21 29 58 28 4C 58 21 20 3D 20 73 75 74 61 74 48  00327
                                    2F 48  00336
```

```
65 74 6E 49 20 44 52 43 20 41 20 2A 2A 2A 23  00337 P.AAV:  .ASCII  \#*** A CRD Internal Error occurred!/\
75 63 63 6F 20 72 6F 72 72 45 20 6C 61 6E 72  00346
                           2F 21 64 65 72 72  00355
77 6F 6C 6C 6F 66 20 65 68 54 20 2A 2A 2A 2E  0035B P.AAW:  .ASCII  \.*** The following are the CRD debug vec\
44 52 43 20 65 68 74 20 65 72 61 20 67 6E 69  0036A
               63 65 76 20 67 75 62 65 64 20  00379
                        2F 21 3A 73 72 6F 74  00383          .ASCII  \tors:!/\
74 63 65 76 20 67 75 62 65 44 20 2A 2A 2A 2B  0038A P.AAX:  .ASCII  \+*** Debug vector [!UL]: !XL(X)   !-!SL(\
58 28 4C 58 21 20 3A 5D 4C 55 21 5B 20 72 6F  00399
               28 4C 53 21 2D 21 20 20 20 29  003A8
                                 2F 21 29 44  003B2          .ASCII  \D)!/\
6E 6F 69 67 65 72 20 65 68 54 20 2A 2A 2A 35  003B6 P.AAY:  .ASCII  \5*** The region was not contracted - Sta\
61 72 74 6E 6F 63 20 74 6F 6E 20 73 61 77 20  003C5
               61 74 53 20 2D 20 64 65 74 63  003D4
    2F 21 29 58 28 4C 58 21 20 3D 20 73 75 74  003DE          .ASCII  \tus = !XL(X)!/\
65 68 74 20 72 65 68 74 69 45 20 2A 2A 2A 48  003EC P.AAZ:  .ASCII  \H*** Either the Null byte is not null, o\
6E 20 73 69 20 65 74 79 62 20 6C 6C 75 4E 20  003FB
               6F 20 2C 6C 6C 75 6E 20 74 6F  0040A
6E 20 6E 6F 69 73 72 65 76 20 65 68 74 20 72  00414          .ASCII  \r the version numbers are wrong!/\
6E 6F 72 77 20 65 72 61 20 73 72 65 62 6D 75  00423
                                 2F 21 67  00432
73 72 65 76 20 65 73 65 68 54 20 2A 2A 2A 38  00435 P.ABA:  .ASCII  \8*** These versions of ENSAB and E*SAA a\
61 20 42 41 53 4E 45 20 66 6F 20 73 6E 6F 69  00444
               61 20 41 41 53 2A 45 20 64 6E  00453
65 6C 62 69 74 61 70 6D 6F 63 6E 69 20 65 72  0045D          .ASCII  \re incompatible!/\
                                 2F 21 6F  0046C
6E 69 6E 6E 69 67 65 42 20 2A 2A 2A 2F 21 6F  0046E P.ABB:  .ASCII  \o!/*** Beginning the CRD Tracing Facilit\
69 63 61 72 54 20 44 52 43 20 65 68 74 20 67  0047D
               74 69 6C 69 63 61 46 20 67 6E  0048C
74 61 74 73 20 6C 6C 41 20 2A 2A 2A 2F 21 79  00496          .ASCII  \y!/*** All statements that begin with "*\
67 65 62 20 74 61 68 74 20 73 74 6E 65 6D 65  004A5
               2A 22 20 68 74 69 77 20 6E 69  004B4
61 72 74 20 44 52 43 20 65 72 61 20 22 2A 2A  004BE          .ASCII  \**" are CRD trace statements!/!/\
2F 21 73 74 6E 65 6D 65 74 61 74 73 20 65 63  004CD
                                 2F 21  004DC
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 34  004DE P.ABC:  .ASCII  \4*** DSR$Load_CRD: Starting location of \
6F 6C 20 67 6E 69 74 72 61 74 53 20 3A 44 52  004ED
               20 66 6F 20 6E 6F 69 74 61 63  004FC
         2F 21 29 58 28 4C 58 21 20 3A 44 52 43  00506          .ASCII  \CRD: !XL(X)!/\
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 36  00513 P.ABD:  .ASCII  \6*** DSR$Load_CRD: Size of the CRD image\
65 68 74 20 66 6F 20 65 7A 69 53 20 3A 44 52  00522
               65 67 61 6D 69 20 44 52 43 20  00531
2F 21 2F 21 29 58 28 4C 58 21 20 20 20 20 3A  0053B          .ASCII  \:    !XL(X)!/!/\
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 42  0054A P.ABE:  .ASCII  \B*** DSR$Load_CRD: Vector count:       \
6E 75 6F 63 20 72 6F 74 63 65 56 20 3A 44 52  00559
               20 20 20 20 20 20 20 20 3A 74  00568
20 20 42 53 34 21 20 3D 20 53 44 20 20 20 20  00572          .ASCII  \    DS = !4SB  CRD = !4SB!/\
         2F 21 42 53 34 21 20 3D 20 44 52 43  00581
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 42  0058D P.ABF:  .ASCII  \B*** DSR$Load_CRD: Positive version numb\
65 76 20 65 76 69 74 69 73 6F 50 20 3A 44 52  0059C
               62 6D 75 6E 20 6E 6F 69 73 72  005AB
20 20 42 53 34 21 20 3D 20 53 44 20 3A 72 65  005B5          .ASCII  \er: DS = !4SB  CRD = !4SB!/\
         2F 21 42 53 34 21 20 3D 20 44 52 43  005C4
43 5F 64 61 6F 4C 24 52 53 44 20 2A 2A 2A 42  005D0 P.ABG:  .ASCII  \B*** DSR$Load_CRD: Negative version numb\
65 76 20 65 76 69 74 61 67 65 4E 20 3A 44 52  005DF
               62 6D 75 6E 20 6E 6F 69 73 72  005EE
20 20 42 53 34 21 20 3D 20 53 44 20 3A 72 65  005F8          .ASCII  \er: DS = !4SB  CRD = !4SB!/\
```

```
            2F  21  42  53  34  21  20  3D  20  44  52  43  00607                                                  ;
43  5F  64  61  6F  4C  24  52  53  44  20  2A  2A  2A  42  00613 P.ABH:  .ASCII  \B*** DSR$Load_CRD: Null byte:          \  ;
20  3A  65  74  79  62  20  6C  6C  75  4E  20  3A  44  52  00622                                                  ;
                    20  20  20  20  20  20  20  20  20  20  00631                                                  ;
20  20  42  53  34  21  20  3D  20  53  44  20  20  20  20  0063B         .ASCII  \    DS = !4SB  CRD = !4SB!/\          ;
            2F  21  42  53  34  21  20  3D  20  44  52  43  0064A                                                  ;

                                                                        .PSECT  WORK,NOEXE,2

                                                FFFFFFFF# 00000 DS$A_CRD_VECTORS:
                                                                        .LONG   -1[25]                                 ;
                                                    0000  00064 DS$Q_CRD_VECTOR_DESC:
                                                                        .WORD   0                                      ;
                                                  00  00  00066         .BYTE   0, 0                                   ;
                                                00000000  00068         .LONG   0                                      ;
                                        00000000  00000000  0006C DS$Q_CRD_IMAGE_DESC:
                                                                        .LONG   0, 0                                   ;
                                                00000000  00074 DS$A_SAVED_LASTADR:
                                                                        .LONG   0                                      ;
                                                00000000  00078 DS$L_SAVED_DSA_FLAGS:
                                                                        .LONG   0                                      ;
                                                00000000  0007C DS$L_SAVED_DS_FLAGS:
                                                                        .LONG   0                                      ;
                                                          00080 DS$A_AUTO_OR_MENU:
                                                                        .BLKB   4

                                                                DSA$AL_APTMAIL=     65024
                                                                DSA$GL_FLAGS=       65024
                                                                DSA$GL_APTCOM=      65028
                                                                DSA$GL_PASSES=      65032
                                                                DSA$GL_UNITS=       65036
                                                                DSA$GL_SECTNO=      65040
                                                                DSA$GL_SID=         65044
                                                                DSA$GL_MSGTYP=      65088
                                                                DSA$GL_ERRNO=       65092
                                                                DSA$GL_EVENT=       65096
                                                                DSA$GL_SUBTNO=      65100
                                                                DSA$GL_TESTNO=      65104
                                                                DSA$GL_PASSNO=      65108
                                                                DSA$GL_DEVLEN=      65112
                                                                DSA$GL_DEVNAM=      65116
                                                                DSA$GL_MSGPTR=      65128
                                                                $MODULE=            P.AAA
                                                                T_CRD_IMAGE_NAME=   P.AAB
                                                                T_CRD_IMAGE_DEFAULT=P.AAD
                                                                T_NULL=             P.AAF
                                                                T_MENU=             P.AAG
                                                                T_AUTO=             P.AAH
                                                                T_FILE_NOT_FOUND=   P.AAI
                                                                T_UNKNOWN_ERROR=    P.AAJ
                                                                T_CONTRACTION_ERROR=P.AAK
                                                                T_INCOMPATIBILITY_ERROR=
                                                                                    P.AAL
                                                                T_CORRUPTED_ERROR=  P.AAM
                                                                T_MM_ON=            P.AAN
                                                                T_AUTO_ABORT_CRD=   P.AAO
                                                                T_AUTO_END_MESSAGE= P.AAP
```

```
                                                    T_AUTO_EXIT_TO_CONSOLE=
                                                                    P.AAQ
                                                    T_NO_DSA_BIT_SET=   P.AAR
                                                    T_DETERMINE_SIZE=   P.AAS
                                                    T_EXPAND_REGION=    P.AAT
                                                    T_LOAD_FILE=        P.AAU
                                                    T_CRD_INTERNAL=     P.AAV
                                                    T_CRD_INTERNAL_HEADER=
                                                                    P.AAW
                                                    T_CRD_INTERNAL_DEBUG=
                                                                    P.AAX
                                                    T_NOT_CONTRACTED=   P.AAY
                                                    T_CORRUPTED=        P.AAZ
                                                    T_INCOMPATIBLE=     P.ABA
                                                    T_TRACE_HEADER=     P.ABB
                                                    T_TRACE_1=          P.ABC
                                                    T_TRACE_2=          P.ABD
                                                    T_TRACE_4=          P.ABE
                                                    T_TRACE_5=          P.ABF
                                                    T_TRACE_6=          P.ABG
                                                    T_TRACE_7=          P.ABH
                                                            .EXTRN  DS_CLEANUP, MAPFREE
                                                            .EXTRN  CRD$EXPREG, EXE$CNTREG
                                                            .EXTRN  DSR$CHECK_MENUTEST_OFF_SET
                                                            .EXTRN  DSR$CHECK_AUTOTEST_OFF_SET
                                                            .EXTRN  EXE$ALONONPAGED
                                                            .EXTRN  EXE$DEANONPAGED
                                                            .EXTRN  DSV$EXIT, BEGIN_BLISS
                                                            .EXTRN  DS$GA_DS_CTRL_C_FIRST
                                                            .EXTRN  DS$GA_DS_CTRL_C_SECOND
                                                            .EXTRN  DS$L_USERCNTRLC
                                                            .EXTRN  DS$GL_FLAGS, DS$AL_DS_DISPATCH_VECTORS
                                                            .EXTRN  DS$GL_CRD_FLAGS
                                                            .EXTRN  DS$GA_LASTADR, DS$GL_CRD_DEBUG
                                                            .EXTRN  DS$MMOFF, DSX$PRINT
                                                            .EXTRN  SYS$OPEN, SYS$CLOSE
                                                            .EXTRN  SYS$EXPREG, DS$LOAD

                                                            .PSECT  CODE,NOWRT,  SHR,2

                                        0FFC 00000          .ENTRY  DSR$LOAD_CRD, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0499
                                                                    R10,R11                                     ;
                          5B 00000000'  EF  9E 00002        MOVAB   DS$A_AUTO_OR_MENU, R11                      ;
                          5E     FF74   CE  9E 00009        MOVAB   -140(SP), SP                                ;
                00000000G 9F            00  FB 0000E        CALLS   #0, @#DS$MMOFF                              ; 0548
                          0C            50  E8 00015        BLBS    R0, 1$                                      ;
                                        01  DD 00018        PUSHL   #1                                          ; 0549
                          7E            08  7D 0001A        MOVQ    #8, -(SP)                                   ;
                00000000V EF            03  FB 0001D        CALLS   #3, CRD_ERROR                               ;
                      F8  AB 0000FE00   9F  D0 00024 1$:    MOVL    @#^X0000FE00, DS$L_SAVED_DSA_FLAGS          ; 0551
                      FC  AB 00000000G  EF  D0 0002C        MOVL    DS$GL_FLAGS, DS$L_SAVED_DS_FLAGS            ; 0552
                0000FE00  9F            02  88 00034        BISB2   #2, @#^X0000FE00                            ; 0561
5A 0000FE02   9F          01            01  EF 0003B        EXTZV   #1, #1, @#^X0000FE02, R10                   ; 0563
                          11            5A  E9 00044        BLBC    R10, 2$                                     ;
                             00000000'  EF  9F 00047        PUSHAB  T_TRACE_HEADER                              ; 0564
                                        01  DD 0004D        PUSHL   #1                                          ;
                                        1A  DD 0004F        PUSHL   #26                                         ;
```

```
                        00000000G  9F                03  FB 00051          CALLS   #3, a#DSX$PRINT                                 ;
                                   07 00000000G       EF  E9 00058 2$:     BLBC    DS$GL_CRD_DEBUG, 3$                             ; 0574
                        00000000G  EF                 01  88 0005F          BISB2   #1, DS$GL_CRD_FLAGS                             ; 0575
                     09 0000FE01   9F                 03  E1 00066 3$:     BBC     #3, a#^X0000FE01, 4$                            ; 0577
                                   6B 00000000'       EF  9E 0006E          MOVAB   T_AUTO, DS$A_AUTO_OR_MENU                       ; 0578
                                                      39  11 00075          BRB     7$                                              ;
                                   08 0000FE02        9F  E8 00077 4$:     BLBS    a#^X0000FE02, 5$                                ; 0579
                     12 0000FE02   9F                 02  E1 0007E          BBC     #2, a#^X0000FE02, 6$                            ;
                                   6B 00000000'       EF  9E 00086 5$:     MOVAB   T_MENU, DS$A_AUTO_OR_MENU                       ; 0582
00000000G  EF        02            01                 01  F0 0008D          INSV    #1, #1, #2, DS$GL_CRD_FLAGS                     ; 0583
                                                      18  11 00096          BRB     7$                                              ;
                                   6B 00000000'       EF  9E 00098 6$:     MOVAB   T_NULL, DS$A_AUTO_OR_MENU                       ; 0588
                                                      01  DD 0009F          PUSHL   #1                                              ; 0589
                                      0000FE00        9F  DD 000A1          PUSHL   a#^X0000FE00                                    ;
                                                      7E  D4 000A7          CLRL    -(SP)                                           ;
                        00000000V  EF                 03  FB 000A9          CALLS   #3, CRD_ERROR                                   ;
                                              E4      AB  B5 000B0 7$:     TSTW    DS$Q_CRD_VECTOR_DESC                            ; 0592
                                                      24  12 000B3          BNEQ    10$                                             ;
                              E4   AB           64     8F  9B 000B5          MOVZBW  #100, DS$Q_CRD_VECTOR_DESC                      ; 0603
                              E8   AB           80     AB  9E 000BA          MOVAB   DS$A_CRD_VECTORS, DS$Q_CRD_VECTOR_DESC+4        ; 0604
                                   51 00000000G       EF  98 000BF          CVTBL   DS$AL_DS_DISPATCH_VECTORS, R1                   ; 0611
                                   50                 01  CE 000C6          MNEGL   #1, VECTOR_INDEX                                ; 0617
                                                      0A  11 000C9          BRB     9$                                              ;
                              E8 BB40 00000000GEF40    D0 000CB 8$:     MOVL    DS$AL_DS_DISPATCH_VECTORS[VECTOR_INDEX], -      ; 0619
                                                                                 @DS$Q_CRD_VECTOR_DESC+4[VECTOR_INDEX]           ;
                     F2                50            51  F2 000D5 9$:     AOBLSS  R1, VECTOR_INDEX, 8$                            ;
                                   58 00000000'       EF  9E 000D9 10$:    MOVAB   T_CRD_IMAGE_NAME, FILE                          ; 0630
                                   57 00000000'       EF  9E 000E0          MOVAB   T_CRD_IMAGE_DEFAULT, DEFAULT                    ; 0631
                                   59           04    AE  9E 000E7          MOVAB   CRD_ACTUAL_LENGTH, RETLEN                       ; 0632
                                   56                 6E  9E 000EB          MOVAB   CRD_FILE_ATTRIBUTES, RETREC                     ; 0633
0050   8F                  00      6E                 00  2C 000EE          MOVC5   #0, (SP), #0, #80, $RMS_PTR                     ; 0636
                                                      34  AE 000F5                                                                  ;
                                 34   AE        5003   8F  B0 000F7          MOVW    #20483, $RMS_PTR                                ;
                                 4A   AE             02  90 000FD          MOVB    #2, $RMS_PTR+22                                 ;
                                 53   AE             02  90 00101          MOVB    #2, $RMS_PTR+31                                 ;
                                 58   AE          08 AE  9E 00105          MOVAB   XAB, $RMS_PTR+36                                ;
                                 60   AE          04 A8  D0 0010A          MOVL    4(FILE), $RMS_PTR+44                            ;
                                 64   AE          04 A7  D0 0010F          MOVL    4(DEFAULT), $RMS_PTR+48                         ;
                                 68   AE             68  90 00114          MOVB    (FILE), $RMS_PTR+52                             ;
                                 69   AE             67  90 00118          MOVB    (DEFAULT), $RMS_PTR+53                          ;
                           2C    00      6E             00  2C 0011C          MOVC5   #0, (SP), #0, #44, $RMS_PTR                     ; 0637
                                                      08  AE 00121                                                                  ;
                                 08   AE        2C1D   8F  B0 00123          MOVW    #11293, $RMS_PTR                                ;
                                                34    AE  9F 00129          PUSHAB  FAB                                             ; 0639
                              00000000G  00             01  FB 0012C          CALLS   #1, SYS$OPEN                                    ;
                                   52                 50  D0 00133          MOVL    R0, STATUS                                      ;
                                   0D                 52  E8 00136          BLBS    STATUS, 11$                                     ;
                                                      01  DD 00139          PUSHL   #1                                              ; 0640
                                                      52  DD 0013B          PUSHL   STATUS                                          ;
                                                      03  DD 0013D          PUSHL   #3                                              ;
                              00000000V  EF             03  FB 0013F          CALLS   #3, CRD_ERROR                                   ;
                     50           18   AE             09  78 00146 11$:     ASHL    #9, XAB+16, R0                                  ; 0642
                                   51           1C    AE  3C 0014B          MOVZWL  XAB+20, R1                                      ;
                                   53        FE00 C140  9E  9F 0014F          MOVAB   -512(R1)[R0], XAB_SIZE                          ;
                                   66           6A    AE  B0 00155          MOVW    FAB+54, (RETREC)                                ; 0650
                                 02   A6          53    AE  90 00159          MOVB    FAB+31, 2(RETREC)                               ; 0651
                                 03   A6          73    AE  90 0015E          MOVB    FAB+63, 3(RETREC)                               ; 0652
```

```
                                34   AE  9F 00163          PUSHAB  FAB                                          ; 0655
        00000000G  00               01  FB 00166          CALLS   #1, SYS$CLOSE                                ;
                   69               53  D0 0016D          MOVL    XAB_SIZE, (RETLEN)                           ; 0657
                   0D               52  E8 00170          BLBS    STATUS, 12$                                  ; 0661
                                    01  DD 00173          PUSHL   #1                                           ; 0663
                                    52  DD 00175          PUSHL   STATUS                                       ;
                                    01  DD 00177          PUSHL   #1                                           ;
        00000000V  EF               03  FB 00179          CALLS   #3, CRD_ERROR                                ;
              F4   AB 00000000G     EF  D0 00180 12$:     MOVL    DS$GA_LASTADR, DS$A_SAVED_LASTADR            ; 0666
     50       04   AE 000001FF      8F  C1 00188          ADDL3   #511, CRD_ACTUAL_LENGTH, R0                  ; 0669
                   50 00000200      8F  C6 00191          DIVL2   #512, PAGE_COUNT                             ;
     10 0000FE03   9F               04  E1 00198          BBC     #4, @#^X0000FE03, 13$                        ; 0672
                                    7E  7C 001A0          CLRQ    -(SP)                                        ; 0682
                                F8  AD  9F 001A2          PUSHAB  CRD_IMAGE_LOCATION                           ;
                                    50  DD 001A5          PUSHL   PAGE_COUNT                                   ;
        00000000G  00               04  FB 001A7          CALLS   #4, SYS$EXPREG                               ;
                                    11  11 001AE          BRB     14$                                          ;
                   7E               03  7D 001B0 13$:     MOVQ    #3, -(SP)                                    ; 0692
                                    7E  D4 001B3          CLRL    -(SP)                                        ;
                                F8  AD  9F 001B5          PUSHAB  CRD_IMAGE_LOCATION                           ;
                                    50  DD 001B8          PUSHL   PAGE_COUNT                                   ; 0693
        00000000G  EF               05  FB 001BA          CALLS   #5, CRD$EXPREG                               ;
                   52               50  D0 001C1 14$:     MOVL    R0, STATUS                                   ;
                   53           F8  AD  D0 001C4          MOVL    CRD_IMAGE_LOCATION, R3                       ; 0704
     50       FC   AD               53  C3 001C8          SUBL3   R3, CRD_IMAGE_LOCATION+4, R0                 ;
              EC   AB           01  A0  9E 001CD          MOVAB   1(R0), DS$Q_CRD_IMAGE_DESC                   ;
              F0   AB               53  D0 001D2          MOVL    R3, DS$Q_CRD_IMAGE_DESC+4                    ; 0706
                   27               52  E8 001D6          BLBS    STATUS, 17$                                  ; 0708
        FFFFFFFF   8F               53  D1 001D9          CMPL    R3, #-1                                      ; 0713
                                    11  12 001E0          BNEQ    15$                                          ;
        FFFFFFFF   8F           FC  AD  D1 001E2          CMPL    CRD_IMAGE_LOCATION+4, #-1                    ; 0714
                                    07  12 001EA          BNEQ    15$                                          ;
                                EC  AB  7C 001EC          CLRQ    DS$Q_CRD_IMAGE_DESC                          ; 0717
                                    01  DD 001EF          PUSHL   #1                                           ; 0720
                                    02  11 001F1          BRB     16$                                          ;
                                    7E  D4 001F3 15$:     CLRL    -(SP)                                        ; 0723
                                    52  DD 001F5 16$:     PUSHL   STATUS                                       ;
                                    02  DD 001F7          PUSHL   #2                                           ;
        00000000V  EF               03  FB 001F9          CALLS   #3, CRD_ERROR                                ;
                                    01  DD 00200 17$:     PUSHL   #1                                           ; 0733
                                04  AE  9F 00202          PUSHAB  CRD_FILE_ATTRIBUTES                          ;
                                0C  AE  9F 00205          PUSHAB  CRD_ACTUAL_LENGTH                            ;
                                    53  DD 00208          PUSHL   R3                                           ;
                                14  AE  DD 0020A          PUSHL   CRD_ACTUAL_LENGTH                            ;
                        00000000'   EF  9F 0020D          PUSHAB  T_CRD_IMAGE_DEFAULT                          ;
                        00000000'   EF  9F 00213          PUSHAB  T_CRD_IMAGE_NAME                             ;
        00000000G  9F               07  FB 00219          CALLS   #7, @#DS$LOAD                                ;
                   52               50  D0 00220          MOVL    R0, STATUS                                   ;
                   0D               52  E8 00223          BLBS    STATUS, 18$                                  ; 0735
                                    7E  D4 00226          CLRL    -(SP)                                        ; 0736
                                    52  DD 00228          PUSHL   STATUS                                       ;
                                    03  DD 0022A          PUSHL   #3                                           ;
        00000000V  EF               03  FB 0022C          CALLS   #3, CRD_ERROR                                ;
        00000000G  EF           04  AE  C0 00233 18$:     ADDL2   CRD_ACTUAL_LENGTH, DS$GA_LASTADR             ; 0738
                   03               5A  E8 0023B          BLBS    R10, 19$                                     ; 0741
                                  0097  31 0023E          BRW     20$                                          ;
                                F0  AB  DD 00241 19$:     PUSHL   DS$Q_CRD_IMAGE_DESC+4                        ; 0749
```

```
                                00000000'  EF  9F 00244          PUSHAB  T_TRACE_1                                       ;
                                           01  DD 0024A          PUSHL   #1                                              ;
                                           1A  DD 0024C          PUSHL   #26                                             ;
                          00000000G  9F        04  FB 0024E          CALLS   #4, a#DSX$PRINT                             ;
                                        EC     AB  DD 00255          PUSHL   DS$Q_CRD_IMAGE_DESC                         ; 0750
                                00000000'  EF  9F 00258          PUSHAB  T_TRACE_2                                       ;
                                           01  DD 0025E          PUSHL   #1                                              ;
                                           1A  DD 00260          PUSHL   #26                                             ;
                          00000000G  9F        04  FB 00262          CALLS   #4, a#DSX$PRINT                             ;
                                     7E        63  98 00269          CVTBL   (R3), -(SP)                                 ; 0754
                                     7E 00000000G  EF  98 0026C          CVTBL   DS$AL_DS_DISPATCH_VECTORS, -(SP)        ;
                                00000000'  EF  9F 00273          PUSHAB  T_TRACE_4                                       ;
                                           01  DD 00279          PUSHL   #1                                              ;
                                           1A  DD 0027B          PUSHL   #26                                             ;
                          00000000G  9F        05  FB 0027D          CALLS   #5, a#DSX$PRINT                             ;
                                     7E     01  A3  98 00284          CVTBL   1(R3), -(SP)                                ; 0758
                                     7E 00000000G  EF  98 00288          CVTBL   DS$AL_DS_DISPATCH_VECTORS+1, -(SP)      ;
                                00000000'  EF  9F 0028F          PUSHAB  T_TRACE_5                                       ;
                                           01  DD 00295          PUSHL   #1                                              ;
                                           1A  DD 00297          PUSHL   #26                                             ;
                          00000000G  9F        05  FB 00299          CALLS   #5, a#DSX$PRINT                             ;
                                     7E     02  A3  98 002A0          CVTBL   2(R3), -(SP)                                ; 0762
                                     7E 00000000G  EF  98 002A4          CVTBL   DS$AL_DS_DISPATCH_VECTORS+2, -(SP)      ;
                                00000000'  EF  9F 002AB          PUSHAB  T_TRACE_6                                       ;
                                           01  DD 002B1          PUSHL   #1                                              ;
                                           1A  DD 002B3          PUSHL   #26                                             ;
                          00000000G  9F        05  FB 002B5          CALLS   #5, a#DSX$PRINT                             ;
                                     7E     03  A3  98 002BC          CVTBL   3(R3), -(SP)                                ; 0766
                                     7E 00000000G  EF  98 002C0          CVTBL   DS$AL_DS_DISPATCH_VECTORS+3, -(SP)      ;
                                00000000'  EF  9F 002C7          PUSHAB  T_TRACE_7                                       ;
                                           01  DD 002CD          PUSHL   #1                                              ;
                                           1A  DD 002CF          PUSHL   #26                                             ;
                          00000000G  9F        05  FB 002D1          CALLS   #5, a#DSX$PRINT                             ;
                                           53  DD 002D8 20$:      PUSHL   R3                                              ; 0769
                          00000000V  EF        01  FB 002DA          CALLS   #1, EXCHANGE_DISPATCH_VECTORS               ;
                                               04  002E1          RET                                                     ; 0773

; Routine Size:  738 bytes,    Routine Base:  CODE + 0000

;   0774  1     %SBTTL 'DSR$Unload_CRD Routine'
;   0775  1
;   0776  1     GLOBAL ROUTINE DSR$Unload_CRD (Return_To_Caller) : NOVALUE =
;   0777  1
;   0778  1     !++
;   0779  1     ! Functional Description:
;   0780  1     !
;   0781  1     !       'Unload' the CRD image from memory. This routines does not actually unload the CRD code and data per se,
;   0782  1     !       but it does deallocate the memory that the CRD code and data occupies. This routine also deallocates
;   0783  1     !       all temporary CRD storage that was allocated in the DSR$Load_CRD routine.
;   0784  1     !
;   0785  1     ! Formal Parameters:
;   0786  1     !
;   0787  1     !       Return_To_Caller: If the image is loaded, then return to the caller of this routine when the
;   0788  1     !                         unloading is done. If it is not loaded, return regardless.
;   0789  1     !
;   0790  1     ! Side Effects:
```

```
;    0791  1      !
;    0792  1      !       Too numerous to mention. See the code!
;    0793  1      !
;    0794  1      ! Completion Codes:
;    0795  1      !
;    0796  1      !       None
;    0797  1      !--
;    0798  2      BEGIN                                   ! Routine DSR$Unload_CRD
;    0799  2              BIND
;    0800  2                      CRD_Vectors = (.DS$Q_CRD_Image_Desc [K_Image_Ptr]) : VECTOR [, BYTE, SIGNED];
;    0801  2
;    0802  2              REGISTER
;    0803  2                      Temp_Vector_Storage_Length;
;    0804  2
;    0805  2              MAP
;    0806  2                      DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE, SIGNED];
;    0807  2
;    0808  2              Module_Debug ('DSR$Unload_CRD - Start:');                                        ![07
;    0809  2
;    0810  2              !+
;    0811  2              ! Clear out the three ^C handler addresses. This must be done before going back to the CLI.
;    0812  2              !-
;    0813  2
;    0814  2              DS$GA_DS_Ctrl_C_First  = 0;
;    0815  2              DS$GA_DS_Ctrl_C_Second = 0;
;    0816  2              DS$L_UserCntrlC        = 0;
;    0817  2
;    0818  2              $DS_CntrlC (Disabl = 1);        ! Disable ^C recognition
;    0819  2
;    0820  2              !+
;    0821  2              ! Check first to see if the CRD image is in there. This is checked by looking at the
;    0822  2              ! image descriptor used to point to the image. If this is null, then nothing has been
;    0823  2              ! loaded in yet. Note that the Vector_Desc has been set up already, but the vectors
;    0824  2              ! have not been exchanged yet. So, it doesn't matter that the Vector_Desc isn't null.
;    0825  2              !-
;    0826  2
;    0827  2              IF (.DS$Q_CRD_Image_Desc [K_Image_Size] EQL 0) THEN
;    0828  3                      BEGIN
;    0829  3                      Module_Debug ('DSR$Unload_CRD - Short end:');                            ![07
;    0830  3
;    0831  3                      $DS_CntrlC (Disabl = 0);        ! Enable ^C recognition
;    0832  3                      RETURN;
;    0833  2                      END;
;    0834  2
;    0835  2              !+
;    0836  2              ! If the Saved_LastAdr is not zero, restore the LastAdr address for the VDS.
;    0837  2              !-
;    0838  2
;    0839  2              IF (.DS$A_Saved_LastAdr NEQ 0) THEN
;    0840  3                      BEGIN
;    0841  3                      DS$GA_LastAdr = .DS$A_Saved_LastAdr;
;    0842  3                      DS$A_Saved_LastAdr = 0;
;    0843  2                      END;
;    0844  2
;    0845  2              !+
;    0846  2              ! Now check to see if an internal error occurred while running CRD. If an internal error did happen,
;    0847  2              ! then the Positive_Version will be one less than the absolute value of the Negative_Version - the
```

```
;    0848  2              ! CRD_Internal_Error routine does this.
;    0849  2              !-
;    0850  2
;    0851  3              IF ((.CRD_Vectors [K_Positive_Version] - 1) EQL
;    0852  3                 (- .CRD_Vectors [K_Negative_Version]))
;    0853  2              THEN
;    0854  2                      !+
;    0855  2                      ! Simply call the error routine so that it can print some error messages, and return to here.
;    0856  2                      !-
;    0857  2                      CRD_Error (K_CRD_Internal, 0, 0);
;    0858  2
;    0859  2              !+
;    0860  2              ! Now transfer the dispatch vectors back from the temporary vector storage to where they
;    0861  2              ! belong - in the DS$AL_DS_Dispatch_Vectors area. This enables CRD to be started again.
;    0862  2              !-
;    0863  2
;    0864  2              Temp_Vector_Storage_Length = .DS$Q_CRD_Vector_Desc [DSC$W_Length];
;    0865  2
;    0866  2              IF (.Temp_Vector_Storage_Length NEQ 0) THEN
;    0867  3                      BEGIN
;    0868  3                      REGISTER
;    0869  3                              Number_Of_Vectors;
;    0870  3
;    0871  4                              BEGIN
;    0872  4                              BIND
;    0873  4                                      Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, BYTE];
;    0874  4
;    0875  4                              Number_Of_Vectors = .Temp_Vectors [K_Number_Of_Vectors];
;    0876  3                              END;
;    0877  3
;    0878  3                      INCR Vector_Index FROM 0 TO .Number_Of_Vectors DO
;    0879  4                              BEGIN
;    0880  4                              MAP
;    0881  4                                      DS$AL_DS_Dispatch_Vectors : VECTOR [, LONG];
;    0882  4
;    0883  4                              BIND
;    0884  4                                      Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, LONG];
;    0885  4
;    0886  4                              DS$AL_DS_Dispatch_Vectors [.Vector_Index] = .Temp_Vectors [.Vector_Index];
;    0887  3                              END;
;    0888  2                      END;                    ! IF . . .
;    0889  2
;    0890  2              !+
;    0891  2              ! Now contract the region occupied by the CRD image.
;    0892  2              !-
;    0893  2
;    0894  2              IF (.DS$Q_CRD_Image_Desc [K_Image_Size] NEQ 0) THEN
;    0895  3                      BEGIN
;    0896  3                      REGISTER
;    0897  3                              Status,
;    0898  3                              Page_Count;
;    0899  3
;    0900  3                      LOCAL
;    0901  3                              Temp_Vector : VECTOR [2, LONG];
;    0902  3
;    0903  3                      Page_Count = (.DS$Q_CRD_Image_Desc [K_Image_Size] + 511) / 512;
;    0904  3
```

```
;    0905  5                    IF (NOT (
;    0906  5                            IF (.DSA$V_User) THEN
; P  0907  5                                    Status = $CntReg (
; P  0908  5                                            PagCnt = .Page_Count,
; P  0909  5                                            RetAdr = Temp_Vector,
;    0910  6                                            Region = 0)
;    0911  5                            ELSE
;    0912  5
;    0913  5                                    Status = EXE$CNTREG (                   ![19]
;    0914  5                                                         .Page_Count,
;    0915  5                                                         Temp_Vector,
;    0916  5                                                         0,             ! ACC MODE
;    0917  5                                                         3,             ! REGION
;    0918  5                                                         0);            ! _VA (DUMMY)
;    0919  4                            .Status))
;    0920  3                    THEN
;    0921  4                            BEGIN
;    0922  4                            CRD_Error (K_Not_Contracted, .Status, K_Region_Expanded);
;    0923  4                            END
;    0924  2                    END;
;    0925  2
;    0926  2                !+
;    0927  2                ! Now, clear out the descriptors, in preparation for next time.
;    0928  2                !-
;    0929  2
;    0930  2                DS$Q_CRD_Image_Desc [K_Image_Size] = 0;
;    0931  2                DS$Q_CRD_Image_Desc [K_Image_Ptr]  = 0;
;    0932  2
;    0933  2                DS$Q_CRD_Vector_Desc [DSC$W_Length]  = 0;
;    0934  2                DS$Q_CRD_Vector_Desc [DSC$B_Class]   = 0;
;    0935  2                DS$Q_CRD_Vector_Desc [DSC$B_DType]   = 0;
;    0936  2                DS$Q_CRD_Vector_Desc [DSC$A_Pointer] = 0;
;    0937  2
;    0938  2                INCR Vec_Index FROM 0 TO (CRD$K_Numb_Dispatch_Vectors - 1) DO
;    0939  2                        DS$A_CRD_Vectors [.Vec_Index] = %X'FFFFFFFF';
;    0940  2
;    0941  2                Module_Debug ('DSR$Unload_CRD - Long end:');                    ![07
;    0942  2
;    0943  2                !+
;    0944  2                ! Now resume echoing ^C's.                                      ![06
;    0945  2                !-
;    0946  2
;    0947  2                CRD$V_Ctrl_C_No_Echo = False;                                   ![07
;    0948  2                CRD$V_Flush_Ctrl_C = False;                                     ![07
;    0949  2
;    0950  2                IF (.Return_To_Caller) THEN
;    0951  2                        RETURN                  ! Return to this routine's caller
;    0952  2                ELSE
;    0953  2                        CRD_Exit ();            ! Either go to Begin or Halt
;    0954  1    END;                                        ! Routine DSR$Unload_CRD


                                                        .EXTRN  DS$CNTRLC, SYS$CNTREG

                                                003C 00000      .ENTRY  DSR$UNLOAD_CRD, Save R2,R3,R4,R5        ; 0776
                                    55 00000000V  EF  9E 00002  MOVAB   CRD_ERROR, R5                           ;
```

```
                    54 00000000G  9F  9E 00009          MOVAB   @#DS$CNTRLC, R4                          ;
                    53 00000000'  EF  9E 00010          MOVAB   DS$Q_CRD_IMAGE_DESC, R3                  ;
                    5E            08  C2 00017          SUBL2   #8, SP                                   ;
                    52        04  A3  D0 0001A          MOVL    DS$Q_CRD_IMAGE_DESC+4, R2                ; 0800
                       00000000G  EF  D4 0001E          CLRL    DS$GA_DS_CTRL_C_FIRST                    ; 0814
                       00000000G  EF  D4 00024          CLRL    DS$GA_DS_CTRL_C_SECOND                   ; 0815
                       00000000G  EF  D4 0002A          CLRL    DS$L_USERCNTRLC                          ; 0816
                                  01  DD 00030          PUSHL   #1                                       ; 0818
                                  7E  D4 00032          CLRL    -(SP)                                    ;
                    64            02  FB 00034          CALLS   #2, DS$CNTRLC                            ;
                                  63  D5 00037          TSTL    DS$Q_CRD_IMAGE_DESC                      ; 0827
                                  06  12 00039          BNEQ    1$                                       ;
                                  7E  7C 0003B          CLRQ    -(SP)                                    ; 0831
                    64            02  FB 0003D          CALLS   #2, DS$CNTRLC                            ;
                                      04 00040          RET                                              ;
                    50        08  A3  D0 00041 1$:      MOVL    DS$A_SAVED_LASTADR, R0                   ; 0839
                                  0A  13 00045          BEQL    2$                                       ;
          00000000G EF            50  D0 00047          MOVL    R0, DS$GA_LASTADR                        ; 0841
                            08    A3  D4 0004E          CLRL    DS$A_SAVED_LASTADR                       ; 0842
                    51        01  A2  98 00051 2$:      CVTBL   1(R2), R1                                ; 0851
                                  51  D7 00055          DECL    R1                                       ;
                    50        02  A2  98 00057          CVTBL   2(R2), R0                                ; 0852
                    50            50  CE 0005B          MNEGL   R0, R0                                   ;
                    50            51  D1 0005E          CMPL    R1, R0                                   ;
                                  07  12 00061          BNEQ    3$                                       ;
                                  7E  7C 00063          CLRQ    -(SP)                                    ; 0857
                                  04  DD 00065          PUSHL   #4                                       ;
                    65            03  FB 00067          CALLS   #3, CRD_ERROR                            ;
                    50        F8  A3  3C 0006A 3$:      MOVZWL  DS$Q_CRD_VECTOR_DESC, -                  ; 0864
                                                                TEMP_VECTOR_STORAGE_LENGTH               ;
                                  17  13 0006E          BEQL    6$                                       ; 0866
                    50        FC  B3  9A 00070          MOVZBL  @DS$Q_CRD_VECTOR_DESC+4, NUMBER_OF_VECTORS ; 0875
                    51            01  CE 00074          MNEGL   #1, VECTOR_INDEX                         ; 0878
                                  0A  11 00077          BRB     5$                                       ;
          00000000GEF41       FC B341  D0 00079 4$:      MOVL    @DS$Q_CRD_VECTOR_DESC+4[VECTOR_INDEX], -  ; 0886
                                                                DS$AL_DS_DISPATCH_VECTORS[VECTOR_INDEX]  ;
       F2           51            50  F3 00083 5$:      AOBLEQ  NUMBER_OF_VECTORS, VECTOR_INDEX, 4$      ;
                    50            63  D0 00087 6$:      MOVL    DS$Q_CRD_IMAGE_DESC, R0                  ; 0894
                                  41  13 0008A          BEQL    9$                                       ;
                    50      01FF  C0  9E 0008C          MOVAB   511(R0), R0                              ; 0903
                    50 00000200   8F  C6 00091          DIVL2   #512, PAGE_COUNT                         ;
       10 0000FE03  9F            04  E1 00098          BBC     #4, @#^X0000FE03, 7$                     ; 0906
                                  7E  7C 000A0          CLRQ    -(SP)                                    ; 0910
                            08    AE  9F 000A2          PUSHAB  TEMP_VECTOR                              ;
                                  50  DD 000A5          PUSHL   PAGE_COUNT                               ;
          00000000G 00            04  FB 000A7          CALLS   #4, SYS$CNTREG                           ;
                                  11  11 000AE          BRB     8$                                       ; 0907
                    7E            03  7D 000B0 7$:      MOVQ    #3, -(SP)                                ; 0913
                                  7E  D4 000B3          CLRL    -(SP)                                    ;
                            0C    AE  9F 000B5          PUSHAB  TEMP_VECTOR                              ;
                                  50  DD 000B8          PUSHL   PAGE_COUNT                               ; 0914
          00000000G EF            05  FB 000BA          CALLS   #5, EXE$CNTREG                           ;
                    09            50  E8 000C1 8$:      BLBS    STATUS, 9$                               ; 0919
                                  7E  D4 000C4          CLRL    -(SP)                                    ; 0922
                                  50  DD 000C6          PUSHL   STATUS                                   ;
                                  07  DD 000C8          PUSHL   #7                                       ;
                    65            03  FB 000CA          CALLS   #3, CRD_ERROR                            ;
```

```
                                                63  7C 000CD 9$:     CLRQ    DS$Q_CRD_IMAGE_DESC                                     ; 0930
                                             F8 A3  7C 000CF         CLRQ    DS$Q_CRD_VECTOR_DESC                                    ; 0933
                                                50  D4 000D2         CLRL    VEC_INDEX                                               ; 0938
                                      94 A340       01  CE 000D4 10$:    MNEGL   #1, DS$A_CRD_VECTORS[VEC_INDEX]                 ; 0939
                                   F7       50      0D  F3 000D9         AOBLEQ  #13, VEC_INDEX, 10$                             ;
                                00000000G   EF      06  8A 000DD         BICB2   #6, DS$GL_CRD_FLAGS                             ; 0947
                                            07  04  AC  E8 000E4         BLBS    RETURN_TO_CALLER, 11$                           ; 0950
                                00000000V   EF      00  FB 000E8         CALLS   #0, CRD_EXIT                                    ; 0953
                                                04 000EF 11$:    RET                                                             ; 0954

; Routine Size:  240 bytes,    Routine Base:  CODE + 02E2


;   0955  1     %SBTTL 'DSR$Restore_CRD_Vectors Routine'
;   0956  1
;   0957  1     GLOBAL ROUTINE DSR$Restore_CRD_Vectors : NOVALUE =
;   0958  1
;   0959  1     !++
;   0960  1     ! Functional Description:
;   0961  1     !
;   0962  1     !       Using the vectors stored in the DS$Q_CRD_Vector_Desc, restore the CRD dispatch vectors. These vectors
;   0963  1     !       are originally set up the the DSVECS module. NOTE: This routine SHOULD ONLY be called from the BEGIN0
;   0964  1     !       label in the KERNEL module!!!!!!
;   0965  1     !
;   0966  1     ! Formal Parameters:
;   0967  1     !
;   0968  1     !       None
;   0969  1     !
;   0970  1     ! Side Effects:
;   0971  1     !
;   0972  1     !       Changes the CRD dispatch vectors located at DS$AL_DS_Dispatch_Vectors
;   0973  1     !
;   0974  1     ! Completion Codes:
;   0975  1     !
;   0976  1     !       None
;   0977  1     !--
;   0978  2     BEGIN                                   ! Routine DSR$Restore_CRD_Vectors
;   0979  2             Module_Debug ('DSR$Restore_CRD_Vectors - Start:');                                              ![07
;   0980  2
;   0981  2             !+
;   0982  2             ! Now resume echoing ^C's. Do this regardless of whether or not CRD is in there (just to be sure).      ![06
;   0983  2             !-
;   0984  2
;   0985  2             CRD$V_Ctrl_C_No_Echo = False;                                                                   ![07
;   0986  2             CRD$V_Flush_Ctrl_C = False;                                                                     ![07
;   0987  2
;   0988  2             !+
;   0989  2             ! Now transfer the dispatch vectors back from the vector storage area to where they
;   0990  2             ! belong - in the DS$AL_DS_Dispatch_Vectors area. This enables CRD to be started again.
;   0991  2             ! Do this ONLY if the vector were changed at some previous point. Determine this by
;   0992  2             ! checking the descriptor that points to the vector storage area for the vectors.
;   0993  2             !-
;   0994  2
;   0995  3             IF ((.DS$Q_CRD_Vector_Desc [DSC$W_Length] NEQ 0) AND
;   0996  3                 (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer] NEQ 0))
;   0997  2             THEN
;   0998  3                     BEGIN
```

```
;    0999  3                    !+
;    1000  3                    ! This will ONLY be done iff CRD was previously running and was stopped by a ^P, then a
;    1001  3                    ! S 10000 console command was done. If this sequence of events is done, then we must restore
;    1002  3                    ! the stored dispatch vectors to their proper place in the DS$AL_DS_Dispatch_Vectors area.
;    1003  3                    ! Also, since we could have been executing a diagnostic at the time the ^P was typed, we must
;    1004  3                    ! clear out the Environ longword in the diagnostic header so that the VDS does not think a
;    1005  3                    ! diagnostic is still loaded once the VDS comes up again (after the S 10000).
;    1006  3                    !-
;    1007  3
;    1008  3                    REGISTER
;    1009  3                            Number_Of_Vectors;
;    1010  3
;    1011  3                    BIND
;    1012  3                            Vector_Info = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, BYTE];
;    1013  3
;    1014  3                    Number_Of_Vectors = .Vector_Info [K_Number_Of_Vectors];
;    1015  3
;    1016  3                    INCR Vector_Index FROM 0 TO .Number_Of_Vectors DO
;    1017  4                            BEGIN
;    1018  4                            MAP
;    1019  4                                    DS$AL_DS_Dispatch_Vectors : VECTOR [, LONG];
;    1020  4
;    1021  4                            BIND
;    1022  4                                    Temp_Vectors = (.DS$Q_CRD_Vector_Desc [DSC$A_Pointer]) : VECTOR [, LONG];
;    1023  4
;    1024  4                            DS$AL_DS_Dispatch_Vectors [.Vector_Index] = .Temp_Vectors [.Vector_Index];
;    1025  3                            END;
;    1026  3
;    1027  3                    L$L_Environ = 0;
;    1028  3
;    1029  3                    DS$A_Saved_LastAdr = 0;
;    1030  3
;    1031  3                    DS$Q_CRD_Image_Desc [K_Image_Size] = 0;
;    1032  3                    DS$Q_CRD_Image_Desc [K_Image_Ptr]  = 0;
;    1033  3
;    1034  3                    DS$Q_CRD_Vector_Desc [DSC$W_Length]  = 0;
;    1035  3                    DS$Q_CRD_Vector_Desc [DSC$B_Class]   = 0;
;    1036  3                    DS$Q_CRD_Vector_Desc [DSC$B_DType]   = 0;
;    1037  3                    DS$Q_CRD_Vector_Desc [DSC$A_Pointer] = 0;
;    1038  3
;    1039  3                    !+
;    1040  3                    ! Also set the temporary vector storage area to all -1's. This will allow the VDS to
;    1041  3                    ! catch all 'buggy' references to an address in this area.
;    1042  3                    !-
;    1043  3
;    1044  3                    INCR Vec_Index FROM 0 TO (CRD$K_Total_Numb_Vectors - 1) DO                              ![07
;    1045  3                            DS$A_CRD_Vectors [.Vec_Index] = %X'FFFFFFFF';
;    1046  2                    END;
;    1047  2            Module_Debug ('DSR$Restore_CRD_Vectors - End:');                                                ![07
;    1048  2            RETURN;
;    1049  1    END;                                    ! Routine DSR$Restore_CRD_Vectors



                                        0004 00000          .ENTRY  DSR$RESTORE_CRD_VECTORS, Save R2                ; 0957
```

```
                                52 00000000'  EF  9E 00002          MOVAB   DS$Q_CRD_VECTOR_DESC+4, R2                                ;
                     00000000G  EF              06  8A 00009          BICB2   #6, DS$GL_CRD_FLAGS                                      ; 0986
                                          FC    A2  B5 00010          TSTW    DS$Q_CRD_VECTOR_DESC                                     ; 0995
                                                35  13 00013          BEQL    4$                                                       ;
                                                62  D5 00015          TSTL    DS$Q_CRD_VECTOR_DESC+4                                   ; 0996
                                                31  13 00017          BEQL    4$                                                       ;
                                     50         00  B2  9A 00019          MOVZBL  @DS$Q_CRD_VECTOR_DESC+4, NUMBER_OF_VECTORS        ; 1014
                                     51             01  CE 0001D          MNEGL   #1, VECTOR_INDEX                                 ; 1016
                                                    0A  11 00020          BRB     2$                                               ;
                     00000000GEF41          00 B241  D0 00022 1$:      MOVL    @DS$Q_CRD_VECTOR_DESC+4[VECTOR_INDEX], -         ; 1024
                                                                                  DS$AL_DS_DISPATCH_VECTORS[VECTOR_INDEX]          ;
                F2                  51             50  F3 0002C 2$:      AOBLEQ  NUMBER_OF_VECTORS, VECTOR_INDEX, 1$              ;
                                 00000204          9F  D4 00030          CLRL    @#^X00000204                                     ; 1027
                                       04          A2  D4 00036          CLRL    DS$Q_CRD_IMAGE_DESC                              ; 1031
                                       08          A2  7C 00039          CLRQ    DS$Q_CRD_IMAGE_DESC+4                            ; 1032
                                       FC          A2  7C 0003C          CLRQ    DS$Q_CRD_VECTOR_DESC                             ; 1034
                                                   50  D4 0003F          CLRL    VEC_INDEX                                        ; 1044
                           98 A240                 01  CE 00041 3$:      MNEGL   #1, DS$A_CRD_VECTORS[VEC_INDEX]                  ; 1045
                F7                  50             18  F3 00046          AOBLEQ  #24, VEC_INDEX, 3$                               ;
                                                   04  0004A 4$:      RET                                                      ; 1049

; Routine Size:  75 bytes,    Routine Base:  CODE + 03D2


;   1050  1     %SBTTL 'Exchange_Dispatch_Vectors Routine'
;   1051  1
;   1052  1     GLOBAL ROUTINE Exchange_Dispatch_Vectors (A_CRD_Dispatch_Vectors) : NOVALUE =                                    ![11
;   1053  1
;   1054  1     !++
;   1055  1     ! Functional Description:
;   1056  1     !
;   1057  1     !       Exchange the CRD and DS dispatch vectors. That is, the DS has a set of dispatch vectors that CRD uses to
;   1058  1     !       access certain DS locations (such as the DSA flags, the Control-C handler address, etc.). These are
;   1059  1     !       stored in the CRDVECS module (part of the Resident-DS image). CRD also has a set of dispatch vectors
;   1060  1     !       that the Resident-DS will use to get to certain CRD locations and routines (such as the CRD routine to
;   1061  1     !       call when something is being printed by the DS). These are stored at the beginning of the CRD-Loadable
;   1062  1     !       image (i.e., in the very first Psect). Exchange_Dispatch_Vectors will exchange these two sets of vectors,
;   1063  1     !       thus enabling the DS to get to the CRD routines, and enabling CRD to get to the DS locations and routines.
;   1064  1     !
;   1065  1     !       Before:
;   1066  1     !               CRD_Dispatch_Vectors:           Contains the addresses of the CRD-Loadable routines and locations
;   1067  1     !               DS$AL_DS_Dispatch_Vectors:      Contains the addresses of certain DS locations and routines
;   1068  1     !       After:
;   1069  1     !               CRD_Dispatch_Vectors:           Contains the addresses of certain DS locations and routines
;   1070  1     !               DS$AL_DS_Dispatch_Vectors:      Contains the addresses of the CRD-Loadable routines and locations
;   1071  1     !
;   1072  1     !       Note also that the first longword in each set of vectors will contain a Count byte (the number of
;   1073  1     !       dispatch vectors), a null byte, and two bytes containing the version number of CRD. One of the version
;   1074  1     !       numbers will be positive, and the other one will be negative. The algorithm for verification is as follows:
;   1075  1     !
;   1076  1     !               1. Read in the CRD-Loadable file
;   1077  1     !               2. Set CRD_Dispatch_Vectors = starting address of the now resident CRD image
;   1078  1     !               3. DS$AL_DS_Dispatch_Vectors [0] <24, 8, 0> = 0                                 ! The null byte
;   1079  1     !               4. DS$AL_DS_Dispatch_Vectors [0] <16, 8, 0> =                                   ! The version #'s
;   1080  1     !                  - DS$AL_DS_Dispatch_Vectors [0] <8, 8, 0>
;   1081  1     !               5. CRD_Dispatch_Vectors [0] <24, 8, 0> = 0                                      ! The null byte
;   1082  1     !               6. CRD_Dispatch_Vectors [0] <16, 8, 0> = - CRD_Dispatch_Vectors [0] <8, 8, 0>   ! The version #'s
```

```
;    1083  1      !
;    1084  1      !       If any of the above statements are not true, an appropriate error message is printed and CRD-AUTOTEST
;    1085  1      !       is aborted. This ensures that not only do the version numbers agree, but also that the image read in is
;    1086  1      !       indeed the CRD-Loadable image file.
;    1087  1      !
;    1088  1      !       Note: The CRD-Loadable image will be linked with a /System=%X0000 qualifer, thus making all addresses relati
;    1089  1      !       to zero. The only address that this affects are those in the CRD_Dispatch_Vectors, the ones that are moved t
;    1090  1      !       the Resident-DS area. These CRD routine addresses must be modified by the amount of the starting location of
;    1091  1      !       the CRD image (i.e., the parameter to this routine).
;    1092  1      ! Picture:
;    1093  1      !
;    1094  1      !                           C R D - L O A D A B L E
;    1095  1      !                           -----------------------
;    1096  1      !                   24              16              8       0 - Bit numbers
;    1097  1      !               ---------------------------------------------
;    1098  1      !               |    0    | - Version # | + Version # |  Count  |  :  .CRD_Dispatch_Vectors [0]
;    1099  1      !               |-------------------------------------------|
;    1100  1      !               |           CRD_Dispatch_Vectors [1]        |
;    1101  1      !               |-------------------------------------------|
;    1102  1      !               |           CRD_Dispatch_Vectors [2]        |
;    1103  1      !               |-------------------------------------------|
;    1104  1      !               |                    ...                    |
;    1105  1      !               |-------------------------------------------|
;    1106  1      !               |         CRD_Dispatch_Vectors [.Count]     |
;    1107  1      !               |-------------------------------------------|
;    1108  1      !
;    1109  1      !
;    1110  1      !                           D S - R E S I D E N T
;    1111  1      !                           ---------------------
;    1112  1      !                   24              16              8       0 - Bit numbers
;    1113  1      !               ---------------------------------------------
;    1114  1      !               |    0    | - Version # | + Version # |  Count  |  :  .DS$AL_DS_Dispatch_Vectors [0]
;    1115  1      !               |-------------------------------------------|
;    1116  1      !               |           DS_Vectors [1]                  |
;    1117  1      !               |-------------------------------------------|
;    1118  1      !               |           DS_Vectors [2]                  |
;    1119  1      !               |-------------------------------------------|
;    1120  1      !               |                    ...                    |
;    1121  1      !               |-------------------------------------------|
;    1122  1      !               |         DS_Vectors [.Count]               |
;    1123  1      !               |-------------------------------------------|
;    1124  1      !
;    1125  1      ! Formal Parameters:
;    1126  1      !
;    1127  1      !       A_CRD_Dispatch_Vectors :   The address of the CRD dispatch vectors. These vectors will contain the addresses
;    1128  1      !                                  of the DS routines and locations that CRD will have to access.
;    1129  1      ! Implicit Inputs:
;    1130  1      !
;    1131  1      !       DS$AL_DS_Dispatch_Vectors : The address of the DS dispatch vectors. These vectors will contain the addresses
;    1132  1      !                                   of the CRD routines and locations that the Resident-DS will have to access.
;    1133  1      !
;    1134  1      ! Side Effects:
;    1135  1      !
;    1136  1      !       Exchanges the DS and the CRD dispatch vectors.
;    1137  1      !
;    1138  1      ! Completion Codes:
;    1139  1      !
```

```
;    1140  1      !       None - If the verification process fails, the CRD_Error routine is called.
;    1141  1      !--
;    1142  2      BEGIN                                       ! Routine Exchange_Dispatch_Vectors
;    1143  2              MACRO
; M  1144  2                      Corrupted_Error =
; M  1145  2                              CRD_Error (K_Corrupted, 0, K_Region_Expanded)
;    1146  2                      %,
;    1147  2
; M  1148  2                      Incompatibility_Error =
; M  1149  2                              CRD_Error (K_Incompatible, 0, K_Region_Expanded)
;    1150  2                      %;
;    1151  2
;    1152  2              !+
;    1153  2              ! Use these when referencing the vectors (the longwords).
;    1154  2              !-
;    1155  2
;    1156  2              BIND
;    1157  2                      CRD_Dispatch_Vectors = (.A_CRD_Dispatch_Vectors)   : VECTOR [, LONG],
;    1158  2                      DS_Dispatch_Vectors  = (DS$AL_DS_Dispatch_Vectors) : VECTOR [, LONG];
;    1159  2
;    1160  2              !+
;    1161  2              ! Use these two when referencing the particular bytes in the first longword.
;    1162  2              !-
;    1163  2
;    1164  2              MAP
;    1165  2                      DS$AL_DS_Dispatch_Vectors :     VECTOR [, BYTE, SIGNED],
;    1166  2                      A_CRD_Dispatch_Vectors    : REF VECTOR [, BYTE, SIGNED];
;    1167  2
;    1168  2              !+
;    1169  2              ! Verify that the image file read in is indeed the CRD-Loadable image file.
;    1170  2              !-
;    1171  2
;    1172  2              IF (.A_CRD_Dispatch_Vectors [K_Null_Byte] NEQ 0) THEN
;    1173  2                      Corrupted_Error;
;    1174  2
;    1175  2              IF (.A_CRD_Dispatch_Vectors [K_Positive_Version] NEQ (- .A_CRD_Dispatch_Vectors [K_Negative_Version])) THEN
;    1176  2                      Corrupted_Error;
;    1177  2
;    1178  2              !+
;    1179  2              ! The format of the verification bytes is okay. Now check to see if the version numbers match
;    1180  2              ! (note: the DS version number must be either equal to or greater than the CRD version number).
;    1181  2              ! Also check to see if the vector counts are equal (see explanation on #2 in history above).          ![02
;    1182  2              !-
;    1183  2
;    1184  2              IF (.DS$AL_DS_Dispatch_Vectors [K_Positive_Version] LSS .A_CRD_Dispatch_Vectors [K_Positive_Version]) THEN
;    1185  2                      Incompatibility_Error;
;    1186  2
;    1187  2              IF (.DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors] NEQ .A_CRD_Dispatch_Vectors [K_Number_Of_Vectors]) THEN
;    1188  2                      Incompatibility_Error;                                                                  ![02
;    1189  2
;    1190  2              !+
;    1191  2              ! Exchange the two sets of vectors. Be sure and add in the CRD offset from zero. Since the CRD image
;    1192  2              ! is LINKed with a base of 0, all the .ADDRESSes in CRDVECS are based on a base of 0, rather than the
;    1193  2              ! actual base that is computed at run-time by the Load_CRD routine. So, I must add the value of the
;    1194  2              ! FIRST longword address in the CRD image to the values contained in each of the CRD dispatch vectors,
;    1195  2              ! and then store this value in the DS dispatch vector area.
;    1196  2              !-
```

```
;   1197  2
;   1198  2             INCR Vector FROM 1 TO (.DS$AL_DS_Dispatch_Vectors [K_Number_Of_Vectors]) DO
;   1199  3                     BEGIN
;   1200  3                     REGISTER
;   1201  3                             Exchange;
;   1202  3
;   1203  3                     Exchange = .DS_Dispatch_Vectors [.Vector];
;   1204  3                     DS_Dispatch_Vectors [.Vector] = .CRD_Dispatch_Vectors [.Vector] + CRD_Dispatch_Vectors;
;   1205  3                                         ! Add in offset of CRD from zero
;   1206  3                     CRD_Dispatch_Vectors [.Vector] = .Exchange;
;   1207  2                     END;
;   1208  1     END;                                ! End of Routine Exchange_Dispatch_Vectors



                                          003C 00000          .ENTRY  EXCHANGE_DISPATCH_VECTORS, Save R2,R3,R4,R5 ; 1052
                                55 00000000V  EF  9E 00002         MOVAB   CRD_ERROR, R5                               ;
                                54 00000000G  EF  9E 00009         MOVAB   DS$AL_DS_DISPATCH_VECTORS, R4               ;
                                52        04  AC  D0 00010         MOVL    A_CRD_DISPATCH_VECTORS, R2                  ; 1157
                                          03  A2  95 00014         TSTB    3(R2)                                       ; 1172
                                              07  13 00017         BEQL    1$                                          ;
                                              7E  7C 00019         CLRQ    -(SP)                                       ;
                                              05  DD 0001B         PUSHL   #5                                          ;
                                65            03  FB 0001D         CALLS   #3, CRD_ERROR                               ;
                                50        02  A2  98 00020 1$:     CVTBL   2(R2), R0                                   ; 1175
                                50            50  CE 00024         MNEGL   R0, R0                                      ;
        50        01    A2      08            00  EC 00027         CMPV    #0, #8, 1(R2), R0                           ;
                                              07  13 0002D         BEQL    2$                                          ;
                                              7E  7C 0002F         CLRQ    -(SP)                                       ;
                                              05  DD 00031         PUSHL   #5                                          ;
                                65            03  FB 00033         CALLS   #3, CRD_ERROR                               ;
                          01    A2        01  A4  91 00036 2$:     CMPB    DS$AL_DS_DISPATCH_VECTORS+1, 1(R2)          ; 1184
                                              07  18 0003B         BGEQ    3$                                          ;
                                              7E  7C 0003D         CLRQ    -(SP)                                       ;
                                              06  DD 0003F         PUSHL   #6                                          ;
                                65            03  FB 00041         CALLS   #3, CRD_ERROR                               ;
                                62            64  91 00044 3$:     CMPB    DS$AL_DS_DISPATCH_VECTORS, (R2)             ; 1187
                                              07  13 00047         BEQL    4$                                          ;
                                              7E  7C 00049         CLRQ    -(SP)                                       ;
                                              06  DD 0004B         PUSHL   #6                                          ;
                                65            03  FB 0004D         CALLS   #3, CRD_ERROR                               ;
                                53            64  98 00050 4$:     CVTBL   DS$AL_DS_DISPATCH_VECTORS, R3               ; 1198
                                              51  D4 00053         CLRL    VECTOR                                      ;
                                              0E  11 00055         BRB     6$                                          ;
                                50          6441  D0 00057 5$:     MOVL    DS_DISPATCH_VECTORS[VECTOR], EXCHANGE       ; 1203
                    6441        6241          52  C1 0005B         ADDL3   R2, (R2)[VECTOR], DS_DISPATCH_VECTORS-      ; 1204
                                                                           [VECTOR]                                    ;
                                6241          50  D0 00061         MOVL    EXCHANGE, (R2)[VECTOR]                      ; 1206
                      EE          51          53  F3 00065 6$:     AOBLEQ  R3, VECTOR, 5$                              ;
                                                  04 00069         RET                                                 ; 1208

; Routine Size:  106 bytes,    Routine Base:  CODE + 041D


;   1209  1     %SBTTL 'CRD_Exit Routine'
```

```
;    1210  1
;    1211  1      ROUTINE CRD_Exit : NOVALUE =
;    1212  1
;    1213  1      !++
;    1214  1      ! Functional Description:
;    1215  1      !
;    1216  1      !       Exit from the CRD-world to either the VDS CLI command mode, or to the 11/730,725 console mode.
;    1217  1      !
;    1218  1      ! Formal Parameters:
;    1219  1      !
;    1220  1      !       None
;    1221  1      !
;    1222  1      ! Implicit Inputs:
;    1223  1      !
;    1224  1      !       None
;    1225  1      !
;    1226  1      ! Side Effects:
;    1227  1      !
;    1228  1      !       Either exits the VDS or goes to CLI mode via the Begin_Bliss label.
;    1229  1      !
;    1230  1      ! Completion Codes:
;    1231  1      !
;    1232  1      !       None - does not return from here
;    1233  1      !--
;    1234  2      BEGIN                                   ! Routine CRD_Exit
;    1235  2              BUILTIN
;    1236  2                      HALT;
;    1237  2
;    1238  2              Module_Debug ('CRD_Exit - Start:');                                                    ![07
;    1239  2
;    1240  2              !+
;    1241  2              ! Cancel any ^C's that happened before this routine was called, and re-enable catching
;    1242  2              ! of ^C's.
;    1243  2              !-
;    1244  2
;    1245  2              DS$GL_Flags <DS$V_CtrlC> = Off;
;    1246  2              $DS_CntrlC (Disabl = 0);
;    1247  2
;    1248  2              !+
;    1249  2              ! Now resume echoing ^C's.                                                             ![06
;    1250  2              !-
;    1251  2
;    1252  2              CRD$V_Ctrl_C_No_Echo = False;                                                          ![07
;    1253  2              CRD$V_Flush_Ctrl_C = False;                                                            ![07
;    1254  2
;    1255  2              !+
;    1256  2              ! Restore the DSA and DS flag settings that were saved in DSR$Load_CRD.
;    1257  2              !-
;    1258  2
;    1259  2              DSA$GL_Flags = .DS$L_Saved_DSA_Flags;
;    1260  2              DS$GL_Flags  = .DS$L_Saved_DS_Flags;
;    1261  2
;    1262  2              !+
;    1263  2              ! Make sure all the CRD bits are cleared. Don't touch BINARY though. Perhaps the operator turned
;    1264  2              ! it on on purpose.
;    1265  2              !-
;    1266  2
```

```
;   1267  2          DSA$V_CRD_Trace    = Off;
;   1268  2          DSA$V_CRD_MenuTest_Off = Off;
;   1269  2                                          !                                        [10]
;   1270  2          DSA$V_CRD_MenuTest_On = Off;
;   1271  2                                          !                                        [10]
;   1272  2          DSA$V_CRD_AutoTest_Off = Off;
;   1273  2                                          !                                        [10]
;   1274  2
;   1275  2          DS_Cleanup ();                  ! Cleanup any residual diagnostics
;   1276  2
;   1277  2          !+
;   1278  2          ! Make the VDS think that no diagnogstic is loaded.
;   1279  2          !-
;   1280  2
;   1281  2          DS$GL_Flags <DS$V_LodFlg> = Off;
;   1282  2          L$L_Environ = 0;
;   1283  2
;   1284  2          Module_Debug ('CRD_Exit - End:');                                        ![07
;   1285  2
;   1286  3          IF (DSR$Check_MenuTest_Off_Set () OR
;   1287  3              DSR$Check_AutoTest_Off_Set ()
;   1288  2          ) THEN
;   1289  3                  BEGIN
;   1290  3                  WHILE (True) DO
;   1291  3                          HALT ();
;   1292  3                  END
;   1293  2          ELSE
;   1294  3                  BEGIN
;   1295  3                  !+
;   1296  3                  ! Since this goes back to the DS$Cli routine, the ^C recognition will be re-enabled there.
;   1297  3                  !-
;   1298  3
;   1299  3                  JSB_None (Begin_Bliss);
;   1300  2                  END;
;   1301  1     END;                                 ! Routine CRD_Exit



                                                 0FFC 00000 CRD_EXIT:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 1211
                        52 00000000G  EF  9E 00002          MOVAB   DS$GL_FLAGS, R2                             ;
                        62            01  8A 00009          BICB2   #1, DS$GL_FLAGS                             ; 1245
                                      7E  7C 0000C          CLRQ    -(SP)                                       ; 1246
            00000000G   9F            02  FB 0000E          CALLS   #2, @#DS$CNTRLC                             ;
            00000000G   EF            06  8A 00015          BICB2   #6, DS$GL_CRD_FLAGS                         ; 1252
            0000FE00    9F 00000000'  EF  D0 0001C          MOVL    DS$L_SAVED_DSA_FLAGS, @#^X0000FE00          ; 1259
                        62 00000000'  EF  D0 00027          MOVL    DS$L_SAVED_DS_FLAGS, DS$GL_FLAGS            ; 1260
            0000FE01    9F      0708  8F  AA 0002E          BICW2   #1800, @#^X0000FE01                         ; 1270
                           00000000G  EF  16 00037          JSB     DS_CLEANUP                                  ; 1275
                        62            02  8A 0003D          BICB2   #2, DS$GL_FLAGS                             ; 1281
                           00000204   9F  D4 00040          CLRL    @#^X00000204                                ; 1282
                           00000000G  EF  16 00046          JSB     DSR$CHECK_MENUTEST_OFF_SET                  ; 1286
                        09            50  E8 0004C          BLBS    R0, 1$                                      ;
                           00000000G  EF  16 0004F          JSB     DSR$CHECK_AUTOTEST_OFF_SET                  ; 1287
                        03            50  E9 00055          BLBC    R0, 2$                                      ;
```

```
                                        00 00058 1$:    HALT                                    ; 1291
                                    FD  11 00059        BRB     1$                              ;
                          00000000G EF  16 0005B 2$:    JSB     BEGIN_BLISS                     ; 1299
                                        04 00061        RET                                     ; 1301

; Routine Size:  98 bytes,    Routine Base:  CODE + 0487


;   1302  1     %SBTTL 'CRD_Error Routine'
;   1303  1
;   1304  1     ROUTINE CRD_Error (Error_Type, More_Info, Memory_Was_Allocated) : NOVALUE =
;   1305  1
;   1306  1     !++
;   1307  1     ! Functional Description:
;   1308  1     !
;   1309  1     !       Prints an error message, and optionally (depending on the Error_Type) unloads the CRD image.
;   1310  1     !
;   1311  1     ! Formal Parameters:
;   1312  1     !
;   1313  1     !       Error_Type:           The reason for the error.
;   1314  1     !       More_Info:            More information on the Error_Type (status values, etc.).
;   1315  1     !       Memory_Was_Allocated: Either K_Region_Expanded or K_Region_Not_Expanded.
;   1316  1     !
;   1317  1     ! Implicit Inputs:
;   1318  1     !
;   1319  1     !       DS$A_Auto_Or_Menu
;   1320  1     !
;   1321  1     ! Side Effects:
;   1322  1     !
;   1323  1     !       Prints an error message, and optionally (depending on the Error_Type) Unloads the CRD image.
;   1324  1     !
;   1325  1     ! Completion Codes:
;   1326  1     !
;   1327  1     !       None
;   1328  1     !--
;   1329  2     BEGIN                                   ! Routine CRD_Error
;   1330  2             REGISTER
;   1331  2                     CRD_Vectors     : REF VECTOR [, BYTE, SIGNED],
;   1332  2                     Save_DSA_Flags,
;   1333  2                     Return_Now,
;   1334  2                     Error_Message;
;   1335  2                                                                                         ![07
;   1336  2             Module_Debug ('CRD_Error - Start:');
;   1337  2
;   1338  2             CRD_Vectors = .DS$Q_CRD_Image_Desc [K_Image_Ptr];
;   1339  2
;   1340  2             Save_DSA_Flags = .DSA$GL_Flags; ! Save them locally
;   1341  2
;   1342  2             DSA$V_CRD_MenuTest_Off = Off;
;   1343  2                                             !                                   [10]
;   1344  2             DSA$V_CRD_MenuTest_On = Off;
;   1345  2                                             !                                   [10]
;   1346  2             DSA$V_CRD_AutoTest_Off = Off;
;   1347  2                                             !                                   [10]
;   1348  2             DSA$V_Binary        = Off;
;   1349  2
;   1350  2             Error_Message = 0;              ! Initialize to point to zer0
```

```
;  1351  2              Return_Now    = False;
;  1352  2
;  1353  2              CASE .Error_Type FROM 0 TO K_Last_Error_Type - 1 OF
;  1354  2                      SET
;  1355  2                      [K_MM_Is_On]:
;  1356  2                              Error_Message = T_MM_On;
;  1357  2
;  1358  2                      [K_No_DSA_Bit_Set]:
;  1359  2                              IF (.DSA$V_CRD_Trace) THEN
;  1360  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_No_DSA_Bit_Set, .More_Info)
;  1361  2                              ELSE
;  1362  2                                      Error_Message = T_Unknown_Error;
;  1363  2
;  1364  2                      [K_Determine_Size]:
;  1365  2                              IF (.DSA$V_CRD_Trace) THEN
;  1366  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Determine_Size, .More_Info)
;  1367  2                              ELSE
;  1368  2                                      IF (.More_Info EQL RMS$_FNF) THEN
;  1369  2                                              Error_Message = T_File_Not_Found
;  1370  2                                      ELSE IF (NOT .More_Info) THEN
;  1371  2                                              Error_Message = T_Unknown_Error;
;  1372  2
;  1373  2                      [K_Expand_Region]:
;  1374  2                              IF (.DSA$V_CRD_Trace) THEN
;  1375  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Expand_Region, .More_Info)
;  1376  2                              ELSE
;  1377  2                                      Error_Message = T_Unknown_Error;
;  1378  2
;  1379  2                      [K_Load_File]:
;  1380  2                              IF (.DSA$V_CRD_Trace) THEN
;  1381  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Load_File, .More_Info)
;  1382  2                              ELSE
;  1383  2                                      Error_Message = T_Unknown_Error;
;  1384  2
;  1385  2                      [K_CRD_Internal]:
;  1386  3                              BEGIN
;  1387  3                              !+
;  1388  3                              ! This check must be made to CRD_Trace, not to CRD_Debug, because the CRD_Debug          ![07
;  1389  3                              ! location may not contain the 0 or 1 that is used to indicate debugging mode.          ![07
;  1390  3                              ! It may contain a CRD dispatch vector (i.e., the vectors may still be switched).       ![07
;  1391  3                              !-
;  1392  3                              IF (.CRD$V_CRD_Debug) THEN                                                            ![07
;  1393  3                                      Internal_Error ()                                                             ![07
;  1394  3                              ELSE
;  1395  3                                      Error_Message = T_Corrupted_Error;
;  1396  3
;  1397  3                              Return_Now = True;
;  1398  2                              END;
;  1399  2
;  1400  2                      [K_Corrupted]:
;  1401  2                              IF (.DSA$V_CRD_Trace) THEN
;  1402  3                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Corrupted)
;  1403  2                              ELSE
;  1404  2                                      Error_Message = T_Corrupted_Error;
;  1405  2
;  1406  2                      [K_Incompatible]:
;  1407  2                              IF (.DSA$V_CRD_Trace) THEN
```

```
;   1408  3                                   $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Incompatible)
;   1409  2                           ELSE
;   1410  2                                   Error_Message = T_Incompatibility_Error;
;   1411  2
;   1412  2                       [K_Not_Contracted]:
;   1413  3                           BEGIN
;   1414  3                           !+
;   1415  3                           ! If the contraction failed, then we must print an error message, call the
;   1416  3                           ! MapFree routine to map all the free areas of memory, and then exit CRD.
;   1417  3                           !-
;   1418  3
;   1419  3                           IF (.DSA$V_CRD_Trace) THEN
;   1420  4                                   $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Not_Contracted, .More_Info)
;   1421  3                           ELSE
;   1422  3                                   $Print (DS$K_Type_General_Error, DS$K_PrintF, T_Contraction_Error);
;   1423  3
;   1424  3                           MapFree ();     ! Do this to Contract the region
;   1425  3                           CRD_Exit ();    ! Exit from here
;   1426  2                           END;
;   1427  2                       TES;
;   1428  2
;   1429  2               IF (.Error_Message NEQ 0) THEN
;   1430  2                       $Print (DS$K_Type_General_Error, DS$K_PrintF, .Error_Message, .DS$A_Auto_Or_Menu);
;   1431  2
;   1432  2               IF (.Return_Now) THEN
;   1433  3                       BEGIN
;   1434  3                       DSA$GL_Flags = .Save_DSA_Flags;
;   1435  3                       RETURN;                     ! All we had to do was to print an error message
;   1436  2                       END;
;   1437  2
;   1438  2               !+
;   1439  2               ! If it was Auto Test that was running (the RPB bit must be set if Auto Test was running),
;   1440  2               ! then print the final three lines of output.
;   1441  2               !-
;   1442  2
;   1443  2               IF (DSR$Check_AutoTest_Off_Set ()) THEN
;   1444  2                                           !                                                   [10]
;   1445  3                       BEGIN
;   1446  3                       $Print (DS$K_Type_General_Error, DS$K_PrintF, T_AUTO_Abort_CRD);
;   1447  3                       $Print (DS$K_Type_General_Error, DS$K_PrintF, T_AUTO_End_Message);
;   1448  3                       $Print (DS$K_Type_General_Error, DS$K_PrintF, T_AUTO_Exit_To_Console);
;   1449  2                       END;
;   1450  2
;   1451  2               IF (.Memory_Was_Allocated) THEN
;   1452  2                       DSR$Unload_CRD (0);     ! This should not return to here
;   1453  2
;   1454  2               DSA$GL_Flags = .Save_DSA_Flags; ! Restore the (locally saved) DSA flag settings
;   1455  2
;   1456  2               Module_Debug ('CRD_Error - End:');                                              ![07
;   1457  2
;   1458  2               CRD_Exit ();                    ! Exit CRD
;   1459  2               RETURN;
;   1460  1     END;                                      ! Routine CRD_Error
```

```
                                                  0FFC 00000 CRD_ERROR:
                                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1304
                                 57        99   AF  9E 00002            MOVAB   CRD_EXIT, R7                                  ;
                                 56 00000000G   9F  9E 00006            MOVAB   @#DSX$PRINT, R6                               ;
                                 55 00000000'   EF  9E 0000D            MOVAB   T_CORRUPTED_ERROR, R5                         ;
                                 50 00000000'   EF  D0 00014            MOVL    DS$Q_CRD_IMAGE_DESC+4, CRD_VECTORS            ; 1338
                                 52 0000FE00    9F  D0 0001B            MOVL    @#^X0000FE00, SAVE_DSA_FLAGS                  ; 1340
                      0000FE01   9F      0508   8F  AA 00022            BICW2   #1288, @#^X0000FE01                           ; 1344
                      0000FE00   9F             02  8A 0002B            BICB2   #2, @#^X0000FE00                              ; 1348
                                                53  7C 00032            CLRQ    RETURN_NOW                                    ; 1351
                   08            00        04   AC  CF 00034            CASEL   ERROR_TYPE, #0, #8                            ; 1353
   0062          0051          0029          0018     00039 1$:         .WORD   3$-1$,-                                       ;
   00C7          00AC          0099          0081     00041                     4$-1$,-                                       ;
                                             0012     00049                     7$-1$,-                                       ;
                                                                                8$-1$,-                                       ;
                                                                                11$-1$,-                                      ;
                                                                                15$-1$,-                                      ;
                                                                                17$-1$,-                                      ;
                                                                                21$-1$,-                                      ;
                                                                                2$-1$                                         ;
                                 54        31   A5  9E 0004B 2$:         MOVAB   T_MM_ON, ERROR_MESSAGE                        ; 1356
                                                7F  11 0004F            BRB     14$                                           ;
                   5A 0000FE02   9F             01  E1 00051 3$:         BBC     #1, @#^X0000FE02, 10$                         ; 1359
                                          08   AC  DD 00059            PUSHL   MORE_INFO                                     ; 1360
                                        013E   C5  9F 0005C            PUSHAB  T_NO_DSA_BIT_SET                              ;
                                                48  11 00060            BRB     9$                                            ;
                   09 0000FE02   9F             01  E1 00062 4$:         BBC     #1, @#^X0000FE02, 5$                          ; 1365
                                          08   AC  DD 0006A            PUSHL   MORE_INFO                                     ; 1366
                                        016E   C5  9F 0006D            PUSHAB  T_DETERMINE_SIZE                              ;
                                                37  11 00071            BRB     9$                                            ;
                      00018292   8F        08   AC  D1 00073 5$:         CMPL    MORE_INFO, #98962                             ; 1368
                                                07  12 0007B            BNEQ    6$                                            ;
                                 54      FF37   C5  9E 0007D            MOVAB   T_FILE_NOT_FOUND, ERROR_MESSAGE               ; 1369
                                                7A  11 00082            BRB     20$                                           ;
                                 76        08   AC  E8 00084 6$:         BLBS    MORE_INFO, 20$                                ; 1370
                                                29  11 00088            BRB     10$                                           ; 1371
                   21 0000FE02   9F             01  E1 0008A 7$:         BBC     #1, @#^X0000FE02, 10$                         ; 1374
                                          08   AC  DD 00092            PUSHL   MORE_INFO                                     ; 1375
                                        01B5   C5  9F 00095            PUSHAB  T_EXPAND_REGION                               ;
                                                0F  11 00099            BRB     9$                                            ;
                   10 0000FE02   9F             01  E1 0009B 8$:         BBC     #1, @#^X0000FE02, 10$                         ; 1380
                                          08   AC  DD 000A3            PUSHL   MORE_INFO                                     ; 1381
                                        01EE   C5  9F 000A6            PUSHAB  T_LOAD_FILE                                   ;
                                                01  DD 000AA 9$:         PUSHL   #1                                            ;
                                                03  DD 000AC            PUSHL   #3                                            ;
                                 66             04  FB 000AE            CALLS   #4, DSX$PRINT                                 ;
                                                79  11 000B1            BRB     24$                                           ;
                                 54      FF76   C5  9E 000B3 10$:        MOVAB   T_UNKNOWN_ERROR, ERROR_MESSAGE                ; 1383
                                                72  11 000B8            BRB     24$                                           ; 1380
                                 09 00000000G   EF  E9 000BA 11$:        BLBC    DS$GL_CRD_FLAGS, 12$                          ; 1392
                      00000000V  EF             00  FB 000C1            CALLS   #0, INTERNAL_ERROR                            ; 1393
                                                03  11 000C8            BRB     13$                                           ;
                                 54             65  9E 000CA 12$:        MOVAB   T_CORRUPTED_ERROR, ERROR_MESSAGE              ; 1395
                                 53             01  D0 000CD 13$:        MOVL    #1, RETURN_NOW                                ; 1397
                                                5A  11 000D0 14$:        BRB     24$                                           ;
                   06 0000FE02   9F             01  E1 000D2 15$:        BBC     #1, @#^X0000FE02, 16$                         ; 1401
```

```
                                  02EA   C5  9F 000DA          PUSHAB  T_CORRUPTED                                  ; 1402
                                         11  11 000DE          BRB     18$                                          ;
                            54           65  9E 000E0 16$:     MOVAB   T_CORRUPTED_ERROR, ERROR_MESSAGE             ; 1404
                                         47  11 000E3          BRB     24$                                          ; 1401
                 0D 0000FE02  9F         01  E1 000E5 17$:     BBC     #1, a#^X0000FE02, 19$                        ; 1407
                                  0333   C5  9F 000ED          PUSHAB  T_INCOMPATIBLE                               ; 1408
                                         01  DD 000F1 18$:     PUSHL   #1                                           ;
                                         03  DD 000F3          PUSHL   #3                                           ;
                            66           03  FB 000F5          CALLS   #3, DSX$PRINT                                ;
                                         32  11 000F8          BRB     24$                                          ;
                            54     DF    A5  9E 000FA 19$:     MOVAB   T_INCOMPATIBILITY_ERROR, ERROR_MESSAGE       ; 1410
                                         2C  11 000FE 20$:     BRB     24$                                          ; 1407
                 10 0000FE02  9F         01  E1 00100 21$:     BBC     #1, a#^X0000FE02, 22$                        ; 1419
                                    08   AC  DD 00108          PUSHL   MORE_INFO                                    ; 1420
                                  02B4   C5  9F 0010B          PUSHAB  T_NOT_CONTRACTED                             ;
                                         01  DD 0010F          PUSHL   #1                                           ;
                                         03  DD 00111          PUSHL   #3                                           ;
                            66           04  FB 00113          CALLS   #4, DSX$PRINT                                ;
                                         0A  11 00116          BRB     23$                                          ;
                                    B4   A5  9F 00118 22$:     PUSHAB  T_CONTRACTION_ERROR                          ; 1422
                                         01  DD 0011B          PUSHL   #1                                           ;
                                         03  DD 0011D          PUSHL   #3                                           ;
                            66           03  FB 0011F          CALLS   #3, DSX$PRINT                                ;
                     00000000G  EF       00  FB 00122 23$:     CALLS   #0, MAPFREE                                  ; 1424
                            67           00  FB 00129          CALLS   #0, CRD_EXIT                                 ; 1425
                                         54  D5 0012C 24$:     TSTL    ERROR_MESSAGE                                ; 1429
                                         0F  13 0012E          BEQL    25$                                          ;
                          00000000'      EF  DD 00130          PUSHL   DS$A_AUTO_OR_MENU                            ; 1430
                                         54  DD 00136          PUSHL   ERROR_MESSAGE                                ;
                                         01  DD 00138          PUSHL   #1                                           ;
                                         03  DD 0013A          PUSHL   #3                                           ;
                            66           04  FB 0013C          CALLS   #4, DSX$PRINT                                ;
                            08           53  E9 0013F 25$:     BLBC    RETURN_NOW, 26$                              ; 1432
                     0000FE00  9F        52  D0 00142          MOVL    SAVE_DSA_FLAGS, a#^X0000FE00                 ; 1434
                                             04 00149          RET                                                  ;
                          00000000G      EF  16 0014A 26$:     JSB     DSR$CHECK_AUTOTEST_OFF_SET                   ; 1443
                            21           50  E9 00150          BLBC    R0, 27$                                      ;
                                  00B8   C5  9F 00153          PUSHAB  T_AUTO_ABORT_CRD                             ; 1446
                                         01  DD 00157          PUSHL   #1                                           ;
                                         03  DD 00159          PUSHL   #3                                           ;
                            66           03  FB 0015B          CALLS   #3, DSX$PRINT                                ;
                                  00E3   C5  9F 0015E          PUSHAB  T_AUTO_END_MESSAGE                           ; 1447
                                         01  DD 00162          PUSHL   #1                                           ;
                                         03  DD 00164          PUSHL   #3                                           ;
                            66           03  FB 00166          CALLS   #3, DSX$PRINT                                ;
                                  0108   C5  9F 00169          PUSHAB  T_AUTO_EXIT_TO_CONSOLE                       ; 1448
                                         01  DD 0016D          PUSHL   #1                                           ;
                                         03  DD 0016F          PUSHL   #3                                           ;
                            66           03  FB 00171          CALLS   #3, DSX$PRINT                                ;
                            07     0C    AC  E9 00174 27$:     BLBC    MEMORY_WAS_ALLOCATED, 28$                    ; 1451
                                         7E  D4 00178          CLRL    -(SP)                                        ; 1452
                       FE5B  C7          01  FB 0017A          CALLS   #1, DSR$UNLOAD_CRD                           ;
                     0000FE00  9F        52  D0 0017F 28$:     MOVL    SAVE_DSA_FLAGS, a#^X0000FE00                 ; 1454
                            67           00  FB 00186          CALLS   #0, CRD_EXIT                                 ; 1458
                                             04 00189          RET                                                  ; 1460

; Routine Size:  394 bytes,    Routine Base:  CODE + 04E9
```

```
;    1461  1      %SBTTL  'Internal_Error Routine'
;    1462  1
;    1463  1      ROUTINE Internal_Error : NOVALUE =                                                    ![07
;    1464  1      !++                                                                                   ! V
;    1465  1      ! Functional Description:
;    1466  1      !
;    1467  1      !       An internal error has occurred. The operator has already been notified. If the Trace or the Debug bits
;    1468  1      !       are set, print this 'error dump' that shows the state of CRD at the time of the internal error. This will
;    1469  1      !       help the Diagnostic Tools group people tremendously in debugging CRD. Hopefully.
;    1470  1      !
;    1471  1      ! Formal Parameters:
;    1472  1      !
;    1473  1      !       None
;    1474  1      !
;    1475  1      ! Implicit Inputs:
;    1476  1      !
;    1477  1      !       DS$Q_CRD_Image_Desc: This points to the start of the CRD image (in memory). The dispatch vectors (or, in
;    1478  1      !                            this case, the debug vectors) are located at the start of the image.
;    1479  1      !
;    1480  1      ! Side Effects:
;    1481  1      !
;    1482  1      !       Prints some debugging information.
;    1483  1      !
;    1484  1      ! Completion Codes:
;    1485  1      !
;    1486  1      !       None
;    1487  1      !--
;    1488  2      BEGIN
;    1489  2              MAP
;    1490  2                      DS$AL_DS_Dispatch_Vectors : VECTOR [, BYTE];
;    1491  2
;    1492  2              REGISTER
;    1493  2                      State;
;    1494  2
;    1495  2              BIND
;    1496  2                      Debug_Vectors = (.DS$Q_CRD_Image_Desc [K_Image_Ptr] + 4) : VECTOR [, LONG];
;    1497  2
;    1498  2              BIND
;    1499  2                      Error_Code              = Debug_Vectors [1],
;    1500  2                      TypeCode                = Debug_Vectors [2],
;    1501  2                      Length                  = Debug_Vectors [3],
;    1502  2                      Address                 = Debug_Vectors [4],
;    1503  2                      Current_State_Table     = Debug_Vectors [5],
;    1504  2                      MM_Current_State        = Debug_Vectors [6],
;    1505  2                      TQ_Current_State        = Debug_Vectors [7],
;    1506  2                      HS_Current_State        = Debug_Vectors [8],
;    1507  2                      Previous_State          = Debug_Vectors [9],
;    1508  2                      DSA_Flags               = Debug_Vectors [10],
;    1509  2                      DS_Flags                = Debug_Vectors [11],
;    1510  2                      Current_DDB_Ptr         = Debug_Vectors [12],
;    1511  2                      Current_HSB_Ptr         = Debug_Vectors [13],
;    1512  2                      Current_HSB_Numb        = Debug_Vectors [14],
;    1513  2                      Current_TQ_Entry        = Debug_Vectors [15],
;    1514  2                      Start_Of_CRD_Image      = Debug_Vectors [16],
;    1515  2                      CRD_Version_Number      = ((.Debug_Vectors [16]) <8, 8, 0>),
```

```
;   1516  2                       VDS_Version_Number      = DS$AL_DS_Dispatch_Vectors [1];
;   1517  2
;   1518  2               MACRO
; M 1519  2                       Error_Type (Error_Type_Name) =
; M 1520  2                               [- %NAME ('CRD$K_', Error_Type_Name)]:
; M 1521  2                                       $Print (DS$K_Type_General_Error, DS$K_PrintF,
; M 1522  2                                               $ASCIC ('Error Type = ', Error_Type_Name, '!/'))
;   1523  2                       %;
;   1524  2
;   1525  2               Module_Debug ('Internal_Error - Start:');
;   1526  2
; P 1527  2               $Print (DS$K_Type_General_Error, DS$K_PrintF,
;   1528  2                       $ASCIC ('!/!/************************ CRD Internal Error ************************!/!/'));
;   1529  2
;   1530  2               ! Print the type of error
;   1531  2
;   1532  2               CASE -.Error_Code FROM 1 TO (-CRD$K_Last_Internal_Error - 1) OF
;   1533  2                       SET
;   1534  2                       Error_Type ('Allocation_Error');
;   1535  2                       Error_Type ('Action_Range_Error');
;   1536  2                       Error_Type ('Action_EQL_Do_Nothing');
;   1537  2                       Error_Type ('Bad_Typecode_Error');
;   1538  2                       Error_Type ('Data_Length_Error');
;   1539  2                       Error_Type ('MM_Internal_Error');
;   1540  2                       Error_Type ('TQ_Internal_Error');
;   1541  2                       Error_Type ('HS_Internal_Error');
;   1542  2                       Error_Type ('Bad_Action_Number');
;   1543  2                       Error_Type ('Load_Sup_File');
;   1544  2                       Error_Type ('Create_HST');
;   1545  2                       [INRANGE]:
; P 1546  2                               $Print (DS$K_Type_General_Error, DS$K_PrintF,
;   1547  2                                       $ASCIC ('Error Type = unknown (!SL)!/'), .Error_Code);
;   1548  2
;   1549  2                       [OUTRANGE]:
; P 1550  2                               $Print (DS$K_Type_General_Error, DS$K_PrintF,
;   1551  2                                       $ASCIC ('Error Type = out of range (!SL)!/'), .Error_Code);
;   1552  2                       TES;
;   1553  2
;   1554  2               ! Print the typecode name
;   1555  2
;   1556  2               $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Typecode !2* = '));
;   1557  2               DSR$Print_TypeCode_Name (.TypeCode);
;   1558  2
;   1559  2               ! Print the Length of the message being printed at the time the internal occurred. Note that this will
;   1560  2               ! not always be a length, as in the internal errors in the MENTSTINI module.
;   1561  2
;   1562  2               $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Length !4* = !UL!/'), .Length);
;   1563  2
;   1564  2               ! Print the address - see previous comment.
;   1565  2
;   1566  2               $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Address !3* = !XL(X)!/'), .Address);
;   1567  2
;   1568  2               ! Print the DSA and the DS flags.
;   1569  2
;   1570  2               $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('DSA flags !1* = !XL(X)!/'), .DSA_Flags);
;   1571  2               $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('DS flags !2* = !XL(X)!/'), .DS_Flags);
;   1572  2
```

```
;    1573  2              ! Print the state table information
;    1574  2
;    1575  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('!/State Table !10* State Number'));
;    1576  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('!/---------- !10* ------------!/'));
;    1577  2
;    1578  2              State = .Current_State_Table;   ! Start at the current state table
;    1579  2              DO
;    1580  3                      BEGIN
;    1581  3                      CASE .State FROM MEN$K_MM_State_Table TO MEN$K_HS_State_Table OF
;    1582  3                              SET
;    1583  3                              [MEN$K_MM_State_Table]:
;    1584  4                                      BEGIN
; P  1585  4                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Main Menu !17* !2UL!/'),
;    1586  4                                              .MM_Current_State);
;    1587  3                                      END;
;    1588  3
;    1589  3                              [MEN$K_TQ_State_Table]:
;    1590  4                                      BEGIN
; P  1591  4                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Test Queue !16* !2UL!/'),
;    1592  4                                              .TQ_Current_State);
;    1593  3                                      END;
;    1594  3
;    1595  3                              [MEN$K_HS_State_Table]:
;    1596  4                                      BEGIN
; P  1597  4                                      $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Hardware Support !10* !2UL!/'
;    1598  4                                              .HS_Current_State);
;    1599  3                                      END;
;    1600  3                              TES;
;    1601  3
;    1602  3                              State = (.State + 1) MOD 3;
;    1603  3                      END                     ! DO . . . UNTIL
;    1604  2              UNTIL (.State EQL .Current_State_Table);
;    1605  2
;    1606  2              ! Print the previous state number
;    1607  2
;    1608  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Previous state !12* !2UL!/!/'), .Previous_State);
;    1609  2
;    1610  2              ! Print the DDB and HSB (Device Data Block and Hardware Support Block) pointers
;    1611  2
;    1612  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current TQ entry    = !UL!/'),    .Current_TQ_Entry);
;    1613  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current DDB address = !XL(X)!/'), .Current_DDB_Ptr);
;    1614  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current HSB number  = !UL!/'),    .Current_HSB_Numb);
;    1615  2              $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Current HSB address = !XL(X)!/!/'), .Current_HSB_Ptr)
;    1616  2
;    1617  2              ! Now, print the remaining debug vectors. The remaining ones are either 0 or contain a PC. Print only
;    1618  2              ! those that are PC's.
;    1619  2
; P  1620  2              $Print (DS$K_Type_General_Error, DS$K_PrintF,
;    1621  2                      $ASCIC ('PCs from the call frame stack (starting with most recent):!/'));
;    1622  2
;    1623  2              INCR Vector FROM 17 to (CRD$K_Total_Numb_Vectors - 1) DO
;    1624  2                      IF (.Debug_Vectors [.Vector] NEQ 0) THEN
; P  1625  2                              $Print (DS$K_Type_General_Error, DS$K_PrintF,
;    1626  3                                      $ASCIC ('  PC from call frame [!UL] = !XL!/'), (.Vector-17), .Debug_Vectors [.Vector
;    1627  2                      ELSE
;    1628  2                              EXITLOOP;
;    1629  2
```

```
;   1630  2          Module_Debug ('Internal_Error - End:');                                  ! ^
;   1631  1      END;                                ! End of Internal_Error                  ![07



                                                    .PSECT  DATA,NOWRT,NOEXE,  SHR,2

2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2F 21 2F 21 4C 00656 P.ABI:   .ASCII  \L!/!/********************** CRD Intern\  ;
20 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 00665                                                      ;
                  6E 72 65 74 6E 49 20 44 52 43 00674                                                   ;
2A 2A 2A 2A 2A 2A 20 72 6F 72 72 45 20 6C 61 0067E          .ASCII  \al Error **********************!/!/\    ;
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 0068D                                                      ;
                        2F 21 2F 21 2A 2A 2A 0069C                                                      ;
41 20 3D 20 65 70 79 54 20 72 6F 72 72 45 1F 006A3 P.ABJ:   .ASCII  <31>\Error Type = Allocation_Error!/\  ;
72 6F 72 72 45 5F 6E 6F 69 74 61 63 6F 6C 6C 006B2                                                      ;
                                    2F 21 006C1                                                         ;
41 20 3D 20 65 70 79 54 20 72 6F 72 72 45 21 006C3 P.ABK:   .ASCII  \!Error Type = Action_Range_Error!/\  ;
72 72 45 5F 65 67 6E 61 52 5F 6E 6F 69 74 63 006D2                                                      ;
                              2F 21 72 6F 006E1                                                         ;
41 20 3D 20 65 70 79 54 20 72 6F 72 72 45 24 006E5 P.ABL:   .ASCII  \$Error Type = Action_EQL_Do_Nothing!/\  ;
6F 4E 5F 6F 44 5F 4C 51 45 5F 6E 6F 69 74 63 006F4                                                      ;
                        2F 21 67 6E 69 68 74 00703                                                      ;
42 20 3D 20 65 70 79 54 20 72 6F 72 72 45 21 0070A P.ABM:   .ASCII  \!Error Type = Bad_Typecode_Error!/\  ;
72 72 45 5F 65 64 6F 63 65 70 79 54 5F 64 61 00719                                                      ;
                              2F 21 72 6F 00728                                                         ;
44 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 0072C P.ABN:   .ASCII  \ Error Type = Data_Length_Error!/\   ;
6F 72 72 45 5F 68 74 67 6E 65 4C 5F 61 74 61 0073B                                                      ;
                                    2F 21 72 0074A                                                      ;
4D 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 0074D P.ABO:   .ASCII  \ Error Type = MM_Internal_Error!/\   ;
6F 72 72 45 5F 6C 61 6E 72 65 74 6E 49 5F 4D 0075C                                                      ;
                              2F 21 72 0076B                                                            ;
54 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 0076E P.ABP:   .ASCII  \ Error Type = TQ_Internal_Error!/\   ;
6F 72 72 45 5F 6C 61 6E 72 65 74 6E 49 5F 51 0077D                                                      ;
                              2F 21 72 0078C                                                            ;
48 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 0078F P.ABQ:   .ASCII  \ Error Type = HS_Internal_Error!/\   ;
6F 72 72 45 5F 6C 61 6E 72 65 74 6E 49 5F 53 0079E                                                      ;
                              2F 21 72 007AD                                                            ;
42 20 3D 20 65 70 79 54 20 72 6F 72 72 45 20 007B0 P.ABR:   .ASCII  \ Error Type = Bad_Action_Number!/\   ;
65 62 6D 75 4E 5F 6E 6F 69 74 63 41 5F 64 61 007BF                                                      ;
                              2F 21 72 007CE                                                            ;
4C 20 3D 20 65 70 79 54 20 72 6F 72 72 45 1C 007D1 P.ABS:   .ASCII  <28>\Error Type = Load_Sup_File!/\   ;
   2F 21 65 6C 69 46 5F 70 75 53 5F 64 61 6F 007E0                                                      ;
43 20 3D 20 65 70 79 54 20 72 6F 72 72 45 19 007EE P.ABT:   .ASCII  <25>\Error Type = Create_HST!/\      ;
         2F 21 54 53 48 5F 65 74 61 65 72 007FD                                                         ;
75 20 3D 20 65 70 79 54 20 72 6F 72 72 45 1C 00808 P.ABU:   .ASCII  <28>\Error Type = unkno  (!SL)!/\   ;
   2F 21 29 4C 53 21 28 20 6E 77 6F 6E 6B 6E 00817                                                      ;
6F 20 3D 20 65 70 79 54 20 72 6F 72 72 45 21 00825 P.ABV:   .ASCII  \!Error Type = out of range (!SL)!/\  ;
53 21 28 20 65 67 6E 61 72 20 66 6F 20 74 75 00834                                                      ;
                              2F 21 29 4C 00843                                                         ;
3D 20 2A 32 21 20 65 64 6F 63 65 70 79 54 0F 00847 P.ABW:   .ASCII  <15>\Typecode !2* = \               ;
                                          20 00856                                                      ;
21 20 3D 20 2A 34 21 20 68 74 67 6E 65 4C 12 00857 P.ABX:   .ASCII  <18>\Length !4* = !UL!/\           ;
                              2F 21 4C 55 00866                                                         ;
20 3D 20 2A 33 21 20 73 73 65 72 64 64 41 16 0086A P.ABY:   .ASCII  <22>\Address !3* = !XL(X)!/\       ;
         2F 21 29 58 28 4C 58 21 00879                                                                  ;
20 2A 31 21 20 73 67 61 6C 66 20 41 53 44 18 00881 P.ABZ:   .ASCII  <24>\DSA flags !1* = !XL(X)!/\     ;
      2F 21 29 58 28 4C 58 21 20 3D 00890                                                               ;
```

```
3D 20 2A 32 21 20 73 67 61 6C 66 20 53 44 17 0089A P.ACA:  .ASCII  <23>\DS flags !2* = !XL(X)!/\                ;
                  2F 21 29 58 28 4C 58 21 20 008A9                                                            ;
20 65 6C 62 61 54 20 65 74 61 74 53 2F 21 1F 008B2 P.ACB:  .ASCII  <31>\!/State Table !10* State Number\        ;
62 6D 75 4E 20 65 74 61 74 53 20 2A 30 31 21 008C1                                                            ;
                                          72 65 008D0                                                         ;
20 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2F 21 21 008D2 P.ACC:  .ASCII  \!!/---------- !10* ----------!/\          ;
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 20 2A 30 31 21 008E1                                                            ;
                                    2F 21 2D 2D 008F0                                                         ;
2A 37 31 21 20 75 6E 65 4D 20 6E 69 61 4D 15 008F4 P.ACD:  .ASCII  <21>\Main Menu !17* !2UL!/\                ;
                        2F 21 4C 55 32 21 20 00903                                                            ;
36 31 21 20 65 75 65 75 51 20 74 73 65 54 16 0090A P.ACE:  .ASCII  <22>\Test Queue !16* !2UL!/\                ;
                        2F 21 4C 55 32 21 20 2A 00919                                                         ;
6F 70 70 75 53 20 65 72 61 77 64 72 61 48 1C 00921 P.ACF:  .ASCII  <28>\Hardware Support !10* !2UL!/\         ;
   2F 21 4C 55 32 21 20 2A 30 31 21 20 74 72 00930                                                            ;
65 74 61 74 73 20 73 75 6F 69 76 65 72 50 1C 0093E P.ACG:  .ASCII  <28>\Previous state !12* !2UL!/!/\         ;
   2F 21 2F 21 4C 55 32 21 20 2A 32 31 21 20 0094D                                                            ;
74 6E 65 20 51 54 20 74 6E 65 72 72 75 43 1B 0095B P.ACH:  .ASCII  <27>\Current TQ entry    = !UL!/\          ;
      2F 21 4C 55 21 20 3D 20 20 20 20 79 72 0096A                                                            ;
64 61 20 42 44 44 20 74 6E 65 72 72 75 43 1E 00977 P.ACI:  .ASCII  <30>\Current DDB address = !XL(X)!/\        ;
21 29 58 28 4C 58 21 20 3D 20 73 73 65 72 64 00986                                                            ;
                                          2F 00995                                                            ;
75 6E 20 42 53 48 20 74 6E 65 72 72 75 43 1B 00996 P.ACJ:  .ASCII  <27>\Current HSB number  = !UL!/\          ;
   2F 21 4C 55 21 20 3D 20 20 72 65 62 6D 009A5                                                               ;
64 61 20 42 53 48 20 74 6E 65 72 72 75 43 20 009B2 P.ACK:  .ASCII  \ Current HSB address = !XL(X)!/!/\        ;
21 29 58 28 4C 58 21 20 3D 20 73 73 65 72 64 009C1                                                            ;
                                    2F 21 2F 009D0                                                            ;
63 20 65 68 74 20 6D 6F 72 66 20 73 43 50 3C 009D3 P.ACL:  .ASCII  \<PCs from the call frame stack (starting\  ;
6B 63 61 74 73 20 65 6D 61 72 66 20 6C 6C 61 009E2                                                            ;
                  67 6E 69 74 72 61 74 73 28 20 009F1                                                         ;
65 63 65 72 20 74 73 6F 6D 20 68 74 69 77 20 009FB         .ASCII  \ with most recent):!/\                    ;
                              2F 21 3A 29 74 6E 00A0A                                                         ;
6C 6C 61 63 20 6D 6F 72 66 20 43 50 20 20 22 00A10 P.ACM:  .ASCII  \"  PC from call frame [!UL] = !XL!/\      ;
20 3D 20 5D 4C 55 21 5B 20 65 6D 61 72 66 20 00A1F                                                            ;
                        2F 21 4C 58 21 00A2E                                                                  ;
```

```
                                                    .PSECT  CODE,NOWRT,  SHR,2

                                          0FFC 00000 INTERNAL_ERROR:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1463
                53 00000000'  EF            04  C1 00002         ADDL3   #4, DS$Q_CRD_IMAGE_DESC+4, R3     ; 1496
                              52        04  A3  9E 0000A         MOVAB   4(R3), R2                         ; 1499
                              59        08  A3  9E 0000E         MOVAB   8(R3), R9                         ; 1500
                              58        0C  A3  9E 00012         MOVAB   12(R3), R8                        ; 1501
                              57        10  A3  9E 00016         MOVAB   16(R3), R7                        ; 1502
                              56        14  A3  9E 0001A         MOVAB   20(R3), R6                        ; 1503
                                        18  A3  9F 0001E         PUSHAB  24(R3)                            ; 1504
                                        1C  A3  9F 00021         PUSHAB  28(R3)                            ; 1505
                                        20  A3  9F 00024         PUSHAB  32(R3)                            ; 1506
                                        24  A3  9F 00027         PUSHAB  36(R3)                            ; 1507
                              55        28  A3  9E 0002A         MOVAB   40(R3), R5                        ; 1508
                              54        2C  A3  9E 0002E         MOVAB   44(R3), R4                        ; 1509
                                        30  A3  9F 00032         PUSHAB  48(R3)                            ; 1510
                                        34  A3  9F 00035         PUSHAB  52(R3)                            ; 1511
                              5B        38  A3  9E 00038         MOVAB   56(R3), R11                       ; 1512
                              5A        3C  A3  9E 0003C         MOVAB   60(R3), R10                       ; 1513
```

```
                                  00000000'  EF  9F 00040          PUSHAB  P.ABI                                  ; 1528
                                             01  DD 00046          PUSHL   #1                                     ;
                                             03  DD 00048          PUSHL   #3                                     ;
                      00000000G  9F          03  FB 0004A          CALLS   #3, a#DSX$PRINT                        ;
                                 50          62  CE 00051          MNEGL   (R2), R0                               ; 1532
                  0A             01          50  CF 00054          CASEL   R0, #1, #10                            ;
     0043        003B          0033        002B     00058 1$:      .WORD   2$-1$,-                                ;
     0063        005B          0053        004B     00060                  3$-1$,-                                ;
                 007B          0073        006B     00068                  4$-1$,-                                ;
                                                                            5$-1$,-                                ;
                                                                            6$-1$,-                                ;
                                                                            7$-1$,-                                ;
                                                                            8$-1$,-                                ;
                                                                            9$-1$,-                                ;
                                                                            10$-1$,-                               ;
                                                                            11$-1$,-                               ;
                                                                            12$-1$                                 ;
                                             62  DD 0006E          PUSHL   (R2)                                   ; 1551
                                  00000000'  EF  9F 00070          PUSHAB  P.ABV                                  ;
                                             01  DD 00076          PUSHL   #1                                     ;
                                             03  DD 00078          PUSHL   #3                                     ;
                      00000000G  9F          04  FB 0007A          CALLS   #4, a#DSX$PRINT                        ;
                                             61  11 00081          BRB     14$                                    ;
                                  00000000'  EF  9F 00083 2$:      PUSHAB  P.ABJ                                  ; 1534
                                             4E  11 00089          BRB     13$                                    ;
                                  00000000'  EF  9F 0008B 3$:      PUSHAB  P.ABK                                  ; 1535
                                             46  11 00091          BRB     13$                                    ;
                                  00000000'  EF  9F 00093 4$:      PUSHAB  P.ABL                                  ; 1536
                                             3E  11 00099          BRB     13$                                    ;
                                  00000000'  EF  9F 0009B 5$:      PUSHAB  P.ABM                                  ; 1537
                                             36  11 000A1          BRB     13$                                    ;
                                  00000000'  EF  9F 000A3 6$:      PUSHAB  P.ABN                                  ; 1538
                                             2E  11 000A9          BRB     13$                                    ;
                                  00000000'  EF  9F 000AB 7$:      PUSHAB  P.ABO                                  ; 1539
                                             26  11 000B1          BRB     13$                                    ;
                                  00000000'  EF  9F 000B3 8$:      PUSHAB  P.ABP                                  ; 1540
                                             1E  11 000B9          BRB     13$                                    ;
                                  00000000'  EF  9F 000BB 9$:      PUSHAB  P.ABQ                                  ; 1541
                                             16  11 000C1          BRB     13$                                    ;
                                  00000000'  EF  9F 000C3 10$:     PUSHAB  P.ABR                                  ; 1542
                                             0E  11 000C9          BRB     13$                                    ;
                                  00000000'  EF  9F 000CB 11$:     PUSHAB  P.ABS                                  ; 1543
                                             06  11 000D1          BRB     13$                                    ;
                                  00000000'  EF  9F 000D3 12$:     PUSHAB  P.ABT                                  ; 1544
                                             01  DD 000D9 13$:     PUSHL   #1                                     ;
                                             03  DD 000DB          PUSHL   #3                                     ;
                      00000000G  9F          03  FB 000DD          CALLS   #3, a#DSX$PRINT                        ;
                                  00000000'  EF  9F 000E4 14$:     PUSHAB  P.ABW                                  ; 1556
                                             01  DD 000EA          PUSHL   #1                                     ;
                                             03  DD 000EC          PUSHL   #3                                     ;
                      00000000G  9F          03  FB 000EE          CALLS   #3, a#DSX$PRINT                        ;
                                             69  DD 000F5          PUSHL   (R9)                                   ; 1557
                      00000000V  EF          01  FB 000F7          CALLS   #1, DSR$PRINT_TYPECODE_NAME            ;
                                             68  DD 000FE          PUSHL   (R8)                                   ; 1562
                                  00000000'  EF  9F 00100          PUSHAB  P.ABX                                  ;
                                             01  DD 00106          PUSHL   #1                                     ;
                                             03  DD 00108          PUSHL   #3                                     ;
```

```
                    00000000G  9F              04  FB 0010A          CALLS   #4, a#DSX$PRINT                          ;
                                                67  DD 00111          PUSHL   (R7)                                     ; 1566
                               00000000'        EF  9F 00113          PUSHAB  P.ABY                                    ;
                                                01  DD 00119          PUSHL   #1                                       ;
                                                03  DD 0011B          PUSHL   #3                                       ;
                    00000000G  9F              04  FB 0011D          CALLS   #4, a#DSX$PRINT                          ;
                                                65  DD 00124          PUSHL   (R5)                                     ; 1570
                               00000000'        EF  9F 00126          PUSHAB  P.ABZ                                    ;
                                                01  DD 0012C          PUSHL   #1                                       ;
                                                03  DD 0012E          PUSHL   #3                                       ;
                    00000000G  9F              04  FB 00130          CALLS   #4, a#DSX$PRINT                          ;
                                                64  DD 00137          PUSHL   (R4)                                     ; 1571
                               00000000'        EF  9F 00139          PUSHAB  P.ACA                                    ;
                                                01  DD 0013F          PUSHL   #1                                       ;
                                                03  DD 00141          PUSHL   #3                                       ;
                    00000000G  9F              04  FB 00143          CALLS   #4, a#DSX$PRINT                          ;
                               00000000'        EF  9F 0014A          PUSHAB  P.ACB                                    ; 1575
                                                01  DD 00150          PUSHL   #1                                       ;
                                                03  DD 00152          PUSHL   #3                                       ;
                    00000000G  9F              03  FB 00154          CALLS   #3, a#DSX$PRINT                          ;
                               00000000'        EF  9F 0015B          PUSHAB  P.ACC                                    ; 1576
                                                01  DD 00161          PUSHL   #1                                       ;
                                                03  DD 00163          PUSHL   #3                                       ;
                    00000000G  9F              03  FB 00165          CALLS   #3, a#DSX$PRINT                          ;
                               52               66  D0 0016C          MOVL    (R6), STATE                              ; 1578
                 02            00              52  CF 0016F 15$:      CASEL   STATE, #0, #2                            ; 1581
               001C          0011          0006      00173 16$:      .WORD   17$-16$,-                                ;
                                                                               18$-16$,-                                ;
                                                                               19$-16$                                  ;
                                     14    BE  DD 00179 17$:      PUSHL   a20(SP)                                  ; 1586
                               00000000'        EF  9F 0017C          PUSHAB  P.ACD                                    ;
                                                14  11 00182          BRB     20$                                      ;
                                     10    BE  DD 00184 18$:      PUSHL   a16(SP)                                  ; 1592
                               00000000'        EF  9F 00187          PUSHAB  P.ACE                                    ;
                                                09  11 0018D          BRB     20$                                      ;
                                     0C    BE  DD 0018F 19$:      PUSHL   a12(SP)                                  ; 1598
                               00000000'        EF  9F 00192          PUSHAB  P.ACF                                    ;
                                                01  DD 00198 20$:      PUSHL   #1                                       ;
                                                03  DD 0019A          PUSHL   #3                                       ;
                    00000000G  9F              04  FB 0019C          CALLS   #4, a#DSX$PRINT                          ;
 7E           01               52              01  7A 001A3          EMUL    #1, STATE, #1, -(SP)                     ; 1602
 52           52               8E              03  7B 001A8          EDIV    #3, (SP)+, STATE, STATE                  ;
                               66              52  D1 001AD          CMPL    STATE, (R6)                              ; 1604
                                                BD  12 001B0          BNEQ    15$                                      ;
                                     08    BE  DD 001B2          PUSHL   a8(SP)                                   ; 1608
                               00000000'        EF  9F 001B5          PUSHAB  P.ACG                                    ;
                                                01  DD 001BB          PUSHL   #1                                       ;
                                                03  DD 001BD          PUSHL   #3                                       ;
                    00000000G  9F              04  FB 001BF          CALLS   #4, a#DSX$PRINT                          ;
                                                6A  DD 001C6          PUSHL   (R10)                                    ; 1612
                               00000000'        EF  9F 001C8          PUSHAB  P.ACH                                    ;
                                                01  DD 001CE          PUSHL   #1                                       ;
                                                03  DD 001D0          PUSHL   #3                                       ;
                    00000000G  9F              04  FB 001D2          CALLS   #4, a#DSX$PRINT                          ;
                                     04    BE  DD 001D9          PUSHL   a4(SP)                                   ; 1613
                               00000000'        EF  9F 001DC          PUSHAB  P.ACI                                    ;
                                                01  DD 001E2          PUSHL   #1                                       ;
```

```
                                        03  DD 001E4          PUSHL   #3                                  ;
                  00000000G  9F         04  FB 001E6          CALLS   #4, a#DSX$PRINT                     ;
                                        6B  DD 001ED          PUSHL   (R11)                               ; 1614
                             00000000'  EF  9F 001EF          PUSHAB  P.ACJ                               ;
                                        01  DD 001F5          PUSHL   #1                                  ;
                                        03  DD 001F7          PUSHL   #3                                  ;
                  00000000G  9F         04  FB 001F9          CALLS   #4, a#DSX$PRINT                     ;
                                   00   BE  DD 00200          PUSHL   a0(SP)                              ; 1615
                             00000000'  EF  9F 00203          PUSHAB  P.ACK                               ;
                                        01  DD 00209          PUSHL   #1                                  ;
                                        03  DD 0020B          PUSHL   #3                                  ;
                  00000000G  9F         04  FB 0020D          CALLS   #4, a#DSX$PRINT                     ;
                             00000000'  EF  9F 00214          PUSHAB  P.ACL                               ; 1621
                                        01  DD 0021A          PUSHL   #1                                  ;
                                        03  DD 0021C          PUSHL   #3                                  ;
                  00000000G  9F         03  FB 0021E          CALLS   #3, a#DSX$PRINT                     ;
                             52         11  D0 00225          MOVL    #17, VECTOR                         ; 1623
                                      6342  D5 00228 21$:     TSTL    (R3)[VECTOR]                        ; 1624
                                        1B  13 0022B          BEQL    22$                                 ;
                                      6342  DD 0022D          PUSHL   (R3)[VECTOR]                        ; 1626
                                   EF   A2  9F 00230          PUSHAB  -17(VECTOR)                         ;
                             00000000'  EF  9F 00233          PUSHAB  P.ACM                               ;
                                        01  DD 00239          PUSHL   #1                                  ;
                                        03  DD 0023B          PUSHL   #3                                  ;
                  00000000G  9F         05  FB 0023D          CALLS   #5, a#DSX$PRINT                     ;
               E0            52         18  F3 00244          AOBLEQ  #24, VECTOR, 21$                    ; 1624
                                            04 00248 22$:     RET                                         ; 1631

; Routine Size:  585 bytes,    Routine Base:  CODE + 0673


;   1632  1     %SBTTL 'DSR$Print_TypeCode_Name Routine'
;   1633  1
;   1634  1     GLOBAL ROUTINE DSR$Print_TypeCode_Name (TypeCode) : NOVALUE =                                  ![07
;   1635  1                                                                                                    ! V
;   1636  1     !++
;   1637  1     ! Functional Description:
;   1638  1     !
;   1639  1     !       Print a short message saying what a given typecode is used for.
;   1640  1     !
;   1641  1     ! Formal Input Parameters:
;   1642  1     !
;   1643  1     !       TypeCode: The typecode constant.
;   1644  1     !
;   1645  1     ! Formal Output Parameters:
;   1646  1     !
;   1647  1     !       None
;   1648  1     !
;   1649  1     ! Side Effects:
;   1650  1     !
;   1651  1     !       None
;   1652  1     !
;   1653  1     ! Completion Codes:
;   1654  1     !
;   1655  1     !       None
;   1656  1     !--
;   1657  2     BEGIN                                   ! Routine DSR$Print_TypeCode_Name
```

```
;    1658  2
;    1659  2                MACRO
; M  1660  2                        TypeCode_Case (TypeCode_Name) =
; M  1661  2                                [%NAME ('DS$K_Type_', TypeCode_Name)]:
; M  1662  2                                        $Print (DS$K_Type_General, DS$K_PrintF, $ASCIC (TypeCode_Name, '!/'))
;    1663  2                        %;
;    1664  2
; P  1665  2                CRD_Assert ((CRD$K_Numb_Typecodes EQL 38),
; L  1666  2                        'You must make changes to the CASE statement below if you add more typecodes');
; %PRINT:       ASSERT: The assertion is true.
;    1667  2
;    1668  2                CASE .TypeCode FROM 0 TO (CRD$K_Numb_TypeCodes - 1) OF
;    1669  2                        SET
;    1670  2                        TypeCode_Case ('General');
;    1671  2                        TypeCode_Case ('DS_Prompt');
;    1672  2                        TypeCode_Case ('User_Prompt');
;    1673  2                        TypeCode_Case ('General_Error');
;    1674  2                        TypeCode_Case ('Errsup');
;    1675  2                        TypeCode_Case ('Errsys');
;    1676  2                        TypeCode_Case ('Errhard');
;    1677  2                        TypeCode_Case ('Errsoft');
;    1678  2                        TypeCode_Case ('Errdev');
;    1679  2                        TypeCode_Case ('Error_Body');
;    1680  2                        TypeCode_Case ('Error_End');
;    1681  2                        TypeCode_Case ('Exception_Head');
;    1682  2                        TypeCode_Case ('Exception');
;    1683  2                        TypeCode_Case ('Err_Halt');
;    1684  2                        TypeCode_Case ('Summary');
;    1685  2                        TypeCode_Case ('Program_Start');
;    1686  2                        TypeCode_Case ('Program_End');
;    1687  2                        TypeCode_Case ('First_Pass');
;    1688  2                        TypeCode_Case ('No_Tests');
;    1689  2                        TypeCode_Case ('Abort_Test');
;    1690  2                        TypeCode_Case ('Abort_Program');
;    1691  2                        TypeCode_Case ('Command_Err');
;    1692  2                        TypeCode_Case ('Command_Out');
;    1693  2                        TypeCode_Case ('Program_Info');
;    1694  2                        TypeCode_Case ('Start_Err');
;    1695  2                        TypeCode_Case ('Sequence_Error');
;    1696  2                        TypeCode_Case ('CRD_AutoTest');
;    1697  2                        TypeCode_Case ('Errprep');
;    1698  2                        TypeCode_Case ('Param_Error');
;    1699  2                        TypeCode_Case ('DS_Start');
;    1700  2                        TypeCode_Case ('Script_Pnf');
;    1701  2                        TypeCode_Case ('Script_Skip');
;    1702  2                        TypeCode_Case ('Script_Prompt');
;    1703  2                        TypeCode_Case ('Script_Echo');
;    1704  2                        TypeCode_Case ('Qio_Nodriver');
;    1705  2                        TypeCode_Case ('Qio_Wrongver');
;    1706  2                        TypeCode_Case ('Qio_Invadp');
;    1707  2                        TypeCode_Case ('Start_List');
;    1708  2
;    1709  2                        [INRANGE]:
;    1710  2                                $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Unknown typecode = !UL!/'), .Typecode
;    1711  2
;    1712  2                        [OUTRANGE]:
;    1713  2                                $Print (DS$K_Type_General_Error, DS$K_PrintF, $ASCIC ('Undefined typecode = !UL!/'), .TypeCo
```

```
;   1714  2                     TES;
;   1715  1     END;                                ! Routine DSR$Print_TypeCode_Name                                    ! ^
                                                                                                                        ![07



                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                                00A33 P.ACN:    .ASCII  <9>\General!/\                                  ;
                                                00A3D P.ACO:    .ASCII  <11>\DS_Prompt!/\                               ;
                                                00A49 P.ACP:    .ASCII  <13>\User_Prompt!/\                             ;
                                                00A57 P.ACQ:    .ASCII  <15>\General_Error!/\                           ;
                                                00A66                                                                  ;
                                                00A67 P.ACR:    .ASCII  <8>\Errsup!/\                                  ;
                                                00A70 P.ACS:    .ASCII  <8>\Errsys!/\                                  ;
                                                00A79 P.ACT:    .ASCII  <9>\Errhard!/\                                 ;
                                                00A83 P.ACU:    .ASCII  <9>\Errsoft!/\                                 ;
                                                00A8D P.ACV:    .ASCII  <8>\Errdev!/\                                  ;
                                                00A96 P.ACW:    .ASCII  <12>\Error_Body!/\                             ;
                                                00AA3 P.ACX:    .ASCII  <11>\Error_End!/\                              ;
                                                00AAF P.ACY:    .ASCII  <16>\Exception_Head!/\                          ;
                                                00ABE                                                                  ;
                                                00AC0 P.ACZ:    .ASCII  <11>\Exception!/\                               ;
                                                00ACC P.ADA:    .ASCII  <10>\Err_Halt!/\                                ;
                                                00AD7 P.ADB:    .ASCII  <9>\Summary!/\                                  ;
                                                00AE1 P.ADC:    .ASCII  <15>\Program_Start!/\                           ;
                                                00AF0                                                                  ;
                                                00AF1 P.ADD:    .ASCII  <13>\Program_End!/\                             ;
                                                00AFF P.ADE:    .ASCII  <12>\First_Pass!/\                              ;
                                                00B0C P.ADF:    .ASCII  <10>\No_Tests!/\                                ;
                                                00B17 P.ADG:    .ASCII  <12>\Abort_Test!/\                              ;
                                                00B24 P.ADH:    .ASCII  <15>\Abort_Program!/\                           ;
                                                00B33                                                                  ;
                                                00B34 P.ADI:    .ASCII  <13>\Command_Err!/\                             ;
                                                00B42 P.ADJ:    .ASCII  <13>\Command_Out!/\                             ;
                                                00B50 P.ADK:    .ASCII  <14>\Program_Info!/\                            ;
                                                00B5F P.ADL:    .ASCII  <11>\Start_Err!/\                               ;
                                                00B6B P.ADM:    .ASCII  <16>\Sequence_Error!/\                          ;
                                                00B7A                                                                  ;
                                                00B7C P.ADN:    .ASCII  <14>\CRD_AutoTest!/\                            ;
                                                00B8B P.ADO:    .ASCII  <9>\Errprep!/\                                  ;
                                                00B95 P.ADP:    .ASCII  <13>\Param_Error!/\                             ;
                                                00BA3 P.ADQ:    .ASCII  <10>\DS_Start!/\                                ;
                                                00BAE P.ADR:    .ASCII  <12>\Script_Pnf!/\                              ;
                                                00BBB P.ADS:    .ASCII  <13>\Script_Skip!/\                             ;
                                                00BC9 P.ADT:    .ASCII  <15>\Script_Prompt!/\                           ;
                                                00BD8                                                                  ;
                                                00BD9 P.ADU:    .ASCII  <13>\Script_Echo!/\                             ;
                                                00BE7 P.ADV:    .ASCII  <14>\Qio_Nodriver!/\                            ;
                                                00BF6 P.ADW:    .ASCII  <14>\Qio_Wrongver!/\                            ;
                                                00C05 P.ADX:    .ASCII  <12>\Qio_Invadp!/\                              ;
                                                00C12 P.ADY:    .ASCII  <12>\Start_List!/\                              ;
                                                00C1F P.ADZ:    .ASCII  <24>\Unknown typecode = !UL!/\                  ;
                                                00C2E                                                                  ;
                                                00C38 P.AEA:    .ASCII  <26>\Undefined typecode = !UL!/\                ;
                                                00C47                                                                  ;
```

```
                                                                  .PSECT  CODE,NOWRT,  SHR,2

                                          000C 00000              .ENTRY  DSR$PRINT_TYPECODE_NAME, Save R2,R3          ; 1634
                          53 00000000G  9F  9E 00002              MOVAB   a#DSX$PRINT, R3                              ;
                          52 00000000'  EF  9E 00009              MOVAB   P.AEA, R2                                    ;
                 25       00       04   AC  CF 00010              CASEL   TYPECODE, #0, #37                            ; 1668
 006B          0065     005F          0059     00015 1$:          .WORD   2$-1$,-                                      ;
 0083          007D     0077          0071     0001D                      3$-1$,-                                      ;
 009B          0095     008F          0089     00025                      4$-1$,-                                      ;
 00B3          00AD     00A7          00A1     0002D                      5$-1$,-                                      ;
 00CB          00C5     00BF          00B9     00035                      6$-1$,-                                      ;
 00E3          00DD     00D7          00D1     0003D                      7$-1$,-                                      ;
 00FB          00F5     00EF          00E9     00045                      8$-1$,-                                      ;
 0113          010D     0107          0101     0004D                      9$-1$,-                                      ;
 0127          0122     011D          0118     00055                      10$-1$,-                                     ;
                        0131          012C     0005D                      11$-1$,-                                     ;
                                                                          12$-1$,-                                     ;
                                                                          13$-1$,-                                     ;
                                                                          14$-1$,-                                     ;
                                                                          15$-1$,-                                     ;
                                                                          16$-1$,-                                     ;
                                                                          17$-1$,-                                     ;
                                                                          18$-1$,-                                     ;
                                                                          19$-1$,-                                     ;
                                                                          20$-1$,-                                     ;
                                                                          21$-1$,-                                     ;
                                                                          22$-1$,-                                     ;
                                                                          23$-1$,-                                     ;
                                                                          25$-1$,-                                     ;
                                                                          27$-1$,-                                     ;
                                                                          29$-1$,-                                     ;
                                                                          31$-1$,-                                     ;
                                                                          33$-1$,-                                     ;
                                                                          35$-1$,-                                     ;
                                                                          37$-1$,-                                     ;
                                                                          39$-1$,-                                     ;
                                                                          41$-1$,-                                     ;
                                                                          43$-1$,-                                     ;
                                                                          45$-1$,-                                     ;
                                                                          46$-1$,-                                     ;
                                                                          48$-1$,-                                     ;
                                                                          50$-1$,-                                     ;
                                                                          52$-1$,-                                     ;
                                                                          54$-1$                                       ;
                                  04   AC  DD 00061              PUSHL   TYPECODE                                     ; 1713
                                       52  DD 00064              PUSHL   R2                                           ;
                                       01  DD 00066              PUSHL   #1                                           ;
                                       03  DD 00068              PUSHL   #3                                           ;
                          63           04  FB 0006A              CALLS   #4, DSX$PRINT                                ;
                                           04 0006D              RET                                                  ;
                                FDFB   C2  9F 0006E 2$:          PUSHAB  P.ACN                                        ; 1670
                                       7C  11 00072              BRB     24$                                          ;
                                FE05   C2  9F 00074 3$:          PUSHAB  P.ACO                                        ; 1671
                                       7C  11 00078              BRB     26$                                          ;
                                FE11   C2  9F 0007A 4$:          PUSHAB  P.ACP                                        ; 1672
                                       7C  11 0007E              BRB     28$                                          ;
```

```
                                FE1F    C2  9F 00080 5$:     PUSHAB  P.ACQ                           ; 1673
                                        7C  11 00084        BRB     30$                             ;
                                FE2F    C2  9F 00086 6$:     PUSHAB  P.ACR                          ; 1674
                                        7C  11 0008A        BRB     32$                             ;
                                FE38    C2  9F 0008C 7$:     PUSHAB  P.ACS                          ; 1675
                                        7C  11 00090        BRB     34$                             ;
                                FE41    C2  9F 00092 8$:     PUSHAB  P.ACT                          ; 1676
                                        7C  11 00096        BRB     36$                             ;
                                FE4B    C2  9F 00098 9$:     PUSHAB  P.ACU                          ; 1677
                                        7C  11 0009C        BRB     38$                             ;
                                FE55    C2  9F 0009E 10$:    PUSHAB  P.ACV                          ; 1678
                                        7C  11 000A2        BRB     40$                             ;
                                FE5E    C2  9F 000A4 11$:    PUSHAB  P.ACW                          ; 1679
                                        7C  11 000A8        BRB     42$                             ;
                                FE6B    C2  9F 000AA 12$:    PUSHAB  P.ACX                          ; 1680
                                        7B  11 000AE        BRB     44$                             ;
                                FE77    C2  9F 000B0 13$:    PUSHAB  P.ACY                          ; 1681
                                        7F  11 000B4        BRB     47$                             ;
                                FE88    C2  9F 000B6 14$:    PUSHAB  P.ACZ                          ; 1682
                                        7E  11 000BA        BRB     49$                             ;
                                FE94    C2  9F 000BC 15$:    PUSHAB  P.ADA                          ; 1683
                                        7D  11 000C0        BRB     51$                             ;
                                FE9F    C2  9F 000C2 16$:    PUSHAB  P.ADB                          ; 1684
                                        7C  11 000C6        BRB     53$                             ;
                                FEA9    C2  9F 000C8 17$:    PUSHAB  P.ADC                          ; 1685
                                        7B  11 000CC        BRB     55$                             ;
                                FEB9    C2  9F 000CE 18$:    PUSHAB  P.ADD                          ; 1686
                                        75  11 000D2        BRB     55$                             ;
                                FEC7    C2  9F 000D4 19$:    PUSHAB  P.ADE                          ; 1687
                                        6F  11 000D8        BRB     55$                             ;
                                FED4    C2  9F 000DA 20$:    PUSHAB  P.ADF                          ; 1688
                                        69  11 000DE        BRB     55$                             ;
                                FEDF    C2  9F 000E0 21$:    PUSHAB  P.ADG                          ; 1689
                                        63  11 000E4        BRB     55$                             ;
                                FEEC    C2  9F 000E6 22$:    PUSHAB  P.ADH                          ; 1690
                                        5D  11 000EA        BRB     55$                             ;
                                FEFC    C2  9F 000EC 23$:    PUSHAB  P.ADI                          ; 1691
                                        57  11 000F0 24$:    BRB     55$                             ;
                                FF0A    C2  9F 000F2 25$:    PUSHAB  P.ADJ                          ; 1692
                                        51  11 000F6 26$:    BRB     55$                             ;
                                FF18    C2  9F 000F8 27$:    PUSHAB  P.ADK                          ; 1693
                                        4B  11 000FC 28$:    BRB     55$                             ;
                                FF27    C2  9F 000FE 29$:    PUSHAB  P.ADL                          ; 1694
                                        45  11 00102 30$:    BRB     55$                             ;
                                FF33    C2  9F 00104 31$:    PUSHAB  P.ADM                          ; 1695
                                        3F  11 00108 32$:    BRB     55$                             ;
                                FF44    C2  9F 0010A 33$:    PUSHAB  P.ADN                          ; 1696
                                        39  11 0010E 34$:    BRB     55$                             ;
                                FF53    C2  9F 00110 35$:    PUSHAB  P.ADO                          ; 1697
                                        33  11 00114 36$:    BRB     55$                             ;
                                FF5D    C2  9F 00116 37$:    PUSHAB  P.ADP                          ; 1698
                                        2D  11 0011A 38$:    BRB     55$                             ;
                                FF6B    C2  9F 0011C 39$:    PUSHAB  P.ADQ                          ; 1699
                                        27  11 00120 40$:    BRB     55$                             ;
                                FF76    C2  9F 00122 41$:    PUSHAB  P.ADR                          ; 1700
                                        21  11 00126 42$:    BRB     55$                             ;
                                  83    A2  9F 00128 43$:    PUSHAB  P.ADS                          ; 1701
```

```
                                        1C  11 0012B 44$:    BRB     55$                                  ;
                                   91   A2  9F 0012D 45$:    PUSHAB  P.ADT                                ; 1702
                                        17  11 00130         BRB     55$                                  ;
                                   A1   A2  9F 00132 46$:    PUSHAB  P.ADU                                ; 1703
                                        12  11 00135 47$:    BRB     55$                                  ;
                                   AF   A2  9F 00137 48$:    PUSHAB  P.ADV                                ; 1704
                                        0D  11 0013A 49$:    BRB     55$                                  ;
                                   BE   A2  9F 0013C 50$:    PUSHAB  P.ADW                                ; 1705
                                        08  11 0013F 51$:    BRB     55$                                  ;
                                   CD   A2  9F 00141 52$:    PUSHAB  P.ADX                                ; 1706
                                        03  11 00144 53$:    BRB     55$                                  ;
                                   DA   A2  9F 00146 54$:    PUSHAB  P.ADY                                ; 1707
                                        01  DD 00149 55$:    PUSHL   #1                                   ;
                                        7E  D4 0014B         CLRL    -(SP)                                ;
                              63        03  FB 0014D         CALLS   #3, DSX$PRINT                        ;
                                            04 00150         RET                                          ; 1715

; Routine Size:  337 bytes,    Routine Base:  CODE + 08BC


;   1716  1     END                                  ! End of CRDIMAGE module
;   1717  0     ELUDOM




;                                  PSECT SUMMARY
;
;       Name                  Bytes                        Attributes
;
;  DATA                        3155  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                         132  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                        2573  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...
$ASCIC	Unbound		456 457 458 459 460 461 546 623 772 808
			829 941 979 1047 1238 1284 1336 1456 1522 1525
			1630 1662
	Macro	Lib02	338 339 340 343 344 346 347 348 349 353
			354 355 356 358 360 361 364 365 366 368
			369 371 374 379 380 382 383 384 385 1528
			1534 1535 1536 1537 1538 1539 1540 1541 1542 1543
			1544 1547 1551 1556 1562 1566 1570 1571 1575 1576
			1585 1591 1597 1608 1612 1613 1614 1615 1621 1626
			1670 1671 1672 1673 1674 1675 1676 1677 1678 1679
			1680 1681 1682 1683 1684 1685 1686 1687 1688 1689
			1690 1691 1692 1693 1694 1695 1696 1697 1698 1699
			1700 1701 1702 1703 1704 1705 1706 1707 1710 1713
$ASCID	Macro	Lib02	331 334
$CLOSE	KeyWMacr	Lib04	655
$CNTREG	KeyWMacr	Lib04	907
$CRD_LITERALS	Macro	Lib03	183
$DS_CNTRLC	KeyWMacr	Lib02	818 831 1246
$DS_DSADEF	Macro	Lib02	186
$DS_DSSDEF	Macro	Lib02	184
$DS_LOAD	KeyWMacr	Lib02	725
$DS_MMOFF	Macro	Lib02	548
$DS_TYPEDEF	Macro	Lib02	185
$EQULST	Macro	Lib04	183 185 195 204 222 233 244
$ER	Comptime		327
$EXPREG	KeyWMacr	Lib04	677
$FAB_DECL	Macro	Lib04	534
$FAB_INIT	KeyWMacr	Lib04	635
$MODULE	Bind		327
$OPEN	KeyWMacr	Lib04	639
$PRINT	Unbound		468 469 471 473 475 477 482 483 485 487
			489 491 546 623 772 808 829 941 979 1047
			1238 1284 1336 1456 1521 1525 1630 1662
	Macro	Lib01	564 749 750 752 756 760 764 1360 1366 1375
			1381 1402 1408 1420 1422 1430 1446 1447 1448 1527
			1534 1535 1536 1537 1538 1539 1540 1541 1542 1543
			1544 1546 1550 1556 1562 1566 1570 1571 1575 1576
			1585 1591 1597 1608 1612 1613 1614 1615 1620 1625
			1670 1671 1672 1673 1674 1675 1676 1677 1678 1679
			1680 1681 1682 1683 1684 1685 1686 1687 1688 1689
			1690 1691 1692 1693 1694 1695 1696 1697 1698 1699
			1700 1701 1702 1703 1704 1705 1706 1707 1710 1713
$RMS_BITFLD_INI	Macro	Lib04	636
$RMS_BITS	Macro	Lib04	636
$RMS_CODFLD_INI	Macro	Lib04	636
$RMS_OR	Macro	Lib04	636
$RMS_PTR	Bind		636 636=
	Bind		637 637=
$RMS_VALFLD_INI	Macro	Lib04	636 637
$XABFHC_DECL	Macro	Lib04	535
$XABFHC_INIT	KeyWMacr	Lib04	637
ADDRESS	Bind		1502 1566.
AUTO$K_EXIT_AUTOSIZER_ERROR	Literal		183
AUTO$K_EXIT_CONTROL_C	Literal		183
AUTO$K_EXIT_DUMMY_2	Literal		183

Symbol	Type	Defined	Referenced ...
AUTO$K_EXIT_DUMMY_3	Literal	183	
AUTO$K_EXIT_ERRORS_COMPLETE	Literal	183	
AUTO$K_EXIT_ERRORS_INCOMPLETE	Literal	183	
AUTO$K_EXIT_INTERNAL_ERROR	Literal	183	
AUTO$K_EXIT_INV_BASE_SYSTEM	Literal	183	
AUTO$K_EXIT_LAST_REASON	Literal	183	183
AUTO$K_EXIT_NOERRORS_COMPLETE	Literal	183	
AUTO$K_EXIT_NOERRORS_INCOMPLETE	Literal	183	
AUTO$K_EXIT_NOERRORS_PREP	Literal	183	
A_CRD_DISPATCH_VECTORS	RoutForm	1052	1157. 1166m 1172a. 1175a. 1184a. 1187a.
BEGIN_BLISS	External	283	1299c
CORRUPTED_ERROR	Macro	1144	1173 1176
CRD$EXPREG	ExtRout	257	691c
CRD$K_ACTION_EQL_DO_NOTHING	Literal	183	1536
CRD$K_ACTION_RANGE_ERROR	Literal	183	1535
CRD$K_ADVANCE_DIAGNOSTIC	Literal	183	
CRD$K_ADVANCE_GENERAL_COM	Literal	183	
CRD$K_ADVANCE_STATE	Literal	183	
CRD$K_ALLOCATION_ERROR	Literal	183	1534
CRD$K_ASSIGN_PTABLE_PTRS	Literal	183	
CRD$K_AUTOSIZER_ERROR	Literal	183	
CRD$K_AUTOSIZER_SET_FLAGS	Literal	183	
CRD$K_BAD_ACTION_NUMBER	Literal	183	1542
CRD$K_BAD_TYPECODE_ERROR	Literal	183	1537
CRD$K_BASE_SYSTEM_IDC	Literal	183	
CRD$K_BASE_SYSTEM_LESI	Literal	183	
CRD$K_BASE_SYSTEM_NULL	Literal	183	
CRD$K_BASE_SYSTEM_UDA_0	Literal	183	
CRD$K_BASE_SYSTEM_UDA_1	Literal	183	
CRD$K_BASE_SYSTEM_UDA_2	Literal	183	
CRD$K_BASE_SYSTEM_UDA_3	Literal	183	
CRD$K_CRD_INITIALIZATION	Literal	183	
CRD$K_CREATE_HST	Literal	183	1544
CRD$K_DATA_LENGTH_ERROR	Literal	183	1538
CRD$K_DDB_AUTO_SUPPORT_ENTRY	Literal	183	
CRD$K_DDB_BLINK	Literal	183	
CRD$K_DDB_FLINK	Literal	183	
CRD$K_DDB_LAST_ENTRY	Literal	183	
CRD$K_DDB_MEMORY_SIZE	Literal	183	
CRD$K_DDB_MENU_DEVICE_NUMB	Literal	183	
CRD$K_DDB_MENU_SUPPORT_ENTRY	Literal	183	
CRD$K_DDB_MENU_SUPPORT_SIZE	Literal	183	
CRD$K_DDB_NUMB_SUPPORT_TABLES	Literal	183	
CRD$K_DDB_PTABLE_ENTRY	Literal	183	
CRD$K_DDB_STATUS	Literal	183	
CRD$K_DDB_SUPBLK_BLOCK_PTR	Literal	183	
CRD$K_DEVICE_NAME	Literal	183	
CRD$K_DIAGNOSTIC_ERROR	Literal	183	
CRD$K_DIAGNOSTIC_NAME	Literal	183	
CRD$K_DONE_GENERAL_COMS	Literal	183	
CRD$K_DO_NOTHING	Literal	183	
CRD$K_DUMMY_10	Literal	183	
CRD$K_DUMMY_11	Literal	183	
CRD$K_DUMMY_12	Literal	183	
CRD$K_DUMMY_13	Literal	183	

Symbol	Type	Defined	Referenced ...
CRD$K_DUMMY_3	Literal	183	
CRD$K_DUMMY_4	Literal	183	
CRD$K_DUMMY_5	Literal	183	
CRD$K_DUMMY_6	Literal	183	
CRD$K_DUMMY_7	Literal	183	
CRD$K_DUMMY_8	Literal	183	
CRD$K_DUMMY_9	Literal	183	
CRD$K_EXIT_CRD	Literal	183	
CRD$K_HOOK_POINT	Literal	183	
CRD$K_HS_INTERNAL_ERROR	Literal	183	1541
CRD$K_IDC_EVDLB	Literal	183	
CRD$K_IDC_EVDLC	Literal	183	
CRD$K_IDC_EVDLD	Literal	183	
CRD$K_IDC_EVRAD_0	Literal	183	
CRD$K_IDC_EVRAD_1	Literal	183	
CRD$K_IDC_LAST_DIAGNOSTIC	Literal	183	
CRD$K_INTERNAL_ERROR	Literal	183	
CRD$K_LAST_ACTION_ROUTINE	Literal	183	
CRD$K_LAST_ENTRY	Literal	183	
CRD$K_LAST_FUNCTION	Literal	183	
CRD$K_LAST_HOOK_POINT	Literal	183	
CRD$K_LAST_INTERNAL_ERROR	Literal	183	1532
CRD$K_LAST_TYPE	Literal	183	
CRD$K_LESI_ENWCA	Literal	183	
CRD$K_LESI_EVRMB_0	Literal	183	
CRD$K_LESI_EVRMB_1	Literal	183	
CRD$K_LESI_LAST_DIAGNOSTIC	Literal	183	
CRD$K_LEVEL_4_PASSED	Literal	183	
CRD$K_LOAD_AUTOSIZER	Literal	183	
CRD$K_LOAD_ERROR	Literal	183	
CRD$K_LOAD_GENERAL_COM	Literal	183	
CRD$K_LOAD_LOAD_COM	Literal	183	
CRD$K_LOAD_SELECT_COM	Literal	183	
CRD$K_LOAD_SET_EVENT_COM	Literal	183	
CRD$K_LOAD_SET_FLAGS_COM	Literal	183	
CRD$K_LOAD_START_COM	Literal	183	
CRD$K_LOAD_SUP_FILE	Literal	183	1543
CRD$K_MM_INTERNAL_ERROR	Literal	183	1539
CRD$K_NEW_LINE	Literal	183	
CRD$K_NUMB_DISPATCH_VECTORS	Literal	Lib03	938
CRD$K_NUMB_TYPECODES	Literal	Lib03	1666 1668
CRD$K_PREPARATION_ERROR	Literal	183	
CRD$K_PREPARATION_ERROR_TEXT	Literal	183	
CRD$K_RUNTIME	Literal	183	
CRD$K_SAME_LINE	Literal	183	
CRD$K_SET_EVENT_ARGS	Literal	183	
CRD$K_SET_FLAGS_ARGS	Literal	183	
CRD$K_SET_UP_DIAGNOSTIC	Literal	183	
CRD$K_START	Literal	183	
CRD$K_START_AUTOSIZER	Literal	183	
CRD$K_START_ERROR	Literal	183	
CRD$K_START_QUALIFIERS	Literal	183	
CRD$K_STAR_LINE	Literal	183	
CRD$K_STATE_ADVANCE_DIAGNOSTIC	Literal	183	
CRD$K_STATE_ADVANCE_GENERAL_COM	Literal	183	

Symbol	Type	Defined	Referenced ...
CRD$K_STATE_ASSIGN_PTABLE_PTRS	Literal	183	
CRD$K_STATE_AUTOSIZER_ERROR	Literal	183	
CRD$K_STATE_AUTOSIZER_RUNNING	Literal	183	
CRD$K_STATE_AUTOSIZER_SET_FLAGS	Literal	183	
CRD$K_STATE_DIAGNOSTIC_ERROR	Literal	183	
CRD$K_STATE_DIAGNOSTIC_RUNNING	Literal	183	
CRD$K_STATE_DONE_DIAGNOSTICS	Literal	183	
CRD$K_STATE_DONE_GENERAL_COMS	Literal	183	
CRD$K_STATE_DUMMY_1	Literal	183	
CRD$K_STATE_DUMMY_10	Literal	183	
CRD$K_STATE_DUMMY_12	Literal	183	
CRD$K_STATE_DUMMY_13	Literal	183	
CRD$K_STATE_DUMMY_2	Literal	183	
CRD$K_STATE_DUMMY_3	Literal	183	
CRD$K_STATE_DUMMY_4	Literal	183	
CRD$K_STATE_DUMMY_6	Literal	183	
CRD$K_STATE_DUMMY_7	Literal	183	
CRD$K_STATE_DUMMY_8	Literal	183	
CRD$K_STATE_EXIT_CRD	Literal	183	
CRD$K_STATE_INTERNAL_ERROR	Literal	183	
CRD$K_STATE_LAST_STATE	Literal	183	
CRD$K_STATE_LEVEL_4_PASSED	Literal	183	
CRD$K_STATE_LOAD_AUTOSIZER	Literal	183	
CRD$K_STATE_LOAD_ERROR	Literal	183	
CRD$K_STATE_LOAD_GENERAL_COM	Literal	183	
CRD$K_STATE_LOAD_LOAD_COM	Literal	183	
CRD$K_STATE_LOAD_SELECT_COM	Literal	183	
CRD$K_STATE_LOAD_SET_EVENT_COM	Literal	183	
CRD$K_STATE_LOAD_SET_FLAGS_COM	Literal	183	
CRD$K_STATE_LOAD_START_COM	Literal	183	
CRD$K_STATE_PREPARATION_ERROR	Literal	183	
CRD$K_STATE_SET_UP_DIAGNOSTIC	Literal	183	
CRD$K_STATE_START	Literal	183	
CRD$K_STATE_START_AUTOSIZER	Literal	183	
CRD$K_STATE_START_ERROR	Literal	183	
CRD$K_TEST_START_TEXT	Literal	183	
CRD$K_TOTAL_NUMB_VECTORS	Literal	Lib03	395 396 603 1044 1623
CRD$K_TQ_INTERNAL_ERROR	Literal	183	1540
CRD$K_TYPECODE	Literal	183	
CRD$K_UDA_0_EVDLB	Literal	183	
CRD$K_UDA_0_EVDLC	Literal	183	
CRD$K_UDA_0_EVDLD	Literal	183	
CRD$K_UDA_0_EVRLA_0	Literal	183	
CRD$K_UDA_0_LAST_DIAGNOSTIC	Literal	183	
CRD$K_UDA_1_EVDLB	Literal	183	
CRD$K_UDA_1_EVDLC	Literal	183	
CRD$K_UDA_1_EVDLD	Literal	183	
CRD$K_UDA_1_EVRLA_0	Literal	183	
CRD$K_UDA_1_EVRLA_1	Literal	183	
CRD$K_UDA_1_LAST_DIAGNOSTIC	Literal	183	
CRD$K_UDA_2_EVDLB	Literal	183	
CRD$K_UDA_2_EVDLC	Literal	183	
CRD$K_UDA_2_EVDLD	Literal	183	
CRD$K_UDA_2_EVRLA_0	Literal	183	
CRD$K_UDA_2_LAST_DIAGNOSTIC	Literal	183	

Symbol	Type	Defined	Referenced ...
CRD$K_UDA_3_EVDLB	Literal	183	
CRD$K_UDA_3_EVDLC	Literal	183	
CRD$K_UDA_3_EVDLD	Literal	183	
CRD$K_UDA_3_EVRLA_0	Literal	183	
CRD$K_UDA_3_EVRLA_1	Literal	183	
CRD$K_UDA_3_LAST_DIAGNOSTIC	Literal	183	
CRD$V_CRD_DEBUG	Macro	Lib01 575	1392
CRD$V_CTRL_C_NO_ECHO	Macro	Lib01 583	947 985 1252
CRD$V_FLUSH_CTRL_C	Macro	Lib01 584	948 986 1253
CRD_ACTUAL_LENGTH	Local	526	632a 669. 733. 733a 738.
CRD_ASSERT	Macro	Lib03 1665	
CRD_BYTES	Bind	747	754. 758. 762. 766.
CRD_DISPATCH_VECTORS	Bind	1157	1204. 1204a 1206=
CRD_ERROR	Unbound		1145 1149
	Routine	1304	173f 549c 589c 640c 663c 720c 723c 736c 857c 922c 1173c 1176c 1185c 1188c
CRD_EXIT	Routine	1211	170f 953c 1425c 1458c
CRD_FILE_ATTRIBUTES	Local	527	633a 733a
CRD_IMAGE_LOCATION	Local	533	682a 694a 704. 706. 713. 714. 733. 747. 769.
CRD_VECTORS	Bind	800	851. 852.
	Register	1331	1338=
CRD_VERSION_NUMBER	Bind	1515	
CURRENT_DDB_PTR	Bind	1510	1613.
CURRENT_HSB_NUMB	Bind	1512	1614.
CURRENT_HSB_PTR	Bind	1511	1615.
CURRENT_STATE_TABLE	Bind	1503	1578. 1604.
CURRENT_TQ_ENTRY	Bind	1513	1612.
DEBUG_OUT_1	Unbound		456 468 482 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
DEBUG_OUT_2	Unbound		457 469 483 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
DEBUG_OUT_3	Unbound		458 471 485 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
DEBUG_OUT_4	Unbound		459 473 487 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
DEBUG_OUT_5	Unbound		460 475 489 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
DEBUG_OUT_6	Unbound		461 477 491 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
DEBUG_STRING	MacrForm	441	456
DEBUG_VECTORS	Bind	1496	1499a 1500a 1501a 1502a 1503a 1504a 1505a 1506a 1507a 1508a 1509a 1510a 1511a 1512a 1513a 1514a 1515. 1624. 1626.
DEFAULT	Local	529	540m 631= 636a.
DS$ABORT	Literal	184	
DS$AL_DS_DISPATCH_VECTORS	External	298	595m 611. 614m 619. 744m 754. 758. 762. 766. 806m 881m 886= 1019m 1024= 1158a 1165m 1184. 1187. 1198. 1490m 1516a
DS$ASKADR	Literal	184	
DS$ASKDATA	Literal	184	
DS$ASKLGCL	Literal	184	
DS$ASKSTR	Literal	184	
DS$ASKVLD	Literal	184	
DS$ATTACH	Literal	184	
DS$A_AUTO_OR_MENU	Own	438	578= 582= 588= 1430.
DS$A_CRD_VECTORS	Unbound		451 476 490 546 623 772 808 829 941 979

Symbol	Type	Defined	Referenced ...
			1047 1238 1284 1336 1456 1525 1630
	Own	395	396= 604a 939= 1045=
DS$A_SAVED_LASTADR	Unbound		478 492 546 623 772 808 829 941 979 1047
			1238 1284 1336 1456 1525 1630
	Own	424	424= 666= 839. 841. 842= 1029=
DS$BGNSUB	Literal	184	
DS$BRANCH	Literal	184	
DS$BREAK	Literal	184	
DS$CANWAIT	Literal	184	
DS$CHANNEL	Literal	184	
DS$CKLOOP	Literal	184	
DS$CLRVEC	Literal	184	
DS$CNTRLC	Literal	184	
	ExtRout	818	818c 831e 831c 1246e 1246c
DS$CVTREG	Literal	184	
DS$ENDPASS	Literal	184	
DS$ENDSUB	Literal	184	
DS$ERRDEV	Literal	184	
DS$ERRHARD	Literal	184	
DS$ERRPREP	Literal	184	
DS$ERRSOFT	Literal	184	
DS$ERRSYS	Literal	184	
DS$ESCAPE	Literal	184	
DS$FREEDBGSYM	Literal	184	
DS$GA_DS_CTRL_C_FIRST	External	286	814=
DS$GA_DS_CTRL_C_SECOND	External	289	815=
DS$GA_LASTADR	Unbound		478 492 546 623 772 808 829 941 979 1047
			1238 1284 1336 1456 1525 1630
	External	304	666. 738= 738. 841=
DS$GETBUF	Literal	184	
DS$GETTERM	Literal	184	
DS$GL_CRD_DEBUG	External	308	574.
DS$GL_CRD_FLAGS	External	301	575= 583= 584= 947= 948= 985= 986= 1252= 1253= 1392.
DS$GL_FLAGS	Unbound		470 484 546 623 772 808 829 941 979 1047
			1238 1284 1336 1456 1525 1630
	External	295	552. 1245= 1260= 1281=
DS$GPHARD	Literal	184	
DS$HELP	Literal	184	
DS$INITSCB	Literal	184	
DS$INLOOP	Literal	184	
DS$K_PRINTB	Literal	185	
DS$K_PRINTF	Unbound		468 469 471 473 475 477 482 483 485 487
			489 491 546 623 772 808 829 941 979 1047
			1238 1284 1336 1456 1521 1525 1630 1662
	Literal	185	564 749 750 754 758 762 766 1360 1366 1375
			1381 1402 1408 1420 1422 1430 1446 1447 1448 1528
			1534 1535 1536 1537 1538 1539 1540 1541 1542 1543
			1544 1547 1551 1556 1562 1566 1570 1571 1575 1576
			1586 1592 1598 1608 1612 1613 1614 1615 1621 1626
			1670 1671 1672 1673 1674 1675 1676 1677 1678 1679
			1680 1681 1682 1683 1684 1685 1686 1687 1688 1689
			1690 1691 1692 1693 1694 1695 1696 1697 1698 1699
			1700 1701 1702 1703 1704 1705 1706 1707 1710 1713
DS$K_PRINTI	Literal	185	
DS$K_PRINTX	Literal	185	

Symbol	Type	Defined	Referenced ...
DS$K_TYPE_ABORT_PROGRAM	Literal	185	1690
DS$K_TYPE_ABORT_TEST	Literal	185	1689
DS$K_TYPE_COMMAND_ERR	Literal	185	1691
DS$K_TYPE_COMMAND_OUT	Literal	185	1692
DS$K_TYPE_CRD_AUTOTEST	Unbound		468 469 471 473 475 477 546 623 772 808
			829 941 979 1047 1238 1284 1336 1456 1525 1630
	Literal	185	564 749 750 754 758 762 766 1696
DS$K_TYPE_DS_PROMPT	Literal	185	1671
DS$K_TYPE_DS_START	Literal	185	1699
DS$K_TYPE_ERRDEV	Literal	185	1678
DS$K_TYPE_ERRHARD	Literal	185	1676
DS$K_TYPE_ERROR_BODY	Literal	185	1679
DS$K_TYPE_ERROR_END	Literal	185	1680
DS$K_TYPE_ERRPREP	Literal	185	1697
DS$K_TYPE_ERRSOFT	Literal	185	1677
DS$K_TYPE_ERRSUP	Literal	185	1674
DS$K_TYPE_ERRSYS	Literal	185	1675
DS$K_TYPE_ERR_HALT	Literal	185	1683
DS$K_TYPE_EXCEPTION	Literal	185	1682
DS$K_TYPE_EXCEPTION_HEAD	Literal	185	1681
DS$K_TYPE_FIRST_PASS	Literal	185	1687
DS$K_TYPE_GENERAL	Unbound		482 483 485 487 489 491 546 623 772 808
			829 941 979 1047 1238 1284 1336 1456 1525 1630
			1662
	Literal	185	1670 1671 1672 1673 1674 1675 1676 1677 1678 1679
			1680 1681 1682 1683 1684 1685 1686 1687 1688 1689
			1690 1691 1692 1693 1694 1695 1696 1697 1698 1699
			1700 1701 1702 1703 1704 1705 1706 1707
DS$K_TYPE_GENERAL_ERROR	Unbound		1521
	Literal	185	1360 1366 1375 1381 1402 1408 1420 1422 1430 1446
			1447 1448 1528 1534 1535 1536 1537 1538 1539 1540
			1541 1542 1543 1544 1547 1551 1556 1562 1566 1570
			1571 1575 1576 1586 1592 1598 1608 1612 1613 1614
			1615 1621 1626 1673 1710 1713
DS$K_TYPE_NO_TESTS	Literal	185	1688
DS$K_TYPE_PARAM_ERROR	Literal	185	1698
DS$K_TYPE_PROGRAM_END	Literal	185	1686
DS$K_TYPE_PROGRAM_INFO	Literal	185	1693
DS$K_TYPE_PROGRAM_START	Literal	185	1685
DS$K_TYPE_QIO_INVADP	Literal	185	1706
DS$K_TYPE_QIO_NODRIVER	Literal	185	1704
DS$K_TYPE_QIO_WRONGVER	Literal	185	1705
DS$K_TYPE_SCRIPT_ECHO	Literal	185	1703
DS$K_TYPE_SCRIPT_PNF	Literal	185	1700
DS$K_TYPE_SCRIPT_PROMPT	Literal	185	1702
DS$K_TYPE_SCRIPT_SKIP	Literal	185	1701
DS$K_TYPE_SEQUENCE_ERROR	Literal	185	1695
DS$K_TYPE_START_ERR	Literal	185	1694
DS$K_TYPE_START_LIST	Literal	185	1707
DS$K_TYPE_SUMMARY	Literal	185	1684
DS$K_TYPE_USER_PROMPT	Literal	185	1672
DS$LOAD	Literal	184	
	ExtRout	733	733c
DS$LOADPCS	Literal	184	
DS$L_SAVED_DSA_FLAGS	Own	431	431= 551= 1259.

Symbol	Type	Defined	Referenced ...
DS$L_SAVED_DS_FLAGS	Own	432	432= 552= 1260.
DS$L_USERCNTRLC	External	292	816=
DS$MAPDBGBLOCK	Literal	184	
DS$MEMSIZE	Literal	184	
DS$MMOFF	Literal	184	
	ExtRout	548	548c
DS$MMON	Literal	184	
DS$PARSE	Literal	184	
DS$PRINTB	Literal	184	
DS$PRINTF	Literal	184	
DS$PRINTS	Literal	184	
DS$PRINTSIG	Literal	184	
DS$PRINTX	Literal	184	
DS$PROBE	Literal	184	
DS$Q_CRD_IMAGE_DESC	Unbound		452 472 486 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
	Own	414	416= 417= 703= 706= 717= 718= 749. 750. 800. 827. 894. 903. 930= 931= 1031= 1032= 1338. 1496.
DS$Q_CRD_VECTOR_DESC	Unbound		453 474 488 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
	Own	402	404= 405= 406= 407= 592. 603= 604= 617. 864. 873. 884. 933= 934= 935= 936= 995. 996. 1012. 1022. 1034= 1035= 1036= 1037=
DS$RELBUF	Literal	184	
DS$SERVICE_DISPATCHER	Literal	184	
DS$SETIPL	Literal	184	
DS$SETMAP	Literal	184	
DS$SETPRIEXV	Literal	184	
DS$SETVEC	Literal	184	
DS$SHOCHAN	Literal	184	
DS$SUMMARY	Literal	184	
DS$V_CTRLC	Macro	Lib01	1245
DS$V_LODFLG	Macro	Lib01	1281
DS$WAITMS	Literal	184	
DS$WAITUS	Literal	184	
DSA$AL_APTMAIL	Bind	186	186
DSA$GL_APTCOM	Bind	186	
DSA$GL_DEVLEN	Bind	186	
DSA$GL_DEVNAM	Bind	186	
DSA$GL_ERRNO	Bind	186	
DSA$GL_EVENT	Bind	186	
DSA$GL_FLAGS	Unbound		470 484 546 623 772 808 829 941 979 1047 1238 1284 1336 1456 1525 1630
	Bind	186	551 561 563 577 579 589 672 741 906 1259 1267 1268 1270 1272 1340 1342 1344 1346 1348 1359 1365 1374 1380 1401 1407 1419 1434 1454
DSA$GL_MSGPTR	Bind	186	
DSA$GL_MSGTYP	Bind	186	
DSA$GL_PASSES	Bind	186	
DSA$GL_PASSNO	Bind	186	
DSA$GL_SECTNO	Bind	186	
DSA$GL_SID	Bind	186	538m
DSA$GL_SUBTNO	Bind	186	
DSA$GL_TESTNO	Bind	186	
DSA$GL_UNITS	Bind	186	

Symbol	Type	Defined	Referenced ...
DSA$V_BINARY	Macro	Lib02	561 1348
DSA$V_CRD_AUTOTEST_OFF	Unbound		464 546 623 772 808 829 941 979 1047 1238
			1284 1336 1456 1525 1630
	Macro	Lib02	577 1272 1346
DSA$V_CRD_MENUTEST_OFF	Unbound		463 546 623 772 808 829 941 979 1047 1238
			1284 1336 1456 1525 1630
	Macro	Lib02	579 1268 1342
DSA$V_CRD_MENUTEST_ON	Unbound		465 546 623 772 808 829 941 979 1047 1238
			1284 1336 1456 1525 1630
	Macro	Lib02	579 1270 1344
DSA$V_CRD_TRACE	Macro	Lib02	563 741 1267 1359 1365 1374 1380 1401 1407 1419
DSA$V_USER	Macro	Lib02	672 906
DSA_FLAGS	Bind	1508	1570.
DSC$A_POINTER	Macro	Lib04	405 604 617 873 884 936 996 1012 1022 1037
DSC$B_CLASS	Macro	Lib04	407 934 1035
DSC$B_DTYPE	Macro	Lib04	406 935 1036
DSC$W_LENGTH	Macro	Lib04	404 592 603 864 933 995 1034
DSR$CHECK_AUTOTEST_OFF_SET	ExtRout	267	1287c 1443c
DSR$CHECK_MENUTEST_OFF_SET	ExtRout	263	1286c
DSR$LOAD_CRD	GlobRout	499	159f
DSR$PRINT_TYPECODE_NAME	GlobRout	1634	179f 1557c
DSR$RESTORE_CRD_VECTORS	GlobRout	957	165f
DSR$UNLOAD_CRD	GlobRout	776	162f 1452c
DSV$EXIT	ExtRout	277	
DSX$PRINT	ExtRout	564	564c 749e 749c 750e 750c 754e 754c 758e 758c 762e
			762c 766e 766c 1360e 1360c 1366e 1366c 1375e 1375c 1381e
			1381c 1402e 1402c 1408e 1408c 1420e 1420c 1422e 1422c 1430e
			1430c 1446e 1446c 1447e 1447c 1448e 1448c 1528e 1528c 1534e
			1534c 1535e 1535c 1536e 1536c 1537e 1537c 1538e 1538c 1539e
			1539c 1540e 1540c 1541e 1541c 1542e 1542c 1543e 1543c 1544e
			1544c 1547e 1551e 1551c 1556e 1556c 1562e 1562c 1566e 1566c
			1570e 1570c 1571e 1571c 1575e 1575c 1576e 1576c 1586e 1586c
			1592e 1592c 1598e 1598c 1608e 1608c 1612e 1612c 1613e 1613c
			1614e 1614c 1615e 1615c 1621e 1621c 1626e 1626c 1670e 1670c
			1671e 1671c 1672e 1672c 1673e 1673c 1674e 1674c 1675e 1675c
			1676e 1676c 1677e 1677c 1678e 1678c 1679e 1679c 1680e 1680c
			1681e 1681c 1682e 1682c 1683e 1683c 1684e 1684c 1685e 1685c
			1686e 1686c 1687e 1687c 1688e 1688c 1689e 1689c 1690e 1690c
			1691e 1691c 1692e 1692c 1693e 1693c 1694e 1694c 1695e 1695c
			1696e 1696c 1697e 1697c 1698e 1698c 1699e 1699c 1700e 1700c
			1701e 1701c 1702e 1702c 1703e 1703c 1704e 1704c 1705e 1705c
			1706e 1706c 1707e 1707c 1710e 1713e 1713c
DS_CLEANUP	ExtRout	251	1275c
DS_DISPATCH_VECTORS	Bind	1158	1203. 1204=
DS_FLAGS	Bind	1509	1571.
ERROR_CODE	Bind	1499	1532. 1547 1551.
ERROR_MESSAGE	Register	1334	1350= 1356= 1362= 1369= 1371= 1377= 1383= 1395= 1404= 1410=
			1429. 1430.
ERROR_TYPE	RoutForm	1304	1353.
	Macro	1519	1534 1535 1536 1537 1538 1539 1540 1541 1542 1543
			1544
ERROR_TYPE_NAME	MacrForm	1519	1520 1522
EXCHANGE	Register	1201	1203= 1206.
EXCHANGE_DISPATCH_VECTORS	GlobRout	1052	156f 769c
EXE$ALONONPAGED	ExtRout	271	

Symbol	Type	Defined	Referenced ...
EXE$CNTREG	ExtRout	260	913c
EXE$DEANONPAGED	ExtRout	274	
FAB	Local	534	636a 639a 650. 651. 652. 655a
FAB$B_BID	Macro	Lib04	636
FAB$B_BKS	Macro	Lib04	636
FAB$B_BLN	Macro	Lib04	636
FAB$B_DNS	Macro	Lib04	636
FAB$B_FAC	Macro	Lib04	636
FAB$B_FNS	Macro	Lib04	636
FAB$B_FSZ	Macro	Lib04	636 652
FAB$B_JOURNAL	Macro	Lib04	636
FAB$B_ORG	Macro	Lib04	636
FAB$B_RAT	Macro	Lib04	636
FAB$B_RFM	Macro	Lib04	636 651
FAB$B_RTV	Macro	Lib04	636
FAB$B_RU_FACILITY	Macro	Lib04	636
FAB$B_SHR	Macro	Lib04	636
FAB$C_BID	Literal	Lib04	636
FAB$C_BLN	Literal	Lib04	534 636
FAB$C_VAR	Literal	Lib04	636
FAB$L_ALQ	Macro	Lib04	636
FAB$L_CTX	Macro	Lib04	636
FAB$L_DNA	Macro	Lib04	636
FAB$L_FNA	Macro	Lib04	636
FAB$L_FOP	Macro	Lib04	636
FAB$L_MRN	Macro	Lib04	636
FAB$L_NAM	Macro	Lib04	636
FAB$L_XAB	Macro	Lib04	636
FAB$M_GET	Literal	Lib04	636
FAB$V_CHAN_MODE	Macro	Lib04	636
FAB$V_FILE_MODE	Macro	Lib04	636
FAB$V_LNM_MODE	Macro	Lib04	636
FAB$W_BLS	Macro	Lib04	636
FAB$W_DEQ	Macro	Lib04	636
FAB$W_GBC	Macro	Lib04	636
FAB$W_MRS	Macro	Lib04	636 650
FALSE	Literal	Lib03	584 947 948 985 986 1252 1253 1351
FILE	Local	528	539m 630= 636a.
GET1ST_	Macro	Lib04	183 185 198 215 227 236 247
GET2ND_	Unbound		183 185 198 215 227 236 247
HALT	Builtin	1236	1291
HS$K_ACTION_ADVANCE_DIAGNOSTIC	Literal	183	
HS$K_ACTION_ADVANCE_GENERAL_COM	Literal	183	
HS$K_ACTION_ADVANCE_STATE	Literal	183	
HS$K_ACTION_CHANGE_TABLE	Literal	183	
HS$K_ACTION_DIAGNOSTIC_ERROR	Literal	183	
HS$K_ACTION_DONE_GENERAL_COMS	Literal	183	
HS$K_ACTION_DO_NOTHING	Literal	183	
HS$K_ACTION_DUMMY_3	Literal	183	
HS$K_ACTION_DUMMY_4	Literal	183	
HS$K_ACTION_DUMMY_5	Literal	183	
HS$K_ACTION_DUMMY_6	Literal	183	
HS$K_ACTION_INIT_HS	Literal	183	
HS$K_ACTION_INTERNAL_ERROR	Literal	183	
HS$K_ACTION_LAST_ACTION	Literal	183	

Symbol	Type	Defined	Referenced ...
HS$K_ACTION_LOAD_ERROR	Literal	183	
HS$K_ACTION_LOAD_GENERAL_COM	Literal	183	
HS$K_ACTION_LOAD_LOAD_COM	Literal	183	
HS$K_ACTION_LOAD_SELECT_COM	Literal	183	
HS$K_ACTION_LOAD_SET_EVENT_COM	Literal	183	
HS$K_ACTION_LOAD_SET_FLAGS_COM	Literal	183	
HS$K_ACTION_LOAD_START_COM	Literal	183	
HS$K_ACTION_PREPARATION_ERROR	Literal	183	
HS$K_ACTION_START_ERROR	Literal	183	
HS$K_STATE_ADVANCE_DIAGNOSTIC	Literal	183	
HS$K_STATE_ADVANCE_GENERAL_COM	Literal	183	
HS$K_STATE_CHANGE_TABLE	Literal	183	
HS$K_STATE_DIAGNOSTIC_ERROR	Literal	183	
HS$K_STATE_DIAGNOSTIC_RUNNING	Literal	183	
HS$K_STATE_DONE_GENERAL_COMS	Literal	183	
HS$K_STATE_DUMMY_1	Literal	183	
HS$K_STATE_DUMMY_2	Literal	183	
HS$K_STATE_DUMMY_3	Literal	183	
HS$K_STATE_DUMMY_4	Literal	183	
HS$K_STATE_DUMMY_6	Literal	183	
HS$K_STATE_INIT_HS	Literal	183	
HS$K_STATE_INTERNAL_ERROR	Literal	183	
HS$K_STATE_LAST_STATE	Literal	183	
HS$K_STATE_LOAD_ERROR	Literal	183	
HS$K_STATE_LOAD_GENERAL_COM	Literal	183	
HS$K_STATE_LOAD_LOAD_COM	Literal	183	
HS$K_STATE_LOAD_SELECT_COM	Literal	183	
HS$K_STATE_LOAD_SET_EVENT_COM	Literal	183	
HS$K_STATE_LOAD_SET_FLAGS_COM	Literal	183	
HS$K_STATE_LOAD_START_COM	Literal	183	
HS$K_STATE_PREPARATION_ERROR	Literal	183	
HS$K_STATE_START_ERROR	Literal	183	
HS_CURRENT_STATE	Bind	1506	1598.
INCOMPATIBILITY_ERROR	Macro	1148	1185 1188
INTERNAL_ERROR	Routine	1463	176f 1393c
JSB_ALLOC	Linkage	Lib01	271
JSB_DEALL	Linkage	Lib01	274
JSB_NONE	Linkage	Lib01	251 263 267 277 1299
K_CORRUPTED	Unbound		1145
	Literal	215	1173 1176 1400
K_CRD_INTERNAL	Literal	215	857 1385
K_DETERMINE_SIZE	Literal	215	663 1364
K_ENDING_ADR	Literal	236	704 714
K_EXPAND_REGION	Literal	215	720 723 1373
K_IMAGE_PTR	Literal	247	417 706 718 749 800 931 1032 1338 1496
K_IMAGE_SIZE	Literal	247	416 703 717 750 827 894 903 930 1031
K_INCOMPATIBLE	Unbound		1149
	Literal	215	1185 1188 1406
K_LAST_ERROR_TYPE	Literal	215	1353
K_LOAD_FILE	Literal	215	640 736 1379
K_MM_IS_ON	Literal	215	549 1355
K_NEGATIVE_VERSION	Literal	227	762 852 1175
K_NOT_CONTRACTED	Literal	215	922 1412
K_NO_DSA_BIT_SET	Literal	215	589 1358
K_NULL_BYTE	Literal	227	766 1172

Symbol	Type	Defined	Referenced ...						
K_NUMBER_OF_VECTORS	Literal	227	611	754	875	1014	1187	1198	
K_POSITIVE_VERSION	Literal	227	758	851	1175	1184			
K_REGION_EXPANDED	Unbound		1145	1149					
	Literal	198	723	736	922	1173	1176	1185	1188
K_REGION_NOT_EXPANDED	Literal	198	549	589	640	663	720		
K_STARTING_ADR	Literal	236	704	706	713	733	747	769	
L$L_ENVIRON	Literal	Lib02	1027	1282					
LENGTH	Bind	1501	1562.						
MAPFREE	ExtRout	254	1424c						
MEMORY_WAS_ALLOCATED	RoutForm	1304	1451.						
MEN$K_HS_STATE_TABLE	Literal	183	1581	1595					
MEN$K_MM_STATE_TABLE	Literal	183	1581	1583					
MEN$K_TQ_STATE_TABLE	Literal	183	1589						
MENU$K_EXIT_AUTOSIZER_ERROR	Literal	183							
MENU$K_EXIT_INTERNAL_ERROR	Literal	183							
MENU$K_EXIT_LAST_REASON	Literal	183							
MENU$K_EXIT_OPERATOR_ABORT	Literal	183							
MENU$K_EXIT_OPERATOR_STOP	Literal	183							
MENU$K_EXIT_SUP_FILE_LOAD_ERR	Literal	183							
MENU$K_EXIT_SUP_FILE_NOT_FOUND	Literal	183							
MM$K_ACTION_ADVANCE_STATE	Literal	183							
MM$K_ACTION_AUTOSIZER_ERROR	Literal	183							
MM$K_ACTION_BUILD_DDBS	Literal	183							
MM$K_ACTION_BUILD_HSTS	Literal	183							
MM$K_ACTION_CHANGE_TABLE	Literal	183							
MM$K_ACTION_DONE_INITS	Literal	183							
MM$K_ACTION_DO_NOTHING	Literal	183							
MM$K_ACTION_DUMMY_3	Literal	183							
MM$K_ACTION_DUMMY_4	Literal	183							
MM$K_ACTION_DUMMY_5	Literal	183							
MM$K_ACTION_DUMMY_6	Literal	183							
MM$K_ACTION_DUMMY_7	Literal	183							
MM$K_ACTION_DUMMY_8	Literal	183							
MM$K_ACTION_INTERNAL_ERROR	Literal	183							
MM$K_ACTION_LAST_ACTION	Literal	183							
MM$K_ACTION_LOAD_AUTOSIZER	Literal	183							
MM$K_ACTION_MENU_PROMPT	Literal	183							
MM$K_ACTION_PRINT_MENU	Literal	183							
MM$K_ACTION_PRINT_MENU_HEADING	Literal	183							
MM$K_ACTION_SET_FLAGS_AUTOSIZER	Literal	183							
MM$K_ACTION_START_AUTOSIZER	Literal	183							
MM$K_STATE_AUTOSIZER_ERROR	Literal	183							
MM$K_STATE_AUTOSIZER_RUNNING	Literal	183							
MM$K_STATE_BUILD_DDBS	Literal	183							
MM$K_STATE_BUILD_HSTS	Literal	183							
MM$K_STATE_CHANGE_TABLE	Literal	183							
MM$K_STATE_DONE_INITS	Literal	183							
MM$K_STATE_DUMMY_1	Literal	183							
MM$K_STATE_DUMMY_2	Literal	183							
MM$K_STATE_DUMMY_3	Literal	183							
MM$K_STATE_DUMMY_5	Literal	183							
MM$K_STATE_DUMMY_7	Literal	183							
MM$K_STATE_DUMMY_8	Literal	183							
MM$K_STATE_INTERNAL_ERROR	Literal	183							
MM$K_STATE_LAST_STATE	Literal	183							

Symbol	Type	Defined	Referenced ...
MM$K_STATE_LOAD_AUTOSIZER	Literal	183	
MM$K_STATE_MENU_PROMPT	Literal	183	
MM$K_STATE_PRINT_MENU	Literal	183	
MM$K_STATE_PRINT_MENU_HEADING	Literal	183	
MM$K_STATE_SET_FLAGS_AUTOSIZER	Literal	183	
MM$K_STATE_START	Literal	183	
MM$K_STATE_START_AUTOSIZER	Literal	183	
MM_CURRENT_STATE	Bind	1504	1586.
MODNAM	Macro	Lib01	327
MODULE_DEBUG	Macro	441	546 623 772 808 829 941 979 1047 1238 1284
			1336 1456 1525 1630
MORE_INFO	RoutForm	1304	1360. 1366. 1368. 1370. 1375. 1381. 1420.
NUL2ND_	Macro	Lib04	183 185 198 215 227 236 247
NUMBER_OF_VECTORS	Register	869	875= 878.
	Register	1009	1014= 1016.
OFF	Literal	Lib03	1245 1267 1268 1270 1272 1281 1342 1344 1346 1348
ON	Literal	Lib03	561
PAGE_COUNT	Register	543	669= 682. 693.
	Register	898	903= 910. 914.
PREVIOUS_STATE	Bind	1507	1608.
RETLEN	Local	530	632= 657a=
RETREC	Local	531	633= 648m 650a= 651a= 652a=
RETURN_NOW	Register	1333	1351= 1397= 1432.
RETURN_TO_CALLER	RoutForm	776	950.
RMS$_FNF	Literal	Lib04	1368
SAVE_DSA_FLAGS	Register	1332	1340= 1434. 1454.
SDL$STARLET_CONCAT	Macro	Lib04	639 655 682 910
SDL$STARLET_OPT	Macro	Lib04	639 655
SDL$STARLET_REQ	Macro	Lib04	639 655 682 910
START_OF_CRD_IMAGE	Bind	1514	
STATE	Register	1493	1578= 1581. 1602= 1602. 1604.
STATUS	Register	544	639= 640. 661. 663. 677= 691= 708. 720. 723. 725=
			735. 736.
	Register	897	907= 913= 919. 922.
SYS$ALLOC	Literal	184	
SYS$ASCTIM	Literal	184	
SYS$ASSIGN	Literal	184	
SYS$BINTIM	Literal	184	
SYS$CANCEL	Literal	184	
SYS$CANTIM	Literal	184	
SYS$CLOSE	Literal	184	
	ExtRout	655	655c
SYS$CLREF	Literal	184	
SYS$CNTREG	ExtRout	910	910c
SYS$CONNECT	Literal	184	
SYS$DALLOC	Literal	184	
SYS$DASSGN	Literal	184	
SYS$DISCONNECT	Literal	184	
SYS$EXPREG	ExtRout	682	682c
SYS$FAO	Literal	184	
SYS$FAOL	Literal	184	
SYS$GET	Literal	184	
SYS$GETCHN	Literal	184	
SYS$GETTIM	Literal	184	
SYS$GTCHAN	Literal	184	

Symbol	Type	Defined	Referenced ...
SYS$HIBER	Literal	184	
SYS$LKWSET	Literal	184	
SYS$NUMTIM	Literal	184	
SYS$OPEN	Literal	184	
	ExtRout	639	639c
SYS$QIO	Literal	184	
SYS$QIOW	Literal	184	
SYS$READ	Literal	184	
SYS$READEF	Literal	184	
SYS$SETAST	Literal	184	
SYS$SETEF	Literal	184	
SYS$SETIMR	Literal	184	
SYS$SETPRI	Literal	184	
SYS$SETPRT	Literal	184	
SYS$SETRWN	Literal	184	
SYS$ULKPAG	Literal	184	
SYS$ULWSET	Literal	184	
SYS$UNWIND	Literal	184	
SYS$WAITFR	Literal	184	
SYS$WFLAND	Literal	184	
SYS$WFLOR	Literal	184	
TEMP_VECTOR	Local	901	910a 915a
TEMP_VECTORS	Bind	617	619=
	Bind	873	875.
	Bind	884	886.
	Bind	1022	1024.
TEMP_VECTOR_STORAGE_LENGTH	Register	803	864= 866.
TQ$K_ACTION_ADVANCE_STATE	Literal	183	
TQ$K_ACTION_CHANGE_TABLE	Literal	183	
TQ$K_ACTION_DONE_TEST_QUEUE	Literal	183	
TQ$K_ACTION_DO_NOTHING	Literal	183	
TQ$K_ACTION_DUMMY_3	Literal	183	
TQ$K_ACTION_DUMMY_4	Literal	183	
TQ$K_ACTION_DUMMY_5	Literal	183	
TQ$K_ACTION_GET_DDB	Literal	183	
TQ$K_ACTION_INIT_TQ	Literal	183	
TQ$K_ACTION_INTERNAL_ERROR	Literal	183	
TQ$K_ACTION_LAST_ACTION	Literal	183	
TQ$K_STATE_CHANGE_TABLE	Literal	183	
TQ$K_STATE_DONE_TEST_QUEUE	Literal	183	
TQ$K_STATE_DUMMY_1	Literal	183	
TQ$K_STATE_DUMMY_2	Literal	183	
TQ$K_STATE_DUMMY_3	Literal	183	
TQ$K_STATE_DUMMY_4	Literal	183	
TQ$K_STATE_DUMMY_5	Literal	183	
TQ$K_STATE_GET_DDB	Literal	183	
TQ$K_STATE_INIT_TQ	Literal	183	
TQ$K_STATE_INTERNAL_ERROR	Literal	183	
TQ$K_STATE_LAST_STATE	Literal	183	
TQ_CURRENT_STATE	Bind	1505	1592.
TRUE	Literal	Lib03 575	583 1290 1397
TYPECODE	Bind	1500	1557.
	RoutForm	1634	1668. 1710 1713.
TYPECODE_CASE	Macro	1660	1670 1671 1672 1673 1674 1675 1676 1677 1678 1679
			1680 1681 1682 1683 1684 1685 1686 1687 1688 1689

Symbol	Type	Defined	Referenced ...								
			1690	1691	1692	1693	1694	1695	1696	1697	1698	1699
			1700	1701	1702	1703	1704	1705	1706	1707		
TYPECODE_NAME	MacrForm	1660	1661	1662								
T_AUTO	Bind	340	578a									
T_AUTO_ABORT_CRD	Bind	353	1446a									
T_AUTO_END_MESSAGE	Bind	354	1447a									
T_AUTO_EXIT_TO_CONSOLE	Bind	355	1448a									
T_CONTRACTION_ERROR	Bind	346	1422a									
T_CORRUPTED	Bind	369	1402a									
T_CORRUPTED_ERROR	Bind	348	1395a	1404a								
T_CRD_IMAGE_DEFAULT	Bind	334	631a	733a								
T_CRD_IMAGE_NAME	Bind	331	630a	733a								
T_CRD_INTERNAL	Bind	364										
T_CRD_INTERNAL_DEBUG	Bind	366										
T_CRD_INTERNAL_HEADER	Bind	365										
T_DETERMINE_SIZE	Bind	358	1366a									
T_EXPAND_REGION	Bind	360	1375a									
T_FILE_NOT_FOUND	Bind	343	1369a									
T_INCOMPATIBILITY_ERROR	Bind	347	1410a									
T_INCOMPATIBLE	Bind	371	1408a									
T_LOAD_FILE	Bind	361	1381a									
T_MENU	Bind	339	582a									
T_MM_ON	Bind	349	1356a									
T_NOT_CONTRACTED	Bind	368	1420a									
T_NO_DSA_BIT_SET	Bind	356	1360a									
T_NULL	Bind	338	588a									
T_TRACE_1	Bind	379	749a									
T_TRACE_2	Bind	380	750a									
T_TRACE_4	Bind	382	754a									
T_TRACE_5	Bind	383	758a									
T_TRACE_6	Bind	384	762a									
T_TRACE_7	Bind	385	766a									
T_TRACE_HEADER	Bind	374	564a									
T_UNKNOWN_ERROR	Bind	344	1362a	1371a	1377a	1383a						
VDS_VERSION_NUMBER	Bind	1516										
VECTOR	Local	1198	1203.	1204.	1206.							
	Local	1623	1624.	1626.								
VECTOR_INDEX	Local	611	619.									
	Local	878	886.									
	Local	1016	1024.									
VECTOR_INFO	Bind	1012	1014.									
VEC_INDEX	Local	938	939.									
	Local	1044	1045.									
XAB	Local	535	636a	637a	642.							
XAB$B_BLN	Macro	Lib04	637									
XAB$B_COD	Macro	Lib04	637									
XAB$C_FHC	Literal	Lib04	637									
XAB$C_FHCLEN	Literal	Lib04	535	637								
XAB$L_EBK	Macro	Lib04	642									
XAB$L_NXT	Macro	Lib04	637									
XAB$W_FFB	Macro	Lib04	642									
XAB_SIZE	Local	525	642=	657.								

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]CRDIMAGE.B32;1
6	Module CRDIMAGE	
143	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
146	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
149	Library #3	DISK$DS:[DS.LOCAL.LIBRARY]CRD.L32;4
152	Library #4	SYS$COMMON:[SYSLIB]LIB.L32;2
1717	Eludom CRDIMAGE	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
@	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

; Library Statistics
;

; File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
; DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	10	1	47	00:00.0
; DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	15	1	96	00:00.0
; DISK$DS:[DS.LOCAL.LIBRARY]CRD.L32;4	491	9	1	63	00:00.0
; SYS$COMMON:[SYSLIB]LIB.L32;2	20450	64	0	1101	00:01.0

; COMMAND QUALIFIERS

```
;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:CRDIMAGE.B32

; Size:         2573 code + 3287 data bytes
; Run Time:        00:16.8
; Elapsed Time:    00:20.7
; Lines/CPU Min:    6128
; Lexemes/CPU-Min:162157
; Memory Used:  542 pages
; Compilation Complete
```

Table of contents

(2) 90 DECLARATIONS
(3) 182 DD_SELECT - Select correct CPUs for this driver
(4) 232 Console TU58 Driver
(4) 487 Character transmission

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CSDDBTDRV.MAR
0000     2 ; *1     6-FEB-1986 15:16:29 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element CSDDBTDRV.MAR
0000     4         .TITLE  CSDDBTDRV - CONSOLE TU58 BOOT DRIVER
0000     5         .IDENT  '06-12'
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  Copyright (c) 1979, 1982, 1983,1985                                     *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    12 ;*                                                                           *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    18 ;*  TRANSFERRED.                                                             *
0000    19 ;*                                                                           *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    22 ;*  CORPORATION.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28 ;
0000    29
0000    30 ;++
0000    31 ; FACILITY:     BOOTS
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       This module contains the bootstrap device driver for the
0000    35 ;       console TU58.
0000    36 ;
0000    37 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    38 ;
0000    39 ; AUTHOR:  Steve Beckhardt,     Creation Date:  1-Nov-1979
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       12      Amy Iorio       June 6, 1985    Version 8.3
0000    44 ;               Branched around halt in do_mapping.
0000    45 ;
0000    46 ;       11      Bob Bergazzi    May 16, 1983    Version 6.11
0000    47 ;               Changed the order of the .LIB statements.
0000    48 ;
0000    49 ;       10      Marion Baggett, 15-Jul-1982, Version 6.9
0000    50 ;               Increased the timeout value so loading of QALOAD for QA-ing
0000    51 ;               can be done on ZAPPA comet.
0000    52 ;
0000    53 ;       09      - Dave Butenhof, 23-Mar-1982, version 6.7
0000    54 ;               Now that TU58 hardware is fixed (Nebula), remove some
0000    55 ;               hack code
0000    56 ;
0000    57 ;       08      - Dave Butenhof, 10-Feb-1982, version 6.6
```

```
0000     58 ;                  Fix driver to work correctly on Nebula.  This is mainly
0000     59 ;                  a matter of allowing some leeway in READY/DONE bit polling
0000     60 ;                  loops to allow console to do something.  Also allow ^C
0000     61 ;                  termination of long TU-58 reads.
0000     62 ;
0000     63 ;          V02-007 KTA0037         Kerbey T. Altmann       26-Oct-1981
0000     64 ;                  Fix start and size to encompass the entire driver.
0000     65 ;
0000     66 ;          02-006  STJ0125         Steven T. Jeffreys      30-Jul-1981
0000     67 ;                  Added support of MRSP protocol.  For an explantion of the
0000     68 ;                  TU58's operation, including RSP and MRSP protocol, please
0000     69 ;                  see the "TU58 Engineering Specification", document number
0000     70 ;                  TU58-0-0.
0000     71 ;
0000     72 ;          02-005  TCM0002         Trudy C. Matthews       29-Jul-1981
0000     73 ;                  Changed all "7ZZ"s to "730"s.
0000     74 ;
0000     75 ;          02-004  TCM0001         Trudy C. Matthews       12-Jun-1981
0000     76 ;                  Altered XMIT_TWO_CHARS routine to work on 11/730 processor,
0000     77 ;                  which requires that the high 3 bytes of the value moved to
0000     78 ;                  PR$_CSTD be zero.  Corrected bug in DO_MAPPING routine.
0000     79 ;                  Added CONSDD_START label to mark start of driver.
0000     80 ;
0000     81 ;          02-003  SRB0001         Steve Beckhardt         10-Jul-1980
0000     82 ;                  Fixed bugs in change CAS0001
0000     83 ;
0000     84 ;          02-02   CAS0001         C.A. Samuelson          30-Apr-1980
0000     85 ;                  Change interface to BOOTDRIVR for UBA pruge of buffered
0000     86 ;                  datapath
0000     87 ;
0000     88 ;--
```

```
0000     90          .SBTTL  DECLARATIONS
0000     91 ;
0000     92 ; INCLUDE FILES:
0000     93 ;
0000     94          .library        /sys$library:lib/       ;                       [11]
0000     95          .library        /$ds/           ;                               [11]
0000     96          .library        /$diag/         ;                               [11]
0000     97
0000     98          DsFDef                          ; Define DS$V_ flags            [08]
0000     99          $BTDDEF                         ; Boot device types
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    100          $IODEF                          ; I/O function codes
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    101          $PRDEF                          ; Processor registers
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    102          $PTEDEF                         ; PTE definitions
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    103          $RPBDEF                         ; RPB offsets
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    104          $SSDEF                          ; Status codes
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    105          $VADEF                          ; Virtual address fields
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    106
0000    107 ;
0000    108 ; MACROS:
0000    109 ;
0000    110
0000    111 ;
0000    112 ; EQUATED SYMBOLS:
0000    113 ;
0000    114
0000    115 ;
0000    116 ; CONSOLE TU58 REGISTER DEFINITIONS
0000    117 ;
0000    118
0000    119          $DEFINI DD                      ;START OF REGISTER DEFINITIONS
0000                 .SAVE   LOCAL_BLOCK
0000    120
0000    121          _VIELD  CSRS,0,<-               ;START OF PROC. REG. CSRS DEFS.
0000    122                  <,6>,-                  ; UNUSED
0000    123                  <IE,,M>,-               ; INTERRUPT ENABLE
0000    124                  <DONE,,M>,-             ; DONE
0000    125          >
0000    126
0000    127          _VIELD  CSRD,0,<-               ;START OF PROC. REG. CSRD DEFS.
0000    128                  <DATA,8>,-              ; DATA
0000    129                  <,7>,-                  ; UNUSED
0000    130                  <ERROR,,M>,-            ; ERROR
0000    131          >
```

```
                        0000   132
                        0000   133          _VIELD  CSTS,0,<-                ;START OF PROC. REG. CSTS DEFS.
                        0000   134                  <BREAK,,M>,-               ; BREAK
                        0000   135                  <,5>,-                     ; UNUSED
                        0000   136                  <IE,,M>,-                  ; INTERRUPT ENABLE
                        0000   137                  <READY,,M>,-               ; READY
                        0000   138          >
                        0000   139
                        0000   140          _VIELD  CSTD,0,<-                ;START OF PROC. REG. CSTD DEFS.
                        0000   141                  <DATA,8>,-                 ; DATA
                        0000   142          >
                        0000   143
                        0000   144          ;
                        0000   145          ; PROTOCOL FLAGS
                        0000   146          ;
                        0000   147
                00000001  0000   148 DD_DATA = 1
                00000002  0000   149 DD_CNTRL = 2
                00000004  0000   150 DD_INIT = 4
                00000010  0000   151 DD_CONTINUE = 16
                        0000   152
                        0000   153          ;
                        0000   154          ; FUNCTION CODES
                        0000   155          ;
                        0000   156
                00000001  0000   157 DD_Init_Cmd = 1
                00000002  0000   158 DD_READ = 2
                00000003  0000   159 DD_WRITE = 3
                00000040  0000   160 DD_ENDPKT = 64
                        0000   161
                        0000   162          ;
                        0000   163          ; SWITCH BITS
                        0000   164          ;
                        0000   165
                00000003  0000   166 MRSP_SWITCH = 3
                        0000   167
                        0000   168          $DEFEND DD
                        0000                .RESTORE
                        0000   169
                        0000   170 ;
                        0000   171 ; Boot driver table entry
                        0000   172 ;
                        0000   173
                        0000   174          $BOOT_DRIVER    DEVTYPE = BTD$K_CONSOLE,-  ; Device type (console)
                        0000   175                          ACTION = DD_SELECT,-       ; Action routine
                        0000   176                          SIZE = CONSDD_DRVSIZ,-     ; Driver size
                        0000   177                          ADDR = CONSDD_START,-     ; Driver start
                        0000   178                          ENTRY = CONSDD_DRIVER,-    ; Driver entry
                        0000   179                          DRIVRNAME = DDNAME         ; Driver file name
                    00000000            .PSECT  BOOTDRIVR_4
                FFFF    0000            .WORD   -1
                0040    0002            .WORD   BTD$K_CONSOLE
            00000000'   0004            .LONG   DD_SELECT-$TABLE
            00000271'   0008            .LONG   CONSDD_DRVSIZ
            00000000'   000C            .LONG   CONSDD_START-$TABLE
            00000033'   0010            .LONG   CONSDD_DRIVER-CONSDD_START
            00000026'   0014            .LONG   DDNAME-CONSDD_START
```

```
                        00000000                .PSECT  BOOTDRIVR_2
                            0000    180
```

```
                        0000   182          .SBTTL  DD_SELECT - Select correct CPUs for this driver
                        0000   183
                        0000   184 ;++
                        0000   185 ; FUNCTIONAL DESCRIPTION:
                        0000   186 ;
                        0000   187 ;       This routine is an action routine called by the boot driver
                        0000   188 ;       select code to determine if this is a cpu that has a TU58
                        0000   189 ;       for a console.
                        0000   190 ;
                        0000   191 ; CALLING SEQUENCE:
                        0000   192 ;
                        0000   193 ;       JSB     DD_SELECT
                        0000   194 ;
                        0000   195 ; INPUT PARAMETERS:
                        0000   196 ;
                        0000   197 ;       EXE$GB_CPUTYPE          Global variable containing cpu type
                        0000   198 ;
                        0000   199 ; OUTPUT PARAMETERS:
                        0000   200 ;
                        0000   201 ;       R0      0 = Do not use this driver
                        0000   202 ;               1 = Use this driver
                        0000   203 ;
                        0000   204 ;       MRSP    1 if this CPU has a TU58 that speaks MRSP, 0 if not.
                        0000   205 ;
                        0000   206 ;               It is assumed that certain CPUs must have MRSP TU58s.
                        0000   207 ;               Currently, only the 11/730 must have an MRSP TU58.
                        0000   208 ;               The 11/780 and 11/730 may have MRSP TU58s, but it is
                        0000   209 ;               not essential that MRSP be used.
                        0000   210 ;
                        0000   211 ;--
                        0000   212
                        0000   213 CONSDD_START:                            ; Label to mark start of driver
                        0000   214
                        0000   215 DD_SELECT:
00000025'GF   94        0000   216         CLRB    G^MRSP                  ; Assume TU58 does not speak MRSP
           50   D4      0006   217         CLRL    R0                      ; Initialize R0
                        0008   218         CPUDISP <CPU_780,-              ; 11/780
                        0008   219                  CPU_750,-              ; 11/750
                        0008   220                  CPU_730,-              ; 11/730
                        0008   221                 >
02'  01    00000000'EF   8F   0008         CASEB   EXE$GB_CPUTYPE,#PR$_SID_TYP780,S^#<<30001$-30000$>/2>-
                        0010       30000$:
                  0006' 0010             .SIGNED_WORD    CPU_780-30000$
                  000D' 0012             .SIGNED_WORD    CPU_750-30000$
                  0007' 0014             .SIGNED_WORD    CPU_730-30000$
                        0016       30001$:
                        0016   222 CPU_780:
                     05 0016   223         RSB                             ; Do not use this driver
                        0017   224
                        0017   225 CPU_730:
00000025'GF   D6        0017   226         INCL    G^MRSP                  ; Use MRSP (and fall through to common code)
                        001D   227
                        001D   228 CPU_750:
           50   D6      001D   229         INCL    R0                      ; Use this driver
                     05 001F   230         RSB
```

```
                        0020   232          .SBTTL  Console TU58 Driver
                        0020   233
                        0020   234 ;++
                        0020   235 ;
                        0020   236 ; Inputs:
                        0020   237 ;
                        0020   238 ;       R1      Address of page table for virtual -> physical mapping
                        0020   239 ;       R2      Base VPN of transfer (Bits 29:9 of R10)
                        0020   240 ;       R5      LBN for current piece of transfer
                        0020   241 ;       R8      Size of transfer in bytes
                        0020   242 ;       R9      Address of the RPB
                        0020   243 ;       R10     Starting address of transfer
                        0020   244 ;
                        0020   245 ;       FUNC(AP)  I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                        0020   246 ;       MODE(AP)  Address interpretation mode:
                        0020   247 ;                    0 -> Physical,  1 -> Virtual
                        0020   248 ;
                        0020   249 ; Outputs:
                        0020   250 ;
                        0020   251 ;       R0      Status code:
                        0020   252 ;
                        0020   253 ;               SS$_NORMAL      Successful transfer
                        0020   254 ;               SS$_BUFBYTALI   Odd byte count
                        0020   255 ;               SS$_CTRLERR     Fatal controller error
                        0020   256 ;--
                00000010 0020   257 FUNC = 16
                00000014 0020   258 MODE = 20
                        0020   259
                        0020   260
                        0020   261 ;
                        0020   262 ; OWN STORAGE:
                        0020   263 ;
                        0020   264
                00000000 0020   265 STKPTR: .LONG   0                       ; Saved stack pointer
                      00 0024   266 INIT:   .BYTE   0                       ; TU58 initialized flag
                      00 0025   267 MRSP:   .BYTE   0                       ; MRSP protocol flag
58 45 2E 52 45 56 49 52 44 44 44 00' 0026   268 DDNAME: .ASCIC  /DDDRIVER.EXE/          ; Driver file name
                      45 0032
                      0C 0026
                        0033   269
                        0033   270 CONSDD_DRIVER:
                0D06 8F   BB    0033   271         PUSHR   #^M<R1,R2,R8,R10,R11>   ; Save input registers
                  0A 58   E9    0037   272         BLBC    R8,1$                   ; Branch if even byte count
             50   030C 8F   3C    003A   273         MOVZWL  #SS$_BUFBYTALI,R0       ; Error - odd byte count
                0D06 8F   BA    003F   274         POPR    #^M<R1,R2,R8,R10,R11>   ; Restore registers
                      05    0043   275         RSB
               D8 AF   5E   D0    0044   276 1$:     MOVL    SP,STKPTR               ; Save stack pointer
                        0048   277
                        0048   278 ;
                        0048   279 ; There are 4 possibilities concerning mapping:  the I/O can be
                        0048   280 ; done virtual or physical (MODE(AP)) and we can be executing virtual
                        0048   281 ; or physical (contents of processor register PR$_MAPEN).  If both
                        0048   282 ; modes match, then we can just copy data to/from the user buffer.
                        0048   283 ; If the I/O is to be done virtual and we are executing physical
                        0048   284 ; then the buffer address has to be translated using the page table
                        0048   285 ; pointed to by R1.  If the I/O is to be done physical and we are
                        0048   286 ; executing virtual then we have to double map the buffer using a spare
```

```
                        0048   287 ; PTE.  At this point, we just compute a mapping switch in R11 as follows:
                        0048   288 ;
                        0048   289 ;       0       Both modes match, just copy the data
                        0048   290 ;       1       Do virtual -> physical translation using page table
                        0048   291 ;       -1      Do physical -> virtual mapping using a spare PTE
                        0048   292 ;
            5B   38 DB  0048   293         MFPR    #PR$_MAPEN,R11          ; Get mapping enabled switch
            5B   5B CE  004B   294         MNEGL   R11,R11                 ; Negate it
         5B   14 AC C0  004E   295         ADDL    MODE(AP),R11            ; Add I/O mode switch
                        0052   296
            56   51 7D  0052   297         MOVQ    R1,R6                   ; R6 = Addr. pf page tbl, R7 = Base VPN
               0201 30  0055   298         BSBW    DO_MAPPING              ; Initialize mapping if required
                        0058   299
              CA AF 95  0058   300         TstB    Mrsp                    ; If Nebula, turn off clock
                 07 13  005B   301         BEql    3$                      ; Branch if not Nebula
      18 80000080 8F DA  005D   302         Mtpr    #<1a31>!<1a7>, #Pr$_Iccs; Turn off clock
              BD AF 95  0064   303 3$:      TSTB    INIT                    ; Is it necessary to initialize TU58?
                 03 13  0067   304         BEql    5$                      ; Yes
               0021 31  0069   305         BrW     10$                     ; No
                        006C   306 5$:
                        006C   307         ;
                        006C   308         ; Initialize TU58.  This code is executed the first time the
                        006C   309         ; driver is called and the next time the driver is called
                        006C   310         ; after exiting with an error.
                        006C   311         ;
                        006C   312
            1E   01 DA  006C   313         MTPR    #CSTS_M_BREAK,#PR$_CSTS ; Set BREAK bit in trans. csr
                 52 D4  006F   314         CLRL    R2                      ; R2 contains null characters
               015D 30  0071   315         BSBW    XMIT_TWO_CHARS          ; Send two nulls
            1E   00 DA  0074   316         MTPR    #0,#PR$_CSTS            ; Clear BREAK bit
            50   1D DB  0077   317         MFPR    #PR$_CSRD,R0            ; Clear receive buffer
      52    0404 8F 3C  007A   318         MOVZWL  #DD_INIT@8+DD_INIT,R2   ; R2 contains two INIT characters
               014F 30  007F   319         BSBW    XMIT_TWO_CHARS          ; Send two INIT characters
               0198 30  0082   320         BSBW    RECV_ONE_CHAR           ; Receive a character
            10   52 91  0085   321         CMPB    R2,#DD_CONTINUE         ; Is it continue?
                 4D 12  0088   322         BNeq    32$                     ; No, error
              97 AF 96  008A   323         INCB    INIT                    ; So that we don't execute this code again
                        008D   324
                        008D   325 ;
                        008D   326 ; Perform the I/O transfer.  Register usage is:
                        008D   327 ;
                        008D   328 ;       R0              Scratch
                        008D   329 ;       R1              Loop counter
                        008D   330 ;       R2              Character to/from TU58
                        008D   331 ;       R3              Character received from TU58
                        008D   332 ;       R4              Checksum
                        008D   333 ;       R5              Logical block number
                        008D   334 ;       R6              Address of page table
                        008D   335 ;       R7              Virtual page number of buffer
                        008D   336 ;       R8              Size of remaining buffer (in bytes)
                        008D   337 ;       R9              Address of RPB
                        008D   338 ;       R10             Address of current spot in buffer
                        008D   339 ;       R11             Mapping switch
                        008D   340 ;
                        008D   341 ;
                        008D   342         ASSUME  DD_WRITE  EQ  DD_READ+1
                        008D   343         ASSUME  DD_DATA  EQ  1
```

```
                          008D
                 54   D4   008D   10$:   CLRL    R4                      ; Clear checksum
        52   0A02 8F   3C   008F          MOVZWL  #10a8+DD_CNTRL,R2       ; Command packet and byte count
               0134   30   0094          BSBW    XMIT_TWO_UPDSUM         ; Send them
            52   02   9A   0097          MOVZBL  #DD_READ,R2             ; Assume read command
         21   10 AC   D1   009A          CMPL    FUNC(AP),#IO$_READLBLK  ; Is it a read?
                 02   13   009E          BEQL    20$                     ; Yes
                 52   D6   00A0          INCL    R2                      ; No, convert to write command
               0126   30   00A2   20$:   BSBW    XMIT_TWO_UPDSUM         ; Send command and modifier (0)
                          00A5   ;
                          00A5   ;       The unit number of the console boot device is always zero.
                          00A5   ;       This is a result of the 11/750 console implementation, where
                          00A5   ;       logical device CSA1: is actually physical device DDA0:.
                          00A5   ;
                 52   D4   00A5          CLRL    R2                      ; Set Unit # and Switch fields
            FF7A CF   95   00A7          TSTB    MRSP                    ; Does this TU58 speak MRSP?
                 04   13   00AB          BEQL    25$                     ; Branch if not
         00 52   0B   E2   00AD          BBSS    #<MRSP_SWITCH+8>,R2,25$ ; Set MRSP switch
               0117   30   00B1   25$:   BSBW    XMIT_TWO_UPDSUM         ; Send unit # and switches (0)
                 52   D4   00B4          CLRL    R2                      ; Set sequence number (unused field)
               0112   30   00B6          BSBW    XMIT_TWO_UPDSUM         ; Send sequence number (0)
            52   58   3C   00B9          MOVZWL  R8,R2                   ; Byte count
               010C   30   00BC          BSBW    XMIT_TWO_UPDSUM         ; Send it
            52   55   3C   00BF          MOVZWL  R5,R2                   ; Block number
               0106   30   00C2          BSBW    XMIT_TWO_UPDSUM         ; Send it
            52   54   3C   00C5          MOVZWL  R4,R2                   ; Checksum
               0106   30   00C8          BSBW    XMIT_TWO_CHARS          ; Send it
         21   10 AC   D1   00CB          CMPL    FUNC(AP),#IO$_READLBLK  ; Is it a read command?
                 42   13   00CF          BEQL    50$
                          00D1
                          00D1   ;
                          00D1   ; Do a write to the TU58
                          00D1   ;
               0149   30   00D1   30$:   BSBW    RECV_ONE_CHAR           ; Receive one character
            10   52   91   00D4          CMPB    R2,#DD_CONTINUE         ; Is it CONTINUE?
                 42   12   00D7   32$:   BNEQ    52$                     ; No, error
                 54   D4   00D9          CLRL    R4                      ; Clear checksum
         51   80 8F   9A   00DB          MOVZBL  #128,R1                 ; Number of bytes to send
            58   51   D1   00DF          CMPL    R1,R8                   ; Is it less than remaining byte count?
                 03   1F   00E2          BLSSU   35$                     ; Yes
            51   58   D0   00E4          MOVL    R8,R1                   ; No, use remaining byte count instead
        52   51   08   78   00E7   35$:   ASHL    #8,R1,R2                ; Put byte count in second byte
                 52   D6   00EB          INCL    R2                      ; Flag byte = DD_DATA
               00DB   30   00ED          BSBW    XMIT_TWO_UPDSUM         ; Send flag byte and byte count
            51   02   C6   00F0          DIVL    #2,R1                   ; Convert byte count to word count
               0144   30   00F3   40$:   BSBW    GETBYTE                 ; Get a byte from memory in R3
            52   53   D0   00F6          MOVL    R3,R2                   ; Save byte in R2
               013E   30   00F9          BSBW    GETBYTE                 ; Get another byte from memory in R3
    52   08   08   53   F0   00FC          INSV    R3,#8,#8,R2             ; Put it in second byte of R2
               00C7   30   0101          BSBW    XMIT_TWO_UPDSUM         ; Send two bytes
              EC 51   F5   0104          SOBGTR  R1,40$                  ; Repeat until byte count is 0
            52   54   3C   0107          MOVZWL  R4,R2                   ; Get checksum
               00C4   30   010A          BSBW    XMIT_TWO_CHARS          ; Send it
                 58   D5   010D          TSTL    R8                      ; Any more data to send?
                 C0   12   010F          BNEQ    30$                     ; Yes, send another packet
                 30   11   0111          BRB     END_OF_DATA             ; No, get End packet
                          0113
```

```
                        0113   401 ;
                        0113   402 ; Do a read from TU58
                        0113   403 ;
                    54  D4  0113   404 50$:    CLRL    R4                      ; Clear checksum
                  00F2  30  0115   405         BSBW    RECV_TWO_UPDSUM         ; Get flag byte in R3, byte count in R2
                01  53  91  0118   406         CMPB    R3,#DD_DATA             ; Is this a data packet?
                  2E  12  011B   407 52$:    BNEQ    DD_ERROR2               ; No, error
              51  52  02  C7  011D   408         DIVL3   #2,R2,R1                ; Convert byte count to word count in R1
                  00E6  30  0121   409 55$:    BSBW    RECV_TWO_UPDSUM         ; Receive next two data bytes in R3, R2
                  011E  30  0124   410         BSBW    PUTBYTE                 ; Store first byte in memory
                53  52  9A  0127   411         MOVZBL  R2,R3                   ; Move second byte to R3
                  0118  30  012A   412         BSBW    PUTBYTE                 ; Store second byte in memory
                 F1 51  F5  012D   413         SOBGTR  R1,55$                  ; Repeat until byte count is 0
                  00E5  30  0130   414         BSBW    RECV_TWO_CHARS          ; Get checksum in R3, R2
          53  08  08  52  F0  0133   415         INSV    R2,#8,#8,R3             ; Assemble checksum into one word
                54  53  B1  0138   416         CMPW    R3,R4                   ; Is checksum correct?
                  79  12  013B   417         BNEQ    DD_ERROR                ; No, error
                  3B  10  013D   418         BsbB    DD_CheckControlC        ; Check for control C                   [08]
                  58  D5  013F   419         TSTL    R8                      ; Anymore data to receive?
                  D0  12  0141   420         BNEQ    50$                     ; Yes, receive next data packet
                      0143   421
                      0143   422 ;
                      0143   423 ; We've sent or received all the data.  Make sure we receive an end
                      0143   424 ; packet with a success code.
                      0143   425 ;
                      0143   426 END_OF_DATA:
                    54  D4  0143   427         CLRL    R4                      ; Clear checksum
                  00C2  30  0145   428         BSBW    RECV_TWO_UPDSUM         ; Get flag byte and byte count
                02  53  91  0148   429         CMPB    R3,#DD_CNTRL            ; Is it a command packet?
                      014B   430 DD_ERROR2:                                    ; (extension for branch to error)
                    69  12  014B   431         BNEQ    DD_ERROR                ; No, error
                  00BA  30  014D   432         BSBW    RECV_TWO_UPDSUM         ; Get opcode and success/failure byte
             40 8F  53  91  0150   433         CMPB    R3,#DD_ENDPKT           ; Is it an end packet?
                    60  12  0154   434         BNEQ    DD_ERROR                ; No, error
                    52  D5  0156   435         TSTL    R2                      ; Is it success?
                    5C  19  0158   436         BLSS    DD_ERROR                ; No, error
                51  04  D0  015A   437         MOVL    #4,R1                   ; Read remainder of packet
                  00AA  30  015D   438 10$:    BSBW    RECV_TWO_UPDSUM         ;
                 FA 51  F5  0160   439         SOBGTR  R1,10$                  ;
                  00B2  30  0163   440         BSBW    RECV_TWO_CHARS          ; Read checksum in R3, R2
          53  08  08  52  F0  0166   441         INSV    R2,#8,#8,R3             ; Form checksum in R3
                54  53  B1  016B   442         CMPW    R3,R4                   ; Is checksum correct?
                    46  12  016E   443         BNEQ    DD_ERROR                ; No, error
                    2C  10  0170   444         BsbB    Reset_Clock             ;
                50  01  3C  0172   445         MOVZWL  #SS$_NORMAL,R0          ; Return success                        [08]
              0D06  8F  BA  0175   446         POPR    #^M<R1,R2,R8,R10,R11>   ; Restore registers
                        05  0179   447         RSB
                        017A   448
                        017A   449 DD_CheckControlC:                            ; Check for control C                   [08]
         00000000'EF    16  017A   450         Jsb     L^Kb_Poll               ; Poll keyboard                         [08]
   00000000'EF  00  E1  0180   451         BbC     S^#Ds$V_CtrlC, -            ; If ^C bit                             [08]
                15      0187   452                  L^Ds$GL_Flags, 20$         ; .. isn't set, continue                [08]
              FE98  CF  94  0188   453         ClrB    Init                    ; Make sure to re-init Tu58 later       [08]
                    10  10  018C   454         BsbB    Reset_Clock             ;
          5E  FE8E  CF  D0  018E   455         MOVL    STKPTR,SP               ; Restore stack pointer                 [08]
              0D06  8F  BA  0193   456         POPR    #^M<R1,R2,R8,R10,R11>   ; Restore registers                     [08]
          50  0651  8F  3C  0197   457         MovZWL  #SS$_ControlC, R0       ; Set status                            [08]
```

```
                    04  019C          Ret                                  ; Return from BOO$QIO, to prevent        [08]
                        019D                                               ;  retry of the operation.               [08]
                    05  019D  20$:    Rsb                                  ; No ^C, just return                     [08]
                        019E
                        019E  Reset_Clock:
        FE83 CF     95  019E          TstB    Mrsp
             0A     13  01A2          BEql    10$
             3F     BB  01A4          PushR   #^M<R0,R1,R2,R3,R4,R5>
   00000000'EF      16  01A6          Jsb     L^Clk$Re_Init
             3F     BA  01AC          PopR    #^M<R0,R1,R2,R3,R4,R5>
                    05  01AE  10$:    Rsb
                        01AF
                        01AF  .Enable LSB                                  ;                                        [08]
                        01AF
                        01AF  DD_TimeOut:                                  ; Operation timed out                    [08]
   50   022C 8F     3C  01AF          MovZWL  #SS$_TimeOut, R0             ; Set timeout code                       [08]
             05     11  01B4          BrB     10$                          ; Join common                            [08]
                        01B6
                        01B6  DD_ERROR:
   50   0054 8F     3C  01B6          MOVZWL  #SS$_CTRLERR,R0              ; Set failure status                     [08]
                        01BB
        FE65 CF     94  01BB  10$:    CLRB    INIT                         ; Initialize TU58 on next entry          [08]
             DD     10  01BF          BsbB    Reset_Clock                  ;
   5E   FE5B CF     D0  01C1          MOVL    STKPTR,SP                    ; Restore stack pointer
        0D06 8F     BA  01C6          POPR    #^M<R1,R2,R8,R10,R11>        ; Restore registers
                    05  01CA          RSB                                  ; Return and retry
                        01CB
                        01CB  .DisAble LSB                                 ;                                        [08]
```

```
                                01CB    487 .Subtitle       Character transmission
                                01CB    488
                                01CB    489 ;++
                                01CB    490 ; XMIT_TWO_UPDSUM - Transmit two characters and update checksum
                                01CB    491 ; XMIT_TWO_CHARS - Transmit two characters
                                01CB    492 ; XMIT_ONE_CHAR - Transmit one character
                                01CB    493 ;
                                01CB    494 ; Inputs:
                                01CB    495 ;       R2      Contains one or two characters in low bytes
                                01CB    496 ;       R4      Checksum so far
                                01CB    497 ;
                                01CB    498 ; Outputs:
                                01CB    499 ;       R4      Updated checksum
                                01CB    500 ;--
                                01CB    501
                                01CB    502 XMIT_TWO_UPDSUM:
                    54   52   A0  01CB    503         ADDW    R2,R4                   ; Add two characters to checksum
                    54   00   D8  01CE    504         ADWC    #0,R4                   ; Add carry
                                01D1    505
                                01D1    506 XMIT_TWO_CHARS:
                         52   DD  01D1    507         PUSHL   R2                      ; Save input on stack.
                    52   52   9A  01D3    508         MOVZBL  R2,R2                   ; Zero high bytes of data.
                         05   10  01D6    509         BSBB    XMIT_ONE_CHAR           ; Transmit one character
                52   8E   F8 8F   78  01D8    510         ASHL    #-8,(SP)+,R2            ; Retrieve second character
                                01DD    511                                         ; and fall through to transmit it
                                01DD    512
                                01DD    513 XMIT_ONE_CHAR:
                    50   1E   9A  01DD    514         MovZBL  #Pr$_CsTs, R0           ; Number of register to poll        [08]
                         06   10  01E0    515         BsbB    Wait_Ready              ; Wait for bit 7 to set             [08]
                    1F   52   DA  01E2    516         MTPR    R2,#PR$_CSTD            ; Transmit character
                         05   01E5    517         RSB
                                01E6    518
                                01E6    519 DD_Error1:                              ; Extension to DD_Error             [08]
                         CE   11  01E6    520         BrB     DD_Error                ; Just chain on to it               [08]
                                01E8    521
                                01E8    522 ;++
                                01E8    523 ; Wait_Ready - wait for done/ready bit in specified IPR to set
                                01E8    524 ;
                                01E8    525 ; Inputs:
                                01E8    526 ;       R0      Number of processor register to poll
                                01E8    527 ;
                                01E8    528 ; Outputs:
                                01E8    529 ;       None
                                01E8    530 ;
                                01E8    531 ; If desired bit does not set within the timeout period, a direct
                                01E8    532 ; exit is made from the driver through DD_TIMEOUT.
                                01E8    533 ;
                                01E8    534 ;--
                                01E8    535
                                01E8    536 Wait_Ready:                                 ; Wait for bit              [08]
                         06   BB  01E8    537         PushR   #^M<R1, R2>             ; Save R1 for loop counter  [08]
                51   001F0000 8F   D0  01EA    538         MovL    #^X1F0000, R1           ; Fetch value               [10]
01'01'01'01'01'01'01'01'01'01'  01F1    539 10$:    .Byte   1[10]                   ; Nops to free console      [08]
                         52   DB  01FB    540         Mfpr    R0, R2                  ; Read console CSR          [08]
                    05   52   07  E0  01FE    541         BbS     #CsRs_V_Done, R2, 20$   ; Exit loop if ready        [08]
                         EC   51  F5  0202    542         SobGtr  R1, 10$                 ; Otherwise, loop           [08]
                         A8   11  0205    543         BrB     DD_TimeOut              ; Timeout if drop here      [08]
```

```
                    06   BA  0207   20$:    PopR    #^M<R1, R2>                    ; Restore register          [08]
                         05  0209          Rsb                                    ; Return to caller          [08]
                             020A   ;++
                             020A   ; RECV_TWO_UPDSUM - Receive two characters and update checksum
                             020A   ; RECV_TWO_CHARS - Receive two characters
                             020A   ; RECV_ONE_CHAR - Receive one character
                             020A   ;
                             020A   ; Inputs:
                             020A   ;       R4      Checksum so far
                             020A   ;
                             020A   ; Outputs:
                             020A   ;       R2      Most recently received character
                             020A   ;       R3      Previous character in low byte
                             020A   ;--
                             020A
                             020A   RECV_TWO_UPDSUM:
                    0C   10  020A          BSBB    RECV_TWO_CHARS          ; Receive two characters in R3,  R2
     53   08   08   52   F0  020C          INSV    R2,#8,#8,R3             ; Put second character into R3
               54   53   A0  0211          ADDW    R3,R4                   ; Add two characters to checksum
               54   00   D8  0214          ADWC    #0,R4                   ; Add carry to checksum
                         05  0217          RSB
                             0218
                             0218   RECV_TWO_CHARS:
                    03   10  0218          BSBB    RECV_ONE_CHAR           ; Get one character in R2
               53   52   9A  021A          MOVZBL  R2,R3                   ; Save first character in R3 and
                             021D                                          ; fall through to get second character
                             021D
                             021D   RECV_ONE_CHAR:
               50   1C   9A  021D          MovZBL  #Pr$_CsRs, R0                  ; Get number of register    [08]
                    C6   10  0220          BsbB    Wait_Ready                     ; Wait for ready to set     [08]
               52   1D   DB  0222          MFPR    #PR$_CSRD,R2                   ; Get next character
               FDFC CF   95  0225          TSTB    MRSP                           ; Does this TU58 speak MRSP?
                    0A   13  0229          BEQL    20$                            ; Branch if not
               50   1E   DB  022B   10$:    MFPR    #PR$_CSTS,R0                   ; Get transmit status
          F9   50   07   E1  022E          BBC     #CSTS_V_READY,R0,10$           ; Loop until ready
               1F   10   DA  0232          MTPR    #DD_CONTINUE,#PR$_CSTD         ; Send CONTINUE character
          AD   52   0F   E0  0235   20$:    BBS     #CSRD_V_ERROR,R2,DD_ERROR1     ; Branch if data overrun
                         05  0239          RSB
                             023A
                             023A
                             023A   ;++
                             023A   ; GETBYTE - Subroutine to get a byte from memory
                             023A   ; PUTBYTE - Subroutine to store a byte in memory
                             023A   ;
                             023A   ;       These two subroutines do two things special:
                             023A   ;
                             023A   ;               1)  Since the floppy always reads or writes 128 bytes
                             023A   ;                   these routines simply return if the byte count is zero.
                             023A   ;               2)  These routines take care of page boundaries if
                             023A   ;                   mapping is required.
                             023A   ;
                             023A   ; Inputs:
                             023A   ;       R3      Byte to store (PUTBYTE)
                             023A   ;       R6      Address of page table
                             023A   ;       R7      Virtual page number of buffer
                             023A   ;       R8      Size of remaining buffer (in bytes)
                             023A   ;       R10     Address of current spot in buffer
```

```
                                023A   ;       R11     Mapping switch:
                                023A   ;                       -1   Do physical -> virtual map
                                023A   ;                       0    No mapping required
                                023A   ;                       1    Do virtual -> physical translation
                                023A   ;
                                023A   ; Outputs:
                                023A   ;       R3      Byte fetched from memory (GETBYTE)
                                023A   ;--
                                023A
                                023A           .ENABL  LSB
                                023A
                                023A   GETBYTE:
                    53    D4    023A           CLRL    R3                      ; Return 0 if byte count = 0
                    58    D5    023C           TSTL    R8                      ; Is byte count 0?
                    2F    13    023E           BEQL    90$                     ; Yes
              53    8A    9A    0240           MOVZBL  (R10)+,R3               ; Get byte
                    07    11    0243           BRB     10$                     ; Branch to common code
                                0245
                                0245
                                0245   PUTBYTE:
                    58    D5    0245           TSTL    R8                      ; Is byte count 0?
                    26    13    0247           BEQL    90$                     ; Yes
              8A    53    90    0249           MOVB    R3,(R10)+               ; Store byte
                                024C
                                024C
                    58    D7    024C   10$:    DECL    R8                      ; Decr. byte count
                    1F    13    024E           BEQL    90$                     ; Reached zero
        5A    01FF  8F    B3    0250           BITW    #VA$M_BYTE,R10          ; Did address overflow onto new page?
                    18    12    0255           BNEQ    90$                     ; No
                    57    D6    0257           INCL    R7                      ; Yes, increment page number
                                0259
                                0259   ;
                                0259   ; Fall through to ...
                                0259   ;
                                0259
                                0259
                                0259   ;++
                                0259   ; DO_MAPPING - Subroutine to perform necessary mapping
                                0259   ;
                                0259   ; Inputs:
                                0259   ;       R6      Address of page table
                                0259   ;       R7      Page number of buffer
                                0259   ;       R10     Address to map
                                0259   ;       R11     Mapping switch:
                                0259   ;                       -1      Do physical -> virtual map
                                0259   ;                       0       No mapping required
                                0259   ;                       1       Do virtual -> physical translation
                                0259   ;
                                0259   ; Outputs:
                                0259   ;       R10     Address to use
                                0259   ;--
                                0259
                                0259   DO_MAPPING:
                    5B    D5    0259           TSTL    R11                     ; Any mapping required?
                    12    11    025B           BRB     90$                     ; No    [changed from BEQL-- 12]
                    11    19    025D           BLSS    100$                    ; Yes, map physical to virtual
    5A    FFFFFE00  8F    CA    025F           BICL    #^C<VA$M_BYTE>,R10      ; Yes, translate virtual to physical
```

```
                                0266   658                                          ; Clear everything but byte offset
          50    6647    D0      0266   659          MOVL    (R6)[R7],R0             ; Get PFN in R0
   5A   15   09   50    F0      026A   660          INSV    R0,#VA$V_VPN,#PTE$S_PFN,R10   ; Insert PFN after byte offset
                        05      026F   661 90$:     RSB
                                0270   662
                                0270   663
                                0270   664 ;
                                0270   665 ; Map physical to virtual
                                0270   666 ;
                        00      0270   667 100$:    HALT                            ; Not implemented yet
                                0271   668
                                0271   669          .DSABL  LSB
                                0271   670
                  00000271      0271   671 CONSDD_DRVSIZ=.-CONSDD_START
                                0271   672
                                0271   673          .END
```

```
$TABLE                          = 00000000 R  D  02
BTD$K_CONSOLE                   = 00000040    D
CLK$RE_INIT                       ********  X  03
CONSDD_DRIVER                     00000033 R  D  03
CONSDD_DRVSIZ                   = 00000271    D
CONSDD_START                      00000000 R  D  03
CPU_730                           00000017 R  D  03
CPU_750                           0000001D R  D  03
CPU_780                           00000016 R  D  03
CSRD_V_ERROR                    = 0000000F    D
CSRS_V_DONE                     = 00000007    D
CSTS_M_BREAK                    = 00000001    D
CSTS_V_READY                    = 00000007    D
DDNAME                            00000026 R  D  03
DD_CHECKCONTROLC                  0000017A R  D  03
DD_CNTRL                        = 00000002    D
DD_CONTINUE                     = 00000010    D
DD_DATA                         = 00000001    D
DD_ENDPKT                       = 00000040    D
DD_ERROR                          000001B6 R  D  03
DD_ERROR1                         000001E6 R  D  03
DD_ERROR2                         0000014B R  D  03
DD_INIT                         = 00000004    D
DD_READ                         = 00000002    D
DD_SELECT                         00000000 R  D  03
DD_TIMEOUT                        000001AF R  D  03
DD_WRITE                        = 00000003    D
DO_MAPPING                        00000259 R  D  03
DS$GL_FLAGS                       ********  X  03
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_NI                         = 04000000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
END_OF_DATA                       00000143 R  D  03
EXE$GB_CPUTYPE                    ********  X  03
FUNC                            = 00000010    D
GETBYTE                           0000023A R  D  03
INIT                              00000024 R  D  03
IO$_READLBLK                    = 00000021    D
KB_POLL                           ********  X  03
MODE                            = 00000014    D
MRSP                              00000025 R  D  03
MRSP_SWITCH                     = 00000003    D
PR$_CSRD                        = 0000001D    D
PR$_CSRS                        = 0000001C    D
PR$_CSTD                        = 0000001F    D
PR$_CSTS                        = 0000001E    D
PR$_ICCS                        = 00000018    D
PR$_MAPEN                       = 00000038    D
PR$_SID_TYP780                  = 00000001    D
PTE$S_PFN                       = 00000015    D
PUTBYTE                           00000245 R  D  03
RECV_ONE_CHAR                     0000021D R  D  03
RECV_TWO_CHARS                    00000218 R  D  03
RECV_TWO_UPDSUM                   0000020A R  D  03
RESET_CLOCK                       0000019E R  D  03
SIZ...                          = 00000008    D
SS$_BUFBYTALI                   = 0000030C    D
SS$_CONTROLC                    = 00000651    D
SS$_CTRLERR                     = 00000054    D
```

```
SS$_NORMAL                      = 00000001     D
SS$_TIMEOUT                     = 0000022C     D
STKPTR                            00000020 R   D   03
VA$M_BYTE                       = 000001FF     D
VA$V_VPN                        = 00000009     D
WAIT_READY                        000001E8 R   D   03
XMIT_ONE_CHAR                     000001DD R   D   03
XMIT_TWO_CHARS                    000001D1 R   D   03
XMIT_TWO_UPDSUM                   000001CB R   D   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000271 (625.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	179 (2)	179 (2)
BIT...	=00000008	98 (2)	98 (2)
BTD$K_CONSOLE	=00000040		179 (2)
CLK$RE_INIT	00000000-XR		466 (4)
CONSDD_DRIVER	00000033-R	270 (4)	179 (2)
CONSDD_DRVSIZ	=00000271	671 (4)	179 (2)
CONSDD_START	00000000-R	213 (3)	179 (2) 671 (4)
CPU_730	00000017-R	225 (3)	221 (3)
CPU_750	0000001D-R	228 (3)	221 (3)
CPU_780	00000016-R	222 (3)	221 (3)
CSRD_V_ERROR	=0000000F		#-580 (4)
CSRS_V_DONE	=00000007		#-541 (4)
CSTS_M_BREAK	=00000001		#-313 (4)
CSTS_V_READY	=00000007		#-578 (4)
DDNAME	00000026-R	268 (4)	179 (2)
DD_CHECKCONTROLC	0000017A-R	449 (4)	#-418 (4)
DD_CNTRL	=00000002		#-346 (4) #-429 (4)
DD_CONTINUE	=00000010		#-321 (4) #-378 (4) #-579 (4)
DD_DATA	=00000001		343 (4) #-406 (4)
DD_ENDPKT	=00000040		#-433 (4)
DD_ERROR	000001B6-R	476 (4)	#-417 (4) #-431 (4) #-434 (4) #-436 (4)
			#-443 (4) #-520 (4)
DD_ERROR1	000001E6-R	519 (4)	#-580 (4)
DD_ERROR2	0000014B-R	430 (4)	#-407 (4)
DD_INIT	=00000004		#-318 (4)
DD_READ	=00000002		342 (4) #-348 (4)
DD_SELECT	00000000-R	215 (3)	179 (2)
DD_TIMEOUT	000001AF-R	472 (4)	#-543 (4)
DD_WRITE	=00000003		342 (4)
DO_MAPPING	00000259-R	653 (4)	#-298 (4)
DS$GL_FLAGS	00000000-XR		452 (4)
DS$M_ABRTFLG	=00000040	98 (2)	
DS$M_BADTIME	=00100000	98 (2)	
DS$M_BATCH	=00400000	98 (2)	
DS$M_BRKCLR	=00001000	98 (2)	
DS$M_BRKPT	=00000800	98 (2)	
DS$M_CHARFLG	=00000100	98 (2)	
DS$M_CMDFLG	=00000080	98 (2)	
DS$M_CTRLC	=00000001	98 (2)	
DS$M_CTRLO	=00010000	98 (2)	
DS$M_DEVFLG	=00000200	98 (2)	
DS$M_DISABLCC	=01000000	98 (2)	
DS$M_DONFLG	=00002000	98 (2)	
DS$M_ERRFLG	=00000010	98 (2)	
DS$M_EXCEPT	=00080000	98 (2)	
DS$M_EXETST	=00040000	98 (2)	
DS$M_HLTFLG	=00000008	98 (2)	
DS$M_LODFLG	=00000002	98 (2)	
DS$M_MEMMGT	=00008000	98 (2)	
DS$M_NI	=04000000	98 (2)	

```
DS$M_OUTPUT                     =00800000     98     (2)
DS$M_RUBFLG                     =00000020     98     (2)
DS$M_SCRIPT                     =00200000     98     (2)
DS$M_SETIMR                     =02000000     98     (2)
DS$M_STRFLG                     =00000004     98     (2)
DS$M_SUBT                       =00004000     98     (2)
DS$M_SYSFLG                     =00000400     98     (2)
DS$M_TIMED                      =02000000     98     (2)
DS$M_TIMED_OUT                  =08000000     98     (2)
DS$M_TIMRON                     =00020000     98     (2)
DS$V_ABRTFLG                    =00000006     98     (2)
DS$V_BADTIME                    =00000014     98     (2)
DS$V_BATCH                      =00000016     98     (2)
DS$V_BRKCLR                     =0000000C     98     (2)
DS$V_BRKPT                      =0000000B     98     (2)
DS$V_CHARFLG                    =00000008     98     (2)
DS$V_CMDFLG                     =00000007     98     (2)
DS$V_CTRLC                      =00000000     98     (2)    #-451     (4)
DS$V_CTRLO                      =00000010     98     (2)
DS$V_DEVFLG                     =00000009     98     (2)
DS$V_DISABLCC                   =00000018     98     (2)
DS$V_DONFLG                     =0000000D     98     (2)
DS$V_ERRFLG                     =00000004     98     (2)
DS$V_EXCEPT                     =00000013     98     (2)
DS$V_EXETST                     =00000012     98     (2)
DS$V_HLTFLG                     =00000003     98     (2)
DS$V_LODFLG                     =00000001     98     (2)
DS$V_MEMMGT                     =0000000F     98     (2)
DS$V_NI                         =0000001A     98     (2)
DS$V_OUTPUT                     =00000017     98     (2)
DS$V_RUBFLG                     =00000005     98     (2)
DS$V_SCRIPT                     =00000015     98     (2)
DS$V_SETIMR                     =00000019     98     (2)
DS$V_STRFLG                     =00000002     98     (2)
DS$V_SUBT                       =0000000E     98     (2)
DS$V_SYSFLG                     =0000000A     98     (2)
DS$V_TIMED                      =00000019     98     (2)
DS$V_TIMED_OUT                  =0000001B     98     (2)
DS$V_TIMRON                     =00000011     98     (2)
END_OF_DATA                      00000143-R   426    (4)    #-399     (4)
EXE$GB_CPUTYPE                   00000000-XR           (3)    #-221     (3)
FUNC                            =00000010     257    (4)    #-349     (4)    #-371     (4)
GETBYTE                          0000023A-R   612    (4)    #-389     (4)    #-391     (4)
INIT                             00000024-R   266    (4)    #-303     (4)    #-323     (4)    #-453     (4)    #-479     (4)
IO$_READLBLK                    =00000021                   #-349     (4)    #-371     (4)
KB_POLL                          00000000-XR           (3)    450       (4)
MODE                            =00000014     258    (4)    #-295     (4)
MRSP                             00000025-R   267    (4)    #-216     (3)    #-226     (3)    #-300     (4)    #-359     (4)
                                                            #-463     (4)    #-575     (4)
MRSP_SWITCH                     =00000003                   #-361     (4)
PR$_CSRD                        =0000001D                   #-317     (4)    #-574     (4)
PR$_CSRS                        =0000001C                   #-572     (4)
PR$_CSTD                        =0000001F                   #-516     (4)    #-579     (4)
PR$_CSTS                        =0000001E                   #-313     (4)    #-316     (4)    #-514     (4)    #-577     (4)
PR$_ICCS                        =00000018                   #-302     (4)
PR$_MAPEN                       =00000038                   #-293     (4)
PR$_SID_TYP780                  =00000001                   #-221     (3)
```

```
PTE$S_PFN                       =00000015                       #-660    (4)
PUTBYTE                          00000245-R   620    (4)      #-410    (4)     #-412    (4)
RECV_ONE_CHAR                    0000021D-R   571    (4)      #-320    (4)     #-377    (4)     #-567    (4)
RECV_TWO_CHARS                   00000218-R   566    (4)      #-414    (4)     #-440    (4)     #-560    (4)
RECV_TWO_UPDSUM                  0000020A-R   559    (4)      #-405    (4)     #-409    (4)     #-428    (4)     #-432    (4)
                                                             #-438    (4)
RESET_CLOCK                      0000019E-R   462    (4)      #-444    (4)     #-454    (4)     #-480    (4)
SIZ...                          =00000008     98     (2)       98      (2)
SS$_BUFBYTALI                   =0000030C                      #-273    (4)
SS$_CONTROLC                    =00000651                      #-457    (4)
SS$_CTRLERR                     =00000054                      #-477    (4)
SS$_NORMAL                      =00000001                      #-445    (4)
SS$_TIMEOUT                     =0000022C                      #-473    (4)
STKPTR                           00000020-R   265    (4)      #-276    (4)     #-455    (4)     #-481    (4)
VA$M_BYTE                       =000001FF                      #-628    (4)     #-657    (4)
VA$V_VPN                        =00000009                      #-660    (4)
WAIT_READY                       000001E8-R   536    (4)      #-515    (4)     #-573    (4)
XMIT_ONE_CHAR                    000001DD-R   513    (4)      #-509    (4)
XMIT_TWO_CHARS                   000001D1-R   506    (4)      #-315    (4)     #-319    (4)     #-370    (4)     #-396    (4)
XMIT_TWO_UPDSUM                  000001CB-R   502    (4)      #-347    (4)     #-352    (4)     #-362    (4)     #-364    (4)
                                                             #-366    (4)     #-368    (4)     #-387    (4)     #-393    (4)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$BOOT_DRIVER    1           174     (2)     174     (2)
$BTDDEF         2           99      (2)     99      (2)
$DEF            1           98      (2)
$DEFINI         1           99      (2)     100     (2)     101     (2)     102     (2)     103     (2)     104     (2)
                                            105     (2)     119     (2)     99      (2)
$EQU            1           98      (2)
$GBLINI         2           98      (2)     98      (2)
$IODEF          15          100     (2)     100     (2)
$PRDEF          5           101     (2)     101     (2)
$PTEDEF         3           102     (2)     102     (2)
$RPBDEF         5           103     (2)     103     (2)
$SSDEF          21          104     (2)     104     (2)
$VADEF          1           105     (2)     105     (2)
$VIELD          1           98      (2)     98      (2)
$VIELD1         1           98      (2)     98      (2)
ASSUME          1           342     (4)     342     (4)     343     (4)
CASE            1           221     (3)     221     (3)
CPUDISP         1           218     (3)     218     (3)
DSFDEF          3           98      (2)     98      (2)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	295 (4)
ADDW	00A0	503 (4) 562 (4)
ADWC	00D8	504 (4) 563 (4)
ASHL	0078	385 (4) 510 (4)
BBC	00E1	451 (4) 578 (4)
BBS	00E0	541 (4) 580 (4)
BBSS	00E2	361 (4)
BEQL	0013	301 (4) 304 (4) 350 (4) 360 (4) 372 (4) 464 (4) 576 (4)
		615 (4) 622 (4) 627 (4)
BICL	00CA	657 (4)
BITW	00B3	628 (4)
BLBC	00E9	272 (4)
BLSS	0019	436 (4) 656 (4)
BLSSU	001F	383 (4)
BNEQ	0012	322 (4) 379 (4) 398 (4) 407 (4) 417 (4) 420 (4) 431 (4)
		434 (4) 443 (4) 629 (4)
BRB	0011	399 (4) 474 (4) 520 (4) 543 (4) 617 (4) 655 (4)
BRW	0031	305 (4)
BSBB	0010	418 (4) 444 (4) 454 (4) 480 (4) 509 (4) 515 (4) 560 (4)
		567 (4) 573 (4)
BSBW	0030	298 (4) 315 (4) 319 (4) 320 (4) 347 (4) 352 (4) 362 (4)
		364 (4) 366 (4) 368 (4) 370 (4) 377 (4) 387 (4) 389 (4)
		391 (4) 393 (4) 396 (4) 405 (4) 409 (4) 410 (4) 412 (4)
		414 (4) 428 (4) 432 (4) 438 (4) 440 (4)
CASEB	008F	221 (3)
CLRB	0094	216 (3) 453 (4) 479 (4)
CLRL	00D4	217 (3) 314 (4) 345 (4) 358 (4) 363 (4) 380 (4) 404 (4)
		427 (4) 613 (4)
CMPB	0091	321 (4) 378 (4) 406 (4) 429 (4) 433 (4)
CMPL	00D1	349 (4) 371 (4) 382 (4)
CMPW	00B1	416 (4) 442 (4)
DECL	00D7	626 (4)
DIVL	00C6	388 (4)
DIVL3	00C7	408 (4)
HALT	0000	667 (4)
INCB	0096	323 (4)
INCL	00D6	226 (3) 229 (3) 351 (4) 386 (4) 630 (4)
INSV	00F0	392 (4) 415 (4) 441 (4) 561 (4) 660 (4)
JSB	0016	450 (4) 466 (4)
MFPR	00DB	293 (4) 317 (4) 540 (4) 574 (4) 577 (4)
MNEGL	00CE	294 (4)
MOVB	0090	623 (4)
MOVL	00D0	276 (4) 384 (4) 390 (4) 437 (4) 455 (4) 481 (4) 538 (4)
		659 (4)
MOVQ	007D	297 (4)
MOVZBL	009A	348 (4) 381 (4) 411 (4) 508 (4) 514 (4) 568 (4) 572 (4)
		616 (4)
MOVZWL	003C	273 (4) 318 (4) 346 (4) 365 (4) 367 (4) 369 (4) 395 (4)
		445 (4) 457 (4) 473 (4) 477 (4)
MTPR	00DA	302 (4) 313 (4) 316 (4) 516 (4) 579 (4)

POPR	00BA	274	(4)	446	(4)	456	(4)	467	(4)	482	(4)	544	(4)			
PUSHL	00DD	507	(4)													
PUSHR	00BB	271	(4)	465	(4)	537	(4)									
RET	0004	458	(4)													
RSB	0005	223	(3)	230	(3)	275	(4)	447	(4)	460	(4)	468	(4)	483	(4)	
		517	(4)	545	(4)	564	(4)	581	(4)	661	(4)					
SOBGTR	00F5	394	(4)	413	(4)	439	(4)	542	(4)							
TSTB	0095	300	(4)	303	(4)	359	(4)	463	(4)	575	(4)					
TSTL	00D5	397	(4)	419	(4)	435	(4)	614	(4)	621	(4)	654	(4)			

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ASCIC          268     (4)
.BYTE           266     (4)     267     (4)     539     (4)
.DISABLE        485     (4)
.DSABL          669     (4)
.ENABL          610     (4)
.ENABLE         470     (4)
.END            673     (4)
.ENDC           179     (2)     342     (4)     343     (4)     98      (2)
.ENDM           100     (2)     101     (2)     102     (2)     103     (2)     104     (2)     105     (2)     174     (2)     218     (3)
                221     (3)     342     (4)     98      (2)     99      (2)
.ENDR           221     (3)     98      (2)
.IDENT          5       (1)
.IF             179     (2)     342     (4)     343     (4)     98      (2)
.IFF            179     (2)     342     (4)     343     (4)     98      (2)
.IIF            98      (2)
.IRP            221     (3)     98      (2)
.LIBRARY        94      (2)     95      (2)     96      (2)
.LONG           179     (2)     265     (4)
.MACRO          98      (2)
.NOCROSS        100     (2)     101     (2)     102     (2)     103     (2)     104     (2)     105     (2)     119     (2)     99      (2)
.PAGE           486     (4)
.PSECT          179     (2)
.RESTORE        100     (2)     101     (2)     102     (2)     103     (2)     104     (2)     105     (2)     168     (2)     99      (2)
.SAVE           100     (2)     101     (2)     102     (2)     103     (2)     104     (2)     105     (2)     119     (2)     99      (2)
.SBTTL          182     (3)     232     (4)     90      (2)
.SIGNED_WORD    221     (3)
.SUBTITLE       487     (4)
.TITLE          4       (1)
.WORD           179     (2)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	3	#-295	(4)	#-349	(4)	#-371	(4)						
R0	17	#-217	(3)	#-229	(3)	#-273	(4)	#-317	(4)	#-445	(4)	#-457	(4)
		#-465	(4)	#-467	(4)	#-473	(4)	#-477	(4)	#-514	(4)	#-540	(4)
		#-572	(4)	#-577	(4)	578	(4)	#-659	(4)	#-660	(4)		
R1	22	#-271	(4)	#-274	(4)	#-297	(4)	#-381	(4)	#-382	(4)	#-384	(4)
		#-385	(4)	#-388	(4)	#-394	(4)	#-408	(4)	#-413	(4)	#-437	(4)
		#-439	(4)	#-446	(4)	#-456	(4)	#-465	(4)	#-467	(4)	#-482	(4)
		#-537	(4)	#-538	(4)	#-542	(4)	#-544	(4)				
R10	10	#-271	(4)	#-274	(4)	#-446	(4)	#-456	(4)	#-482	(4)	#-616	(4)
		#-623	(4)	#-628	(4)	#-657	(4)	660	(4)				
R11	10	#-271	(4)	#-274	(4)	#-293	(4)	#-294	(4)	#-295	(4)	#-446	(4)
		#-456	(4)	#-482	(4)	#-654	(4)						
R2	44	#-271	(4)	#-274	(4)	#-314	(4)	#-318	(4)	#-321	(4)	#-346	(4)
		#-348	(4)	#-351	(4)	#-358	(4)	361	(4)	#-363	(4)	#-365	(4)
		#-367	(4)	#-369	(4)	#-378	(4)	#-385	(4)	#-386	(4)	#-390	(4)
		392	(4)	#-395	(4)	#-408	(4)	#-411	(4)	#-415	(4)	#-435	(4)
		#-441	(4)	#-446	(4)	#-456	(4)	#-465	(4)	#-467	(4)	#-482	(4)
		#-503	(4)	#-507	(4)	#-508	(4)	#-510	(4)	#-516	(4)	#-537	(4)
		#-540	(4)	541	(4)	#-544	(4)	#-561	(4)	#-568	(4)	#-574	(4)
		580	(4)										
R3	18	#-390	(4)	#-392	(4)	#-406	(4)	#-411	(4)	415	(4)	#-416	(4)
		#-429	(4)	#-433	(4)	441	(4)	#-442	(4)	#-465	(4)	#-467	(4)
		561	(4)	#-562	(4)	#-568	(4)	#-613	(4)	#-616	(4)	#-623	(4)
R4	14	#-345	(4)	#-369	(4)	#-380	(4)	#-395	(4)	#-404	(4)	#-416	(4)
		#-427	(4)	#-442	(4)	#-465	(4)	#-467	(4)	#-503	(4)	#-504	(4)
		#-562	(4)	#-563	(4)								
R5	3	#-367	(4)	#-465	(4)	#-467	(4)						
R6	2	#-297	(4)	#-659	(4)								
R7	2	#-630	(4)	#-659	(4)								
R8	14	#-271	(4)	#-272	(4)	#-274	(4)	#-365	(4)	#-382	(4)	#-384	(4)
		#-397	(4)	#-419	(4)	#-446	(4)	#-456	(4)	#-482	(4)	#-614	(4)
		#-621	(4)	#-626	(4)								
SP	4	#-276	(4)	#-455	(4)	#-481	(4)	#-510	(4)				

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.20
Command processing	892	00:00:00.26	00:00:00.76
Pass 1	786	00:00:02.61	00:00:04.54
Symbol table sort	0	00:00:00.33	00:00:00.33
Pass 2	63	00:00:00.73	00:00:01.39
Symbol table output	2	00:00:00.02	00:00:00.06
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	5	00:00:00.17	00:00:00.32
Assembler run totals	1774	00:00:04.16	00:00:07.61

```
The working set limit was 2900 pages.
132768 bytes (260 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1150 non-local and 26 local symbols.
673 source lines were read in Pass 1, producing 76 object records in Pass 2.
55 pages of virtual memory were used to define 21 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	3
SYS$COMMON:[SYSLIB]LIB.MLB;2	4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	17

```
1203 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CSDDBTDRV.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

Table of contents

(1) 219 CONVERT BINARY TIME TO ASCII STRING
(1) 314 CONVERT ASCII STRING TO BINARY TIME
(1) 567 CONVERT BINARY TIME TO NUMERIC TIME

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CVRTIM.MAR
0000     2 ; *1     6-FEB-1986 15:16:43 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element CVRTIM.MAR
0000     4         .TITLE  CVRTIM  *** CVRTIM time conversion routines
0000     5         .IDENT  /V02-04/
0000     6
0000     7 ;
0000     8 ;******************************************************************************
0000     9 ;*                                                                            *
0000    10 ;*  COPYRIGHT (c) 1976, 1977, 1978, 1979, 1980                                *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                          *
0000    12 ;*                                                                            *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED    *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE   *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER    *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY    *
0000    18 ;*  TRANSFERRED.                                                              *
0000    19 ;*                                                                            *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE    *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    22 ;*  CORPORATION.                                                              *
0000    23 ;*                                                                            *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS    *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
0000    26 ;*                                                                            *
0000    27 ;******************************************************************************
0000    28 ;
0000    29 ; D. N. CUTLER 6-JAN-76
0000    30 ;
0000    31 ; SYSTEM SERVICES TO CONVERT TIME
0000    32 ;
0000    33 ;       CONVERT BINARY TIME TO ASCII STRING
0000    34 ;       CONVERT ASCII STRING TO BINARY TIME
0000    35 ;       CONVERT BINARY TIME TO NUMERIC FORMAT
0000    36 ;
0000    37 ;       THE CONVERSION ALGORITHMS USED HEREIN WERE DEVELOPED BY P. CONKLIN,
0000    38 ;       M. SPIER, AND D. ROSENBERY ON THE PDP-10.
0000    39 ;
0000    40 ; MODIFIED BY:
0000    41 ;
0000    42 ;                  Roger Riggs, 8-May-1980, Version 5.04
0000    43 ;       02         Emulate character string instructions
0000    44 ;
0000    45 ;       03         Bob Bergazzi     Aug 22, 1984    Version 7.1
0000    46 ;                  Made DATETABLE a global symbol.
0000    47 ;--
0000    48 ;       V02        LMK0001          LEN KAWELL      14-MAR-1979
0000    49 ;                  FIX LOSS OF PRECISION IN HUNDRETHS OF SECONDS IN $NUMTIM
0000    50 ;
0000    51 ;       04         Ted Salem        March 22, 1985  Version 8.2
0000    52 ;                  Interlock routine exe$numtim.
0000    53 ;
0000    54 ; MACRO LIBRARY CALLS
0000    55 ;
0000    56         .LIBRARY        /$DS/           ;                                  [04]
0000    57
```

```
                        0000     58          .library        /$diag/
                        0000     59          $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     60          $MP_ILDEFS                      ;DEFINE MP$R_ACTIVE_ROUTINE offsets [04]
                        0000     61 ;
                        0000     62 ; EXTERNALS:
                        0000     63 ;
                        0000     64
                        0000     65 ;        .EXTRN          MP$R_ACTIVE_ROUTINES    ;BITVECTOR FOR INTERLOCKING [04]
                        0000     66 ;        .EXTRN          MP$R_GET_PROCESSOR_ID   ;                           [04]
                        0000     67 ;        .EXTRN          MP$R_ROUTINE_COUNT_TABLE ;                          [04]
                        0000     68 ;        .EXTRN          MP$R_ROUTINE_ID_TABLE   ;                           [04]
                        0000     69 ;
                        0000     70
                        0000     71 ; LOCAL SYMBOLS
                        0000     72 ;
                        0000     73 ; ARGUMENT LIST OFFSET DEFINTIONS FOR CONVERT BINARY TIME TO ASCII STRING
                        0000     74 ;
                        0000     75
00000004                0000     76 ATIMLEN=4                               ;ADDRESS OF WORD TO STORE LENGTH
00000008                0000     77 ATIMBUF=8                               ;ADDRESS OF OUTPUT BUFFER DESCRIPTOR
0000000C                0000     78 ATIMADR=12                              ;ADDRESS OF 64-BIT ABSOLUTE OR DELTA TIME
00000010                0000     79 ACVTFLG=16                              ;CONVERSION INDICATOR
                        0000     80
                        0000     81 ;
                        0000     82 ; ARGUMENT LIST OFFSET DEFINITIONS FOR CONVERT ASCII STRING TO BINARY TIME
                        0000     83 ;
                        0000     84
00000004                0000     85 BTIMBUF=4                               ;ADDRESS OF ASCII STRING DESCRIPTOR
00000008                0000     86 BTIMADR=8                               ;ADDRESS TO STORE 64-BIT ABSOLUTE OR DELTA T
                        0000     87
                        0000     88 ;
                        0000     89 ; ARGUMENT LIST OFFSET DEFINITIONS FOR CONVERT BINARY TIME TO NUMERIC TIME
                        0000     90 ;
                        0000     91
00000004                0000     92 NTIMBUF=4                               ;ADDRESS OF 7-WORD BUFFER TO RECEIVE TIME
00000008                0000     93 NTIMADR=8                               ;ADDRESS OF 64-BIT ABSOLUTE OR DELTA TIME
                        0000     94
                        0000     95 ;
                        0000     96 ; CONVERSION CONSTANTS
                        0000     97 ;
                        0000     98 ; TOTAL DAYS IN A CENTURY
                        0000     99 ;
                        0000    100
00008EAC                0000    101 CENTURYDAYS=<100*365>+<100/4>-<100/100> ;
                        0000    102
                        0000    103 ;
                        0000    104 ; AVERAGE QUARTER DAYS PER CENTURY
                        0000    105 ;
                        0000    106
00023AB1                0000    107 QDAYSPCENT=<<<100*365>+<100/4>-<100/100>>*4>+<400/400> ;
                        0000    108
                        0000    109 ;
                        0000    110 ; AVERAGE QUARTER DAYS PER YEAR
                        0000    111 ;
                        0000    112
```

```
                    000005B5  0000    113 QDAYSPYEAR=<365*4>+1                    ;
                              0000    114
                              0000    115 ;
                              0000    116 ; TOTAL DAYS IN A QUADRICENTURY
                              0000    117 ;
                              0000    118
                    00023AB1  0000    119 QUADRIDAYS=<400*365>+<400/4>-<400/100>+<400/400> ;
                              0000    120
                              0000    121 ;
                              0000    122 ; TOTAL DAYS IN A QUADYEAR
                              0000    123 ;
                              0000    124
                    000005B5  0000    125 QUADYEARDAYS=<365*4>+1                  ;
                              0000    126
                              0000    127 ;
                              0000    128 ; OFFSET IN DAYS FROM 1-JAN-1501 TO 17-NOV-1858
                              0000    129 ;
                              0000    130
                    0001FE98  0000    131 TIMOFF1=<<1858-1501>*365>+<<1858-1501>/4>-<<1858-1501>/100>+<<1858-1501>/400>+ -;
                              0000    132       31+28+31+30+31+30+31+31+30+31+17 ;
                              0000    133
                              0000    134 ;
                              0000    135 ; OFFSET IN DAYS FROM 1-JAN-1601 TO 17-NOV-1858
                              0000    136 ;
                              0000    137
                              0000    138 TIMOFF2=<<1858-1601>*365>+<<1858-1601>/4>-<<1858-1601>/100>+<<1858-1601>/400>+ -;
                    00016FEC  0000    139       31+28+31+30+31+30+31+31+30+31+17 ;
                              0000    140
                              0000    141 ;
                              0000    142 ; CHARACTER CODE DEFINITIONS
                              0000    143 ;
                              0000    144
                    00000020  0000    145 BLANK=32                                ;
                    0000003A  0000    146 COLON=58                                ;
                    0000002D  0000    147 HYPHEN=45                               ;
                    00000039  0000    148 NINE=57                                 ;
                    00000030  0000    149 ONE=48                                  ;
                    0000002E  0000    150 PERIOD=46                               ;
                              0000    151
                              0000    152 ;
                              0000    153 ; NUMERIC TIME BUFFER OFFSET DEFINITIONS
                              0000    154 ;
                              0000    155
                    00000000  0000    156 YEAR=0                                  ;
                    00000002  0000    157 MONTH=2                                 ;
                    00000004  0000    158 DAY=4                                   ;
                    00000006  0000    159 HOUR=6                                  ;
                    00000008  0000    160 MINUTE=8                                ;
                    0000000A  0000    161 SECOND=10                               ;
                    0000000C  0000    162 HUNDREDTH=12                            ;
                              0000    163
                              0000    164 ;
                              0000    165 ; LOCAL DATA
                              0000    166 ;
                              0000    167 ; MONTH, DAY CONVERSION TABLE
                              0000    168 ;
                          00000000    169         .PSECT  SEP, SHR, EXE, WRT, LONG
```

```
                                  0000  DATETABLE::                                  ;DATE CONVERSION TABLE          [03]
                              1F  0000          .BYTE   31                           ;JANUARY
                              1D  0001          .BYTE   29                           ;FEBRUARY
                              1F  0002          .BYTE   31                           ;MARCH
                              1E  0003          .BYTE   30                           ;APRIL
                              1F  0004          .BYTE   31                           ;MAY
                              1E  0005          .BYTE   30                           ;JUNE
                              1F  0006          .BYTE   31                           ;JULY
                              1F  0007          .BYTE   31                           ;AUGUST
                              1E  0008          .BYTE   30                           ;SEPTEMBER
                              1F  0009          .BYTE   31                           ;OCTOBER
                              1E  000A          .BYTE   30                           ;NOVEMBER
                              1F  000B          .BYTE   31                           ;DECEMBER
                                  000C
                                  000C  ;
                                  000C  ; MONTH CONVERSION TABLE
                                  000C  ;
                                  000C
                                  000C  MONTHTAB:                                    ;
                     4E 41 4A 03  000C          .ASCII  <3>/JAN/                     ;
                     42 45 46 03  0010          .ASCII  <3>/FEB/                     ;
                     52 41 4D 03  0014          .ASCII  <3>/MAR/                     ;
                     52 50 41 03  0018          .ASCII  <3>/APR/                     ;
                     59 41 4D 03  001C          .ASCII  <3>/MAY/                     ;
                     4E 55 4A 03  0020          .ASCII  <3>/JUN/                     ;
                     4C 55 4A 03  0024          .ASCII  <3>/JUL/                     ;
                     47 55 41 03  0028          .ASCII  <3>/AUG/                     ;
                     50 45 53 03  002C          .ASCII  <3>/SEP/                     ;
                     54 43 4F 03  0030          .ASCII  <3>/OCT/                     ;
                     56 4F 4E 03  0034          .ASCII  <3>/NOV/                     ;
                     43 45 44 03  0038          .ASCII  <3>/DEC/                     ;
                                  003C
                                  003C  ;
                                  003C  ; HOURS, MINUTES, SECONDS, HUNDREDTHS CONVERSION TABLE
                                  003C  ;
                                  003C
                                  003C  TIMETABLE:                                   ;TIME CONVERSION TABLE
                              64  003C          .BYTE   100                          ;HUNDREDTHS
                              3C  003D          .BYTE   60                           ;SECONDS
                              3C  003E          .BYTE   60                           ;MINUTES AND HOURS
                                  003F
                                  003F  ;
                                  003F  ; CONVERSION CONTROL STRINGS
                                  003F  ;
                                  003F
5A 34 21 2D 43 41 21 2D 57 53 32 21  003F  DATE:   .ASCII  /!2SW-!AC-!4ZW /            ;
                             20 57  004B
                     20 57 53 34 21  004D  DELTA:  .ASCII  /!4SW /                     ;
32 21 3A 57 5A 32 21 3A 57 5A 32 21  0052  TIME:   .ASCII  /!2ZW:!2ZW:!2ZW.!2ZW/       ;
                57 5A 32 21 2E 57 5A  005E
```

```
                                0065          .SBTTL  CONVERT BINARY TIME TO ASCII STRING
                                0065  ;+
                                0065  ; EXE$ASCTIM - CONVERT BINARY TIME TO ASCII STRING
                                0065  ;
                                0065  ; THIS SERVICE PROVIDES THE CAPABILITY TO CONVERT AN ABSOLUTE OR DELTA
                                0065  ; TIME FROM 64-BIT FORMAT TO AN ASCII STRING.
                                0065  ;
                                0065  ; INPUTS:
                                0065  ;
                                0065  ;       ATIMLEN(AP) = ADDRESS OF WORD TO RECEIVE OUTPUT LENGTH.
                                0065  ;       ATIMBUF(AP) = ADDRESS OF OUTPUT BUFFER DESCRIPTOR.
                                0065  ;       ATIMADR(AP) = ADDRESS OF 64-BIT TIME VALUE. IF ZERO, THEN THE CURRENT
                                0065  ;               SYSTEM TIME IS USED. POSITIVE VALUES ARE INTERPRETED AS
                                0065  ;               ABSOLUTE TIMES AND NEGATIVE VALUES AS DELTA TIMES.
                                0065  ;       ACVTFLG(AP) = CONVERSION INDICATOR.
                                0065  ;               LOW BIT CLEAR INDICATES BOTH DATE AND TIME ARE TO BE CON-
                                0065  ;                       VERTED.
                                0065  ;               LOW BIT SET INDICATES ONLY TIME IS TO BE CONVERTED.
                                0065  ;
                                0065  ; OUTPUTS:
                                0065  ;
                                0065  ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CONVERT TIME TO ASCII.
                                0065  ;
                                0065  ;               R0 = SS$_ACCVIO - 64-BIT TIME VALUE OR OUTPUT BUFFER DESCRIPTOR
                                0065  ;                       CANNOT BE READ BY CALLING ACCESS MODE, OR OUTPUT BUFFER
                                0065  ;                       CANNOT BE WRITTEN BY CALLING ACCESS MODE.
                                0065  ;
                                0065  ;               R0 = SS$_IVTIME - SPECIFIED DELTA TIME IS GREATER THAN 9999
                                0065  ;                       DAYS.
                                0065  ;
                                0065  ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0065  ;
                                0065  ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0065  ;-
                                0065
                                0065  EXE$ASCTIM::                                  ;CONVERT TIME TO ASCII
                        007C    0065          .WORD   ^M<R2,R3,R4,R5,R6>            ;ENTRY MASK
          7E    08 BC     7D    0067          MOVQ    @ATIMBUF(AP),-(SP)            ;SAVE OUTPUT BUFFER DESCRIPTOR
             56    5E     D0    006B          MOVL    SP,R6                         ;SAVE ADDRESS OF OUTPUT BUFFER DESCRIPTOR
                   7E     D4    006E          CLRL    -(SP)                         ;CLEAR SPACE FOR LENGTH FROM FAO
             55    5E     D0    0070          MOVL    SP,R5                         ;SAVE ADDRESS OF LENGTH
                   52     D4    0073          CLRL    R2                            ;ASSUME ABSOLUTE TIME SPECIFIED
          53    0C AC     D0    0075          MOVL    ATIMADR(AP),R3                ;GET ADDRESS OF 64-BIT TIME VALUE
                   07     13    0079          BEQL    10$                           ;IF EQL NONE SPECIFIED
             50    63     7D    007B          MOVQ    (R3),R0                       ;GET 64-BIT TIME VALUE
                   02     18    007E          BGEQ    10$                           ;IF GEQ ABSOLUTE TIME
                   52     D6    0080          INCL    R2                            ;INDICATE DELTA TIME
             5E    10     C2    0082  10$:    SUBL    #<<<7*2>+3>/4>*4,SP           ;ALLOCATE NUMERIC TIME BUFFER
             54    5E     D0    0085          MOVL    SP,R4                         ;SAVE ADDRESS OF NUMERIC TIME BUFFER
                                0088          $NUMTIM_S (R4),(R3)                   ;CONVERT TIME TO NUMERIC FORMAT
                   63     7F    0088                  PUSHAQ  (R3)
                   64     DF    008A                  PUSHAL  (R4)
  00000000'GF      02     FB    008C                  CALLS   #2,G^SYS$NUMTIM
                6F 50     E9    0093          BLBC    R0,60$                        ;IF LBC CONVERSION FAILURE
                                0096
                                0096  ;
                                0096  ; CONVERT TIME TO ASCII FORMAT
```

```
                                0096   ;
                                0096
          3E 10 AC    E8        0096           BLBS    ACVTFLG(AP),40$          ;IF LBS ONLY TIME IS TO BE CONVERTED
             12 52    E8        009A           BLBS    R2,20$                   ;IF LBS DELTA TIME SPECIFIED
                                009D
                                009D   ;
                                009D   ; CONVERT DATE
                                009D   ;
                                009D
       52    02 A4    3C        009D           MOVZWL  MONTH(R4),R2             ;GET NUMERIC MONTH VALUE
   52    FF62 CF42    DE        00A1           MOVAL   W^MONTHTAB-4[R2],R2      ;GET ADDRESS OF MONTH COUNTED STRING
          FF94 CF     DF        00A7           PUSHAL  W^DATE                   ;BUILD DESCRIPTOR FOR CONTROL STRING
                0E    DD        00AB           PUSHL   #DELTA-DATE              ;
                06    11        00AD           BRB     30$                      ;
                                00AF
                                00AF   ;
                                00AF   ; CONVERT DELTA TIME
                                00AF   ;
                                00AF
          FF9A CF     DF        00AF   20$:    PUSHAL  W^DELTA                  ;BUILD CONTROL STRING DESCRIPTOR
                05    DD        00B3           PUSHL   #TIME-DELTA              ;
          51    5E    D0        00B5   30$:    MOVL    SP,R1                    ;COPY ADDRESS OF CONTROL STRING DESCRIPTOR
                                00B8           $FAO_S  (R1),(R5),(R6),DAY(R4),R2,YEAR(R4) ;CONVERT DELTA TIME OR DATE
                64    DD        00B8                   PUSHL   YEAR(R4)
                52    DD        00BA                   PUSHL   R2
             04 A4    DD        00BC                   PUSHL   DAY(R4)
                66    7F        00BF                   PUSHAQ  (R6)
                65    3F        00C1                   PUSHAW  (R5)
                61    7F        00C3                   PUSHAQ  (R1)
 00000000'GF    06    FB        00C5                   CALLS   #$$T2,G^SYS$FAO
             36 50    E9        00CC           BLBC    R0,60$                   ;IF LBC CONVERT FAILURE
          66    65    A2        00CF           SUBW    (R5),(R6)                ;ANY SPACE LEFT IN TIME BUFFER?
                27    15        00D2           BLEQ    50$                      ;IF LEQ NO
       04 A6    65    C0        00D4           ADDL    (R5),4(R6)               ;UPDATE TIME BUFFER ADDRESS
                                00D8
                                00D8   ;
                                00D8   ; CONVERT TIME
                                00D8   ;
                                00D8
          FF76 CF     DF        00D8   40$:    PUSHAL  W^TIME                   ;BUILD CONTROL STRING DESCRIPTOR
                13    DD        00DC           PUSHL   #EXE$ASCTIM-TIME         ;
          51    5E    D0        00DE           MOVL    SP,R1                    ;COPY ADDRESS OF CONTROL STRING DESCRIPTOR
                                00E1           $FAO_S  (R1),2(R5),(R6),HOUR(R4),MINUTE(R4),SECOND(R4),HUNDREDTH(R4) ;
             0C A4    DD        00E1                   PUSHL   HUNDREDTH(R4)
             0A A4    DD        00E4                   PUSHL   SECOND(R4)
             08 A4    DD        00E7                   PUSHL   MINUTE(R4)
             06 A4    DD        00EA                   PUSHL   HOUR(R4)
                66    7F        00ED                   PUSHAQ  (R6)
             02 A5    3F        00EF                   PUSHAW  2(R5)
                61    7F        00F2                   PUSHAQ  (R1)
 00000000'GF    07    FB        00F4                   CALLS   #$$T2,G^SYS$FAO
       51    04 AC    D0        00FB   50$:    MOVL    ATIMLEN(AP),R1           ;LENGTH ADDRESS SPECIFIED?
                04    13        00FF           BEQL    60$                      ;IF EQL NO
    61    65    85    A1        0101           ADDW3   (R5)+,(R5),(R1)          ;COMPUTE AND RETURN OUTPUT LENGTH
                      04        0105   60$:    RET                              ;
```

```
                                0106   314          .SBTTL  CONVERT ASCII STRING TO BINARY TIME
                                0106   315 ;+
                                0106   316 ; EXE$BINTIM - CONVERT ASCII STRING TO BINARY TIME
                                0106   317 ;
                                0106   318 ; THIS SERVICE PROVIDES THE CAPABILITY TO CONVERT AN ASCII STRING TO A
                                0106   319 ; 64-BIT ABSOLUTE OR DELTA TIME.
                                0106   320 ;
                                0106   321 ; INPUTS:
                                0106   322 ;
                                0106   323 ;       BTIMBUF(AP) = ADDRESS OF ASCII STRING DESCRIPTOR.
                                0106   324 ;       BTIMADR(AP) = ADDRESS TO STORE 64-BIT TIME VALUE.
                                0106   325 ;
                                0106   326 ; OUTPUTS:
                                0106   327 ;
                                0106   328 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CONVERT TIME TO ASCII.
                                0106   329 ;
                                0106   330 ;               R0 = SS$_IVTIME - ASCII STRING HAS INVALID SYNTAX OR TIME
                                0106   331 ;                    COMPONENT IS OUT OF RANGE.
                                0106   332 ;
                                0106   333 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0106   334 ;
                                0106   335 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0106   336 ;-
                                0106   337
                                0106   338 EXE$BINTIM::                                ;CONVERT ASCII STRING TO BINARY TIME
                         01FC   0106   339          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8> ;ENTRY MASK
                5E      10  C2 0108   340          SUBL    #<<<7*2>+3>/4>*4,SP      ;ALLOCATE NUMERIC TIME BUFFER
                57      5E  D0 010B   341          MOVL    SP,R7                    ;SAVE ADDRESS OF NUMERIC TIME BUFFER
            55    04    BC  7D 010E   342          MOVQ    @BTIMBUF(AP),R5          ;GET ADDRESS AND LENGTH OF ASCII STRING
                  58    D4  0112   343          CLRL    R8                       ;ASSUME DELTA TIME
                  55    B7  0114   344 10$:     DECW    R5                       ;ANY MORE CHARACTERS?
                  6F    19  0116   345          BLSS    30$                      ;IF LSS NO
                86    20    91 0118   346          CMPB    #BLANK,(R6)+             ;SKIP LEADING BLANK?
                      F7    13 011B   347          BEQL    10$                      ;IF EQL YES
                      55    B6 011D   348          INCW    R5                       ;CORRECT NUMBER OF CHARACTERS
                            011F   349 ;+
                            011F   350 ;       REPLACED        LOCC    #HYPHEN,R5,-(R6)         ;ABSOLUTE TIME FORMAT?
                            011F   351 ;-
                51    76    9E 011F   352          MOVAB   -(R6),R1
                50    55    3C 0122   353          MOVZWL  R5,R0
                      0C    13 0125   354          BEQL    10002$
                            0127   355 10000$:
                2D    81    91 0127   356          CMPB    (R1)+,#HYPHEN
                      05    13 012A   357          BEQL    10001$
                  F8  50    F5 012C   358          SOBGTR  R0,10000$
                      02    11 012F   359          BRB     10002$
                            0131   360 10001$:
                      51    D7 0131   361          DECL    R1
                            0133   362 10002$:
                      52    13 0133   363          BEQL    30$                      ;IF EQL NO
                      58    D6 0135   364          INCL    R8                       ;INDICATE ABSOLUTE TIME
                            0137   365          $NUMTIM_S (R7)                   ;CONVERT CURRENT TIME TO NUMERIC FORMAT
                      00    DD 0137                PUSHL   #0
                      67    DF 0139                PUSHAL  (R7)
00000000'GF           02    FB 013B                CALLS   #2,G^SYS$NUMTIM
                            0142   366
                            0142   367 ;
```

```
                                0142   368 ; CONVERT ABSOLUTE TIME
                                0142   369 ;
                                0142   370
          54    04 A7    DE     0142   371         MOVAL   DAY(R7),R4              ;SET ADDRESS TO STORE DAY
                6B    10       0146   372         BSBB    CONVERT                 ;CONVERT DAY FIELD
                2D    2D       0148   373         .BYTE   HYPHEN                  ;EXPECTED TERMINATOR
                55    B5       0149   374         TSTW    R5                      ;ANY MORE CHARACTERS?
                64    13       014B   375         BEQL    60$                     ;IF EQL NO
             86 2D    91       014D   376         CMPB    #HYPHEN,(R6)+           ;MONTH FIELD VOID?
                2E    13       0150   377         BEQL    20$                     ;IF EQL YES
                               0152   378 ;+
                               0152   379 ;       REPLACED        MATCHC  #3,-(R6),#4*12,W^MONTHTAB ;SEARCH FOR MONTH SUBSTRIN
                               0152   380 ;-
             51    76    9E     0152   381         MOVAB   -(R6),R1
          50 61    08    78     0155   382         ASHL    #8,(R1),R0
             50    03    90     0159   383         MOVB    #3,R0
                52    D4       015C   384         CLRL    R2                      ; Clear month index
                               015E   385
       FEA8 CF42    50    D1     015E   386 15$:    CMPL    R0,MONTHTAB[R2]         ; Correct month?
                06    13       0164   387         BEQL    19$                     ; Branch if so
             F4 52    0C    F2     0166   388         AOBLSS  #12,R2,15$              ; Increment month and branch if more
                74    11       016A   389         BRB     IVTIME                  ; Month not found, flag invalid
                               016C   390
       02 A7    52    01    A1     016C   391 19$:    ADDW3   #1,R2,MONTH(R7)         ;CONVERT TO MONTH AND STORE
                56    03    C0     0171   392         ADDL    #3,R6                   ;UPDATE ADDRESS OF ASCII STRING
                55    03    A2     0174   393         SUBW    #3,R5                   ;UPDATE COUNT OF REMAINING CHARACTERS
                67    19       0177   394         BLSS    IVTIME                  ;IF LSS INVALID SYNTAX
                36    13       0179   395         BEQL    60$                     ;IF EQL END OF STRING
             86 2D    91       017B   396         CMPB    #HYPHEN,(R6)+           ;FIELD TERMINATED PROPERLY?
                60    12       017E   397         BNEQ    IVTIME                  ;IF NEQ NO
                55    B7       0180   398 20$:    DECW    R5                      ;DECREMENT COUNT OF REMAINING CHARACTERS
          54    67    DE       0182   399         MOVAL   YEAR(R7),R4             ;SET ADDRESS TO STORE YEAR
                0A    11       0185   400         BRB     40$                     ;
                               0187   401
                               0187   402 ;
                               0187   403 ; CONVERT DELTA TIME
                               0187   404 ;
                               0187   405
          54    67    DE       0187   406 30$:    MOVAL   YEAR(R7),R4             ;GET ADDRESS TO STORE YEAR
                84    D4       018A   407         CLRL    (R4)+                   ;CLEAR YEAR AND MONTH
                64    7C       018C   408         CLRQ    (R4)                    ;CLEAR DAY, HOUR, MINUTE, AND SECOND
             0C A7    B4       018E   409         CLRW    HUNDREDTH(R7)           ;CLEAR HUNDREDTH
                20    10       0191   410 40$:    BSBB    CONVERT                 ;CONVERT RELATIVE DAY OR YEAR FIELD
                20    03       0193   411         .BYTE   BLANK                   ;EXPECTED TERMINATOR
                55    B7       0194   412 50$:    DECW    R5                      ;ANY REMAINING CHARACTERS?
                19    19       0196   413         BLSS    60$                     ;IF LSS NO
             86 20    91       0198   414         CMPB    #BLANK,(R6)+            ;NEXT CHARACTER BLANK?
                F7    13       019B   415         BEQL    50$                     ;IF EQL YES
                56    D7       019D   416         DECL    R6                      ;BACK UP TO NONBLANK CHARACTER
                55    D6       019F   417         INCL    R5                      ;ADJUST REMAINING CHARACTER COUNT
                               01A1   418
                               01A1   419 ;
                               01A1   420 ; CONVERT TIME
                               01A1   421 ;
                               01A1   422
       54    06 A7    DE       01A1   423         MOVAL   HOUR(R7),R4             ;SET ADDRESS TO STORE HOUR
                0C    10       01A5   424         BSBB    CONVERT                 ;CONVERT HOUR FIELD
```

```
                        3A  01A7   425          .BYTE   COLON                   ;EXPECTED TERMINATOR
                    09  10  01A8   426          BSBB    CONVERT                 ;CONVERT MINUTE FIELD
                        3A  01AA   427          .BYTE   COLON                   ;EXPECTED TERMINATOR
                    06  10  01AB   428          BSBB    CONVERT                 ;CONVERT SECOND FIELD
                        2E  01AD   429          .BYTE   PERIOD                  ;EXPECTED TERMINATOR
                    03  10  01AE   430          BSBB    CONVERT                 ;CONVERT HUNDREDTH FIELD
                        20  01B0   431          .BYTE   BLANK                   ;EXPECTED TERMINATOR
                    33  11  01B1   432 60$:     BRB     CVRTIME                 ;
                            01B3   433
                            01B3   434 ;
                            01B3   435 ; SUBROUTINE TO CONVERT NUMERIC FIELD TO BINARY
                            01B3   436 ;
                            01B3   437
                            01B3   438 CONVERT:                                 ;CONVERT FIELD
                        50  D4  01B3   439          CLRL    R0                  ;CLEAR ACCUMULATED VALUE
                        84  B5  01B5   440 10$:     TSTW    (R4)+               ;POINT PAST NEXT FIELD
                        55  B7  01B7   441          DECW    R5                  ;ANY MORE CHARACTERS?
                        2B  19  01B9   442          BLSS    CVRTIME             ;IF LSS NO
                51      86  9A  01BB   443          MOVZBL  (R6)+,R1            ;GET NEXT CHARACTER
             00 BE      51  91  01BE   444          CMPB    R1,a(SP)            ;EXPECTED TERMINATOR?
                        19  13  01C2   445          BEQL    20$                 ;IF EQL YES
                51      30  C2  01C4   446          SUBL    #ONE,R1             ;SUBTRACT OUT CHARACTER BIAS
                        17  19  01C7   447          BLSS    IVTIME              ;IF LSS INVALID CHARACTER
                51      09  D1  01C9   448          CMPL    #NINE-ONE,R1        ;RESULT VALUE WITHIN RANGE?
                        12  19  01CC   449          BLSS    IVTIME              ;IF LSS INVALID CHARACTER
                50      0A  A4  01CE   450          MULW    #10,R0              ;MULTIPLY PARTIAL RESULT BY 10
                        0D  1D  01D1   451          BVS     IVTIME              ;IF VS INVALID TIME
                50      51  A0  01D3   452          ADDW    R1,R0               ;ACCUMULATE VALUE
                        08  1D  01D6   453          BVS     IVTIME              ;IF VS INVALID TIME VALUE
                74      50  B0  01D8   454          MOVW    R0,-(R4)            ;STORE VALUE
                        D8  11  01DB   455          BRB     10$                 ;
                        6E  D6  01DD   456 20$:     INCL    (SP)                ;INCREMENT PAST TERMINATOR
                            05  01DF   457          RSB                         ;
                            01E0   458
                            01E0   459 ;
                            01E0   460 ; INVALID SYNTAX OR TIME COMPONENT
                            01E0   461 ;
                            01E0   462
                50    0184  8F  3C  01E0   463 IVTIME: MOVZWL  #SS$_IVTIME,R0  ;SET INVALID TIME
                            04  01E5   464          RET                         ;
                            01E6   465
                            01E6   466 ;
                            01E6   467 ; CHECK CONVERTED DATE AND TIME VALUES
                            01E6   468 ;
                            01E6   469
                            01E6   470 CVRTIME:                                 ;
          04 A7     270F  8F  B1  01E6   471          CMPW    #9999,DAY(R7)       ;DAY WITHIN UPPER LIMIT?
                        F2  1F  01EC   472          BLSSU   IVTIME              ;IF LSSU NO
             06 A7      18  B1  01EE   473          CMPW    #24,HOUR(R7)        ;HOUR WITHIN LIMITS?
                        EC  1B  01F2   474          BLEQU   IVTIME              ;IF LEQU NO
             08 A7      3C  B1  01F4   475          CMPW    #60,MINUTE(R7)      ;MINUTE WITHIN LIMITS?
                        E6  1B  01F8   476          BLEQU   IVTIME              ;IF LEQU NO
             0A A7      3C  B1  01FA   477          CMPW    #60,SECOND(R7)      ;SECOND WITHIN LIMITS?
                        E0  1B  01FE   478          BLEQU   IVTIME              ;IF LEQU NO
          0C A7     0064  8F  B1  0200   479          CMPW    #100,HUNDREDTH(R7)  ;HUNDREDTH WITHIN LIMITS?
                        D8  1B  0206   480          BLEQU   IVTIME              ;IF LEQU NO
               55     04  A7  3C  0208   481          MOVZWL  DAY(R7),R5          ;GET DAY VALUE
```

```
                                03 58    E8  020C    482          BLBS    R8,5$                   ;IF LBS ABSOLUTE TIME
                                 0097    31  020F    483          BRW     40$                     ;
                                         0212    484
                                         0212    485 ;
                                         0212    486 ; CONVERT YEARS TO QUADRICENTURIES, CENTURIES, QUADYEARS, YEARS
                                         0212    487 ;
                                         0212    488
                                    CC   13  0212    489 5$:      BEQL    IVTIME                  ;IF EQL INVALID TIME
                                50  67   3C  0214    490          MOVZWL  YEAR(R7),R0             ;GET YEAR VALUE
                            50  F9BF C0   3E  0217    491          MOVAW   -1601(R0),R0            ;CALCULATE YEARS PAST 1601
                                    C2   19  021C    492          BLSS    IVTIME                  ;IF LSS INVALID TIME
                                    51   D4  021E    493          CLRL    R1                      ;CLEAR HIGH PART OF DIVIDEND
   51   50   50   00000190 8F   7B  0220    494          EDIV    #400,R0,R0,R1           ;CALCULATE QUADRICENTURIES
                                    52   D4  0229    495          CLRL    R2                      ;CLEAR HIGH PART OF DIVIDEND
   52   51   51   00000064 8F   7B  022B    496          EDIV    #100,R1,R1,R2           ;CALCULATE CENTURIES
                                    53   D4  0234    497          CLRL    R3                      ;CLEAR HIGH PART OF DIVIDEND
        53   52   52   04   7B  0236    498          EDIV    #4,R2,R2,R3             ;CALCULATE QUADYEARS AND YEARS
                                         023B    499
                                         023B    500 ;
                                         023B    501 ; CONVERT QUADRICENTURIES, CENTURIES, QUADYEARS, YEARS TO DAYS
                                         023B    502 ;
                                         023B    503
                            53  016D 8F   A4  023B    504          MULW    #365,R3                 ;CALCULATE NUMBER OF DAYS PAST LEAP YEAR
   52   53   52   000005B5 8F   7A  0240    505          EMUL    #QUADYEARDAYS,R2,R3,R2  ;CALCULATE NUMBER OF QUADYEAR DAYS AND SUM
             51   00008EAC 8F   C4  0249    506          MULL    #CENTURYDAYS,R1         ;CALCULATE NUMBER OF CENTURY DAYS
   55   51   50   00023AB1 8F   7A  0250    507          EMUL    #QUADRIDAYS,R0,R1,R5    ;CALCULATE NUMBER OF QUADRIDAYS AND SUM
                                    50   D4  0259    508          CLRL    R0                      ;CLEAR INITIAL LOOP INDEX
                           56   02  A7   3C  025B    509          MOVZWL  MONTH(R7),R6            ;GET SPECIFIED MONTH VALUE
                                55  52   C0  025F    510 10$:     ADDL    R2,R5                   ;ACCUMULATE TOTAL DAYS
                       52   FD99 CF40   9A  0262    511          MOVZBL  W^DATETABLE[R0],R2      ;GET NUMBER OF DAYS IN MONTH
                                50  01   D1  0268    512          CMPL    #1,R0                   ;SECOND MONTH OF YEAR?
                                    1E   12  026B    513          BNEQ    30$                     ;IF NEQ NO
                                53  67   3C  026D    514          MOVZWL  YEAR(R7),R3             ;GET SPECIFIED YEAR VALUE
                                53  03   D3  0270    515          BITL    #3,R3                   ;YEAR MULTIPLE OF 4?
                                    14   12  0273    516          BNEQ    20$                     ;IF NEQ NO
                                    54   D4  0275    517          CLRL    R4                      ;CLEAR HIGH PART OF DIVIDEND
   54   53   53   00000064 8F   7B  0277    518          EDIV    #100,R3,R3,R4           ;CALCULATE CENTURY AND YEAR IN CENTURY
                                    54   D5  0280    519          TSTL    R4                      ;YEAR MULTIPLE OF 100?
                                    07   12  0282    520          BNEQ    30$                     ;IF NEQ NO
                                53  03   D3  0284    521          BITL    #3,R3                   ;YEAR MULTIPLE OF 400?
                                    02   13  0287    522          BEQL    30$                     ;IF EQL YES
                                    52   D7  0289    523 20$:     DECL    R2                      ;REDUCE NUMBER OF DAYS IN MONTH
                            D0  50   56   F2  028B    524 30$:     AOBLSS  R6,R0,10$               ;ANY MORE DAYS TO ACCUMULATE?
                       50   0184 8F   3C  028F    525          MOVZWL  #SS$_IVTIME,R0          ;ASSUME INVALID DAY OF MONTH
                        51   04   A7   3C  0294    526          MOVZWL  DAY(R7),R1              ;GET SPECIFIED DAY
                   55   00016FEC 8F   C2  0298    527          SUBL    #TIMOFF2,R5             ;SUBTRACT OUT NUMBER OF DAYS TO 17-NOV-1858
                                55  51   C0  029F    528          ADDL    R1,R5                   ;CALCULATE TOTAL NUMBER OF DAYS
                                    57   19  02A2    529          BLSS    60$                     ;IF LSS INVALID TIME
                                52  51   D1  02A4    530          CMPL    R1,R2                   ;DAY WITHIN LIMITS?
                                    52   1A  02A7    531          BGTRU   60$                     ;IF GTRU NO
                                         02A9    532
                                         02A9    533 ;
                                         02A9    534 ; CONVERT TIME TO TENTHS OF MICROSECONDS
                                         02A9    535 ;
                                         02A9    536
                            50   06  A7   3C  02A9    537 40$:     MOVZWL  HOUR(R7),R0             ;GET HOUR VALUE
                            51   08  A7   3C  02AD    538          MOVZWL  MINUTE(R7),R1           ;GET MINUTE VALUE
```

```
                50   51   50    3C   7A  02B1   539          EMUL    #60,R0,R1,R0            ;CONVERT HOURS TO MINUTES AND SUM
                       51    0A A7   3C  02B6   540          MOVZWL  SECOND(R7),R1           ;GET SECOND VALUE
                50   51   50    3C   7A  02BA   541          EMUL    #60,R0,R1,R0            ;CONVERT MINUTES TO SECONDS AND SUM
                       51    0C A7   3C  02BF   542          MOVZWL  HUNDREDTH(R7),R1        ;GET HUNDREDTH VALUE
     50   51   50    00000064 8F   7A  02C3   543          EMUL    #100,R0,R1,R0           ;CONVERT SECONDS TO HUNDREDTHS AND SUM
     50   00   50    000186A0 8F   7A  02CC   544          EMUL    #100000,R0,#0,R0        ;CONVERT TO TENTHS OF MICROSECONDS
                                        02D5   545
                                        02D5   546 ;
                                        02D5   547 ; CONVERT DAYS TO TENTHS OF MICROSECONDS
                                        02D5   548 ;
                                        02D5   549
     52   00   55    324A9A70 8F   7A  02D5   550          EMUL    #843750000,R5,#0,R2     ;MULTIPLY BY 864000000000/1024
                52   52    0A   79  02DE   551          ASHQ    #10,R2,R2               ;MULTIPLY BY 1024
                                        02E2   552
                                        02E2   553 ;
                                        02E2   554 ; COMBINE RESULTS AND STORE 64-BIT TIME
                                        02E2   555 ;
                                        02E2   556
                     52   50   C0   02E2   557          ADDL    R0,R2                   ;ADD LOW ORDER PARTS
                     53   51   D8   02E5   558          ADWC    R1,R3                   ;ADD HIGH ORDER PARTS
                     50   01   3C   02E8   559          MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION
                       09 58   E8   02EB   560          BLBS    R8,50$                  ;IF LBS ABSOLUTE TIME
                     53   53   CE   02EE   561          MNEGL   R3,R3                   ;CONVERT TO DELTA TIME
                     52   52   CE   02F1   562          MNEGL   R2,R2                   ;
                     53   00   D9   02F4   563          SBWC    #0,R3                   ;
                08 BC   52   7D   02F7   564 50$:       MOVQ    R2,@BTIMADR(AP)         ;STORE 64-BIT TIME VALUE
                             04   02FB   565 60$:       RET                             ;
```

```
                                02FC    567          .SBTTL  CONVERT BINARY TIME TO NUMERIC TIME
                                02FC    568 ;+
                                02FC    569 ; EXE$NUMTIM - CONVERT BINARY TIME TO NUMERIC TIME
                                02FC    570 ;
                                02FC    571 ; THIS SERVICE PROVIDES THE CAPABILITY TO CONVERT AN ABSOLUTE OR DELTA TIME
                                02FC    572 ; FROM 64-BIT FORMAT TO INTEGER DATE AND TIME VALUES.
                                02FC    573 ;
                                02FC    574 ; INPUTS:
                                02FC    575 ;
                                02FC    576 ;       NTIMBUF(AP) = ADDRESS OF 7-WORD BUFFER TO RECEIVE CONVERTED DATE AND
                                02FC    577 ;                     TIME VALUES.
                                02FC    578 ;       NTIMADR(AP) = ADDRESS OF 64-BIT TIME VALUE. IF ZERO, THEN THE CURRENT
                                02FC    579 ;                     SYSTEM TIME IS USED. POSITIVE VALUES ARE INTERPRETED AS
                                02FC    580 ;                     ABSOLUTE TIMES AND NEGATIVE VALUES AS DELTA TIMES.
                                02FC    581 ;
                                02FC    582 ; OUTPUTS:
                                02FC    583 ;
                                02FC    584 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CONVERT TO NUMERIC TIME.
                                02FC    585 ;
                                02FC    586 ;                R0 = SS$_ACCVIO - 64-BIT TIME VALUE CANNOT BE READ BY CALLING
                                02FC    587 ;                     ACCESS MODE OR TIME BUFFER CANNOT BE WRITTEN BY
                                02FC    588 ;                     CALLING ACCESS MODE.
                                02FC    589 ;
                                02FC    590 ;                R0 = SS$_IVTIME - SPECIFIED DELTA TIME IS GREATER THAN 9999
                                02FC    591 ;                     DAYS.
                                02FC    592 ;
                                02FC    593 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                02FC    594 ;
                                02FC    595 ;                R0 = SS$_NORMAL - NORMAL COMPLETION.
                                02FC    596 ;-
                                02FC    597
                                02FC    598 EXE$NUMTIM::                                ;CONVERT TO NUMERIC TIME
                          00FC  02FC    599          .WORD   ^M<R2,R3,R4,R5,R6,R7>      ;ENTRY MASK
                                02FE    600
                                02FE    601          $MP_SET_INTERLOCK  #MP$V_IL_NUMT ; Interlock this routine  [04]
                       00   DD  02FE                 PUSHL   #MP$V_IL_NUMT
                       08   DD  0300                 PUSHL   #$SRVCODE_SET_INTERLOCK
00000000'9F            02   FB  0302                 CALLS   #2, a#DS$SERVICE_DISPATCHER
                                0309    602
             57   04 AC   D0  0309    603          MOVL    NTIMBUF(AP),R7             ;GET ADDRESS OF 7-WORD TIME BUFFER
                                030D    604          IFNOWRT #7*2,(R7),10$              ;CAN TIME BUFFER BE WRITTEN?
          67   0E   00   0D  030D                 PROBEW  #0,#7*2,(R7)
                       28   13  0311                 BEQL    10$
                  50   01   3C  0313    605          MOVZWL  #SS$_NORMAL,R0             ;ASSUME NORMAL COMPLETION
51    00000000'EF   7D  0316    606          MOVQ    EXE$GQ_SYSTIME,R1          ;ASSUME TIME NOT SPECIFIED
             53   08 AC   D0  031D    607          MOVL    NTIMADR(AP),R3             ;GET ADDRESS OF 64-TIME VALUE
                       1E   13  0321    608          BEQL    20$                        ;IF EQL NONE SPECIFIED
                                0323    609          IFNORD  #8,(R3),10$                ;CAN 64-BIT TIME VALUE BE READ?
          63   08   00   0C  0323                 PROBER  #0,#8,(R3)
                       12   13  0327                 BEQL    10$
                  51   63   7D  0329    610          MOVQ    (R3),R1                    ;GET 64-BIT TIME VALUE
                       13   18  032C    611          BGEQ    20$                        ;IF GEQ ABSOLUTE TIME
                  52   52   CE  032E    612          MNEGL   R2,R2                      ;NEGATE DELTA TIME VALUE
                  51   51   CE  0331    613          MNEGL   R1,R1                      ;
                  52   00   D9  0334    614          SBWC    #0,R2                      ;
             06   50   00   E4  0337    615          BBSC    #0,R0,20$                  ;INDICATE DELTA TIME VALUE
                  50   0C   3C  033B    616 10$:     MOVZWL  #SS$_ACCVIO,R0             ;SET ACCESS VIOLATION
```

```
                                  033E
                     00D9    31   033E          BRW     91$                     ; JUMP TO CLEAR INTERLOCK AND RET [04]
                                  0341
                                  0341
                                  0341  ;
                                  0341  ; R1 AND R2 CONTAIN 64-BIT ABSOLUTE TIME VALUE IN UNITS OF TENTHS OF MICRO-
                                  0341  ; SECONDS. CALCULATE DAYS PAST BASE TIME AND FRACTION OF DAY BY DIVIDING
                                  0341  ; BY 864000000000 WHICH IS THE NUMBER OF TENTHS OF MICROSECONDS IN A DAY.
                                  0341  ; THE DIVISION IS PERFORMED IN THREE STEPS TO INSURE BOTH QUOTIENT AND
                                  0341  ; REMAINDER STAY WITHIN 32 BITS.
                                  0341  ;
                                  0341  ; CALCULATE DAYS BY DIVIDING BY 1024 AND THEN 843750000. QUOTIENT IS DAYS
                                  0341  ; AND REMAINDER IS FRACTION OF DAY.
                                  0341  ;
                                  0341
          54   51   0A   00   EF  0341  20$:    EXTZV   #0,#10,R1,R4            ;SAVE REMAINDER FROM NEXT DIVIDE
            51   51   F6 8F   79  0346          ASHQ    #-10,R1,R1              ;DIVIDE BY 1024
   52   51   51   324A9A70 8F   7B  034B          EDIV    #843750000,R1,R1,R2     ;CALCULATE DAYS AND FRACTION OF DAY
                                  0354
                                  0354  ;
                                  0354  ; R1 CONTAINS DAYS PAST BASE TIME, R2 PLUS R4 CONTAIN FRACTION OF DAY.
                                  0354  ; R2 CONTAINS PART OF FRACTION IN UNITS OF 864000000000/1024 AND
                                  0354  ; R4 CONTAINS REMAINDER IN UNITS OF TENTHS OF MICROSECONDS.
                                  0354  ;
                                  0354  ; CALCULATE FRACTION OF DAY IN HUNDREDTHS OF SECONDS BY DIVIDING BY
                                  0354  ; 100000 WHICH IS THE NUMBER OF TENTHS OF MICROSECONDS IN A HUNDRETH
                                  0354  ; OF A SECOND.
                                  0354  ;
                                  0354
                        53   D4   0354          CLRL    R3                      ;CLEAR HIGH PART OF DIVIDEND
                52   52   0A   79  0356          ASHQ    #10,R2,R2               ;CONVERT BACK TO TENTHS OF MICROSECONDS
                     52   54   C8  035A          BISL    R4,R2                   ;ADD REMAINDER BACK
   52   55   52   000186A0 8F   7B  035D          EDIV    #100000,R2,R5,R2        ;CALCULATE FRACTION OF DAY IN HUNDREDTHS
                                  0366
                                  0366  ;
                                  0366  ; R1 CONTAINS DAYS PAST THE BASE TIME AND R5 CONTAINS THE FRACTION OF DAY
                                  0366  ; IN HUNDREDTHS OF A SECOND.
                                  0366  ;
                                  0366
                 7E 50   00   E3  0366          BBCS    #0,R0,70$               ;IF CLR, DELTA TIME SPECIFIED
                                  036A
                                  036A  ;
                                  036A  ; ADD TIME OFFSET SO THAT DAY IS RELATIVE TO 1-JAN-1501.
                                  036A  ;
                                  036A
          51   0001FE98 8F   C0  036A          ADDL    #TIMOFF1,R1             ;ADD TIME OFFSET
                                  0371
                                  0371  ;
                                  0371  ; CALCULATE NUMBER OF QUADRICENTURIES THAT HAVE PAST SINCE 1501.
                                  0371  ;
                                  0371
                        52   D4   0371          CLRL    R2                      ;CLEAR HIGH PART OF DIVIDEND
   52   51   51   00023AB1 8F   7B  0373          EDIV    #QUADRIDAYS,R1,R1,R2    ;CALCULATE NUMBER OF QUADRICENTURIES
                                  037C
                                  037C  ;
                                  037C  ; R1 CONTAINS THE NUMBER OF QUADRICENTURIES AND R2 CONTAINS THE NUMBER OF
                                  037C  ; DAYS INTO THE NEXT QUADRICENTURY. CALCULATE THE NUMBER OF CENTURIES BY
```

```
                                037C   674 ; CONVERTING TO QUARTER DAYS INTO NEXT QUADRICENTURY AND THEN DIVIDING BY
                                037C   675 ; THE AVERAGE NUMBER OF QUARTER DAYS IN A CENTURY.
                                037C   676 ;
                                037C   677
                          52    04   C4  037C   678         MULL    #4,R2                   ;CALCULATE NUMBER OF QUARTER DAYS
                                53   D4  037F   679         CLRL    R3                      ;CLEAR HIGH PART OF DIVIDEND
    53    52    52    00023AB1  8F   7B  0381   680         EDIV    #QDAYSPCENT,R2,R2,R3    ;CALCULATE NUMBER OF CENTURIES
                                038A   681
                                038A   682 ;
                                038A   683 ; R2 CONTAINS THE NUMBER OF CENTURIES AND R3 CONTAINS THE NUMBER OF QUARTER
                                038A   684 ; DAYS INTO THE NEXT CENTURY.
                                038A   685 ;
                                038A   686 ; CALCULATE YEARS BY DISCARDING ANY FRACTION OF A DAY, ADDING 3/4'THS OF A
                                038A   687 ; DAY, AND DIVIDING BY THE AVERAGE NUMBER OF DAYS IN A YEAR. THE LEAP DAY
                                038A   688 ; OF EACH FOUR YEAR CYCLE IS FORCED INTO THE FOURTH YEAR.
                                038A   689 ;
                                038A   690
                                54   D4  038A   691         CLRL    R4                      ;CLEAR HIGH PART OF DIVIDEND
                          53    03   C8  038C   692         BISL    #3,R3                   ;TRUNCATE FRACTION AND ADD 3/4'THS OF DAY
    54    53    53    000005B5  8F   7B  038F   693         EDIV    #QDAYSPYEAR,R3,R3,R4    ;CALCULATE NUMBER OF YEARS
                          54    04   C6  0398   694         DIVL    #4,R4                   ;CALCULATE NUMBER OF DAYS MINUS ONE
                                54   D6  039B   695         INCL    R4                      ;CONVERT TO ACTUAL JULIAN DAY OF YEAR
                                039D   696
                                039D   697 ;
                                039D   698 ; R1 CONTAINS NUMBER OF QUADRICENTURIES.
                                039D   699 ; R2 CONTAINS NUMBER OF CENTURIES.
                                039D   700 ; R3 CONTAINS NUMBER OF YEARS.
                                039D   701 ; R4 CONTAINS JULIAN DAY OF YEAR.
                                039D   702 ;
                                039D   703 ; CALCULATE ACTUAL YEAR.
                                039D   704 ;
                                039D   705
                    51    6241    DE  039D   706         MOVAL   (R2)[R1],R1             ;COMBINE CENTURIES AND QUADRICENTURIES
                      51    32    C4  03A1   707         MULL    #50,R1                  ;CALCULATE NUMBER OF DOUBLE CENTURIES
                51    05DD C341   3E  03A4   708         MOVAW   1501(R3)[R1],R1         ;CALCULATE ACTUAL YEAR
                      87    51    B0  03AA   709         MOVW    R1,(R7)+                ;STORE YEAR
                                03AD   710
                                03AD   711 ;
                                03AD   712 ; TEST FOR NONLEAP YEAR AND BIAS DAY IF AFTER 28-FEB.
                                03AD   713 ;
                                03AD   714
                          51    03   D3  03AD   715         BITL    #3,R1                   ;YEAR MULTIPLE OF 4?
                                14   12  03B0   716         BNEQ    30$                     ;IF NEQ NO
                                52   D4  03B2   717         CLRL    R2                      ;CLEAR HIGH PART OF DIVIDEND
    52    51    51    00000064  8F   7B  03B4   718         EDIV    #100,R1,R1,R2           ;CALCULATE CENTURY AND YEAR IN CENTURY
                                52   D5  03BD   719         TSTL    R2                      ;YEAR MULTIPLE OF 100?
                                0C   12  03BF   720         BNEQ    40$                     ;IF NEQ NO
                          51    03   D3  03C1   721         BITL    #3,R1                   ;YEAR MULTIPLE OF 400?
                                07   13  03C4   722         BEQL    40$                     ;IF EQL YES
                          54    3B   D1  03C6   723 30$:    CMPL    #31+28,R4               ;AFTER 28-FEB?
                                02   18  03C9   724         BGEQ    40$                     ;IF GEQ NO
                                54   D6  03CB   725         INCL    R4                      ;ADJUST FOR TABLE BIAS
                          51    01   D0  03CD   726 40$:    MOVL    #1,R1                   ;INITIALIZE MONTH
                52    FC2A CF41   9A  03D0   727 50$:    MOVZBL  W^DATETABLE-1[R1],R2    ;GET NUMBER OF DAYS IN MONTH
                      54    52    C2  03D6   728         SUBL    R2,R4                   ;SUBTRACT FROM JULIAN DAY
                                04   15  03D9   729         BLEQ    60$                     ;IF LEQ CORRECT MONTH FOUND
                   F1 51    0C   F3  03DB   730         AOBLEQ  #12,R1,50$              ;LOOP THROUGH ALL MONTHS
```

```
                87   51   B0   03DF   731 60$:    MOVW    R1,(R7)+                ;STORE MONTH
             87 54   52   A1   03E2   732         ADDW3   R2,R4,(R7)+             ;STORE DAY
                16   11   03E6   733         BRB     80$                     ;
                          03E8   734
                          03E8   735 ;
                          03E8   736 ; DELTA TIME SPECIFIED - STORE RELATIVE DAY
                          03E8   737 ;
                          03E8   738
                87   D4   03E8   739 70$:    CLRL    (R7)+                   ;CLEAR YEAR AND MONTH
             87   51   B0   03EA   740         MOVW    R1,(R7)+                ;STORE DAY
          51   00002710 8F   D1   03ED   741         CMPL    #10000,R1               ;RELATIVE DAY WITHIN LIMITS?
                08   1A   03F4   742         BGTRU   80$                     ;IF GTRU YES
          50   0184 8F   3C   03F6   743         MOVZWL  #SS$_IVTIME,R0          ;SET INVALID TIME
                          03FB   744
                001C   31   03FB   745         BRW     91$                     ; JUMP TO CLEAR INTERLOCK AND RET [04]
                          03FE   746
                          03FE   747 ;
                          03FE   748 ; R5 CONTAINS FRACTION OF DAY IN HUNDREDTHS OF SECONDS.
                          03FE   749 ;
                          03FE   750 ; CALCULATE HOUR, MINUTE, SECOND, AND HUNDREDTH OF SECOND.
                          03FE   751 ;
                          03FE   752
                57   08   C0   03FE   753 80$:    ADDL    #8,R7                   ;POINT TWO BYTES PAST END OF BUFFER
                51   D4   0401   754         CLRL    R1                      ;CLEAR LOOP INDEX
          52   FC34 CF41   9A   0403   755 90$:    MOVZBL  W^TIMETABLE[R1],R2      ;GET NEXT UNIT DIVISOR
                56   D4   0409   756         CLRL    R6                      ;CLEAR HIGH PART OF DIVIDEND
       56   55   55   52   7B   040B   757         EDIV    R2,R5,R5,R6             ;CALCULATE NEXT PART
                77   56   B0   0410   758         MOVW    R6,-(R7)                ;STORE NEXT PART
             EC 51   02   F3   0413   759         AOBLEQ  #2,R1,90$               ;LOOP FOR HUNDREDTHS, SECONDS, AND MINUTES
                77   55   B0   0417   760         MOVW    R5,-(R7)                ;STORE HOUR
                          041A   761 91$:
                          041A   762
                          041A   763         $MP_CLEAR_INTERLOCK  #MP$V_IL_NUMT ; Clear Interlock       [04]
                00   DD   041A                       PUSHL   #MP$V_IL_NUMT
                09   DD   041C                       PUSHL   #$SRVCODE_CLEAR_INTERLOCK
 00000000'9F    02   FB   041E                       CALLS   #2, @#DS$SERVICE_DISPATCHER
                          0425   764
                04   0425   765         RET                                     ;
                          0426   766
                          0426   767         .END
```

```
$$T2                            = 00000007     D
$SRVCODE_BOOTATTACHED           = 00000000     D
$SRVCODE_CLEAR_INTERLOCK        = 00000009     D
$SRVCODE_GETSYSBUF              = 00000004     D
$SRVCODE_HALTATTACHED           = 00000002     D
$SRVCODE_INTERLOCK_TEST         = 00000007     D
$SRVCODE_PPSCB                  = 0000000A     D
$SRVCODE_PRINTREV               = 00000006     D
$SRVCODE_RELSYSBUF              = 00000005     D
$SRVCODE_SET_INTERLOCK          = 00000008     D
$SRVCODE_SHOWIDLE               = 00000003     D
$SRVCODE_STARTATTACHED          = 00000001     D
ACVTFLG                         = 00000010     D
ATIMADR                         = 0000000C     D
ATIMBUF                         = 00000008     D
ATIMLEN                         = 00000004     D
BLANK                           = 00000020     D
BTIMADR                         = 00000008     D
BTIMBUF                         = 00000004     D
CENTURYDAYS                     = 00008EAC     D
COLON                           = 0000003A     D
CONVERT                           000001B3 R   D   02
CVRTIME                           000001E6 R   D   02
DATE                              0000003F R   D   02
DATETABLE                         00000000 RG  D   02
DAY                             = 00000004     D
DELTA                             0000004D R   D   02
DS$SERVICE_DISPATCHER             ********     X   02
EXE$ASCTIM                        00000065 RG  D   02
EXE$BINTIM                        00000106 RG  D   02
EXE$GQ_SYSTIME                    ********     X   02
EXE$NUMTIM                        000002FC RG  D   02
HOUR                            = 00000006     D
HUNDREDTH                       = 0000000C     D
HYPHEN                          = 0000002D     D
IVTIME                            000001E0 R   D   02
MINUTE                          = 00000008     D
MONTH                           = 00000002     D
MONTHTAB                          0000000C R   D   02
MP$V_IL_ALLN                    = 00000004     D
MP$V_IL_CNDB                    = 00000007     D
MP$V_IL_CNHN                    = 00000006     D
MP$V_IL_DELN                    = 00000005     D
MP$V_IL_DSBK                    = 00000008     D
MP$V_IL_ERSP                    = 00000003     D
MP$V_IL_GTLN                    = 00000009     D
MP$V_IL_NUMT                    = 00000000     D
MP$V_IL_PRNT                    = 00000001     D
MP$V_IL_RRPT                    = 00000002     D
NINE                            = 00000039     D
NTIMADR                         = 00000008     D
NTIMBUF                         = 00000004     D
ONE                             = 00000030     D
PERIOD                          = 0000002E     D
QDAYSPCENT                      = 00023AB1     D
QDAYSPYEAR                      = 000005B5     D
QUADRIDAYS                      = 00023AB1     D
QUADYEARDAYS                    = 000005B5     D
SECOND                          = 0000000A     D
SS$_ACCVIO                      = 0000000C     D
SS$_IVTIME                      = 00000184     D
SS$_NORMAL                      = 00000001     D
SYS$FAO                           ********     X   02
SYS$NUMTIM                        ********    GX   02
TIME                              00000052 R   D   02
TIMETABLE                         0000003C R   D   02
TIMOFF1                         = 0001FE98     D
TIMOFF2                         = 00016FEC     D
YEAR                            = 00000000     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000426 (1062.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T2	=00000007	308 (1)	295 (1) 308 (1)
$SRVCODE_BOOTATTACHED	=00000000	763 (1)	
$SRVCODE_CLEAR_INTERLOCK	=00000009	763 (1)	#-763 (1)
$SRVCODE_GETSYSBUF	=00000004	763 (1)	
$SRVCODE_HALTATTACHED	=00000002	763 (1)	
$SRVCODE_INTERLOCK_TEST	=00000007	763 (1)	
$SRVCODE_PPSCB	=0000000A	763 (1)	
$SRVCODE_PRINTREV	=00000006	763 (1)	
$SRVCODE_RELSYSBUF	=00000005	763 (1)	
$SRVCODE_SET_INTERLOCK	=00000008	763 (1)	#-601 (1)
$SRVCODE_SHOWIDLE	=00000003	763 (1)	
$SRVCODE_STARTATTACHED	=00000001	763 (1)	
ACVTFLG	=00000010	79 (1)	#-275 (1)
ATIMADR	=0000000C	78 (1)	#-261 (1)
ATIMBUF	=00000008	77 (1)	#-256 (1)
ATIMLEN	=00000004	76 (1)	#-309 (1)
BLANK	=00000020	145 (1)	#-346 (1) 411 (1) #-414 (1) 431 (1)
BTIMADR	=00000008	86 (1)	#-564 (1)
BTIMBUF	=00000004	85 (1)	#-342 (1)
CENTURYDAYS	=00008EAC	101 (1)	#-506 (1)
COLON	=0000003A	146 (1)	425 (1) 427 (1)
CONVERT	000001B3-R	438 (1)	#-372 (1) #-410 (1) #-424 (1) #-426 (1)
			#-428 (1) #-430 (1)
CVRTIME	000001E6-R	470 (1)	#-432 (1) #-442 (1)
DATE	0000003F-R	215 (1)	284 (1) #-285 (1)
DATETABLE	00000000-R	170 (1)	#-511 (1) #-727 (1)
DAY	=00000004	158 (1)	#-295 (1) 371 (1) #-471 (1) #-481 (1)
			#-526 (1)
DELTA	0000004D-R	216 (1)	#-285 (1) 292 (1) #-293 (1)
DS$SERVICE_DISPATCHER	00000000-XR		601 (1) 763 (1)
EXE$ASCTIM	00000065-R	254 (1)	#-306 (1)
EXE$BINTIM	00000106-R	338 (1)	
EXE$GQ_SYSTIME	00000000-XR		#-606 (1)
EXE$NUMTIM	000002FC-R	598 (1)	
HOUR	=00000006	159 (1)	#-308 (1) 423 (1) #-473 (1) #-537 (1)
HUNDREDTH	=0000000C	162 (1)	#-308 (1) #-409 (1) #-479 (1) #-542 (1)
HYPHEN	=0000002D	147 (1)	#-356 (1) 373 (1) #-376 (1) #-396 (1)
IVTIME	000001E0-R	463 (1)	#-389 (1) #-394 (1) #-397 (1) #-447 (1)
			#-449 (1) #-451 (1) #-453 (1) #-472 (1)
			#-474 (1) #-476 (1) #-478 (1) #-480 (1)
			#-489 (1) #-492 (1)
MINUTE	=00000008	160 (1)	#-308 (1) #-475 (1) #-538 (1)
MONTH	=00000002	157 (1)	#-282 (1) #-391 (1) #-509 (1)
MONTHTAB	0000000C-R	188 (1)	283 (1) #-386 (1)
MP$V_IL_ALLN	=00000004	60 (1)	
MP$V_IL_CNDB	=00000007	60 (1)	
MP$V_IL_CNHN	=00000006	60 (1)	
MP$V_IL_DELN	=00000005	60 (1)	
MP$V_IL_DSBK	=00000008	60 (1)	
MP$V_IL_ERSP	=00000003	60 (1)	

```
MP$V_IL_GTLN                    =00000009      60      (1)
MP$V_IL_NUMT                    =00000000      60      (1)     #-601    (1)     #-763    (1)
MP$V_IL_PRNT                    =00000001      60      (1)
MP$V_IL_RRPT                    =00000002      60      (1)
NINE                            =00000039     148      (1)     #-448    (1)
NTIMADR                         =00000008      93      (1)     #-607    (1)
NTIMBUF                         =00000004      92      (1)     #-603    (1)
ONE                             =00000030     149      (1)     #-446    (1)     #-448    (1)
PERIOD                          =0000002E     150      (1)       429    (1)
QDAYSPCENT                      =00023AB1     107      (1)     #-680    (1)
QDAYSPYEAR                      =000005B5     113      (1)     #-693    (1)
QUADRIDAYS                      =00023AB1     119      (1)     #-507    (1)     #-669    (1)
QUADYEARDAYS                    =000005B5     125      (1)     #-505    (1)
SECOND                          =0000000A     161      (1)     #-308    (1)     #-477    (1)     #-540    (1)
SS$_ACCVIO                      =0000000C                      #-616    (1)
SS$_IVTIME                      =00000184                      #-463    (1)     #-525    (1)     #-743    (1)
SS$_NORMAL                      =00000001                      #-559    (1)     #-605    (1)
SYS$FAO                          00000000-XR                     295    (1)       308    (1)
SYS$NUMTIM                       00000000-XR                     268    (1)       365    (1)
TIME                             00000052-R    217      (1)     #-293    (1)       305    (1)     #-306    (1)
TIMETABLE                        0000003C-R    206      (1)     #-755    (1)
TIMOFF1                         =0001FE98      132      (1)     #-662    (1)
TIMOFF2                         =00016FEC      139      (1)     #-527    (1)
YEAR                            =00000000      156      (1)     #-295    (1)       399    (1)       406    (1)     #-490    (1)
                                                                #-514    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	59 (1)	59 (1)
$DS_SRVCODE_DEF	2	763 (1)	601 (1) 763 (1)
$FAO_S	2	295 (1)	295 (1) 308 (1)
$MP_CLEAR_INTERLOCK	1	763 (1)	763 (1)
$MP_ILDEFS	1	60 (1)	60 (1)
$MP_SET_INTERLOCK	1	601 (1)	601 (1)
$NUMTIM_S	1	268 (1)	268 (1) 365 (1)
$PUSHADR	1	268 (1)	268 (1) 295 (1) 308 (1) 365 (1)
$SSDEF	21	59 (1)	59 (1)
IFNORD	1	609 (1)	609 (1)
IFNOWRT	1	604 (1)	604 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	299 (1) 392 (1) 510 (1) 528 (1) 557 (1) 662 (1) 753 (1)
ADDW	00A0	452 (1)
ADDW3	00A1	311 (1) 391 (1) 732 (1)
ADWC	00D8	558 (1)
AOBLEQ	00F3	730 (1) 759 (1)
AOBLSS	00F2	388 (1) 524 (1)
ASHL	0078	382 (1)
ASHQ	0079	551 (1) 633 (1) 647 (1)
BBCS	00E3	656 (1)
BBSC	00E4	615 (1)
BEQL	0013	262 (1) 310 (1) 347 (1) 354 (1) 357 (1) 363 (1) 375 (1)
		377 (1) 387 (1) 395 (1) 415 (1) 445 (1) 489 (1) 522 (1)
		604 (1) 608 (1) 609 (1) 722 (1)
BGEQ	0018	264 (1) 611 (1) 724 (1)
BGTRU	001A	531 (1) 742 (1)
BISL	00C8	648 (1) 692 (1)
BITL	00D3	515 (1) 521 (1) 715 (1) 721 (1)
BLBC	00E9	269 (1) 296 (1)
BLBS	00E8	275 (1) 276 (1) 482 (1) 560 (1)
BLEQ	0015	298 (1) 729 (1)
BLEQU	001B	474 (1) 476 (1) 478 (1) 480 (1)
BLSS	0019	345 (1) 394 (1) 413 (1) 442 (1) 447 (1) 449 (1) 492 (1)
		529 (1)
BLSSU	001F	472 (1)
BNEQ	0012	397 (1) 513 (1) 516 (1) 520 (1) 716 (1) 720 (1)
BRB	0011	286 (1) 359 (1) 389 (1) 400 (1) 432 (1) 455 (1) 733 (1)
BRW	0031	483 (1) 618 (1) 745 (1)
BSBB	0010	372 (1) 410 (1) 424 (1) 426 (1) 428 (1) 430 (1)
BVS	001D	451 (1) 453 (1)
CALLS	00FB	268 (1) 295 (1) 308 (1) 365 (1) 601 (1) 763 (1)
CLRL	00D4	258 (1) 260 (1) 343 (1) 384 (1) 407 (1) 439 (1) 493 (1)
		495 (1) 497 (1) 508 (1) 517 (1) 646 (1) 668 (1) 679 (1)
		691 (1) 717 (1) 739 (1) 754 (1) 756 (1)
CLRQ	007C	408 (1)
CLRW	00B4	409 (1)
CMPB	0091	346 (1) 356 (1) 376 (1) 396 (1) 414 (1) 444 (1)
CMPL	00D1	386 (1) 448 (1) 512 (1) 530 (1) 723 (1) 741 (1)
CMPW	00B1	471 (1) 473 (1) 475 (1) 477 (1) 479 (1)
DECL	00D7	361 (1) 416 (1) 523 (1)
DECW	00B7	344 (1) 398 (1) 412 (1) 441 (1)
DIVL	00C6	694 (1)
EDIV	007B	494 (1) 496 (1) 498 (1) 518 (1) 634 (1) 649 (1) 669 (1)
		680 (1) 693 (1) 718 (1) 757 (1)
EMUL	007A	505 (1) 507 (1) 539 (1) 541 (1) 543 (1) 544 (1) 550 (1)
EXTZV	00EF	632 (1)
INCL	00D6	265 (1) 364 (1) 417 (1) 456 (1) 695 (1) 725 (1)
INCW	00B6	348 (1)
MNEGL	00CE	561 (1) 562 (1) 612 (1) 613 (1)
MOVAB	009E	352 (1) 381 (1)
MOVAL	00DE	283 (1) 371 (1) 399 (1) 406 (1) 423 (1) 706 (1)

MOVAW	003E	491	(1)	708	(1)										
MOVB	0090	383	(1)												
MOVL	00D0	257	(1)	259	(1)	261	(1)	267	(1)	294	(1)	307	(1)	309	(1)
		341	(1)	603	(1)	607	(1)	726	(1)						
MOVQ	007D	256	(1)	263	(1)	342	(1)	564	(1)	606	(1)	610	(1)		
MOVW	00B0	454	(1)	709	(1)	731	(1)	740	(1)	758	(1)	760	(1)		
MOVZBL	009A	443	(1)	511	(1)	727	(1)	755	(1)						
MOVZWL	003C	282	(1)	353	(1)	463	(1)	481	(1)	490	(1)	509	(1)	514	(1)
		525	(1)	526	(1)	537	(1)	538	(1)	540	(1)	542	(1)	559	(1)
		605	(1)	616	(1)	743	(1)								
MULL	00C4	506	(1)	678	(1)	707	(1)								
MULW	00A4	450	(1)	504	(1)										
PROBER	000C	609	(1)												
PROBEW	000D	604	(1)												
PUSHAL	00DF	268	(1)	284	(1)	292	(1)	305	(1)	365	(1)				
PUSHAQ	007F	268	(1)	295	(1)	308	(1)								
PUSHAW	003F	295	(1)	308	(1)										
PUSHL	00DD	285	(1)	293	(1)	295	(1)	306	(1)	308	(1)	365	(1)	601	(1)
		763	(1)												
RET	0004	312	(1)	464	(1)	565	(1)	765	(1)						
RSB	0005	457	(1)												
SBWC	00D9	563	(1)	614	(1)										
SOBGTR	00F5	358	(1)												
SUBL	00C2	266	(1)	340	(1)	446	(1)	527	(1)	728	(1)				
SUBW	00A2	297	(1)	393	(1)										
TSTL	00D5	519	(1)	719	(1)										
TSTW	00B5	374	(1)	440	(1)										

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCII	189	(1)	190	(1)	191	(1)	192	(1)	193	(1)	194	(1)	195	(1)	196	(1)
	197	(1)	198	(1)	199	(1)	200	(1)	215	(1)	216	(1)	217	(1)		
.BYTE	171	(1)	172	(1)	173	(1)	174	(1)	175	(1)	176	(1)	177	(1)	178	(1)
	179	(1)	180	(1)	181	(1)	182	(1)	207	(1)	208	(1)	209	(1)	373	(1)
	411	(1)	425	(1)	427	(1)	429	(1)	431	(1)						
.END	767	(1)														
.ENDC	268	(1)	295	(1)	308	(1)	365	(1)								
.ENDM	268	(1)	295	(1)	308	(1)	59	(1)	60	(1)	601	(1)	604	(1)	609	(1)
	763	(1)														
.GLOBL	268	(1)	365	(1)												
.IDENT	5	(1)														
.IF	268	(1)	295	(1)	308	(1)	365	(1)								
.IFF	268	(1)	295	(1)	308	(1)	365	(1)								
.IRP	295	(1)	308	(1)												
.LIBRARY	56	(1)	58	(1)												
.MDELETE	601	(1)	763	(1)												
.NOCROSS	59	(1)														
.PAGE	218	(1)	313	(1)	566	(1)										
.PSECT	169	(1)														
.RESTORE	59	(1)														
.SAVE	59	(1)														
.SBTTL	219	(1)	314	(1)	567	(1)										
.TITLE	4	(1)														
.WORD	255	(1)	339	(1)	599	(1)										

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	8	#-256	(1)	#-261	(1)	#-275	(1)	#-309	(1)	#-342	(1)	#-564	(1)
		#-603	(1)	#-607	(1)								
R0	40	#-263	(1)	#-269	(1)	#-296	(1)	#-353	(1)	#-358	(1)	#-382	(1)
		#-383	(1)	#-386	(1)	#-439	(1)	#-450	(1)	#-452	(1)	#-454	(1)
		#-463	(1)	#-490	(1)	491	(1)	#-494	(1)	#-507	(1)	#-508	(1)
		#-511	(1)	#-512	(1)	#-524	(1)	#-525	(1)	#-537	(1)	#-539	(1)
		#-541	(1)	#-543	(1)	#-544	(1)	#-557	(1)	#-559	(1)	#-605	(1)
		615	(1)	#-616	(1)	656	(1)	#-743	(1)				
R1	63	#-294	(1)	295	(1)	#-307	(1)	308	(1)	#-309	(1)	#-311	(1)
		#-352	(1)	#-356	(1)	#-361	(1)	#-381	(1)	#-382	(1)	#-443	(1)
		#-444	(1)	#-446	(1)	#-448	(1)	#-452	(1)	#-493	(1)	#-494	(1)
		#-496	(1)	#-506	(1)	#-507	(1)	#-526	(1)	#-528	(1)	#-530	(1)
		#-538	(1)	#-539	(1)	#-540	(1)	#-541	(1)	#-542	(1)	#-543	(1)
		#-558	(1)	#-606	(1)	#-610	(1)	#-613	(1)	632	(1)	#-633	(1)
		#-634	(1)	#-662	(1)	#-669	(1)	706	(1)	#-707	(1)	708	(1)
		#-709	(1)	#-715	(1)	#-718	(1)	#-721	(1)	#-726	(1)	#-727	(1)
		#-730	(1)	#-731	(1)	#-740	(1)	#-741	(1)	#-754	(1)	#-755	(1)
		#-759	(1)										
R2	54	255	(1)	#-260	(1)	#-265	(1)	#-276	(1)	#-282	(1)	283	(1)
		#-295	(1)	339	(1)	#-384	(1)	#-386	(1)	#-388	(1)	#-391	(1)
		#-495	(1)	#-496	(1)	#-498	(1)	#-505	(1)	#-510	(1)	#-511	(1)
		#-523	(1)	#-530	(1)	#-550	(1)	#-551	(1)	#-557	(1)	#-562	(1)
		#-564	(1)	599	(1)	#-612	(1)	#-614	(1)	#-634	(1)	#-647	(1)
		#-648	(1)	#-649	(1)	#-668	(1)	#-669	(1)	#-678	(1)	#-680	(1)
		706	(1)	#-717	(1)	#-718	(1)	#-719	(1)	#-727	(1)	#-728	(1)
		#-732	(1)	#-755	(1)	#-757	(1)						
R3	29	255	(1)	#-261	(1)	#-263	(1)	268	(1)	339	(1)	#-497	(1)
		#-498	(1)	#-504	(1)	#-505	(1)	#-514	(1)	#-515	(1)	#-518	(1)
		#-521	(1)	#-558	(1)	#-561	(1)	#-563	(1)	599	(1)	#-607	(1)
		609	(1)	#-610	(1)	#-646	(1)	#-679	(1)	#-680	(1)	#-692	(1)
		#-693	(1)	708	(1)								
R4	33	255	(1)	#-267	(1)	268	(1)	#-282	(1)	#-295	(1)	#-308	(1)
		339	(1)	#-371	(1)	#-399	(1)	#-406	(1)	#-407	(1)	#-408	(1)
		#-423	(1)	#-440	(1)	#-454	(1)	#-517	(1)	#-518	(1)	#-519	(1)
		599	(1)	#-632	(1)	#-648	(1)	#-691	(1)	#-693	(1)	#-694	(1)
		#-695	(1)	#-723	(1)	#-725	(1)	#-728	(1)	#-732	(1)		
R5	30	255	(1)	#-259	(1)	295	(1)	#-297	(1)	#-299	(1)	308	(1)
		#-311	(1)	339	(1)	#-342	(1)	#-344	(1)	#-348	(1)	#-353	(1)
		#-374	(1)	#-393	(1)	#-398	(1)	#-412	(1)	#-417	(1)	#-441	(1)
		#-481	(1)	#-507	(1)	#-510	(1)	#-527	(1)	#-528	(1)	#-550	(1)
		599	(1)	#-649	(1)	#-757	(1)	#-760	(1)				
R6	22	255	(1)	#-257	(1)	295	(1)	#-297	(1)	#-299	(1)	308	(1)
		339	(1)	#-346	(1)	352	(1)	#-376	(1)	381	(1)	#-392	(1)
		#-396	(1)	#-414	(1)	#-416	(1)	#-443	(1)	#-509	(1)	#-524	(1)
		599	(1)	#-756	(1)	#-757	(1)	#-758	(1)				
R7	34	339	(1)	#-341	(1)	365	(1)	371	(1)	#-391	(1)	399	(1)
		406	(1)	#-409	(1)	423	(1)	#-471	(1)	#-473	(1)	#-475	(1)
		#-477	(1)	#-479	(1)	#-481	(1)	#-490	(1)	#-509	(1)	#-514	(1)
		#-526	(1)	#-537	(1)	#-538	(1)	#-540	(1)	#-542	(1)	599	(1)
		#-603	(1)	604	(1)	#-709	(1)	#-731	(1)	#-732	(1)	#-739	(1)

```
                          #-740    (1)    #-753    (1)    #-758    (1)    #-760    (1)
R8                 5        339    (1)    #-343    (1)    #-364    (1)    #-482    (1)    #-560    (1)
SP                12       #-256    (1)    #-257    (1)    #-258    (1)    #-259    (1)    #-266    (1)    #-267    (1)
                          #-294    (1)    #-307    (1)    #-340    (1)    #-341    (1)    #-444    (1)    #-456    (1)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    22    00:00:00.01    00:00:00.24
Command processing               880    00:00:00.21    00:00:00.79
Pass 1                           429    00:00:01.42    00:00:03.01
Symbol table sort                  0    00:00:00.13    00:00:00.13
Pass 2                             9    00:00:00.47    00:00:00.99
Symbol table output                0    00:00:00.02    00:00:00.02
Psect synopsis output              0    00:00:00.00    00:00:00.00
Cross-reference output             3    00:00:00.20    00:00:00.37
Assembler run totals            1343    00:00:02.46    00:00:05.55

The working set limit was 2900 pages.
61946 bytes (121 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 476 non-local and 36 local symbols.
767 source lines were read in Pass 1, producing 40 object records in Pass 2.
20 pages of virtual memory were used to define 15 macros.

                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              2
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           2
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       7
TOTALS (all libraries)                                14

584 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CVRTIM.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(2) 62 DECLARATIONS
(3) 89 CVTFILNAM - CONVERT FILE NAME FROM ASCII TO RAD50
(4) 215 STORER50BYTE - CONVERT AND STORE ASCII BYTE TO RAD50
(5) 252 PACKRAD50 - PACK 3 BYTES OF RAD50 CHARACTERS INTO A WORD
(6) 284 ASCIITORAD50 - CONVERT ASCII CHARACTER TO RAD50

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element CVTFILNAM.MAR
0000     2 ; *1     6-FEB-1986 15:16:46 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element CVTFILNAM.MAR
0000     4         .TITLE  CVTFILNAM - Converts ASCII to RAD50
0000     5         .IDENT  '06-02'
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983 BY                                 *
0000    11 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    12 ;*  ALL RIGHTS RESERVED.                                                     *
0000    13 ;*                                                                           *
0000    14 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    15 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    16 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    17 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    18 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    19 ;*  TRANSFERRED.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    22 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    23 ;*  CORPORATION.                                                             *
0000    24 ;*                                                                           *
0000    25 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    26 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    27 ;*                                                                           *
0000    28 ;*                                                                           *
0000    29 ;*****************************************************************************
0000    30 ;
0000    31 ;++
0000    32 ; FACILITY:
0000    33 ;
0000    34 ;       Used by BOOT58 and FILEREAD
0000    35 ;
0000    36 ; ABSTRACT:
0000    37 ;
0000    38 ;       Converts an ASCII string file name, type, and version number
0000    39 ;       to a 6-word RAD50 file name and type, and a 16-bit binary
0000    40 ;       version number.
0000    41 ;
0000    42 ; ENVIRONMENT:
0000    43 ;
0000    44 ;       any
0000    45 ;
0000    46 ; AUTHOR:
0000    47 ;
0000    48 ;       Carol Peters    22 March 1979
0000    49 ;               Extracted from the Release 1 module FILEREAD.MAR; all
0000    50 ;               code and definitions are unchanged from Release 1 form.
0000    51 ;
0000    52 ; MODIFIED BY:
0000    53 ;
0000    54 ;       02      Bob Bergazzi    30-Jun-1983     Version 6.12
0000    55 ;               Adapted for VDS
0000    56 ;
0000    57 ;       V03-001 TCM0001         Trudy C. Matthews       15-Nov-1982
```

```
                                0000     58 ;
                                0000     59 ;
                                0000     60 ;--
```

Added G^ to call to LIB$CVT_DTB.

```
                        0000     62          .SBTTL  DECLARATIONS
                        0000     63
                        0000     64 ;
                        0000     65 ; Include files
                        0000     66 ;
                        0000     67
                        0000     68          $SSDEF                          ; Status code definitions
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     69
                        0000     70 ;
                        0000     71 ; Equated symbols
                        0000     72 ;
                        0000     73
00000061                0000     74          LCA     =       ^O141           ;LOWER CASE A
0000007A                0000     75          LCZ     =       ^O172           ;LOWER CASE Z
00000041                0000     76          UCA     =       ^O101           ;UPPER CASE A
0000005A                0000     77          UCZ     =       ^O132           ;UPPER CASE Z
00000030                0000     78          ZERO    =       ^O60            ;ASCII ZERO
00000039                0000     79          NINE    =       ^O71            ;ASCII NINE
0000002E                0000     80          DOT     =       ^O56            ;ASCII PERIOD
0000003B                0000     81          SEMI    =       ^O73            ;ASCII SEMICOLON
                        0000     82
                        0000     83 ;
                        0000     84 ; PSECT declaration
                        0000     85 ;
                        0000     86
                    00000000     87          .PSECT  code, exe, nowrt, shr, long     ;                                       [02]
```

```
                                        .SBTTL  CVTFILNAM - CONVERT FILE NAME FROM ASCII TO RAD50

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       THIS ROUTINE CONVERTS THE ASCII FILE NAME (NAME, TYPE, VERSION)
                                ; TO THE FILE NAME BLOCK FORM OF 3 WORDS RAD50 NAME, 1 WORD OF TYPE
                                ; AND 1 WORD OF VERSION.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       CALLG   ARGLIST,FIL$CVTFILNAM
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       FILNAM(AP)      =               ;STRING DESCRIPTOR OF FILE NAME STRING
                                ;       FILNAMBLK(AP)   =               ;ADDRESS OF 5 WORD BLOCK
                                ;                                               3 WORD RAD50 FILE NAME
                                ;                                               1 WORD RAD50 FILE TYPE
                                ;                                               1 WORD BINARY VERSION
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       NONE
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       R0 = SYSTEM STATUS CODE
                                ;       FILNAME BLOCK FILLED IN
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       NONE
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       SS$_NORMAL                      SUCCESSFUL  JMPLETION
                                ;       SS$_BADFILENAME                 SYNTAX ERROR IN FILE NAME
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       NONE
                                ;
                                ; EQUATED SYMBOLS:
                                ;
                                ;   OFFSETS FROM AP
                                ;
00000004  0000  136                     FILNAM          =       4       ;DESCRIPTOR OF ASCII FILE NAME STRING
00000008  0000  137                     FILNAMBLK       =       8       ;ADDRESS OF 5 WORD FILE NAME BLOCK
                0000  138 ;
                0000  139 ;   OFFSETS FROM FP
                0000  140 ;
FFFFFFFD  0000  141                     TYPE            =       -3      ;BEGINNING OF TYPE FIELD
FFFFFFF4  0000  142                     NAME            =       -12     ;BEGINNING OF NAME FIELD
                0000  143 ;
                0000  144 ;--
                0000  145
```

```
                          0000   146 FIL$CVTFILNAM::
                   007C   0000   147         .WORD   ^M<R2,R3,R4,R5,R6>
                7E   7C   0002   148         CLRQ    -(SP)                   ;RESERVE AND ZERO A
                7E   D4   0004   149         CLRL    -(SP)                   ;12 BYTE NAME STRING
           55   04 BC   7D   0006   150         MOVQ    @FILNAM(AP),R5          ;R5 = SIZE, R6 = ADR OF STRING
              55   55   3C   000A   151         MOVZWL  R5,R5                   ; only low word used                    [02]
           54   F4 AD   9E   000D   152         MOVAB   NAME(FP),R4             ;ADDRESS TO STORE NAME
              53   09   D0   0011   153         MOVL    #9,R3                   ;UP TO 9 CHARACTERS
                0F   11   0014   154         BRB     40$
                          0016   155 ;
                          0016   156 ; STORE UP TO 9 CHARACTERS OF RAD50 IN THE BYTE ARRAY ADDRESSED BY R4
                          0016   157 ; IN PREPARATION FOR CONVERTING INTO THE PACKED RAD50 FORMAT
                          0016   158 ;
                          0016   159 20$:
              50   86   90   0016   160         MOVB    (R6)+,R0                ;GET THE NEXT CHARACTER
              50   2E   91   0019   161         CMPB    #DOT,R0                 ;IS IT A DOT?
                0C   13   001C   162         BEQL    FILETYPE                ;BRANCH IF YES
              50   3B   91   001E   163         CMPB    #SEMI,R0                ;IS IT A SEMICOLON
                24   13   0021   164         BEQL    VERSION                 ;BRANCH IF YES
                59   10   0023   165         BSBB    STORER50BYTE            ;CONVERT AND STORE THE CHARACTER
                          0025   166 40$:
                 EE 55   F4   0025   167         SOBGEQ  R5,20$                  ;COUNT THE CHARACTERS IN THE STRING
                2F   11   0028   168         BRB     BUILDFNB                ;END OF STRING, NO TYPE, NO VERSION
                          002A   169 ;
                          002A   170 ; NOW SET UP THE 3 BYTE ARRAY OF FILE TYPE
                          002A   171 ;
                          002A   172 FILETYPE:
           54   FD AD   9E   002A   173         MOVAB   TYPE(FP),R4             ;ADDRESS OF BYTE ARRAY
              53   03   D0   002E   174         MOVL    #3,R3                   ;MAX SIZE OF FILE TYPE
                0F   11   0031   175         BRB     40$
                          0033   176 20$:
              50   86   90   0033   177         MOVB    (R6)+,R0                ;GET NEXT CHARACTER
              50   3B   91   0036   178         CMPB    #SEMI,R0                ;IS IT A SEMICOLON?
                0C   13   0039   179         BEQL    VERSION                 ;BRANCH IF YES
              50   2E   91   003B   180         CMPB    #DOT,R0                 ;DOT FOR VERSION TOO
                07   13   003E   181         BEQL    VERSION                 ;BRANCH IF VERSION DELIMITER
                3C   10   0040   182         BSBB    STORER50BYTE            ;CONVERT AND STORE THE BYTE
                          0042   183 40$:
                 EE 55   F4   0042   184         SOBGEQ  R5,20$                  ;LOOP THROUGH THE CHARACTERS
                12   11   0045   185         BRB     BUILDFNB                ;END OF STRING, NO VERSION
                          0047   186 ;
                          0047   187 ; VERSION DELIMITER FOUND, CONVERT THE VERSION
                          0047   188 ;
                          0047   189 VERSION:
 7E   08 AC   08   C1   0047   190         ADDL3   #4*2,FILNAMBLK(AP),-(SP) ;ADDRESS TO STORE VERSION
              7E   55   7D   004C   191         MOVQ    R5,-(SP)                ;PUSH ADDRESS, PUSH SIZE OF VERSION STRING
00000000'GF   03   FB   004F   192         CALLS   #3,G^LIB$CVT_DTB        ;CONVERT DECIMAL VERSION STRING TO BINARY
                 2A 50   E9   0056   193         BLBC    R0,BADFILNAM            ;BRANCH IF BAD FILE NAME
                          0059   194 ;
                          0059   195 ; NOW PACK THE TEMPORARY BYTE STRING INTO THE RAD50 WORDS
                          0059   196 ;
                          0059   197 BUILDFNB:
           54   08 AC   D0   0059   198         MOVL    FILNAMBLK(AP),R4        ;ADDRESS TO STORE PACKED RAD50
           53   F4 AD   9E   005D   199         MOVAB   NAME(FP),R3             ;ADDRESS OF NAME STRING
                63   95   0061   200         TSTB    (R3)                    ;ANY NAME GIVEN?
                06   13   0063   201         BEQL    20$                     ;BRANCH IF NO
                26   10   0065   202         BSBB    PACKRAD50               ;1ST 3 CHARACTERS
```

```
                24   10  0067   203          BSBB    PACKRAD50                ;2ND 3 CHARACTERS
                22   10  0069   204          BSBB    PACKRAD50                ;3RD 3 CHARACTERS
                         006B   205 20$:
     54   08 AC  06   C1  006B   206          ADDL3   #3*2,FILNAMBLK(AP),R4    ;ADDRESS TO STORE FILE TYPE
          53   FD AD   9E  0070   207          MOVAB   TYPE(FP),R3              ;ADDRESS OF TYPE STRING
                63   95  0074   208          TSTB    (R3)                     ;ANY FILE TYPE PRESENT?
                02   13  0076   209          BEQL    40$                      ;BRANCH IF NOT
                13   10  0078   210          BSBB    PACKRAD50                ;CONVERT 3 CHARACTERS
                         007A   211 40$:
          50    01   3C  007A   212          MOVZWL  #SS$_NORMAL,R0           ;INDICATE SUCCESS
                     04  007D   213          RET
```

```
                        007E   215          .SBTTL  STORER50BYTE - CONVERT AND STORE ASCII BYTE TO RAD50
                        007E   216 ;**
                        007E   217 ; FUNCTIONAL DESCRIPTION:
                        007E   218 ;
                        007E   219 ;       CONVERT ASCII BYTE TO RAD50 AND STORE IN THE SPECIFIED BYTE ARRAY
                        007E   220 ;
                        007E   221 ; CALLING SEQUENCE:
                        007E   222 ;
                        007E   223 ;       BSBB    STORER50BYTE
                        007E   224 ;
                        007E   225 ; INPUT:
                        007E   226 ;
                        007E   227 ;       R0 = ASCII BYTE
                        007E   228 ;       R3 = SIZE OF BYTE ARRAY TO STORE IN
                        007E   229 ;       R4 = ADDRESS OF BYTE ARRAY TO STORE IN
                        007E   230 ;
                        007E   231 ; OUTPUT:
                        007E   232 ;
                        007E   233 ;       RSB IF SUCCESSFUL, RET WITH ERROR CODE IN R0 IF ERROR
                        007E   234 ;       R3, R4 UPDATED TO REFLECT THE ADDITIONAL BYTE STORED
                        007E   235 ;
                        007E   236 ;--
                        007E   237
                        007E   238          .ENABL  LSB
                        007E   239
                        007E   240 STORER50BYTE:
              26   10   007E   241          BSBB    ASCIITORAD50            ;CONVERT TO RAD50
            06 53  F4   0080   242          SOBGEQ  R3,20$                  ;BRANCH IF ROOM TO STORE THIS CHARACTER
                        0083   243 BADFILNAM:
     50  0818 8F  3C   0083   244          MOVZWL  #SS$_BADFILENAME,R0     ;RETURN ERROR CODE
                  04   0088   245          RET
                        0089   246 20$:
        84  50  90   0089   247          MOVB    R0,(R4)+                ;STORE IT
                  05   008C   248          RSB                             ;AND RSB SUCCESSFULLY
                        008D   249
                        008D   250          .DSABL  LSB
```

```
                                008D   252          .SBTTL  PACKRAD50 - PACK 3 BYTES OF RAD50 CHARACTERS INTO A WORD
                                008D   253 ;++
                                008D   254 ; FUNCTIONAL DESCRIPTION:
                                008D   255 ;
                                008D   256 ;       PACK 3 BYTES OF RAD50 CHARACTERS INTO A WORD
                                008D   257 ;
                                008D   258 ; CALLING SEQUENCE:
                                008D   259 ;
                                008D   260 ;       BSBB    PACKRAD50
                                008D   261 ;
                                008D   262 ; INPUTS:
                                008D   263 ;
                                008D   264 ;       R3 = RAD50 BYTE STRING ADDRESS
                                008D   265 ;       R4 = ADDRESS OF WORD ARRAY TO STORE IN
                                008D   266 ;
                                008D   267 ; OUTPUTS:
                                008D   268 ;
                                008D   269 ;       R3 UPDATED TO POINT TO NEXT GROUP OF 3 CHARACTERS
                                008D   270 ;       R4 UPDATED TO POINT AT NEXT WORD TO STORE IN
                                008D   271 ;
                                008D   272 ;--
                                008D   273
                                008D   274 PACKRAD50:
          50    83    9A      008D   275          MOVZBL  (R3)+,R0                ;BUILD PACKED WORD IN R0
    50    0640  8F    A4      0090   276          MULW    #40*40,R0               ;START ACCUMULUATING IN R0
          51    83    9A      0095   277          MOVZBL  (R3)+,R1                ;SECOND CHARACTER TO TEMP
          51    28    A4      0098   278          MULW    #40,R1                  ;MULTIPLY BY THE RADIX
          50    51    A0      009B   279          ADDW    R1,R0                   ;AND ACCUMULATE
          51    83    9A      009E   280          MOVZBL  (R3)+,R1                ;LAST CHARACTER
    84    50    51    A1      00A1   281          ADDW3   R1,R0,(R4)+             ;ACCUMULATE AND STORE RESULT
                      05      00A5   282          RSB
```

```
                     00A6    284          .SBTTL  ASCIITORAD50 - CONVERT ASCII CHARACTER TO RAD50
                     00A6    285 ;**
                     00A6    286 ; FUNCTIONAL DESCRIPTION:
                     00A6    287 ;
                     00A6    288 ;       CONVERT ASCII CHARACTER TO ITS RAD50 EQUIVALENT
                     00A6    289 ;
                     00A6    290 ; CALLING SEQUENCE:
                     00A6    291 ;
                     00A6    292 ;       BSBB    ASCIITORAD50
                     00A6    293 ;
                     00A6    294 ; INPUTS:
                     00A6    295 ;
                     00A6    296 ;       R0 = CHARACTER TO CONVERT
                     00A6    297 ;
                     00A6    298 ; OUTPUTS:
                     00A6    299 ;
                     00A6    300 ;       RSB IF SUCCESSFUL, RET WITH ERROR CODE IN R0 IF ERROR
                     00A6    301 ;       R0 = RAD50 EQUIVALENT
                     00A6    302 ;       R1,R2,R3 PRESERVED
                     00A6    303 ;
                     00A6    304 ;--
                     00A6    305
                     00A6    306 ASCIITORAD50:
7A 8F    50    91    00A6    307          CMPB    R0,#LCZ                 ;LOWER CASE ALPHA?
         D7    1A    00AA    308          BGTRU   BADFILNAM               ;BRANCH IF BAD RAD50
61 8F    50    91    00AC    309          CMPB    R0,#LCA
         05    19    00B0    310          BLSS    10$                     ;BRANCH IF NOT
50   60 8F    82    00B2    311          SUBB    #LCA-1,R0               ;CONVERT TO ALPHA
               05    00B6    312          RSB                             ;AND RETURN SUCCESSFULLY
                     00B7    313 10$:
5A 8F    50    91    00B7    314          CMPB    R0,#UCZ                 ;UPPER CASE ALPHA?
         C6    14    00BB    315          BGTR    BADFILNAM               ;BRANCH IF BAD RAD50
41 8F    50    91    00BD    316          CMPB    R0,#UCA
         05    19    00C1    317          BLSS    20$                     ;BRANCH IF NOT ALPHA
50   40 8F    82    00C3    318          SUBB    #UCA-1,R0               ;CONVERT TO RAD50 ALPHA
               05    00C7    319          RSB                             ;AND RETURN SUCCESSFULLY
                     00C8    320 20$:
   39    50    91    00C8    321          CMPB    R0,#NINE                ;NUMERIC?
         B6    14    00CB    322          BGTR    BADFILNAM               ;BRANCH IF BAD RAD50
   30    50    91    00CD    323          CMPB    R0,#ZERO
         B1    19    00D0    324          BLSS    BADFILNAM               ;BRANCH IF BAD RAD50
   50    12    82    00D2    325          SUBB    #ZERO-30,R0             ;CONVERT TO RAD50 NUMERIC
               05    00D5    326          RSB                             ;AND RETURN SUCCESSFULLY
                     00D6    327
                     00D6    328          .END
```

```
ASCIITORAD50                    000000A6 R   D  02
BADFILNAM                       00000083 R   D  02
BUILDFNB                        00000059 R   D  02
DOT                           = 0000002E     D
FIL$CVTFILNAM                   00000000 RG  D  02
FILETYPE                        0000002A R   D  02
FILNAM                        = 00000004     D
FILNAMBLK                     = 00000008     D
LCA                           = 00000061     D
LCZ                           = 0000007A     D
LIB$CVT_DTB                     ********   X   02
NAME                          = FFFFFFF4     D
NINE                          = 00000039     D
PACKRAD50                       0000008D R   D  02
SEMI                          = 0000003B     D
SS$_BADFILENAME               = 00000818     D
SS$_NORMAL                    = 00000001     D
STORERSOBYTE                    0000007E R   D  02
TYPE                          = FFFFFFFD     D
UCA                           = 00000041     D
UCZ                           = 0000005A     D
VERSION                         00000047 R   D  02
ZERO                          = 00000030     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	000000D6 (214.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                      VALUE        DEFINITION      REFERENCES...
------                      -----        ----------      -------------
ASCIITORAD50                000000A6-R   306    (6)    #-241    (4)
BADFILNAM                   00000083-R   243    (4)    #-193    (3)    #-308    (6)    #-315    (6)    #-322    (6)
                                                       #-324    (6)
BUILDFNB                    00000059-R   197    (3)    #-168    (3)    #-185    (3)
DOT                        =0000002E     80     (2)    #-161    (3)    #-180    (3)
FIL$CVTFILNAM               00000000-R   146    (3)
FILETYPE                    0000002A-R   172    (3)    #-162    (3)
FILNAM                     =00000004     136    (3)    #-150    (3)
FILNAMBLK                  =00000008     137    (3)    #-190    (3)    #-198    (3)    #-206    (3)
LCA                        =00000061     74     (2)    #-309    (6)    #-311    (6)
LCZ                        =0000007A     75     (2)    #-307    (6)
LIB$CVT_DTB                 00000000-XR                 192    (3)
NAME                       =FFFFFFF4     142    (3)      152    (3)      199    (3)
NINE                       =00000039     79     (2)    #-321    (6)
PACKRAD50                   0000008D-R   274    (5)    #-202    (3)    #-203    (3)    #-204    (3)    #-210    (3)
SEMI                       =0000003B     81     (2)    #-163    (3)    #-178    (3)
SS$_BADFILENAME            =00000818                   #-244    (4)
SS$_NORMAL                 =00000001                   #-212    (3)
STORER50BYTE                0000007E-R   240    (4)    #-165    (3)    #-182    (3)
TYPE                       =FFFFFFFD     141    (3)      173    (3)      207    (3)
UCA                        =00000041     76     (2)    #-316    (6)    #-318    (6)
UCZ                        =0000005A     77     (2)    #-314    (6)
VERSION                     00000047-R   189    (3)    #-164    (3)    #-179    (3)    #-181    (3)
ZERO                       =00000030     78     (2)    #-323    (6)    #-325    (6)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEFINI         1           68     (2)      68     (2)
$SSDEF          21          68     (2)      68     (2)
```

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDL3           00C1      190     (3)     206     (3)
ADDW            00A0      279     (5)
ADDW3           00A1      281     (5)
BEQL            0013      162     (3)     164     (3)     179     (3)     181     (3)     201     (3)     209     (3)
BGTR            0014      315     (6)     322     (6)
BGTRU           001A      308     (6)
BLBC            00E9      193     (3)
BLSS            0019      310     (6)     317     (6)     324     (6)
BRB             0011      154     (3)     168     (3)     175     (3)     185     (3)
BSBB            0010      165     (3)     182     (3)     202     (3)     203     (3)     204     (3)     210     (3)     241     (4)
CALLS           00FB      192     (3)
CLRL            00D4      149     (3)
CLRQ            007C      148     (3)
CMPB            0091      161     (3)     163     (3)     178     (3)     180     (3)     307     (6)     309     (6)     314     (6)
                          316     (6)     321     (6)     323     (6)
MOVAB           009E      152     (3)     173     (3)     199     (3)     207     (3)
MOVB            0090      160     (3)     177     (3)     247     (4)
MOVL            00D0      153     (3)     174     (3)     198     (3)
MOVQ            007D      150     (3)     191     (3)
MOVZBL          009A      275     (5)     277     (5)     280     (5)
MOVZWL          003C      151     (3)     212     (3)     244     (4)
MULW            00A4      276     (5)     278     (5)
RET             0004      213     (3)     245     (4)
RSB             0005      248     (4)     282     (5)     312     (6)     319     (6)     326     (6)
SOBGEQ          00F4      167     (3)     184     (3)     242     (4)
SUBB            0082      311     (6)     318     (6)     325     (6)
TSTB            0095      200     (3)     208     (3)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.DSABL          250     (4)
.ENABL          238     (4)
.END            328     (6)
.ENDM           68      (2)
.IDENT          5       (1)
.NOCROSS        68      (2)
.PSECT          87      (2)
.RESTORE        68      (2)
.SAVE           68      (2)
.SBTTL          215     (4)     252     (5)     284     (6)     62      (2)     89      (3)
.TITLE          4       (1)
.WORD           147     (3)
```

```
                                        +---------------------------+
                                        ! Register Cross Reference !
                                        +---------------------------+

REGISTER  NO. REFERENCES     REFERENCES...
--------  --------------     -------------
AP                4          #-150    (3)    #-190    (3)    #-198    (3)    #-206    (3)
FP                4            152    (3)      173    (3)      199    (3)      207    (3)
R0               23          #-160    (3)    #-161    (3)    #-163    (3)    #-177    (3)    #-178    (3)    #-180    (3)
                             #-193    (3)    #-212    (3)    #-244    (4)    #-247    (4)    #-275    (5)    #-276    (5)
                             #-279    (5)    #-281    (5)    #-307    (6)    #-309    (6)    #-311    (6)    #-314    (6)
                             #-316    (6)    #-318    (6)    #-321    (6)    #-323    (6)    #-325    (6)
R1                5          #-277    (5)    #-278    (5)    #-279    (5)    #-280    (5)    #-281    (5)
R2                1            147    (3)
R3               11            147    (3)    #-153    (3)    #-174    (3)    #-199    (3)    #-200    (3)    #-207    (3)
                             #-208    (3)    #-242    (4)    #-275    (5)    #-277    (5)    #-280    (5)
R4                7            147    (3)    #-152    (3)    #-173    (3)    #-198    (3)    #-206    (3)    #-247    (4)
                             #-281    (5)
R5                7            147    (3)    #-150    (3)    #-151    (3)    #-167    (3)    #-184    (3)    #-191    (3)
R6                3            147    (3)    #-160    (3)    #-177    (3)
SP                4          #-148    (3)    #-149    (3)    #-190    (3)    #-191    (3)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     22    00:00:00.04    00:00:00.25
Command processing                892    00:00:00.21    00:00:00.76
Pass 1                            452    00:00:00.92    00:00:01.57
Symbol table sort                   0    00:00:00.13    00:00:00.14
Pass 2                             49    00:00:00.29    00:00:00.52
Symbol table output                 2    00:00:00.01    00:00:00.01
Psect synopsis output               2    00:00:00.00    00:00:00.00
Cross-reference output              6    00:00:00.06    00:00:00.10
Assembler run totals             1427    00:00:01.67    00:00:03.37

The working set limit was 1900 pages.
43684 bytes (86 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 431 non-local and 9 local symbols.
328 source lines were read in Pass 1, producing 36 object records in Pass 2.
9 pages of virtual memory were used to define 7 macros.

                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                                Macros defined
------------------                                --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5                    0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                      0
SYS$COMMON:[SYSLIB]LIB.MLB;2                                 0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                             4
TOTALS (all libraries)                                       4
```

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 6 Frame M15          Sequence 1224
CVTFILNAM                      - Converts ASCII to RAD50        11-NOV-1987 17:31:46  VAX/VMS Macro V04-00        Page  16
VAX-11 Macro Run Statistics                                      3-JAN-1985 16:49:59  CVTFILNAM.MAR;1                  (6)

469 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:CVTFILNAM.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

Table of contents

(2) 77 DEASSIGN I/O CHANNEL

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DASSGN.MAR
0000     2 ; *1     6-FEB-1986 15:16:48 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element DASSGN.MAR
0000     4         .TITLE  DASSGN  *** DASSGN deassign QIO channel
0000     5         .IDENT  /05-02/
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                    *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    12 ;*                                                                           *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    18 ;*  TRANSFERRED.                                                             *
0000    19 ;*                                                                           *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    22 ;*  CORPORATION.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28 ;
0000    29 ; D. N. CUTLER 26-AUG-76
0000    30 ;
0000    31 ;
0000    32 ;               N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
0000    33 ;       01      DIAGNOSTIC SUPERVISOR INTEGRATION
0000    34 ;               Dave Butenhof   13-may-1980, Version 5.4
0000    35 ;       02      Create .UPD file to convert VMS V2.0 source to Supervisor
0000    36 ;               environment
0000    37 ;               Roger Riggs, 19-Jun-1980, Version 5.5
0000    38 ;       03      Put normal in R0 instead of R1
0000    39 ;       V03     LMK0002         LEN KAWELL      27-SEP-1979,03-OCT-1979
0000    40 ;       Disassociate mailbox if CCB$M_AMB is set in channel status.
0000    41 ;
0000    42 ;       V02     LMK0001         LEN KAWELL      27-DEC-1978
0000    43 ;       Clear DEV$M_RCK, DEV$M_WCK, and DEV$M_SWL when dismounting
0000    44 ;       a foreign mounted volume.
0000    45 ;
0000    46 ;
0000    47 ;       DS01    M. Baggett      MARCH-8-1983    VDS 6.11
0000    48 ;               Fix truncation error.
0000    49 ;
0000    50 ; SYSTEM SERVICE DEASSIGN I/O CHANNEL
0000    51 ;
0000    52 ; MACRO LIBRARY CALLS
0000    53 ;
0000    54
0000    55         $CCBDEF                         ;DEFINE CCB OFFSETS
0000               .SAVE   LOCAL_BLOCK
0000               .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
```

```
                    00000004  0000                        .BLKL   1
                              0004              .IIF      NB,CCB$L_WIND,  CCB$L_WIND:
                    00000008  0004                        .BLKL   1
                              0008              .IIF      NB,CCB$B_STS,   CCB$B_STS:
                    00000009  0008                        .BLKB   1
                              0009              .IIF      NB,CCB$B_AMOD,  CCB$B_AMOD:
                    0000000A  0009                        .BLKB   1
                              000A              .IIF      NB,CCB$W_IOC,   CCB$W_IOC:
                    0000000C  000A                        .BLKW   1
                              000C              .IIF      NB,CCB$L_DIRP,  CCB$L_DIRP:
                    00000010  000C                        .BLKL   1
                              0010              .IIF      NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                              0010              .IIF      NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                              0000              .RESTORE
                              0000      56      $DDBDEF                           ;DEFINE DDB OFFSETS
                              0000              .SAVE     LOCAL_BLOCK
                              0000              .PSECT    $ABS$,ABS
                    00000000  0010              .=0
                              0000              .IIF      NB,DDB$L_LINK,  DDB$L_LINK:
                    00000004  0000                        .BLKL   1
                              0004              .IIF      NB,DDB$L_UCB,   DDB$L_UCB:
                    00000008  0004                        .BLKL   1
                              0008              .IIF      NB,DDB$W_SIZE,  DDB$W_SIZE:
                    0000000A  0008                        .BLKW   1
                              000A              .IIF      NB,DDB$B_TYPE,  DDB$B_TYPE:
                    0000000B  000A                        .BLKB   1
                    0000000C  000B                        .BLKB   1
                              000C              .IIF      NB,DDB$L_DDT,   DDB$L_DDT:
                    00000010  000C                        .BLKL   1
                              0010              .IIF      NB,DDB$L_ACPD,  DDB$L_ACPD:
                    00000013  0010                        .BLKB   3
                              0013              .IIF      NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
                    00000014  0013                        .BLKB   1
                              0014              .IIF      NB,DDB$T_NAME,  DDB$T_NAME:
                    00000015  0014                        .BLKB   1
                    00000024  0015                        .BLKB   15
                              0024              .IIF      NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
                    00000025  0024                        .BLKB   1
                    00000034  0025                        .BLKB   15
                              0034              .IIF      NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                              0034              .IIF      NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                              0000              .RESTORE
                              0000      57      $DDTDEF                           ;DEFINE DDT OFFSETS
                              0000              .SAVE     LOCAL_BLOCK
                              0000              .PSECT    $ABS$,ABS
                    00000000  0034              .=0
                              0000              .IIF      NB,DDT$L_START, DDT$L_START:
                    00000004  0000                        .BLKL   1
                              0004              .IIF      NB,DDT$L_UNSOLINT,      DDT$L_UNSOLINT:
                    00000008  0004                        .BLKL   1
                              0008              .IIF      NB,DDT$L_FDT,   DDT$L_FDT:
                    0000000C  0008                        .BLKL   1
                              000C              .IIF      NB,DDT$L_CANCEL,        DDT$L_CANCEL:
                    00000010  000C                        .BLKL   1
                              0010              .IIF      NB,DDT$L_REGDUMP,       DDT$L_REGDUMP:
                    00000014  0010                        .BLKL   1
                              0014              .IIF      NB,DDT$W_DIAGBUF,       DDT$W_DIAGBUF:
```

```
00000016  0014                                  .BLKW   1
          0016                  .IIF    NB,DDT$W_ERRORBUF,      DDT$W_ERRORBUF:
00000018  0016                                  .BLKW   1
          0018                  .IIF    NB,DDT$L_UNITINIT,      DDT$L_UNITINIT:
0000001C  0018                                  .BLKL   1
          001C                  .IIF    NB,DDT$L_ALTSTART,      DDT$L_ALTSTART:
00000020  001C                                  .BLKL   1
          0000                  .RESTORE
          0000     58           $IODEF                          ;DEFINE I/O FUNCTION CODES
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  0034                  .=0
          0000                  .RESTORE
          0000     59           $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  0034                  .=0
          0000                  .RESTORE
          0000     60           $PCBDEF                         ;DEFINE PCB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  0034                  .=0
          0000                  .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D  000C                                  .BLKB   1
          000D                  .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E  000D                                  .BLKB   1
          000E                  .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010  000E                                  .BLKW   1
          0010                  .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018  0014                                  .BLKL   1
          0018                  .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C  0018                                  .BLKL   1
          001C                  .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020  001C                                  .BLKL   1
          0020                  .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024  0020                                  .BLKL   1
          0024                  .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028  0024                                  .BLKL   1
          0028                  .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C  0028                                  .BLKL   1
          002C                  .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E  002C                                  .BLKW   1
          002E                  .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                                  .BLKB   1
          002F                  .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
```

```
                    00000030  002F                        .BLKB   1
                              0030          .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
                    00000032  0030                        .BLKW   1
                              0032          .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
                    00000034  0032                        .BLKW   1
                              0034          .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
                    00000036  0034                        .BLKW   1
                              0036          .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
                    00000038  0036                        .BLKW   1
                              0038          .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
                    0000003A  0038                        .BLKW   1
                              003A          .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
                    0000003C  003A                        .BLKW   1
                              003C          .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
                    0000003E  003C                        .BLKW   1
                              003E          .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
                    00000040  003E                        .BLKW   1
                              0040          .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
                    00000042  0040                        .BLKW   1
                              0042          .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
                    00000044  0042                        .BLKW   1
                              0044          .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
                    0000004C  0044                        .BLKB   8
                              004C          .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                              004C          .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
                    00000050  004C                        .BLKL   1
                              0050          .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
                    00000054  0050                        .BLKL   1
                              0054          .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
                    00000058  0054                        .BLKL   1
                              0058          .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
                    0000005C  0058                        .BLKL   1
                              005C          .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
                    00000060  005C                        .BLKL   1
                              0060          .IIF    NB,PCB$L_PID,   PCB$L_PID:
                    00000064  0060                        .BLKL   1
                              0064          .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
                    00000068  0064                        .BLKL   1
                              0068          .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
                    00000078  0068                        .BLKB   16
                              0078          .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
                    0000007C  0078                        .BLKL   1
                              007C          .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
                    00000084  007C                        .BLKQ   1
                              0084          .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
                    00000088  0084                        .BLKL   1
                              0088          .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                              0088          .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
                    0000008A  0088                        .BLKW   1
                              008A          .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
                    0000008C  008A                        .BLKW   1
                              008C          .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                              008C          .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                              0000          .RESTORE
                              0000    61    $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                              0000          .SAVE   LOCAL_BLOCK
                              0000          .PSECT  $ABS$,ABS
```

```
                    00000000  008C                .=0
                              0000                .RESTORE
                              0000      62        $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                              0000                .SAVE   LOCAL_BLOCK
                              0000                .PSECT  $ABS$,ABS
                    00000000  008C                .=0
                              0000                .RESTORE
                              0000      63        $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                              0000                .SAVE   LOCAL_BLOCK
                              0000                .PSECT  $ABS$,ABS
                    00000000  008C                .=0
                              0000                .RESTORE
                              0000      64        $UCBDEF                         ;DEFINE UCB OFFSETS
                              0000                .SAVE   LOCAL_BLOCK
                              0000                .PSECT  $ABS$,ABS
                    00000000  008C                .=0
                              0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                              0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                    00000004  0000                        .BLKL   1
                              0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                              0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                    00000008  0004                        .BLKL   1
                              0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                    0000000A  0008                        .BLKW   1
                              000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                    0000000B  000A                        .BLKB   1
                              000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                    0000000C  000B                        .BLKB   1
                              000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                              000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                              000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                    0000000D  000C                        .BLKB   1
                    00000010  000D                        .BLKB   3
                              0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                              0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
                    00000014  0010                        .BLKL   1
                              0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                              0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                              0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
                    00000016  0014                        .BLKW   1
                              0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
                    00000018  0016                        .BLKW   1
                              0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                              0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
                    0000001A  0018                        .BLKW   1
                              001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                              001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                    0000001C  001A                        .BLKW   1
                              001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                    00000020  001C                        .BLKL   1
                              0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                    00000024  0020                        .BLKL   1
                              0024                .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                    00000028  0024                        .BLKL   1
                              0028                .IIF    NB,UCB$L_PID,   UCB$L_PID:
                    0000002C  0028                        .BLKL   1
                              002C                .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
```

```
                    00000030  002C                        .BLKL   1
                              0030          .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
                    00000034  0030                        .BLKL   1
                              0034          .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
                    00000038  0034                        .BLKL   1
                              0038          .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
                    00000039  0038                        .BLKB   1
                              0039          .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
                    0000003A  0039                        .BLKB   1
                              003A          .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
                    0000003C  003A                        .BLKW   1
                              003C          .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                              003C          .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                              003C          .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
                    0000003D  003C                        .BLKB   1
                              003D          .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                              003D          .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
                    0000003E  003D                        .BLKB   1
                              003E          .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                              003E          .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
                    0000003F  003E                        .BLKB   1
                              003F          .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
                    00000040  003F                        .BLKB   1
                              0040          .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
                    00000044  0040                        .BLKL   1
                              0044          .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
                    00000048  0044                        .BLKL   1
                              0048          .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
                    0000004A  0048                        .BLKW   1
                              004A          .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                              004A          .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
                    0000004B  004A                        .BLKB   1
                              004B          .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
                    0000004C  004B                        .BLKB   1
                              004C          .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
                    00000050  004C                        .BLKL   1
                              0050          .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
                    00000052  0050                        .BLKW   1
                              0052          .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                              0052          .IIF    NB,UCB$B_STATE, UCB$B_STATE:
                    00000053  0052                        .BLKB   1
                              0053          .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
                    00000054  0053                        .BLKB   1
                              0054          .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
                    00000058  0054                        .BLKL   1
                              0058          .IIF    NB,UCB$W_STS,   UCB$W_STS:
                    0000005A  0058                        .BLKW   1
                              005A          .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
                    0000005C  005A                        .BLKW   1
                              005C          .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                              005C          .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
                    00000060  005C                        .BLKL   1
                              0060          .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
                    00000064  0060                        .BLKL   1
                              0064          .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                              0064          .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
                    00000068  0064                        .BLKL   1
```

```
                     0068          .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C             0068                  .BLKL   1
                     006C          .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E             006C                  .BLKW   1
                     006E          .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070             006E                  .BLKW   1
                     0070          .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071             0070                  .BLKB   1
                     0071          .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072             0071                  .BLKB   1
                     0072          .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074             0072                  .BLKW   1
                     0074          .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                     0074          .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000             0074          . = 0
                     0000          .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002             0000                  .BLKW   1
00000074             0002          . = 116
                     0074          .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078             0074                  .BLKL   1
                     0078          .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C             0078                  .BLKL   1
                     007C          .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080             007C                  .BLKL   1
                     0080          .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084             0080                  .BLKL   1
                     0084          .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088             0084                  .BLKL   1
                     0088          .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C             0088                  .BLKL   1
                     008C          .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090             008C                  .BLKL   1
                     0090          .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                     0090          .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074             0090          . = 116
                     0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075             0074                  .BLKB   1
                     0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076             0075                  .BLKB   1
                     0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077             0076                  .BLKB   1
                     0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078             0077                  .BLKB   1
                     0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C             0078                  .BLKL   1
0000007E             007C                  .BLKW   1
                     007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080             007E                  .BLKW   1
                     0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084             0080                  .BLKL   1
00000080             0084          . = 128
00000084             0080                  .BLKL   1
                     0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088             0084                  .BLKL   1
                     0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A             0088                  .BLKW   1
                     008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
```

```
                    0000008C  008A                        .BLKW   1
                              008C              .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                              008C              .IIF    NB,UCB$W_DA,    UCB$W_DA:
                    0000008E  008C                        .BLKW   1
                              008E              .IIF    NB,UCB$W_DC,    UCB$W_DC:
                    00000090  008E                        .BLKW   1
                              0090              .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
                    00000092  0090                        .BLKW   1
                              0092              .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
                    00000094  0092                        .BLKW   1
                              0094              .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
                    00000095  0094                        .BLKB   1
                              0095              .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
                    00000096  0095                        .BLKB   1
                              0096              .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
                    00000098  0096                        .BLKW   1
                    00000096  0098              . = 150
                    00000098  0096                        .BLKW   1
                              0098              .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
                    0000009C  0098                        .BLKL   1
                              009C              .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
                    000000A0  009C                        .BLKL   1
                              00A0              .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
                    000000A4  00A0                        .BLKL   1
                              00A4              .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
                    000000A6  00A4                        .BLKW   1
                              00A6              .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
                    000000A7  00A6                        .BLKB   1
                    000000A8  00A7                        .BLKB   1
                    00000074  00A8              . = 116
                    00000078  0074                        .BLKL   1
                    0000007C  0078                        .BLKL   1
                    00000080  007C                        .BLKL   1
                    00000084  0080                        .BLKL   1
                    00000088  0084                        .BLKL   1
                    0000008C  0088                        .BLKL   1
                              008C              .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
                    00000090  008C                        .BLKL   1
                    00000094  0090                        .BLKL   1
                    00000098  0094                        .BLKL   1
                    0000009C  0098                        .BLKL   1
                              009C              .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
                    0000009D  009C                        .BLKB   1
                              009D              .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
                    0000009E  009D                        .BLKB   1
                              009E              .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
                    0000009F  009E                        .BLKB   1
                    000000A0  009F                        .BLKB   1
                              00A0              .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
                    000000A1  00A0                        .BLKB   1
                              00A1              .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
                    000000A2  00A1                        .BLKB   1
                              00A2              .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
                    000000A3  00A2                        .BLKB   1
                    000000A4  00A3                        .BLKB   1
                              00A4              .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
                    000000A5  00A4                        .BLKB   1
```

```
                    00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                          .BLKW   1
000000A8  00A7                                          .BLKB   1
                    00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                          .BLKL   1
000000B0  00AC                                          .BLKL   1
000000B4  00B0                                          .BLKL   1
000000B8  00B4                                          .BLKL   1
                    00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                          .BLKL   1
                    00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                    00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC                          . = 116
                    0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                                          .BLKL   1
                    0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                          .BLKL   1
                    007C                .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                          .BLKW   1
00000080  007E                                          .BLKW   1
                    0000                .RESTORE
                    0000    65
                    0000    66 ;
                    0000    67 ; LOCAL SYMBOLS
                    0000    68 ;
                    0000    69 ; ARGUMENT LIST OFFSET DEFINITIONS
                    0000    70 ;
                    0000    71
00000004            0000    72 CHAN=4                                       ;I/O CHANNEL NUMBER
                    0000    73
                00000000    74          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                    0000    75        MODNAM  SYSDASSGN
4E 47 53 53 41 44 53 59 53 00' 0000     $MODULE:        .ASCIC  \SYSDASSGN\
                      09    0000
```

```
                                000A     77          .SBTTL  DEASSIGN I/O CHANNEL
                                000A     78 ;+
                                000A     79 ; EXE$DASSGN - DEASSIGN I/O CHANNEL
                                000A     80 ;
                                000A     81 ; THIS SERVICE DEASSIGNS A PREVIOUSLY ASSIGNED I/O CHANNEL AND CLEARS THE
                                000A     82 ; LINKAGE AND CONTROL INFORMATION IN THE CORRESPONDING CHANNEL CONTROL BLOCK.
                                000A     83 ; IF ANY I/O IS OUTSTANDING ON THE CHANNEL IT IS CANCELLED. IF A FILE IS
                                000A     84 ; OPEN ON THE CHANNEL IT IS CLOSED. IF A MAILBOX WAS ASSOCIATED WITH THE
                                000A     85 ; DEVICE WHEN IT WAS ASSIGNED, THE LINKAGE TO THE MAILBOX IS CLEARED. IF THE
                                000A     86 ; THE CHANNEL IS LAST ONE ASSIGNED TO THE DEVICE AND IT IS MARKED FOR DIS-
                                000A     87 ; MOUNT, THEN THE DISMOUNT IS COMPLETED.
                                000A     88 ;
                                000A     89 ; INPUTS:
                                000A     90 ;
                                000A     91 ;       CHAN(AP) = NUMBER OF THE I/O CHANNEL TO DEASSIGN.
                                000A     92 ;
                                000A     93 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                000A     94 ;
                                000A     95 ; OUTPUTS:
                                000A     96 ;
                                000A     97 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO DEASSIGN CHANNEL.
                                000A     98 ;
                                000A     99 ;              R0 = SS$_IVCHAN - INVALID CHANNEL NUMBER SPECIFIED.
                                000A    100 ;
                                000A    101 ;              R0 = SS$_NOPRIV - SPECIFIED CHANNEL IS NOT ASSIGNED TO A
                                000A    102 ;                   DEVICE OR THE CALLER DOES NOT HAVE SUFFICIENT
                                000A    103 ;                   PRIVILEGE TO ACCESS THE CHANNEL.
                                000A    104 ;
                                000A    105 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                000A    106 ;
                                000A    107 ;              R0 = SS$_NORMAL - NORMAL COMPLETION.
                                000A    108 ;-
                                000A    109
                          00FC  000A    110          .ENTRY  EXE$DASSGN,^M<R2,R3,R4,R5,R6,R7>
54  00000000'EF    9E         000C    111          MOVAB   DS$AX_SOFTPCB, R4       ; Software PCB to R4
     55  04 AC    3C         0013    112          MOVZWL  CHAN(AP),R5             ;GET CHANNEL NUMBER
        50  55    D0         0017    113          MOVL    R5,R0                   ;COPY I/O CHANNEL NUMBER
     00000000'EF    16         001A    114          JSB     IOC$VERIFYCHAN          ;VERIFY CHANNEL NUMBER                  [DS0
           21  50  E9         0020    115          BLBC    R0,50$                  ;IF LBC INVALID CHANNEL
         56  51    D0         0023    116          MOVL    R1,R6                   ;COPY ADDRESS OF CCB
         57  52    D0         0026    117          MOVL    R2,R7                   ;SAVE CHANNEL INDEX
                    0029    118          $CANCEL_S R5                    ;CANCEL I/O ON CHANNEL
         7E  55    3C         0029                 MOVZWL  R5,-(SP)
00000000'GF  01    FB         002C                 CALLS   #1,G^SYS$CANCEL
                    0033    119 20$:
            7E    DC         0033    120 30$:     MOVPSL  -(SP)                   ;SAVE CURRENT PROCESSOR STATUS
                    0035    121          SETIPL  #IPL$_ASTDEL            ;RAISE TO AST DELIVERY LEVEL
         12  02    DA         0035                 MTPR    #IPL$_ASTDEL,S^#PR$_IPL
           0A A6    B5         0038    122          TSTW    CCB$W_IOC(R6)           ;ANY I/O STILL OUTSTANDING?
               08    13         003B    123          BEQL    60$                     ;IF EQL NO
               8E    D5         003D    124          TSTL    (SP)+                   ; Remove PSL before repeat
                    003F    125 40$:     SETIPL  #0                      ;ALLOW INTERRUPTS
         12  00    DA         003F                 MTPR    #0,S^#PR$_IPL
               EF    11         0042    126          BRB     20$                     ;
                     04         0044    127 50$:     RET                             ;
                    0045    128 ;
                    0045    129 ; DEASSIGN CHANNEL AND CHECK IF THIS WAS THE LAST CHANNEL ASSIGNED TO DEVICE
```

```
                                0045   ;
                                0045   60$:
                55   66   D0   0045           MOVL    CCB$L_UCB(R6),R5          ;GET ASSIGNED DEVICE UCB ADDRESS
                  09 A6   94   0048           CLRB    CCB$B_AMOD(R6)            ;DEASSIGN CHANNEL
                  50 A5   B7   004B           DECW    UCB$W_REFC(R5)            ;DECREMENT DEVICE REFERENCE COUNT
                     14   13   004E           BEQL    70$                       ;IF EQL LAST REFERENCE
            50 A5    01   B1   0050           CMPW    #1,UCB$W_REFC(R5)         ;UCB REFERENCE COUNT ONE?
                     5E   12   0054           BNEQ    120$                      ;IF NEQ NO
         59 34 A5    00'  E1   0056           BBC     S^#DEV$V_ALL,UCB$L_DEVCHAR(R5),120$ ;IF CLR, DEVICE NOT ALLOCATED
         28 A5    60 A4   D1   005B           CMPL    PCB$L_PID(R4),UCB$L_PID(R5) ;ALLOCATED TO CURRENT PROCESS?
                     52   12   0060           BNEQ    120$                      ;IF NEQ NO
                     03   11   0062           BRB     80$                       ;
                                0064   ;
                                0064   ; DEALLOCATE DEVICE AND IF DISMOUNTING, CLEAR MOUNT OPTIONS AND DEALLOCATE VCB
                                0064   ;
                  28 A5   D4   0064   70$:    CLRL    UCB$L_PID(R5)             ;CLEAR OWNER PROCESS ID
         25 34 A5    00'  E1   0067   80$:    BBC     S^#DEV$V_DMT,UCB$L_DEVCHAR(R5),90$ ;IF CLR, NOT MARKED FOR DISMOUNT
         20 34 A5    00'  E1   006C           BBC     S^#DEV$V_FOR,UCB$L_DEVCHAR(R5),90$ ;IF CLR, STRUCTURED VOLUME
                          CA   0071           BICL    #DEV$M_DMT!DEV$M_FOR!-    ;CLEAR MARKED FOR DISMOUNT, FOREIGN, AND
                                0072                   DEV$M_RCK!DEV$M_WCK!-     ;READ/WRITE CHECK, AND
                                0072                   DEV$M_SWL!-               ;SOFTWARE WRITE LOCKED, AND
   34 A5   00000000'8F          0072                   DEV$M_MNT,UCB$L_DEVCHAR(R5) ;MOUNTED
            50   30 A5   D0   0079           MOVL    UCB$L_VCB(R5),R0          ;GET ADDRESS OF VCB
                     12   13   007D           BEQL    90$                       ;IF EQL NONE
                  30 A5   D4   007F           CLRL    UCB$L_VCB(R5)             ;CLEAR ADDRESS OF VCB
         00000000'EF      16   0082           JSB     EXE$DEANONPAGED           ;DEALLOCATE VCB
                  1A A5   B4   0088           CLRW    UCB$W_VPROT(R5)           ;CLEAR VOLUME PROTECTION MASK
      58 A5    0800 8F   AA   008B           BICW    #UCB$M_VALID,UCB$W_STS(R5) ;CLEAR SOFTWARE VOLUME VALID
                                0091   ;
                                0091   ; CALL DRIVER'S CANCEL ROUTINE FOR ANY SPECIAL CLEANUP
                                0091   ;
            50   24 A5   D0   0091   90$:    MOVL    UCB$L_DDB(R5),R0          ;GET ADDRESS OF DDB
            50   0C A0   D0   0095           MOVL    DDB$L_DDT(R0),R0          ;GET ADDRESS OF DDT
                                0099           SETIPL  UCB$B_FIPL(R5)            ;RAISE TO DEVICE FORK IPL
            12   0B A5   DA   0099                   MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
            53   4C A5   D0   009D           MOVL    UCB$L_IRP(R5),R3          ;GET CURRENT I/O PACKET ADDRESS
               52   57   D0   00A1           MOVL    R7,R2                     ;SET CHANNEL INDEX
            51   0C A0   D0   00A4           MOVL    DDT$L_CANCEL(R0),R1       ;GET ADDRESS OF CANCEL I/O ENTRY POINT
            03 51   10   E0   00A8           BBS     #16,R1,100$               ;++++ BRANCH IF SUPERVISOR ABSOLUTE
               51   50   C0   00AC           ADDL    R0,R1                     ;++++ MAKE ABSOLUTE ADDRESS
                     61   16   00AF   100$:   JSB     (R1)                      ;CALL CANCEL I/O ROUTINE
                                00B1           SETIPL  #IPL$_ASTDEL              ;LOWER TO AST DELIVERY LEVEL
               12   02   DA   00B1                   MTPR    #IPL$_ASTDEL,S^#PR$_IPL
               50   01   D0   00B4   120$:   MOVL    #SS$_NORMAL, R0           ; Set success code
         00000000'EF      17   00B7           JMP     IOC$UNLOCK                ;UNLOCK I/O DATA BASE AND RETURN
                                00BD
                                00BD           .END
```

```
$ER                             = 00000001     D
$MODULE                           00000000 R   D   02
CCB$B_AMOD                        00000009     D
CCB$B_STS                         00000008     D
CCB$C_LENGTH                      00000010     D
CCB$K_LENGTH                      00000010     D
CCB$L_DIRP                        0000000C     D
CCB$L_UCB                         00000000     D
CCB$L_WIND                        00000004     D
CCB$W_IOC                         0000000A     D
CHAN                            = 00000004     D
DDB$B_ACPCLASS                    00000013     D
DDB$B_TYPE                        0000000A     D
DDB$C_LENGTH                      00000034     D
DDB$K_LENGTH                      00000034     D
DDB$L_ACPD                        00000010     D
DDB$L_DDT                         0000000C     D
DDB$L_LINK                        00000000     D
DDB$L_UCB                         00000004     D
DDB$T_DRVNAME                     00000024     D
DDB$T_NAME                        00000014     D
DDB$W_SIZE                        00000008     D
DDT$L_ALTSTART                    0000001C     D
DDT$L_CANCEL                      0000000C     D
DDT$L_FDT                         00000008     D
DDT$L_REGDUMP                     00000010     D
DDT$L_START                       00000000     D
DDT$L_UNITINIT                    00000018     D
DDT$L_UNSOLINT                    00000004     D
DDT$W_DIAGBUF                     00000014     D
DDT$W_ERRORBUF                    00000016     D
DEV$M_DMT                         ********     X   02
DEV$M_FOR                         ********     X   02
DEV$M_MNT                         ********     X   02
DEV$M_RCK                         ********     X   02
DEV$M_SWL                         ********     X   02
DEV$M_WCK                         ********     X   02
DEV$V_ALL                         ********     X   02
DEV$V_DMT                         ********     X   02
DEV$V_FOR                         ********     X   02
DS$AX_SOFTPCB                     ********     X   02
EXE$DASSGN                        0000000A  RG D   02
EXE$DEANONPAGED                   ********     X   02
IOC$UNLOCK                        ********     X   02
IOC$VERIFYCHAN                    ********     X   02
IPL$_ASTDEL                     = 00000002     D
PCB$B_ASTACT                      0000000C     D
PCB$B_ASTEN                       0000000D     D
PCB$B_PRI                         0000000B     D
PCB$B_PRIB                        0000002F     D
PCB$B_TYPE                        0000000A     D
PCB$B_WEFC                        0000002E     D
PCB$C_LENGTH                      0000008C     D
PCB$K_LENGTH                      0000008C     D
PCB$L_ARB                         00000084     D
PCB$L_ASTQBL                      00000014     D
PCB$L_ASTQFL                      00000010     D
PCB$L_EFC2P                       00000058     D
PCB$L_EFC3P                       0000005C     D
PCB$L_EFCS                        00000050     D
PCB$L_EFCU                        00000054     D
PCB$L_EFWM                        0000004C     D
PCB$L_JIB                         00000078     D
PCB$L_OWNER                       0000001C     D
PCB$L_PHD                         00000064     D
PCB$L_PHYPCB                      00000018     D
PCB$L_PID                         00000060     D
PCB$L_PQB                         0000004C     D
PCB$L_SQBL                        00000004     D
PCB$L_SQFL                        00000000     D
PCB$L_STS                         00000024     D
PCB$L_UIC                         00000088     D
PCB$L_WSSWP                       00000020     D
PCB$L_WTIME                       00000028     D
PCB$Q_PRIV                        0000007C     D
PCB$T_LNAME                       00000068     D
PCB$T_TERMINAL                    00000044     D
PCB$W_APTCNT                      00000030     D
PCB$W_ASTCNT                      00000038     D
PCB$W_BIOCNT                      0000003A     D
PCB$W_BIOLM                       0000003C     D
PCB$W_DIOCNT                      0000003E     D
PCB$W_DIOLM                       00000040     D
PCB$W_GPGCNT                      00000034     D
PCB$W_GRP                         0000008A     D
PCB$W_MEM                         00000088     D
PCB$W_MTXCNT                      0000000E     D
PCB$W_PPGCNT                      00000036     D
PCB$W_PRCCNT                      00000042     D
PCB$W_SIZE                        00000008     D
PCB$W_STATE                       0000002C     D
PCB$W_TMBU                        00000032     D
PR$_IPL                         = 00000012     D
SIZ...                          = 00000002     D
SS$_NORMAL                      = 00000001     D
SYS$CANCEL                        ********   GX   02
UCB$B_AMOD                        00000053     D
UCB$B_CEX                         00000077     D
UCB$B_CM1                         0000004A     D
UCB$B_CM2                         0000004B     D
UCB$B_DEVCLASS                    00000038     D
UCB$B_DEVTYPE                     00000039     D
UCB$B_DIPL                        00000052     D
UCB$B_DX_SCTCNT                   000000A6     D
UCB$B_ERTCNT                      00000070     D
UCB$B_ERTMAX                      00000071     D
UCB$B_FEX                         00000076     D
UCB$B_FIPL                        0000000B     D
UCB$B_LOCSRV                      0000003C     D
UCB$B_OFFNDX                      00000094     D
UCB$B_OFFRTC                      00000095     D
UCB$B_REMSRV                      0000003D     D
UCB$B_SECTORS                     0000003C     D
UCB$B_SLAVE                       00000074     D
```

UCB$B_SPR	00000075	D	UCB$L_RQBL		00000004	D
UCB$B_STATE	00000052	D	UCB$L_RQFL		00000000	D
UCB$B_TRACKS	0000003D	D	UCB$L_SVAPTE		00000068	D
UCB$B_TT_CRFILL	0000009D	D	UCB$L_SVPN		00000064	D
UCB$B_TT_DECRF	000000A1	D	UCB$L_TT_DECHAR		000000A8	D
UCB$B_TT_DELFF	000000A2	D	UCB$L_TT_RDUE		0000008C	D
UCB$B_TT_DESPEE	000000A0	D	UCB$L_TT_RTIMOU		000000B8	D
UCB$B_TT_DETYPE	000000A4	D	UCB$L_VCB		00000030	D
UCB$B_TT_LFFILL	0000009E	D	UCB$M_VALID	=	00000800	D
UCB$B_TT_SPEED	0000009C	D	UCB$T_PARTNER		0000000C	D
UCB$B_TYPE	0000000A	D	UCB$W_BCNT		0000006E	D
UCB$B_VERTSZ	0000003F	D	UCB$W_BCR		00000096	D
UCB$C_LENGTH	00000074	D	UCB$W_BOFF		0000006C	D
UCB$C_MB_LENGTH	00000090	D	UCB$W_BUFQUO		00000018	D
UCB$C_TT_LENGTH	000000BC	D	UCB$W_BYTESTOGO		0000003E	D
UCB$K_LENGTH	00000074	D	UCB$W_CHARGE		0000004A	D
UCB$K_MB_LENGTH	00000090	D	UCB$W_CYLINDERS		0000003E	D
UCB$K_TT_LENGTH	000000BC	D	UCB$W_DA		0000008C	D
UCB$L_AMB	00000054	D	UCB$W_DC		0000008E	D
UCB$L_ASTQBL	00000010	D	UCB$W_DEVBUFSIZ		0000003A	D
UCB$L_ASTQFL	0000000C	D	UCB$W_DEVSTS		0000005A	D
UCB$L_CPID	0000005C	D	UCB$W_DIRSEQ		00000088	D
UCB$L_CRB	00000020	D	UCB$W_DSTADDR		00000018	D
UCB$L_DDB	00000024	D	UCB$W_DX_BCR		000000A4	D
UCB$L_DEVCHAR	00000034	D	UCB$W_EC1		00000090	D
UCB$L_DEVDEPEND	0000003C	D	UCB$W_EC2		00000092	D
UCB$L_DPC	00000080	D	UCB$W_ERRCNT		00000072	D
UCB$L_DUETIM	0000005C	D	UCB$W_FUNC		0000007E	D
UCB$L_DX_BFPNT	0000009C	D	UCB$W_MB_SEED		00000000	D
UCB$L_DX_BUF	00000098	D	UCB$W_MSGCNT		00000016	D
UCB$L_DX_RXDB	000000A0	D	UCB$W_MSGMAX		00000014	D
UCB$L_EMB	00000078	D	UCB$W_NT_CHAN		0000007C	D
UCB$L_FIRST	00000014	D	UCB$W_OFFSET		0000008A	D
UCB$L_FPC	0000000C	D	UCB$W_REFC		00000050	D
UCB$L_FQBL	00000004	D	UCB$W_SIZE		00000008	D
UCB$L_FQFL	00000000	D	UCB$W_SRCADDR		0000001A	D
UCB$L_FR3	00000010	D	UCB$W_STS		00000058	D
UCB$L_FR4	00000014	D	UCB$W_TT_DESIZE		000000A5	D
UCB$L_IOQBL	00000044	D	UCB$W_UNIT		00000048	D
UCB$L_IOQFL	00000040	D	UCB$W_VPROT		0000001A	D
UCB$L_IRP	0000004C	D				
UCB$L_LINK	0000002C	D				
UCB$L_LOGADR	00000064	D				
UCB$L_MAXBLOCK	00000084	D				
UCB$L_MB_MBX	0000007C	D				
UCB$L_MB_PORT	0000008C	D				
UCB$L_MB_RAST	00000078	D				
UCB$L_MB_SHB	00000080	D				
UCB$L_MB_WAST	00000074	D				
UCB$L_MB_WIOQBL	00000088	D				
UCB$L_MB_WIOQFL	00000084	D				
UCB$L_MEDIA	0000008C	D				
UCB$L_NT_DATSSB	00000074	D				
UCB$L_NT_INTSSB	00000078	D				
UCB$L_OPCNT	00000060	D				
UCB$L_OWNUIC	0000001C	D				
UCB$L_PID	00000028	D				

+-----------------+
! Psect synopsis !
+-----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	000000BD (189.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	75 (1)	
$MODULE	00000000-R	75 (1)	
CCB$B_AMOD	00000009		#-133 (2)
CCB$L_UCB	00000000		#-132 (2)
CCB$W_IOC	0000000A		#-122 (2)
CHAN	=00000004	72 (1)	#-112 (2)
DDB$L_DDT	0000000C		#-162 (2)
DDT$L_CANCEL	0000000C		#-166 (2)
DEV$M_DMT	00000000-XR		#-148 (2)
DEV$M_FOR	00000000-XR		#-148 (2)
DEV$M_MNT	00000000-XR		#-151 (2)
DEV$M_RCK	00000000-XR		#-149 (2)
DEV$M_SWL	00000000-XR		#-150 (2)
DEV$M_WCK	00000000-XR		#-149 (2)
DEV$V_ALL	00000000-XR		#-138 (2)
DEV$V_DMT	00000000-XR		#-146 (2)
DEV$V_FOR	00000000-XR		#-147 (2)
DS$AX_SOFTPCB	00000000-XR		111 (2)
EXE$DASSGN	0000000A-R	110 (2)	
EXE$DEANONPAGED	00000000-XR		155 (2)
IOC$UNLOCK	00000000-XR		172 (2)
IOC$VERIFYCHAN	00000000-XR		114 (2)
IPL$_ASTDEL	=00000002		#-121 (2) #-170 (2)
PCB$L_PID	00000060		#-139 (2)
PR$_IPL	=00000012		#-121 (2) #-125 (2) #-163 (2) #-170 (2)
SS$_NORMAL	=00000001		#-171 (2)
SYS$CANCEL	00000000-XR		118 (2)
UCB$B_FIPL	0000000B		#-163 (2)
UCB$L_DDB	00000024		#-161 (2)
UCB$L_DEVCHAR	00000034		138 (2) 146 (2) 147 (2) #-151 (2)
UCB$L_IRP	0000004C		#-164 (2)
UCB$L_PID	00000028		#-139 (2) #-145 (2)
UCB$L_VCB	00000030		#-152 (2) #-154 (2)
UCB$M_VALID	=00000800		#-157 (2)
UCB$W_REFC	00000050		#-134 (2) #-136 (2)
UCB$W_STS	00000058		#-157 (2)
UCB$W_VPROT	0000001A		#-156 (2)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CANCEL_S	1	118 (2)	118 (2)
$CCBDEF	1	55 (1)	55 (1)
$DDBDEF	1	56 (1)	56 (1)
$DDTDEF	1	57 (1)	57 (1)
$DEFINI	1	55 (1)	55 (1) 56 (1) 57 (1) 58 (1) 59 (1)
			60 (1) 61 (1) 62 (1) 63 (1) 64 (1)
$IODEF	15	58 (1)	58 (1)
$IPLDEF	1	59 (1)	59 (1)
$PCBDEF	4	60 (1)	60 (1)
$PRDEF	5	61 (1)	61 (1)
$RSNDEF	1	62 (1)	62 (1)
$SSDEF	21	63 (1)	63 (1)
$UCBDEF	10	64 (1)	64 (1)
MODNAM	1	75 (1)	75 (1)
SETIPL	1	121 (2)	121 (2) 125 (2) 163 (2) 170 (2)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	168	(2)												
BBC	00E1	138	(2)	146	(2)	147	(2)								
BBS	00E0	167	(2)												
BEQL	0013	123	(2)	135	(2)	153	(2)								
BICL	00CA	148	(2)												
BICW	00AA	157	(2)												
BLBC	00E9	115	(2)												
BNEQ	0012	137	(2)	140	(2)										
BRB	0011	126	(2)	141	(2)										
CALLS	00FB	118	(2)												
CLRB	0094	133	(2)												
CLRL	00D4	145	(2)	154	(2)										
CLRW	00B4	156	(2)												
CMPL	00D1	139	(2)												
CMPW	00B1	136	(2)												
DECW	00B7	134	(2)												
JMP	0017	172	(2)												
JSB	0016	114	(2)	155	(2)	169	(2)								
MOVAB	009E	111	(2)												
MOVL	00D0	113	(2)	116	(2)	117	(2)	132	(2)	152	(2)	161	(2)	162	(2)
		164	(2)	165	(2)	166	(2)	171	(2)						
MOVPSL	00DC	120	(2)												
MOVZWL	003C	112	(2)	118	(2)										
MTPR	00DA	121	(2)	125	(2)	163	(2)	170	(2)						
RET	0004	127	(2)												
TSTL	00D5	124	(2)												
TSTW	00B5	122	(2)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	75	(1)														
.END	174	(2)														
.ENDC	121	(2)	125	(2)	163	(2)	170	(2)								
.ENDM	118	(2)	121	(2)	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)
	61	(1)	62	(1)	63	(1)	64	(1)	75	(1)						
.ENTRY	110	(2)														
.GLOBL	118	(2)														
.IDENT	5	(1)														
.IF	121	(2)	125	(2)	163	(2)	170	(2)								
.IFF	121	(2)	125	(2)	163	(2)	170	(2)								
.NOCROSS	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)												
.PSECT	74	(1)														
.RESTORE	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)												
.SAVE	55	(1)	56	(1)	57	(1)	58	(1)	59	(1)	60	(1)	61	(1)	62	(1)
	63	(1)	64	(1)												
.SBTTL	77	(2)														
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	1	#-112	(2)										
R0	9	#-113	(2)	#-115	(2)	#-152	(2)	#-161	(2)	#-162	(2)	#-166	(2)
		#-168	(2)	#-171	(2)								
R1	5	#-116	(2)	#-166	(2)	167	(2)	#-168	(2)	169	(2)		
R2	3	110	(2)	#-117	(2)	#-165	(2)						
R3	2	110	(2)	#-164	(2)								
R4	3	110	(2)	#-111	(2)	#-139	(2)						
R5	20	110	(2)	#-112	(2)	#-113	(2)	#-118	(2)	#-132	(2)	#-134	(2)
		#-136	(2)	138	(2)	#-139	(2)	#-145	(2)	146	(2)	147	(2)
		#-151	(2)	#-152	(2)	#-154	(2)	#-156	(2)	#-157	(2)	#-161	(2)
		#-163	(2)	#-164	(2)								
R6	5	110	(2)	#-116	(2)	#-122	(2)	#-132	(2)	#-133	(2)		
R7	3	110	(2)	#-117	(2)	#-165	(2)						
SP	3	#-118	(2)	#-120	(2)	#-124	(2)						

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.20
Command processing	893	00:00:00.23	00:00:00.82
Pass 1	742	00:00:02.62	00:00:04.12
Symbol table sort	0	00:00:00.33	00:00:00.34
Pass 2	119	00:00:00.67	00:00:01.27
Symbol table output	2	00:00:00.03	00:00:00.09
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	5	00:00:00.06	00:00:00.15
Assembler run totals	1787	00:00:03.97	00:00:07.00

The working set limit was 2400 pages.
148416 bytes (290 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1158 non-local and 10 local symbols.
174 source lines were read in Pass 1, producing 76 object records in Pass 2.
63 pages of virtual memory were used to define 22 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	6
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	2
SYS$COMMON:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	18

1459 GETS were required to define 18 macros.

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 7 Frame J1          Sequence 1245
DASSGN                          *** DASSGN deassign QIO channel      11-NOV-1987 17:31:53  VAX/VMS Macro V04-00        Page  20
VAX-11 Macro Run Statistics                                           3-JAN-1985 16:50:02  DASSGN.MAR;1                (2)

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DASSGN.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(2) 179 DECLARATIONS
(7) 429 SET BASE ADDRESS SUBROUTINE
(8) 472 SHOW BASE ADDRESS SUBROUTINE ; [13]
(9) 514 SET DEFAULTS SUBROUTINE
(11) 587 SHOW DEFAULTS SUBROUTINE
(13) 663 EXAMINE MEMORY AND REGISTERS SUBROUTINE
(15) 852 VRTYPEXAMINE Type out location EXAMINE'd
(20) 980 DEPOSIT MEMORY AND REGISTERS SUBROUTINE
(23) 1171 SET BREAKPOINT SUBROUTINE
(25) 1246 CLEAR BREAKPOINT SUBROUTINE
(27) 1313 DSP$REMOVEBREAK Remove breakpoints
(28) 1369 SHOW BREAKPOINTS SUBROUTINE
(29) 1420 FIND_BPT FIND BPT IN BPT TABLE
(30) 1460 DEBUGGER CONDITION HANDLER
(31) 1533 BREAKPOINT CONDITION SUBROUTINE
(33) 1704 TBIT CONDITION SUBROUTINE
(35) 1763 RESTORE SCB VECTORS SUBROUTINE ;G:5
(36) 1810 TEMPORARY SCB VECTOR STORAGE SUBROUTINE ;G:5
(37) 1869 TEMPORARY BREAKPOINT REMOVAL SUBROUTINE
(38) 1911 BREAKPOINT REPLACEMEMT SUBROUTINE
(39) 1964 FETCH_STORE COPY BYTE ROUTINE
(42) 2186 Handle EXAMINE/DEPOSIT of IPRs in KERNEL mode
(42) 2210 FETCH_STOR_PREG Move data to/from IPR
(42) 2255 DSV$DEBUG - Process DEBUG Command [14]

```
0000    ; DEC/CMS REPLACEMENT HISTORY, Element DEBUG.MAR
0000    ; *8    16-SEP-1987 15:36:09 GEARIN "ADDED GLOBAL TO STORE VALUE OF IPRS"
0000    ; *7    14-MAY-1987 10:11:46 FORTUNA "MODIFY CODE TO PASS PARAMETERS OF BASE ADDR AN
0000    ; *6    12-MAY-1987 11:25:03 FORTUNA "REFORM THE OUTPUT LINE OF A SHOW BREAK AND BRE
0000    ;OFFSET"
0000    ; *5    25-MAR-1987 11:03:02 GEARIN "ADDED CODE TO SAVE/RESTORE CERTAIN SCB VECTORS"
0000    ; *4    29-JAN-1987 15:05:44 GEARIN "INSTRUCTION NOW SHOWS ACTUAL OPCODE NOT FOR BPT
0000    ; *3    23-JUL-1986 14:27:34 SALEM "DONE"
0000    ; *2    30-APR-1986 10:02:00 ANDELLA "<no reason given>"
0000    ; *1     6-FEB-1986 15:16:49 DS ""
0000    ; DEC/CMS REPLACEMENT HISTORY, Element DEBUG.MAR
0000            .TITLE  DEBUG   EXAMINE, DEPOSIT, AND BREAKPOINT SUBROUTINES          ;G:5
0000            .IDENT  /10.10-8/                                                     ;G:8
0000            .NLIST  CND
0000            .LIST   MEB
0000
0000    ;
0000    ; COPYRIGHT (C) 1977, 1980, 1984, 1985, 1986, 1987                            ;G:4
0000    ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    ;
0000    ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    ; REMAIN IN DEC.
0000    ;
0000    ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    ; CORPORATION.
0000    ;
0000    ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000
0000    ;++
0000    ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ; ENVIRONMENT:
0000    ;
0000    ; AUTHOR:       TOM SOUTTER     10-NOV-77       VERSION 01
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;               TOM SOUTTER     18-NOV-77       VERSION 02
0000    ;       01      DSPR #10 - LOCATE BREAKPOINTS VIA OPERATOR SPECIFIED BASE
0000    ;
0000    ;               KEN CHAPMAN     16-FEB-78       VERSION 03 (ESSAA-3.07).
0000    ;       02      REMOVED EXCEPTION HANDLER TO SCB.
0000    ;               Roger Riggs     18-JUL-78               (4.03)
0000    ;       03      Changed SS$_ARITH TO DS$_ARITH
0000    ;       04      Various enhancements and cleanup done,
0000    ;               'NEXT [n]' command.
0000    ;       05      Added code to restore opcode on hidden breakpoint
0000    ;       06      Modified FETCH_STORE to correctly report address on
```

```
                                0000     58 ;               COMET machine checks
                                0000     59 ;               Roger Riggs     29-NOV-1979
                                0000     60 ;       07      Added BSBW SCRIPT$STOP before call to CLI to halt script
                                0000     61 ;               mechanism on breakpoints and single step.
                                0000     62 ;               Dave Butenhof  16-JAN-80
                                0000     63 ;       08      Add code to examine/deposit internal processor registers.
                                0000     64 ;               Roger Riggs, 27-May-1980, Version 5.4
                                0000     65 ;       09      Changed radix on processor register output to hex.
                                0000     66 ;
                                0000     67 ;               Dave Butenhof, 30-sep-1980, Version 6.1
                                0000     68 ;       10      Make PSL typeout unaffected by /WORD or /BYTE.
                                0000     69 ;               Also change /ASCII to use !AF FAO directive, and also type
                                0000     70 ;               the longword in hex to aid in decoding non-printing strings.
                                0000     71 ;               Dave Butenhof, 23-feb-1981, version 6.3
                                0000     72 ;       11      fix truncation errors
                                0000     73 ;
                                0000     74 ;       12      - Jack Stansbury, 22-Oct-1981, Version 6.-
                                0000     75 ;               Fixed truncation errors. Also added .LIBRARY statements
                                0000     76 ;               for $DS and $DIAG.
                                0000     77 ;
                                0000     78 ;       13      - Jack Stansbury, 10-Nov-1981, Version 6.-
                                0000     79 ;               Added routines for VRSHOWBASE (to show the base address)
                                0000     80 ;               and VRSHOWDFLT (to show the default size and radix). Also
                                0000     81 ;               changed the error messages to mixed case.
                                0000     82 ;
                                0000     83 ;       14      - Richard Brown, 19-May-82, Version 6.8
                                0000     84 ;               Addition of routine to process the DEBUG user command.
                                0000     85 ;
                                0000     86 ;       15      - Domenic Andella, 17-Jul-84, Version 7.0
                                0000     87 ;               The 'DEP' command does a write followed by
                                0000     88 ;               read to display the data written. This causes
                                0000     89 ;               a mchk on VENUS when depositing a one into
                                0000     90 ;               the Unibus Adapter Control Register (UACR),
                                0000     91 ;               and could cause similar problems in other
                                0000     92 ;               new CPU's. Problem solved by removing the
                                0000     93 ;               final examine from the 'DEPOSIT' command.
                                0000     94 ;               This makes VDS consistent with the DEPOSIT
                                0000     95 ;               command in XDELTA and DEBUG.
                                0000     96 ;
                                0000     97 ;       16      Domenic Andella, 4-Apr-1985     Version 8.2
                                0000     98 ;               Add mod to COND_DEBUG to call MP$_STOP_EVERYTHING
                                0000     99 ;               if mp system.
                                0000    100 ;
                                0000    101 ;
                                0000    102 ;       17      Ted Salem       23_April-1985   Version 8.2
                                0000    103 ;               Interlocked COND_DEBUG.
                                0000    104 ;
                                0000    105 ;       18      Ted Salem       6-May-1985      Version 8.2
                                0000    106 ;               Added code in order to Examine and Deposit from and
                                0000    107 ;               to an AP.
                                0000    108 ;
                                0000    109 ;       19      Domenic Andella 22-May-1985     Version 8.2
                                0000    110 ;               Add code in COND_DEBUG to support breakpoints in a
                                0000    111 ;               multi-processing system. This edit changes #16.
                                0000    112 ;
                                0000    113 ;       20      Jim Baranski    24-June-1985    Version 8.3
                                0000    114 ;               Add code in VREXAMINE, VRDEPOSIT, VRSETBRK, VRCLRBRK,
```

```
0000   115 ;                for /BASE:temp_base & /NOBASE.
0000   116 ;
0000   117 ;        21      Domenic Andella 2-July-1985     Version 8.3
0000   118 ;                Repositioned mp interlock commands in routine
0000   119 ;                COND_DEBUG to solve interlock problem with
0000   120 ;                BCI750 diagnostic (EVCKA T18-s3) that holds
0000   121 ;                interlock time for long duration.
0000   122 ;
0000   123 ;        22      Ted Salem       25-July-1985    Version 8.3
0000   124 ;                Fixed a bug in DSV$DEBUG.  DS$LOAD was not being
0000   125 ;                called with the proper number of arguments.
0000   126 ;
0000   127 ;        23      Domenic Andella 21-August-1985  Version 9.0
0000   128 ;                Reinstated edit 21, fixed bug in DBG_BREAK
0000   129 ;                fetching cpu identifier.
0000   130 ;
0000   131 ;        24      Domenic Andella 9-Jan-1986      Version 9.1
0000   132 ;                PRINTF for Set Cpu command requires address
0000   133 ;                argument for MP$GT_DEVICE_NAME, in routine
0000   134 ;                VRTYPEXAMINE.
0000   135 ;                                                                ;G:2
0000   136 ;        25      Domenic Andella 29-Apr-1986     Version 10.0    ;G:2
0000   137 ;                Make fix in routine VREXAMINE to load address   ;G:2
0000   138 ;                of an Ap's ipr data. Re-instate instruction     ;G:2
0000   139 ;                to load IPR data in routine VRDEPOSIT. This     ;G:2
0000   140 ;                instruction disappeared from V8.3.              ;G:6
0000   141 ;                                                                ;G:2
0000   142 ;        26      Ted Salem       8-Jul_1986      Version 10.1    ;G:3
0000   143 ;                Made dbg_addr_rtn a global symbol               ;G:3
0000   144 ;                                                                ;G:3
0000   145 ;        27      David Gearin    29-Jan-1987     VERSION 10.5    ;G:5
0000   146 ;                Modified DBG_TBIT subroutine so the call to     ;G:6
0000   147 ;                RBREAK_IN is performed after the call to FETCH_ ;G:4
0000   148 ;                STORE.  This permits the actual opcode to be shown ;G:4
0000   149 ;                while stepping through a program instead of '03' ;G:6
0000   150 ;                (the opcode for Breakpoint). Also added another call ;G:4
0000   151 ;                to RBREAK_IN in case the BBCC command does branch. ;G:4
0000   152 ;                This way the breakpoints will be reinserted.    ;G:4
0000   153 ;                                                                ;G:4
0000   154 ;        28      David Gearin    12-Feb-1987     Version 10.8    ;G:5
0000   155 ;                Added sub-routines SAVE_SCB_VECTORS and RESTORE_SCB_VECTORS ;G:5
0000   156 ;                to override a diagnostic program's exception handler during ;G:5
0000   157 ;                an examine or deposit command.                  ;G:5
0000   158 ;                                                                ;G:5
0000   159 ;                Steven Fortuna  11-May-1987     Version 10.8    ;G:6
0000   160 ;                Add output lines to print the Base and Offset when a show ;G:6
0000   161 ;                break is entered or a break point is hit.  Also add the ;G:6
0000   162 ;                capability to modify the break points Base addres and offset ;G:6
0000   163 ;                by entering the SET BREAK/BASE=.... command.  No symptom ;G:6
0000   164 ;                ovserved.                                       ;G:6
0000   165 ;                                                                ;G:7
0000   166 ;                Steven Fortuna  12-May-1987     Version 10.8    ;G:8
0000   167 ;                Add Parameters to pass the base address and offset of ;G:8
0000   168 ;                breakpoints to MPCOMMON so Ap breakpoint message would ;G:7
0000   169 ;                be consent with Primary breakpoint message.     ;G:7
0000   170 ;                No symptom ovserved.                            ;G:7
0000   171 ;                                                                ;G:8
```

```
0000   172 ;              David Gearin    Sept-15-1987    Version 10.10                ;G:8
0000   173 ;              Added an external variable, ds$gl_ipr_value                  ;G:8
0000   174 ;              This variable will be used to store ipr values to            ;G:8
0000   175 ;              be used by bliss routines.                                   ;G:8
0000   176 ;--                                                                         ;G:6
0000   177                                                                            ;G:6
```

```
                     0000    179          .SBTTL  DECLARATIONS
                     0000    180 ;
                     0000    181 ; INCLUDE FILES:
                     0000    182 ;
                     0000    183          .LIBRARY        /$DS/                   ;                               [12]
                     0000    184          .LIBRARY        /$DIAG/                 ;                               [12]
                     0000    185
                     0000    186 ;
                     0000    187 ; MACROS:
                     0000    188 ;
                     0000    189          $MP_ILdefs                      ; mp$R_active_routines  offsets  [17]
                     0000    190
                     0000    191 ;
                     0000    192 ; EXTERNALS:
                     0000    193 ;
                     0000    194          .EXTRN  DS$GL_IPR_VALUE                                                 ;G:8
                     0000    195          .EXTRN  MP$GB_SETCPU,MP$GT_DEVICE_NAME                  ;       [18]
                     0000    196          .EXTRN  MP$_GET_IPR,MP$_GET_GPR                         ;       [18]
                     0000    197          .EXTRN  MP$GR_BOOTED_PROCESSORS                         ;       [19]
                     0000    198
                     0000    199          .WEAK   MP$_GET_IPR,MP$_GET_GPR                         ;       [18]
                     0000    200          .WEAK   MP$GL_AP_BKPT_STORE                             ;       [19]
                     0000    201          .WEAK   MP$GL_AP_BASE_ADDR_BKPT                                         ;G:8
                     0000    202          .WEAK   MP$GL_AP_OFFSET_BKPT                                            ;G:8
                     0000    203          .WEAK   MP$_COND_DEBUG                                  ;       [19]
                     0000    204          .WEAK   MP$_GET_PROCESSOR_ID                            ;       [19]
                     0000    205          .WEAK   MP$_RESUME_ATTACHED_PROCESSORS                  ;       [19]
                     0000    206          .WEAK   MP$GR_SAVED_GPRS                                ;       [19]
                     0000    207
                     0000    208 ;
                     0000    209 ; EQUATED SYMBOLS:
                     0000    210 ;
                     0000    211
                     0000    212
00000009             0000    213          HTAB      =   9                 ; ASCII HORIZONTAL TAB
00000003             0000    214          BPTCODE   == 03                 ; BREAKPOINT OPCODE
00000010             0000    215          PSLREG    =  16                 ; ERSATZ ADDRESS FOR 'PSL'
0000000F             0000    216          BPTMAX    == 15                 ; MAXIMUM NUMBER OF BREAKPOINTS
00000020             0000    217          EXAMSIZE  =  32                 ; SIZE OF DISPLAY BUFFER FOR EXAMINE
00000050             0000    218          PSLBUFSZ  =  80                 ; SIZE OF PSL MNEMONICS BUFFER
00000000             0000    219          INITBASE  =   0                 ; INITIAL BASE ADDRESS FOR EXAM/DEP
00000004             0000    220          INITSIZE  =   4                 ; INITIAL BYTE COUNT FOR EXAM/DEP
                     0000    221
00000001             0000    222          SS$_NORMAL      = 1             ; SUCCESSFUL COMPLETION
                     0000    223
00010E00             0000    224          DEBUGGER_SIZE = 135 * 512       ;NO. OF PAGES TIMES BYTES PER PAGE[14] ;G:5
                     0000    225
                     0000    229          CLIDEF                          ; CLI COMMAND BLOCK OFFSETS
                     0000    230          DSFDEF                          ; CONTROL FLAG DEFINITIONS
                     0000    231          $DS_DSDEF                       ; SUPERVISOR STATUS CODES
                     0000    232          $PSLDEF                         ; PROCESSOR STATUS BITS
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .PSECT  $ABS$,ABS
00000000             044C                 .=0
                     0000                 .RESTORE
                     0000    233          $PRDEF                          ; PROCESSOR REGISTERS
                     0000                 .SAVE   LOCAL_BLOCK
```

```
                    0000                .PSECT  $ABS$,ABS
00000000            044C                .=0
                    0000                .RESTORE
                    0000    234         $SETPRTDEF                      ; SETPRT arglist offsets
                    0000    235         $DS_DSADEF                      ; Define DSA$x codes
                    0000    236         CMKDEF                          ; Define CHMK codes
```

```
                                  0000  ; GLOBAL REFERENCES                                                    [14]
                                  0000
                                  0000          .GLOBAL DIAG_FILE_NAME          ;FILENAME OF LAST DIAG. LOADED  [14]
                                  0000
                                  0000  ;
                                  0000  ; OWN STORAGE:
                                  0000  ;
                                  0000
                              00000000          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                  0000
                                  0000  DEBUG$GB_FLAGS::
                                  00 0000          .BYTE   0
                          00000000 0001                  V_TBIT=0                ; T-BIT TRAP EXPECTED
                          00000001 0001                          M_TBIT=1 @ V_TBIT
                          00000001 0001                  V_STEP=1                ; STEP EXECUTION, SET T-BIT AFTER T-BIT TRAP
                          00000002 0001                          M_STEP=1 @ V_STEP
                                  0001  DS$AB_BPTINST::
00'00'00'00'00'00'00'00'00'00'00'00' 0001          .BYTE   0[BPTMAX+1]             ; SAVED INSTRUCTION TABLE
                        00'00'00'00' 000D
                                  0011          .ALIGN  LONG                                                    ;G:7
                                  0014  DS$AA_BPTADDR::
00000000'00000000'00000000'00000000' 0014          .LONG   0[BPTMAX+1]             ; BREAKPOINT LOCATION TABLE
00000000'00000000'00000000'00000000' 0024
00000000'00000000'00000000'00000000' 0034
00000000'00000000'00000000'00000000' 0044
                                  0054  DS$AA_BPTBASE::                                                         ;G:6
00000000'00000000'00000000'00000000' 0054          .LONG   0[BPTMAX+1]             ; BREAKPOINT BASE LOCATION TABLE      ;G:6
00000000'00000000'00000000'00000000' 0064
00000000'00000000'00000000'00000000' 0074
00000000'00000000'00000000'00000000' 0084
                                  0094  DS$AA_OFFSET:                                                           ;G:6
                          00000000 0094          .LONG   0                       ; BREAKPOINT OFFSET LOCAION           ;G:6
                                  0098  DFLTSIZE:
                          00000004 0098          .LONG   INITSIZE                ; EXAMINE/DEPOSIT SIZE QUALIFIER
                                  009C
                                  009C  BASEADDRESS:
                          00000000 009C          .LONG   INITBASE                ; ADDRESS BIAS FOR EXAMINE/DEPOSIT
                          000000B0 00A0  SAVE_SCB_VEC:   .BLKL   4               ; SPACE FOR SAVED SCB VECTOR ADRESSES
                          000000B8 00B0  NEW_AP_FP:      .BLKQ   1               ; SPACE FOR AP/FP
                          000000BC 00B8  NEW_SP:         .BLKL   1               ; OPERATOR MODIFIED 'SP'
                          000000C4 00BC  NEW_PC_PSL:     .BLKQ   1               ; OPERATOR MODIFIED 'PC/PSL'
                                  00C4
                          00000000 00C4  ID_TEMP:        .LONG   0               ; Storage for cpu id [23]
                                  00C8                                                                          ;G:6
                 000000D0'00000003' 00C8  FMTDIRECTIVE:   .LONG   20$-10$,10$     ; QUADWORD STRING DESCRIPTOR
                          3F 3F 21 00D0  10$:    .ASCII  .!??.                   ; FORMAT DIRECTIVE STRING
                                  00D3  20$:
                                  00D3
                                  00D3  EXAMQWDBUF:
                 000000DB'00000020 00D3          .LONG   EXAMSIZE,EXAMBUFFER     ; QUADWORD BUFFER DESCRIPTOR
                                  00DB  EXAMBUFFER:
                          000000FB 00DB          .BLKB   EXAMSIZE                ; DISPLAY BUFFER FOR EXAMINE
                                  00FB  PSLBUFFER:
                          0000014B 00FB          .BLKB   PSLBUFSZ                ; BUFFER FOR PSL MNEMONICS
                                  014B
                                  014B  FAOLST:
                                  014B          $FAOL   FMTDIRECTIVE,         - ; CONTROL STRING POINTER
```

```
                                    014B   291                EXAMQWDBUF+2,        - ; RETURNED STRING LENGTH (WORD)
                                    014B   292                EXAMQWDBUF,          - ; RETURN STRING BUFFER DESCRIPTOR
                                    014B   293                FAOARG                 ; ARGLIST POINTER (UNCOUNTED)
                           00000004'014B                      .ADDRESS       4
                           000000C8'014F                      .ADDRESS       FMTDIRECTIVE
                           000000D5'0153                      .ADDRESS       EXAMQWDBUF+2
                           000000D3'0157                      .ADDRESS       EXAMQWDBUF
                           0000015F'015B                      .ADDRESS       FAOARG
                                    015F   294                                                                        ;G:6
                                    015F   295 FAOARG:
                           00000163 015F   296        .BLKL   1                       ; FAO OPTIONAL ARGS LIST
                                    0163   297
                                    0163   298
                                    0163   299 ; The following storage allocation is for use by the routine which      [14]
                                    0163   300 ; loads the debugger.                                                  [14]
                                    0163   301
62 65 64 20 67 6E 69 64 61 6F 4C 00' 0163   302 FMT_DBG_MSG:    .ASCIC "Loading debugger...!/"          ;              [14]
      2F 21 2E 2E 2E 72 65 67 67 75  016F
                                 15  0163
6F 6C 20 72 65 67 67 75 62 65 44 00' 0179   303 FMT_DBG_LOAD:   .ASCIC "Debugger loaded at !XL!/"        ;              [14]
21 4C 58 21 20 74 61 20 64 65 64 61  0185
                                 2F  0191
                                 18  0179
4F 4C 20 54 27 4E 41 43 20 3F 3F 00' 0192   304 FMT_NO_ROOM:    .ASCIC "?? CAN'T LOAD DEBUGGER - NOT ENOUGH FREE MEMORY.!/";[14]
20 52 45 47 47 55 42 45 44 20 44 41  019E
48 47 55 4F 4E 45 20 54 4F 4E 20 2D  01AA
59 52 4F 4D 45 4D 20 45 45 52 46 20  01B6
                        2F 21 2E     01C2
                                 32  0192
                  00000000 00000000 01C5   305 DBG_ADDR_REQ:   .QUAD 0                 ;ARRAY TO REQUEST ADDR.        [14]
                                    01CD   306                                        ; RANGE                        [14]
                  00000000 00000000 01CD   307 DBG_ADDR_RTN:: .QUAD 0                 ;ARRAY TO RECEIVED ACTUAL ARRAY [14][26] ;G:
                                    01D5   308                                        ; ALLOCATED                    [14]
                           00000000 01D5   309 DEBUG_LO::      .LONG 0                ;LOWEST ADDR. ALLOCATED TO DEBUGGER[14]
                           00000000 01D9   310 DEBUG_HI::      .LONG 0                ;HIGHEST ADDR. ALLOCATED TO DEBUGGER[14]
                           00000000 01DD   311 SYMTAB_HI::     .LONG 0                ;TOP OF SYMBOL TABLE AREA      [14]
                                    01E1   312
                                 00 01E1   313 MP$GB_FUNCTION   ::     .BYTE   0     ; DEPOSIT OR EXAMINE           [18]
                           00000000 01E2   314 MP$GL_IPR_STATUS ::     .LONG   0     ; PP'S IPR ROUTINE STATUS      [18]
                           00000000 01E6   315 MP$GL_IPR_VALUE  ::     .LONG   0     ; IPR VALUE                    [18]
                           00000000 01EA   316 MP$GL_REG_ADDRESS::     .LONG   0     ; GPRS REG ADDRESS             [18]
                           00000000 01EE   317 MP$GL_REG_VALUE  ::     .LONG   0     ; IPRS REG VALUE               [18]
                           00000000 01F2   318 MP$GL_RET_STATUS ::     .LONG   0     ; AP'S RETURN STATUS           [18]
                           00000000 01F6   319 MP$GL_SAVED_R1   ::     .LONG   0     ; PP'S VALUE OF R1             [18]
                                    01FA   320
                           00000000 01FA   321 RET_REC :       .LONG   0             ; RECORD ATTRIBUTES FROM DS$LOAD [22]
                           00000000 01FE   322 RET_LEN :       .LONG   0             ; RECORD LENGTH FROM DS$LOAD     [22]
                                    0202   323
                                    0202   324
```

```
                                00000000   326          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                    0000   327
                                    0000   328          MODNAM  DEBUG                   ; SUPERVISOR MODULE IDENTIFIER
                47 55 42 45 44 00' 0000       $MODULE:        .ASCIC  \DEBUG\
                               05  0000
                                03  0006   329 BPT_LITERAL:    .BYTE   BPTCODE         ; A BPT INSTRUCTION
                                    0007   330
                                    0007   331 FMT_BASE_OUT:                                                    ;       [13]
74 6E 65 72 72 75 63 20 65 68 54 00' 0007   332          .ASCIC  "The current base address is !XL(X).!/"        ;       [13]
73 65 72 64 64 61 20 65 73 61 62 20  0013
2E 29 58 28 4C 58 21 20 73 69 20 73  001F
                           2F 21     002B
                              25     0007
                                    002D   333 FMT_DFLT_OUT:                                                    ;       [13]
64 61 72 20 74 6E 65 72 72 75 43 00' 002D   334          .ASCIC  "Current radix: !AC.!/Current size:  !AC.!/"   ;       [13]
75 43 2F 21 2E 43 41 21 20 3A 78 69  0039
20 3A 65 7A 69 73 20 74 6E 65 72 72  0045
                  2F 21 2E 43 41 21  0051
                              2A     002D
                                    0058   335 FMT_HEX:                                                       ;       [13]
6C 61 6D 69 63 65 64 61 78 65 48 00' 0058   336          .ASCIC  "Hexadecimal"                                 ;       [13]
                               0B   0058
                                    0064   337 FMT_DEC:                                                       ;       [13]
            6C 61 6D 69 63 65 44 00' 0064   338          .ASCIC  "Decimal"                                     ;       [13]
                               07   0064
                                    006C   339 FMT_OCT:                                                       ;       [13]
                6C 61 74 63 4F 00' 006C   340          .ASCIC  "Octal"                                       ;       [13]
                               05   006C
                                    0072   341 FMT_LONG:                                                      ;       [13]
         64 72 6F 77 67 6E 6F 4C 00' 0072   342          .ASCIC  "Longword"                                    ;       [13]
                               08   0072
                                    007B   343 FMT_WORD:                                                      ;       [13]
                   64 72 6F 57 00' 007B   344          .ASCIC  "Word"                                        ;       [13]
                               04   007B
                                    0080   345 FMT_BYTE:                                                      ;       [13]
                   65 74 79 42 00' 0080   346          .ASCIC  "Byte"                                        ;       [13]
                               04   0080
                                    0085   347 FMTBPT::                                                      ;       [19]
20 74 6E 69 6F 70 6B 61 65 72 42 00' 0085   348          .ASCIC  .Breakpoint at:!_!XL(X)!_(Base=!XL(X) Offset=!XL(X))!/.        ;G:6
21 29 58 28 4C 58 21 5F 21 3A 74 61  0091
58 28 4C 58 21 3D 65 73 61 42 28 5F  009D
4C 58 21 3D 74 65 73 66 66 4F 20 29  00A9
                  2F 21 29 29 58 28  00B5
                              35     0085
                                    00BB   349 FMTSTEP:                                                      ;G:7
   2F 21 4C 58 21 20 3A 4C 58 21 00' 00BB   350          .ASCIC  "!XL: !XL!/"
                               0A   00BB
                                    00C6   351 FMTBPTADR:
21 29 58 28 4C 58 21 5F 21 5F 21 00' 00C6   352          .ASCIC  .!_!_!XL(X)!_(Base=!XL(X) Offset=!XL(X))!/.                    ;G:6
58 28 4C 58 21 3D 65 73 61 42 28 5F  00D2
4C 58 21 3D 74 65 73 66 66 4F 20 29  00DE
                  2F 21 29 29 58 28  00EA
                              29     00C6
                                    00F0   353 FMTBPTLST:
73 74 6E 69 6F 70 6B 61 65 72 42 00' 00F0   354          .ASCIC  .Breakpoints located at:!/.                    ;G:7
3A 74 61 20 64 65 74 61 63 6F 6C 20  00FC
                              2F 21  0108
```

```
                                        19  00F0
                                            010A    355 FMTBPTNONE:
6E 65 72 72 75 63 20 65 6E 6F 4E 00'        010A    356         .ASCIC  .None currently set!/.                          ;       [13]
         2F 21 74 65 73 20 79 6C 74         0116
                                        14  010A
                                            011F    357 FMTBPTNOTSET:
20 74 6E 69 6F 70 6B 61 65 72 42 00'        011F    358         .ASCIC  .Breakpoint was not set!!!/.                    ;       [13]
21 74 65 73 20 74 6F 6E 20 73 61 77         012B
                              2F 21 21      0137
                                        1A  011F
                                            013A    359 FMTINVLDREG:
20 64 69 6C 61 76 6E 49 20 3F 3F 00'        013A    360         .ASCIC  .?? Invalid register mnemonic!/.                ;       [13]
65 6E 6D 20 72 65 74 73 69 67 65 72         0146
                  2F 21 63 69 6E 6F 6D      0152
                                        1E  013A
                                            0159    361 PRNT_AP_NAME::
               20 2F 20 43 41 21 20 00'     0159    362         .ASCIC  " !AC / "                                       ;       [18]
                                        07  0159
                                            0161    363 FMTEXAM_ASCII:
21 22 46 41 21 22 00000169'010E0000'        0161    364         .ASCID  ."!AF"!_!#XL.
                        4C 58 23 21 5F      016F
                                            0174    365 FMTEXAM:
43 41 21 5F 21 53 41 21 20 20 20 00'        0174    366         .ASCIC  "   !AS!_!AC!/"
                                 2F 21      0180
                                        0D  0174
                                            0182    367 FMTNOBPTSLEFT:
65 72 62 20 4C 55 21 20 6C 6C 41 00'        0182    368         .ASCIC  .All !UL breakpoints currently in use!!!/.      ;       [13]
72 75 63 20 73 74 6E 69 6F 70 6B 61         018E
73 75 20 6E 69 20 79 6C 74 6E 65 72         019A
                        2F 21 21 21 65      01A6
                                        28  0182
```

```
6F 70 73 65 72 20 6F 4E 20 3F 3F 00' 01AB   370 T_MACHCK:       .ASCIC  '?? No response/invalid operation'          ;       [13]
20 64 69 6C 61 76 6E 69 2F 65 73 6E  01B7
         6E 6F 69 74 61 72 65 70 6F  01C3
                                 20  01AB
76 20 73 73 65 63 63 41 20 3F 3F 00' 01CC   371 T_ACCVIO:       .ASCIC  '?? Access violation'                       ;       [13]
            6E 6F 69 74 61 6C 6F 69  01D8
                                 13  01CC
74 61 6C 73 6E 61 72 54 20 3F 3F 00' 01E0   372 T_TRANSL:       .ASCIC  '?? Translation not valid'                  ;       [13]
69 6C 61 76 20 74 6F 6E 20 6E 6F 69  01EC
                                 64  01F8
                                 18  01E0
20 67 6E 69 72 75 64 20 43 41 21 00' 01F9   373 FMT_EXCEPTION:  .ASCIC  "!AC during reference to X'!XL'!/"          ;       [13]
6F 74 20 65 63 6E 65 72 65 66 65 72  0205
         2F 21 27 4C 58 21 27 58 20  0211
                                 20  01F9
      6D 6F 72 66 20 64 61 65 72 00' 021A   374 T_PREGREAD:     .ASCIC  "read from"                                 ;       [13]
                                 09  021A
         6F 74 20 65 74 69 72 77 00' 0224   375 T_PREGWRITE:    .ASCIC  "write to"                                  ;       [13]
                                 08  0224
41 21 20 74 27 6E 61 43 20 3F 3F 00' 022D   376 FMT_BADPREG:    .ASCIC  "?? Can't !AC P!XB  !/"                     ;       [13]
      2F 21 20 20 42 58 21 50 20 43  0239
                                 15  022D
                                     0243   377 APSLMNE::
44 50 46 2C 2C 2C 50 54 2C 4D 43 00' 0243   378          .ASCIC  \CM,TP,,,FPD,IS,CUR=2a,PRV=2a,,IPL=5^X,\ -
50 2C 40 32 3D 52 55 43 2C 53 49 2C  024F
35 3D 4C 50 49 2C 2C 40 32 3D 56 52  025B
                           2C 58 5E  0267
46 2C 56 44 2C 2C 2C 2C 2C 2C 2C 2C  026A   379                 \,,,,,,,,DV,FU,IV,T,N,Z,V,C\
56 2C 5A 2C 4E 2C 54 2C 56 49 2C 55  0276
                              43 2C  0282
                                 40  0243
                                     0284   380
                           00000004  0284   381 MODELST::  .LONG        4
                           00000298' 0288   382          .ADDRESS        AKM
                           0000029F' 028C   383          .ADDRESS        AEM
                           000002A9' 0290   384          .ADDRESS        ASM
                           000002B4' 0294   385          .ADDRESS        AUM
                                     0298   386
               4C 45 4E 52 45 4B 00' 0298   387 AKM:     .ASCIC  .KERNEL.
                                 06  0298
      45 56 49 54 55 43 45 58 45 00' 029F   388 AEM:     .ASCIC  .EXECUTIVE.
                                 09  029F
   52 4F 53 49 56 52 45 50 55 53 00' 02A9   389 ASM:     .ASCIC  .SUPERVISOR.
                                 0A  02A9
                     52 45 53 55 00' 02B4   390 AUM:     .ASCIC  .USER.
                                 04  02B4
```

```
                    02B9    392 REGNAMLST:
000002FD'           02B9    393             .ADDRESS  AR0
00000300'           02BD    394             .ADDRESS  AR1
00000303'           02C1    395             .ADDRESS  AR2
00000306'           02C5    396             .ADDRESS  AR3                                   ;G:6
00000309'           02C9    397             .ADDRESS  AR4
0000030C'           02CD    398             .ADDRESS  AR5
0000030F'           02D1    399             .ADDRESS  AR6
00000312'           02D5    400             .ADDRESS  AR7
00000315'           02D9    401             .ADDRESS  AR8
00000318'           02DD    402             .ADDRESS  AR9
0000031B'           02E1    403             .ADDRESS  AR10
0000031F'           02E5    404             .ADDRESS  AR11
00000323'           02E9    405             .ADDRESS  AAP
00000326'           02ED    406             .ADDRESS  AFP
00000329'           02F1    407             .ADDRESS  ASP
0000032C'           02F5    408             .ADDRESS  APC
0000032F'           02F9    409             .ADDRESS  APSL
                    02FD    410
30 52 00'           02FD    411 AR0:        .ASCIC  .R0.
      02            02FD
31 52 00'           0300    412 AR1:        .ASCIC  .R1.
      02            0300
32 52 00'           0303    413 AR2:        .ASCIC  .R2.
      02            0303
33 52 00'           0306    414 AR3:        .ASCIC  .R3.
      02            0306
34 52 00'           0309    415 AR4:        .ASCIC  .R4.
      02            0309
35 52 00'           030C    416 AR5:        .ASCIC  .R5.
      02            030C
36 52 00'           030F    417 AR6:        .ASCIC  .R6.
      02            030F
37 52 00'           0312    418 AR7:        .ASCIC  .R7.
      02            0312
38 52 00'           0315    419 AR8:        .ASCIC  .R8.
      02            0315
39 52 00'           0318    420 AR9:        .ASCIC  .R9.
      02            0318
30 31 52 00'        031B    421 AR10:       .ASCIC  .R10.
         03         031B
31 31 52 00'        031F    422 AR11:       .ASCIC  .R11.
         03         031F
   50 41 00'        0323    423 AAP:        .ASCIC  .AP.
      02            0323
   50 46 00'        0326    424 AFP:        .ASCIC  .FP.
      02            0326
   50 53 00'        0329    425 ASP:        .ASCIC  .SP.
      02            0329
   43 50 00'        032C    426 APC:        .ASCIC  .PC.
      02            032C
4C 53 50 00'        032F    427 APSL:       .ASCIC  .PSL.
         03         032F
```

```
                     0333   429          .SBTTL  SET BASE ADDRESS SUBROUTINE
                  00000000  430          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000   431 ;++
                     0000   432 ; FUNCTIONAL DESCRIPTION:
                     0000   433 ;
                     0000   434 ;       SETS UP AN ADDRESS BIAS VALUE TO BE USED BY SUBSEQUENT
                     0000   435 ;       EXAMINE AND DEPOSIT OPERATOR COMMANDS
                     0000   436 ;
                     0000   437 ; CALLING SEQUENCE:
                     0000   438 ;
                     0000   439 ;       BSBW    VRSETBASE
                     0000   440 ;
                     0000   441 ; INPUT PARAMETERS:
                     0000   442 ;
                     0000   443 ;       R2      = BASE OF CLI DATA FRAME.
                     0000   444 ;       CLI$L_ADDRESS(R2)       ; OPERATOR SPECIFIED BIAS
                     0000   445 ;
                     0000   446 ; IMPLICIT INPUTS:
                     0000   447 ;
                     0000   448 ;       NONE
                     0000   449 ;
                     0000   450 ; OUTPUT PARAMETERS:
                     0000   451 ;
                     0000   452 ;       BASEADDRESS                     ; RECEIVES BIAS VALUE
                     0000   453 ;
                     0000   454 ; IMPLICIT OUTPUTS:
                     0000   455 ;
                     0000   456 ;       NONE
                     0000   457 ;
                     0000   458 ; COMPLETION CODES:
                     0000   459 ;
                     0000   460 ;       NONE
                     0000   461 ;
                     0000   462 ; SIDE EFFECTS:
                     0000   463 ;
                     0000   464 ;       NONE
                     0000   465 ;
                     0000   466 ;--
                     0000   467
                     0000   468 VRSETBASE::                             ;
0000009C'EF    18 A2 D0     0000   469          MOVL    CLI$L_ADDRESS(R2),L^BASEADDRESS ; STORE THE SPECIFIED BIAS      [12]
                   05     0008   470          RSB                             ; RETURN                                        :G:6
```

```
                     0009   472          .SBTTL  SHOW BASE ADDRESS SUBROUTINE                                ;       [13]
                     0009   473 ;++
                     0009   474 ; FUNCTIONAL DESCRIPTION:
                     0009   475 ;
                     0009   476 ;       Shows the current base address.
                     0009   477 ;
                     0009   478 ; CALLING SEQUENCE:
                     0009   479 ;
                     0009   480 ;       BSBW    VRSHOWBASE
                     0009   481 ;       JSB     VRSHOWBASE
                     0009   482 ;
                     0009   483 ; INPUT PARAMETERS:
                     0009   484 ;
                     0009   485 ;       NONE
                     0009   486 ;
                     0009   487 ; IMPLICIT INPUTS:
                     0009   488 ;
                     0009   489 ;       BASEADDRESS : The current base address
                     0009   490 ;
                     0009   491 ; OUTPUT PARAMETERS:
                     0009   492 ;
                     0009   493 ;       NONE
                     0009   494 ;
                     0009   495 ; IMPLICIT OUTPUTS:
                     0009   496 ;
                     0009   497 ;       NONE
                     0009   498 ;
                     0009   499 ; COMPLETION CODES:
                     0009   500 ;
                     0009   501 ;       NONE
                     0009   502 ;
                     0009   503 ; SIDE EFFECTS:
                     0009   504 ;
                     0009   505 ;       NONE
                     0009   506 ;
                     0009   507 ;--
                     0009   508
                     0009   509 VRSHOWBASE::                                                    ;               [13]
                     0009   510          $DS_PRINTF_S    L^FMT_BASE_OUT, -                      ; Print the base address        [13]
                     0009   511                            L^BASEADDRESS                       ;               [13]
     0000009C'EF   DD  0009                         PUSHL   L^BASEADDRESS
     00000007'EF   9F  000F                         PUSHAB  L^FMT_BASE_OUT
00000000'9F    02   FB  0015                         CALLS   #$$N, a#DS$PRINTF
                    05  001C   512          RSB                                                ; Return to caller              [13]
```

```
                        514          .SBTTL  SET DEFAULTS SUBROUTINE
                        515 ;++
                        516 ; FUNCTIONAL DESCRIPTION:
                        517 ;
                        518 ;       SETS UP DEFAULT PARAMETERS TO BE USED BY SUBSEQUENT
                        519 ;       EXAMINE AND DEPOSIT OPERATOR COMMANDS
                        520 ;
                        521 ; CALLING SEQUENCE:
                        522 ;
                        523 ;       BSBW    VRSETDFLT
                        524 ;
                        525 ; INPUT PARAMETERS:
                        526 ;
                        527 ;       R2      = BASE OF CLI DATA FRAME.
                        528 ;       CLI$L_FLAGS(R2) ; OPERATOR SPECIFIED DEFAULT(S)
                        529 ;
                        530 ; IMPLICIT INPUTS:
                        531 ;
                        532 ;       NONE
                        533 ;
                        534 ; OUTPUT PARAMETERS:
                        535 ;
                        536 ;       DS$GL_RADIX                     ; RECEIVES RADIX VALUE
                        537 ;       DFLTSIZE                        ; RECEIVES SIZE VALUE IN BYTES
                        538 ;
                        539 ; IMPLICIT OUTPUTS:
                        540 ;
                        541 ;       NONE
                        542 ;
                        543 ; COMPLETION CODES:
                        544 ;
                        545 ;       NONE
                        546 ;
                        547 ; SIDE EFFECTS:
                        548 ;
                        549 ;       NONE
                        550 ;
                        551 ;--
```

```
                         001D   553 VRSETDFLT::                             ;
                         001D   554
                         001D   555 ;+
                         001D   556 ; FIRST, CHECK RADIX FLAGS AND SET DEFAULT TO SPECIFIED VALUE
                         001D   557 ; (IF MORE THAN ONE SPECIFIED, USE MAXIMIZED VALUE)
                         001D   558 ;-
                         001D   559
      07 62    11   E1   001D   560          BBC     #CLI$V_OCT,CLI$L_FLAGS(R2),10$ ; OCTAL ?
00000000'EF    08   D0   0021   561          MOVL    #8,L^DS$GL_RADIX            ; YES, SET DEFAULT                [12]
      07 62    10   E1   0028   562 10$:     BBC     #CLI$V_DEC,CLI$L_FLAGS(R2),20$ ; DECIMAL ?                    [12]
00000000'EF    0A   D0   002C   563          MOVL    #10,L^DS$GL_RADIX           ; YES, SET DEFAULT                [12]
      07 62    12   E1   0033   564 20$:     BBC     #CLI$V_HEX,CLI$L_FLAGS(R2),30$ ; HEX ?                        [12]
00000000'EF    10   D0   0037   565          MOVL    #16,L^DS$GL_RADIX           ; YES, SET DEFAULT                [12]
                         003E   566 30$:
                         003E   567
                         003E   568 ;+
                         003E   569 ; HERE, CHECK SIZE FLAGS AND SET DEFAULT TO SPECIFIED NUMBER OF
                         003E   570 ; BITS  (IF MORE THAN ONE SPECIFIED, USE MAXIMIZED VALUE)
                         003E   571 ;-
                         003E   572
      07 62    0D   E1   003E   573          BBC     #CLI$V_BYTE,CLI$L_FLAGS(R2),40$ ; BYTE ?
00000098'EF    01   D0   0042   574          MOVL    #1,L^DFLTSIZE               ; YES, SET SIZE IN BYTES          [12]
      07 62    0E   E1   0049   575 40$:     BBC     #CLI$V_WORD,CLI$L_FLAGS(R2),50$ ; WORD ?                      [12]
00000098'EF    02   D0   004D   576          MOVL    #2,L^DFLTSIZE               ; YES, SET SIZE IN BYTES          [12]
      07 62    0F   E1   0054   577 50$:     BBC     #CLI$V_LONG,CLI$L_FLAGS(R2),60$ ; LONGWORD ?                  [12]
00000098'EF    04   D0   0058   578          MOVL    #4,L^DFLTSIZE               ; YES, SET SIZE IN BYTES          [12]
                         005F   579 60$:
                         005F   580
                         005F   581 ;+
                         005F   582 ; RETURN
                         005F   583 ;-
                         005F   584
                   05    005F   585          RSB                                 ; RETURN
```

```
                     0060    587          .SBTTL  SHOW DEFAULTS SUBROUTINE
                     0060    588 ;++
                     0060    589 ; FUNCTIONAL DESCRIPTION:
                     0060    590 ;
                     0060    591 ;       Show the current default size and radix.
                     0060    592 ;
                     0060    593 ; CALLING SEQUENCE:
                     0060    594 ;
                     0060    595 ;       BSBW    VRSHOWDFLT
                     0060    596 ;       JSB     VRSHOWDFLT
                     0060    597 ;
                     0060    598 ; INPUT PARAMETERS:
                     0060    599 ;
                     0060    600 ;       NONE
                     0060    601 ;
                     0060    602 ; IMPLICIT INPUTS:
                     0060    603 ;
                     0060    604 ;       NONE
                     0060    605 ;
                     0060    606 ; OUTPUT PARAMETERS:
                     0060    607 ;
                     0060    608 ;       NONE
                     0060    609 ;
                     0060    610 ; IMPLICIT OUTPUTS:
                     0060    611 ;
                     0060    612 ;       NONE
                     0060    613 ;
                     0060    614 ; COMPLETION CODES:
                     0060    615 ;
                     0060    616 ;       NONE
                     0060    617 ;
                     0060    618 ; SIDE EFFECTS:
                     0060    619 ;
                     0060    620 ;       NONE
                     0060    621 ;
                     0060    622 ;--
```

```
                                0060   VRSHOWDFLT::                                  ;                                                [13]
                                0060           ;                                                                                      [13]
                                0060           ; First, save R0, R1, and R2.         ;                                                [13]
                                0060           ;                                                                                      [13]
                     07     BB  0060           PUSHR   #^M<R0,R1,R2>                 ;                                                [13]
                                0062           ;                                                                                      [13]
                                0062           ; Put the FMT for the radix in R0,                                                     [13]
                                0062           ; the FMT for the size in R1.                                                          [13]
                                0062           ;                                                                                      [13]
     52   00000000'EF     D0  0062           MOVL    L^DS$GL_RADIX, R2             ; Store the radix in R2                          [13]
               52   10     D1  0069           CMPL    #16, R2                       ; Hexadecimal?                                   [13]
                    09     12  006C           BNEQ    10$                           ; If not, branch                                 [13]
     50   00000058'EF     9E  006E           MOVAB   L^FMT_HEX, R0                 ; Store format for hex                           [13]
                    15     11  0075           BRB     30$                           ;                                                [13]
               52   0A     D1  0077   10$:    CMPL    #10, R2                       ; Decimal ?                                      [13]
                    09     12  007A           BNEQ    20$                           ; If not, branch                                 [13]
     50   00000064'EF     9E  007C           MOVAB   L^FMT_DEC, R0                 ; Store format for decimal                       [13]
                    07     11  0083           BRB     30$                           ;                                                [13]
     50   0000006C'EF     9E  0085   20$:    MOVAB   L^FMT_OCT, R0                 ; Must be octal                                  [13]
                                008C   30$:                                         ;                                                [13]
     52   00000098'EF     D0  008C           MOVL    L^DFLTSIZE, R2                ; Store the size in R2                           [13]
               52   04     D1  0093           CMPL    #4, R2                        ; Longword?                                      [13]
                    09     12  0096           BNEQ    40$                           ; If not, branch                                 [13]
     51   00000072'EF     9E  0098           MOVAB   L^FMT_LONG, R1                ; Store format for long                          [13]
                    15     11  009F           BRB     60$                           ;                                                [13]
               52   02     D1  00A1   40$:    CMPL    #2, R2                        ; Word?                                          [13]
                    09     12  00A4           BNEQ    50$                           ; If not, branch                                 [13]
     51   0000007B'EF     9E  00A6           MOVAB   L^FMT_WORD, R1                ; Store format for word                          [13]
                    07     11  00AD           BRB     60$                           ;                                                [13]
     51   00000080'EF     9E  00AF   50$:    MOVAB   L^FMT_BYTE, R1                ; Store format for byte                          [13]
                                00B6   60$:                                         ;                                                [13]
                                00B6           ;                                                                                      [13]
                                00B6           ; Print the size and radix.                                                            [13]
                                00B6           ;                                                                                      [13]
                                00B6           $DS_PRINTF_S    L^FMT_DFLT_OUT, -         ; Print in this format                       [13]
                                00B6                             R0, R1               ; Size and radix                                 [13]
                    51     DD  00B6                   PUSHL   R1
                    50     DD  00B8                   PUSHL   R0
          0000002D'EF     9F  00BA                   PUSHAB  L^FMT_DFLT_OUT
     00000000'9F    03     FB  00C0                   CALLS   #$$N, @#DS$PRINTF
                    07     BA  00C7           POPR    #^M<R0,R1,R2>                 ; Restore registers                              [13]
                           05  00C9           RSB                                   ; Return to caller                               [13]
```

```
                        .SBTTL  EXAMINE MEMORY AND REGISTERS SUBROUTINE
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       LOCATES AND DISPLAYS SPECIFIED MEMORY OR GENERAL REGISTERS
                ;       TO THE OPERATOR IN THE SPECIFIED RADIX OR ASCII.
                ;       ALL SUPERVISOR OVERHEAD (I.E., STACK & REGISTER USAGE)
                ;       IS TRANSPARENT TO THE OPERATOR.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    VREXAMINE
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R2      = BASE OF CLI DATA FRAME.
                ;       CLI$L_FLAGS(R2)         ; RADIX/SIZE SPECIFIERS
                ;       CLI$L_ADDRESS(R2)       ; ADDRESS SPECIFIER
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       (AP) = BASE ADDRESS OF PROGRAM REGISTER CONTEXT
                ;       DS$GL_RADIX                     ; DEFAULT RADIX
                ;       DFLTSIZE                        ; DEFAULT SIZE IN BYTES
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;
                ;       NONE
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ;--
```

```
                        00CA   706 VREXAMINE::                                ;
                00F8 8F   BB    00CA   707          PUSHR   #^M<R3,R4,R5,R6,R7>     ; SAVE SOME REGISTERS
                   0A39   30    00CE   708          BSBW    RBREAK_OUT             ; REMOVE ALL BREAKPOINTS
                          00D1   709
                          00D1   710 10$:
             53   18 A2   D0    00D1   711          MOVL    CLI$L_ADDRESS(R2),R3   ; Get address for examine
                          00D5   712 ;+
                          00D5   713 ; First, determine where specified item is: if it is an internal CPU
                          00D5   714 ; register, special handling is required; but external registers are
                          00D5   715 ; available in memory.
                          00D5   716 ;-
                          00D5   717
          3F 62    14    E1    00D5   718          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),20$ ; Branch if not register
             10    53    D1    00D9   719          CMPL    R3,#16                 ; REG ADR EXCEED 'PSL' ?                ;G:5
                   10    1B    00DC   720          BLEQU   15$                    ; Out of range
                          00DE   721                                            ;
                          00DE   722          $DS_PRINTF_S  L^FMTINVLDREG    ; COMPLAIN ABOUT REGISTER NAME          [12]
                00000013A'EF    9F    00DE                PUSHAB  L^FMTINVLDREG
          00000000'9F    01    FB    00E4                CALLS   #$$N, a#DS$PRINTF
                      0173    31    00EB   723          BRW     VREXAMINEX             ; AND EXIT
                          00EE   724                                            ;
                00000000'EF    95    00EE   725 15$:     TSTB    MP$GB_SETCPU           ; See if we set cpu to an AP            [18]
                        1C    13    00F4   726          BEQL    17$                    ; Branch if not AP                      [18]
          000001E1'EF    01    90    00F6   727          MOVB    #1,MP$GB_FUNCTION      ; Set function to examine i.e. not deposit[1
                00000000'EF    16    00FD   728          JSB     MP$_GET_GPR            ; Get the value  of the register        [18]
                      03 50    E8    0103   729          BLBS    R0,16$                 ; Success?                              [18]
                          0106   730                                            ; Yes, then branch                      [18]
                          0106   731                                            ; Else, AP did not respond in time
                      0158    31    0106   732          BRW     VREXAMINEX             ; EXIT                                  [18]
       53    000001EE'EF    DE    0109   733 16$:     MOVAL   MP$GL_REG_VALUE ,R3    ; and put it in R3                      [18]
                        1E    11    0110   734          BRB     30$                    ; Continue
                          0112   735
                 53   6C43   DE    0112   736 17$:     MOVAL   (AP)[R3],R3            ; Locate saved register
                        18    11    0116   737          BRB     30$                    ; Continue
                          0118   738                                            ;
                14 62    1A    E0    0118   739 20$:     BBS     #CLI$V_PREG, -
                          011C   740                      CLI$L_FLAGS(R2),30$    ; Branch if not memory
         07 0444 C2    02    E1    011C   741          BBC     #CLI$V_BASE_QUAL, -                                     ;G:6
                          0122   742                      CLI$L_FLAGS2(R2), 25$  ;                                [20]
             53   0448 C2    C0    0122   743          ADDL2   CLI$L_TEMP_BASE(R2), R3 ;Add Temp Base Adr, or         [20]
                        07    11    0127   744          BRB     30$                    ;                               [20]  ;G:5
       53    0000009C'EF    C0    0129   745 25$:     ADDL2   L^BASEADDRESS, R3      ; Compute memory address        [12]
                          0130   746 30$:
                          0130   747
                          0130   748 ;+
                          0130   749 ; DETERMINE SIZE OF ITEM, USE DEFAULT IF NOT SPECIFIED.
                          0130   750 ;-
                   54    04    D0    0130   751          MOVL    #4,R4                  ; ASSUME LONGWORD
                1F 62    0F    E0    0133   752          BBS     #CLI$V_LONG,CLI$L_FLAGS(R2),78$ ; LONGWORD SPECIFIED ?  ;G:5
                06 62    14    E1    0137   753          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),40$  ; Skip if not register examine
                10   18 A2    D1    013B   754          CMPL    CLI$L_ADDRESS(R2),#PSLREG       ; Is the register the PSL?
                        15    13    013F   755          BEQL    78$                    ; If so, ignore data type switches ;
                          0141   756 40$:
                   54    02    D0    0141   757          MOVL    #2,R4                  ; ASSUME WORD
                0E 62    0E    E0    0144   758          BBS     #CLI$V_WORD,CLI$L_FLAGS(R2),78$ ; WORD SPECIFIED ?      ;G:5
                   54    01    D0    0148   759          MOVL    #1,R4                  ; ASSUME BYTE
                07 62    0D    E0    014B   760          BBS     #CLI$V_BYTE,CLI$L_FLAGS(R2),78$ ; BYTE SPECIFIED ?      ;G:5
```

```
54   00000098'EF   D0  014F   761          MOVL    L^DFLTSIZE,R4          ; ELSE USE DEFAULT SIZE                [12]
                       0156   762 78$:                                                                                ;G:5
```

```
                          0156   764 ;+
                          0156   765 ; EXTRACT THE TARGET ITEM FROM MEMORY
                          0156   766 ;-
                          0156   767
          03 62   1A   E1 0156   768          BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),79$ ; IF not IPR treat as memory  ;G:5
                 002B   31 015A   769          BRW     82$                             ; ELSE go and treat the IPR    [28]
    03 0000FE00'EF   1C   E0 015D   770 79$:     BBS     #DSA$V_USER,L^DSA$GL_FLAGS,80$ ; IF user mode THEN branch     [28]
                 0940   30 0165   771          BSBW    SAVE_SCB_VECTORS                ; SAVE PROGRAM'S SCB VECTORS   ;G:5
                        0168   772                                                  ; AND RE-INSERT SUPERVISORS    [28]
                   7E   D4 0168   773 80$:     CLRL    -(SP)                           ; SPACE FOR DATA               ;G:5
                        016A   774
                   6E   9F 016A   775          PUSHAB  (SP)                            ; WHERE TO STORE CURRENT CONTENTS
                   63   9F 016C   776          PUSHAB  (R3)                            ; AND WHERE TO FETCH IT FROM
                   54   DD 016E   777          PUSHL   R4                              ; NUMBER OF BYTES TO SHOW      ;G:5
       00000B83'EF   03   FB 0170   778          CALLS   #3,FETCH_STORE                  ; READ THE NECESSARY BYTES
                        0177   779                                                                                 ;G:5
    03 0000FE00'EF   1C   E0 0177   780          BBS     #DSA$V_USER,L^DSA$GL_FLAGS,81$ ; IF user mode THEN branch     [28]
                 08F0   30 017F   781          BSBW    RESTORE_SCB_VECTORS             ; RESTORE PROGRAM'S VECTORS    [28]
                   55 8ED0 0182   782 81$:     POPL    R5                              ; Get data into R5             ;G:5
                 0086   31 0185   783          BRW     89$                             ; Continue                     ;G:5
                   54   01   D0 0188   784 82$:     MOVL    #1,R4                           ; In case NEXT used, increment by 1   ;G:5
                   55   01   D0 018B   785          MOVL    #1,R5                           ; CMK$SGIPR code for EXAMINE function
    10 0000FE00'EF   1C   E1 018E   786          BBC     #DSA$V_USER,L^DSA$GL_FLAGS,83$  ; User mode?                 ;G:5
         00000000'EF   9F 0196   787          PUSHAB  L^T_EMESSG1                     ; User mode error message      [12]
       00000000'EF   01   FB 019C   788          CALLS   #1,DS$PRINTF                    ; Type it
                 00BB   31 01A3   789          BRW     VREXAMINEX                      ; Leave this place
                        01A6   790 83$:                                                                             ;G:5
         00000000'EF   95   01A6   791          TSTB    MP$GB_SETCPU                    ; See if we set cpu to an AP   [18]
                   35   13 01AC   792          BEQL    85$                             ; Branch if not AP             [18]
                   54   01   D0 01AE   793          MOVL    #1,R4                           ; In case NEXT used, increment by 1   [18]
       000001E1'EF   01   90 01B1   794          MOVB    #1,MP$GB_FUNCTION               ; CMK$SGIPR code for EXAMINE function [18]
       000001F6'EF   51   D0 01B8   795          MOVL    R1,MP$GL_SAVED_R1               ; STORE r1 for store/get value [18]
         00000000'EF   16   01BF   796          JSB     MP$_GET_IPR                     ; Get the value of the IP register [18]
    51   000001E6'EF   D0   01C5   797          MOVL    MP$GL_IPR_VALUE,R1              ; and put it in R1             [25]
                   55   51 D0 01CC   798          MOVL    R1,R5                           ; and also R5                  [25]
    50   000001F2'EF   D0   01CF   799          MOVL    MP$GL_RET_STATUS,R0             ; Also, store the returned status in r0 [18]
      03 000001E2'EF   E9   01D6   800          BLBC    MP$GL_IPR_STATUS,84$            ; Check status of GET_IPR routine [18]
                 002E   31 01DD   801          BRW     89$                                                         ;G:5
                 007E   31 01E0   802 84$:     BRW     VREXAMINEX                      ; Exit if ERROR routine        ;G:5
                        01E3   803                                                                                  ;G:5
    03 0000FE00'EF   1C   E0 01E3   804 85$:     BBS     #DSA$V_USER,L^DSA$GL_FLAGS,87$ ; IF user mode THEN branch     [28]
                 08BA   30 01EB   805          BSBW    SAVE_SCB_VECTORS                ; SAVE PROGRAM'S SCB VECTORS   [28]
                        01EE   806                                                                                  ;G:5
                        01EE   807 87$:     CMK     SGIPR                           ; Store/Get IPR value via CHMK trap ;G:5
                   7E   DC 01EE            MOVPSL  -(SP)
             06 6E   1A   E0 01F0            BBS     #PSL$V_IS,(SP),30000$
                   8E   D4 01F4            CLRL    (SP)+
                   0A   BC 01F6            CHMK    #CMK$_SGIPR
                   06   11 01F8            BRB     30001$
         00000D7A'EF   16   01FA      30000$: JSB     CMK$SGIPR
                        0200      30001$:
                   55   51 D0 0200   808          MOVL    R1,R5                           ; Get the value into R5
                        0203   809                                                                                  ;G:6
    03 0000FE00'EF   1C   E0 0203   810          BBS     #DSA$V_USER,L^DSA$GL_FLAGS,89$ ; IF user mode THEN branch     [28]
                 0864   30 020B   811          BSBW    RESTORE_SCB_VECTORS             ; RESTORE PROGRA'S VECTORS     [28]
                        020E   812                                                                                  ;G:5
                 1F 50   E8   020E   813 89$:     BLBS    R0,100$                         ; Type it, if no error         ;G:5
```

```
                          0211   814                                                                    ;G:6
                 18 62  1A E1  0211   815          BBC     #CLI$V_PREG,CLI$L_FLAGS(r2),90$ ; Exit if not IPR function
           50  0000021A'EF 9E  0215   816          MOVAB   L^T_PREGREAD,R0         ; Set up for IPR read access error      [12]
                          021C   817          $PRINTI_S       L^FMT_BADPREG,R0,R3 ; Type error                            [12]
                       53 DD  021C                          PUSHL   R3
                       50 DD  021E                          PUSHL   R0
                0000022D'EF 9F  0220                        PUSHAB  L^FMT_BADPREG
            00000000'EF 03 FB  0226                         CALLS   #$$N, DSX$PRINTI
                     0031 31  022D   818 90$:    BRW     VREXAMINEX              ; Exit routine
                     0036 30  0230   819 100$:   BSBW    VRTYPEXAMINE            ; Type out value                        ;G:5
                          0233   820
                          0233   821 ;+
                          0233   822 ; If NEXT is not zero, decrement it and loop back for next item, unless
                          0233   823 ; examining registers and PSL has just been displayed, or examining IPRs
                          0233   824 ; and P255 has just been displayed.
                          0233   825 ;-
                          0233   826
                    30 A2 D5  0233   827          TSTL    CLI$L_NEXT(R2)          ; All specified items displayed?
                       29 13  0236   828          BEQL    VREXAMINEX              ; Yes, exit
                 08 62  14 E1  0238   829          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),110$ ; Skip if not registers
                 10  18 A2 D1  023C   830          CMPL    CLI$L_ADDRESS(R2),#PSLREG ; Pointing at PSL?
                       1F 13  0240   831          BEQL    VREXAMINEX              ; Yes, time to exit
                       0E 11  0242   832          BRB     120$                    ; Increment register-style
                 0F 62  1A E1  0244   833 110$:   BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),130$ ; Skip if not IPRs
           000000FF 8F  18 A2 D1  0248   834          CMPL    CLI$L_ADDRESS(R2),#255  ; Pointing at top IPR?
                       0F 13  0250   835          BEQL    VREXAMINEX              ; Yes, time to exit
                    18 A2 D6  0252   836 120$:   INCL    CLI$L_ADDRESS(R2)       ; Bump to next register
                       04 11  0255   837          BRB     140$                    ; Skip to loop back
                 18 A2  54 C0  0257   838 130$:   ADDL2   R4,CLI$L_ADDRESS(R2)   ; Bump to next data item
                    30 A2 D7  025B   839 140$:   DECL    CLI$L_NEXT(R2)          ; Count this item
                     FE70 31  025E   840          BRW     10$                     ; Do it all again
                          0261   841
                          0261   842 ;+
                          0261   843 ; Return
                          0261   844 ;-
                          0261   845
                          0261   846 VREXAMINEX:
                       08D1 30  0261   847          BSBW    RBREAK_IN               ; Re-insert breakpoints
                  00F8 8F BA  0264   848          POPR    #^M<R3,R4,R5,R6,R7>     ; Restore registers
                       05 05  0268   849          RSB                             ; Return
                          0269   850
```

```
                                0269   852 .SBTTL  VRTYPEXAMINE    Type out location EXAMINE'd
                                0269   853
                                0269   854 ;+
                                0269   855 ; DETERMINE DISPLAY RADIX TO BE USED, DEFAULT IF NOT SPECIFIED.
                                0269   856 ;  (CONSTRUCT A CONTROL STRING FOR "FAO")
                                0269   857 ;-
                                0269   858 VRTYPEXAMINE:
                    00C0 8F     BB  0269   859          PUSHR   #^M<R6,R7>
                                026D   860
    000000D2'EF       4C 8F     90  026D   861          MOVB    #^A"L",L^FMTDIRECTIVE+10        ; SET UP FOR ADDRESS CONVERSION [12]
              2A 62     12      E0  0275   862          BBS     #CLI$V_HEX,CLI$L_FLAGS(R2),140$ ; BRANCH IF HEX SPECIFIED
              56   5A 8F     90  0279   863          MOVB    #^A'Z',R6                       ; SETUP FOR DECIMAL             ;G:5
              26 62     10      E0  027D   864          BBS     #CLI$V_DEC,CLI$L_FLAGS(R2),150$ ; BRANCH IF DECIMAL SPECIFIED
              56   4F 8F     90  0281   865          MOVB    #^A'O',R6                       ; SETUP OCTAL
              1E 62     11      E0  0285   866          BBS     #CLI$V_OCT,CLI$L_FLAGS(R2),150$ ; BRANCH IF OCTAL SPECIFIED
                                0289   867
              56   4F 8F     90  0289   868          MOVB    #^A'O',R6                       ; SETUP OCTAL
        08  00000000'EF       D1  028D   869          CMPL    L^DS$GL_RADIX,#8                ; OCTAL DEFAULT ?                [12]
                    11      13  0294   870          BEQL    150$                            ; YES, USE IT
              56   5A 8F     90  0296   871          MOVB    #^A'Z',R6                       ; SETUP DECIMAL
        0A  00000000'EF       D1  029A   872          CMPL    L^DS$GL_RADIX,#10               ; DECIMAL DEFAULT ?              [12]
                    04      13  02A1   873          BEQL    150$                            ; YES, USE IT
               56   58 8F     90  02A3   874 140$:    MOVB    #^A"X",R6                       ; OTHERWIZE USE HEXIDECIMAL
                                02A7   875
    000000D1'EF       56      90  02A7   876 150$:    MOVB    R6,L^FMTDIRECTIVE+9             ; INSERT DIRECTIVE BYTE          [12]
```

```
                                02AE    878 ;+
                                02AE    879 ; CONVERT ADDRESS (REGISTER OR MEMORY) OF ITEM TO ASCII
                                02AE    880 ; AND INSERT INTO OUTPUT BUFFER, FOLLOWED BY SYNTACTICAL SEPARATORS.
                                02AE    881 ;-
                                02AE    882
          000000D3'EF   20   D0 02AE    883         MOVL    #EXAMSIZE,L^EXAMQWDBUF          ; SET UP BUFFER CHAR. COUNT     [12]
 000000D7'EF   000000DB'EF   DE 02B5    884         MOVAL   L^EXAMBUFFER,L^EXAMQWDBUF+4     ; SET UP BUFFER POINTER         [12]
                                02C0    885
          0000015F'EF   63   DE 02C0    886         MOVAL   (R3),L^FAOARG                   ; SET ADDRESS TO CONVERT        [12]
                57   18 A2   D0 02C7    887         MOVL    CLI$L_ADDRESS(R2),R7            ; Get address for conversion    [12]
                17 62   14   E1 02CB    888         BBC     #CLI$V_REG,CLI$L_FLAGS(R2),170$ ; Branch if not external reg.  [12]
     000000D1'EF   4341 8F   B0 02CF    889         MOVW    #^A"AC",L^FMTDIRECTIVE+9        ; YES, SET TO COPY A STRING     [12]
0000015F'EF   000002B9'EF47   D0 02D8    890         MOVL    L^REGNAMLST[R7],L^FAOARG        ; STRING ADDRESS FOR FAO        [12]
                      27   11 02E4    891         BRB     200$                            ;                               [12]
                23 62   1A   E1 02E6    892 170$:   BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),200$ ; Branch if not internal reg.  [12]
          000000D7'FF   50 8F   90 02EA    893         MOVB    #^A"P",@L^EXAMQWDBUF+4          ; Move a "P" into output        [12]
                000000D7'EF   D6 02F2    894         INCL    L^EXAMQWDBUF+4                  ; Update buffer avail. address  [12]
                000000D3'EF   D7 02F8    895         DECL    L^EXAMQWDBUF                    ; Update buffer avail. byte count.[1
          000000D2'EF   42 8F   90 02FE    896         MOVB    #^A"B",L^FMTDIRECTIVE+10        ; Set for byte edit in radix    [12]
             0000015F'EF   57   D0 0306    897         MOVL    R7,L^FAOARG                     ; Point FAO to the register name.[12
                                030D    898
                                030D    899 200$:   $FAOL_G  L^FAOLST                       ; CONVERT ADDRESS TO ASCII      [12]
 00000000'GF   0000014B'EF   FA 030D                        CALLG   L^FAOLST,G^SYS$FAOL
 000000D3'EF   000000D5'EF   A2 0318    900         SUBW2   L^EXAMQWDBUF+2,L^EXAMQWDBUF     ; ADJUST BUFFER DESCRIPTOR      [12]
          57   000000D5'EF   3C 0323    901         MOVZWL  L^EXAMQWDBUF+2,R7               ;  (BY NUMBER INSERTED)         [12]
          000000D7'EF   57   C0 032A    902         ADDL2   R7,L^EXAMQWDBUF+4               ;  TO REMAINING BYTES           [12]
                                0331    903
000000D7'FF   093A 8F   B0 0331    904         MOVW    #^A':   ',@L^EXAMQWDBUF+4       ; STORE <:<TAB>> AFTER NUMBER   [12]
     000000D7'EF   02   C0 033A    905         ADDL2   #2,EXAMQWDBUF+4                 ; UPDATE AVAILABLE ADDRESS
     000000D3'EF   02   A2 0341    906         SUBW2   #2,EXAMQWDBUF                   ; SHORTEN AVAILABLE LENGTH
```

```
                                0348   908 ;+
                                0348   909 ; IF ITEM IS NUMERIC, CONVERT IT TO ASCII INTO OUTPUT BUFFER.
                                0348   910 ;-
                                0348   911
          000000FB'EF    94     0348   912          CLRB    L^PSLBUFFER                ; ERASE ANY PREVIOUS MNEMONICS          [12]
          00000000'EF    95     034E   913          TSTB    MP$GB_SETCPU               ; See if we set cpu to an AP            [18]
                  13     13     0354   914          BEQL    198$                       ; Branch if not AP                      [18]
          00000000'EF    9F     0356   915          PUSHAB  MP$GT_DEVICE_NAME          ; Device name of the Ap         [24]
          00000159'EF    9F     035C   916          PUSHAB  L^PRNT_AP_NAME             ; Control String                [24]
     00000000'9F    02     FB     0362   917          CALLS   #2, a#DS$PRINTF            ; Print the name of the device  [24]
           03 62    13     E1     0369   918 198$:    BBC     #CLI$V_ASCII,CLI$L_FLAGS(R2),199$ ; Skip branch, if ASCII      [12]
                 008C     31     036D   919          BRW     400$                       ; Branch if not ASCII                   [12]
                                0370   920 199$:                                       ;                                       [12]
           12 62    1A     E0     0370   921          BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),300$ ; Always longword, if IPR
           57    42 8F     90     0374   922          MOVB    #^A"B",R7                  ; SELECT BYTE OUTPUT
              01    54     D1     0378   923          CMPL    R4,#1                      ; SIZE = BYTE ?
                    0D     13     037B   924          BEQL    330$                       ; YES, USE IT
           57    57 8F     90     037D   925          MOVB    #^A"W",R7                  ; YES, SELECT DIRECTIVE
              02    54     D1     0381   926          CMPL    R4,#2                      ; SIZE = WORD ?
                    04     13     0384   927          BEQL    330$                       ; YES, USE IT
           57    4C 8F     90     0386   928 300$:    MOVB    #^A"L",R7                  ; Use longword form
     000000D1'EF    56     90     038A   929 330$:    MOVB    R6,L^FMTDIRECTIVE+9        ; RADIX TO DIRECTIVE STRING             [12]
     000000D2'EF    57     90     0391   930          MOVB    R7,L^FMTDIRECTIVE+10       ; SIZE TO DIRECTIVE STRING              [12]
     0000015F'EF    55     D0     0398   931          MOVL    R5,L^FAOARG                ; DATA ITEM FOR FAO                     [12]
                           039F   932          $FAOL_G  L^FAOLST                  ; CONVERT ITEM TO ASCII                 [12]
00000000'GF    0000014B'EF    FA     039F                       CALLG   L^FAOLST,G^SYS$FAOL
          57    000000D5'EF    3C     03AA   933          MOVZWL  L^EXAMQWDBUF+2,R7          ; GET COUNT OF BYTES INSERTED           [12]
     000000D3'EF    57     A2     03B1   934          SUBW2   R7,L^EXAMQWDBUF            ; ADJUST BUFFER DESCRIPTOR              [12]
     000000D7'EF    57     C0     03B8   935          ADDL2   R7,L^EXAMQWDBUF+4          ; UPDATE ADDRESS                        [12]
                           03BF   936
                           03BF   937 ;+
                           03BF   938 ; AND IF ITEM IS 'PSL', CONVERT ITS CONTENTS TO A MNEMONIC STRING.
                           03BF   939 ;-
                           03BF   940
           37 62    14     E1     03BF   941          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),350$ ; BRANCH IF NOT REGISTER
           10    18 A2     D1     03C3   942          CMPL    CLI$L_ADDRESS(R2),#PSLREG ; REGISTER = "PSL" ?
                    31     12     03C7   943          BNEQ    350$                       ; NO, SO SKIP OUT
                           03C9   944          $DS_CVTREG_S  #31,R5,L^APSLMNE,L^PSLBUFFER,#PSLBUFSZ, - ; CNVRT TO MNCS [12]
                           03C9   945                        MODELST,MODELST          ; INCLUDING ACCESS MODE MNEMONICS
                    00     DD     03C9                       PUSHL   #0
                    00     DD     03CB                       PUSHL   #0
                    00     DD     03CD                       PUSHL   #0
                    00     DD     03CF                       PUSHL   #0
          00000284'EF    DF     03D1                       PUSHAL  MODELST
          00000284'EF    DF     03D7                       PUSHAL  MODELST
          00000050 8F    DD     03DD                       PUSHL   #PSLBUFSZ
          000000FB'EF    9F     03E3                       PUSHAB  L^PSLBUFFER
          00000243'EF    9F     03E9                       PUSHAB  L^APSLMNE
                    55   DD     03EF                       PUSHL   R5
                    1F   DD     03F1                       PUSHL   #31
     00000000'9F    0B     FB     03F3                       CALLS   #11, a#DS$CVTREG
                    3F     11     03FA   946 350$:    BRB     410$                       ; SKIP TO DISPLAY
```

```
                                  03FC   948 ;+
                                  03FC   949 ; ASCII examine.  Print the data in !AF format (non-printing characters appear
                                  03FC   950 ; as ".") and follow by hex printout.
                                  03FC   951 ;-
                                  03FC   952
                                  03FC   953 400$:
          56    54    01    78    03FC   954          ASHL    #1,R4,R6                  ; Calculate # hex digits to type
                      55    DD    0400   955          PUSHL   R5                        ; Make data available in memory
                50    6E    9E    0402   956          MOVAB   (SP),R0                   ; For convenience, point to it
                51    7E    DE    0405   957          MOVAL   -(SP),R1                  ; Allocate a longword
                                  0408   958          $FAO_S  L^FMTEXAM_ASCII,(R1),-    ; Format for typeout                [12]
                                  0408   959                   L^EXAMQWDBUF,-           ;                                   [12]
                                  0408   960                   R4,R0,R6,R5
                      55    DD    0408                        PUSHL   R5
                      56    DD    040A                        PUSHL   R6
                      50    DD    040C                        PUSHL   R0
                      54    DD    040E                        PUSHL   R4
          000000D3'EF       7F    0410                        PUSHAQ  L^EXAMQWDBUF
                      61    3F    0416                        PUSHAW  (R1)
          00000161'EF       7F    0418                        PUSHAQ  L^FMTEXAM_ASCII
     00000000'GF      07    FB    041E                        CALLS   #$$T2,G^SYS$FAO
                      51  8ED0    0425   961          POPL    R1                        ; Get returned length
                51    51    3C    0428   962          MOVZWL  R1,R1                     ; Clear upper word
                      8E    D4    042B   963          CLRL    (SP)+                     ; De-allocate temporary
     000000D3'EF      51    A2    042D   964          SUBW2   R1,L^EXAMQWDBUF           ; Cut buffer size                   [12]
     000000D7'EF      51    A0    0434   965          ADDW2   R1,L^EXAMQWDBUF+4         ; Increase buffer address           [12]
                                  043B   966
                                  043B   967 ;+
                                  043B   968 ; THEN PRINT THE OUTPUT BUFFER(S).
                                  043B   969 ;-
                                  043B   970
000000D3'EF   20   000000D3'EF    A3    043B   971 410$:   SUBW3   L^EXAMQWDBUF,#EXAMSIZE,L^EXAMQWDBUF ; SET UP STRING LENGTH   [12]
     000000D7'EF   000000DB'EF    DE    0447   972          MOVAL   L^EXAMBUFFER,L^EXAMQWDBUF+4 ; SET UP BUFFER POINTER        [12]
                                  0452   973          $DS_PRINTF_S  L^FMTEXAM, -        ;                                   [12]
                                  0452   974                   #EXAMQWDBUF, -            ; Address and contents
                                  0452   975                   #PSLBUFFER                ; Register Mnemonics
          000000FB'8F       DD    0452                        PUSHL   #PSLBUFFER
          000000D3'8F       DD    0458                        PUSHL   #EXAMQWDBUF
          00000174'EF       9F    045E                        PUSHAB  L^FMTEXAM
     00000000'9F      03    FB    0464                        CALLS   #$$N, @#DS$PRINTF
                00C0 8F     BA    046B   976          POPR    #^M<R6,R7>                ; Restore registers
                            05    046F   977          RSB                               ; Return
```

```
                        0470   980          .SBTTL  DEPOSIT MEMORY AND REGISTERS SUBROUTINE
                        0470   981 ;++
                        0470   982 ; FUNCTIONAL DESCRIPTION:
                        0470   983 ;
                        0470   984 ;       LOCATES AND MODIFIES MEMORY OR GENERAL REGISTERS AS SPECIFIED
                        0470   985 ;       BY THE OPERATOR.
                        0470   986 ;       ALL SUPERVISOR OVERHEAD (I.E., STACK & REGISTER USAGE)
                        0470   987 ;       IS TRANSPARENT TO THE OPERATOR.
                        0470   988 ;
                        0470   989 ; CALLING SEQUENCE:
                        0470   990 ;
                        0470   991 ;       BSBW    VRDEPOSIT
                        0470   992 ;
                        0470   993 ; INPUT PARAMETERS:
                        0470   994 ;
                        0470   995 ;       R2      = BASE OF CLI DATA FRAME.
                        0470   996 ;       CLI$L_FLAGS(R2)         ; SIZE SPECIFIERS
                        0470   997 ;       CLI$L_ADDRESS(R2)       ; ADDRESS SPECIFIER
                        0470   998 ;       CLI$L_DATA(R2)          ; ITEM TO BE STORED
                        0470   999 ;
                        0470  1000 ; IMPLICIT INPUTS:
                        0470  1001 ;
                        0470  1002 ;       (AP) = BASE ADDRESS OF PROGRAM REGISTER CONTEXT
                        0470  1003 ;       DFLTSIZE                        ; DEFAULT SIZE IN BYTES
                        0470  1004 ;
                        0470  1005 ; OUTPUT PARAMETERS:
                        0470  1006 ;
                        0470  1007 ;       NONE
                        0470  1008 ;
                        0470  1009 ; IMPLICIT OUTPUTS:
                        0470  1010 ;
                        0470  1011 ;       (AP) = BASE ADDRESS OF PROGRAM REGISTER CONTEXT (MODIFIED)
                        0470  1012 ;
                        0470  1013 ; COMPLETION CODES:
                        0470  1014 ;
                        0470  1015 ;       NONE
                        0470  1016 ;
                        0470  1017 ; SIDE EFFECTS:
                        0470  1018 ;
                        0470  1019 ;       NONE
                        0470  1020 ;
                        0470  1021 ;--
```

```
                                0470  1023 VRDEPOSIT::                                  ;
                0118 8F    BB   0470  1024          PUSHR   #^M<R3,R4,R8>                ; SAVE SOME REGISTERS                  ;G:5
                                0474  1025
                                0474  1026 10$:
            53    18 A2    D0   0474  1027          MOVL    CLI$L_ADDRESS(R2),R3         ; Get address
                                0478  1028
                                0478  1029 ;+
                                0478  1030 ; FIRST, DETERMINE LOCATION OF ITEM SPECIFIED.  IF A REGISTER,
                                0478  1031 ; IT'S STORED SOMEWHERE IN MEMORY AT THIS POINT IN TIME.
                                0478  1032 ;-
                                0478  1033
            3E 62    14    E1   0478  1034          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),20$ ; Branch if not external reg
                10    53    D1   047C  1035          CMPL    R3,#16                       ; REG ADR EXCEED 'PSL' ?
                      10    1B   047F  1036          BLEQU   15$                                                                ;G:6
                                0481  1037
                                0481  1038          $DS_PRINTF_S  L^FMTINVLDREG          ; COMPLAIN ABOUT REGISTER NAME          [12]
          0000013A'EF      9F   0481                          PUSHAB  L^FMTINVLDREG
     00000000'9F    01     FB   0487                          CALLS   #$$N, a#DS$PRINTF
                  0175     31   048E  1039          BRW     VRDEPOSITX                   ; AND EXIT
                                0491  1040                                               ;
                                0491  1041
          00000000'EF      95   0491  1042 15$:     TSTB    MP$GB_SETCPU                 ; See if we set cpu to an AP            [18]
                    1B     13   0497  1043          BEQL    17$                          ; Branch if not AP                      [18]
          000001E1'EF      94   0499  1044          CLRB    MP$GB_FUNCTION               ; Set function to DEPOSIT               [18]
          00000000'EF      16   049F  1045          JSB     MP$_GET_GPR                  ; Get the address of the register       [18]
                 03 50     E8   04A5  1046          BLBS    R0,16$                       ; Success?                              [18]
                                04A8  1047                                               ; Yes, then branch                      [18]
                                04A8  1048                                               ; Else, AP did not respond in time
                  015B     31   04A8  1049          BRW     VRDEPOSITX                   ; AND EXIT                              [18]
     53   000001EA'EF      D0   04AB  1050 16$:     MOVL    MP$GL_REG_ADDRESS ,R3        ; and put it in R3                      [18]
                    1E     11   04B2  1051          BRB     30$                          ; Continue
                                04B4  1052
            53    6C43     DE   04B4  1053 17$:     MOVAL   (AP)[R3],R3                  ; Calculate real address
                    18     11   04B8  1054          BRB     30$                          ; Continue
                                04BA  1055
            14 62    1A    E0   04BA  1056 20$:     BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),30$ ; Branch if not memory
      07 0444 C2     02    E1   04BE  1057          BBC     #CLI$V_BASE_QUAL, CLI$L_FLAGS2(R2), 25$                         ;[20
         53   0448 C2      C0   04C4  1058          ADDL2   CLI$L_TEMP_BASE(R2), R3 ;Add Temp Base Adr, or                   [20]
                    07     11   04C9  1059          BRB     30$                          ;                                       [20]
     53   0000009C'EF      C0   04CB  1060 25$:     ADDL2   L^BASEADDRESS,R3             ; Compute memory address                [12]
                                04D2  1061 30$:
                                04D2  1062
                                04D2  1063 ;+
                                04D2  1064 ; DETERMINE SIZE OF ITEM, USE DEFAULT IF NOT SPECIFIED.
                                04D2  1065 ;-
                                04D2  1066
                54    04    D0   04D2  1067          MOVL    #4,R4                                ; YES, SET SIZE IN BYTES
            15 62    0F    E0   04D5  1068          BBS     #CLI$V_LONG,CLI$L_FLAGS(R2),79$ ; BRANCH IF LONGWORD SPECIFIED ;G:5
                54    02    D0   04D9  1069          MOVL    #2,R4                                ; YES, SET SIZE IN BYTES
            0E 62    0E    E0   04DC  1070          BBS     #CLI$V_WORD,CLI$L_FLAGS(R2),79$ ; BRANCH IF WORD SPECIFIED    ;G:5
                54    01    D0   04E0  1071          MOVL    #1,R4                                ; YES, SET SIZE IN BYTES
            07 62    0D    E0   04E3  1072          BBS     #CLI$V_BYTE,CLI$L_FLAGS(R2),79$ ; BRANCH IF BYTE SPECIFIED    ;G:5
     54   00000098'EF      D0   04E7  1073          MOVL    L^DFLTSIZE,R4                ; ELSE USE DEFAULT SIZE IN BYTES        [12]
                                04EE  1074 79$:                                                                                ;G:5
```

```
                                04EE  1076 ;+
                                04EE  1077 ; INSERT THE TARGET ITEM INTO MEMORY
                                04EE  1078 ;-
                                04EE  1079
                     0619   30  04EE  1080          BSBW    RBREAK_OUT                      ; REMOVE ALL BREAKPOINTS
                                04F1  1081
                  03 62   1A   E1  04F1  1082          BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),80$ ; IF NOT IPR treat as memory    [28]
                     0027   31  04F5  1083          BRW     90$                             ; ELSE treat as IPR             ;G:5
                                04F8  1084                                                                                  ;G:5
            03 0000FE00'EF   1C   E0  04F8  1085 80$:    BBS     #DSA$V_USER,L^DSA$GL_FLAGS,81$ ; IF user mode THEN branch    [28]
                     05A5   30  0500  1086          BSBW    SAVE_SCB_VECTORS                ; SAVE PROGRAM'S SCB VECTORS    ;G:5
                                0503  1087                                                  ; AND RE-INSERT SUPERVISORS      [28]
                                0503  1088                                                                                  ;G:5
                       63   9F  0503  1089 81$:    PUSHAB  (R3)                            ; ADDRESS TO STORE IT           ;G:5
                     1C A2  9F  0505  1090          PUSHAB  CLI$L_DATA(R2)                  ; ADDRESS OF DATA TO STORE
                       54   DD  0508  1091          PUSHL   R4                              ; NUMBER OF BYTES TO STORE
               00000B83'EF   03   FB  050A  1092          CALLS   #3,FETCH_STORE                  ; STORE THOSE BYTES THERE       ;G:5
            03 0000FE00'EF   1C   E0  0511  1093          BBS     #DSA$V_USER,L^DSA$GL_FLAGS,82$ ; IF user mode THEN branch    [28]
                     0556   30  0519  1094          BSBW    RESTORE_SCB_VECTORS             ; RESTORE PROGRAM'S VECTORS     [28]
                                051C  1095                                                                                  ;G:6
                                051C  1096 82$:                                                                             ;G:5
                     0085   31  051C  1097          BRW     100$                            ; Continue                      ;G:5
                                051F  1098 ;
                     54   01   D0  051F  1099 90$:    MOVL    #1,R4                           ; In case NEXT used, increment by 1
                       55   D4  0522  1100          CLRL    R5                              ; CMK$SGIPR code for DEPOSIT function
                  51   1C A2   D0  0524  1101          MOVL    CLI$L_DATA(R2),R1               ; Data value to store   [25]     ;G:2
            10 0000FE00'EF   1C   E1  0528  1102          BBC     #DSA$V_USER,L^DSA$GL_FLAGS,94$          ; User mode?
                00000000'EF   9F  0530  1103          PUSHAB  L^T_EMESSG1                     ; User mode error message       [12]  ;G:2
               00000000'EF   01   FB  0536  1104          CALLS   #1,DS$PRINTF                    ; Type it
                     00C6   31  053D  1105          BRW     VRDEPOSITX                      ; Leave this place
                                0540  1106 94$:
                00000000'EF   95  0540  1107          TSTB    MP$GB_SETCPU                    ; See if we set cpu to an AP             [18]
                       31   13  0546  1108          BEQL    96$                             ; Branch if not AP                       [18]
                     54   01   D0  0548  1109          MOVL    #1,R4                           ; In case NEXT used, increment by 1      [18]
                000001E1'EF   94  054B  1110          CLRB    MP$GB_FUNCTION                  ; CMK$SGIPR code for DEPOSIT function    [18]
            000001F6'EF   51   D0  0551  1111          MOVL    R1,MP$GL_SAVED_R1               ; STORE r1 = value to deposit            [18]
                00000000'EF   16  0558  1112          JSB     MP$_GET_IPR                     ; Get the Value of the ip register       [18]
         55     000001E6'EF   D0  055E  1113          MOVL    MP$GL_IPR_VALUE ,R5             ; and put it in R5                       [18]
         50     000001F2'EF   D0  0565  1114          MOVL    MP$GL_RET_STATUS,R0             ; Also, store the returned status in r0  [18]
           03 000001E2'EF   E9  056C  1115          BLBC    MP$GL_IPR_STATUS,95$            ; Check status of GET_IPR routine        [18]
                     002E   31  0573  1116          BRW     100$                                                                ;G:5
                     008D   31  0576  1117 95$:    BRW     VRDEPOSITX                      ; Exit if ERROR routine                  ;G:5
                                0579  1118
            03 0000FE00'EF   1C   E0  0579  1119 96$:    BBS     #DSA$V_USER,L^DSA$GL_FLAGS,98$ ; IF user mode THEN branch    [28]
                     0524   30  0581  1120          BSBW    SAVE_SCB_VECTORS                ; SAVE PROGRAM'S SCB VECTORS    ;G:5
                                0584  1121                                                  ; AND RE-INSERT SUPERVISORS      [28]
                                0584  1122 98$:    CMK     SGIPR                           ; Store/Get IPR value via CHMK trap ;G:5
                       7E   DC  0584          MOVPSL  -(SP)
                  06 6E   1A   E0  0586          BBS     #PSL$V_IS,(SP),30002$
                       8E   D4  058A          CLRL    (SP)+
                       0A   BC  058C          CHMK    #CMK$_SGIPR
                       06   11  058E          BRB     30003$
                00000D7A'EF   16  0590          JSB     CMK$SGIPR               30002$:
                                0596                                          30003$:
                     55   51   D0  0596  1123          MOVL    R1,R5                           ; Save read-back verify value
                                0599  1124                                                                                  ;G:5
            03 0000FE00'EF   1C   E0  0599  1125 99$:    BBS     #DSA$V_USER,L^DSA$GL_FLAGS,100$ ; IF user mode THEN branch    [28]
```

```
                     04CE   30  05A1  1126          BSBW    RESTORE_SCB_VECTORS     ; RESTORE PROGRAM'S VECTORS             [28]
                                05A4  1127                                                                                  ;G:5
                                05A4  1128 100$:
                                05A4  1129
                       01   BB  05A4  1130          PUSHR   #^M<R0>                 ; SAVE RETURN CODE
                     058C   30  05A6  1131          BSBW    RBREAK_IN               ; PUT BREAKPOINTS BACK
                       01   BA  05A9  1132          POPR    #^M<R0>                 ; RESTORE RETURN CODE
                                05AB  1133
                    2B 50   E8  05AB  1134          BLBS    R0,120$                 ; Continue if no error
                 54 62   1A  E1  05AE  1135          BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),VRDEPOSITX ; Exit if not IPR
                 09 50   01  E0  05B2  1136          BBS     #1,R0,110$              ; Was it the DEPOSIT that failed?
              50   00000224'EF  9E  05B6  1137          MOVAB   L^T_PREGWRITE,R0        ; Yes--set up correct error message     [12]
                       07   11  05BD  1138          BRB     115$                    ; Continue                              [12]
              50   0000021A'EF  9E  05BF  1139 110$:   MOVAB   L^T_PREGREAD,R0         ; Error was on read-back                [12]
                                05C6  1140 115$:    $PRINTI_S       L^FMT_BADPREG,R0,R3 ; Type error message                [12]
                       53   DD  05C6                        PUSHL   R3
                       50   DD  05C8                        PUSHL   R0
                 0000022D'EF  9F  05CA                        PUSHAB  L^FMT_BADPREG
              00000000'EF   03  FB  05D0                        CALLS   #$$N, DSX$PRINTI
                       2D   11  05D7  1141          BRB     VRDEPOSITX              ; Exit routine now.
                                05D9  1142
                                05D9  1143 ;+                                                                               ;G:8
                                05D9  1144 ; IF "NEXT" IS NOT ZERO, DECREMENT IT AND LOOP BACK FOR NEXT ITEM.
                                05D9  1145 ;  (EXCEPT WHEN DEPOSITING REGISTERS AND "PSL" HAS JUST BEEN MODIFIED)
                                05D9  1146 ;-
                                05D9  1147
                    30 A2   D5  05D9  1148 120$:   TSTL    CLI$L_NEXT(R2)          ; All specified items displayed?
                       1F   13  05DC  1149          BEQL    150$                    ; YES, EXIT
                                05DE  1150                                          ;
                 08 62   14  E1  05DE  1151          BBC     #CLI$V_REG,CLI$L_FLAGS(R2),125$ ; Skip if not registers
              10   18 A2   D1  05E2  1152          CMPL    CLI$L_ADDRESS(R2),#PSLREG ; POINTING AT "PSL" ?
                       15   13  05E6  1153          BEQL    150$                    ; YES, THEN IT'S TIME TO EXIT
                       04   11  05E8  1154          BRB     128$                    ; Increment
              05 62   1A  E1  05EA  1155 125$:   BBC     #CLI$V_PREG,CLI$L_FLAGS(R2),130$ ; Skip if not IPRs
                 18 A2   D6  05EE  1156 128$:   INCL    CLI$L_ADDRESS(R2)       ; Bump to next register
                       04   11  05F1  1157          BRB     140$                    ; AND SKIP TO LOOP BACK
                                05F3  1158                                          ;
              18 A2   54   C0  05F3  1159 130$:   ADDL2   R4,CLI$L_ADDRESS(R2)    ; BUMP TO NEXT DATA ITEM
                 30 A2   D7  05F7  1160 140$:   DECL    CLI$L_NEXT(R2)          ; DO ACCOUNTING
                    FE77   31  05FA  1161          BRW     10$                     ; GO DO WHOLE THING AGAIN
                                05FD  1162
              02 62   1A  E0  05FD  1163 150$:   BBS     #CLI$V_PREG,CLI$L_FLAGS(R2),160$ ; Branch if IPR
                       03   11  0601  1164          BRB     VRDEPOSITX              ; Then exit
                    FC63   30  0603  1165 160$:   BSBW    VRTYPEXAMINE            ; Only do display part of VREXAMINE
                                0606  1166
                                0606  1167 VRDEPOSITX:                            ;
                0118 8F   BA  0606  1168          POPR    #^M<R3,R4,R8>           ; RESTORE REGISTERS                     ;G:5
                       05   060A  1169          RSB                             ; RETURN
```

```
                        060B  1171          .SBTTL  SET BREAKPOINT SUBROUTINE
                        060B  1172 ;++
                        060B  1173 ; FUNCTIONAL DESCRIPTION:
                        060B  1174 ;
                        060B  1175 ;       SAVES AN 'OPCODE' FROM THE SPECIFIED ADDRESS AND REPLACES
                        060B  1176 ;       IT WITH A BREAKPOINT CODE                                    ;G:5
                        060B  1177 ;
                        060B  1178 ; CALLING SEQUENCE:
                        060B  1179 ;
                        060B  1180 ;       BSBW    VRSETBRK
                        060B  1181 ;
                        060B  1182 ; INPUT PARAMETERS:
                        060B  1183 ;
                        060B  1184 ;       CLI$L_ADDRESS(DS$GL_CLIBASE)
                        060B  1185 ;       CLI$L_FLAGS(DS$GL_CLIBASE)
                        060B  1186 ;
                        060B  1187 ; IMPLICIT INPUTS:      NONE
                        060B  1188 ;
                        060B  1189 ; OUTPUT PARAMETERS:    NONE
                        060B  1190 ;
                        060B  1191 ; IMPLICIT OUTPUTS:     NONE
                        060B  1192 ;
                        060B  1193 ; COMPLETION CODES:     NONE
                        060B  1194 ;
                        060B  1195 ; SIDE EFFECTS:         NONE
                        060B  1196 ;--
```

```
                        060B  1198 VRSETBRK::
                  1F    BB  060B  1199          PUSHR   #^M<R0,R1,R2,R3,R4>     ; SAVE WORKING REGISTERS                ;G:6
                  54    D4  060D  1200          CLRL    R4                      ;                                       ;G:6
                04F8    30  060F  1201          BSBW    RBREAK_OUT              ; REMOVE ALL BREAKPOINTS
      52  00000000'EF   DE  0612  1202          MOVAL   L^DS$GL_CLIBASE, R2     ; GET CLI ARGUMENT FRAME BASE.  [12]
       4E 62    17      E4  0619  1203          BBSC    #CLI$V_KERNEL, -        ; SET HIDDEN BRK IF KERNEL REQ
                        061D  1204                  CLI$L_FLAGS(R2), 10$
       50   18 A2       D0  061D  1205          MOVL    CLI$L_ADDRESS(R2), R0   ;GET ADDRESS GIVEN                      [20]
                        0621  1206
    0C 0444 C2    02    E1  0621  1207          BBC     #CLI$V_BASE_QUAL, -     ;                                       [20]
                        0627  1208                  CLI$L_FLAGS2(R2), 3$    ;                                       [20]
       50   0448 C2       C0  0627  1209          ADDL2   CLI$L_TEMP_BASE(R2), R0 ;Add Temp Base,                         [20]  ;G:6
       54   0448 C2       D0  062C  1210          MOVL    CLI$L_TEMP_BASE(R2), R4 ;Move Temp Base Address to R4                 ;G:6
                        0631  1211                                              ;to be store in a base address table  ;G:7
                  0E    11  0631  1212          BRB     6$                      ;                                       [20]
      50  0000009C'EF    C0  0633  1213 3$:      ADDL2   L^BASEADDRESS, R0       ;Add Def Base Adr.                      [12]
      54  0000009C'EF    D0  063A  1214          MOVL    L^BASEADDRESS, R4       ;Move Base Address to R4                ;G:6
                        0641  1215                                              ;to be store in a base address table  ;G:6
            53  50      D0  0641  1216 6$:      MOVL    R0,R3                   ;Save A Copy Of The Address             [20]
                        0644  1217
                014D    30  0644  1218          BSBW    FIND_BPT                ; SEE IF IT'S ALREADY THERE
              07 50     E9  0647  1219          BLBC    R0, 8$                  ; NOT FOUND, INSERT IN AN EMPTY         ;G:6
                        064A  1220                                              ; SLOT                                  ;G:6
                  54    D5  064A  1221          TSTL    R4                      ; SEE IF A NEW BASE IS GOING TO         ;G:6
                        064C  1222                                              ; INSERTED AT BREAK REFERENCE           ;G:6
                  13    12  064C  1223          BNEQ    9$                      ; MODIFY OLD BASE                       ;G:6
                0033    31  064E  1224          BRW     VRSETBRKX               ; NO CHANGE NEEDED, EXIT QUIETLY        ;G:6
                        0651  1225                                                                                      ;G:6
                  50    D4  0651  1226 8$:      CLRL    R0                      ; TRY FOR AN EMPTY SLOT                 ;G:6
                013E    30  0653  1227          BSBW    FIND_BPT                ; LOCATE EMPTY SLOT
              1C 50     E9  0656  1228          BLBC    R0, 20$                 ; BRANCH IF NO FREE SLOTS
00000014'EF41     53    D0  0659  1229          MOVL    R3, L^DS$AA_BPTADDR[R1] ; STORE ADDRESS OF BPT IN TABLE [12]
00000054'EF41     54    D0  0661  1230 9$:      MOVL    R4, L^DS$AA_BPTBASE[R1] ; STORE BASE ADDRESS CORRESPONDING      ;G:6
                        0669  1231                                              ; TO BIAS ADDRESS IN DS$AA_BPTADDR      ;G:6
                        0669  1232                                              ; TABLE                                 ;G:7
                  19    11  0669  1233          BRB     VRSETBRKX               ; EXIT
                        066B  1234
00000014'EF   18 A2     D0  066B  1235 10$:     MOVL    CLI$L_ADDRESS(R2), -
                        0673  1236                  L^DS$AA_BPTADDR             ; SET HIDDEN BREAKPOINT                 [12]
                  0F    11  0673  1237          BRB     VRSETBRKX               ; EXIT
                        0675  1238
                        0675  1239 20$:     $DS_PRINTF_S  L^FMTNOBPTSLEFT,#BPTMAX ; NO MORE BREAK POINTS            [12]
                  0F    DD  0675                        PUSHL   #BPTMAX
          00000182'EF   9F  0677                        PUSHAB  L^FMTNOBPTSLEFT
    00000000'9F   02    FB  067D                        CALLS   #$$N, a#DS$PRINTF
                        0684  1240
                        0684  1241 VRSETBRKX:
                04AE    30  0684  1242          BSBW    RBREAK_IN               ; PUT ALL BREAKPOINTS BACK IN
                  1F    BA  0687  1243          POPR    #^M<R0,R1,R2,R3,R4>     ; RESTORE REGISTERS                     ;G:6
                        05  0689  1244          RSB                             ; RETURN
```

```
                        068A  1246          .SBTTL  CLEAR BREAKPOINT SUBROUTINE
                        068A  1247 ;++
                        068A  1248 ; FUNCTIONAL DESCRIPTION:
                        068A  1249 ;
                        068A  1250 ;       RESTORES THE SAVED 'OPCODE' TO THE SPECIFIED ADDRESS,
                        068A  1251 ;       IF IT IS A KNOWN BREAKPOINT
                        068A  1252 ;
                        068A  1253 ; CALLING SEQUENCE:
                        068A  1254 ;
                        068A  1255 ;       BSBW    VRCLRBRK
                        068A  1256 ;
                        068A  1257 ; INPUT PARAMETERS:
                        068A  1258 ;
                        068A  1259 ;       R2      = BASE OF CLI DATA FRAME.
                        068A  1260 ;       CLI$L_ADDRESS(R2)
                        068A  1261 ; IMPLICIT INPUTS:
                        068A  1262 ;
                        068A  1263 ;       CLI$L_DATA(R2)          ; IF -1, INDICATES "ALL"
                        068A  1264 ;
                        068A  1265 ; OUTPUT PARAMETERS:    NONE
                        068A  1266 ;
                        068A  1267 ; IMPLICIT OUTPUTS:     NONE
                        068A  1268 ;
                        068A  1269 ; COMPLETION CODES:     NONE
                        068A  1270 ;
                        068A  1271 ; SIDE EFFECTS:         NONE
                        068A  1272 ;--
```

```
                                068A  VRCLRBRK::
                     07   BB    068A          PUSHR   #^M<R0,R1,R2>            ; SAVE WORKING REGISTERS
                   047B   30    068C          BSBW    RBREAK_OUT               ; REMOVE ALL BREAKPOINTS
        52   00000000'EF   DE    068F          MOVAL   L^DS$GL_CLIBASE, R2      ; BASE CLI DATA BASE                    [12]
                                0696
     FFFFFFFF 8F   1C A2   D1    0696          CMPL    CLI$L_DATA(R2),#-1       ; CHECK FOR 'ALL' ARGUMENT
                     3D   13    069E          BEQL    30$                      ; YES, GO CLEAR THEM ALL
                                06A0
        09 0444 C2   02   E1    06A0          BBC     #CLI$V_BASE_QUAL, CLI$L_FLAGS2(R2), 3$                  ;[20]
     50   18 A2   0448 C2   C1    06A6          ADDL3   CLI$L_TEMP_BASE(R2), -   ;Add Temp Base Adr, or          [20]
                                06AD                  CLI$L_ADDRESS(R2),R0     ;                               [20]
                     09   11    06AD          BRB     6$                       ;                               [20]
  50   18 A2   0000009C'EF   C1    06AF  3$:     ADDL3   L^BASEADDRESS, -         ;Add Def Base Adr.              [12]
                                06B8                  CLI$L_ADDRESS(R2),R0     ;
                                06B8
                   00D9   30    06B8  6$:     BSBW    FIND_BPT                 ; LOCATE IT
                  10 50   E9    06BB          BLBC    R0,10$                   ; NOT FOUND!
           00000014'EF41   D4    06BE          CLRL    L^DS$AA_BPTADDR[R1]      ; REMOVE BREAKPOINT                     [12]
           00000054'EF41   D4    06C5          CLRL    L^DS$AA_BPTBASE[R1]      ; REMOVE BASE ADDRESS CORRESPONDING   ;G:6
                                06CC                                           ; TO BIAS ADDRESS IN DS$AA_BPTADDR    ;G:6
                                06CC                                           ; TABLE                               ;G:7
                     23   11    06CC          BRB     VRCLRBRKX                ; AND EXIT
                                06CE  10$:
                                06CE          $DS_PRINTF_S  L^FMTBPTNOTSET     ; WAS NOT SET !                         [12]
           0000011F'EF   9F    06CE                  PUSHAB  L^FMTBPTNOTSET
        00000000'9F   01   FB    06D4                  CALLS   #$$N, @#DS$PRINTF
                     14   11    06DB          BRB     VRCLRBRKX                ; EXIT
                                06DD
                                06DD  30$:
                 51   0F   D0    06DD          MOVL    #BPTMAX,R1               ; MAX NUMBER OF BPT'S
           00000014'EF41   D4    06E0  40$:    CLRL    L^DS$AA_BPTADDR[R1]      ; FREE UP THIS BREAKPOINT               [12]
           00000054'EF41   D4    06E7          CLRL    L^DS$AA_BPTBASE[R1]      ; REMOVE BASE ADDRESS CORRESPONDING   ;G:6
                                06EE                                           ; TO BIAS ADDRESS IN DS$AA_BPTADDR    ;G:6
                                06EE                                           ; TABLE                               ;G:7
                  EF 51   F5    06EE          SOBGTR  R1,40$                   ; CHECK WHOLE TABLE (EXCEPT [0])
                                06F1
                                06F1  VRCLRBRKX:                               ;
                   0441   30    06F1          BSBW    RBREAK_IN                ; PUT BREAKPOINTS BACK IN
                     07   BA    06F4          POPR    #^M<R0,R1,R2>            ; RESTORE REGISTERS
                          05    06F6          RSB                              ; RETURN
```

```
                                06F7  1313          .SBTTL  DSP$REMOVEBREAK Remove breakpoints
                                06F7  1314 ;++
                                06F7  1315 ;
                                06F7  1316 ; FUNCTIONAL DESCRIPTION:
                                06F7  1317 ;
                                06F7  1318 ;       This routine can be called to remove breakpoints from
                                06F7  1319 ;       a section of memory.
                                06F7  1320 ;
                                06F7  1321 ; CALLING SEQUENCE:
                                06F7  1322 ;
                                06F7  1323 ;       CALLS/G arglist,DSP$REMOVEBREAK
                                06F7  1324 ;
                                06F7  1325 ; INPUT PARAMETERS:
                                06F7  1326 ;
                                06F7  1327 ;       0       #2
                                06F7  1328 ;       4       Lowest address of range
                                06F7  1329 ;       8(AP)   Highest address of range
                                06F7  1330 ;
                                06F7  1331 ; IMPLICIT INPUTS:
                                06F7  1332 ;
                                06F7  1333 ;       DSA$AA_BPTADDR[n]
                                06F7  1334 ;
                                06F7  1335 ; OUTPUT PARAMETERS:    NONE
                                06F7  1336 ;
                                06F7  1337 ; IMPLICIT OUTPUTS:
                                06F7  1338 ;
                                06F7  1339 ;       Some breakpoints are removed from DSA$AA_BPTADDR.
                                06F7  1340 ;
                                06F7  1341 ; COMPLETION CODES:     NONE
                                06F7  1342 ;
                                06F7  1343 ; SIDE EFFECTS:         NONE
                                06F7  1344 ;--
                                06F7  1345
                         001C   06F7  1346 .ENTRY  DSP$REMOVEBREAK,^M<R2,R3,R4>                          ;G:6
                                06F9  1347
                  040E   30     06F9  1348          BSBW    RBREAK_OUT                  ; Put opcodes back in memory
53  00000014'EF   9E            06FC  1349          MOVAB   L^DS$AA_BPTADDR,R3          ; Base BPT address table      [12]
54  00000054'EF   9E            0703  1350          MOVAB   L^DS$AA_BPTBASE,R4          ; Base Address corresponding  ;G:6
                                070A  1351                                              ; to BIAS ADDRESS in DS$AA_BPTADDR ;G:6
                                070A  1352                                              ; table                       ;G:7
          51  04 AC   7D        070A  1353          MOVQ    4(AP),R1                    ; Get low and high limits
              50  0F   D0        070E  1354          MOVL    #BPTMAX,R0                  ; Get length of breakpoint table
                                0711  1355
        6340  51       D1        0711  1356 10$:     CMPL    R1,(R3)[R0]                 ; Compare lower with this BPT addr
                0C     1A        0715  1357          BGTRU   20$                         ; Branch if lower GTR this BPT
        6340  52       D1        0717  1358          CMPL    R2,(R3)[R0]                 ; Compare higher with this BPT addr
                06     1B        071B  1359          BLEQU   20$                         ; Branch if higher LSS this BPT
              6340     D4        071D  1360          CLRL    (R3)[R0]                    ; Remove the BPT
              6440     D4        0720  1361          CLRL    (R4)[R0]                    ; Remove Base Address corresponding ;G:6
                                0723  1362                                              ; to BIAS ADDRESS in DS$AA_BPTADDR ;G:6
                                0723  1363                                              ; table                       ;G:7
            EB 50      F5        0723  1364 20$:     SOBGTR  R0,10$                      ; Count and continue
                                0726  1365
              040C     30        0726  1366          BSBW    RBREAK_IN                   ; Put remaining breakpoints back
                       04        0729  1367          RET                                 ; Return
```

```
                        072A  1369          .SBTTL  SHOW BREAKPOINTS SUBROUTINE
                        072A  1370 ;++
                        072A  1371 ; FUNCTIONAL DESCRIPTION:
                        072A  1372 ;                                                                    ;G:7
                        072A  1373 ;       DISPLAYS ALL CURRENT BREAKPOINTS TO THE OPERATOR
                        072A  1374 ;
                        072A  1375 ; CALLING SEQUENCE:
                        072A  1376 ;
                        072A  1377 ;       BSBW    VRSHOWBRK
                        072A  1378 ;
                        072A  1379 ; INPUT PARAMETERS:     NONE
                        072A  1380 ;
                        072A  1381 ; IMPLICIT INPUTS:
                        072A  1382 ;
                        072A  1383 ;       DS$AA_BPTADDR                    ; ARRAY OF KNOWN BREAKPOINTS
                        072A  1384 ;
                        072A  1385 ; OUTPUT PARAMETERS:    NONE
                        072A  1386 ;
                        072A  1387 ; IMPLICIT OUTPUTS:     NONE
                        072A  1388 ;
                        072A  1389 ; COMPLETION CODES:     NONE
                        072A  1390 ;
                        072A  1391 ; SIDE EFFECTS:         NONE
                        072A  1392 ;--
                        072A  1393
                        072A  1394 VRSHOWBRK::                           ;
                0C   BB 072A  1395          PUSHR   #^M<R2,R3>           ; NEED A TEMPORARY
                53   D4 072C  1396          CLRL    R3                   ; SO FAR NO BREAK POINTS SET
                        072E  1397
                52   0F D0 072E  1398          MOVL    #BPTMAX,R2           ; SET UP TABLE LENGTH
     00000014'EF42   D5 0731  1399 10$:       TSTL    L^DS$AA_BPTADDR[R2]  ; SCAN BREAKPOINT ADDRESSES          [12]
                44   13 0738  1400          BEQL    30$                  ; SKIP UNTIL FOUND A VALID BPT
                        073A  1401
              0D 53  00   E2 073A  1402          BBSS    #0,R3,20$            ; IF HEADER ALREADY OUT, SKIP
                        073E  1403          $DS_PRINTF_S  L^FMTBPTLST    ; DISPLAY A SMALL HEADER             [12]
      000000F0'EF   9F 073E                       PUSHAB  L^FMTBPTLST
   00000000'9F   01   FB 0744                       CALLS   #$$N, a#DS$PRINTF
                        074B  1404 20$:
00000014'EF42   00000054'EF42   C3 074B  1405          SUBL3   L^DS$AA_BPTBASE[R2],L^DS$AA_BPTADDR[R2],L^DS$AA_OFFSET    ;G:6
     00000094'EF                        0758
                        075D  1406                                       ;Calculate Offset to display    ;G:6
                        075D  1407          $DS_PRINTF_S  L^FMTBPTADR,L^DS$AA_BPTADDR[R2],L^DS$AA_BPTBASE[R2],-    ;G:6
                        075D  1408                        L^DS$AA_OFFSET                                ;G:6
      00000094'EF   DD 075D                       PUSHL   L^DS$AA_OFFSET
   00000054'EF42   DD 0763                       PUSHL   L^DS$AA_BPTBASE[R2]
   00000014'EF42   DD 076A                       PUSHL   L^DS$AA_BPTADDR[R2]
      000000C6'EF   9F 0771                       PUSHAB  L^FMTBPTADR
   00000000'9F   04   FB 0777                       CALLS   #$$N, a#DS$PRINTF
                        077E  1409                                       ; DISPLAY 'PC' OF BRKPNT        [12]  ;G:6
                        077E  1410 30$:
                B0 52   F5 077E  1411          SOBGTR  R2,10$               ; GO TRY SOME MORE (EXCEPT [0])
                0D 53   E8 0781  1412          BLBS    R3,VRSHOWBRKX        ; EXIT IF THERE WERE ANY BREAKPOINTS
                        0784  1413
                        0784  1414          $DS_PRINTF_S    L^FMTBPTNONE ; 'NONE CURRENTLY SET'                  [12]
      0000010A'EF   9F 0784                       PUSHAB  L^FMTBPTNONE
   00000000'9F   01   FB 078A                       CALLS   #$$N, a#DS$PRINTF
                        0791  1415
```

```
                0791  1416 VRSHOWBRKX:
      0C   BA   0791  1417          POPR    #^M<R2,R3>                ; RESTORE TEMPORARIES
           05   0793  1418          RSB                               ; RETURN
```

```
                     0794  1420          .SBTTL  FIND_BPT        FIND BPT IN BPT TABLE
                     0794  1421 ;++
                     0794  1422 ; FUNCTIONAL DESCRIPTION:
                     0794  1423 ;                                                                    ;G:7
                     0794  1424 ;       Locate a specific breakpoint in the breakpoint table.
                     0794  1425 ;
                     0794  1426 ; CALLING SEQUENCE:
                     0794  1427 ;
                     0794  1428 ;       BSBW    FIND_BPT
                     0794  1429 ;
                     0794  1430 ; INPUT PARAMETERS:
                     0794  1431 ;
                     0794  1432 ;       R0      Address of desired BPT or zero if empty slot wanted
                     0794  1433 ;
                     0794  1434 ; IMPLICIT INPUTS:      NONE
                     0794  1435 ;
                     0794  1436 ; OUTPUT PARAMETERS:
                     0794  1437 ;
                     0794  1438 ;       R1      Index of BPT entry  (DS$AA_BPTADDR[x] as a LONGWORD)
                     0794  1439 ;
                     0794  1440 ; IMPLICIT OUTPUTS:     NONE
                     0794  1441 ;
                     0794  1442 ; RETURN CODES:
                     0794  1443 ;
                     0794  1444 ;       R0      SS$_NORMAL IF FOUND ELSE ZERO
                     0794  1445 ;
                     0794  1446 ; SIDE EFFECTS:         NONE
                     0794  1447 ;--
                     0794  1448
                     0794  1449 FIND_BPT:
          51   0F   D0  0794  1450          MOVL    #BPTMAX,R1              ; START AT THE TOP
                     0797  1451 10$:
00000014'EF41   50   D1  0797  1452          CMPL    R0,L^DS$AA_BPTADDR[R1]  ; IS IT?                        [12]
              06   13  079F  1453          BEQL    20$                     ; YES, FOUND IT
            F3 51   F5  07A1  1454          SOBGTR  R1,10$                  ; CONTINUE UNTIL AFTER CHECKED [1]
              50   D4  07A4  1455          CLRL    R0                      ; SET NOT FOUND
                   05  07A6  1456          RSB                             ; RETURN NOT FOUND
          50   01   D0  07A7  1457 20$:    MOVL    #SS$_NORMAL,R0          ; SUCCESS
                   05  07AA  1458          RSB                             ; RETURN
```

```
                                07AB  1460          .SBTTL  DEBUGGER CONDITION HANDLER
                                07AB  1461 ;++
                                07AB  1462 ; FUNCTIONAL DESCRIPTION:
                                07AB  1463 ;
                                07AB  1464 ;       This routine is the secondary condition handler.
                                07AB  1465 ;       It is here to allow the supervisor to capture BPT and TBIT
                                07AB  1466 ;       traps. It discards all other conditions.
                                07AB  1467 ;
                                07AB  1468 ; CALLING SEQUENCE:
                                07AB  1469 ;
                                07AB  1470 ;       STANDARD PROCEEDURE CALL WITH STANDARD EXCEPTION ARGS
                                07AB  1471 ;
                                07AB  1472 ; INPUT PARAMETERS:
                                07AB  1473 ;
                                07AB  1474 ;       STANDARD EXCEPTION SIGNAL AND MECHANISM ARRAYS
                                07AB  1475 ;
                                07AB  1476 ;
                                07AB  1477 ; IMPLICIT INPUTS:      NONE
                                07AB  1478 ;
                                07AB  1479 ; OUTPUT PARAMETERS:    NONE
                                07AB  1480 ;
                                07AB  1481 ; IMPLICIT OUTPUTS:     NONE
                                07AB  1482 ;
                                07AB  1483 ; RETURN CODES:
                                07AB  1484 ;
                                07AB  1485 ;       SS$_CONTINUE    if one of our break points or T-bit.
                                07AB  1486 ;       SS$_RESIGNAL    otherwise
                                07AB  1487 ;
                                07AB  1488 ; SIDE EFFECTS:
                                07AB  1489 ;
                                07AB  1490 ;       UNKNOWN SINCE THE OPERATOR CAN CHANGE THINGS WHILE IN COMMAND MODE
                                07AB  1491 ;--
                          0000  07AB  1492 .ENTRY  COND_DEBUG,^M<>
                                07AD  1493
            50    0000'8F   3C  07AD  1494          MOVZWL  #SS$_RESIGNAL,R0          ; ASSUME IT'S SOMETHING ELSE
              51    04 BC     DE  07B2  1495          MOVAL   @4(AP), R1                ; GET SIGNAL ARRAY POINTER
04 A1    00000000'8F    D1  07B6  1496          CMPL    #SS$_BREAK, 4(R1)         ; IS THIS A BREAKPOINT CONDITION?
                   11    12  07BE  1497          BNEQ    20$                       ; NO, TRY T-BIT
                         07C0  1498          $MP_SET_INTERLOCK #MP$V_IL_CNDB ; Interlock routine     [17][21][23]
                   07    DD  07C0                      PUSHL   #MP$V_IL_CNDB
                   08    DD  07C2                      PUSHL   #$SRVCODE_SET_INTERLOCK
   00000000'9F     02    FB  07C4                      CALLS   #2, @#DS$SERVICE_DISPATCHER
                 0051    30  07CB  1499          BSBW    DBG_BREAK                 ; HANDLE IT
                 001B    31  07CE  1500          BRW     COND_DEBUG_X              ; EXIT
                         07D1  1501 20$:
04 A1    00000000'8F    D1  07D1  1502          CMPL    #SS$_TBIT, 4(R1)          ; IS THIS A T-BIT TRAP?
                   03    13  07D9  1503          BEQL    30$                       ; YES, CONTINUE
                 0040    31  07DB  1504          BRW     QUICK_X                   ; NO, EXIT                    [21]
                         07DE  1505 30$:
                         07DE  1506          $MP_SET_INTERLOCK #MP$V_IL_CNDB ; Interlock routine     [17][21][23]
                   07    DD  07DE                      PUSHL   #MP$V_IL_CNDB
                   08    DD  07E0                      PUSHL   #$SRVCODE_SET_INTERLOCK
   00000000'9F     02    FB  07E2                      CALLS   #2, @#DS$SERVICE_DISPATCHER
                 020F    30  07E9  1507          BSBW    DBG_TBIT                  ; HANDLE T-BIT
                         07EC  1508
                         07EC  1509 COND_DEBUG_X:                             ;                             [17]
                         07EC  1510
```

```
                                07EC  1511 ;+
                                07EC  1512 ; NOTE: The Ap will clear the interlock (in MP$_COND_DEBUG before going to
                                07EC  1513 ;       its idle loop in order to allow other cpu's to use the service. The
                                07EC  1514 ;       following code prohibits an Ap that is continuing from a BPT from
                                07EC  1515 ;       executing two successive clears of interlock.
                                07EC  1516 ;-
                                07EC  1517
          00000000'EF   D5      07EC  1518          TSTL    MP$GR_BOOTED_PROCESSORS ; Are we an mp system?                  [19]
                  1F    13      07F2  1519          BEQL    40$                     ; Not mp, therefore branch              [19]
               5E   04    C2    07F4  1520          SUBL2   #4,SP                   ; Reseve Storage        [23]
                    5E    DD    07F7  1521          PUSHL   SP                      ; Storage for CPU identifier    [19][23]
       00000000'EF   01    FB    07F9  1522          CALLS   #1, MP$_GET_PROCESSOR_ID        ; Get current CPU identifier    [19]
    6E 00000000'EF   91    0800  1523          CMPB    DS$GB_PRIMARY_CPU_IDENTIFIER,(SP) ; Are we the primary? [19][23]
                  02    12    0807  1524          BNEQ    35$                     ; Were the Ap, exit [19][23]
                  05    11    0809  1525          BRB     37$                     ; Primary flow [23]
               5E   04    C0    080B  1526 35$:    ADDL2   #4,SP                           ; Reset the stack       [23]
                    0E    11    080E  1527          BRB     QUICK_X                         ; [23]
               5E   04    C0    0810  1528 37$:    ADDL2   #4,SP                           ; Reset the stack       [23]
                           0813  1529 40$:    $MP_CLEAR_INTERLOCK #MP$V_IL_CNDB ; CLEAR Interlocked routine   [17][23]
                    07    DD    0813                PUSHL   #MP$V_IL_CNDB
                    09    DD    0815                PUSHL   #$SRVCODE_CLEAR_INTERLOCK
       00000000'9F   02    FB    0817                CALLS   #2, a#DS$SERVICE_DISPATCHER
                           081E  1530
                    04    081E  1531 QUICK_X:        RET                             ; RETURN                [21]
```

```
                        081F  1533          .SBTTL  BREAKPOINT CONDITION SUBROUTINE
                        081F  1534 ;++
                        081F  1535 ; FUNCTIONAL DESCRIPTION:
                        081F  1536 ;
                        081F  1537 ;       This routine setups the context necessary and announces the break point.
                        081F  1538 ;
                        081F  1539 ; CALLING SEQUENCE:
                        081F  1540 ;
                        081F  1541 ;       BSBW    DBG_BREAK
                        081F  1542 ;
                        081F  1543 ; INPUT PARAMETERS:
                        081F  1544 ;
                        081F  1545 ;       STANDARD EXCEPTION ARGS
                        081F  1546 ;
                        081F  1547 ; IMPLICIT INPUTS:      NONE
                        081F  1548 ;
                        081F  1549 ; OUTPUT PARAMETERS:    NONE
                        081F  1550 ;
                        081F  1551 ; IMPLICIT OUTPUTS:     NONE
                        081F  1552 ;
                        081F  1553 ; COMPLETION CODES:     NONE
                        081F  1554 ;
                        081F  1555 ; SIDE EFFECTS:         NONE
                        081F  1556 ;--
```

```
                                 081F  1558 DBG_BREAK:
                                 081F  1559
                                 081F  1560 ;+
                                 081F  1561 ; DETERMINE IF THIS IS A KNOWN BREAKPOINT
                                 081F  1562 ; IF NOT, "RESIGNAL" THE EXCEPTION HANDLER
                                 081F  1563 ;-
                                 081F  1564
            50    04 BC    DE    081F  1565          MOVAL   a4(AP),R0                ; GET ADDR OF SIGNAL ARRAY              ;G:6
            50    08 A0    D0    0823  1566          MOVL    8(R0),R0                 ; GET PC OF BREAKPOINT
                                 0827  1567
  00000014'EF     50    D1    0827  1568          CMPL    R0,L^DS$AA_BPTADDR       ; HIDDEN BREAKPOINT?                    [12]
                  1A    12    082E  1569          BNEQ    10$                      ; Branch if not
                  50    DD    0830  1570          PUSHL   R0                       ; Address to store opcode
  00000001'EF     9F    0832  1571          PUSHAB  L^DS$AB_BPTINST          ; Address of Saved opcode               [12]
                  01    DD    0838  1572          PUSHL   #1                       ; Length to store
  00000B83'EF     03    FB    083A  1573          CALLS   #3,FETCH_STORE           ; Restore the Opcode
       00000014'EF      D4    0841  1574          CLRL    L^DS$AA_BPTADDR          ; Clear hidden breakpoint               [12]
            00B7    31    0847  1575          BRW     CALL_CLI                 ; Enter CLI now                         ;G:6
                                 084A  1576 10$:
            FF47    30    084A  1577          BSBW    FIND_BPT                 ; LOCATE THE BREAKPOINT
          06 50     E8    084D  1578          BLBS    R0,15$                   ; FOUND, TYPE THE PC                    [19]
                                 0850  1579
      50    0000'8F   3C    0850  1580          MOVZWL  #SS$_RESIGNAL,R0         ; NOT OURS
                      05    0855  1581          RSB                              ; RETURN
                                 0856  1582
       00000000'EF      D5    0856  1583 15$:     TSTL    MP$GR_BOOTED_PROCESSORS  ; Are we an mp system?                  [19]
                  70    13    085C  1584          BEQL    17$                      ; Not mp, therefore branch              [19]
  00000000'EF   00000014'EF41   D0    085E  1585          MOVL    DS$AA_BPTADDR[R1],MP$GL_AP_BKPT_STORE ; Store Ap breakpoint info[19]
00000014'EF41   00000054'EF41   C3    086A  1586          SUBL3   L^DS$AA_BPTBASE[R1],L^DS$AA_BPTADDR[R1],L^DS$AA_OFFSET    ;G:8
       00000094'EF              0877
                                 087C  1587                                   ;Calculate Offset to display            ;G:8
  00000000'EF   00000054'EF41   D0    087C  1588          MOVL    DS$AA_BPTBASE[R1],MP$GL_AP_BASE_ADDR_BKPT                ;G:7
    00000000'EF   00000094'EF   D0    0888  1589          MOVL    DS$AA_OFFSET,MP$GL_AP_OFFSET_BKPT                        ;G:7
       00000000'EF              16    0893  1590          JSB     MP$_COND_DEBUG           ; Call mp code                  [19]
                                 0899  1591
                                 0899  1592 ;++                                                                           [19]
                                 0899  1593 ; If the Ap of the breakpoint is resumed is will return here, restore the     [19]
                                 0899  1594 ; context, and REI back to the instruction that causes the BPT.              [19]
                                 0899  1595 ;--                                                                          [19]
                                 0899  1596
                  5E    04    C2    0899  1597          SUBL2   #4,SP                    ; Reserve storage       [23]
                        5E    DD    089C  1598          PUSHL   SP                       ; Storage for CPU identifier    [19]
          00000000'EF   01    FB    089E  1599          CALLS   #1, MP$_GET_PROCESSOR_ID ; Get current CPU identifier    [19]
          6E    00000000'EF   91    08A5  1600          CMPB    DS$GB_PRIMARY_CPU_IDENTIFIER,(SP) ; Are we the primary? [19][23]
                        1D    13    08AC  1601          BEQL    16$                      ; Were the Primary, branch.     [19]
                  5E    04    C0    08AE  1602          ADDL2   #4,SP                    ; Reset the stack               [23]
                                 08B1  1603
                                 08B1  1604 ; Even though the Ap does not call CLI, it must execute the same code such that the
                                 08B1  1605 ; restored.
                                 08B1  1606
                  50    04 AC   7D    08B1  1607          MOVQ    4(AP),R0                 ; GET SIGNAL AND MECHANISM POINTERS
                  7E    08 A0   7D    08B5  1608          MOVQ    8(R0),-(SP)              ; PC/PSL TO STACK
                        10 A0   DF    08B9  1609          PUSHAL  16(R0)                   ; SP TO STACK
                  7E    08 AD   7D    08BC  1610          MOVQ    8(FP),-(SP)              ; AP/FP TO STACK
                     0FFC 8F    BB    08C0  1611          PUSHR   #^XFFC                   ; R11, R10, . . ., R2 TO STACK
                  7E    0C A1   7D    08C4  1612          MOVQ    12(R1),-(SP)             ; R0/R1 TO STACK
                        0069    31    08C8  1613          BRW     CONT                     ; CONTINUE CONTEXT RESTORATION  [19]
```

```
                                08CB  1614
                                08CB  1615 ;++
                                08CB  1616 ; DISPLAY THE 'PC' TO THE OPERATOR
                                08CB  1617 ; THEN CALL THE COMMAND LINE INTERPRETER
                                08CB  1618 ;-
                                08CB  1619
                   5E   04   C0 08CB  1620 16$:    ADDL2   #4,SP                          ; Reset the stack       [23]
00000014'EF41   00000054'EF41   C3 08CE  1621 17$:    SUBL3   L^DS$AA_BPTBASE[R1],L^DS$AA_BPTADDR[R1],L^DS$AA_OFFSET        ;G:6
                  00000094'EF      08DB
                                08E0  1622                                             ;Calculate Offset to display          ;G:6
                                08E0  1623          $DS_PRINTF_S  L^FMTBPT,L^DS$AA_BPTADDR[R1],L^DS$AA_BPTBASE[R1],-     ;G:6
                                08E0  1624                        L^DS$AA_OFFSET                                             ;G:6
                  00000094'EF   DD 08E0                          PUSHL   L^DS$AA_OFFSET
                00000054'EF41   DD 08E6                          PUSHL   L^DS$AA_BPTBASE[R1]
                00000014'EF41   DD 08ED                          PUSHL   L^DS$AA_BPTADDR[R1]
                  00000085'EF   9F 08F4                          PUSHAB  L^FMTBPT
             00000000'9F   04   FB 08FA                          CALLS   #$$N, a#DS$PRINTF
                                0901  1625                                           ; DISPLAY BRKPNT & ITS 'PC'     [12][19] ;G:
                                0901  1626
                                0901  1627 CALL_CLI:
                50   04 AC   7D 0901  1628          MOVQ    4(AP),R0                ; GET SIGNAL AND MECHANISM POINTERS
                7E   08 A0   7D 0905  1629          MOVQ    8(R0),-(SP)             ; PC/PSL TO STACK
                     10 A0   DF 0909  1630          PUSHAL  16(R0)                  ; SP TO STACK
                7E   08 AD   7D 090C  1631          MOVQ    8(FP),-(SP)             ; AP/FP TO STACK
                   0FFC 8F   BB 0910  1632          PUSHR   #^XFFC                  ; R11, R10, . . ., R2 TO STACK
                7E   0C A1   7D 0914  1633          MOVQ    12(R1),-(SP)            ; R0/R1 TO STACK
                                0918  1634
                00000000'EF   16 0918  1635          JSB     SCRIPT$STOP             ; Get out of script mode        [14]
                                091E  1636
             00000000'EF   6E   FA 091E  1637 10$:    CALLG   (SP),L^DS$CLI           ; GO TALK TO OPERATOR                   [16]
                                0925  1638
                                0925  1639 ; OK, NOW RESTORE THE SAVED CONTEXT IN SUCH A WAY THAT ANY MODIFICATIONS
                                0925  1640 ; WILL TAKE EFFECT AFTER FINAL RETURN TO THE PROGRAM
                                0925  1641 ;-
                                0925  1642
                  00000000'EF   D5 0925  1643          TSTL    MP$GR_BOOTED_PROCESSORS ; Are we an mp system?          [19]
                        07      13 092B  1644          BEQL    CONT                    ; Not mp, therefore branch      [19]
             00000000'EF   00   FB 092D  1645          CALLS   #0,MP$_RESUME_ATTACHED_PROCESSORS ; Call mp code         [19]
                                0934  1646
                50   04 AC   7D 0934  1647 CONT:    MOVQ    4(AP),R0                ; GET SIGNAL AND MECHANISM POINTERS     [16]
                0C A1   8E   7D 0938  1648          MOVQ    (SP)+,12(R1)            ; R0/R1 BACK TO MECHANISM ARRAY
                   0FFC 8F   BA 093C  1649          POPR    #^XFFC                  ; R2 - R11 BACK TO REGISTERS
             000000B0'EF   8E   7D 0940  1650          MOVQ    (SP)+,L^NEW_AP_FP       ; AP/FP SAFE                    [12]
             000000B8'EF   8E   D0 0947  1651          MOVL    (SP)+,L^NEW_SP          ; SP TO A SAFE PLACE            [12]
04 AE   40000010 8F   CA 094E  1652          BICL2   #PSL$M_TP!PSL$M_TBIT,4(SP) ; MAY HAVE BEEN SET BY OPERATOR       [12]
             000000BC'EF   8E   7D 0956  1653          MOVQ    (SP)+,L^NEW_PC_PSL      ; PC/PSL TO A SAFE PLACE        [12]
                   0C A0   DC 095D  1654          MOVPSL  12(R0)                  ; PUT CURR PSL ON ORIGINAL STACK
             08 A0   69'AF   9E 0960  1655          MOVAB   B^200$,8(R0)            ; PUT LOCAL PC ON ORIGINAL STACK
                                0965  1656
                50   00'     D0 0965  1657          MOVL    S^#SS$_CONTINUE,R0      ; INDICATE "HANDLED"
                             05 0968  1658          RSB                             ; RETURN TO EXCEPTION HANDLER
                                0969  1659
                                0969  1660 ;+
                                0969  1661 ; THE FOLLOWING CODE IS EXECUTED BY THE 'REI' AT THE END OF THE
                                0969  1662 ; EXCEPTION HANDLER.  THIS FACILITATES THE MODIFICATION OF THE
                                0969  1663 ; 'SP', 'PC' & 'PSL' (BY THE OPERATOR) BEFORE CONTROL RETURNS TO
                                0969  1664 ; THE DIAGNOSTIC PROGRAM.
```

```
                                0969  1665 ;-
                                0969  1666
                                0969  1667 200$:
                     03   BB    0969  1668          PUSHR   #^M<R0,R1>                ; SAVE THESE
                   019C   30    096B  1669          BSBW    RBREAK_OUT               ; REMOVE BREAKPOINTS/PUT OPCODES BACK
                     03   BA    096E  1670          POPR    #^M<R0,R1>               ; RESTORE THEM
          00000014'EF     D4    0970  1671          CLRL    DS$AA_BPTADDR            ; DON'T RESTORE NULL BREAKPOINT
     5C   000000B0'EF     7D    0976  1672          MOVQ    L^NEW_AP_FP,AP           ; SET NEW AP/FP                          [12]
     5E   000000B8'EF     D0    097D  1673          MOVL    L^NEW_SP,SP              ; JAM A NEW POINTER INTO 'SP'            [12]
     7E   000000BC'EF     7D    0984  1674          MOVQ    L^NEW_PC_PSL,-(SP)       ; SETUP FOR REI                          [12]
           04 AE   10     C8    098B  1675          BISL    #PSL$M_TBIT,4(SP)        ; SET THE T-BIT                          [12]
00000000'EF    01     88    098F  1676          BISB2   #1@V_TBIT,L^DEBUG$GB_FLAGS  ; SET T-BIT EXPECTED FLAG                [12]
                                0996  1677
          00000000'EF     D5    0996  1678          TSTL    MP$GR_BOOTED_PROCESSORS          ; Are we an mp system?   [19]
                     5C   13    099C  1679          BEQL    210$                             ; Not mp, therefore branch [19]
                5E   04   C2    099E  1680          SUBL2   #4,SP                            ; Reserve storage       [23]
                     5E   DD    09A1  1681          PUSHL   SP                               ; Storage for CPU identifier [23]
00000000'EF    01     FB    09A3  1682          CALLS   #1, MP$_GET_PROCESSOR_ID         ; Get current CPU identifier
     6E   00000000'EF     91    09AA  1683          CMPB    DS$GB_PRIMARY_CPU_IDENTIFIER,(SP); Are we the primary? [19]
                     44   13    09B1  1684          BEQL    205$                             ; Branch if Pp [23]
                5E   04   C0    09B3  1685          ADDL2   #4,SP                            ; Reset the stack [23]
                                09B6  1686 ;++
                                09B6  1687 ; The following code is used to restore GPR context of an Ap in        [19]
                                09B6  1688 ; the event the user DEPOSITS via SET CPU after hitting an Ap BPT.
                                09B6  1689 ;--
                                09B6  1690
     50   00000000'EF     7D    09B6  1691          MOVQ    MP$GR_SAVED_GPRS,R0      ; Restore Ap GPR's                       [19]
     52   00000008'EF     7D    09BD  1692          MOVQ    MP$GR_SAVED_GPRS+8,R2    ; ...
     54   00000010'EF     7D    09C4  1693          MOVQ    MP$GR_SAVED_GPRS+16,R4   ; ...
     56   00000018'EF     7D    09CB  1694          MOVQ    MP$GR_SAVED_GPRS+24,R6   ; ...
     58   00000020'EF     7D    09D2  1695          MOVQ    MP$GR_SAVED_GPRS+32,R8   ; ...
     5A   00000028'EF     7D    09D9  1696          MOVQ    MP$GR_SAVED_GPRS+40,R10  ; ...
     5C   00000030'EF     7D    09E0  1697          MOVQ    MP$GR_SAVED_GPRS+48,AP   ; ...
     5E   00000038'EF     D0    09E7  1698          MOVL    MP$GR_SAVED_GPRS+56,SP   ; ...
     6E   0000003C'EF     D0    09EE  1699          MOVL    MP$GR_SAVED_GPRS+60,(SP) ; Restore PC                     [19]
                     03   11    09F5  1700          BRB     210$                     ; [23]
                5E   04   C0    09F7  1701 205$:    ADDL2   #4,SP                    ; Reset the stack       [23]
                     02   09FA  1702 210$:    REI                              ; RETURN TO PROGRAM
```

```
                        09FB  1704          .SBTTL  TBIT CONDITION SUBROUTINE
                        09FB  1705 ;++
                        09FB  1706 ; FUNCTIONAL DESCRIPTION:
                        09FB  1707 ;
                        09FB  1708 ;
                        09FB  1709 ; CALLING SEQUENCE:
                        09FB  1710 ;
                        09FB  1711 ;       BSBW    DBG_TBIT
                        09FB  1712 ;
                        09FB  1713 ; INPUT PARAMETERS:     NONE
                        09FB  1714 ;
                        09FB  1715 ; IMPLICIT INPUTS:      NONE
                        09FB  1716 ;
                        09FB  1717 ; OUTPUT PARAMETERS:    NONE
                        09FB  1718 ;
                        09FB  1719 ; IMPLICIT OUTPUTS:     NONE
                        09FB  1720 ;
                        09FB  1721 ; COMPLETION CODES:     NONE
                        09FB  1722 ;
                        09FB  1723 ; SIDE EFFECTS:         NONE                                    ;G:6
                        09FB  1724 ;--
```

```
                                  09FB  DBG_TBIT:
        50   0000'8F     3C      09FB          MOVZWL  #SS$_RESIGNAL,R0          ; PREPARE FOR NOT WANTING THIS
  69 00000000'EF   00   E5      0A00          BBCC    #V_TBIT,L^DEBUG$GB_FLAGS,40$  ; EXIT IF NOT OUR T-BIT          [12
                                  0A08                                          ;                                      [27]
     00000000'EF   01   E5      0A08          BBCC    #V_STEP,L^DEBUG$GB_FLAGS -    ;                                [12]
                   4B           0A0F                  , 20$                     ; FINISH UP IF NOT STEPPING
          50   04 AC   D0      0A10          MOVL    4(AP),R0                  ; GET ADDRESS OF SIGNAL ARGS
                   7E   DF      0A14          PUSHAL  -(SP)                     ; STORE CONTENTS HERE
               08 A0   DD      0A16          PUSHL   8(R0)                     ; CURRENT PC
                   04   DD      0A19          PUSHL   #4                        ; FETCH 4 BYTES
     00000B83'EF   03   FB      0A1B          CALLS   #3,FETCH_STORE            ; GET CONTENTS
                                  0A22  ;                                                                          ;G:4
                 0110   30      0A22          BSBW    RBREAK_IN                 ; RE-INSERT ALL BREAKPOINTS           [27]
                                  0A25  ;                                                                          ;G:6
          50   04 AC   D0      0A25          MOVL    4(AP),R0                  ; GET ADDRESS OF SIGNAL ARGS AGAIN
               08 A0   DD      0A29          PUSHL   8(R0)                     ; PUT PC ON STACK
        000000BB'EF   9F      0A2C          PUSHAB  L^FMTSTEP                 ; ADDRESS OF TEXT                     [12]
     00000000'9F   03   FB      0A32          CALLS   #3,@#DS$PRINTF            ; PRINT 'PC: CONTENTS'
                                  0A39
     50   00000000'EF   DE      0A39          MOVAL   L^DS$GL_CLIBASE,R0        ; BASE CLI DATA                       [12]
             03 1C A0   F5      0A40          SOBGTR  CLI$L_DATA(R0),10$        ; LAST PASS?
                 FEBA   31      0A44          BRW     CALL_CLI                  ; ENTER COMMAND INTERPRETER
                                  0A47  10$:                                                                       ;G:6
                 00C0   30      0A47          BSBW    RBREAK_OUT                ; RESTORE OPCODES
     00000000'EF   03   88      0A4A          BISB    #M_TBIT!M_STEP,L^DEBUG$GB_FLAGS ; SET STEPPING AND T-BIT EXPECT [12]
          50   04 AC   D0      0A51          MOVL    4(AP),R0                  ; GET ADDRESS OF SIGNAL ARRAY
          0C A0   10   C8      0A55          BISL    #PSL$M_TBIT, 12(R0)       ; SET THE T-BIT
                   0F   11      0A59          BRB     30$                       ; EXIT HANDLED
                                  0A5B  20$:
                 00D7   30      0A5B          BSBW    RBREAK_IN                 ;
          50   04 BC   DE      0A5E          MOVAL   @4(AP),R0                 ; GET ADDRESS OF SIGNAL ARRAY
  0C A0   40000010 8F   CA      0A62          BICL    #PSL$M_TBIT!PSL$M_TP,12(R0) ; CLEAR T-BIT(S)
                                  0A6A  30$:
     50   00000000'8F   D0      0A6A          MOVL    #SS$_CONTINUE,R0          ; INDICATE "HANDLED"
                                  0A71  40$:
                        05      0A71          RSB                               ; RETURN
```

```
                        0A72  1763          .SBTTL  RESTORE SCB VECTORS SUBROUTINE                                    ;G:5
                        0A72  1764 ;++                                                                                 ;G:5
                        0A72  1765 ; FUNCTIONAL DESCRIPTION:                                                           ;G:5
                        0A72  1766 ;                                                                                   ;G:5
                        0A72  1767 ;       RESTORES VECTORS SAVED DURING THE EXAMINE AND DEPOSIT COMMANDS           ;G:5
                        0A72  1768 ;                                                                                   ;G:5
                        0A72  1769 ; CALLING SEQUENCE:                                                                 ;G:5
                        0A72  1770 ;                                                                                   ;G:5
                        0A72  1771 ;       BSBW    RESTORE_SCB_VECTORS                                                 ;G:6
                        0A72  1772 ;                                                                                   ;G:5
                        0A72  1773 ; INPUT PARAMETERS:                                                                 ;G:5
                        0A72  1774 ;                                                                                   ;G:5
                        0A72  1775 ;       NONE                                                                        ;G:5
                        0A72  1776 ;                                                                                   ;G:5
                        0A72  1777 ; IMPLICIT INPUTS:                                                                  ;G:5
                        0A72  1778 ;                                                                                   ;G:5
                        0A72  1779 ;       SAVE_SCB_VEC                    ; SAVED SCB VECTORS                         ;G:5
                        0A72  1780 ;       SCB_BASE                        ; SUPERVISORS SCB VECTORS                   ;G:5
                        0A72  1781 ;                                                                                   ;G:5
                        0A72  1782 ; OUTPUT PARAMETERS:    NONE                                                        ;G:5
                        0A72  1783 ;                                                                                   ;G:5
                        0A72  1784 ; IMPLICIT OUTPUTS:     NONE                                                        ;G:5
                        0A72  1785 ;                                                                                   ;G:5
                        0A72  1786 ;       SCB_BASE                        ; SUPERVISORS SCB VECTORS                   ;G:5
                        0A72  1787 ;                                                                                   ;G:5
                        0A72  1788 ; COMPLETION CODES:     NONE                                                        ;G:5
                        0A72  1789 ;                                                                                   ;G:5
                        0A72  1790 ; SIDE EFFECTS:         NONE                                                        ;G:5
                        0A72  1791 ;--                                                                                 ;G:5
                        0A72  1792 RESTORE_SCB_VECTORS:                                                                ;G:5
2D 0000FE00'EF   1C  E0 0A72  1793          BBS     #DSA$V_USER, L^DSA$GL_FLAGS, 10$ ; IF user mode THEN branch     [28]
           0100 8F   BB 0A7A  1794          PUSHR   #^M<R8>                                                         ;G:5
                        0A7E  1795 ;+                                                                                  ;G:5
                        0A7E  1796 ;  RESTORE DIAGNOSTIC PROGRAM'S VECTORS                                             ;G:5
                        0A7E  1797 ;-                                                                                  ;G:5
                     58 D4  0A7E  1798          CLRL    R8                                                          ;G:5
   58   000000A0'EF   DE  0A80  1799          MOVAL   SAVE_SCB_VEC,R8         ; Base Addresss of Saved Vectors        ;G:5
   00000004'EF   88  D0  0A87  1800          MOVL    (R8)+,SCB_BASE+^X4      ; Re-insert Saved Mchk Handler          [28]
   00000018'EF   88  D0  0A8E  1801          MOVL    (R8)+,SCB_BASE+^X18     ; Re-insert Saved Reserv Op Handler
   00000020'EF   88  D0  0A95  1802          MOVL    (R8)+,SCB_BASE+^X20     ; Re-insert Saved Acc Vio Handler
   00000024'EF   88  D0  0A9C  1803          MOVL    (R8)+,SCB_BASE+^X24     ; Re-insert Saved Trans Not Valid Handler
           0100 8F   BA  0AA3  1804          POPR    #^M<R8>                                                         ;G:5
                        0AA7  1805 10$:                                                                                ;G:5
                     05 0AA7  1806          RSB                                                                     ;G:5
                        0AA8  1807                                                                                     ;G:5
                        0AA8  1808                                                                                     ;G:5
```

```
                     0AA8  1810          .SBTTL  TEMPORARY SCB VECTOR STORAGE SUBROUTINE                          ;G:5
                     0AA8  1811 ;++                                                                               ;G:5
                     0AA8  1812 ; FUNCTIONAL DESCRIPTION:                                                         ;G:5
                     0AA8  1813 ;                                                                                 ;G:5
                     0AA8  1814 ;       SAVES SCB VECTORS AND REINSERTS SUPERVISOR'S VECTORS (HANDLERS)            ;G:5
                     0AA8  1815 ;                                                                                 ;G:5
                     0AA8  1816 ; CALLING SEQUENCE:                                                               ;G:5
                     0AA8  1817 ;                                                                                 ;G:5
                     0AA8  1818 ;       BSBW    SAVE_SCB_VECTORS                NO PARAMETERS                     ;G:5
                     0AA8  1819 ;                                                                                 ;G:5
                     0AA8  1820 ; INPUT PARAMETERS:     NONE                                                      ;G:5
                     0AA8  1821 ;                                                                                 ;G:5
                     0AA8  1822 ; IMPLICIT INPUTS:                                                                ;G:5
                     0AA8  1823 ;                                                                                 ;G:5
                     0AA8  1824 ;       SAVE_SCB_VEC                    ; STORAGE FOR SAVED VECTORS               ;G:5
                     0AA8  1825 ;       SCB_IMAGE                       ; INITIAL SCB VECTORS                     ;G:5
                     0AA8  1826 ;       SCB_BASE                        ; MODIFIABLE SCB VECTORS                  ;G:5
                     0AA8  1827 ;                                                                                 ;G:5
                     0AA8  1828 ; OUTPUT PARAMETERS:    NONE                                                      ;G:5
                     0AA8  1829 ;                                                                                 ;G:5
                     0AA8  1830 ; IMPLICIT OUTPUTS:                                                               ;G:6
                     0AA8  1831 ;                                                                                 ;G:5
                     0AA8  1832 ;       SAVE_SCB_VEC                    ; STORAGE FOR SAVED VECTORS               ;G:5
                     0AA8  1833 ;       SCB_IMAGE                       ; INITIAL SCB VECTORS                     ;G:5
                     0AA8  1834 ;       SCB_BASE                        ; MODIFIABLE SCB VECTORS                  ;G:5
                     0AA8  1835 ;                                                                                 ;G:5
                     0AA8  1836 ; COMPLETION CODES:     NONE                                                      ;G:5
                     0AA8  1837 ;                                                                                 ;G:5
                     0AA8  1838 ; SIDE EFFECTS:         NONE                                                      ;G:5
                     0AA8  1839 ;--                                                                               ;G:5
                     0AA8  1840                                                                                   ;G:5
                     0AA8  1841 SAVE_SCB_VECTORS:                                                                 ;G:5
59 0000FE00'EF   1C  E0  0AA8  1842          BBS     #DSA$V_USER, L^DSA$GL_FLAGS, 10$ ; IF user mode THEN branch  [28]
         0100 8F     BB  0AB0  1843          PUSHR   #^M<R8>                 ; SAVE TEMPORARY STORAGE              ;G:5
                     0AB4  1844 ;+                                                                                ;G:5
                     0AB4  1845 ;  SAVE DIAGNOSTIC PROGRAM'S VECTORS                                              ;G:6
                     0AB4  1846 ;-                                                                                ;G:5
               58    D4  0AB4  1847          CLRL    R8                      ;
   58  000000A0'EF   DE  0AB6  1848          MOVAL   SAVE_SCB_VEC,R8         ; Base Address of Saved Vectors
   88  00000004'EF   D0  0ABD  1849          MOVL    SCB_BASE+^X4,(R8)+      ; Save Machine Check Address
   88  00000018'EF   D0  0AC4  1850          MOVL    SCB_BASE+^X18,(R8)+     ; Save Reserved Operand Address
   88  00000020'EF   D0  0ACB  1851          MOVL    SCB_BASE+^X20,(R8)+     ; Save Access Violation Address
   88  00000024'EF   D0  0AD2  1852          MOVL    SCB_BASE+^X24,(R8)+     ; Save Translation Not Valid Address
                     0AD9  1853 ;+                                                                                ;G:5
                     0AD9  1854 ;  RESTORE SUPERVISOR'S VECTORS                                                   ;G:5
                     0AD9  1855 ;-                                                                                ;G:5
       00000004'EF   D0  0AD9  1856          MOVL    SCB_IMAGE+^X4,-                                            ;G:6
       00000004'EF       0ADF  1857                  SCB_BASE+^X4            ; Restore Supervisors MCHK handler
       00000018'EF   D0  0AE4  1858          MOVL    SCB_IMAGE+^X18,-                                           ;G:6
       00000018'EF       0AEA  1859                  SCB_BASE+^X18           ; Restore Supervisors Reserved Operand handl
       00000020'EF   D0  0AEF  1860          MOVL    SCB_IMAGE+^X20,-                                           ;G:6
       00000020'EF       0AF5  1861                  SCB_BASE+^X20           ; Restore Supervisors Access Violation handl
       00000024'EF   D0  0AFA  1862          MOVL    SCB_IMAGE+^X24,-                                           ;G:6
       00000024'EF       0B00  1863                  SCB_BASE+^X24           ; Restore Supervisors Trans not valid handle
         0100 8F     BA  0B05  1864          POPR    #^M<R8>                                                    ;G:5
                     0B09  1865 10$:                                                                            ;G:5
                     05  0B09  1866          RSB                             ; RETURN                          ;G:5
```

```
                    0B0A  1867                                                                  ;G:5
```

```
                        0B0A  1869          .SBTTL  TEMPORARY BREAKPOINT REMOVAL SUBROUTINE
                        0B0A  1870 ;**
                        0B0A  1871 ; FUNCTIONAL DESCRIPTION:
                        0B0A  1872 ;
                        0B0A  1873 ;       REMOVES KNOWN BREAKPOINTS
                        0B0A  1874 ;
                        0B0A  1875 ; CALLING SEQUENCE:
                        0B0A  1876 ;
                        0B0A  1877 ;       BSBW    RBREAK_OUT              NO PARAMETERS
                        0B0A  1878 ;
                        0B0A  1879 ; INPUT PARAMETERS:     NONE
                        0B0A  1880 ;
                        0B0A  1881 ; IMPLICIT INPUTS:
                        0B0A  1882 ;
                        0B0A  1883 ;       DS$AA_BPTADDR                   ; ARRAY OF KNOWN BREAKPOINTS
                        0B0A  1884 ;       DS$AB_BPTINST                   ; ARRAY OF SAVED 'OPCODES'
                        0B0A  1885 ;
                        0B0A  1886 ; OUTPUT PARAMETERS:    NONE
                        0B0A  1887 ;
                        0B0A  1888 ; IMPLICIT OUTPUTS:     NONE
                        0B0A  1889 ;
                        0B0A  1890 ; COMPLETION CODES:     NONE
                        0B0A  1891 ;
                        0B0A  1892 ; SIDE EFFECTS:         NONE
                        0B0A  1893 ;--
                        0B0A  1894
                        0B0A  1895 RBREAK_OUT:
            04    BB    0B0A  1896          PUSHR   #^M<R2>                 ; PRESERVE A TEMPORARY
          52      0F    D0    0B0C  1897          MOVL    #BPTMAX,R2              ; SET UP TABLE LENGTH
00000014'EF42     D5    0B0F  1898 10$:     TSTL    L^DS$AA_BPTADDR[R2]     ; SCAN FOR KNOWN BREAKPOINTS        [12]
            17    13    0B16  1899          BEQL    20$                     ; SKIP IF NOT SET
                        0B18  1900                                          ;
00000014'EF42     DD    0B18  1901          PUSHL   L^DS$AA_BPTADDR[R2]     ; ADDRESS TO STORE ORIGINAL OPCODE  [12]
00000001'EF42     9F    0B1F  1902          PUSHAB  L^DS$AB_BPTINST[R2]     ; ADDRESS OF DATA TO STORE          [12]
            01    DD    0B26  1903          PUSHL   #1                      ; ONLY 1 BYTE
00000B83'EF       03    FB    0B28  1904          CALLS   #3,FETCH_STORE          ; STORE ORIGINAL OPCODE BACK
                        0B2F  1905 20$:
          DD 52     F4    0B2F  1906          SOBGEQ  R2,10$                  ; LOOP UNTIL DONE (INCLUDING [0])
            04    BA    0B32  1907          POPR    #^M<R2>                 ; RESTORE TEMPORARY
                  05    0B34  1908          RSB                             ; RETURN
                        0B35  1909                                                                      ;G:5
```

```
                                0B35          .SBTTL  BREAKPOINT REPLACEMEMT SUBROUTINE
                                0B35  ;++
                                0B35  ; FUNCTIONAL DESCRIPTION:
                                0B35  ;
                                0B35  ;       RESTORES ALL KNOWN BREAKPOINTS
                                0B35  ;
                                0B35  ; CALLING SEQUENCE:
                                0B35  ;
                                0B35  ;       BSBW    RBREAK_IN                                                   ;G:6
                                0B35  ;
                                0B35  ; INPUT PARAMETERS:
                                0B35  ;
                                0B35  ;       NONE
                                0B35  ;
                                0B35  ; IMPLICIT INPUTS:
                                0B35  ;
                                0B35  ;       DS$AA_BPTADDR                   ; ARRAY OF KNOWN BREAKPOINTS
                                0B35  ;       DS$AB_BPTINST                   ; ARRAY OF SAVED 'OPCODES'
                                0B35  ;
                                0B35  ; OUTPUT PARAMETERS:    NONE
                                0B35  ;
                                0B35  ; IMPLICIT OUTPUTS:     NONE
                                0B35  ;
                                0B35  ; COMPLETION CODES:     NONE
                                0B35  ;
                                0B35  ; SIDE EFFECTS:         NONE
                                0B35  ;--
                                0B35
                                0B35  RBREAK_IN:                                ;
                  04    BB      0B35          PUSHR   #^MR2                   ; PRESERVE A TEMPORARY
                                0B37                                          ;
            52    0F    D0      0B37          MOVL    #BPTMAX,R2              ; SET UP TABLE LENGTH
     00000014'EF42      D5      0B3A  10$:    TSTL    L^DS$AA_BPTADDR[R2]     ; SCAN FOR KNOWN BREAKPOINTS          [12]
                  3A    13      0B41          BEQL    30$                     ; SKIP IF NOT SET                     [12]
                                0B43                                          ;                                     [12]
     00000001'EF42      9F      0B43          PUSHAB  L^DS$AB_BPTINST[R2]     ; ADDRESS TO STORE ORIGINAL OPCODE    [12]
     00000014'EF42      DD      0B4A          PUSHL   L^DS$AA_BPTADDR[R2]     ; ADDRESS TO INSERT BREAKPOINT        [12]
                  01    DD      0B51          PUSHL   #1                      ; INSERT ONLY ONE BYTE
  00000B83'EF     03    FB      0B53          CALLS   #3,FETCH_STORE          ; SAVE OPCODE
               19 50    E9      0B5A          BLBC    R0,20$                  ; CAN'T READ, REMOVE BREAKPOINT
                                0B5D
     00000014'EF42      DD      0B5D          PUSHL   L^DS$AA_BPTADDR[R2]     ; ADDRESS TO INSERT BREAKPOINT        [12]
       00000006'EF      9F      0B64          PUSHAB  L^BPT_LITERAL           ; BPT INSTRUCTION                     [12]
                  01    DD      0B6A          PUSHL   #1                      ; INSERT 1 BYTE
  00000B83'EF     03    FB      0B6C          CALLS   #3,FETCH_STORE          ; STORE IT
               07 50    E8      0B73          BLBS    R0,30$                  ; OK, NEXT
                                0B76  20$:
     00000014'EF42      D4      0B76          CLRL    L^DS$AA_BPTADDR[R2]     ; REMOVE BREAKPOINT                   [12]
               BA 52    F4      0B7D  30$:    SOBGEQ  R2,10$                  ; LOOP UNTIL DONE (INCLUDING [0])
                                0B80                                          ;
                  04    BA      0B80          POPR    #^MR2                   ; RESTORE TEMPORARY
                        05      0B82          RSB                             ; RETURN
```

```
                     0B83  1964          .SBTTL  FETCH_STORE     COPY BYTE ROUTINE
                     0B83  1965 ;++
                     0B83  1966 ; FUNCTIONAL DESCRIPTION:
                     0B83  1967 ;
                     0B83  1968 ;       This routine will copy bytes of memory from one place to another.
                     0B83  1969 ;
                     0B83  1970 ; CALLING SEQUENCE:
                     0B83  1971 ;
                     0B83  1972 ;       CALLG   (AP),FETCH_STORE
                     0B83  1973 ;
                     0B83  1974 ; INPUT PARAMETERS:
                     0B83  1975 ;
                     0B83  1976 ;        4(AP)  Count of bytes to store
                     0B83  1977 ;        8(AP)  Address to fetch data from
                     0B83  1978 ;       12(AP)  Address to store data
                     0B83  1979 ;
                     0B83  1980 ; IMPLICIT INPUTS:      NONE
                     0B83  1981 ;
                     0B83  1982 ; OUTPUT PARAMETERS:    NONE
                     0B83  1983 ;
                     0B83  1984 ; IMPLICIT OUTPUTS:     NONE
                     0B83  1985 ;
                     0B83  1986 ; COMPLETION CODES:
                     0B83  1987 ;
                     0B83  1988 ;       SS$_NORMAL      If all went well
                     0B83  1989 ;       DS$_MACHCK      If bad addresses
                     0B83  1990 ;       SS$_ACCVIO      If memory managment faults
                     0B83  1991 ;       DS$_TRANSL      If translation not valid fault
                     0B83  1992 ;
                     0B83  1993 ; SIDE EFFECTS:
                     0B83  1994 ;
                     0B83  1995 ;       Memory as specified by 12(AP) is changed
                     0B83  1996 ;--                                                          ;G:6
                     0B83  1997          $OFFSET 0,NEGATIVE,<-
                     0B83  1998                  <DATA,8>, -             ; Space to save data
                     0B83  1999                  <ARG1,6*4>, -           ; Space for SETPRT arglist
                     0B83  2000                  <ADR1,8>, -             ; Descriptor of page(s)
                     0B83  2001                  <ARG2,6*4>, -           ; Space for SETPRT arglist
                     0B83  2002                  <ADR2,8>, -             ; Descriptor of page(s)
                     0B83  2003                  <LOCAL,0>>              ; Length of local storage required
                     0B83                .SAVE   LOCAL_BLOCK
                     0B83                .PSECT $ABS$ ABS
00000000  FE70                           .=0
FFFFFFF8  0000                           .BLKB   -8
          FFF8       DATA:
FFFFFFE0  FFF8                           .BLKB   -6*4
          FFE0       ARG1:
FFFFFFD8  FFE0                           .BLKB   -8
          FFD8       ADR1:
FFFFFFC0  FFD8                           .BLKB   -6*4
          FFC0       ARG2:
FFFFFFB8  FFC0                           .BLKB   -8
          FFB8       ADR2:
          FFB8       LOCAL:
       00000B83                          .RESTORE
```

```
                     003C  0B83  2005 .ENTRY  FETCH_STORE,^M<R2,R3,R4,R5>
                           0B85  2006
          55    5D   D0   0B85  2007         MOVL    FP,R5                       ; Base of local storage
        5E   B8 AD   9E   0B88  2008         MOVAB   LOCAL(FP),SP                ; Allocate space for local storage
           F0 AD   D4   0B8C  2009         CLRL    ARG1+SETPRT$_PROT(FP)       ; Mark protection not changed
           D0 AD   D4   0B8F  2010         CLRL    ARG2+SETPRT$_PROT(FP)       ; Mark protection not changed
                           0B92  2011
      6D   00000BFF'EF   9E   0B92  2012         MOVAB   L^FS_HANDLR,(FP)            ; Set exception handler                 [12]
                           0B99  2013                                                                                  ;G:5
                           0B99  2014                                                                                  ;G:5
          52   04 AC   7D   0B99  2015         MOVQ    4(AP),R2                    ; Get Length and source address         ;G:5
             51   53   D0   0B9D  2016         MOVL    R3,R1                       ; Desired address
                           0BA0  2017
        63   52   00   0C   0BA0  2018         PROBER  #PSL$C_KERNEL,R2,(R3)       ; Readable?
                  06   12   0BA4  2019         BNEQ    10$                         ; Branch if readable
             50   00'  D0   0BA6  2020         MOVL    S^#PRT$C_UR,R0              ; Desired protection
                010F   30   0BA9  2021         BSBW    FS_SETUP                    ; Setup and set protections
                           0BAC  2022
          53   08 AC   D0   0BAC  2023 10$:    MOVL    8(AP),R3                    ; Get source address
          54   F8 AD   9E   0BB0  2024         MOVAB   DATA(FP),R4                 ; Intermediate
                0023   30   0BB4  2025         BSBW    FS_COPY                     ; Copy from (R3)+ to (R4)+ by R2
                0197   30   0BB7  2026         BSBW    FS_RESET                    ; Put input protections back
                           0BBA  2027
          54   0C AC   D0   0BBA  2028         MOVL    12(AP),R4                   ; Output address
             51   54   D0   0BBE  2029         MOVL    R4,R1                       ; Desired address
        64   52   00   0D   0BC1  2030         PROBEW  #PSL$C_KERNEL,R2,(R4)       ; Writeable?
                  06   12   0BC5  2031         BNEQ    20$                         ; YES, DO IT
             50   00'  D0   0BC7  2032         MOVL    S^#PRT$C_UW,R0              ; Desired protection
                00EE   30   0BCA  2033         BSBW    FS_SETUP                    ; Setup and set protections
                           0BCD  2034
          53   F8 AD   9E   0BCD  2035 20$:    MOVAB   DATA(FP),R3                 ; Source for copy
                07   10   0BD1  2036         BSBB    FS_COPY                     ; Copy data to target
              017B   30   0BD3  2037         BSBW    FS_RESET                    ; Put output protections back
           50   01   D0   0BD6  2038         MOVL    S^#SS$_NORMAL,R0            ; Success!
                     04   0BD9  2039         RET
```

```
                        0BDA  2041 ;+
                        0BDA  2042 ; Copy data from one place to another, respect desired length Q,L,W,B
                        0BDA  2043 ; Read every address so we get machine checks instead of write timeouts
                        0BDA  2044 ;       R1 = address to report if error
                        0BDA  2045 ;       R2 = length 0-0F
                        0BDA  2046 ;       R3 = source address
                        0BDA  2047 ;       R4 = destination address
                        0BDA  2048 ;-
                        0BDA  2049 FS_COPY:
   06 52   03   E1      0BDA  2050         BBC     #3,R2,10$                ; Branch if NOT quadword
      64   64   7D      0BDE  2051         MOVQ    (R4),(R4)                ; Read to check for accessability
      84   83   7D      0BE1  2052         MOVQ    (R3)+,(R4)+              ; Copy quadword
   05 52   02   E1      0BE4  2053 10$:    BBC     #2,R2,20$                ; Branch if NOT longword
           64   D5      0BE8  2054         TSTL    (R4)                     ; Check accessability
      84   83   D0      0BEA  2055         MOVL    (R3)+,(R4)+              ; Copy longword
   05 52   01   E1      0BED  2056 20$:    BBC     #1,R2,30$                ; Branch if NOT word
           64   B5      0BF1  2057         TSTW    (R4)                     ; Check accessability
      84   83   B0      0BF3  2058         MOVW    (R3)+,(R4)+              ; Copy word
      05 52   E9        0BF6  2059 30$:    BLBC    R2,40$                   ; Branch if NOT byte
           64   95      0BF9  2060         TSTB    (R4)                     ; Check accessability
      84   83   90      0BFB  2061         MOVB    (R3)+,(R4)+              ; Copy byte
                05      0BFE  2062 40$:    RSB
                        0BFF  2063
```

```
                                0BFF  2065 ;+
                                0BFF  2066 ; Exception handler for FETCH_STORE and FETCH_STOR_PREG routines.
                                0BFF  2067 ;-
                                0BFF  2068 FS_HANDLR:
                          0020  0BFF  2069         .WORD   ^M<R5>                  ; Need this register for old FP
                                0C01  2070 ;+                                                                   ;G:5
                                0C01  2071 ;  RESTORE DIAGNOSTIC PROGAM'S SCB VECTORS                           ;G:5
                                0C01  2072 ;-                                                                   ;G:5
                                0C01  2073                                                                      ;G:5
    25 0000FE00'EF    1C    E0  0C01  2074         BBS     #DSA$V_USER,L^DSA$GL_FLAGS,109$ ; IF user mode THEN branch  [28]
                      58    D4  0C09  2075         CLRL    R8                                                    ;G:5
       58 000000A0'EF     DE  0C0B  2076         MOVAL   SAVE_SCB_VEC,R8         ; Base Addresss of Saved Vectors ;G:5
       00000004'EF    88   D0  0C12  2077         MOVL    (R8)+,SCB_BASE+^X4      ; Re-insert Saved Mchk Handler  [28]
       00000018'EF    88   D0  0C19  2078         MOVL    (R8)+,SCB_BASE+^X18     ; Re-insert Saved Reserv Op Handler
       00000020'EF    88   D0  0C20  2079         MOVL    (R8)+,SCB_BASE+^X20     ; Re-insert Saved Acc Vio Handler
       00000024'EF    88   D0  0C27  2080         MOVL    (R8)+,SCB_BASE+^X24     ; Re-insert Saved Trans Not Valid Handler
                                0C2E  2081                                                                      ;G:5
             50    04 AC    7D  0C2E  2082 109$:   MOVQ    4(AP),R0                ; Get address of mech/signal args ;G:5
             55    04 A1    D0  0C32  2083         MOVL    4(R1),R5                ; Get FP at time of exception
    04 A0    00660088 8F    D1  0C36  2084         CMPL    #DS$_MCHK,4(R0)         ; MACHINE CHECK?
                      0D    12  0C3E  2085         BNEQ    110$                    ; NO, TRY NEXT
       50 000001AB'EF     9E  0C40  2086         MOVAB   L^T_MACHCK,R0           ; POINT TO ERROR TEXT           [12]
             51    10 A1    D0  0C47  2087         MOVL    16(R1),R1               ; GET VIRTURAL ADDRESS
                      45    11  0C4B  2088         BRB     150$                    ; AND PRINT IT
    04 A0    006600A0 8F    D1  0C4D  2089 110$:   CMPL    #DS$_TRANSL,4(R0)       ; TRANSLATION NOT VALID?
                      0D    12  0C55  2090         BNEQ    120$                    ; NO, TRY NEXT
             51    0C A0    D0  0C57  2091         MOVL    8+4(R0),R1              ; Get virtural address
       50 000001E0'EF     9E  0C5B  2092         MOVAB   L^T_TRANSL,R0           ; POINT TO ERROR TEXT           [12]
                      2E    11  0C62  2093         BRB     150$                    ; PRINT
    04 A0    00000000'8F    D1  0C64  2094 120$:   CMPL    #SS$_ACCVIO,4(R0)       ; ACCESS VIOLATION?
                      0D    12  0C6C  2095         BNEQ    130$                    ; No, try next
             51    0C A0    D0  0C6E  2096         MOVL    8+4(R0),R1              ; Get virtural address
       50 000001CC'EF     9E  0C72  2097         MOVAB   L^T_ACCVIO,R0           ; POINT TO TEXT                 [12]
                      17    11  0C79  2098         BRB     150$                    ; PRINT
                                0C7B  2099 130$:
    04 A0    00000000'8F    D1  0C7B  2100         CMPL    #SS$_ROPRAND,4(R0)      ; Invalid operand? (must be IPR if so)
                      07    12  0C83  2101         BNEQ    145$                    ; Nope.
          25 0C A1    00    E4  0C85  2102         BBSC    #0,12(R1),155$          ; Clear success bit in return code
                      23    11  0C8A  2103         BRB     155$                    ; Exit
                                0C8C  2104 145$:
             50    0000'8F    3C  0C8C  2105         MOVZWL  #SS$_RESIGNAL,R0        ; PAWN IT OFF ON SOMEONE ELSE
                      04    0C91  2106         RET                             ; RETURN FROM CONDITION HANDLER
                                0C92  2107
                                0C92  2108 150$:   $PRINTI_S L^FMT_EXCEPTION,R0,R1 ; TYPE THE ERROR TEXT        [12]
                      51    DD  0C92                       PUSHL   R1
                      50    DD  0C94                       PUSHL   R0
          000001F9'EF     9F  0C96                       PUSHAB  L^FMT_EXCEPTION
    00000000'EF       03    FB  0C9C                       CALLS   #$$N, DSX$PRINTI
             50    04 AC    7D  0CA3  2109         MOVQ    4(AP),R0                ; SIGNAL AND MECHANISM ADDRESSES
          0C A1    04 A0    D0  0CA7  2110         MOVL    4(R0),12(R1)            ; Put condition in R0
                      00A2  30  0CAC  2111         BSBW    FS_RESET                ; Put protections back
                                0CAF  2112 155$:   $UNWIND_S                       ; Return to caller
                      00    DD  0CAF                       PUSHL   #0
                      00    DD  0CB1                       PUSHL   #0
    00000000'GF       02    FB  0CB3                       CALLS   #2,G^SYS$UNWIND
                      04    0CBA  2113         RET                             ; RETURN
```

```
                                0CBB  2115 ;+
                                0CBB  2116 ; Set protections for page(s)
                                0CBB  2117 ;       R0 = desired protection
                                0CBB  2118 ;       R1 = desired address, preserved
                                0CBB  2119 ;       R2 = length of area, 0-F
                                0CBB  2120 ;-
                                0CBB  2121 FS_SETUP:
                     02   BB    0CBB  2122          PUSHR   #^M<R1>                 ; Save address
                F0 AD   50   9A    0CBD  2123          MOVZBL  R0,ARG1+SETPRT$_PROT(FP); Set desired protection
                D0 AD   50   9A    0CC1  2124          MOVZBL  R0,ARG2+SETPRT$_PROT(FP); Set desired protection
             50   01FF 8F   3C    0CC5  2125          MOVZWL  #^X1FF,R0               ; Byte within page bits
          D8 AD   51   50   CB    0CCA  2126          BICL3   R0,R1,ADR1(FP)          ; Starting address for this page
             51   FF A142 9E    0CCF  2127          MOVAB   -1(R1)[R2],R1           ; Address of last byte to be affected
          DC AD   51   50   CB    0CD4  2128          BICL3   R0,R1,ADR1+4(FP)        ; Ending address
            DC AD   D8 AD   D1    0CD9  2129          CMPL    ADR1(FP),ADR1+4(FP)     ; Start same as end?
                     2B   13    0CDE  2130          BEQL    20$                     ; Branch if only one page need modifications
                                0CE0  2131
            B8 AD   DC AD   D0    0CE0  2132          MOVL    ADR1+4(FP),ADR2(FP)     ; Set start of page
            BC AD   B8 AD   D0    0CE5  2133          MOVL    ADR2(FP),ADR2+4(FP)     ; Set end of page
            D4 AD   D2 AD   DE    0CEA  2134          MOVAL   ARG2+SETPRT$_PROT+2(FP), -
                                0CEF  2135                  ARG2+SETPRT$_PRVPRT(FP) ; Address to return protection
                     C8 AD   7C    0CEF  2136          CLRQ    ARG2+SETPRT$_RETADR(FP) ; No return address and kernel mode
            C4 AD   B8 AD   7E    0CF2  2137          MOVAQ   ADR2(FP), -
                                0CF7  2138                  ARG2+SETPRT$_INADR(FP)  ; Address of start and end pages
                C0 AD   05   D0    0CF7  2139          MOVL    #SETPRT$_NARGS,ARG2(FP) ; Set count of args
                                0CFB  2140
        00000000'EF   C0 AD   FA    0CFB  2141          CALLG   ARG2(FP),SYS$SETPRT     ; Set the protection
                     2E 50   E9    0D03  2142          BLBC    R0,30$                  ; Branch if it failed
                D0 AD   40 8F   88    0D06  2143          BISB    #64,ARG2+SETPRT$_PROT(FP); Flag protection changed
                                0D0B  2144
                DC AD   D8 AD   D0    0D0B  2145 20$:     MOVL    ADR1(FP),ADR1+4(FP)     ; Set end address of range
                F4 AD   F2 AD   DE    0D10  2146          MOVAL   ARG1+SETPRT$_PROT+2(FP), -
                                0D15  2147                  ARG1+SETPRT$_PRVPRT(FP) ; Address to return protection
                     E8 AD   7C    0D15  2148          CLRQ    ARG1+SETPRT$_RETADR(FP) ; No return address and kernel mode
                E4 AD   D8 AD   7E    0D18  2149          MOVAQ   ADR1(FP), -
                                0D1D  2150                  ARG1+SETPRT$_INADR(FP)  ; Address of start and end pages
                   E0 AD   05   D0    0D1D  2151          MOVL    #SETPRT$_NARGS,ARG1(FP) ; Set count of args
                                0D21  2152
        00000000'EF   E0 AD   FA    0D21  2153          CALLG   ARG1(FP),SYS$SETPRT     ; Set the protection
                     08 50   E9    0D29  2154          BLBC    R0,30$                  ; Branch if it failed
                F0 AD   40 8F   88    0D2C  2155          BISB    #64,ARG1+SETPRT$_PROT(FP); Flag protection changed
                     02   BA    0D31  2156          POPR    #^M<R1>                 ; Restore R1
                     05   05    0D33  2157          RSB
                                0D34  2158
                52   50   D0    0D34  2159 30$:     MOVL    R0,R2                   ; Save SETPRT return code
                                0D37  2160                                          ; Address already on stack
              000001CC'EF   9F    0D37  2161          PUSHAB  L^T_ACCVIO              ; ADDRESS OF ACCESS VIOLATION INSERT    [12]
              000001F9'EF   9F    0D3D  2162          PUSHAB  L^FMT_EXCEPTION         ; FAO STRING                            [12]
           00000000'EF   03   FB    0D43  2163          CALLS   #3, DSX$PRINTI          ; TYPE ACCESS VIOLATION TEXT
                                0D4A  2164
                0004   30    0D4A  2165          BSBW    FS_RESET                ; Reset the protections
             50   52   D0    0D4D  2166          MOVL    R2,R0                   ; Restore return code
                     04   04    0D50  2167          RET                             ; RETURN TO CALLER OF FETCH_STORE
                                0D51  2168 ;+
                                0D51  2169 ; Routine to restore protections on pages changed
                                0D51  2170 ;       R5 = FP of local data structure
                                0D51  2171 ;-
```

```
                        0D51  2172 FS_RESET:
          50    F0 A5   9E  0D51  2173          MOVAB   ARG1+SETPRT$_PROT(R5),R0; Address of protections
          0C 60   06    E5  0D55  2174          BBCC    #6,(R0),10$             ; Branch if protection not changed
          60    02 A0   90  0D59  2175          MOVB    2(R0),(R0)              ; Put old protection in arglist
00000000'EF     E0 A5   FA  0D5D  2176          CALLG   ARG1(R5),SYS$SETPRT     ; Set protection back
                        0D65  2177
          50    D0 A5   9E  0D65  2178 10$:     MOVAB   ARG2+SETPRT$_PROT(R5),R0; Address of protections
          0C 60   06    E5  0D69  2179          BBCC    #6,(R0),20$             ; Branch if protection not changed
          60    02 A0   90  0D6D  2180          MOVB    2(R0),(R0)              ; Put old protection in arglist
00000000'EF     C0 A5   FA  0D71  2181          CALLG   ARG2(R5),SYS$SETPRT     ; Set protection back
                        0D79  2182
                    05  0D79  2183 20$:     RSB                             ; Return
                        0D7A  2184
```

```
                        0D7A  2186          .SBTTL  Handle EXAMINE/DEPOSIT of IPRs in KERNEL mode
                        0D7A  2187 ;+
                        0D7A  2188 ; CHMK handler: gets control via CHMK dispatch routine.  It calls routine
                        0D7A  2189 ; FETCH_STOR_PREG, in KERNEL mode, to allow use of MFPR/MTPR instructions
                        0D7A  2190 ; and handling of possible exception due to access
                        0D7A  2191 ;       PARAMETERS:
                        0D7A  2192 ;               R1      Value to deposit (if function is DEPOSIT)
                        0D7A  2193 ;               R3      IPR code
                        0D7A  2194 ;               R5      function code: 1 if EXAMINE, 0 if DEPOSIT
                        0D7A  2195 ;       RETURN:
                        0D7A  2196 ;               R1      Value examined--is "read-back" of deposited
                        0D7A  2197 ;                       value if function is DEPOSIT
                        0D7A  2198 ;-
                        0D7A  2199 CMK$SGIPR::
              51    DD  0D7A  2200          PUSHL   R1                      ; Data, if deposit (else save space)
              6E    DF  0D7C  2201          PUSHAL  (SP)                    ; Address of where to put/get data
              55    DD  0D7E  2202          PUSHL   R5                      ; Flags word to determine if exam/dep
              53    DD  0D80  2203          PUSHL   R3                      ; Register Code
        90'AF     03    FB  0D82  2204          CALLS   #3,B^FETCH_STOR_PREG    ; Do it
              02    BA  0D86  2205          POPR    #^M<R1>                 ; Get data back if examine (else clear stack
  00000000'EF   51    D0  0D88  2206          MOVL    R1,DS$GL_IPR_VALUE      ; Variable to store value if called by a bli
                        02  0D8F  2207          REI                             ; Return from the interrupt
                            0D90  2208
```

```
                                0D90  2210 .SBTTL  FETCH_STOR_PREG Move data to/from IPR
                                0D90  2211 ;++
                                0D90  2212 ; FUNCTIONAL DESCRIPTION:
                                0D90  2213 ;
                                0D90  2214 ;       This routine will either copy data from memory to an internal CPU
                                0D90  2215 ;       register (DEPOSIT) or move data from an internal CPU register into
                                0D90  2216 ;       memory (EXAMINE).  If function is DEPOSIT, it will also preform an
                                0D90  2217 ;       EXAMINE to the same IPR, providing a verify read-back.
                                0D90  2218 ;
                                0D90  2219 ; CALLING SEQUENCE:
                                0D90  2220 ;
                                0D90  2221 ;       CALLS   #3,FETCH_STOR_PREG
                                0D90  2222 ;
                                0D90  2223 ; INPUT PARAMETERS:
                                0D90  2224 ;
                                0D90  2225 ;       4(AP)   The code for the desired IPR (range 0 to FF(X))
                                0D90  2226 ;       8(AP)   A flag (=1 for EXAMINE, =0 for DEPOSIT)
                                0D90  2227 ;       12(AP)  Address of location to get/store data
                                0D90  2228 ;
                                0D90  2229 ; IMPLICIT INPUTS:      None
                                0D90  2230 ;
                                0D90  2231 ; OUTPUT PARAMETERS:    None
                                0D90  2232 ;
                                0D90  2233 ; IMPLICIT OUTPUTS:     None
                                0D90  2234 ;
                                0D90  2235 ; COMPLETION CODES:
                                0D90  2236 ;
                                0D90  2237 ;       R0 bits define operations completed:
                                0D90  2238 ;                bit0 set = no error
                                0D90  2239 ;                bit1 set = DEPOSIT completed with no error
                                0D90  2240 ;
                                0D90  2241 ; SIDE EFFECTS:         None
                                0D90  2242 ;--
                                0D90  2243
                          000C  0D90  2244 .ENTRY  FETCH_STOR_PREG,^M<R2,R3>
6D  FFFFFE67 EF     9E  0D92  2245          MOVAB   L^FS_HANDLR,(FP)        ; Set up exception handler              [12]
     52   04 AC    D0  0D99  2246          MOVL    4(AP),R2                ; Get IPR code
     53   0C AC    D0  0D9D  2247          MOVL    12(AP),R3               ; Get memory address
        50   01    D0  0DA1  2248          MOVL    #1,R0                   ; Presume success
      07 08 AC    E8  0DA4  2249          BLBS    8(AP),10$               ; EXAMINE only
        52   63    DA  0DA8  2250          MTPR    (R3),R2                 ; Attempt to move data to register
     00 50   01    E2  0DAB  2251          BBSS    #1,R0,10$               ; DEPOSIT worked--now do read-back
        63   52    DB  0DAF  2252 10$:     MFPR    R2,(R3)                 ; Try to move data from register
                   04  0DB2  2253          RET
```

```
                                0DB3  2255 .SBTTL  DSV$DEBUG - Process DEBUG Command                                [14]
                                0DB3  2256 ;++                                                                        [14]
                                0DB3  2257 ; FUNCTIONAL DESCRIPTION                                                   [14]
                                0DB3  2258 ;       This routine will process the DEBUG command by allocating      [14]
                                0DB3  2259 ;       memory space for the debugger, loading the debugger, and       [14]
                                0DB3  2260 ;       then calling it.                                               [14]
                                0DB3  2261 ;                                                                      [14]
                                0DB3  2262 ; CALLING SEQUENCE                                                         [14]
                                0DB3  2263 ;       BSBW    DSV$DEBUG                                              [14]
                                0DB3  2264 ;                                                                      [14]
                                0DB3  2265 ; INPUT PARAMETERS                                                         [14]
                                0DB3  2266 ;       NONE                                                           [14]
                                0DB3  2267 ;                                                                      [14]
                                0DB3  2268 ; OUTPUT PARAMETERS                                                        [14]
                                0DB3  2269 ;       NONE                                                           [14]
                                0DB3  2270 ;                                                                      [14]
                                0DB3  2271 ;--                                                                    [14]
                                0DB3  2272
                   00000002     0DB3  2273 TRANS_VECTOR_OFFSET = 2                ;BYTE OFFSET FROM TOP OF IMAGE  [14]
                                0DB3  2274                                        ;   HEADER, CONTAINING OFFSET   [14]
                                0DB3  2275                                        ;   TO TRANSFER VECTOR ARRAY    [14]
                                0DB3  2276
45 58 45 2E 00000DBB'010E0000'  0DB3  2277 DBG_DEF:        .ASCID /.EXE/          ;                              [14]
                                0DBF  2278 DSV$DEBUG::                            ;                              [14]
                     1A   E1    0DBF  2279         BBC     #DSA$V_DEBUG,-         ;IF DEBUGGER ALREADY LOADED    [14]
          03 0000FE00'EF        0DC1  2280                 DSA$GL_FLAGS,5$        ;THEN                          [14]
                   0111   31    0DC7  2281         BRW     27$                    ;   DON'T RELOAD               [14]
                                0DCA  2282 5$:     $DS_PRINTF_S    FMT_DBG_MSG    ;PRINT MESSAGE                 [14]
          00000163'EF   9F      0DCA                       PUSHAB  FMT_DBG_MSG
       00000000'9F    01    FB  0DD0                       CALLS   #$$N, @#DS$PRINTF
                     1C   E1    0DD7  2283         BBC     #DSA$V_USER,-          ;IF USER MODE                  [14]
            48 0000FE00'EF      0DD9  2284                 DSA$GL_FLAGS,10$       ;THEN                          [14]
000001C9'EF    000FFFFF 8F    D0  0DDF  2285         MOVL    #^X000FFFFF,-          ;   SET VA FOR TOP OF DEBUGGER [14]
                                0DEA  2286                 DBG_ADDR_REQ+4         ;    (1000K)                   [14]
                00010E00 8F    C3  0DEA  2287         SUBL3   #DEBUGGER_SIZE,-       ;   SUBTRACT DEBUGGER SIZE     [14]
                000001C9'EF        0DF0  2288                 DBG_ADDR_REQ+4,-       ;    FROM TOP ADDRESS          [14]
                000001C5'EF        0DF5  2289                 DBG_ADDR_REQ           ;    AND STORE AS START VA     [14]
                                0DFA  2290         $CRETVA_S       DBG_ADDR_REQ, - ;   CALL SYSTEM SERVICE TO ALLO-[14]
                                0DFA  2291                         DBG_ADDR_RTN   ;    CATE MEMORY FOR DEBUGGER  [14]
                          00    DD  0DFA                       PUSHL   #0
                000001CD'EF    7F  0DFC                       PUSHAQ  DBG_ADDR_RTN
                000001C5'EF    7F  0E02                       PUSHAQ  DBG_ADDR_REQ
          00000000'GF    03    FB  0E08                       CALLS   #3,G^SYS$CRETVA
000001D5'EF     000001CD'EF    D0  0E0F  2292         MOVL    DBG_ADDR_RTN,DEBUG_LO  ;   STORE DEBUGGER LOW ADDRESS  [14]
000001D9'EF     000001D1'EF    D0  0E1A  2293         MOVL    DBG_ADDR_RTN+4,DEBUG_HI;   STORE DEBUGGER HIGH ADDRESS [14]
                     5A   11    0E25  2294         BRB     20$                    ;ELSE                          [14]
                00010E00 8F    C3  0E27  2295 10$:    SUBL3   #DEBUGGER_SIZE,-       ;   SUBTRACT DEBUGGER SIZE     [14]
                00000000'EF        0E2D  2296                 DS$GL_MEMSIZE,-        ;    FROM TOP ADDRESS          [14]
                000001CD'EF        0E32  2297                 DBG_ADDR_RTN           ;    AND STORE AS START VA     [14]
000001CD'EF     00000200 8F    C2  0E37  2298         SUBL2   #512,DBG_ADDR_RTN      ;    DON'T USE TOP PAGE OF MEM. [14]
                                0E42  2299                                        ;    (IT'S USED BY A DIAGNOSTIC!)[14]
00000000'EF     00000200 8F    C3  0E42  2300         SUBL3   #512,DS$GL_MEMSIZE,-   ;   TOP OF DEBUGGER AT TOP OF MEM[14]
                000001D9'EF        0E4D  2301                 DEBUG_HI               ;    MINUS LAST PAGE           [14]
000001D5'EF     000001CD'EF    D0  0E52  2302         MOVL    DBG_ADDR_RTN,DEBUG_LO  ;   BOTTOM OF DEBUGGER         [14]
000001DD'EF     000001D5'EF    D0  0E5D  2303         MOVL    DEBUG_LO,SYMTAB_HI     ;   ALSO WHERE SYM. TABLES START[14]
                00000000'EF    16  0E68  2304         JSB     MAP_DEBUGGER           ;   MAP MEMORY SPACE FOR DEBUGGER[14]
                      10 50    E8  0E6E  2305         BLBS    R0,20$                 ;   IF MAPPING ERROR THEN      [14]
```

```
                                0E71  2306          $DS_PRINTF_S    FMT_NO_ROOM     ;       PRINT MSG                       [14]
         00000192'EF    9F     0E71                        PUSHAB  FMT_NO_ROOM
   00000000'9F    01    FB     0E77                        CALLS   #$$N, a#DS$PRINTF
                00AB    31     0E7E  2307          BRW     DSV$DEBUG_EXIT          ;       LEAVE                           [14]
                                0E81  2308 20$:
                  00    DD     0E81  2309          PUSHL   #0                      ;FAKE VBN NUMBER                        [22]
         000001FA'EF    DF     0E83  2310          PUSHAL  RET_REC                 ;LONGWORD TO RECEICE REC. ATTR.         [22]
         000001FE'EF    DF     0E89  2311          PUSHAL  RET_LEN                 ;LONGWORD TO RECEIVE LENGTH             [22]
         000001CD'EF    DD     0E8F  2312          PUSHL   DBG_ADDR_RTN            ;ADDRESS TO LOAD DEBUGGER               [14]
         00010E00 8F    DD     0E95  2313          PUSHL   #DEBUGGER_SIZE          ;LENGTH OF LOAD AREA                    [14]
         FFFFFF12 EF    7F     0E9B  2314          PUSHAQ  L^DBG_DEF               ;ADDRESS OF FILE DEFAULTS               [14]
               08 A2    7F     0EA1  2315          PUSHAQ  CLI$Q_FILE(R2)          ;ADDRESS OF FILE DESCRIPTOR             [14]
   00000000'EF    07    FB     0EA4  2316          CALLS   #7,L^DSX$LOAD           ;LOAD THE DEBUGGER                      [14]
         00000000'EF    16     0EAB  2317          JSB     DSR$COMPLETION          ;TYPE ERROR TEXT IF NECESSARY           [14]
               09 50    E8     0EB1  2318          BLBS    R0,25$                  ;IF ERROR THEN                          [14]
         00000000'EF    16     0EB4  2319          JSB     UNMAP_DEBUGGER          ;   UNMAP MEMORY WE JUST ALLOC'D        [14]
                006F    31     0EBA  2320          BRW     DSV$DEBUG_EXIT          ;   LEAVE                               [14]
                                0EBD  2321 25$:     $DS_PRINTF_S    FMT_DBG_LOAD,-  ;PRINT MESSAGE                          [14]
                                0EBD  2322                      DBG_ADDR_RTN    ;                                       [14]
         000001CD'EF    DD     0EBD                        PUSHL   DBG_ADDR_RTN
         00000179'EF    9F     0EC3                        PUSHAB  FMT_DBG_LOAD
   00000000'9F    02    FB     0EC9                        CALLS   #$$N, a#DS$PRINTF
         04000000 8F    C8     0ED0  2323          BISL    #1@DSA$V_DEBUG,-        ;SET DEBUGGER RESIDENT FLAG             [14]
         0000FE00'EF           0ED6  2324                  DSA$GL_FLAGS            ;                                       [14]
                  0E    BB     0EDB  2325 27$:     PUSHR   #^M<R1,R2,R3>           ;SAVE REGISTERS                         [14]
   51    80000000 8F    D0     0EDD  2326          MOVL    #^X80000000,R1          ;SET DS_ENVIRON BIT                     [14]
                  1C    E1     0EE4  2327          BBC     #DSA$V_USER,-           ;IF USER MODE                           [14]
     07 0000FE00'EF           0EE6  2328                  DSA$GL_FLAGS,30$        ;THEN                                   [14]
   51    40000000 8F    C8     0EEC  2329          BISL    #1@30,R1                ;   SET USER MODE BIT                   [14]
                  51    DD     0EF3  2330 30$:     PUSHL   R1                      ;CLI STATUS BITS                        [14]
                  00    DD     0EF5  2331          PUSHL   #0                      ;NULL ARGUMENT                          [14]
         00000000'EF    DF     0EF7  2332          PUSHAL  DIAG_FILE_NAME          ;PASS ADDR. OF COUNTED STRING           [14]
         00000000'EF    DF     0EFD  2333          PUSHAL  IMAGE_HEADER            ;PASS ADDR. OF IMAGE HEADER             [14]
                  00    DD     0F03  2334 50$:     PUSHL   #0                      ;NULL ARGUMENT                          [14]
         00000F2D'EF    DF     0F05  2335          PUSHAL  TRANS_VECTOR            ;TRANSFER ADDRESS VECTOR                [14]
   51    000001CD'EF    D0     0F0B  2336          MOVL    DBG_ADDR_RTN,R1         ;PUT START OF DEBUGGER IMAGE            [14]
                                0F12  2337                                         ; HEADER IN R1                          [14]
       52    02 A1    9A     0F12  2338          MOVZBL  TRANS_VECTOR_OFFSET(R1),R2;GET OFFSET TO TRANSFER VECTOR        [14]
     53    52    51    C1     0F16  2339          ADDL3   R1,R2,R3                ;GET ADDRESS OF VECTOR                  [14]
           53    04    C0     0F1A  2340          ADDL2   #4,R3                   ;GET SECOND ENTRY                       [14]
           51    63    C0     0F1D  2341          ADDL2   (R3),R1                 ;MAKE TRANSFER ADDRESS ABSOLUTE         [14]
           61    06    FB     0F20  2342          CALLS   #6,(R1)                 ;CALL THE DEBUGGER                      [14]
                                0F23  2343 CONT_FROM_DEBUG:                        ;                                       [14]
                  0E    BA     0F23  2344          POPR    #^M<R1,R2,R3>           ;RESTORE REGISTERS                      [14]
   5E    00000058 8F    C0     0F25  2345          ADDL2   #<22*4>,SP              ;RESTORE STACK (DEBUGGER MESSED         [14]
                                0F2C  2346                                         ; IT UP).                               [14]
                                0F2C  2347 DSV$DEBUG_EXIT:                         ;                                       [14]
                        05      0F2C  2348          RSB                            ;RETURN                                 [14]
                                0F2D  2349 TRANS_VECTOR:                           ;USED BY DEBUGGER                       [14]
                   00000000     0F2D  2350                  .LONG   0              ;                                       [14]
                   00000F21'    0F31  2351                  .LONG   CONT_FROM_DEBUG-2;                                     [14]
                                0F35  2352
                                0F35  2353          .END
```

```
$$ARGS                          = 00000004     D
$$N                             = 00000002     D
$$T1                            = 00000014     D
$$T2                            = 00000007     D
$ER                             = 00000001     D
$MODULE                           00000000 R   D   03
$SRVCODE_BOOTATTACHED           = 00000000     D
$SRVCODE_CLEAR_INTERLOCK        = 00000009     D
$SRVCODE_GETSYSBUF              = 00000004     D
$SRVCODE_HALTATTACHED           = 00000002     D
$SRVCODE_INTERLOCK_TEST         = 00000007     D
$SRVCODE_PPSCB                  = 0000000A     D
$SRVCODE_PRINTREV               = 00000006     D
$SRVCODE_RELSYSBUF              = 00000005     D
$SRVCODE_SET_INTERLOCK          = 00000008     D
$SRVCODE_SHOWIDLE               = 00000003     D
$SRVCODE_STARTATTACHED          = 00000001     D
AAP                               00000323 R   D   03
ADR1                              FFFFFFD8     D
ADR2                              FFFFFFB8     D
AEM                               0000029F R   D   03
AFP                               00000326 R   D   03
AKM                               00000298 R   D   03
APC                               0000032C R   D   03
APSL                              0000032F R   D   03
APSLMNE                           00000243 RG  D   03
AR0                               000002FD R   D   03
AR1                               00000300 R   D   03
AR10                              0000031B R   D   03
AR11                              0000031F R   D   03
AR2                               00000303 R   D   03
AR3                               00000306 R   D   03
AR4                               00000309 R   D   03
AR5                               0000030C R   D   03
AR6                               0000030F R   D   03
AR7                               00000312 R   D   03
AR8                               00000315 R   D   03
AR9                               00000318 R   D   03
ARG1                              FFFFFFE0     D
ARG2                              FFFFFFC0     D
ASM                               000002A9 R   D   03
ASP                               00000329 R   D   03
AUM                               000002B4 R   D   03
BASEADDRESS                       0000009C R   D   02
BIT...                          = 0000000F     D
BPTCODE                         = 00000003   G D
BPTMAX                          = 0000000F   G D
BPT_LITERAL                       00000006 R   D   03
CALL_CLI                          00000901 R   D   04
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_SIZE                        0000044C     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_FLAGS2                      00000444     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEMP_BASE                   00000448     D
CLI$L_TEST                        00000020     D
CLI$M_START_OR_RUN              = 00000008     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BASE_QUAL                 = 00000002     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NEXT                      = 00000001     D
CLI$V_NOALLOC                   = 00000000     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_START_OR_RUN              = 00000003     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
CMKSSGIPR                         00000D7A RG  D   04
CMK$_APBOOT                     = 00000004     D
CMK$_ASTEXIT                    = 00000000     D
CMK$_CANTIM                     = 0000000B     D
CMK$_HIBER                      = 00000007     D
CMK$_INITSCB                    = 00000008     D
CMK$_LAST                       = 0000000E     D
CMK$_MMENABLE                   = 00000003     D
CMK$_RCONSOLE                   = 00000001     D
```

```
CMK$_REFRESH                    = 00000005    D
CMK$_SCHDWK                     = 00000006    D
CMK$_SETIMR                     = 0000000C    D
CMK$_SETIPL                     = 00000009    D
CMK$_SETPRT                     = 0000000D    D
CMK$_SGIPR                      = 0000000A    D
CMK$_TCONSOLE                   = 00000002    D
COND_DEBUG                        000007AB RG D   04
COND_DEBUG_X                      000007EC R  D   04
CONT                              00000934 R  D   04
CONT_FROM_DEBUG                   00000F23 R  D   04
DATA                              FFFFFFF8    D
DBG_ADDR_REQ                      000001C5 R  D   02
DBG_ADDR_RTN                      000001CD RG D   02
DBG_BREAK                         0000081F R  D   04
DBG_DEF                           00000DB3 R  D   04
DBG_TBIT                          000009FB R  D   04
DEBUG$GB_FLAGS                    00000000 RG D   02
DEBUGGER_SIZE                   = 00010E00    D
DEBUG_HI                          000001D9 RG D   02
DEBUG_LO                          000001D5 RG D   02
DFLTSIZE                          00000098 R  D   02
DIAG_FILE_NAME                    ********   GX   00
DIR...                          = FFFFFFFF    D
DS$AA_BPTADDR                     00000014 RG D   02
DS$AA_BPTBASE                     00000054 RG D   02
DS$AA_OFFSET                      00000094 R  D   02
DS$AB_BPTINST                     00000001 RG D   02
DS$CLI                            ********    X   04
DS$CVTREG                         ********    X   04
DS$GB_PRIMARY_CPU_IDENTIFIER      ********    X   04
DS$GL_CLIBASE                     ********    X   04
DS$GL_IPR_VALUE                   ********    X   00
DS$GL_MEMSIZE                     ********    X   04
DS$GL_RADIX                       ********    X   04
DS$K_ERROR                      = 00000002    D
DS$K_NORMAL                     = 00000001    D
DS$K_SEVERE                     = 00000004    D
DS$K_SUBSYS                     = 00000066    D
DS$K_WARNING                    = 00000000    D
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_NI                         = 04000000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$PRINTF                         ********    X   04
DS$SERVICE_DISPATCHER             ********    X   04
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DS$_AP_NORMAL_BREAK             = 00660150    D
DS$_ARITH                       = 006600D0    D
DS$_ASBE                        = 00660118    D
DS$_BADLINK                     = 006600F0    D
DS$_BADTYPE                     = 006600E8    D
DS$_BIIC                        = 00660120    D
DS$_CHME                        = 006600A8    D
DS$_CHMK                        = 006600E0    D
DS$_DEVNAME                     = 00660108    D
DS$_ERROR                       = 00660002    D
DS$_FHWE                        = 00660068    D
DS$_FRAGBUF                     = 00660080    D
DS$_ICBUSY                      = 006600C8    D
DS$_ICERR                       = 006600C0    D
```

DS$_IHWE	=	00660060	D		DSX$LOAD		********		X	04	
DS$_ILLCHAR	=	00660018	D		DSX$PRINTI		********		X	04	
DS$_ILLPAGCNT	=	00660078	D		EXAMBUFFER		000000DB	R	D	02	
DS$_ILLUNIT	=	00660100	D		EXAMQWDBUF		000000D3	R	D	02	
DS$_INIT_FAIL	=	00660140	D		EXAMSIZE	=	00000020		D		
DS$_INSFMEM	=	00660050	D		FAOARG		0000015F	R	D	02	
DS$_INVCPU	=	00660130	D		FAOL$_CTRSTR	=	00000004		D		
DS$_IPL2HI	=	006600B8	D		FAOL$_NARGS	=	00000004		D		
DS$_IVADDR	=	00660040	D		FAOL$_OUTBUF	=	0000000C		D		
DS$_IVVECT	=	00660038	D		FAOL$_OUTLEN	=	00000008		D		
DS$_KRNLSTK	=	00660090	D		FAOL$_PRMLST	=	00000010		D		
DS$_LOGIC	=	00660070	D		FAOLST		0000014B	R	D	02	
DS$_MCHK	=	00660088	D		FETCH_STORE		00000B83	RG	D	04	
DS$_MEM_ALLOC_ERR	=	00660138	D		FETCH_STOR_PREG		00000D90	RG	D	04	
DS$_MMOFF	=	00660058	D		FIND_BPT		00000794	R	D	04	
DS$_NEEDUNIT	=	006600F8	D		FMTBPT		00000085	RG	D	03	
DS$_NODE	=	00660128	D		FMTBPTADR		000000C6	R	D	03	
DS$_NOPCS	=	00660110	D		FMTBPTLST		000000F0	R	D	03	
DS$_NORMAL	=	00660001	D		FMTBPTNONE		0000010A	R	D	03	
DS$_NOSUPPORT	=	006600B1	D		FMTBPTNOTSET		0000011F	R	D	03	
DS$_NOTALLOWMP	=	00660148	D		FMTDIRECTIVE		000000C8	R	D	02	
DS$_NOTDON	=	00660030	D		FMTEXAM		00000174	R	D	03	
DS$_NOTIMP	=	006600B0	D		FMTEXAM_ASCII		00000161	R	D	03	
DS$_NULLSTR	=	00660010	D		FMTINVLDREG		0000013A	R	D	03	
DS$_OVERFLOW	=	00660008	D		FMTNOBPTSLEFT		00000182	R	D	03	
DS$_POWER	=	00660098	D		FMTSTEP		000000BB	R	D	03	
DS$_PROGERR	=	00660020	D		FMT_BADPREG		0000022D	R	D	03	
DS$_SEVERE	=	00660004	D		FMT_BASE_OUT		00000007	R	D	03	
DS$_TRANSL	=	006600A0	D		FMT_BYTE		00000080	R	D	03	
DS$_TRUNCATE	=	00660028	D		FMT_DBG_LOAD		00000179	R	D	02	
DS$_UNEXPINT	=	006600D8	D		FMT_DBG_MSG		00000163	R	D	02	
DS$_UNSUPPDEV	=	00660158	D		FMT_DEC		00000064	R	D	03	
DS$_VASFULL	=	00660048	D		FMT_DFLT_OUT		0000002D	R	D	03	
DS$_WARNING	=	00660000	D		FMT_EXCEPTION		000001F9	R	D	03	
DSA$AL_APTMAIL		0000FE00	D		FMT_HEX		00000058	R	D	03	
DSA$AT_APTTXT		0000FA00	D		FMT_LONG		00000072	R	D	03	
DSA$GL_APTCOM		0000FE04	D		FMT_NO_ROOM		00000192	R	D	02	
DSA$GL_DEVLEN		0000FE58	D		FMT_OCT		0000006C	R	D	03	
DSA$GL_ERRNO		0000FE44	D		FMT_WORD		0000007B	R	D	03	
DSA$GL_EVENT		0000FE48	D		FS_COPY		00000BDA	R	D	04	
DSA$GL_FLAGS		0000FE00	D		FS_HANDLR		00000BFF	R	D	04	
DSA$GL_MSGTYP		0000FE40	D		FS_RESET		00000D51	R	D	04	
DSA$GL_PASSES		0000FE08	D		FS_SETUP		00000CBB	R	D	04	
DSA$GL_PASSNO		0000FE54	D		HTAB	=	00000009		D		
DSA$GL_SECTNO		0000FE10	D		ID_TEMP		000000C4	R	D	02	
DSA$GL_SID		0000FE14	D		IMAGE_HEADER		********		X	04	
DSA$GL_SUBTNO		0000FE4C	D		INITBASE	=	00000000		D		
DSA$GL_TESTNO		0000FE50	D		INITSIZE	=	00000004		D		
DSA$GL_UNITS		0000FE0C	D		LOCAL		FFFFFFB8		D		
DSA$GQ_MSGPTR		0000FE68	D		MAP_DEBUGGER		********		X	04	
DSA$GT_DEVNAM		0000FE5C	D		MODELST		00000284	RG	D	03	
DSA$V_DEBUG	=	0000001A	D		MP$GB_FUNCTION		000001E1	RG	D	02	
DSA$V_USER	=	0000001C	D		MP$GB_SETCPU		********		X	00	
DSP$REMOVEBREAK		000006F7	RG	D	04	MP$GL_AP_BASE_ADDR_BKPT	********W	GX		00	
DSR$COMPLETION		********		X	04	MP$GL_AP_BKPT_STORE	********W	GX		00	
DSV$DEBUG		00000DBF	RG	D	04	MP$GL_AP_OFFSET_BKPT	********W	GX		00	
DSV$DEBUG_EXIT		00000F2C	R	D	04	MP$GL_IPR_STATUS	000001E2	RG	D	02	

```
MP$GL_IPR_VALUE                 000001E6 RG D  02
MP$GL_REG_ADDRESS               000001EA RG D  02
MP$GL_REG_VALUE                 000001EE RG D  02
MP$GL_RET_STATUS                000001F2 RG D  02
MP$GL_SAVED_R1                  000001F6 RG D  02
MP$GR_BOOTED_PROCESSORS         ********   X   00
MP$GR_SAVED_GPRS                ********W GX   00
MP$GT_DEVICE_NAME               ********   X   00
MP$V_IL_ALLN                  = 00000004     D
MP$V_IL_CNDB                  = 00000007     D
MP$V_IL_CNHN                  = 00000006     D
MP$V_IL_DELN                  = 00000005     D
MP$V_IL_DSBK                  = 00000008     D
MP$V_IL_ERSP                  = 00000003     D
MP$V_IL_GTLN                  = 00000009     D
MP$V_IL_NUMT                  = 00000000     D
MP$V_IL_PRNT                  = 00000001     D
MP$V_IL_RRPT                  = 00000002     D
MP$_COND_DEBUG                  ********W GX   00
MP$_GET_GPR                     ********W GX   00
MP$_GET_IPR                     ********W GX   00
MP$_GET_PROCESSOR_ID            ********W GX   00
MP$_RESUME_ATTACHED_PROCESSORS  ********W GX   00
M_STEP                        = 00000002     D
M_TBIT                        = 00000001     D
NEW_AP_FP                       000000B0 R  D  02
NEW_PC_PSL                      000000BC R  D  02
NEW_SP                          000000B8 R  D  02
PRNT_AP_NAME                    00000159 RG D  03
PRT$C_UR                        ********   X   04
PRT$C_UW                        ********   X   04
PSL$C_KERNEL                  = 00000000     D
PSL$M_TBIT                    = 00000010     D
PSL$M_TP                      = 40000000     D
PSL$V_IS                      = 0000001A     D
PSLBUFFER                       000000FB R  D  02
PSLBUFSZ                      = 00000050     D
PSLREG                        = 00000010     D
QUICK_X                         00000B1E R  D  04
RBREAK_IN                       00000B35 R  D  04
RBREAK_OUT                      00000B0A R  D  04
REGNAMLST                       000002B9 R  D  03
RESTORE_SCB_VECTORS             00000A72 R  D  04
RET_LEN                         000001FE R  D  02
RET_REC                         000001FA R  D  02
SAVABS...                     = FFFFFFB8     D
SAVE_SCB_VEC                    000000A0 R  D  02
SAVE_SCB_VECTORS                00000AA8 R  D  04
SCB_BASE                        ********    X  04
SCB_IMAGE                       ********    X  04
SCRIPT$STOP                     ********    X  04
SETPRT$_ACMODE                = 0000000C     D
SETPRT$_INADR                 = 00000004     D
SETPRT$_NARGS                 = 00000005     D
SETPRT$_PROT                  = 00000010     D
SETPRT$_PRVPRT                = 00000014     D
SETPRT$_RETADR                = 00000008     D
SIZ...                        = 00000001     D
SS$_ACCVIO                      ********    X  04
SS$_BREAK                       ********    X  04
SS$_CONTINUE                    ********    X  04
SS$_NORMAL                    = 00000001     D
SS$_RESIGNAL                    ********    X  04
SS$_ROPRAND                     ********    X  04
SS$_TBIT                        ********    X  04
SYMTAB_HI                       000001DD RG D  02
SYS$CRETVA                      ********  GX   04
SYS$FAO                         ********   X   04
SYS$FAOL                        ********  GX   04
SYS$SETPRT                      ********   X   04
SYS$UNWIND                      ********  GX   04
TRANS_VECTOR                    00000F2D R  D  04
TRANS_VECTOR_OFFSET           = 00000002     D
T_ACCVIO                        000001CC R  D  03
T_EMESSG1                       ********    X  04
T_MACHCK                        000001AB R  D  03
T_PREGREAD                      0000021A R  D  03
T_PREGWRITE                     00000224 R  D  03
T_TRANSL                        000001E0 R  D  03
UNMAP_DEBUGGER                  ********    X  04
VRCLRBRK                        0000068A RG D  04
VRCLRBRKX                       000006F1 R  D  04
VRDEPOSIT                       00000470 RG D  04
VRDEPOSITX                      00000606 R  D  04
VREXAMINE                       000000CA RG D  04
VREXAMINEX                      00000261 R  D  04
VRSETBASE                       00000000 RG D  04
VRSETBRK                        0000060B RG D  04
VRSETBRKX                       00000684 R  D  04
VRSETDFLT                       0000001D RG D  04
VRSHOWBASE                      00000009 RG D  04
VRSHOWBRK                       0000072A RG D  04
VRSHOWBRKX                      00000791 R  D  04
VRSHOWDFLT                      00000060 RG D  04
VRTYPEXAMINE                    00000269 R  D  04
V_STEP                        = 00000001     D
V_TBIT                        = 00000000     D
```

```
                                    +-----------------+
                                    ! Psect synopsis !
                                    +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           FFFFFFF8  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
WORK                            00000202  (  514.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
DATA                            00000333  (  819.)  03 (  3.)  NOPIC   USR   CON   REL   LCL   SHR NOEXE   RD  NOWRT NOVEC BYTE
CODE                            00000F35  ( 3893.)  04 (  4.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000004	293 (3)	234 (2) 293 (3)
$$N	=00000002	2322 (42)	#-1038 (21) 1140 (22) 1239 (24) #-1297 (26)
			#-1403 (28) 1408 (28) #-1414 (28) 1624 (32)
			2108 (41) #-2282 (42) #-2306 (42) 2322 (42)
			511 (8) 659 (12) #-722 (14) 817 (15)
			975 (18)
$$T1	=00000014	293 (3)	234 (2) 293 (3)
$$T2	=00000007	960 (18)	960 (18)
$ER	=00000001	328 (4)	
$MODULE	00000000-R	328 (4)	
$SRVCODE_BOOTATTACHED	=00000000	1529 (30)	
$SRVCODE_CLEAR_INTERLOCK	=00000009	1529 (30)	#-1529 (30)
$SRVCODE_GETSYSBUF	=00000004	1529 (30)	
$SRVCODE_HALTATTACHED	=00000002	1529 (30)	
$SRVCODE_INTERLOCK_TEST	=00000007	1529 (30)	
$SRVCODE_PPSCB	=0000000A	1529 (30)	
$SRVCODE_PRINTREV	=00000006	1529 (30)	
$SRVCODE_RELSYSBUF	=00000005	1529 (30)	
$SRVCODE_SET_INTERLOCK	=00000008	1529 (30)	#-1498 (30) #-1506 (30)
$SRVCODE_SHOWIDLE	=00000003	1529 (30)	
$SRVCODE_STARTATTACHED	=00000001	1529 (30)	
AAP	00000323-R	423 (6)	405 (6)
ADR1	FFFFFFD8	2003 (39)	#-2126 (42) #-2128 (42) #-2129 (42) #-2132 (42)
			#-2145 (42) 2149 (42)
ADR2	FFFFFFB8	2003 (39)	#-2132 (42) #-2133 (42) 2137 (42)
AEM	0000029F-R	388 (5)	383 (5)
AFP	00000326-R	424 (6)	406 (6)
AKM	00000298-R	387 (5)	382 (5)
APC	0000032C-R	426 (6)	408 (6)
APSL	0000032F-R	427 (6)	409 (6)
APSLMNE	00000243-R	377 (5)	945 (17)
AR0	000002FD-R	411 (6)	393 (6)
AR1	00000300-R	412 (6)	394 (6)
AR10	0000031B-R	421 (6)	403 (6)
AR11	0000031F-R	422 (6)	404 (6)
AR2	00000303-R	413 (6)	395 (6)
AR3	00000306-R	414 (6)	396 (6)
AR4	00000309-R	415 (6)	397 (6)
AR5	0000030C-R	416 (6)	398 (6)
AR6	0000030F-R	417 (6)	399 (6)
AR7	00000312-R	418 (6)	400 (6)
AR8	00000315-R	419 (6)	401 (6)
AR9	00000318-R	420 (6)	402 (6)
ARG1	FFFFFFE0	2003 (39)	#-2009 (40) #-2123 (42) 2146 (42) #-2147 (42)
			#-2148 (42) #-2150 (42) #-2151 (42) 2153 (42)
			#-2155 (42) 2173 (42) 2176 (42)
ARG2	FFFFFFC0	2003 (39)	#-2010 (40) #-2124 (42) 2134 (42) #-2135 (42)
			#-2136 (42) #-2138 (42) #-2139 (42) 2141 (42)
			#-2143 (42) 2178 (42) 2181 (42)
ASM	000002A9-R	389 (5)	384 (5)

```
ASP                             00000329-R      425     (6)       407     (6)
AUM                             000002B4-R      390     (5)       385     (5)
BASEADDRESS                     0000009C-R      269     (3)     #-1060    (21)    #-1213    (24)    #-1214    (24)    #-1286    (26)
                                                                #-469     (7)     #-511     (8)     #-745     (14)
BIT...                          =0000000F       236     (2)       230     (2)       231     (2)       236     (2)
BPTCODE                         =00000003       214     (2)       329     (4)
BPTMAX                          =0000000F       216     (2)     #-1239    (24)    #-1301    (26)    #-1354    (27)    #-1398    (28)
                                                                #-1450    (29)    #-1897    (37)    #-1942    (38)      258     (3)
                                                                  261     (3)       263     (3)
BPT_LITERAL                     00000006-R      329     (4)      1953     (38)
CALL_CLI                        00000901-R      1627    (32)    #-1575    (32)    #-1747    (34)
CLI$K_BUFSIZ                    =00000100       229     (2)       229     (2)
CLI$K_SIZE                      0000044C        229     (2)
CLI$L_ADDRESS                   00000018        229     (2)     #-1027    (21)    #-1152    (22)    #-1156    (22)    #-1159    (22)
                                                                #-1205    (24)    #-1235    (24)    #-1284    (26)    #-1287    (26)
                                                                #-469     (7)     #-711     (14)    #-754     (14)    #-830     (15)
                                                                #-834     (15)    #-836     (15)    #-838     (15)    #-887     (16)
                                                                #-942     (17)
CLI$L_COMMAND                   00000004        229     (2)
CLI$L_DATA                      0000001C        229     (2)      1090     (22)    #-1101    (22)    #-1279    (26)    #-1746    (34)
CLI$L_FLAGS                     00000000        229     (2)      1034     (21)      1056    (21)      1068    (21)      1070    (21)
                                                                1072     (21)      1082    (22)      1135    (22)      1151    (22)
                                                                1155     (22)      1163    (22)      1204    (24)       560    (10)
                                                                 562     (10)       564    (10)       573    (10)       575    (10)
                                                                 577     (10)       718    (14)       740    (14)       752    (14)
                                                                 753     (14)       758    (14)       760    (14)       768    (15)
                                                                 815     (15)       829    (15)       833    (15)       862    (15)
                                                                 864     (15)       866    (15)       888    (16)       892    (16)
                                                                 918     (17)       921    (17)       941    (17)
CLI$L_FLAGS2                    00000444        229     (2)      1057     (21)      1208    (24)      1282    (26)       742    (14)
CLI$L_LAST                      00000024        229     (2)
CLI$L_NEXT                      00000030        229     (2)     #-1148    (22)    #-1160    (22)    #-827     (15)    #-839     (15)
CLI$L_PASS                      0000002C        229     (2)
CLI$L_SUBT                      00000028        229     (2)
CLI$L_TEMP_BASE                 00000448        229     (2)     #-1058    (21)    #-1209    (24)    #-1210    (24)    #-1283    (26)
                                                                #-743     (14)
CLI$L_TEST                      00000020        229     (2)
CLI$M_START_OR_RUN              =00000008       229     (2)
CLI$Q_BUFQWD                    00000034        229     (2)
CLI$Q_FILE                      00000008        229     (2)      2315     (42)
CLI$Q_SECTION                   00000010        229     (2)
CLI$Q_TIME                      0000043C        229     (2)
CLI$T_BUFFER                    0000003C        229     (2)
CLI$V_ADAPTER                   =00000018       229     (2)
CLI$V_ADR                       =0000000B       229     (2)
CLI$V_ASCII                     =00000013       229     (2)     #-918     (17)
CLI$V_BASE_QUAL                 =00000002       229     (2)     #-1057    (21)    #-1207    (24)    #-1282    (26)    #-741     (14)
CLI$V_BREAK                     =0000000A       229     (2)
CLI$V_BRIEF                     =0000001B       229     (2)
CLI$V_BYTE                      =0000000D       229     (2)     #-1072    (21)    #-573     (10)    #-760     (14)
CLI$V_CLEAR                     =00000002       229     (2)
CLI$V_DEC                       =00000010       229     (2)     #-562     (10)    #-864     (15)
CLI$V_DEFAULT                   =0000000C       229     (2)
CLI$V_DEPOSIT                   =00000019       229     (2)
CLI$V_EVENT                     =00000008       229     (2)
CLI$V_EXAM                      =00000005       229     (2)
CLI$V_FLAGS                     =00000009       229     (2)
```

```
CLI$V_HEX                =00000012     229     (2)     #-564    (10)    #-862    (15)
CLI$V_KERNEL             =00000017     229     (2)     #-1203   (24)
CLI$V_LOAD               =00000006     229     (2)
CLI$V_LONG               =0000000F     229     (2)     #-1068   (21)    #-577    (10)    #-752    (14)
CLI$V_NEXT               =00000001     229     (2)
CLI$V_NOALLOC            =00000000     229     (2)
CLI$V_NOTNUF             =00000001     229     (2)
CLI$V_OCT                =00000011     229     (2)     #-560    (10)    #-866    (15)
CLI$V_PREG               =0000001A     229     (2)     #-1056   (21)    #-1082   (22)    #-1135   (22)    #-1155   (22)
                                                       #-1163   (22)    #-739    (14)    #-768    (15)    #-815    (15)
                                                       #-833    (15)    #-892    (16)    #-921    (17)
CLI$V_QA                 =00000007     229     (2)
CLI$V_QACKLOOPLOOPS      =0000001C     229     (2)
CLI$V_QAERRORPRINTS      =0000001B     229     (2)
CLI$V_QAMULTIPLEPASS     =0000001F     229     (2)
CLI$V_QASUBTESTLOOPS     =0000001E     229     (2)
CLI$V_QATESTLOOPS        =0000001D     229     (2)
CLI$V_REG                =00000014     229     (2)     #-1034   (21)    #-1151   (22)    #-718    (14)    #-753    (14)
                                                       #-829    (15)    #-888    (16)    #-941    (17)
CLI$V_REQUIRED           =00000000     229     (2)
CLI$V_RUN                =00000015     229     (2)
CLI$V_SET                =00000003     229     (2)
CLI$V_SHOW               =00000004     229     (2)
CLI$V_START_OR_RUN       =00000003     229     (2)       229    (2)
CLI$V_VALSEC             =00000016     229     (2)
CLI$V_WORD               =0000000E     229     (2)     #-1070   (21)    #-575    (10)    #-758    (14)
CMK$SGIPR                 00000D7A-R   2199    (42)      1122   (22)     807     (15)
CMK$_APBOOT              =00000004     236     (2)
CMK$_ASTEXIT             =00000000     236     (2)
CMK$_CANTIM              =0000000B     236     (2)
CMK$_HIBER               =00000007     236     (2)
CMK$_INITSCB             =00000008     236     (2)
CMK$_LAST                =0000000E     236     (2)
CMK$_MMENABLE            =00000003     236     (2)
CMK$_RCONSOLE            =00000001     236     (2)
CMK$_REFRESH             =00000005     236     (2)
CMK$_SCHDWK              =00000006     236     (2)
CMK$_SETIMR              =0000000C     236     (2)
CMK$_SETIPL              =00000009     236     (2)
CMK$_SETPRT              =0000000D     236     (2)
CMK$_SGIPR               =0000000A     236     (2)     #-1122   (22)    #-807    (15)
CMK$_TCONSOLE            =00000002     236     (2)
COND_DEBUG                000007AB-R   1492    (30)
COND_DEBUG_X              000007EC-R   1509    (30)    #-1500   (30)
CONT                      00000934-R   1647    (32)    #-1613   (32)    #-1644   (32)
CONT_FROM_DEBUG           00000F23-R   2343    (42)      2351   (42)
DATA                      FFFFFFF8     2003    (39)      2024   (40)      2035   (40)
DBG_ADDR_REQ              000001C5-R   305     (3)     #-2286   (42)    #-2288   (42)    #-2289   (42)      2291   (42)
DBG_ADDR_RTN              000001CD-R   307     (3)       2291   (42)    #-2292   (42)    #-2293   (42)    #-2297   (42)
                                                       #-2298   (42)    #-2302   (42)    #-2312   (42)    #-2322   (42)
                                                       #-2336   (42)
DBG_BREAK                 0000081F-R   1558    (32)    #-1499   (30)
DBG_DEF                   00000DB3-R   2277    (42)      2314   (42)
DBG_TBIT                  000009FB-R   1726    (34)    #-1507   (30)
DEBUG$GB_FLAGS            00000000-R   251     (3)     #-1676   (32)      1728   (34)      1730   (34)    #-1750   (34)
DEBUGGER_SIZE            =00010E00     224     (2)     #-2287   (42)    #-2295   (42)    #-2313   (42)
DEBUG_HI                  000001D9-R   310     (3)     #-2293   (42)    #-2301   (42)
```

```
DEBUG_LO                        000001D5-R    309    (3)    #-2292   (42)   #-2302   (42)   #-2303   (42)
DFLTSIZE                        00000098-R    266    (3)    #-1073   (21)   #-574    (10)   #-576    (10)   #-578    (10)
                                                            #-644    (12)   #-761    (14)
DIAG_FILE_NAME                  00000000-XR                   2332   (42)     243    (3)
DIR...                          =FFFFFFFF     2003   (39)    2003   (39)
DS$AA_BPTADDR                   00000014-R    260    (3)    #-1229   (24)   #-1236   (24)   #-1291   (26)   #-1302   (26)
                                                              1349   (27)   #-1399   (28)   #-1405   (28)   #-1408   (28)
                                                            #-1452   (29)   #-1568   (32)   #-1574   (32)   #-1585   (32)
                                                            #-1586   (32)   #-1621   (32)   #-1624   (32)   #-1671   (32)
                                                            #-1898   (37)   #-1901   (37)   #-1943   (38)   #-1947   (38)
                                                            #-1952   (38)   #-1958   (38)
DS$AA_BPTBASE                   00000054-R    262    (3)    #-1230   (24)   #-1292   (26)   #-1303   (26)     1350   (27)
                                                            #-1405   (28)   #-1408   (28)   #-1586   (32)   #-1588   (32)
                                                            #-1621   (32)   #-1624   (32)
DS$AA_OFFSET                    00000094-R    264    (3)    #-1405   (28)   #-1408   (28)   #-1586   (32)   #-1589   (32)
                                                            #-1621   (32)   #-1624   (32)
DS$AB_BPTINST                   00000001-R    257    (3)     1571   (32)     1902   (37)     1946   (38)
DS$CLI                          00000000-XR                   1637   (32)
DS$CVTREG                       00000000-XR                    945   (17)
DS$GB_PRIMARY_CPU_IDENTIFIER    00000000-XR                 #-1523   (30)   #-1600   (32)   #-1683   (32)
DS$GL_CLIBASE                   00000000-XR                   1202   (24)     1277   (26)     1745   (34)
DS$GL_IPR_VALUE                 00000000-XR                    194   (2)    #-2206   (42)
DS$GL_MEMSIZE                   00000000-XR                 #-2296   (42)   #-2300   (42)
DS$GL_RADIX                     00000000-XR                 #-561    (10)   #-563    (10)   #-565    (10)   #-633    (12)
                                                            #-869    (15)   #-872    (15)
DS$K_ERROR                      =00000002     231    (2)
DS$K_NORMAL                     =00000001     231    (2)
DS$K_SEVERE                     =00000004     231    (2)
DS$K_SUBSYS                     =00000066     231    (2)     231    (2)
DS$K_WARNING                    =00000000     231    (2)
DS$M_ABRTFLG                    =00000040     230    (2)
DS$M_BADTIME                    =00100000     230    (2)
DS$M_BATCH                      =00400000     230    (2)
DS$M_BRKCLR                     =00001000     230    (2)
DS$M_BRKPT                      =00000800     230    (2)
DS$M_CHARFLG                    =00000100     230    (2)
DS$M_CMDFLG                     =00000080     230    (2)
DS$M_CTRLC                      =00000001     230    (2)
DS$M_CTRLO                      =00010000     230    (2)
DS$M_DEVFLG                     =00000200     230    (2)
DS$M_DISABLCC                   =01000000     230    (2)
DS$M_DONFLG                     =00002000     230    (2)
DS$M_ERRFLG                     =00000010     230    (2)
DS$M_EXCEPT                     =00080000     230    (2)
DS$M_EXETST                     =00040000     230    (2)
DS$M_HLTFLG                     =00000008     230    (2)
DS$M_LODFLG                     =00000002     230    (2)
DS$M_MEMMGT                     =00008000     230    (2)
DS$M_NI                         =04000000     230    (2)
DS$M_OUTPUT                     =00800000     230    (2)
DS$M_RUBFLG                     =00000020     230    (2)
DS$M_SCRIPT                     =00200000     230    (2)
DS$M_SETIMR                     =02000000     230    (2)
DS$M_STRFLG                     =00000004     230    (2)
DS$M_SUBT                       =00004000     230    (2)
DS$M_SYSFLG                     =00000400     230    (2)
DS$M_TIMED                      =02000000     230    (2)
```

```
DS$M_TIMED_OUT                  =08000000     230     (2)
DS$M_TIMRON                     =00020000     230     (2)
DS$PRINTF                        00000000-XR                    1038    (21)    1104    (22)    1239    (24)    1297    (26)
                                                                1403    (28)    1408    (28)    1414    (28)    1624    (32)
                                                                1743    (34)    2282    (42)    2306    (42)    2322    (42)
                                                                511     (8)     659     (12)    722     (14)    788     (15)
                                                                917     (17)    975     (18)
DS$SERVICE_DISPATCHER            00000000-XR                    1498    (30)    1506    (30)    1529    (30)
DS$V_ABRTFLG                    =00000006     230     (2)
DS$V_BADTIME                    =00000014     230     (2)
DS$V_BATCH                      =00000016     230     (2)
DS$V_BRKCLR                     =0000000C     230     (2)
DS$V_BRKPT                      =0000000B     230     (2)
DS$V_CHARFLG                    =00000008     230     (2)
DS$V_CMDFLG                     =00000007     230     (2)
DS$V_CTRLC                      =00000000     230     (2)
DS$V_CTRLO                      =00000010     230     (2)
DS$V_DEVFLG                     =00000009     230     (2)
DS$V_DISABLCC                   =00000018     230     (2)
DS$V_DONFLG                     =0000000D     230     (2)
DS$V_ERRFLG                     =00000004     230     (2)
DS$V_EXCEPT                     =00000013     230     (2)
DS$V_EXETST                     =00000012     230     (2)
DS$V_HLTFLG                     =00000003     230     (2)
DS$V_LODFLG                     =00000001     230     (2)
DS$V_MEMMGT                     =0000000F     230     (2)
DS$V_NI                         =0000001A     230     (2)
DS$V_OUTPUT                     =00000017     230     (2)
DS$V_RUBFLG                     =00000005     230     (2)
DS$V_SCRIPT                     =00000015     230     (2)
DS$V_SETIMR                     =00000019     230     (2)
DS$V_STRFLG                     =00000002     230     (2)
DS$V_SUBT                       =0000000E     230     (2)
DS$V_SYSFLG                     =0000000A     230     (2)
DS$V_TIMED                      =00000019     230     (2)
DS$V_TIMED_OUT                  =0000001B     230     (2)
DS$V_TIMRON                     =00000011     230     (2)
DS$_AP_NORMAL_BREAK             =00660150     231     (2)
DS$_ARITH                       =006600D0     231     (2)
DS$_ASBE                        =00660118     231     (2)
DS$_BADLINK                     =006600F0     231     (2)
DS$_BADTYPE                     =006600E8     231     (2)
DS$_BIIC                        =00660120     231     (2)
DS$_CHME                        =006600A8     231     (2)
DS$_CHMK                        =006600E0     231     (2)
DS$_DEVNAME                     =00660108     231     (2)
DS$_ERROR                       =00660002     231     (2)
DS$_FHWE                        =00660068     231     (2)
DS$_FRAGBUF                     =00660080     231     (2)
DS$_ICBUSY                      =006600C8     231     (2)
DS$_ICERR                       =006600C0     231     (2)
DS$_IHWE                        =00660060     231     (2)
DS$_ILLCHAR                     =00660018     231     (2)
DS$_ILLPAGCNT                   =00660078     231     (2)
DS$_ILLUNIT                     =00660100     231     (2)
DS$_INIT_FAIL                   =00660140     231     (2)
DS$_INSFMEM                     =00660050     231     (2)
```

```
DS$_INVCPU                      =00660130     231     (2)
DS$_IPL2HI                      =006600B8     231     (2)
DS$_IVADDR                      =00660040     231     (2)
DS$_IVVECT                      =00660038     231     (2)
DS$_KRNLSTK                     =00660090     231     (2)
DS$_LOGIC                       =00660070     231     (2)
DS$_MCHK                        =00660088     231     (2)     #-2084    (41)
DS$_MEM_ALLOC_ERR               =00660138     231     (2)
DS$_MMOFF                       =00660058     231     (2)
DS$_NEEDUNIT                    =006600F8     231     (2)
DS$_NODE                        =00660128     231     (2)
DS$_NOPCS                       =00660110     231     (2)
DS$_NORMAL                      =00660001     231     (2)
DS$_NOSUPPORT                   =006600B1     231     (2)
DS$_NOTALLOWMP                  =00660148     231     (2)
DS$_NOTDON                      =00660030     231     (2)
DS$_NOTIMP                      =006600B0     231     (2)       231     (2)
DS$_NULLSTR                     =00660010     231     (2)
DS$_OVERFLOW                    =00660008     231     (2)
DS$_POWER                       =00660098     231     (2)
DS$_PROGERR                     =00660020     231     (2)
DS$_SEVERE                      =00660004     231     (2)
DS$_TRANSL                      =006600A0     231     (2)     #-2089    (41)
DS$_TRUNCATE                    =00660028     231     (2)
DS$_UNEXPINT                    =006600D8     231     (2)
DS$_UNSUPPDEV                   =00660158     231     (2)
DS$_VASFULL                     =00660048     231     (2)
DS$_WARNING                     =00660000     231     (2)       231     (2)
DSA$GL_FLAGS                     0000FE00                       1085    (22)    1093    (22)    1102    (22)    1119    (22)
                                                                1125    (22)    1793    (35)    1842    (36)    2074    (41)
                                                                2280    (42)    2284    (42)  #-2324    (42)    2328    (42)
                                                                770     (15)    780     (15)    786     (15)    804     (15)
                                                                810     (15)
DSA$V_DEBUG                     =0000001A                     #-2279    (42)  #-2323    (42)
DSA$V_USER                      =0000001C                     #-1085    (22)  #-1093    (22)  #-1102    (22)  #-1119    (22)
                                                              #-1125    (22)  #-1793    (35)  #-1842    (36)  #-2074    (41)
                                                              #-2283    (42)  #-2327    (42)  #-770     (15)  #-780     (15)
                                                              #-786     (15)  #-804     (15)  #-810     (15)
DSP$REMOVEBREAK                  000006F7-R   1346    (27)
DSR$COMPLETION                   00000000-XR                    2317    (42)
DSV$DEBUG                        00000DBF-R   2278    (42)
DSV$DEBUG_EXIT                   00000F2C-R   2347    (42)  #-2307    (42)  #-2320    (42)
DSX$LOAD                         00000000-XR                    2316    (42)
DSX$PRINTI                       00000000-XR                    1140    (22)    2108    (41)    2163    (42)     817    (15)
EXAMBUFFER                       000000DB-R    284    (3)      283    (3)      884    (16)     972    (18)
EXAMQWDBUF                       000000D3-R    282    (3)      293    (3)    #-883    (16)  #-884    (16)  #-893    (16)
                                                              #-894    (16)  #-895    (16)  #-900    (16)  #-901    (16)
                                                              #-902    (16)  #-904    (16)  #-905    (16)  #-906    (16)
                                                              #-933    (17)  #-934    (17)  #-935    (17)     960    (18)
                                                              #-964    (18)  #-965    (18)  #-971    (18)  #-972    (18)
                                                              #-975    (18)
EXAMSIZE                        =00000020     217     (2)      283    (3)      285    (3)    #-883    (16)  #-971    (18)
FAOARG                           0000015F-R   295     (3)      293    (3)    #-886    (16)  #-890    (16)  #-897    (16)
                                                              #-931    (17)
FAOL$_CTRSTR                    =00000004     293     (3)
FAOL$_NARGS                     =00000004     293     (3)
FAOL$_OUTBUF                    =0000000C     293     (3)
```

```
FAOL$_OUTLEN                    =00000008     293    (3)
FAOL$_PRMLST                    =00000010     293    (3)
FAOLST                           0000014B-R   289    (3)     899   (16)     932   (17)
FETCH_STORE                      00000B83-R   2005   (40)    1092   (22)    1573   (32)    1736   (34)    1904   (37)
                                                             1949   (38)    1955   (38)     778   (15)
FETCH_STOR_PREG                  00000D90-R   2244   (42)    2204   (42)
FIND_BPT                         00000794-R   1449   (29)  #-1218   (24)  #-1227   (24)  #-1289   (26)  #-1577   (32)
FMTBPT                           00000085-R   347    (4)     1624   (32)
FMTBPTADR                        000000C6-R   351    (4)     1408   (28)
FMTBPTLST                        000000F0-R   353    (4)     1403   (28)
FMTBPTNONE                       0000010A-R   355    (4)     1414   (28)
FMTBPTNOTSET                     0000011F-R   357    (4)     1297   (26)
FMTDIRECTIVE                     000000C8-R   278    (3)      293   (3)   #-861    (15)  #-876    (15)  #-889    (16)
                                                           #-896   (16)  #-929    (17)  #-930    (17)
FMTEXAM                          00000174-R   365    (4)     975   (18)
FMTEXAM_ASCII                    00000161-R   363    (4)     960   (18)
FMTINVLDREG                      0000013A-R   359    (4)     1038   (21)    722   (14)
FMTNOBPTSLEFT                    00000182-R   367    (4)     1239   (24)
FMTSTEP                          000000BB-R   349    (4)     1742   (34)
FMT_BADPREG                      0000022D-R   376    (5)     1140   (22)    817   (15)
FMT_BASE_OUT                     00000007-R   331    (4)      511   (8)
FMT_BYTE                         00000080-R   345    (4)      653   (12)
FMT_DBG_LOAD                     00000179-R   303    (3)     2322   (42)
FMT_DBG_MSG                      00000163-R   302    (3)     2282   (42)
FMT_DEC                          00000064-R   337    (4)      640   (12)
FMT_DFLT_OUT                     0000002D-R   333    (4)      659   (12)
FMT_EXCEPTION                    000001F9-R   373    (5)     2108   (41)    2162   (42)
FMT_HEX                          00000058-R   335    (4)      636   (12)
FMT_LONG                         00000072-R   341    (4)      647   (12)
FMT_NO_ROOM                      00000192-R   304    (3)     2306   (42)
FMT_OCT                          0000006C-R   339    (4)      642   (12)
FMT_WORD                         0000007B-R   343    (4)      651   (12)
FS_COPY                          00000BDA-R   2049   (41)  #-2025   (40)  #-2036   (40)
FS_HANDLR                        00000BFF-R   2068   (41)    2012   (40)    2245   (42)
FS_RESET                         00000D51-R   2172   (42)  #-2026   (40)  #-2037   (40)  #-2111   (41)  #-2165   (42)
FS_SETUP                         00000CBB-R   2121   (42)  #-2021   (40)  #-2033   (40)
HTAB                            =00000009     213    (2)
ID_TEMP                          000000C4-R   276    (3)
IMAGE_HEADER                     00000000-XR                2333   (42)
INITBASE                        =00000000     219    (2)     270   (3)
INITSIZE                        =00000004     220    (2)     267   (3)
LOCAL                            FFFFFFB8     2003   (39)    2008   (40)
MAP_DEBUGGER                     00000000-XR                2304   (42)
MODELST                          00000284-R   381    (5)      945   (17)
MP$GB_FUNCTION                   000001E1-R   313    (3)  #-1044   (21)  #-1110   (22)  #-727    (14)  #-794    (15)
MP$GB_SETCPU                     00000000-XR               #-1042   (21)  #-1107   (22)    195    (2)  #-725    (14)
                                                           #-791   (15)  #-913    (17)
MP$GL_AP_BASE_ADDR_BKPT          00000000-XR               #-1588   (32)    201   (2)
MP$GL_AP_BKPT_STORE              00000000-XR               #-1585   (32)    200   (2)
MP$GL_AP_OFFSET_BKPT             00000000-XR               #-1589   (32)    202   (2)
MP$GL_IPR_STATUS                 000001E2-R   314    (3)  #-1115   (22)  #-800    (15)
MP$GL_IPR_VALUE                  000001E6-R   315    (3)  #-1113   (22)  #-797    (15)
MP$GL_REG_ADDRESS                000001EA-R   316    (3)  #-1050   (21)
MP$GL_REG_VALUE                  000001EE-R   317    (3)     733   (14)
MP$GL_RET_STATUS                 000001F2-R   318    (3)  #-1114   (22)  #-799    (15)
MP$GL_SAVED_R1                   000001F6-R   319    (3)  #-1111   (22)  #-795    (15)
MP$GR_BOOTED_PROCESSORS          00000000-XR               #-1518   (30)  #-1583   (32)  #-1643   (32)  #-1678   (32)
```

```
                                                  197    (2)
MP$GR_SAVED_GPRS                00000000-XR      #-1691   (32)   #-1692   (32)   #-1693   (32)   #-1694   (32)
                                                 #-1695   (32)   #-1696   (32)   #-1697   (32)   #-1698   (32)
                                                 #-1699   (32)     206    (2)
MP$GT_DEVICE_NAME               00000000-XR        195    (2)     915    (17)
MP$V_IL_ALLN                   =00000004    189   (2)
MP$V_IL_CNDB                   =00000007    189   (2)  #-1498   (30)   #-1506   (30)   #-1529   (30)
MP$V_IL_CNHN                   =00000006    189   (2)
MP$V_IL_DELN                   =00000005    189   (2)
MP$V_IL_DSBK                   =00000008    189   (2)
MP$V_IL_ERSP                   =00000003    189   (2)
MP$V_IL_GTLN                   =00000009    189   (2)
MP$V_IL_NUMT                   =00000000    189   (2)
MP$V_IL_PRNT                   =00000001    189   (2)
MP$V_IL_RRPT                   =00000002    189   (2)
MP$_COND_DEBUG                  00000000-XR        1590   (32)     203    (2)
MP$_GET_GPR                     00000000-XR        1045   (21)     196    (2)      199    (2)      728    (14)
MP$_GET_IPR                     00000000-XR        1112   (22)     196    (2)      199    (2)      796    (15)
MP$_GET_PROCESSOR_ID            00000000-XR        1522   (30)     1599   (32)     1682   (32)     204    (2)
MP$_RESUME_ATTACHED_PROCESSORS  00000000-XR        1645   (32)     205    (2)
M_STEP                         =00000002    256   (3)  #-1750   (34)
M_TBIT                         =00000001    254   (3)  #-1750   (34)
NEW_AP_FP                       000000B0-R   272   (3)  #-1650   (32)   #-1672   (32)
NEW_PC_PSL                      000000BC-R   274   (3)  #-1653   (32)   #-1674   (32)
NEW_SP                          000000B8-R   273   (3)  #-1651   (32)   #-1673   (32)
PRNT_AP_NAME                    00000159-R   361   (4)    916    (17)
PRT$C_UR                        00000000-XR        #-2020   (40)
PRT$C_UW                        00000000-XR        #-2032   (40)
PSL$C_KERNEL                   =00000000          #-2018   (40)   #-2030   (40)
PSL$M_TBIT                     =00000010          #-1652   (32)   #-1675   (32)   #-1752   (34)   #-1757   (34)
PSL$M_TP                       =40000000          #-1652   (32)   #-1757   (34)
PSL$V_IS                       =0000001A    236   (2)  #-1122   (22)   #-807    (15)
PSLBUFFER                       000000FB-R   286   (3)  #-912    (17)     945    (17)   #-975    (18)
PSLBUFSZ                       =00000050    218   (2)    287    (3)    #-945    (17)
PSLREG                         =00000010    215   (2)  #-1152   (22)   #-754    (14)   #-830    (15)   #-942    (17)
QUICK_X                         0000081E-R   1531  (30)  #-1504   (30)   #-1527   (30)
RBREAK_IN                       00000B35-R   1939  (38)  #-1131   (22)   #-1242   (24)   #-1309   (26)   #-1366   (27)
                                                 #-1738   (34)   #-1755   (34)   #-847    (15)
RBREAK_OUT                      00000B0A-R   1895  (37)  #-1080   (22)   #-1201   (24)   #-1276   (26)   #-1348   (27)
                                                 #-1669   (32)   #-1749   (34)   #-708    (14)
REGNAMLST                       000002B9-R   392   (6)  #-890    (16)
RESTORE_SCB_VECTORS             00000A72-R   1792  (35)  #-1094   (22)   #-1126   (22)   #-781    (15)   #-811    (15)
RET_LEN                         000001FE-R   322   (3)    2311   (42)
RET_REC                         000001FA-R   321   (3)    2310   (42)
SAVABS...                      =FFFFFFB8    2003  (39)
SAVE_SCB_VEC                    000000A0-R   271   (3)    1799   (35)     1848   (36)     2076   (41)
SAVE_SCB_VECTORS                00000AA8-R   1841  (36)  #-1086   (22)   #-1120   (22)   #-771    (15)   #-805    (15)
SCB_BASE                        00000000-XR        #-1800   (35)   #-1801   (35)   #-1802   (35)   #-1803   (35)
                                                 #-1849   (36)   #-1850   (36)   #-1851   (36)   #-1852   (36)
                                                 #-1857   (36)   #-1859   (36)   #-1861   (36)   #-1863   (36)
                                                 #-2077   (41)   #-2078   (41)   #-2079   (41)   #-2080   (41)
SCB_IMAGE                       00000000-XR        #-1856   (36)   #-1858   (36)   #-1860   (36)   #-1862   (36)
SCRIPT$STOP                     00000000-XR        1635   (32)
SETPRT$_ACMODE                 =0000000C    234   (2)
SETPRT$_INADR                  =00000004    234   (2)  #-2138   (42)   #-2150   (42)
SETPRT$_NARGS                  =00000005    234   (2)  #-2139   (42)   #-2151   (42)
SETPRT$_PROT                   =00000010    234   (2)  #-2009   (40)   #-2010   (40)   #-2123   (42)   #-2124   (42)
```

```
                                                  2134   (42)   #-2143   (42)     2146   (42)   #-2155   (42)
                                                  2173   (42)     2178   (42)
SETPRT$_PRVPRT                =00000014     234   (2)   #-2135   (42)   #-2147   (42)
SETPRT$_RETADR                =00000008     234   (2)   #-2136   (42)   #-2148   (42)
SIZ...                        =00000001     230   (2)     230    (2)
SS$_ACCVIO                     00000000-XR                #-2094   (41)
SS$_BREAK                      00000000-XR                #-1496   (30)
SS$_CONTINUE                   00000000-XR                #-1657   (32)   #-1759   (34)
SS$_NORMAL                    =00000001     222   (2)   #-1457   (29)   #-2038   (40)
SS$_RESIGNAL                   00000000-XR                #-1494   (30)   #-1580   (32)   #-1727   (34)   #-2105   (41)
SS$_ROPRAND                    00000000-XR                #-2100   (41)
SS$_TBIT                       00000000-XR                #-1502   (30)
SYMTAB_HI                      000001DD-R    311   (3)   #-2303   (42)
SYS$CRETVA                     00000000-XR                  2291   (42)
SYS$FAO                        00000000-XR                  960    (18)
SYS$FAOL                       00000000-XR                  899    (16)     932    (17)
SYS$SETPRT                     00000000-XR                  2141   (42)     2153   (42)     2176   (42)     2181   (42)
SYS$UNWIND                     00000000-XR                  2112   (41)
TRANS_VECTOR                   00000F2D-R    2349  (42)     2335   (42)
TRANS_VECTOR_OFFSET           =00000002     2273  (42)   #-2338   (42)
T_ACCVIO                       000001CC-R    371   (5)     2097   (41)     2161   (42)
T_EMESSG1                      00000000-XR                  1103   (22)     787    (15)
T_MACHCK                       000001AB-R    370   (5)     2086   (41)
T_PREGREAD                     0000021A-R    374   (5)     1139   (22)     816    (15)
T_PREGWRITE                    00000224-R    375   (5)     1137   (22)
T_TRANSL                       000001E0-R    372   (5)     2092   (41)
UNMAP_DEBUGGER                 00000000-XR                  2319   (42)
VRCLRBRK                       0000068A-R    1274  (26)
VRCLRBRKX                      000006F1-R    1308  (26)   #-1295   (26)   #-1298   (26)
VRDEPOSIT                      00000470-R    1023  (21)
VRDEPOSITX                     00000606-R    1167  (22)   #-1039   (21)   #-1049   (21)   #-1105   (22)   #-1117   (22)
                                                        #-1135   (22)   #-1141   (22)   #-1164   (22)
VREXAMINE                      000000CA-R    706   (14)
VREXAMINEX                     00000261-R    846   (15)   #-723    (14)   #-732    (14)   #-789    (15)   #-802    (15)
                                                        #-818    (15)   #-828    (15)   #-831    (15)   #-835    (15)
VRSETBASE                      00000000-R    468   (7)
VRSETBRK                       0000060B-R    1198  (24)
VRSETBRKX                      00000684-R    1241  (24)   #-1224   (24)   #-1233   (24)   #-1237   (24)
VRSETDFLT                      0000001D-R    553   (10)
VRSHOWBASE                     00000009-R    509   (8)
VRSHOWBRK                      0000072A-R    1394  (28)
VRSHOWBRKX                     00000791-R    1416  (28)   #-1412   (28)
VRSHOWDFLT                     00000060-R    624   (12)
VRTYPEXAMINE                   00000269-R    858   (15)   #-1165   (22)   #-819    (15)
V_STEP                        =00000001     255   (3)   #-1730   (34)     256    (3)
V_TBIT                        =00000000     253   (3)   #-1676   (32)   #-1728   (34)     254    (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION		REFERENCES...							
$CRETVA_S	1	2290	(42)	2290	(42)						
$D1_PRINT_S	1	511	(8)	1038	(21)	1140	(22)	1239	(24)	1297	(26)
				1403	(28)	1408	(28)	1414	(28)	1624	(32)
				2108	(41)	2282	(42)	2306	(42)	2322	(42)
				511	(8)	659	(12)	722	(14)	817	(15)
				975	(18)						
$DEF	1	236	(2)								
$DEFINI	1	232	(2)	232	(2)	233	(2)	235	(2)		
$DS_CVTREG_S	1	944	(17)	944	(17)						
$DS_DSADEF	5	235	(2)	235	(2)						
$DS_DSDEF	3	231	(2)	231	(2)						
$DS_PRINTF_S	1	510	(8)	1038	(21)	1239	(24)	1297	(26)	1403	(28)
				1407	(28)	1414	(28)	1623	(32)	2282	(42)
				2306	(42)	2321	(42)	510	(8)	658	(12)
				722	(14)	973	(18)				
$DS_SRVCODE_DEF	2	1529	(30)	1498	(30)	1506	(30)	1529	(30)		
$EQU	1	236	(2)	231	(2)	236	(2)				
$EQULS1	1	236	(2)	231	(2)	236	(2)				
$EQULST	1	231	(2)	231	(2)	236	(2)				
$FAOL	1	290	(3)	290	(3)						
$FAOLDEF	1	293	(3)	293	(3)						
$FAOL_G	1	899	(16)	899	(16)	932	(17)				
$FAO_S	2	958	(18)	958	(18)						
$GBLINI	2	230	(2)	230	(2)	231	(2)	236	(2)		
$MP_CLEAR_INTERLOCK	1	1529	(30)	1529	(30)						
$MP_ILDEFS	1	189	(2)	189	(2)						
$MP_SET_INTERLOCK	1	1498	(30)	1498	(30)	1506	(30)				
$OFFDEF	1	234	(2)	234	(2)	293	(3)				
$OFFSET	2	1997	(39)	1997	(39)						
$OFFST1	1	2003	(39)	2003	(39)						
$PRDEF	5	233	(2)	233	(2)						
$PRINTI_S	1	817	(15)	1140	(22)	2108	(41)	817	(15)		
$PSLDEF	2	232	(2)	232	(2)						
$PUSHADR	1	945	(17)	2112	(41)	2291	(42)	945	(17)	960	(18)
$SETPRTDEF	1	234	(2)	234	(2)						
$UNWIND_S	1	2112	(41)	2112	(41)						
$VIELD	1	230	(2)	230	(2)						
$VIELD1	1	236	(2)	230	(2)						
CLIDEF	4	229	(2)	229	(2)						
CMK	1	807	(15)	1122	(22)	807	(15)				
CMKDEF	1	236	(2)	236	(2)						
DSFDEF	3	230	(2)	230	(2)						
MODNAM	1	328	(4)	328	(4)						

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL2	00C0	1058 (21) 1060 (21) 1159 (22) 1209 (24) 1213 (24) 1526 (30) 1528 (30)
		1602 (32) 1620 (32) 1685 (32) 1701 (32) 2340 (42) 2341 (42) 2345 (42)
		743 (14) 745 (14) 838 (15) 902 (16) 905 (16) 935 (17)
ADDL3	00C1	1283 (26) 1286 (26) 2339 (42)
ADDW2	00A0	965 (18)
ASHL	0078	954 (18)
BBC	00E1	1034 (21) 1057 (21) 1082 (22) 1102 (22) 1135 (22) 1151 (22) 1155 (22)
		1207 (24) 1282 (26) 2050 (41) 2053 (41) 2056 (41) 2279 (42) 2283 (42)
		2327 (42) 560 (10) 562 (10) 564 (10) 573 (10) 575 (10) 577 (10)
		718 (14) 741 (14) 753 (14) 768 (15) 786 (15) 815 (15) 829 (15)
		833 (15) 888 (16) 892 (16) 918 (17) 941 (17)
BBCC	00E5	1728 (34) 1730 (34) 2174 (42) 2179 (42)
BBS	00E0	1056 (21) 1068 (21) 1070 (21) 1072 (21) 1085 (22) 1093 (22) 1119 (22)
		1122 (22) 1125 (22) 1136 (22) 1163 (22) 1793 (35) 1842 (36) 2074 (41)
		739 (14) 752 (14) 758 (14) 760 (14) 770 (15) 780 (15) 804 (15)
		807 (15) 810 (15) 862 (15) 864 (15) 866 (15) 921 (17)
BBSC	00E4	1203 (24) 2102 (41)
BBSS	00E2	1402 (28) 2251 (42)
BEQL	0013	1043 (21) 1108 (22) 1149 (22) 1153 (22) 1280 (26) 1400 (28) 1453 (29)
		1503 (30) 1519 (30) 1584 (32) 1601 (32) 1644 (32) 1679 (32) 1684 (32)
		1899 (37) 1944 (38) 2130 (42) 726 (14) 755 (14) 792 (15) 828 (15)
		831 (15) 835 (15) 870 (15) 873 (15) 914 (17) 924 (17) 927 (17)
BGTRU	001A	1357 (27)
BICL	00CA	1757 (34)
BICL2	00CA	1652 (32)
BICL3	00CB	2126 (42) 2128 (42)
BISB	0088	1750 (34) 2143 (42) 2155 (42)
BISB2	0088	1676 (32)
BISL	00C8	1675 (32) 1752 (34) 2323 (42) 2329 (42)
BLBC	00E9	1115 (22) 1219 (24) 1228 (24) 1290 (26) 1950 (38) 2059 (41) 2142 (42)
		2154 (42) 800 (15)
BLBS	00E8	1046 (21) 1134 (22) 1412 (28) 1578 (32) 1956 (38) 2249 (42) 2305 (42)
		2318 (42) 729 (14) 813 (15)
BLEQU	001B	1036 (21) 1359 (27) 720 (14)
BNEQ	0012	1223 (24) 1497 (30) 1524 (30) 1569 (32) 2019 (40) 2031 (40) 2085 (41)
		2090 (41) 2095 (41) 2101 (41) 635 (12) 639 (12) 646 (12) 650 (12)
		943 (17)
BRB	0011	1051 (21) 1054 (21) 1059 (21) 1122 (22) 1138 (22) 1141 (22) 1154 (22)
		1157 (22) 1164 (22) 1212 (24) 1233 (24) 1237 (24) 1285 (26) 1295 (26)
		1298 (26) 1525 (30) 1527 (30) 1700 (32) 1753 (34) 2088 (41) 2093 (41)
		2098 (41) 2103 (41) 2294 (42) 637 (12) 641 (12) 648 (12) 652 (12)
		734 (14) 737 (14) 744 (14) 807 (15) 832 (15) 837 (15) 891 (16)
		946 (17)
BRW	0031	1039 (21) 1049 (21) 1083 (22) 1097 (22) 1105 (22) 1116 (22) 1117 (22)
		1161 (22) 1224 (24) 1500 (30) 1504 (30) 1575 (32) 1613 (32) 1747 (34)
		2281 (42) 2307 (42) 2320 (42) 723 (14) 732 (14) 769 (15) 783 (15)
		789 (15) 801 (15) 802 (15) 818 (15) 840 (15) 919 (17)
BSBB	0010	2036 (40)
BSBW	0030	1080 (22) 1086 (22) 1094 (22) 1120 (22) 1126 (22) 1131 (22) 1165 (22)
		1201 (24) 1218 (24) 1227 (24) 1242 (24) 1276 (26) 1289 (26) 1309 (26)

```
                    1348  (27)  1366  (27)  1499  (30)  1507  (30)  1577  (32)  1669  (32)  1738  (34)
                    1749  (34)  1755  (34)  2021  (40)  2025  (40)  2026  (40)  2033  (40)  2037  (40)
                    2111  (41)  2165  (42)  708   (14)  771   (15)  781   (15)  805   (15)  811   (15)
                    819   (15)  847   (15)
CALLG     00FA      1637  (32)  2141  (42)  2153  (42)  2176  (42)  2181  (42)  899   (16)  932   (17)
CALLS     00FB      1038  (21)  1092  (22)  1104  (22)  1140  (22)  1239  (24)  1297  (26)  1403  (28)
                    1408  (28)  1414  (28)  1498  (30)  1506  (30)  1522  (30)  1529  (30)  1573  (32)
                    1599  (32)  1624  (32)  1645  (32)  1682  (32)  1736  (34)  1743  (34)  1904  (37)
                    1949  (38)  1955  (38)  2108  (41)  2112  (41)  2163  (42)  2204  (42)  2282  (42)
                    2291  (42)  2306  (42)  2316  (42)  2322  (42)  2342  (42)  511   (8)   659   (12)
                    722   (14)  778   (15)  788   (15)  817   (15)  917   (17)  945   (17)  960   (18)
                    975   (18)
CHMK      00BC      1122  (22)  807   (15)
CLRB      0094      1044  (21)  1110  (22)  912   (17)
CLRL      00D4      1100  (22)  1122  (22)  1200  (24)  1226  (24)  1291  (26)  1292  (26)  1302  (26)
                    1303  (26)  1360  (27)  1361  (27)  1396  (28)  1455  (29)  1574  (32)  1671  (32)
                    1798  (35)  1847  (36)  1958  (38)  2009  (40)  2010  (40)  2075  (41)  773   (15)
                    807   (15)  963   (18)
CLRQ      007C      2136  (42)  2148  (42)
CMPB      0091      1523  (30)  1600  (32)  1683  (32)
CMPL      00D1      1035  (21)  1152  (22)  1279  (26)  1356  (27)  1358  (27)  1452  (29)  1496  (30)
                    1502  (30)  1568  (32)  2084  (41)  2089  (41)  2094  (41)  2100  (41)  2129  (42)
                    634   (12)  638   (12)  645   (12)  649   (12)  719   (14)  754   (14)  830   (15)
                    834   (15)  869   (15)  872   (15)  923   (17)  926   (17)  942   (17)
DECL      00D7      1160  (22)  839   (15)  895   (16)
INCL      00D6      1156  (22)  836   (15)  894   (16)
JSB       0016      1045  (21)  1112  (22)  1122  (22)  1590  (32)  1635  (32)  2304  (42)  2317  (42)
                    2319  (42)  728   (14)  796   (15)  807   (15)
MFPR      00DB      2252  (42)
MOVAB     009E      1137  (22)  1139  (22)  1349  (27)  1350  (27)  1655  (32)  2008  (40)  2012  (40)
                    2024  (40)  2035  (40)  2086  (41)  2092  (41)  2097  (41)  2127  (42)  2173  (42)
                    2178  (42)  2245  (42)  636   (12)  640   (12)  642   (12)  647   (12)  651   (12)
                    653   (12)  816   (15)  956   (18)
MOVAL     00DE      1053  (21)  1202  (24)  1277  (26)  1495  (30)  1565  (32)  1745  (34)  1756  (34)
                    1799  (35)  1848  (36)  2076  (41)  2134  (42)  2146  (42)  733   (14)  736   (14)
                    884   (16)  886   (16)  957   (18)  972   (18)
MOVAQ     007E      2137  (42)  2149  (42)
MOVB      0090      2061  (41)  2175  (42)  2180  (42)  727   (14)  794   (15)  861   (15)  863   (15)
                    865   (15)  868   (15)  871   (15)  874   (15)  876   (15)  893   (16)  896   (16)
                    922   (17)  925   (17)  928   (17)  929   (17)  930   (17)
MOVL      00D0      1027  (21)  1050  (21)  1067  (21)  1069  (21)  1071  (21)  1073  (21)  1099  (22)
                    1101  (22)  1109  (22)  1111  (22)  1113  (22)  1114  (22)  1123  (22)  1205  (24)
                    1210  (24)  1214  (24)  1216  (24)  1229  (24)  1230  (24)  1235  (24)  1301  (26)
                    1354  (27)  1398  (28)  1450  (29)  1457  (29)  1566  (32)  1585  (32)  1588  (32)
                    1589  (32)  1651  (32)  1657  (32)  1673  (32)  1698  (32)  1699  (32)  1732  (34)
                    1740  (34)  1751  (34)  1759  (34)  1800  (35)  1801  (35)  1802  (35)  1803  (35)
                    1849  (36)  1850  (36)  1851  (36)  1852  (36)  1856  (36)  1858  (36)  1860  (36)
                    1862  (36)  1897  (37)  1942  (38)  2007  (40)  2016  (40)  2020  (40)  2023  (40)
                    2028  (40)  2029  (40)  2032  (40)  2038  (40)  2055  (41)  2077  (41)  2078  (41)
                    2079  (41)  2080  (41)  2083  (41)  2087  (41)  2091  (41)  2096  (41)  2110  (41)
                    2132  (42)  2133  (42)  2139  (42)  2145  (42)  2151  (42)  2159  (42)  2166  (42)
                    2206  (42)  2246  (42)  2247  (42)  2248  (42)  2285  (42)  2292  (42)  2293  (42)
                    2302  (42)  2303  (42)  2326  (42)  2336  (42)  469   (7)   561   (10)  563   (10)
                    565   (10)  574   (10)  576   (10)  578   (10)  633   (12)  644   (12)  711   (14)
                    751   (14)  757   (14)  759   (14)  761   (14)  784   (15)  785   (15)  793   (15)
                    795   (15)  797   (15)  798   (15)  799   (15)  808   (15)  883   (16)  887   (16)
                    890   (16)  897   (16)  931   (17)
```

MOVPSL	00DC	1122	(22)	1654	(32)	807	(15)								
MOVQ	007D	1353	(27)	1607	(32)	1608	(32)	1610	(32)	1612	(32)	1628	(32)	1629	(32)
		1631	(32)	1633	(32)	1647	(32)	1648	(32)	1650	(32)	1653	(32)	1672	(32)
		1674	(32)	1691	(32)	1692	(32)	1693	(32)	1694	(32)	1695	(32)	1696	(32)
		1697	(32)	2015	(40)	2051	(41)	2052	(41)	2082	(41)	2109	(41)		
MOVW	00B0	2058	(41)	889	(16)	904	(16)								
MOVZBL	009A	2123	(42)	2124	(42)	2338	(42)								
MOVZWL	003C	1494	(30)	1580	(32)	1727	(34)	2105	(41)	2125	(42)	901	(16)	933	(17)
		962	(18)												
MTPR	00DA	2250	(42)												
POPL	8ED0	782	(15)	961	(18)										
POPR	00BA	1132	(22)	1168	(22)	1243	(24)	1310	(26)	1417	(28)	1649	(32)	1670	(32)
		1804	(35)	1864	(36)	1907	(37)	1961	(38)	2156	(42)	2205	(42)	2344	(42)
		660	(12)	848	(15)	976	(18)								
PROBER	000C	2018	(40)												
PROBEW	000D	2030	(40)												
PUSHAB	009F	1038	(21)	1089	(22)	1090	(22)	1103	(22)	1140	(22)	1239	(24)	1297	(26)
		1403	(28)	1408	(28)	1414	(28)	1571	(32)	1624	(32)	1742	(34)	1902	(37)
		1946	(38)	1953	(38)	2108	(41)	2161	(42)	2162	(42)	2282	(42)	2306	(42)
		2322	(42)	511	(8)	659	(12)	722	(14)	775	(15)	776	(15)	787	(15)
		817	(15)	915	(17)	916	(17)	945	(17)	975	(18)				
PUSHAL	00DF	1609	(32)	1630	(32)	1733	(34)	2201	(42)	2310	(42)	2311	(42)	2332	(42)
		2333	(42)	2335	(42)	945	(17)								
PUSHAQ	007F	2291	(42)	2314	(42)	2315	(42)	960	(18)						
PUSHAW	003F	960	(18)												
PUSHL	00DD	1091	(22)	1140	(22)	1239	(24)	1408	(28)	1498	(30)	1506	(30)	1521	(30)
		1529	(30)	1570	(32)	1572	(32)	1598	(32)	1624	(32)	1681	(32)	1734	(34)
		1735	(34)	1741	(34)	1901	(37)	1903	(37)	1947	(38)	1948	(38)	1952	(38)
		1954	(38)	2108	(41)	2112	(41)	2200	(42)	2202	(42)	2203	(42)	2291	(42)
		2309	(42)	2312	(42)	2313	(42)	2322	(42)	2330	(42)	2331	(42)	2334	(42)
		511	(8)	659	(12)	777	(15)	817	(15)	945	(17)	955	(18)	960	(18)
		975	(18)												
PUSHR	00BB	1024	(21)	1130	(22)	1199	(24)	1275	(26)	1395	(28)	1611	(32)	1632	(32)
		1668	(32)	1794	(35)	1843	(36)	1896	(37)	1940	(38)	2122	(42)	2325	(42)
		628	(12)	707	(14)	859	(15)								
REI	0002	1702	(32)	2207	(42)										
RET	0004	1367	(27)	1531	(30)	2039	(40)	2106	(41)	2113	(41)	2167	(42)	2253	(42)
RSB	0005	1169	(22)	1244	(24)	1311	(26)	1418	(28)	1456	(29)	1458	(29)	1581	(32)
		1658	(32)	1761	(34)	1806	(35)	1866	(36)	1908	(37)	1962	(38)	2062	(41)
		2157	(42)	2183	(42)	2348	(42)	470	(7)	512	(8)	585	(10)	661	(12)
		849	(15)	977	(18)										
SOBGEQ	00F4	1906	(37)	1959	(38)										
SOBGTR	00F5	1306	(26)	1364	(27)	1411	(28)	1454	(29)	1746	(34)				
SUBL2	00C2	1520	(30)	1597	(32)	1680	(32)	2298	(42)						
SUBL3	00C3	1405	(28)	1586	(32)	1621	(32)	2287	(42)	2295	(42)	2300	(42)		
SUBW2	00A2	900	(16)	906	(16)	934	(17)	964	(18)						
SUBW3	00A3	971	(18)												
TSTB	0095	1042	(21)	1107	(22)	2060	(41)	725	(14)	791	(15)	913	(17)		
TSTL	00D5	1148	(22)	1221	(24)	1399	(28)	1518	(30)	1583	(32)	1643	(32)	1678	(32)
		1898	(37)	1943	(38)	2054	(41)	827	(15)						
TSTW	00B5	2057	(41)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	293	(3)	382	(5)	383	(5)	384	(5)	385	(5)	393	(6)	394	(6)	395	(6)
	396	(6)	397	(6)	398	(6)	399	(6)	400	(6)	401	(6)	402	(6)	403	(6)
	404	(6)	405	(6)	406	(6)	407	(6)	408	(6)	409	(6)				
.ALIGN	259	(3)														
.ASCIC	302	(3)	303	(3)	304	(3)	328	(4)	332	(4)	334	(4)	336	(4)	338	(4)
	340	(4)	342	(4)	344	(4)	346	(4)	348	(4)	350	(4)	352	(4)	354	(4)
	356	(4)	358	(4)	360	(4)	362	(4)	366	(4)	368	(4)	370	(5)	371	(5)
	372	(5)	373	(5)	374	(5)	375	(5)	376	(5)	378	(5)	387	(5)	388	(5)
	389	(5)	390	(5)	411	(6)	412	(6)	413	(6)	414	(6)	415	(6)	416	(6)
	417	(6)	418	(6)	419	(6)	420	(6)	421	(6)	422	(6)	423	(6)	424	(6)
	425	(6)	426	(6)	427	(6)										
.ASCID	2277	(42)	364	(4)												
.ASCII	279	(3)														
.BLKB	2003	(39)	285	(3)	287	(3)										
.BLKL	229	(2)	271	(3)	273	(3)	296	(3)								
.BLKQ	272	(3)	274	(3)												
.BYTE	252	(3)	258	(3)	313	(3)	329	(4)								
.END	2353	(42)														
.ENDC	1038	(21)	1140	(22)	1239	(24)	1297	(26)	1403	(28)	1408	(28)	1414	(28)	1624	(32)
	2003	(39)	2108	(41)	2112	(41)	2282	(42)	2291	(42)	230	(2)	2306	(42)	231	(2)
	2322	(42)	236	(2)	511	(8)	659	(12)	722	(14)	817	(15)	945	(17)	960	(18)
	975	(18)														
.ENDM	1498	(30)	1506	(30)	1529	(30)	189	(2)	1997	(39)	2003	(39)	2112	(41)	229	(2)
	2290	(42)	230	(2)	231	(2)	232	(2)	233	(2)	234	(2)	235	(2)	236	(2)
	290	(3)	293	(3)	328	(4)	510	(8)	511	(8)	807	(15)	817	(15)	899	(16)
	944	(17)	945	(17)	958	(18)	960	(18)								
.ENDR	1038	(21)	1140	(22)	1239	(24)	1297	(26)	1403	(28)	1408	(28)	1414	(28)	1624	(32)
	2003	(39)	2108	(41)	2282	(42)	230	(2)	2306	(42)	231	(2)	2322	(42)	236	(2)
	511	(8)	659	(12)	722	(14)	817	(15)	975	(18)						
.ENTRY	1346	(27)	1492	(30)	2005	(40)	2244	(42)								
.EXTRN	194	(2)	195	(2)	196	(2)	197	(2)								
.GLOBAL	243	(3)														
.GLOBL	2112	(41)	2291	(42)	899	(16)	932	(17)								
.IDENT	13	(1)														
.IF	1038	(21)	1140	(22)	1239	(24)	1297	(26)	1403	(28)	1408	(28)	1414	(28)	1624	(32)
	2003	(39)	2108	(41)	2112	(41)	2282	(42)	2291	(42)	230	(2)	2306	(42)	231	(2)
	2322	(42)	236	(2)	511	(8)	659	(12)	722	(14)	817	(15)	945	(17)	960	(18)
	975	(18)														
.IFF	2003	(39)	2112	(41)	2291	(42)	230	(2)	231	(2)	236	(2)	945	(17)	960	(18)
.IIF	230	(2)	231	(2)	236	(2)										
.IRP	1038	(21)	1140	(22)	1239	(24)	1297	(26)	1403	(28)	1408	(28)	1414	(28)	1624	(32)
	2003	(39)	2108	(41)	2282	(42)	230	(2)	2306	(42)	231	(2)	2322	(42)	234	(2)
	236	(2)	293	(3)	511	(8)	659	(12)	722	(14)	817	(15)	960	(18)	975	(18)
.LIBRARY	183	(2)	184	(2)												
.LIST	15	(1)	2003	(39)	228	(2)	229	(2)	234	(2)	235	(2)	238	(2)	239	(2)
	293	(3)														
.LONG	2350	(42)	2351	(42)	261	(3)	263	(3)	265	(3)	267	(3)	270	(3)	276	(3)
	278	(3)	283	(3)	309	(3)	310	(3)	311	(3)	314	(3)	315	(3)	316	(3)
	317	(3)	318	(3)	319	(3)	321	(3)	322	(3)	381	(5)				
.MACRO	230	(2)	231	(2)	236	(2)										

.MDELETE	1498	(30)	1506	(30)	1529	(30)	231	(2)	236	(2)						
.NLIST	14	(1)	2003	(39)	226	(2)	227	(2)	229	(2)	234	(2)	235	(2)	237	(2)
	293	(3)														
.NOCROSS	232	(2)	233	(2)	235	(2)										
.PAGE	2064	(41)	2185	(42)	2209	(42)	2254	(42)	851	(15)						
.PSECT	2003	(39)	229	(2)	249	(3)	326	(4)	430	(7)						
.QUAD	305	(3)	307	(3)												
.RESTORE	2003	(39)	229	(2)	232	(2)	233	(2)	235	(2)						
.SAVE	2003	(39)	229	(2)	232	(2)	233	(2)	235	(2)						
.SBTTL	1171	(23)	1246	(25)	1313	(27)	1369	(28)	1420	(29)	1460	(30)	1533	(31)	1704	(33)
	1763	(35)	179	(2)	1810	(36)	1869	(37)	1911	(38)	1964	(39)	2186	(42)	2210	(42)
	2255	(42)	429	(7)	472	(8)	514	(9)	587	(11)	663	(13)	852	(15)	980	(20)
.TITLE	12	(1)														
.WEAK	199	(2)	200	(2)	201	(2)	202	(2)	203	(2)	204	(2)	205	(2)	206	(2)
.WORD	2069	(41)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	22	1053 (21) #-1353 (27) 1495 (30) 1565 (32) #-1607 (32) #-1628 (32)
		#-1647 (32) #-1672 (32) #-1697 (32) #-1732 (34) #-1740 (34) #-1751 (34)
		1756 (34) #-2015 (40) #-2023 (40) #-2028 (40) #-2082 (41) #-2109 (41)
		#-2246 (42) #-2247 (42) #-2249 (42) 736 (14)
FP	38	#-1610 (32) #-1631 (32) #-2007 (40) 2008 (40) #-2009 (40) #-2010 (40)
		#-2012 (40) 2024 (40) 2035 (40) #-2123 (42) #-2124 (42) #-2126 (42)
		#-2128 (42) #-2129 (42) #-2132 (42) #-2133 (42) 2134 (42) #-2135 (42)
		#-2136 (42) 2137 (42) #-2138 (42) #-2139 (42) 2141 (42) #-2143 (42)
		#-2145 (42) 2146 (42) #-2147 (42) #-2148 (42) 2149 (42) #-2150 (42)
		#-2151 (42) 2153 (42) #-2155 (42) #-2245 (42)
R0	118	#-1046 (21) #-1114 (22) #-1130 (22) #-1132 (22) #-1134 (22) 1136 (22)
		#-1137 (22) #-1139 (22) #-1140 (22) #-1199 (24) #-1205 (24) #-1209 (24)
		#-1213 (24) #-1216 (24) #-1219 (24) #-1226 (24) #-1228 (24) #-1243 (24)
		#-1275 (26) #-1284 (26) #-1287 (26) #-1290 (26) #-1310 (26) #-1354 (27)
		#-1356 (27) #-1358 (27) #-1360 (27) #-1361 (27) #-1364 (27) #-1452 (29)
		#-1455 (29) #-1457 (29) #-1494 (30) #-1565 (32) #-1566 (32) #-1568 (32)
		#-1570 (32) #-1578 (32) #-1580 (32) #-1607 (32) #-1608 (32) 1609 (32)
		#-1628 (32) #-1629 (32) 1630 (32) #-1647 (32) #-1654 (32) #-1655 (32)
		#-1657 (32) #-1668 (32) #-1670 (32) #-1691 (32) #-1727 (34) #-1732 (34)
		#-1734 (34) #-1740 (34) #-1741 (34) #-1745 (34) #-1746 (34) #-1751 (34)
		#-1752 (34) #-1756 (34) #-1757 (34) #-1759 (34) #-1950 (38) #-1956 (38)
		#-2020 (40) #-2032 (40) #-2038 (40) #-2082 (41) #-2084 (41) #-2086 (41)
		#-2089 (41) #-2091 (41) #-2092 (41) #-2094 (41) #-2096 (41) #-2097 (41)
		#-2100 (41) #-2105 (41) #-2108 (41) #-2109 (41) #-2110 (41) #-2123 (42)
		#-2124 (42) #-2125 (42) #-2126 (42) #-2128 (42) #-2142 (42) #-2154 (42)
		#-2159 (42) #-2166 (42) #-2173 (42) 2174 (42) #-2175 (42) #-2178 (42)
		2179 (42) #-2180 (42) #-2248 (42) 2251 (42) #-2305 (42) #-2318 (42)
		#-628 (12) #-636 (12) #-640 (12) #-642 (12) #-659 (12) #-660 (12)
		#-729 (14) #-799 (15) #-813 (15) #-816 (15) #-817 (15) #-956 (18)
		#-960 (18)
R1	82	#-1101 (22) #-1111 (22) #-1123 (22) #-1199 (24) #-1229 (24) #-1230 (24)
		#-1243 (24) #-1275 (26) #-1291 (26) #-1292 (26) #-1301 (26) #-1302 (26)
		#-1303 (26) #-1306 (26) #-1310 (26) #-1353 (27) #-1356 (27) #-1450 (29)
		#-1452 (29) #-1454 (29) #-1495 (30) #-1496 (30) #-1502 (30) #-1585 (32)
		#-1586 (32) #-1588 (32) #-1612 (32) #-1621 (32) #-1624 (32) #-1633 (32)
		#-1648 (32) #-1668 (32) #-1670 (32) #-2016 (40) #-2029 (40) #-2083 (41)
		#-2087 (41) #-2091 (41) #-2096 (41) 2102 (41) #-2108 (41) #-2110 (41)
		#-2122 (42) #-2126 (42) 2127 (42) #-2128 (42) #-2156 (42) #-2200 (42)
		#-2205 (42) #-2206 (42) #-2325 (42) #-2326 (42) #-2329 (42) #-2330 (42)
		#-2336 (42) #-2338 (42) #-2339 (42) #-2341 (42) 2342 (42) #-2344 (42)
		#-628 (12) #-647 (12) #-651 (12) #-653 (12) #-659 (12) #-660 (12)
		#-795 (15) #-797 (15) #-798 (15) #-808 (15) #-957 (18) 960 (18)
		#-961 (18) #-962 (18) #-964 (18) #-965 (18)
R10	1	#-1696 (32)
R2	130	#-1027 (21) 1034 (21) 1056 (21) 1057 (21) #-1058 (21) 1068 (21)
		1070 (21) 1072 (21) 1082 (22) 1090 (22) #-1101 (22) 1135 (22)
		#-1148 (22) 1151 (22) #-1152 (22) 1155 (22) #-1156 (22) #-1159 (22)
		#-1160 (22) 1163 (22) #-1199 (24) #-1202 (24) 1204 (24) #-1205 (24)
		1208 (24) #-1209 (24) #-1210 (24) #-1235 (24) #-1243 (24) #-1275 (26)
		#-1277 (26) #-1279 (26) 1282 (26) #-1283 (26) #-1284 (26) #-1287 (26)

```
                #-1310  (26)    1346  (27)  #-1358  (27)  #-1395  (28)  #-1398  (28)  #-1399  (28)
                #-1405  (28)  #-1408  (28)  #-1411  (28)  #-1417  (28)  #-1692  (32)  #-1896  (37)
                #-1897  (37)  #-1898  (37)  #-1901  (37)    1902  (37)  #-1906  (37)  #-1907  (37)
                #-1940  (38)  #-1942  (38)  #-1943  (38)    1946  (38)  #-1947  (38)  #-1952  (38)
                #-1958  (38)  #-1959  (38)  #-1961  (38)    2005  (40)  #-2015  (40)  #-2018  (40)
                #-2030  (40)    2050  (41)    2053  (41)    2056  (41)  #-2059  (41)    2127  (42)
                #-2159  (42)  #-2166  (42)    2244  (42)  #-2246  (42)  #-2250  (42)  #-2252  (42)
                  2315  (42)  #-2325  (42)  #-2338  (42)  #-2339  (42)  #-2344  (42)  #-469   (7)
                   560  (10)     562  (10)     564  (10)     573  (10)     575  (10)     577  (10)
                #-628   (12)  #-633   (12)  #-634   (12)  #-638   (12)  #-644   (12)  #-645   (12)
                #-649   (12)  #-660   (12)  #-711   (14)     718  (14)     740  (14)     742  (14)
                #-743   (14)     752  (14)     753  (14)  #-754   (14)     758  (14)     760  (14)
                   768  (15)     815  (15)  #-827   (15)     829  (15)  #-830   (15)     833  (15)
                #-834   (15)  #-836   (15)  #-838   (15)  #-839   (15)     862  (15)     864  (15)
                   866  (15)  #-887   (16)     888  (16)     892  (16)     918  (17)     921  (17)
                   941  (17)  #-942   (17)
R3      56      #-1024  (21)  #-1027  (21)  #-1035  (21)  #-1050  (21)    1053  (21)  #-1058  (21)
                #-1060  (21)    1089  (22)  #-1140  (22)  #-1168  (22)  #-1199  (24)  #-1216  (24)
                #-1229  (24)  #-1243  (24)    1346  (27)  #-1349  (27)  #-1356  (27)  #-1358  (27)
                #-1360  (27)  #-1395  (28)  #-1396  (28)    1402  (28)  #-1412  (28)  #-1417  (28)
                  2005  (40)  #-2016  (40)    2018  (40)  #-2023  (40)  #-2035  (40)  #-2052  (41)
                #-2055  (41)  #-2058  (41)  #-2061  (41)  #-2203  (42)    2244  (42)  #-2247  (42)
                #-2250  (42)  #-2252  (42)  #-2325  (42)  #-2339  (42)  #-2340  (42)  #-2341  (42)
                #-2344  (42)  #-707   (14)  #-711   (14)  #-719   (14)  #-733   (14)     736  (14)
                #-743   (14)  #-745   (14)     776  (15)  #-817   (15)  #-848   (15)     886  (16)
R4      49      #-1024  (21)  #-1067  (21)  #-1069  (21)  #-1071  (21)  #-1073  (21)  #-1091  (22)
                #-1099  (22)  #-1109  (22)  #-1159  (22)  #-1168  (22)  #-1199  (24)  #-1200  (24)
                #-1210  (24)  #-1214  (24)  #-1221  (24)  #-1230  (24)  #-1243  (24)    1346  (27)
                #-1350  (27)  #-1361  (27)  #-1693  (32)    2005  (40)  #-2024  (40)  #-2028  (40)
                #-2029  (40)    2030  (40)  #-2051  (41)  #-2052  (41)  #-2054  (41)  #-2055  (41)
                #-2057  (41)  #-2058  (41)  #-2060  (41)  #-2061  (41)  #-707   (14)  #-751   (14)
                #-757   (14)  #-759   (14)  #-761   (14)  #-777   (15)  #-784   (15)  #-793   (15)
                #-838   (15)  #-848   (15)  #-923   (17)  #-926   (17)  #-954   (18)  #-960   (18)
R5      22      #-1100  (22)  #-1113  (22)  #-1123  (22)    2005  (40)  #-2007  (40)    2069  (41)
                #-2083  (41)    2173  (42)    2176  (42)    2178  (42)    2181  (42)  #-2202  (42)
                #-707   (14)  #-782   (15)  #-785   (15)  #-798   (15)  #-808   (15)  #-848   (15)
                #-931   (17)  #-945   (17)  #-955   (18)  #-960   (18)
R6      14      #-1694  (32)  #-707   (14)  #-848   (15)  #-859   (15)  #-863   (15)  #-865   (15)
                #-868   (15)  #-871   (15)  #-874   (15)  #-876   (15)  #-929   (17)  #-954   (18)
                #-960   (18)  #-976   (18)
R7      16      #-707   (14)  #-848   (15)  #-859   (15)  #-887   (16)  #-890   (16)  #-897   (16)
                #-901   (16)  #-902   (16)  #-922   (17)  #-925   (17)  #-928   (17)  #-930   (17)
                #-933   (17)  #-934   (17)  #-935   (17)  #-976   (18)
R8      25      #-1024  (21)  #-1168  (22)  #-1695  (32)  #-1794  (35)  #-1798  (35)  #-1799  (35)
                #-1800  (35)  #-1801  (35)  #-1802  (35)  #-1803  (35)  #-1804  (35)  #-1843  (36)
                #-1847  (36)  #-1848  (36)  #-1849  (36)  #-1850  (36)  #-1851  (36)  #-1852  (36)
                #-1864  (36)  #-2075  (41)  #-2076  (41)  #-2077  (41)  #-2078  (41)  #-2079  (41)
                #-2080  (41)
SP      47      #-1122  (22)  #-1520  (30)  #-1521  (30)  #-1523  (30)  #-1526  (30)  #-1528  (30)
                #-1597  (32)  #-1598  (32)  #-1600  (32)  #-1602  (32)  #-1608  (32)  #-1610  (32)
                #-1612  (32)  #-1620  (32)  #-1629  (32)  #-1631  (32)  #-1633  (32)    1637  (32)
                #-1648  (32)  #-1650  (32)  #-1651  (32)  #-1652  (32)  #-1653  (32)  #-1673  (32)
                #-1674  (32)  #-1675  (32)  #-1680  (32)  #-1681  (32)  #-1683  (32)  #-1685  (32)
                #-1698  (32)  #-1699  (32)  #-1701  (32)    1733  (34)  #-2008  (40)    2201  (42)
                #-2345  (42)  #-773   (15)     775  (15)  #-807   (15)     956  (18)     957  (18)
                #-963   (18)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	45	00:00:00.05	00:00:00.20
Command processing	956	00:00:00.20	00:00:00.75
Pass 1	1002	00:00:03.96	00:00:06.05
Symbol table sort	0	00:00:00.21	00:00:00.20
Pass 2	252	00:00:01.38	00:00:02.75
Symbol table output	2	00:00:00.07	00:00:00.15
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	12	00:00:00.70	00:00:01.10
Assembler run totals	2273	00:00:06.58	00:00:11.21

The working set limit was 2900 pages.
185374 bytes (363 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 634 non-local and 147 local symbols.
2353 source lines were read in Pass 1, producing 75 object records in Pass 2.
84 pages of virtual memory were used to define 35 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	8
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	9
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	18
TOTALS (all libraries)	35

741 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DEBUG.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

Table of contents

(1) 89 ALLOCATE DEVICE
(1) 211 DEALLOCATE DEVICE
(1) 286 LOCK I/O DATA BASE AND SEARCH FOR DEVICE
(1) 302 DECREMENT REFERENCE COUNT, CLEAR OWNERSHIP, AND CALL DRIVER
(1) 328 DSX$MOUNT - Mount a device
(1) 365 DSX$DISMOUNT - Dismount a device

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DEVALC.MAR
0000     2 ; *1     6-FEB-1986 15:16:54 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element DEVALC.MAR
0000     4         .TITLE  DEVALC  *** DEVALC device allocation routines
0000     5         .IDENT  /05-05/
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980, 1984                               *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    12 ;*                                                                           *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    18 ;*  TRANSFERRED.                                                             *
0000    19 ;*                                                                           *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    22 ;*  CORPORATION.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28 ;
0000    29 ; D. N. CUTLER 9-SEP-76
0000    30 ;
0000    31 ; MODIFICATION HISTORY:
0000    32 ;
0000    33 ;                N. HOWGATE      17-FEB-78       VERSION 02 (ESSAA-4.00).
0000    34 ;       01       DIAGNOSTIC SUPERVISOR INTEGRATION
0000    35 ;                Dave Butenhof   13-may-1980, Version 5.4
0000    36 ;       02       Modify VMS V2.0 source to supervisor environment.
0000    37 ;
0000    38 ;       V02      LMK0001         LEN KAWELL              20-JAN-1980
0000    39 ;                ADDED NONSHAREABLE DEVICE ALLOCATION PRIVILEGE CHECK.
0000    40 ;
0000    41 ;       [03]     Richard Brown   12-July-84      Version 7.0
0000    42 ;                Added DSX$MOUNT and DSX$DISMOUNT
0000    43 ;
0000    44 ;       [04]     Richard Brown   20-Sep-84       Version 7.1
0000    45 ;                Removed DSX$MOUNT and DSX$DISMOUNT.
0000    46 ;
0000    47 ;       05       Ron Parker      6-NOV-1984
0000    48 ;                Fixed trucation error
0000    49 ;
0000    50 ; SYSTEM SERVICES FOR DEVICE ALLOCATION
0000    51 ;
0000    52 ;       ALLOCATE DEVICE
0000    53 ;       DEALLOCATE DEVICE
0000    54 ;
0000    55 ; MACRO LIBRARY CALLS
0000    56 ;
0000    57
```

```
                        0000     58          $ALLOCDEF                       ; $ALLOC CALL OFFSETS
                        0000     59          $DALLOCDEF                      ;$DALLOC CALL OFFSETS
                        0000     60          $DDBDEF                         ;DEFINE DDB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
00000004                0000                         .BLKL   1
                        0004                 .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
00000008                0004                         .BLKL   1
                        0008                 .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
0000000A                0008                         .BLKW   1
                        000A                 .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
0000000B                000A                         .BLKB   1
0000000C                000B                         .BLKB   1
                        000C                 .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
00000010                000C                         .BLKL   1
                        0010                 .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
00000013                0010                         .BLKB   3
                        0013                 .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
00000014                0013                         .BLKB   1
                        0014                 .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
00000015                0014                         .BLKB   1
00000024                0015                         .BLKB   15
                        0024                 .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
00000025                0024                         .BLKB   1
00000034                0025                         .BLKB   15
                        0034                 .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                        0034                 .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                        0000                 .RESTORE
                        0000     61          $DDTDEF                         ;DEFINE DDT OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                0034                 .=0
                        0000                 .IIF    NB,DDT$L_START, DDT$L_START:
00000004                0000                         .BLKL   1
                        0004                 .IIF    NB,DDT$L_UNSOLINT,      DDT$L_UNSOLINT:
00000008                0004                         .BLKL   1
                        0008                 .IIF    NB,DDT$L_FDT,   DDT$L_FDT:
0000000C                0008                         .BLKL   1
                        000C                 .IIF    NB,DDT$L_CANCEL,        DDT$L_CANCEL:
00000010                000C                         .BLKL   1
                        0010                 .IIF    NB,DDT$L_REGDUMP,       DDT$L_REGDUMP:
00000014                0010                         .BLKL   1
                        0014                 .IIF    NB,DDT$W_DIAGBUF,       DDT$W_DIAGBUF:
00000016                0014                         .BLKW   1
                        0016                 .IIF    NB,DDT$W_ERRORBUF,      DDT$W_ERRORBUF:
00000018                0016                         .BLKW   1
                        0018                 .IIF    NB,DDT$L_UNITINIT,      DDT$L_UNITINIT:
0000001C                0018                         .BLKL   1
                        001C                 .IIF    NB,DDT$L_ALTSTART,      DDT$L_ALTSTART:
00000020                001C                         .BLKL   1
                        0000                 .RESTORE
                        0000     62          $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                0034                 .=0
                        0000                 .RESTORE
                        0000     63          $PCBDEF                         ;DEFINE PCB OFFSETS
```

```
                        0000            .SAVE   LOCAL_BLOCK
                        0000            .PSECT  $ABS$,ABS
                00000000 0034           .=0
                        0000            .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
                00000004 0000                           .BLKL   1
                        0004            .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
                00000008 0004                           .BLKL   1
                        0008            .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
                0000000A 0008                           .BLKW   1
                        000A            .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
                0000000B 000A                           .BLKB   1
                        000B            .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
                0000000C 000B                           .BLKB   1
                        000C            .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
                0000000D 000C                           .BLKB   1
                        000D            .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
                0000000E 000D                           .BLKB   1
                        000E            .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
                00000010 000E                           .BLKW   1
                        0010            .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
                00000014 0010                           .BLKL   1
                        0014            .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
                00000018 0014                           .BLKL   1
                        0018            .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
                0000001C 0018                           .BLKL   1
                        001C            .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
                00000020 001C                           .BLKL   1
                        0020            .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
                00000024 0020                           .BLKL   1
                        0024            .IIF    NB,PCB$L_STS,   PCB$L_STS:
                00000028 0024                           .BLKL   1
                        0028            .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
                0000002C 0028                           .BLKL   1
                        002C            .IIF    NB,PCB$W_STATE, PCB$W_STATE:
                0000002E 002C                           .BLKW   1
                        002E            .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
                0000002F 002E                           .BLKB   1
                        002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
                00000030 002F                           .BLKB   1
                        0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
                00000032 0030                           .BLKW   1
                        0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
                00000034 0032                           .BLKW   1
                        0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
                00000036 0034                           .BLKW   1
                        0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
                00000038 0036                           .BLKW   1
                        0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
                0000003A 0038                           .BLKW   1
                        003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
                0000003C 003A                           .BLKW   1
                        003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
                0000003E 003C                           .BLKW   1
                        003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
                00000040 003E                           .BLKW   1
                        0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
                00000042 0040                           .BLKW   1
```

```
                     0042          .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044             0042                  .BLKW   1
                     0044          .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C             0044                  .BLKB   8
                     004C          .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                     004C          .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050             004C                  .BLKL   1
                     0050          .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054             0050                  .BLKL   1
                     0054          .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058             0054                  .BLKL   1
                     0058          .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C             0058                  .BLKL   1
                     005C          .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060             005C                  .BLKL   1
                     0060          .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064             0060                  .BLKL   1
                     0064          .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068             0064                  .BLKL   1
                     0068          .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078             0068                  .BLKB   16
                     0078          .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C             0078                  .BLKL   1
                     007C          .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084             007C                  .BLKQ   1
                     0084          .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088             0084                  .BLKL   1
                     0088          .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                     0088          .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A             0088                  .BLKW   1
                     008A          .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C             008A                  .BLKW   1
                     008C          .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                     008C          .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                     0000          .RESTORE
                     0000    64    $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                     0000          .SAVE   LOCAL_BLOCK
                     0000          .PSECT  $ABS$,ABS
00000000             008C          .=0
                     0000          .RESTORE
                     0000    65    $PRVDEF                         ;DEFINE PRIVILEGE BITS
                     0000          .SAVE   LOCAL_BLOCK
                     0000          .PSECT  $ABS$,ABS
00000000             008C          .=0
                     0000          .RESTORE
                     0000    66    $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                     0000          .SAVE   LOCAL_BLOCK
                     0000          .PSECT  $ABS$,ABS
00000000             008C          .=0
                     0000          .RESTORE
                     0000    67    $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                     0000          .SAVE   LOCAL_BLOCK
                     0000          .PSECT  $ABS$,ABS
00000000             008C          .=0
                     0000          .RESTORE
                     0000    68    $UCBDEF                         ;DEFINE UCB OFFSETS
                     0000          .SAVE   LOCAL_BLOCK
```

```
                    0000            .PSECT  $ABS$,ABS
          00000000  008C            .=0
                    0000            .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                    0000            .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
          00000004  0000                    .BLKL   1
                    0004            .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                    0004            .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
          00000008  0004                    .BLKL   1
                    0008            .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
          0000000A  0008                    .BLKW   1
                    000A            .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
          0000000B  000A                    .BLKB   1
                    000B            .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
          0000000C  000B                    .BLKB   1
                    000C            .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C            .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C            .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
          0000000D  000C                    .BLKB   1
          00000010  000D                    .BLKB   3
                    0010            .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010            .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
          00000014  0010                    .BLKL   1
                    0014            .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014            .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014            .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
          00000016  0014                    .BLKW   1
                    0016            .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
          00000018  0016                    .BLKW   1
                    0018            .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018            .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
          0000001A  0018                    .BLKW   1
                    001A            .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A            .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
          0000001C  001A                    .BLKW   1
                    001C            .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
          00000020  001C                    .BLKL   1
                    0020            .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
          00000024  0020                    .BLKL   1
                    0024            .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
          00000028  0024                    .BLKL   1
                    0028            .IIF    NB,UCB$L_PID,   UCB$L_PID:
          0000002C  0028                    .BLKL   1
                    002C            .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
          00000030  002C                    .BLKL   1
                    0030            .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
          00000034  0030                    .BLKL   1
                    0034            .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
          00000038  0034                    .BLKL   1
                    0038            .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
          00000039  0038                    .BLKB   1
                    0039            .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
          0000003A  0039                    .BLKB   1
                    003A            .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
          0000003C  003A                    .BLKW   1
                    003C            .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C            .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C            .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
```

```
                    0000003D  003C                              .BLKB   1
                              003D              .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                              003D              .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
                    0000003E  003D                              .BLKB   1
                              003E              .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                              003E              .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
                    0000003F  003E                              .BLKB   1
                              003F              .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
                    00000040  003F                              .BLKB   1
                              0040              .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
                    00000044  0040                              .BLKL   1
                              0044              .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
                    00000048  0044                              .BLKL   1
                              0048              .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
                    0000004A  0048                              .BLKW   1
                              004A              .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                              004A              .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
                    0000004B  004A                              .BLKB   1
                              004B              .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
                    0000004C  004B                              .BLKB   1
                              004C              .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
                    00000050  004C                              .BLKL   1
                              0050              .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
                    00000052  0050                              .BLKW   1
                              0052              .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                              0052              .IIF    NB,UCB$B_STATE, UCB$B_STATE:
                    00000053  0052                              .BLKB   1
                              0053              .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
                    00000054  0053                              .BLKB   1
                              0054              .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
                    00000058  0054                              .BLKL   1
                              0058              .IIF    NB,UCB$W_STS,   UCB$W_STS:
                    0000005A  0058                              .BLKW   1
                              005A              .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
                    0000005C  005A                              .BLKW   1
                              005C              .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                              005C              .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
                    00000060  005C                              .BLKL   1
                              0060              .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
                    00000064  0060                              .BLKL   1
                              0064              .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                              0064              .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
                    00000068  0064                              .BLKL   1
                              0068              .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
                    0000006C  0068                              .BLKL   1
                              006C              .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
                    0000006E  006C                              .BLKW   1
                              006E              .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
                    00000070  006E                              .BLKW   1
                              0070              .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
                    00000071  0070                              .BLKB   1
                              0071              .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
                    00000072  0071                              .BLKB   1
                              0072              .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
                    00000074  0072                              .BLKW   1
                              0074              .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                              0074              .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
```

```
00000000  0074          . = 0
          0000          .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                          .BLKW   1
00000074  0002          . = 116
          0074          .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                          .BLKL   1
          0078          .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                          .BLKL   1
          0080          .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                          .BLKL   1
          0084          .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                          .BLKL   1
          008C          .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                          .BLKL   1
          0090          .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
          0090          .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090          . = 116
          0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                          .BLKB   1
          0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                          .BLKB   1
          0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                          .BLKB   1
          0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                          .BLKB   1
          0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                          .BLKL   1
0000007E  007C                          .BLKW   1
          007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                          .BLKW   1
          0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                          .BLKL   1
00000080  0084          . = 128
00000084  0080                          .BLKL   1
          0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                          .BLKW   1
          008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                          .BLKW   1
          008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                          .BLKW   1
          008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                          .BLKW   1
          0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                          .BLKW   1
          0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                          .BLKW   1
          0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                          .BLKB   1
          0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                          .BLKB   1
```

```
                        0096            .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                  .BLKW   1
00000096  0098          . = 150
00000098  0096                                  .BLKW   1
                        0098            .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                  .BLKL   1
                        009C            .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                  .BLKL   1
                        00A0            .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                                  .BLKL   1
                        00A4            .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                                  .BLKW   1
                        00A6            .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                                  .BLKB   1
000000A8  00A7                                  .BLKB   1
00000074  00A8          . = 116
00000078  0074                                  .BLKL   1
0000007C  0078                                  .BLKL   1
00000080  007C                                  .BLKL   1
00000084  0080                                  .BLKL   1
00000088  0084                                  .BLKL   1
0000008C  0088                                  .BLKL   1
                        008C            .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                  .BLKL   1
00000094  0090                                  .BLKL   1
00000098  0094                                  .BLKL   1
0000009C  0098                                  .BLKL   1
                        009C            .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                  .BLKB   1
                        009D            .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                  .BLKB   1
                        009E            .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                  .BLKB   1
000000A0  009F                                  .BLKB   1
                        00A0            .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                  .BLKB   1
                        00A1            .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                  .BLKB   1
                        00A2            .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                  .BLKB   1
000000A4  00A3                                  .BLKB   1
                        00A4            .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                  .BLKB   1
                        00A5            .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                  .BLKW   1
000000A8  00A7                                  .BLKB   1
                        00A8            .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                  .BLKL   1
000000B0  00AC                                  .BLKL   1
000000B4  00B0                                  .BLKL   1
000000B8  00B4                                  .BLKL   1
                        00B8            .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                  .BLKL   1
                        00BC            .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                        00BC            .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC          . = 116
                        0074            .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
```

```
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                  .BLKW   1
00000080  007E                                  .BLKW   1
          0000                  .RESTORE
          0000
          0000                  $DS_DSADEF                      ; Define VDS flags              [03]
          0000  ;
          0000  ; LOCAL SYMBOLS
          0000  ;
          0000  ; ARGUMENT LIST OFFSET DEFINITIONS FOR ALLOCATE DEVICE
          0000  ;
          0000
00000004  0000  DEVNAM=4                                  ;ADDRESS OF DEVICE NAME STRING DESCRIPTOR
00000008  0000  PHYLEN=8                                  ;ADDRESS TO STORE LENGTH OF PHYSICAL NAME
0000000C  0000  PHYBUF=12                                 ;ADDRESS OF PHYSICAL NAME BUFFER DESCRIPTOR
00000010  0000  ALACMODE=16                               ;ACCESS MODE
          0000
          0000  ;
          0000  ; ARGUMENT LIST OFFSET DEFINITIONS FOR DEALLOCATE DEVICE
          0000  ;
          0000
00000004  0000  DEVNAM=4                                  ;ADDRESS OF DEVICE NAME STRING DESCRIPTOR
00000008  0000  DLACMODE=8                                ;ACCESS MODE
```

```
0000     89          .SBTTL  ALLOCATE DEVICE
0000     90 ;+
0000     91 ; EXE$ALLOC - ALLOCATE DEVICE
0000     92 ;
0000     93 ; THIS SERVICE PROVIDES THE CAPABILITY TO RESERVE A DEVICE FOR EXCLUSIVE
0000     94 ; USE.
0000     95 ;
0000     96 ; INPUTS:
0000     97 ;
0000     98 ;       DEVNAM(AP) = ADDRESS OF DEVICE NAME STRING DESCRIPTOR.
0000     99 ;       PHYLEN(AP) = ADDRESS TO STORE LENGTH OF PHYSICAL NAME.
0000    100 ;       PHYBUF(AP) = ADDRESS OF PHYSICAL NAME BUFFER DESCRIPTOR.
0000    101 ;       ALACMODE(AP) = ACCESS MODE DEVICE IS TO BE ALLOCATED TO.
0000    102 ;
0000    103 ;       R4 = CURRENT PROCESS PCB ADDRESS.
0000    104 ;
0000    105 ; OUTPUTS:
0000    106 ;
0000    107 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO ALLOCATE DEVICE.
0000    108 ;
0000    109 ;               R0 = SS$_ACCVIO - DEVICE NAME STRING, DEVICE NAME STRING
0000    110 ;                    DESCRIPTOR, OR PHYSICAL NAME BUFFER DESCRIPTOR
0000    111 ;                    CANNOT BE READ BY CALLING ACCESS MODE, OR PHYSICAL
0000    112 ;                    NAME BUFFER CANNOT BE WRITTEN BY CALLING ACCESS
0000    113 ;                    MODE.
0000    114 ;
0000    115 ;               R0 = SS$_DEVALLOC - DEVICE ALREADY ALLOCATED TO ANOTHER
0000    116 ;                    PROCESS.
0000    117 ;
0000    118 ;               R0 = SS$_DEVMOUNT - DEVICE CURRENTLY MOUNTED AND CANNOT
0000    119 ;                    BE ALLOCATED.
0000    120 ;
0000    121 ;               R0 = SS$_IVDEVNAM - DEVICE NAME STRING CONTAINS INVALID
0000    122 ;                    CHARACTERS OR NO DEVICE NAME STRING DESCRIPTOR
0000    123 ;                    SPECIFIED.
0000    124 ;
0000    125 ;               R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
0000    126 ;                    DEVICE NAME STRING SPECIFIED.
0000    127 ;
0000    128 ;               R0 = SS$_NONLOCAL - SPECIFIED DEVICE EXISTS ON A REMOTE
0000    129 ;                    SYSTEM.
0000    130 ;
0000    131 ;               R0 = SS$_NOPRIV - DEVICE CURRENTLY SPOOLED AND PROCESS DOES
0000    132 ;                    NOT HAVE PRIVILEGE TO ALLOCATE.
0000    133 ;
0000    134 ;               R0 = SS$_NOSUCHDEV - SPECIFIED DEVICE DOES NOT EXIST ON
0000    135 ;                    HOST SYSTEM OR NO ALLOCATABLE UNIT CAN BE FOUND.
0000    136 ;
0000    137 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
0000    138 ;
0000    139 ;               R0 = SS$_DEVALRALLOC - DEVICE IS ALREADY ALLOCATED TO CALLING
0000    140 ;                    PROCESS.
0000    141 ;
0000    142 ;               R0 = SS$_BUFFEROVF - NORMAL COMPLETION, PHYSICAL NAME OVER-
0000    143 ;                    FLOWED PHYSICAL NAME BUFFER.
0000    144 ;
0000    145 ;               R0 = SS$_NORMAL - NORMAL COMPLETION, PHYSICAL NAME TRANSFERED
```

```
                                0000   146 ;                           TO PHYSICAL NAME BUFFER.
                                0000   147 ;-
                                0000   148
                             00000000   149          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                                0000   150          MODNAM  SYSDEVALC
 43 4C 41 56 45 44 53 59 53 00' 0000        $MODULE:        .ASCIC  \SYSDEVALC\
                                09  0000
                               007C  000A   151          .ENTRY  EXE$ALLOC,^M<R2,R3,R4,R5,R6>
        54   00000000'EF         9E  000C   152          MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
             55   0144 8F        3C  0013   153          MOVZWL  #SS$_IVDEVNAM,R5        ;SET INVALID DEVICE NAME STATUS
                     00F0        30  0018   154          BSBW    SEARCHDEV               ;LOCK I/O DATA BASE AND GET DEVICE NAME
                       3F        13  001B   155          BEQL    27$                     ;IF EQL NO DEVICE NAME SPECIFIED
                    FFE0'        30  001D   156          BSBW    IOC$SEARCHGEN           ;SEARCH FOR GENERIC DEVICE
                  55    50       D0  0020   157          MOVL    R0,R5                   ;SAVE SEARCH STATUS
                    36  55       E9  0023   158          BLBC    R5,27$                  ;IF LBC SEARCH FAILURE
                  56    51       D0  0026   159          MOVL    R1,R6                   ;SAVE DEVICE UCB ADDRESS
                                0029   160 ;
                                0029   161 ; RETURN PHYSICAL DEVICE NAME
                                0029   162 ;
                51    0C AC     D0  0029   163          MOVL    PHYBUF(AP),R1           ;GET ADDRESS OF NAME BUFFER DESCRIPTOR
                        1C      13  002D   164          BEQL    20$                     ;IF EQL NONE
                   50   61      3C  002F   165          MOVZWL  (R1),R0                 ;GET LENGTH OF NAME BUFFER
                        17      13  0032   166          BEQL    20$                     ;IF EQL ZERO LENGTH BUFFER
                51    04 A1     D0  0034   167          MOVL    4(R1),R1                ;GET ADDRESS OF NAME BUFFER
                53    08 AC     D0  0038   168          MOVL    PHYLEN(AP),R3           ;GET ADDRESS TO STORE NAME LENGTH
                        00      13  003C   169          BEQL    10$                     ;IF EQL NONE
                     FFBF'      30  003E   170 10$:     BSBW    IOC$CVT_DEVNAM          ;CONVERT DEVICE NAME AND UNIT
                   55   50      D0  0041   171          MOVL    R0,R5                   ;SAVE COMPLETION STATUS
                        53      D5  0044   172          TSTL    R3                      ;ADDRESS TO STORE LENGTH SPECIFIED?
                        03      13  0046   173          BEQL    20$                     ;IF EQL NONE SPECIFIED
                63      51      B0  0048   174          MOVW    R1,(R3)                 ;INSERT LENGTH OF CONVERTED STRING
                                004B   175 ;
                                004B   176 ; CHECK IF PROCESS ALREADY ALLOCATED DEVICE OR CAN ALLOCATE THE DEVICE
                                004B   177 ;
              50    28 A6       D0  004B   178 20$:     MOVL    UCB$L_PID(R6),R0        ;GET PROCESS ID OF OWNER
                        0D      13  004F   179          BEQL    30$                     ;IF EQL DEVICE NOT OWNED
              60 A4     50      D1  0051   180          CMPL    R0,PCB$L_PID(R4)        ;OWNED BY CURRENT PROCESS?
                        1D      13  0055   181          BEQL    60$                     ;IF EQL YES
           55   0840 8F         3C  0057   182 25$:     MOVZWL  #SS$_DEVALLOC,R5        ;SET DEVICE ALREADY ALLOCATED
                        42      11  005C   183 27$:     BRB     80$                     ;
                                005E   184
        02 34 A6        06      E1  005E   185 30$:     BBC     #DEV$V_SPL,UCB$L_DEVCHAR(R6),40$ ;IF CLR, DEVICE NOT SPOOLED
                        05      11  0063   186          BRB     47$                     ;
                     50 A6      B5  0065   187 40$:     TSTW    UCB$W_REFC(R6)          ;ANY CHANNELS ASSIGNED TO DEVICE?
                        ED      12  0068   188          BNEQ    25$                     ;IF NEQ YES
                   00AE         31  006A   189 47$:     BRW     NOPRIV                  ;ELSE NO PRIVILEGE
                        31      11  006D   190 50$:     BRB     80$                     ;
                55      0C      3C  006F   191 55$:     MOVZWL  #SS$_ACCVIO,R5          ;SET ACCESS VIOLATION STATUS
                        2C      11  0072   192          BRB     80$                     ;
                                0074   193
                55      01      B1  0074   194 60$:     CMPW    #SS$_NORMAL,R5          ;NORMAL COMPLETION STATUS?
                        0A      12  0077   195          BNEQ    70$                     ;IF NEQ NO
        05 34 A6        17      E1  0079   196          BBC     #DEV$V_ALL,UCB$L_DEVCHAR(R6),70$ ;IF CLR, DEVICE NOT ALLOCATED
         55   0641 8F           3C  007E   197          MOVZWL  #SS$_DEVALRALLOC,R5     ;SET DEVICE ALREADY ALLOCATED STATUS
                                0083   198 ;
                                0083   199 ; ALLOCATE THE DEVICE BY SETTING OWNER PID, SETTING ALLOCATED CHARACTERISTIC,
                                0083   200 ; INCREMENTING THE REFERENCE COUNT, AND SETTING THE ALLOCATOR'S ACCESS MODE
```

```
                                0083   201 ;
          28 A6    60 A4   D0   0083   202 70$:    MOVL    PCB$L_PID(R4),UCB$L_PID(R6) ;SET OWNER PROCESS ID
          13 34 A6    17   E2   0088   203         BBSS    #DEV$V_ALL,UCB$L_DEVCHAR(R6),80$ ;IF SET, DEVICE ALREADY ALLOCATED
                   50 A6   B6   008D   204         INCW    UCB$W_REFC(R6)           ;INCREMENT REFERENCE COUNT
   50   10 AC    02    00   EF   0090   205         EXTZV   #0,#2,ALACMODE(AP),R0    ;GET SPECIFIED ACCESS MODE
          00000000'EF      16   0096   206         JSB     EXE$MAXACMODE            ;MAXIMIZE ACCESS MODE
            53 A6    50   90   009C   207         MOVB    R0,UCB$B_AMOD(R6)        ;SET ALLOCATING ACCESS MODE
               50    55   D0   00A0   208 80$:    MOVL    R5,R0                    ;SET FINAL STATUS
                  FF5A'   31   00A3   209         BRW     IOC$UNLOCK               ;
```

```
                        00A6   211          .SBTTL  DEALLOCATE DEVICE
                        00A6   212 ;+
                        00A6   213 ; EXE$DALLOC - DEALLOCATE DEVICE
                        00A6   214 ;
                        00A6   215 ; THIS SERVICE PROVIDES THE CAPABILITY TO RELINQUISH EXCLUSIVE USE OF A
                        00A6   216 ; DEVICE.
                        00A6   217 ;
                        00A6   218 ; INPUTS:
                        00A6   219 ;
                        00A6   220 ;       DEVNAM(AP) = ADDRESS OF DEVICE NAME STRING DESCRIPTOR. ZERO
                        00A6   221 ;                    IMPLIES ALL.
                        00A6   222 ;       DLACMODE(AP) = ACCESS MODE FOR DEALLOCATION.
                        00A6   223 ;
                        00A6   224 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        00A6   225 ;
                        00A6   226 ; OUTPUTS:
                        00A6   227 ;
                        00A6   228 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO DEALLOCATE DEVICE.
                        00A6   229 ;
                        00A6   230 ;              R0 = SS$_ACCVIO - DEVICE NAME STRING OR DEVICE NAME STRING
                        00A6   231 ;                   DESCRIPTOR CANNOT BE READ BY CALLING ACCESS MODE.
                        00A6   232 ;
                        00A6   233 ;              R0 = SS$_DEVASSIGN - DEVICE CANNOT BE DEALLOCATED BECAUSE
                        00A6   234 ;                   PROCESS STILL HAS CHANNELS ASSIGNED.
                        00A6   235 ;
                        00A6   236 ;              R0 = SS$_DEVNOTALLOC - DEVICE NOT ALLOCATED OR NOT ALLOCATED
                        00A6   237 ;                   TO PROCESS.
                        00A6   238 ;
                        00A6   239 ;              R0 = SS$_IVDEVNAM - DEVICE NAME STRING CONTAINS INVALID
                        00A6   240 ;                   CHARACTERS.
                        00A6   241 ;
                        00A6   242 ;              R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
                        00A6   243 ;                   DEVICE NAME STRING SPECIFIED.
                        00A6   244 ;
                        00A6   245 ;              R0 = SS$_NOPRIV - CALLING ACCESS MODE DOES NOT HAVE PRIVILEGE
                        00A6   246 ;                   TO DEALLOCATE DEVICE.
                        00A6   247 ;
                        00A6   248 ;              R0 = SS$_NOSUCHDEV - SPECIFIED DEVICE DOES NOT EXIST ON
                        00A6   249 ;                   HOST SYSTEM.
                        00A6   250 ;
                        00A6   251 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                        00A6   252 ;
                        00A6   253 ;              R0 = SS$_NORMAL - NORMAL COMPLETION.
                        00A6   254 ;-
                        00A6   255
                 003C   00A6   256          .ENTRY  EXE$DALLOC,^M<R2,R3,R4,R5>
54  00000000'EF   9E   00A8   257          MOVAB   DS$AX_SOFTPCB,R4        ;SOFTWARE PCB TO R4
              5A   10   00AF   258          BSBB    SEARCHDEV               ;LOCK I/O DATA BASE AND SEARCH FOR DEVICE
              1B   13   00B1   259          BEQL    10$                     ;IF EQL NO DEVICE NAME SPECIFIED
          FF4A'   30   00B3   260          BSBW    IOC$SEARCHDEV           ;SEARCH FOR PHYSICAL DEVICE
        50 50   E9   00B6   261          BLBC    R0,50$                  ;IF LBC SEARCH FAILURE
60 A4   28 A1   D1   00B9   262          CMPL    UCB$L_PID(R1),PCB$L_PID(R4) ;DEVICE ALLOCATED TO CURRENT PROCESS?
              44   12   00BE   263          BNEQ    40$                     ;IF NEQ NO
   50 A1    01   B1   00C0   264          CMPW    #1,UCB$W_REFC(R1)       ;REFERENCE COUNT ONE?
              43   12   00C4   265          BNEQ    50$                     ;IF NEQ NO
  00000126'EF   16   00C6   266          JSB     DECREF                  ;DECREMENT REFERENCE COUNT AND CALL DRIVER
              52   11   00CC   267          BRB     NORMAL                  ;
```

```
                53   00000000'EF   9E   00CE    268 10$:    MOVAB   L^IOC$GL_DEVLIST-DDB$L_LINK,R3 ;GET ADDRESS OF I/O DATABASE LISTHEAD
                    53    63   D0   00D5    269 20$:    MOVL    DDB$L_LINK(R3),R3       ;GET ADDRESS OF NEXT DDB
                          46   13   00D8    270         BEQL    NORMAL                  ;IF EQL END OF LIST
                     51    D8 A3   9E   00DA    271         MOVAB   DDB$L_UCB-UCB$L_LINK(R3),R1 ;GET ADDRESS OF NEXT UCB ADDRESS
                     51    2C A1   D0   00DE    272 30$:    MOVL    UCB$L_LINK(R1),R1       ;GET ADDRESS OF NEXT UCB
                           F1   13   00E2    273         BEQL    20$                     ;IF EQL END OF LIST
                60 A4    28 A1   D1   00E4    274         CMPL    UCB$L_PID(R1),PCB$L_PID(R4) ;DEVICE ALLOCATED TO CURRENT PROCESS?
                           F3   12   00E9    275         BNEQ    30$                     ;IF NEQ NO
                   53 A1    55   91   00EB    276         CMPB    R5,UCB$B_AMOD(R1)       ;CAN ACCESS MODE DEALLOCATE DEVICE?
                           ED   14   00EF    277         BGTR    30$                     ;IF GTR NO
                E8 34 A1    17   E1   00F1    278         BBC     #DEV$V_ALL,UCB$L_DEVCHAR(R1),30$ ;IF CLR, DEVICE NOT ALLOCATED
                   50 A1    01   B1   00F6    279         CMPW    #1,UCB$W_REFC(R1)       ;REFERENCE COUNT ONE?
                           E2   12   00FA    280         BNEQ    30$                     ;IF NEQ NO
                 00000126'EF   16   00FC    281         JSB     DECREF                  ;DECREMENT REFERENCE COUNT AND CALL DRIVER
                           DA   11   0102    282         BRB     30$                     ;
                50    0858 8F   3C   0104    283 40$:    MOVZWL  #SS$_DEVNOTALLOC,R0     ;SET DEVICE NOT ALLOCATED TO PROCESS
                           18   11   0109    284 50$:    BRB     UNLOCK                  ;
```

```
                        010B   286          .SBTTL  LOCK I/O DATA BASE AND SEARCH FOR DEVICE
                        010B   287 ;
                        010B   288 ; SUBROUTINE TO LOCK I/O DATA BASE, SEARCH FOR DEVICE, AND CHECK FOR MAILBOX.
                        010B   289 ;
                        010B   290
                        010B   291 SEARCHDEV:                                   ;LOCK I/O DATA BASE AND SEARCH FOR DEVICE
      51   04 AC   D0   010B   292          MOVL    DEVNAM(AP),R1                ;GET ADDRESS OF DEVICE NAME STRING DESCRIPTO
           09   13   010F   293          BEQL    10$                          ;IF EQL NONE
      50   0C   3C   0111   294          MOVZWL  #SS$_ACCVIO,R0               ;SET ACCESS VIOLATION STATUS
                     0114   295          IFNORD  #8,(R1),UNLOCK               ;CAN DEVICE NAME DESCRIPTOR BE READ?
  61   08   00   0C 0114                          PROBER  #0,#8,(R1)
           09   13   0118                        BEQL    UNLOCK
           05   011A   296 10$:     RSB                                      ;
      50   24   3C   011B   297 NOPRIV: MOVZWL  #SS$_NOPRIV,R0               ;SET NO PRIVILEGE
           03   11   011E   298          BRB     UNLOCK                       ;
      50   01   3C   0120   299 NORMAL: MOVZWL  #SS$_NORMAL,R0               ;SET NORMAL COMPLETION
         FEDA'  31   0123   300 UNLOCK: BRW     IOC$UNLOCK                   ;UNLOCK I/O DATA BASE AND RETURN
```

```
                        0126   302          .SBTTL  DECREMENT REFERENCE COUNT, CLEAR OWNERSHIP, AND CALL DRIVER
                        0126   303 ;
                        0126   304 ; SUBROUTINE TO DECREMENT REFERENCE COUNT, CLEAR OWNERSHIP, AND CALL DRIVER
                        0126   305 ;
                        0126   306
                        0126   307 DECREF:                                                ;
00 34 A1    17    E5    0126   308          BBCC    #DEV$V_ALL,UCB$L_DEVCHAR(R1),10$ ;CLEAR DEVICE ALLOCATED
      50 A1    B7    012B   309 10$:       DECW    UCB$W_REFC(R1)                   ;DECREMENT REFERENCE COUNT
      28 A1    D4    012E   310          CLRL    UCB$L_PID(R1)                    ;CLEAR OWNER PROCESS ID
         2A    BB    0131   311          PUSHR   #^M<R1,R3,R5>                    ;SAVE REGISTERS
   50   24 A1    D0    0133   312          MOVL    UCB$L_DDB(R1),R0                 ;GET ADDRESS OF DDB
   50   0C A0    D0    0137   313          MOVL    DDB$L_DDT(R0),R0                 ;GET ADDRESS OF DDT
      55   51    D0    013B   314          MOVL    R1,R5                            ;SET ADDRESS OF DEVICE UCB
           52    D4    013E   315          CLRL    R2                               ;CLEAR CHANNEL NUMBER
                        0140   316          DSBINT  UCB$B_FIPL(R5)                   ;RAISE TO DRIVER FORK LEVEL
      7E   12    DB    0140                MFPR    S^#PR$_IPL,-(SP)
   12   0B A5    DA    0143                MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
   53   4C A5    D0    0147   317          MOVL    UCB$L_IRP(R5),R3                 ;GET ADDRESS OF CURRENT I/O PACKET
   51   0C A0    D0    014B   318          MOVL    DDT$L_CANCEL(R0), R1             ; Get Cancel entry point
   06 51    10    E0    014F   319          BBS     #16, R1, 20$                     ; +++++ Skip if Supervisor absolute
      51    50    C0    0153   320          ADDL2   R0, R1                           ; +++++ Convert to absolute
      51    50    C2    0156   321          SUBL    R0,R1                            ;SUBTRACT OUT BASE ADDRESS
           61    16    0159   322 20$:       JSB     (R1)                             ;CALL CANCEL I/O ROUTINE
                        015B   323          ENBINT                                   ;ENABLE INTERRUPTS
      12   8E    DA    015B                MTPR    (SP)+,S^#PR$_IPL
           2A    BA    015E   324          POPR    #^M<R1,R3,R5>                    ;RESTORE REGISTERS
                 05    0160   325          RSB                                      ;
                        0161   326
```

```
                     0161    328 .sbttl  DSX$MOUNT - Mount a device
                     0161    329
                     0161    330 ;++                                                                 [03]
                     0161    331 ; Functional Description:                                           [03]
                     0161    332 ;                                                                   [03]
                     0161    333 ;       Calls the VMS $MOUNT service.                               [03]
                     0161    334 ;                                                                   [03]
                     0161    335 ; Inputs:                                                           [03]
                     0161    336 ;                                                                   [03]
                     0161    337 ;       4(AP) - Address of list of item identifiers                 [03]
                     0161    338 ;                                                                   [03]
                     0161    339 ; Implicit Inputs:                                                  [03]
                     0161    340 ;                                                                   [03]
                     0161    341 ;       DSA$GL_FLAGS                                                [03]
                     0161    342 ;                                                                   [03]
                     0161    343 ; Outputs:                                                          [03]
                     0161    344 ;                                                                   [03]
                     0161    345 ; Implicit Outputs:                                                 [03]
                     0161    346 ;                                                                   [03]
                     0161    347 ; Side Effects:                                                     [03]
                     0161    348 ;                                                                   [03]
                     0161    349 ; Return Status:                                                    [03]
                     0161    350 ;                                                                   [03]
                     0161    351 ;       The return status from VMS is passed to the caller.         [03]
                     0161    352 ;--                                                                 [03]
                     0161    353
                     0161    354 ;[04] .ENTRY    DSX$MOUNT, ^M<>         ;                          [03]
                     0161    355 ;[04]   BBS     #DSA$V_USER, -   ; IF executing in standalone mode  [03]
                     0161    356 ;[04]           DSA$GL_FLAGS,10$; THEN                              [03]
                     0161    357 ;[04]   MOVL    #SS$_NORMAL, R0 ;       Set SS$_NORMAL return code. [03]
                     0161    358 ;[04]   BRB     20$             ;       Just return.                [03]
                     0161    359 ;[04] 10$:                              ; ELSE (user mode)          [03]
                     0161    360 ;[04]   PUSHL   4(AP)           ;       Pass argument.              [03]
                     0161    361 ;[04]   CALLS   #1,G^SYS$MOUNT  ;       Call service.               [03]
                     0161    362 ;[04] 20$:      RET                         ; Return.               [03]
                     0161    363 ;[04]
```

```
0161    365 .sbttl  DSX$DISMOUNT - Dismount a device                                    [03]
0161    366
0161    367 ;++                                                                         [03]
0161    368 ; Functional Description:                                                   [03]
0161    369 ;                                                                           [03]
0161    370 ;       Calls the VMS $DISMOU service.                                      [03]
0161    371 ;                                                                           [03]
0161    372 ; Inputs:                                                                   [03]
0161    373 ;                                                                           [03]
0161    374 ;       4(AP) - Addr. of char. string descriptor pointing to the            [03]
0161    375 ;               device name                                                 [03]
0161    376 ;       8(AP) - Options flag                                                [03]
0161    377 ;                                                                           [03]
0161    378 ; Implicit Inputs:                                                         [03]
0161    379 ;                                                                           [03]
0161    380 ;       DSA$GL_FLAGS                                                        [03]
0161    381 ;                                                                           [03]
0161    382 ; Outputs:                                                                 [03]
0161    383 ;                                                                           [03]
0161    384 ; Implicit Outputs:                                                        [03]
0161    385 ;                                                                           [03]
0161    386 ; Side Effects:                                                            [03]
0161    387 ;                                                                           [03]
0161    388 ; Return Status:                                                           [03]
0161    389 ;                                                                           [03]
0161    390 ;       The return status from VMS is passed to the caller.                 [03]
0161    391 ;--                                                                         [03]
0161    392
0161    393 ;[04] .ENTRY DSX$DISMOUNT, ^M<> ;                                          [03]
0161    394 ;[04]   BBS     #DSA$V_USER, -   ; IF executing in standalone mode      [03]
0161    395 ;[04]           DSA$GL_FLAGS,10$; THEN                                      [03]
0161    396 ;[04]   MOVL    #SS$_NORMAL, R0 ;       Set SS$_NORMAL return code.     [03]
0161    397 ;[04]   BRB     20$             ;       Just return.                    [03]
0161    398 ;[04] 10$:                      ;       ; ELSE (user mode)                  [03]
0161    399 ;[04]   PUSHL   8(AP)           ;       Pass arguments.                 [03]
0161    400 ;[04]   PUSHL   4(AP)           ;                                       [03]
0161    401 ;[04]   CALLS   #2,G^SYS$DISMOU ;       Call service.                   [03]
0161    402 ;[04] 20$:      RET                     ; Return.                           [03]
0161    403
0161    404         .END
```

```
$$ARGS                    = 00000002    D          EXE$DALLOC                  000000A6 RG D  02
$$T1                      = 0000000C    D          EXE$MAXACMODE               ********   X   02
$ER                       = 00000001    D          IOC$CVT_DEVNAM              ********   X   02
$MODULE                     00000000 R  D  02      IOC$GL_DEVLIST              ********   X   02
ALACMODE                  = 00000010    D          IOC$SEARCHDEV               ********   X   02
ALLOC$_ACMODE             = 00000010    D          IOC$SEARCHGEN               ********   X   02
ALLOC$_DEVNAM             = 00000004    D          IOC$UNLOCK                  ********   X   02
ALLOC$_NARGS              = 00000004    D          NOPRIV                      0000011B R  D  02
ALLOC$_PHYBUF             = 0000000C    D          NORMAL                      00000120 R  D  02
ALLOC$_PHYLEN             = 00000008    D          PCB$B_ASTACT                0000000C    D
DALLOC$_ACMODE            = 00000008    D          PCB$B_ASTEN                 0000000D    D
DALLOC$_DEVNAM            = 00000004    D          PCB$B_PRI                   0000000B    D
DALLOC$_NARGS             = 00000002    D          PCB$B_PRIB                  0000002F    D
DDB$B_ACPCLASS              00000013    D          PCB$B_TYPE                  0000000A    D
DDB$B_TYPE                  0000000A    D          PCB$B_WEFC                  0000002E    D
DDB$C_LENGTH                00000034    D          PCB$C_LENGTH                0000008C    D
DDB$K_LENGTH                00000034    D          PCB$K_LENGTH                0000008C    D
DDB$L_ACPD                  00000010    D          PCB$L_ARB                   00000084    D
DDB$L_DDT                   0000000C    D          PCB$L_ASTQBL                00000014    D
DDB$L_LINK                  00000000    D          PCB$L_ASTQFL                00000010    D
DDB$L_UCB                   00000004    D          PCB$L_EFC2P                 00000058    D
DDB$T_DRVNAME               00000024    D          PCB$L_EFC3P                 0000005C    D
DDB$T_NAME                  00000014    D          PCB$L_EFCS                  00000050    D
DDB$W_SIZE                  00000008    D          PCB$L_EFCU                  00000054    D
DDT$L_ALTSTART              0000001C    D          PCB$L_EFWM                  0000004C    D
DDT$L_CANCEL                0000000C    D          PCB$L_JIB                   00000078    D
DDT$L_FDT                   00000008    D          PCB$L_OWNER                 0000001C    D
DDT$L_REGDUMP               00000010    D          PCB$L_PHD                   00000064    D
DDT$L_START                 00000000    D          PCB$L_PHYPCB                00000018    D
DDT$L_UNITINIT              00000018    D          PCB$L_PID                   00000060    D
DDT$L_UNSOLINT              00000004    D          PCB$L_PQB                   0000004C    D
DDT$W_DIAGBUF               00000014    D          PCB$L_SQBL                  00000004    D
DDT$W_ERRORBUF              00000016    D          PCB$L_SQFL                  00000000    D
DECREF                      00000126 R  D  02      PCB$L_STS                   00000024    D
DEV$V_ALL                 = 00000017    D          PCB$L_UIC                   00000088    D
DEV$V_SPL                 = 00000006    D          PCB$L_WSSWP                 00000020    D
DEVNAM                    = 00000004    D          PCB$L_WTIME                 00000028    D
DLACMODE                  = 00000008    D          PCB$Q_PRIV                  0000007C    D
DS$AX_SOFTPCB               ********   X   02      PCB$T_LNAME                 00000068    D
DSA$AL_APTMAIL              0000FE00    D          PCB$T_TERMINAL              00000044    D
DSA$AT_APTTXT               0000FA00    D          PCB$W_APTCNT                00000030    D
DSA$GL_APTCOM               0000FE04    D          PCB$W_ASTCNT                00000038    D
DSA$GL_DEVLEN               0000FE58    D          PCB$W_BIOCNT                0000003A    D
DSA$GL_ERRNO                0000FE44    D          PCB$W_BIOLM                 0000003C    D
DSA$GL_EVENT                0000FE48    D          PCB$W_DIOCNT                0000003E    D
DSA$GL_FLAGS                0000FE00    D          PCB$W_DIOLM                 00000040    D
DSA$GL_MSGTYP               0000FE40    D          PCB$W_GPGCNT                00000034    D
DSA$GL_PASSES               0000FE08    D          PCB$W_GRP                   0000008A    D
DSA$GL_PASSNO               0000FE54    D          PCB$W_MEM                   00000088    D
DSA$GL_SECTNO               0000FE10    D          PCB$W_MTXCNT                0000000E    D
DSA$GL_SID                  0000FE14    D          PCB$W_PPGCNT                00000036    D
DSA$GL_SUBTNO               0000FE4C    D          PCB$W_PRCCNT                00000042    D
DSA$GL_TESTNO               0000FE50    D          PCB$W_SIZE                  00000008    D
DSA$GL_UNITS                0000FE0C    D          PCB$W_STATE                 0000002C    D
DSA$GQ_MSGPTR               0000FE68    D          PCB$W_TMBU                  00000032    D
DSA$GT_DEVNAM               0000FE5C    D          PHYBUF                    = 0000000C    D
EXE$ALLOC                   0000000A RG D  02      PHYLEN                    = 00000008    D
```

PR$_IPL	=	00000012	D		UCB$L_DX_BUF		00000098	D
SEARCHDEV		0000010B	R	D 02	UCB$L_DX_RXDB		000000A0	D
SIZ...	=	00000001		D	UCB$L_EMB		00000078	D
SS$_ACCVIO	=	0000000C		D	UCB$L_FIRST		00000014	D
SS$_DEVALLOC	=	00000840		D	UCB$L_FPC		0000000C	D
SS$_DEVALRALLOC	=	00000641		D	UCB$L_FQBL		00000004	D
SS$_DEVNOTALLOC	=	00000858		D	UCB$L_FQFL		00000000	D
SS$_IVDEVNAM	=	00000144		D	UCB$L_FR3		00000010	D
SS$_NOPRIV	=	00000024		D	UCB$L_FR4		00000014	D
SS$_NORMAL	=	00000001		D	UCB$L_IOQBL		00000044	D
UCB$B_AMOD		00000053		D	UCB$L_IOQFL		00000040	D
UCB$B_CEX		00000077		D	UCB$L_IRP		0000004C	D
UCB$B_CM1		0000004A		D	UCB$L_LINK		0000002C	D
UCB$B_CM2		0000004B		D	UCB$L_LOGADR		00000064	D
UCB$B_DEVCLASS		00000038		D	UCB$L_MAXBLOCK		00000084	D
UCB$B_DEVTYPE		00000039		D	UCB$L_MB_MBX		0000007C	D
UCB$B_DIPL		00000052		D	UCB$L_MB_PORT		0000008C	D
UCB$B_DX_SCTCNT		000000A6		D	UCB$L_MB_RAST		00000078	D
UCB$B_ERTCNT		00000070		D	UCB$L_MB_SHB		00000080	D
UCB$B_ERTMAX		00000071		D	UCB$L_MB_WAST		00000074	D
UCB$B_FEX		00000076		D	UCB$L_MB_WIOQBL		00000088	D
UCB$B_FIPL		0000000B		D	UCB$L_MB_WIOQFL		00000084	D
UCB$B_LOCSRV		0000003C		D	UCB$L_MEDIA		0000008C	D
UCB$B_OFFNDX		00000094		D	UCB$L_NT_DATSSB		00000074	D
UCB$B_OFFRTC		00000095		D	UCB$L_NT_INTSSB		00000078	D
UCB$B_REMSRV		0000003D		D	UCB$L_OPCNT		00000060	D
UCB$B_SECTORS		0000003C		D	UCB$L_OWNUIC		0000001C	D
UCB$B_SLAVE		00000074		D	UCB$L_PID		00000028	D
UCB$B_SPR		00000075		D	UCB$L_RQBL		00000004	D
UCB$B_STATE		00000052		D	UCB$L_RQFL		00000000	D
UCB$B_TRACKS		0000003D		D	UCB$L_SVAPTE		00000068	D
UCB$B_TT_CRFILL		0000009D		D	UCB$L_SVPN		00000064	D
UCB$B_TT_DECRF		000000A1		D	UCB$L_TT_DECHAR		000000A8	D
UCB$B_TT_DELFF		000000A2		D	UCB$L_TT_RDUE		0000008C	D
UCB$B_TT_DESPEE		000000A0		D	UCB$L_TT_RTIMOU		000000B8	D
UCB$B_TT_DETYPE		000000A4		D	UCB$L_VCB		00000030	D
UCB$B_TT_LFFILL		0000009E		D	UCB$T_PARTNER		0000000C	D
UCB$B_TT_SPEED		0000009C		D	UCB$W_BCNT		0000006E	D
UCB$B_TYPE		0000000A		D	UCB$W_BCR		00000096	D
UCB$B_VERTSZ		0000003F		D	UCB$W_BOFF		0000006C	D
UCB$C_LENGTH		00000074		D	UCB$W_BUFQUO		00000018	D
UCB$C_MB_LENGTH		00000090		D	UCB$W_BYTESTOGO		0000003E	D
UCB$C_TT_LENGTH		000000BC		D	UCB$W_CHARGE		0000004A	D
UCB$K_LENGTH		00000074		D	UCB$W_CYLINDERS		0000003E	D
UCB$K_MB_LENGTH		00000090		D	UCB$W_DA		0000008C	D
UCB$K_TT_LENGTH		000000BC		D	UCB$W_DC		0000008E	D
UCB$L_AMB		00000054		D	UCB$W_DEVBUFSIZ		0000003A	D
UCB$L_ASTQBL		00000010		D	UCB$W_DEVSTS		0000005A	D
UCB$L_ASTQFL		0000000C		D	UCB$W_DIRSEQ		00000088	D
UCB$L_CPID		0000005C		D	UCB$W_DSTADDR		00000018	D
UCB$L_CRB		00000020		D	UCB$W_DX_BCR		000000A4	D
UCB$L_DDB		00000024		D	UCB$W_EC1		00000090	D
UCB$L_DEVCHAR		00000034		D	UCB$W_EC2		00000092	D
UCB$L_DEVDEPEND		0000003C		D	UCB$W_ERRCNT		00000072	D
UCB$L_DPC		00000080		D	UCB$W_FUNC		0000007E	D
UCB$L_DUETIM		0000005C		D	UCB$W_MB_SEED		00000000	D
UCB$L_DX_BFPNT		0000009C		D	UCB$W_MSGCNT		00000016	D

```
UCB$W_MSGMAX                    00000014    D
UCB$W_NT_CHAN                   0000007C    D
UCB$W_OFFSET                    0000008A    D
UCB$W_REFC                      00000050    D
UCB$W_SIZE                      00000008    D
UCB$W_SRCADDR                   0000001A    D
UCB$W_STS                       00000058    D
UCB$W_TT_DESIZE                 000000A5    D
UCB$W_UNIT                      00000048    D
UCB$W_VPROT                     0000001A    D
UNLOCK                          00000123 R  D  02

                                          +----------------+
                                          ! Psect synopsis !
                                          +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           0000FE70  (65136.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
SEP                             00000161  (  353.)  02 (  2.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD    WRT NOVEC LONG
```

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                      VALUE        DEFINITION      REFERENCES...
------                      -----        ----------      -------------
$$ARGS                     =00000002     59     (1)      58     (1)      59     (1)
$$T1                       =0000000C     59     (1)      58     (1)      59     (1)
$ER                        =00000001     150    (1)
$MODULE                     00000000-R   150    (1)
ALACMODE                   =00000010     80     (1)      205    (1)
ALLOC$_ACMODE              =00000010     58     (1)
ALLOC$_DEVNAM              =00000004     58     (1)
ALLOC$_NARGS               =00000004     58     (1)
ALLOC$_PHYBUF              =0000000C     58     (1)
ALLOC$_PHYLEN              =00000008     58     (1)
DALLOC$_ACMODE             =00000008     59     (1)
DALLOC$_DEVNAM             =00000004     59     (1)
DALLOC$_NARGS              =00000002     59     (1)
DDB$L_DDT                   0000000C                  #-313    (1)
DDB$L_LINK                  00000000                    268    (1)    #-269    (1)
DDB$L_UCB                   00000004                    271    (1)
DDT$L_CANCEL                0000000C                  #-318    (1)
DECREF                      00000126-R   307    (1)      266    (1)      281    (1)
DEV$V_ALL                  =00000017                  #-196    (1)    #-203    (1)    #-278    (1)    #-308    (1)
DEV$V_SPL                  =00000006                  #-185    (1)
DEVNAM                     =00000004     86     (1)   #-292    (1)
DLACMODE                   =00000008     87     (1)
DS$AX_SOFTPCB               00000000-XR                 152    (1)      257    (1)
EXE$ALLOC                   0000000A-R   151    (1)
EXE$DALLOC                  000000A6-R   256    (1)
EXE$MAXACMODE               00000000-XR                 206    (1)
IOC$CVT_DEVNAM              00000000-XR               #-170    (1)
IOC$GL_DEVLIST              00000000-XR                 268    (1)
IOC$SEARCHDEV               00000000-XR               #-260    (1)
IOC$SEARCHGEN               00000000-XR               #-156    (1)
IOC$UNLOCK                  00000000-XR               #-209    (1)    #-300    (1)
NOPRIV                      0000011B-R   297    (1)   #-189    (1)
NORMAL                      00000120-R   299    (1)   #-267    (1)    #-270    (1)
PCB$L_PID                   00000060                  #-180    (1)    #-202    (1)    #-262    (1)    #-274    (1)
PHYBUF                     =0000000C     79     (1)   #-163    (1)
PHYLEN                     =00000008     78     (1)   #-168    (1)
PR$_IPL                    =00000012                  #-316    (1)    #-323    (1)
SEARCHDEV                   0000010B-R   291    (1)   #-154    (1)    #-258    (1)
SS$_ACCVIO                 =0000000C                  #-191    (1)    #-294    (1)
SS$_DEVALLOC               =00000840                  #-182    (1)
SS$_DEVALRALLOC            =00000641                  #-197    (1)
SS$_DEVNOTALLOC            =00000858                  #-283    (1)
SS$_IVDEVNAM               =00000144                  #-153    (1)
SS$_NOPRIV                 =00000024                  #-297    (1)
SS$_NORMAL                 =00000001                  #-194    (1)    #-299    (1)
UCB$B_AMOD                  00000053                  #-207    (1)    #-276    (1)
UCB$B_FIPL                  0000000B                  #-316    (1)
UCB$L_DDB                   00000024                  #-312    (1)
UCB$L_DEVCHAR               00000034                    185    (1)      196    (1)      203    (1)      278    (1)
                                                        308    (1)
```

```
UCB$L_IRP                       0000004C                 #-317    (1)
UCB$L_LINK                      0000002C                  271     (1)     #-272    (1)
UCB$L_PID                       00000028                 #-178    (1)     #-202    (1)     #-262    (1)     #-274    (1)
                                                         #-310    (1)
UCB$W_REFC                      00000050                 #-187    (1)     #-204    (1)     #-264    (1)     #-279    (1)
                                                         #-309    (1)
UNLOCK                          00000123-R  300    (1)   #-284    (1)     #-295    (1)     #-298    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ALLOCDEF	1	58 (1)	58 (1)
$DALLOCDEF	1	59 (1)	59 (1)
$DDBDEF	1	60 (1)	60 (1)
$DDTDEF	1	61 (1)	61 (1)
$DEFINI	1	60 (1)	60 (1) 61 (1) 62 (1) 63 (1) 64 (1)
			65 (1) 66 (1) 67 (1) 68 (1) 70 (1)
$DEVDEF	1	62 (1)	62 (1)
$DS_DSADEF	5	70 (1)	70 (1)
$OFFDEF	1	58 (1)	58 (1) 59 (1)
$PCBDEF	4	63 (1)	63 (1)
$PRDEF	5	64 (1)	64 (1)
$PRVDEF	4	65 (1)	65 (1)
$PSLDEF	2	66 (1)	66 (1)
$SSDEF	21	67 (1)	67 (1)
$UCBDEF	10	68 (1)	68 (1)
DSBINT	1	316 (1)	316 (1)
ENBINT	1	323 (1)	323 (1)
IFNORD	1	295 (1)	295 (1)
MODNAM	1	150 (1)	150 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL2	00C0	320 (1)
BBC	00E1	185 (1) 196 (1) 278 (1)
BBCC	00E5	308 (1)
BBS	00E0	319 (1)
BBSS	00E2	203 (1)
BEQL	0013	155 (1) 164 (1) 166 (1) 169 (1) 173 (1) 179 (1) 181 (1)
		259 (1) 270 (1) 273 (1) 293 (1) 295 (1)
BGTR	0014	277 (1)
BLBC	00E9	158 (1) 261 (1)
BNEQ	0012	188 (1) 195 (1) 263 (1) 265 (1) 275 (1) 280 (1)
BRB	0011	183 (1) 186 (1) 190 (1) 192 (1) 267 (1) 282 (1) 284 (1)
		298 (1)
BRW	0031	189 (1) 209 (1) 300 (1)
BSBB	0010	258 (1)
BSBW	0030	154 (1) 156 (1) 170 (1) 260 (1)
CLRL	00D4	310 (1) 315 (1)
CMPB	0091	276 (1)
CMPL	00D1	180 (1) 262 (1) 274 (1)
CMPW	00B1	194 (1) 264 (1) 279 (1)
DECW	00B7	309 (1)
EXTZV	00EF	205 (1)
INCW	00B6	204 (1)
JSB	0016	206 (1) 266 (1) 281 (1) 322 (1)
MFPR	00DB	316 (1)
MOVAB	009E	152 (1) 257 (1) 268 (1) 271 (1)
MOVB	0090	207 (1)
MOVL	00D0	157 (1) 159 (1) 163 (1) 167 (1) 168 (1) 171 (1) 178 (1)
		202 (1) 208 (1) 269 (1) 272 (1) 292 (1) 312 (1) 313 (1)
		314 (1) 317 (1) 318 (1)
MOVW	00B0	174 (1)
MOVZWL	003C	153 (1) 165 (1) 182 (1) 191 (1) 197 (1) 283 (1) 294 (1)
		297 (1) 299 (1)
MTPR	00DA	316 (1) 323 (1)
POPR	00BA	324 (1)
PROBER	000C	295 (1)
PUSHR	00BB	311 (1)
RSB	0005	296 (1) 325 (1)
SUBL	00C2	321 (1)
TSTL	00D5	172 (1)
TSTW	00B5	187 (1)

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	150	(1)														
.END	404	(1)														
.ENDC	316	(1)	323	(1)												
.ENDM	150	(1)	295	(1)	316	(1)	323	(1)	58	(1)	59	(1)	60	(1)	61	(1)
	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)	70	(1)
.ENTRY	151	(1)	256	(1)												
.IDENT	5	(1)														
.IF	316	(1)	323	(1)												
.IFF	316	(1)	323	(1)												
.IRP	58	(1)	59	(1)												
.LIST	58	(1)	59	(1)	70	(1)										
.NLIST	58	(1)	59	(1)	70	(1)										
.NOCROSS	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)
	68	(1)	70	(1)												
.PAGE	210	(1)	285	(1)	301	(1)	327	(1)	364	(1)	88	(1)				
.PSECT	149	(1)														
.RESTORE	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)
	68	(1)	70	(1)												
.SAVE	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)
	68	(1)	70	(1)												
.SBTTL	211	(1)	286	(1)	302	(1)	328	(1)	365	(1)	89	(1)				
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	4	#-163 (1) #-168 (1) 205 (1) #-292 (1)
R0	19	#-157 (1) #-165 (1) #-171 (1) #-178 (1) #-180 (1) #-205 (1)
		#-207 (1) #-208 (1) #-261 (1) #-283 (1) #-294 (1) #-297 (1)
		#-299 (1) #-312 (1) #-313 (1) #-318 (1) #-320 (1) #-321 (1)
R1	29	#-159 (1) #-163 (1) #-165 (1) #-167 (1) #-174 (1) #-262 (1)
		#-264 (1) #-271 (1) #-272 (1) #-274 (1) #-276 (1) 278 (1)
		#-279 (1) #-292 (1) 295 (1) 308 (1) #-309 (1) #-310 (1)
		#-311 (1) #-312 (1) #-314 (1) #-318 (1) 319 (1) #-320 (1)
		#-321 (1) 322 (1) #-324 (1)
R2	3	151 (1) 256 (1) #-315 (1)
R3	12	151 (1) #-168 (1) #-172 (1) #-174 (1) 256 (1) #-268 (1)
		#-269 (1) 271 (1) #-311 (1) #-317 (1) #-324 (1)
R4	8	151 (1) #-152 (1) #-180 (1) #-202 (1) 256 (1) #-257 (1)
		#-262 (1) #-274 (1)
R5	17	151 (1) #-153 (1) #-157 (1) #-158 (1) #-171 (1) #-182 (1)
		#-191 (1) #-194 (1) #-197 (1) #-208 (1) 256 (1) #-276 (1)
		#-311 (1) #-314 (1) #-316 (1) #-317 (1) #-324 (1)
R6	10	151 (1) #-159 (1) #-178 (1) 185 (1) #-187 (1) 196 (1)
		#-202 (1) 203 (1) #-204 (1) #-207 (1)
SP	2	#-316 (1) #-323 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	44	00:00:00.03	00:00:00.21
Command processing	905	00:00:00.25	00:00:00.73
Pass 1	782	00:00:02.73	00:00:04.05
Symbol table sort	0	00:00:00.30	00:00:00.30
Pass 2	129	00:00:00.72	00:00:01.43
Symbol table output	2	00:00:00.05	00:00:00.12
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	5	00:00:00.13	00:00:00.24
Assembler run totals	1871	00:00:04.21	00:00:07.08

The working set limit was 2400 pages.
159281 bytes (312 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1055 non-local and 20 local symbols.
404 source lines were read in Pass 1, producing 73 object records in Pass 2.
71 pages of virtual memory were used to define 27 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              7
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                1
SYS$COMMON:[SYSLIB]LIB.MLB;2                           3
SYS$COMMON:[SYSLIB]STARLET.MLB;2                      11
TOTALS (all libraries)                                22

1331 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DEVALC.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(3) 65 DS$GPHARD Retreive address of P-table
(4) 120 DSP$GEN_PTABLES Setup UUT's to test

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DEVICE.MAR
0000     2 ; *1     6-FEB-1986 15:16:57 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element DEVICE.MAR
0000     4         .TITLE  DEVICE  *** DEVICE Handle PTABLEs
0000     5         .IDENT  /06-01/
0000     6         .LIST   MEB
0000     7         .NLIST  CND
0000     8         .DSABL  GBL
0000     9
0000    10 ;
0000    11 ; COPYRIGHT (C) 1977, 1980
0000    12 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    13 ;
0000    14 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    15 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    16 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    17 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    18 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    19 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    20 ; REMAIN IN DEC.
0000    21 ;
0000    22 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    23 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    24 ; CORPORATION.
0000    25 ;
0000    26 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    27 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    28
0000    29 ;++
0000    30 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ; ENVIRONMENT:
0000    35 ;
0000    36 ; AUTHOR:       R. RIGGS        15-JAN-79       VERSION 01.
0000    37 ;               Dave Butenhof, 17-feb-1981, version 6.3
0000    38 ;       01      Fix linker truncation errors; change DSA$x_? references
0000    39 ;               to longword relative.
0000    40 ;--
```

```
                        0000     42 ;
                        0000     43 ; MACRO LIBRARIES:
                        0000     44 ;
                        0000     45
                        0000     46
                00000001 0000    47 SS$_NORMAL=1
                        0000     51          $DS_HPODEF
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
                00000008 0000                .IIF    NB,.BLKQ,       .BLKQ   1
                        0008                 .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
                0000000A 0008                .IIF    NB,.BLKW,       .BLKW   1
                        000A                 .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
                0000000B 000A                .IIF    NB,.BLKB,       .BLKB   1
                        000B                 .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
                0000000C 000B                .IIF    NB,.BLKB,       .BLKB   1
                        000C                 .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
                00000018 000C                .IIF    NB,.BLKB,       .BLKB   12
                        0018                 .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
                0000001C 0018                .IIF    NB,.BLKL,       .BLKL   1
                        001C                 .IIF    NB,HP$A_DVA,    HP$A_DVA:
                00000020 001C                .IIF    NB,.BLKL,       .BLKL   1
                        0020                 .IIF    NB,HP$A_LINK,   HP$A_LINK:
                00000024 0020                .IIF    NB,.BLKL,       .BLKL   1
                        0024                 .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
                00000026 0024                .IIF    NB,.BLKW,       .BLKW   1
                        0026                 .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
                00000032 0026                .IIF    NB,.BLKB,       .BLKB   12
                        0032                 .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                        0000                 .RESTORE
                        0000     52          $DS_HDRDEF
                        0000     53          $DS_DSADEF
                        0000     54          $DS_DSDEF
                        0000     58
                        0000     59          .EXTRN  DS$GA_SELECTS, DS$GA_SELECTED, DS$GA_PTABLE
                        0000     60          .EXTRN  DS$PRINTF,      DS_ERRSUP,      DS$GPHARD
                        0000     61
                00000000 0000    62          .PSECT  DATA, SHR,NOEXE, NOWRT, BYTE
                        0000     63          MODNAM  <DEVICE>
45 43 49 56 45 44 00' 0000       $MODULE:            .ASCIC  \DEVICE\
                        06  0000
```

```
                     0007    65          .SBTTL  DS$GPHARD       Retreive address of P-table
                  00000000   66          .PSECT  CODE, SHR, NOWRT, EXE, BYTE
                     0000    67 ;++
                     0000    68 ; FUNCTIONAL DESCRIPTION:
                     0000    69 ;
                     0000    70 ;       Given a logical unit number, This routine returns the base
                     0000    71 ;       address of the associated P-table.
                     0000    72 ;
                     0000    73 ; CALLING SEQUENCE:
                     0000    74 ;
                     0000    75 ;       DS$GPHARD(UNIT,%REF(ADDRESS))
                     0000    76 ;
                     0000    77 ; INPUT PARAMETERS:
                     0000    78 ;
                     0000    79 ;       4(AP)   Logical unit number
                     0000    80 ;
                     0000    81 ; IMPLICIT INPUTS:
                     0000    82 ;
                     0000    83 ;       DS$GA_SELECTED  List of P-tables of units to be tested.
                     0000    84 ;
                     0000    85 ; OUTPUT PARAMETERS:
                     0000    86 ;
                     0000    87 ;       8(AP)   Address of longword to receive the address for this unit
                     0000    88 ;
                     0000    89 ; IMPLICIT OUTPUTS:
                     0000    90 ;
                     0000    91 ;       R1      Same as 8(AP)
                     0000    92 ;
                     0000    93 ; COMPLETION CODES:
                     0000    94 ;
                     0000    95 ;       SS$_NORMAL      If unit number was in range
                     0000    96 ;       DS$_ERROR       If unit number was too large (unsigned)
                     0000    97 ;
                     0000    98 ; SIDE EFFECTS:
                     0000    99 ;
                     0000   100 ;       R1       Returns with address of specified P-table
                     0000   101 ;--
                     0000   102
                0004 0000   103 .ENTRY  DSX$GPHARD,^M<R2>
                     0002   104
51   00000000'EF   DE  0002   105          MOVAL   DS$GA_SELECTED,R1       ; Address table
 50    04 AC      01  C1  0009   106          ADDL3   #1,4(AP),R0             ; Change to origin 1
          61      50  D1  000E   107          CMPL    R0,(R1)                 ; Compare with length of select table
                  17  1A  0011   108          BGTRU   10$                     ; Branch if unit too large for table
0000FE0C'EF       50  D1  0013   109          CMPL    R0,L^DSA$GL_UNITS       ; Compare with number of units tested
                  0E  1A  001A   110          BGTRU   10$                     ; Branch if unit too large
        51    6140  D0  001C   111          MOVL    (R1)[R0],R1             ; Get address of P-table
       08 BC      51  D0  0020   112          MOVL    R1,@8(AP)               ; Store where user wants it
                  04  13  0024   113          BEQL    10$                     ; Branch if bogus
          50      01  D0  0026   114          MOVL    S^#SS$_NORMAL,R0        ; Set success
                  04  0029   115          RET                             ; Return to user
                      002A   116 10$:
50   00660002 8F      D0  002A   117          MOVL    #DS$_ERROR,R0           ; Spaz
                  04  0031   118          RET
```

```
                     0032   120          .SBTTL  DSP$GEN_PTABLES Setup UUT's to test
                00000032   121          .PSECT  CODE, SHR, NOWRT, EXE, BYTE
                     0032   122 ;++
                     0032   123 ; FUNCTIONAL DESCRIPTION:
                     0032   124 ;
                     0032   125 ;       This routine compares the unit types selected by the select
                     0032   126 ;       command with the device types the current program tests
                     0032   127 ;       and enters them in DS$GA_SELECTED.
                     0032   128 ;
                     0032   129 ; CALLING SEQUENCE:
                     0032   130 ;
                     0032   131 ;       CALLG   #0,DSP$GEN_PTABLES
                     0032   132 ;
                     0032   133 ; INPUT PARAMETERS:     NONE
                     0032   134 ;
                     0032   135 ; IMPLICIT INPUTS:
                     0032   136 ;
                     0032   137 ;       DS$GA_PTABLES, DS$GA_SELECTS
                     0032   138 ;       PROGRAM HEADER
                     0032   139 ;
                     0032   140 ; OUTPUT PARAMETERS:    NONE
                     0032   141 ;
                     0032   142 ; IMPLICIT OUTPUTS:
                     0032   143 ;
                     0032   144 ;       The P-TABLE's for the program
                     0032   145 ;
                     0032   146 ;--
```

```
                        007C  0032   148 .ENTRY  DSP$GEN_PTABLES,^M<R2,R3,R4,R5,R6>
                              0034   149
              0000FE0C'EF   D4  0034   150         CLRL    L^DSA$GL_UNITS          ; Clear count of units to test
              00000220'EF   D5  003A   151         TSTL    L$L_UNIT
                         65 13  0040   152         BEQL    60$                     ; EXIT IF NOT TESTING ANY UNITS
          56   00000000'EF   DE  0042   153         MOVAL   DS$GA_PTABLE,R6         ; Point to P-table list
                    55   66  D0  0049   154         MOVL    (R6),R5                 ; Get count of entries
       50 00000000'EF   55  E1  004C   155 10$:     BBC     R5,DS$GA_SELECTS,50$    ; Branch if not selected
                              0054   156
          54   0000021C 9F  D0  0054   157         MOVL    a#L$A_DEVP,R4           ; Get address of DEVTYP in program
                         4A 13  005B   158         BEQL    60$                     ; Branch if none
                    53   84  D0  005D   159         MOVL    (R4)+,R3                ; Get count of entries
                         45 13  0060   160         BEQL    60$                     ; Branch if none
                              0062   161 20$:
                    51   84  D0  0062   162         MOVL    (R4)+,R1                ; Get address of ASCIC devtype
                    50   81  9A  0065   163         MOVZBL  (R1)+,R0                ; Get length of devtype
                 52   6645  D0  0068   164         MOVL    (R6)[R5],R2             ; Get address of P-table
                         36 13  006C   165         BEQL    50$                     ; Branch if no P-table
                52   26 A2  9E  006E   166         MOVAB   HP$T_TYPE(R2),R2        ; Point to type
                    50   82  91  0072   167         CMPB    (R2)+,R0                ; Compare string lengths
                         2A 12  0075   168         BNEQ    40$                     ; Branch if no match
                    82   81  91  0077   169 30$:    CMPB    (R1)+,(R2)+             ; Compare
                         25 12  007A   170         BNEQ    40$                     ; Branch if not the same
                      F8 50  F5  007C   171         SOBGTR  R0,30$                  ; Branch if more to compare
       50   0000FE0C'EF   01  C1  007F   172         ADDL3   #1,L^DSA$GL_UNITS,R0    ; Get next unit number
            00000220'EF   50  D1  0087   173         CMPL    R0,L$L_UNIT             ; Program test enough units?
                         17 14  008E   174         BGTR    60$                     ; Already selected enough units
              0000FE0C'EF   D6  0090   175         INCL    L^DSA$GL_UNITS          ; Update units being tested
     00000000'EF40   6645  D0  0096   176         MOVL    (R6)[R5],DS$GA_SELECTED[R0] ; Insert P-table as new unit
                         03 11  009F   177         BRB     50$                     ; Branch since match found
                              00A1   178 40$:
          BE 53        F5  00A1   179         SOBGTR  R3,20$                  ; Count and continue in DEVTYP list
                              00A4   180 50$:
          A5 55        F5  00A4   181         SOBGTR  R5,10$                  ; Next P-table
                              00A7   182 60$:
                    04        00A7   183         RET                             ; Return
                              00A8   184
                              00A8   185         .END
```

```
$ER                             = 00000001    D
$MODULE                           00000000 R  D  02
BIT...                          = 00660160    D
DS$GA_PTABLE                      ********    X  00
DS$GA_SELECTED                    ********    X  00
DS$GA_SELECTS                     ********    X  00
DS$GPHARD                         ********    X  00
DS$K_ERROR                      = 00000002    D
DS$K_NORMAL                     = 00000001    D
DS$K_SEVERE                     = 00000004    D
DS$K_SUBSYS                     = 00000066    D
DS$K_WARNING                    = 00000000    D
DS$PRINTF                         ********    X  00
DS$_AP_NORMAL_BREAK             = 00660150    D
DS$_ARITH                       = 006600D0    D
DS$_ASBE                        = 00660118    D
DS$_BADLINK                     = 006600F0    D
DS$_BADTYPE                     = 006600E8    D
DS$_BIIC                        = 00660120    D
DS$_CHME                        = 006600A8    D
DS$_CHMK                        = 006600E0    D
DS$_DEVNAME                     = 00660108    D
DS$_ERROR                       = 00660002    D
DS$_FHWE                        = 00660068    D
DS$_FRAGBUF                     = 00660080    D
DS$_ICBUSY                      = 006600C8    D
DS$_ICERR                       = 006600C0    D
DS$_IHWE                        = 00660060    D
DS$_ILLCHAR                     = 00660018    D
DS$_ILLPAGCNT                   = 00660078    D
DS$_ILLUNIT                     = 00660100    D
DS$_INIT_FAIL                   = 00660140    D
DS$_INSFMEM                     = 00660050    D
DS$_INVCPU                      = 00660130    D
DS$_IPL2HI                      = 006600B8    D
DS$_IVADDR                      = 00660040    D
DS$_IVVECT                      = 00660038    D
DS$_KRNLSTK                     = 00660090    D
DS$_LOGIC                       = 00660070    D
DS$_MCHK                        = 00660088    D
DS$_MEM_ALLOC_ERR               = 00660138    D
DS$_MMOFF                       = 00660058    D
DS$_NEEDUNIT                    = 006600F8    D
DS$_NODE                        = 00660128    D
DS$_NOPCS                       = 00660110    D
DS$_NORMAL                      = 00660001    D
DS$_NOSUPPORT                   = 006600B1    D
DS$_NOTALLOWMP                  = 00660148    D
DS$_NOTDON                      = 00660030    D
DS$_NOTIMP                      = 006600B0    D
DS$_NULLSTR                     = 00660010    D
DS$_OVERFLOW                    = 00660008    D
DS$_POWER                       = 00660098    D
DS$_PROGERR                     = 00660020    D
DS$_SEVERE                      = 00660004    D
DS$_TRANSL                      = 006600A0    D
DS$_TRUNCATE                    = 00660028    D
DS$_UNEXPINT                    = 006600D8    D
DS$_UNSUPPDEV                   = 00660158    D
DS$_VASFULL                     = 00660048    D
DS$_WARNING                     = 00660000    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSP$GEN_PTABLES                   00000032 RG D  03
DSX$GPHARD                        00000000 RG D  03
DS_ERRSUP                         ********    X  00
HP$A_DEPENDENT                    00000032    D
HP$A_DEVICE                       00000018    D
HP$A_DVA                          0000001C    D
HP$A_LINK                         00000020    D
HP$B_DRIVE                        0000000B    D
HP$B_FLAGS                        0000000A    D
HP$Q_DEVICE                       00000000    D
HP$T_DEVICE                       0000000C    D
HP$T_TYPE                         00000026    D
HP$W_SIZE                         00000008    D
HP$W_VECTOR                       00000024    D
L$A_CCP                           00000240    D
L$A_DEVP                          0000021C    D
L$A_DREG                          00000224    D
L$A_DTP                           00000218    D
L$A_ICP                           0000023C    D
L$A_LASTADR                       00000214    D
L$A_NAME                          00000208    D
L$A_REPP                          00000244    D
L$A_SECNAM                        00000250    D
L$A_STATAB                        00000248    D
L$A_TSTCNT                        00000254    D
L$L_ENVIRON                       00000204    D
L$L_ERRTYP                        0000024C    D
L$L_HEADLENGTH                    00000200    D
L$L_REV                           0000020C    D
L$L_UNIT                          00000220    D
L$L_UNUSED                        00000228    D
L$L_UPDATE                        00000210    D
SIZ...                          = 00000001    D
SS$_NORMAL                      = 00000001    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000007 (7.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	000000A8 (168.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	63 (2)	
$MODULE	00000000-R	63 (2)	
BIT...	=00660160	54 (2)	54 (2)
DS$GA_PTABLE	00000000-XR		153 (5) 59 (2)
DS$GA_SELECTED	00000000-XR		105 (3) #-176 (5) 59 (2)
DS$GA_SELECTS	00000000-XR		155 (5) 59 (2)
DS$GPHARD	00000000-XR		60 (2)
DS$K_ERROR	=00000002	54 (2)	
DS$K_NORMAL	=00000001	54 (2)	
DS$K_SEVERE	=00000004	54 (2)	
DS$K_SUBSYS	=00000066	54 (2)	54 (2)
DS$K_WARNING	=00000000	54 (2)	
DS$PRINTF	00000000-XR		60 (2)
DS$_AP_NORMAL_BREAK	=00660150	54 (2)	
DS$_ARITH	=006600D0	54 (2)	
DS$_ASBE	=00660118	54 (2)	
DS$_BADLINK	=006600F0	54 (2)	
DS$_BADTYPE	=006600E8	54 (2)	
DS$_BIIC	=00660120	54 (2)	
DS$_CHME	=006600A8	54 (2)	
DS$_CHMK	=006600E0	54 (2)	
DS$_DEVNAME	=00660108	54 (2)	
DS$_ERROR	=00660002	54 (2)	#-117 (3)
DS$_FHWE	=00660068	54 (2)	
DS$_FRAGBUF	=00660080	54 (2)	
DS$_ICBUSY	=006600C8	54 (2)	
DS$_ICERR	=006600C0	54 (2)	
DS$_IHWE	=00660060	54 (2)	
DS$_ILLCHAR	=00660018	54 (2)	
DS$_ILLPAGCNT	=00660078	54 (2)	
DS$_ILLUNIT	=00660100	54 (2)	
DS$_INIT_FAIL	=00660140	54 (2)	
DS$_INSFMEM	=00660050	54 (2)	
DS$_INVCPU	=00660130	54 (2)	
DS$_IPL2HI	=006600B8	54 (2)	
DS$_IVADDR	=00660040	54 (2)	
DS$_IVVECT	=00660038	54 (2)	
DS$_KRNLSTK	=00660090	54 (2)	
DS$_LOGIC	=00660070	54 (2)	
DS$_MCHK	=00660088	54 (2)	
DS$_MEM_ALLOC_ERR	=00660138	54 (2)	
DS$_MMOFF	=00660058	54 (2)	
DS$_NEEDUNIT	=006600F8	54 (2)	
DS$_NODE	=00660128	54 (2)	
DS$_NOPCS	=00660110	54 (2)	
DS$_NORMAL	=00660001	54 (2)	
DS$_NOSUPPORT	=006600B1	54 (2)	
DS$_NOTALLOWMP	=00660148	54 (2)	
DS$_NOTDON	=00660030	54 (2)	
DS$_NOTIMP	=006600B0	54 (2)	54 (2)

```
DS$_NULLSTR                     =00660010      54      (2)
DS$_OVERFLOW                    =00660008      54      (2)
DS$_POWER                       =00660098      54      (2)
DS$_PROGERR                     =00660020      54      (2)
DS$_SEVERE                      =00660004      54      (2)
DS$_TRANSL                      =006600A0      54      (2)
DS$_TRUNCATE                    =00660028      54      (2)
DS$_UNEXPINT                    =006600D8      54      (2)
DS$_UNSUPPDEV                   =00660158      54      (2)
DS$_VASFULL                     =00660048      54      (2)
DS$_WARNING                     =00660000      54      (2)     54      (2)
DSA$GL_UNITS                     0000FE0C                      #-109   (3)     #-150   (5)     #-172   (5)     #-175   (5)
DSP$GEN_PTABLES                  00000032-R    148     (5)
DSX$GPHARD                       00000000-R    103     (3)
DS_ERRSUP                        00000000-XR                   60      (2)
HP$T_TYPE                        00000026                      166     (5)
L$A_DEVP                         0000021C                      #-157   (5)
L$L_UNIT                         00000220                      #-151   (5)     #-173   (5)
SS$_NORMAL                      =00000001      47      (2)     #-114   (3)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEF            1           54      (2)
$DEFINI         1           51      (2)     51      (2)     52      (2)     53      (2)
$DS_DSADEF      5           53      (2)     53      (2)
$DS_DSDEF       3           54      (2)     54      (2)
$DS_HDRDEF      2           52      (2)     52      (2)
$DS_HPODEF      2           51      (2)     51      (2)
$EQU            1           54      (2)     54      (2)
$EQULS1         1           54      (2)     54      (2)
$EQULST         1           54      (2)     54      (2)
$GBLINI         2                           54      (2)
$VIELD1         1           54      (2)
MODNAM          1           63      (2)     63      (2)
```

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDL3           00C1      106     (3)     172     (5)
BBC             00E1      155     (5)
BEQL            0013      113     (3)     152     (5)     158     (5)     160     (5)     165     (5)
BGTR            0014      174     (5)
BGTRU           001A      108     (3)     110     (3)
BNEQ            0012      168     (5)     170     (5)
BRB             0011      177     (5)
CLRL            00D4      150     (5)
CMPB            0091      167     (5)     169     (5)
CMPL            00D1      107     (3)     109     (3)     173     (5)
INCL            00D6      175     (5)
MOVAB           009E      166     (5)
MOVAL           00DE      105     (3)     153     (5)
MOVL            00D0      111     (3)     112     (3)     114     (3)     117     (3)     154     (5)     157     (5)     159     (5)
                          162     (5)     164     (5)     176     (5)
MOVZBL          009A      163     (5)
RET             0004      115     (3)     118     (3)     183     (5)
SOBGTR          00F5      171     (5)     179     (5)     181     (5)
TSTL            00D5      151     (5)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ASCIC          63      (2)
.DSABL          8       (1)
.END            185     (5)
.ENDC           54      (2)
.ENDM           51      (2)     52      (2)     53      (2)     54      (2)     63      (2)
.ENDR           54      (2)
.ENTRY          103     (3)     148     (5)
.EXTRN          59      (2)     60      (2)
.IDENT          5       (1)
.IF             54      (2)
.IFF            54      (2)
.IIF            54      (2)
.IRP            54      (2)
.LIST           50      (2)     52      (2)     53      (2)     56      (2)     57      (2)     6       (1)
.MACRO          54      (2)
.MDELETE        54      (2)
.NLIST          48      (2)     49      (2)     52      (2)     53      (2)     55      (2)     7       (1)
.NOCROSS        51      (2)     52      (2)     53      (2)
.PSECT          121     (4)     62      (2)     66      (3)
.RESTORE        51      (2)     52      (2)     53      (2)
.SAVE           51      (2)     52      (2)     53      (2)
.SBTTL          120     (4)     65      (3)
.TITLE          4       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	2	#-106 (3) #-112 (3)
R0	12	#-106 (3) #-107 (3) #-109 (3) #-111 (3) #-114 (3) #-117 (3) #-163 (5) #-167 (5) #-171 (5) #-172 (5) #-173 (5) #-176 (5)
R1	8	#-105 (3) #-107 (3) #-111 (3) #-112 (3) #-162 (5) #-163 (5) #-169 (5)
R2	7	103 (3) 148 (5) #-164 (5) 166 (5) #-167 (5) #-169 (5)
R3	3	148 (5) #-159 (5) #-179 (5)
R4	4	148 (5) #-157 (5) #-159 (5) #-162 (5)
R5	6	148 (5) #-154 (5) #-155 (5) #-164 (5) #-176 (5) #-181 (5)
R6	5	148 (5) #-153 (5) #-154 (5) #-164 (5) #-176 (5)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.03	00:00:00.20
Command processing	880	00:00:00.19	00:00:00.70
Pass 1	333	00:00:00.96	00:00:01.96
Symbol table sort	0	00:00:00.04	00:00:00.04
Pass 2	12	00:00:00.21	00:00:00.36
Symbol table output	0	00:00:00.02	00:00:00.05
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	3	00:00:00.08	00:00:00.11
Assembler run totals	1252	00:00:01.54	00:00:03.43

The working set limit was 2900 pages.
53950 bytes (106 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 175 non-local and 7 local symbols.
185 source lines were read in Pass 1, producing 30 object records in Pass 2.
43 pages of virtual memory were used to define 15 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	10

249 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DEVICE.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element DIRECTORY.B32
;    0002  0      ! *12    2-NOV-1987 15:41:42 BAGGETT "FINISHED"
;    0003  0      ! *11    2-NOV-1987 15:21:08 MI "FIXED THE BUG CAUSING OCCASIONAL ACCESS VIOLATION WHILEDOING DIRECTORY"
;    0004  0      !  10B1  2-NOV-1987 15:01:25 BAGGETT "FINISHED"
;    0005  0      ! *10   14-APR-1987 16:30:11 MARTIN "UPDATED FOR PSTAR"
;    0006  0      ! *9     4-DEC-1986 15:35:43 MARTIN "<no reason given>"
;    0007  0      ! *8    20-AUG-1986 09:16:14 MARTIN "<no reason given>"
;    0008  0      ! *7    20-AUG-1986 09:14:29 MARTIN "MADE DOCUMENTATION ADDITION"
;    0009  0      ! *6    18-AUG-1986 10:09:57 MARTIN "MODIFIED MAIN CODE"
;    0010  0      ! *5    23-JUL-1986 10:06:41 MARTIN "AVOID UD. BUG"
;    0011  0      ! *4    28-MAY-1986 10:55:05 SALEM "DONE"
;    0012  0      ! *3    27-MAY-1986 15:13:43 SALEM "ALL DONE"
;    0013  0      ! *2     8-APR-1986 15:40:55 SALEM "DONE"
;    0014  0      ! *1     6-FEB-1986 15:17:24 DS ""
;    0015  0      ! DEC/CMS REPLACEMENT HISTORY, Element DIRECTORY.B32
;    0016  0      %title '*** DIRECTORY Handle DIRECTORY command'
;    0017  0      module directory (                              ! Supervisor DIRECTORY command
;    0018  0                        ident = '10.10-12',                                          !G:12
;    0019  0                        Debug,
;    0020  0                        OptLevel = 3
;    0021  0                        ) =
;    0022  0      !
;    0023  0      ! Copyright (c) 1980, 1982, 1983, 1984, 1986, 1987                           !G:10
;    0024  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0025  0      !
;    0026  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0027  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0028  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0029  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0030  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0031  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0032  0      ! REMAIN IN DEC.
;    0033  0      !
;    0034  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0035  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0036  0      ! CORPORATION.
;    0037  0      !
;    0038  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0039  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0040  0      !
;    0041  0
;    0042  0      !++
;    0043  0      ! FACILITY:
;    0044  0      !       DIAGNOSTIC SUPERVISOR TEST PROGRAM
;    0045  0      !
;    0046  0      ! ABSTRACT:
;    0047  0      !
;    0048  0      !       Read and print directories for supported mass storage volume formats:
;    0049  0      !       ODS-1/2 disks, ANSI magtapes, and RT-11 console media
;    0050  0      !
;    0051  0      !
;    0052  0      ! ENVIRONMENT:  DIAGNOSTIC SUPERVISOR
;    0053  0      !
;    0054  0      ! AUTHOR: Dave Butenhof         1-oct-1980
;    0055  0      !
;    0056  0      ! MODIFIED BY:
;    0057  0      !               Dave Butenhof, 18-mar-1981, version 6.3
```

```
;    0058  0     !        01      Allow ODS subdirectory support under user mode.
;    0059  0     !               -Dave butenhof, 02-Jun-1981, version 6.3
;    0060  0     !        02      Change library specs for more flexibility
;    0061  0     !                - Dave butenhof, 26-Jun-1981, version 6.4
;    0062  0     !        03      Fix to use BOO$QIO instead of DSP$XX_READ (the latter does
;    0063  0     !                not do error retries, nor does it handle errors well).
;    0064  0     !        04      - Dave Butenhof, 31-aug-1981, version 6.5
;    0065  0     !                Support DQ device type (IDC disks) for Nebula.  Disks are
;    0066  0     !                R80 and RI02.
;    0067  0     !        05      - dave butenhof, 16-Sep-1981, version 6.5
;    0068  0     !                Fix directory so unattached device results in 'nice' error
;    0069  0     !                message, not general RMS$_DEV.
;    0070  0     !
;    0071  0     !        06      - Jack Stansbury, 22-Oct-1981, Version 6.-
;    0072  0     !                Fixed a truncation error by changing the PSECT declarations to
;    0073  0     !                conform to the .MAR file's PSECT's.
;    0074  0     !
;    0075  0     !        07      - Dave Butenhof, 26-Oct-1981, Version 6.-
;    0076  0     !                Alter call to LOC$PTABLE to include adapter address
;    0077  0     !                argument.
;    0078  0     !
;    0079  0     !        08      - Dave Butenhof, 12-Jan-1982, Version 6.6
;    0080  0     !                Set up Control C trap in DIRECTORY routines; this will allow
;    0081  0     !                closing of any open files.
;    0082  0     !
;    0083  0     !        09      - Dave Butenhof, 25-Jan-1982, Version 6.6
;    0084  0     !                Use DSR$ALLOCATE_PSEUDO_RPB routine instead of explicit init.
;    0085  0     !
;    0086  0     !        10      - Dave Butenhof, 03-Feb-1982, Version 6.6
;    0087  0     !                Add ability to check filespec against (wildcarded) argument
;    0088  0     !                before typing.  This is almost free, since the checking
;    0089  0     !                routine exists in HELP already for checking keywords.  All
;    0090  0     !                we need here is minimal pre-parsing to add 'implicit
;    0091  0     !                wildcards', and the code to call the compare routine.
;    0092  0     !
;    0093  0     !        11      Dave Butenhof, 15-Apr-1982
;    0094  0     !                Add UDA50 disk support.
;    0095  0     !
;    0096  0     !        12      Dave Butenhof, 24-May-1982
;    0097  0     !                Add TM78 support (by removing special case code which disallows
;    0098  0     !                 TU78)
;    0099  0     !
;    0100  0     !        13      M. Baggett,     9-March-1983, Version 6.11
;    0101  0     !                Added support for CI-HSC50 and RC25/RCF25.
;    0102  0     !
;    0103  0     !        14      Bob Bergazzi    May 17, 1983    Version 6.11
;    0104  0     !                Changed the order of searching the libraries.
;    0105  0     !        15      M. Baggett      11-Oct-1983     Version 6.13
;    0106  0     !                Added support for CI-HSC50 magtape.
;    0107  0     !
;    0108  0     !        16      Bob Bergazzi    April 13, 1984  Version 7.0
;    0109  0     !                Added support for ZZZ console storage device in ODS format.
;    0110  0     !                to DSX$DIRECTORY routine.
;    0111  0     !
;    0112  0     !        17      Bob Bergazzi    May 21, 1984    Version 7.0
;    0113  0     !                Added support for DIR/WIDE, which displays the directory
;    0114  0     !                1 filename per line. This avoids the truncation of
```

```
;    0115  0     !                long filenames under VMS V4.
;    0116  0     !
;    0117  0     !        18      RE MUSE          22 MAY,1984     Version 7.0
;    0118  0     !                increased buffer size to handle long filenames for VMS v4
;    0119  0     !
;    0120  0     !        19      M. Baggett       2-Nov-1984      version 7.1
;    0121  0     !                Added support for TK50. Suppress re-inits for DIRECTORY
;    0122  0     !
;    0123  0     !        20      M. Baggett       26-NOV-1984     Version 8.0
;    0124  0     !                Added support for RUX50 devices.
;    0125  0     !
;    0126  0     !        21      J. Shelhamer     18 Dec, 1984    Version 8.0
;    0127  0     !                Fixed bug with Directory for RT-11 devices. A pattern match was
;    0128  0     !                looking for a version number (wild-carded).
;    0129  0     !
;    0130  0     !        22      M. Baggett       1-Jun-1985      Version 8.3
;    0131  0     !                Added support for ultrix disk files.
;    0132  0     !
;    0133  0     !        23      Bob Bergazzi     20-Aug-1985     Version 8.3
;    0134  0     !                Added support for AAA console storage device in ODS format.
;    0135  0     !                to DSX$DIRECTORY routine.
;    0136  0     !
;    0137  0     !        24      M. Baggett       27-Aug-1985     Version 9.0
;    0138  0     !                Added support for RD52, RD53.
;    0139  0     !                                                                                 !G:2
;    0140  0     !        25      Ted Salem        7-April-1986    Version 10.0                    !G:2
;    0141  0     !                Added support to the Console on ANDROMEDA.                       !G:2
;    0142  0     !                                                                                 !G:11
;    0143  0     !        26      Ingrid Martin / Steve Meier                                      !G:6
;    0144  0     !                                 11-August-1986   Version 10.3                   !G:7
;    0145  0     !                Revised the main code to allow directories of logical names,     !G:6
;    0146  0     !                and removed the requirement to attach a device when doing a directory. (USER MODE ONLY) !G:6
;    0147  0     !                [Lines 1000 - 1100 have been rearranged/changed accordingly]     !G:6
;    0148  0     !                                                                                 !G:9
;    0149  0     !        27      Ingrid Martin   25-October-1986 Version 10.3                     !G:10
;    0150  0     !                Enhanced the directory command such that logical names and       !G:9
;    0151  0     !                searchlists are allowed online.  VMS does the actual directory   !G:9
;    0152  0     !                for ODS-1, ODS-2 and ANSI formats via new routine VMS_DIR.       !G:9
;    0153  0     !                                                                                 !G:10
;    0154  0     !        28      Ingrid Martin   30-Mar-1987     Version 10.7                     !G:10
;    0155  0     !                Updates for RRR.                                                 !G:10
;    0156  0     !                                                                                 !G:10
;    0157  0     !        29      Ingrid Martin   14-Apr-1987     Version 10.7                     !G:10
;    0158  0     !                More updates                                                     !G:10
;    0159  0     !                                                                                 !G:11
;    0160  0     !                Jason Mi        28-Oct-1987     Version 10.10                    !G:11
;    0161  0     !                Fixed the bug causing access violation while doing a "DIRECTORY". !G:11
;    0162  0     !                                                                                 !G:12
;    0163  0     !                M. Baggett      28-Oct-1987     Version 10.10-12                 !G:12
;    0164  0     !                Added flags to directory command so tape driver can avoid unnecessary drive clear commands. !G:12
;    0165  0     !--
;    0166  0
;    0167  1     begin
;    0168  1     !
;    0169  1     ! Table of contents:
;    0170  1     !
;    0171  1     !
```

```
;    0172  1      ! Include files:
;    0173  1      !
;    0174  1
;    0175  1      library '$diag';                !
;    0176  1
;    0177  1      library '$ds';                  !
;    0178  1
;    0179  1      library 'sys$library:lib';      !
;    0180  1
;    0181  1      !
;    0182  1      ! Macros:
;    0183  1      !
;    0184  1
;    0185  1
;    0186  1      !
;    0187  1      ! psect definitions:
;    0188  1      !
;    0189  1
;    0190  1      psect
;    0191  1          plit  = data ( noexecute, share, nowrite, addressing_mode (long_relative));                !
;    0192  1
;    0193  1      psect
;    0194  1          own   = work ( noexecute, noshare,  write, addressing_mode (long_relative));               !
;    0195  1
;    0196  1      psect
;    0197  1          global = work ( noexecute, noshare,  write, addressing_mode (long_relative));              !
;    0198  1
;    0199  1      psect
;    0200  1          code   = code ( execute, share, nowrite, addressing_mode(long_relative));                  !
;    0201  1
;    0202  1      !
;    0203  1      ! equated symbols:
;    0204  1      !
;    0205  1      $ds_dsadef;                                     ! Define DSA locations
;    0206  1
;    0207  1      MAP
;    0208  1          DSA$GL_SID : BLOCK;                                                                        !
;    0209  1
;    0210  1      bind
;    0211  1          f_line = $ascic ('!AD!/');
;    0212  1
;    0213  1      linkage
;    0214  1          jsb_fake = jsb : nopreserve (2),
;    0215  1          jsb_scan = jsb (register = 3, register = 4, register = 5) : nopreserve (2);
;    0216  1
;    0217  1      !
;    0218  1      ! Own storage:
;    0219  1      !
;    0220  1          Own                                         !                                              [08]
;    0221  1              FileSpecMatch : Block [8, Byte],        !                                              [10]
;    0222  1              ControlCFlag : Byte;                    !                                              [08]
;    0223  1
;    0224  1          GLOBAL                                      !                                              [17]
;    0225  1              WIDE_FLAG : BYTE,                       ! A 1 value indicates /WIDE was typed, 0 value - no /WIDE      [17]
;    0226  1              end_size : BYTE;                        ! Size of ultrix directory string after OPEN   [22]
;    0227  1
;    0228  1      !
```

```
;   0229  1     ! Literals                                          !                                                   [16]
;   0230  1     !
;   0231  1
;   0232  1     LITERAL
;   0233  1             ultrix_v_mode = 0,                          !                                                   [22]    !G:10
;   0234  1             pr$_sid_typ8RR = 17;                        ! RRR (this can be removed when the macro shows up) [28] !G:10
;   0235  1
;   0236  1     !
;   0237  1     ! External references:
;   0238  1     !
;   0239  1
;   0240  1     external literal
;   0241  1         def$c_ultrix_len,                               ! Length of file spec backup            [22]
;   0242  1         def$c_bkstp_len,                                ! Length of file spec backup                                !G:3
;   0243  1         pr$_sys_typ900;                                 ! System type for Andromeda [25]                            !G:3
;   0244  1
;   0245  1     external
;   0246  1         Ultrix_flags : Byte addressing_mode (long_relative),        ! Flag for ULTRIX disk structure [22]
;   0247  1         DS$gb_systype: Byte addressing_mode (long_relative),        ! Byte for System Type                      [25] !G:4
;   0248  1         begin_bliss : addressing_mode (long_relative),      ! Address of BEGIN code
;   0249  1         ds$gl_clibase : addressing_mode (long_relative),
;   0250  1         boo$select,                                     ! Non-interrupt "QIO" package stuff
;   0251  1         boo$al_vector : addressing_mode (long_relative),
;   0252  1         def$q_backstop : addressing_mode (long_relative),   ! "last-chance" file spec backup string
;   0253  1         def$q_ultrix : addressing_mode (long_relative),     ! "last-chance" file spec backup string [22]
;   0254  1         def$q_dev : bblock addressing_mode (long_relative), ! Descriptor of system default device
;   0255  1         def$q_dir : bblock addressing_mode (long_relative); ! Descriptor of system default directory
;   0256  1
;   0257  1     external routine
;   0258  1         boo$qio : addressing_mode (long_relative),
;   0259  1         scan$numeric : jsb_scan addressing_mode (long_relative),
;   0260  1         kb_check : jsb_none addressing_mode (long_relative),
;   0261  1         script$flush : jsb_none addressing_mode (long_relative),
;   0262  1         init_context : jsb_none addressing_mode (long_relative),
;   0263  1         ds_cleanup : jsb_none addressing_mode (long_relative),
;   0264  1         qio$cleanup,
;   0265  1         breakup_file,
;   0266  1         Dsr$Key_Equal,                          ! compare two keywords                                                  [10]
;   0267  1         loc$ptable : jsb_loc$ptable addressing_mode (long_relative),        !                                          [05]
;   0268  1         Dsr$Allocate_Pseudo_RPB : Addressing_Mode (Long_Relative),  !                                                  [09]
;   0269  1         Dsr$DeAllocate_Pseudo_RPB : Addressing_Mode (Long_Relative),        !                                          [09]
;   0270  1         dsr$completion : jsb_0,                                                                                 !G:9
;   0271  1         sys$search : addressing_mode (general),                                                                 !G:9
;   0272  1         lib$signal : addressing_mode (general),                                                                 !G:9
;   0273  1         sys$parse : addressing_mode (general),                                                                  !G:9
;   0274  1         sys$getsyi : addressing_mode (general);                                                                 !G:9
;   0275  1                                                                                                                 !G:9
```

```
;   0277  1     %sbttl 'print a directory heading'              ! begin [27]                                    !G:9
;   0278  1     routine header (nam_blk, device_d) : novalue =                                                  !G:9
;   0279  1                                                                                                     !G:9
;   0280  1     !+                                                                                              !G:9
;   0281  1     ! function:                                                                                     !G:11
;   0282  1     !       print a directory heading,  device:[directory]                                          !G:9
;   0283  1     !                                                                                               !G:9
;   0284  1     ! input:                                                                                        !G:11
;   0285  1     !       descriptor of a full file specification, device:[directory]filename.typ;ver             !G:11
;   0286  1     !                                                                                               !G:9
;   0287  1     ! output:                                                                                       !G:11
;   0288  1     !       descriptor of device and directory parts, device:[dir]                                  !G:9
;   0289  1     !-                                                                                              !G:9
;   0290  1                                                                                                     !G:9
;   0291  1                                                                                                     !G:9
;   0292  2     begin                                                                                           !G:9
;   0293  2     bind                                                                                            !G:9
;   0294  2        nam = .nam_blk,                                                                              !G:9
;   0295  2        dev_d = .device_d;                                                                           !G:9
;   0296  2     map                                                                                             !G:9
;   0297  2        nam : block [ ,byte],                                                                        !G:9
;   0298  2        dev_d : bblock [dsc$c_s_bln];                                                                !G:9
;   0299  2             ch$move (.nam [nam$b_dev], .nam [nam$l_dev], .dev_d [dsc$a_pointer]);                   !G:9
;   0300  2             ch$move (.nam [nam$b_dir], .nam [nam$l_dir], .dev_d [dsc$a_pointer] + .nam [nam$b_dev]); !G:9
;   0301  2             dev_d [dsc$w_length] = (.nam [nam$b_dev] + .nam [nam$b_dir]);                           !G:9
;   0302  2             $ds_printf ($ascic ('!/!/Directory !AS!/!/'),dev_d);                  ! print heading !G:9
;   0303  1     end;                                                                          ! end [27]      !G:9


                                                                  .TITLE  DIRECTORY *** DIRECTORY Handle DIRECTORY comman
;                                                                                 d
                                                                  .IDENT  \10.10-12\

                                                                  .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                  2F  21  44  41  21  05  00000 P.AAA:  .ASCII  <5>\!AD!/\                                ;
20  79  72  6F  74  63  65  72  69  44  2F  21  2F  21  15  00006 P.AAB:  .ASCII  <21>\!/!/Directory !AS!/!/\                ;
                                2F  21  2F  21  53  41  21  00015                                                        ;

                                                                  .PSECT  WORK,NOEXE,2

                                                        00000 FILESPECMATCH:
                                                                  .BLKB   8
                                                        00008 CONTROLCFLAG:
                                                                  .BLKB   1
                                                        00009 WIDE_FLAG::
                                                                  .BLKB   1
                                                        0000A END_SIZE::
                                                                  .BLKB   1

                                                              DSA$AL_APTMAIL=     65024
                                                              DSA$GL_FLAGS=       65024
                                                              DSA$GL_APTCOM=      65028
                                                              DSA$GL_PASSES=      65032
                                                              DSA$GL_UNITS=       65036
                                                              DSA$GL_SECTNO=      65040
                                                              DSA$GL_SID=         65044
```

```
                                                  DSA$GL_MSGTYP=      65088
                                                  DSA$GL_ERRNO=       65092
                                                  DSA$GL_EVENT=       65096
                                                  DSA$GL_SUBTNO=      65100
                                                  DSA$GL_TESTNO=      65104
                                                  DSA$GL_PASSNO=      65108
                                                  DSA$GL_DEVLEN=      65112
                                                  DSA$GL_DEVNAM=      65116
                                                  DSA$GL_MSGPTR=      65128
                                                  F_LINE=             P.AAA
                                                          .EXTRN  DEF$C_ULTRIX_LEN
                                                          .EXTRN  DEF$C_BKSTP_LEN
                                                          .EXTRN  PR$_SYS_TYP900, ULTRIX_FLAGS
                                                          .EXTRN  DS$GB_SYSTYPE, BEGIN_BLISS
                                                          .EXTRN  DS$GL_CLIBASE, BOO$SELECT
                                                          .EXTRN  BOO$AL_VECTOR, DEF$Q_BACKSTOP
                                                          .EXTRN  DEF$Q_ULTRIX, DEF$Q_DEV
                                                          .EXTRN  DEF$Q_DIR, BOO$QIO
                                                          .EXTRN  SCAN$NUMERIC, KB_CHECK
                                                          .EXTRN  SCRIPT$FLUSH, INIT_CONTEXT
                                                          .EXTRN  DS_CLEANUP, QIO$CLEANUP
                                                          .EXTRN  BREAKUP_FILE, DSR$KEY_EQUAL
                                                          .EXTRN  LOC$PTABLE, DSR$ALLOCATE_PSEUDO_RPB
                                                          .EXTRN  DSR$DEALLOCATE_PSEUDO_RPB
                                                          .EXTRN  DSR$COMPLETION, SYS$SEARCH
                                                          .EXTRN  LIB$SIGNAL, SYS$PARSE
                                                          .EXTRN  SYS$GETSYI, DS$PRINTF

                                                          .PSECT  CODE,NOWRT,  SHR,2

                                          00FC 00000 HEADER: .WORD   Save R2,R3,R4,R5,R6,R7                    ; 0278
                                 56       04   AC  7D 00002        MOVQ    NAM_BLK, R6                              ; 0294
                                 50       39   A6  9A 00006        MOVZBL  57(R6), R0                               ; 0299
                 04   B7       44   B6           50  28 0000A        MOVC3   R0, @68(R6), @4(R7)                      ;
                                 51       3A   A6  9A 00010        MOVZBL  58(R6), R1                               ; 0300
                                 50       39   A6  9A 00014        MOVZBL  57(R6), R0                               ;
                                 50       04   A7  C0 00018        ADDL2   4(R7), R0                                ;
                      60       48   B6           51  28 0001C        MOVC3   R1, @72(R6), (R0)                        ;
                                 50       39   A6  9A 00021        MOVZBL  57(R6), R0                               ; 0301
                                 51       3A   A6  9A 00025        MOVZBL  58(R6), R1                               ;
                      67          50           51  A1 00029        ADDW3   R1, R0, (R7)                             ;
                                                57  DD 0002D        PUSHL   R7                                       ; 0302
                                     00000000' EF  9F 0002F        PUSHAB  P.AAB                                    ;
                         00000000G 9F           02  FB 00035        CALLS   #2, @#DS$PRINTF                          ;
                                                    04 0003C        RET                                              ; 0303

; Routine Size:  61 bytes,    Routine Base:  CODE + 0000


;   0304  1                                                                                                  !G:9
```

```
;    0306  1     %sbttl 'put 1 line of the directory on the screen'                  ! begin [27]                              !G:9
;    0307  1     routine put_line (line_d, ptr) : novalue =                                                                    !G:11
;    0308  1                                                                                                                   !G:9
;    0309  2     begin                                                                                                         !G:9
;    0310  2     map                                                                                                           !G:9
;    0311  2        line_d : ref bblock;                                                                                       !G:9
;    0312  2             $ds_printf ($ascic ('!AS!/'),.line_d);                                                                !G:9
;    0313  2             ch$fill (%c' ',80, .line_d [dsc$a_pointer]);                                                          !G:9
;    0314  2             (.ptr) <0,8,0> = 0;                                        ! ptr to end of line is reset              !G:9
;    0315  1     end;                                                               ! end [27]                                 !G:9


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                2F  21  53  41  21  05  0001C P.AAC:  .ASCII  <5>\!AS!/\                                     ;


                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                    003C 00000 PUT_LINE:
                                                                .WORD   Save R2,R3,R4,R5                                    ; 0307
                                    52      04      AC  D0 00002        MOVL    LINE_D, R2                                  ; 0312
                                                    52  DD 00006        PUSHL   R2                                          ;
                                        00000000'   EF  9F 00008        PUSHAB  P.AAC                                       ;
                                00000000G   9F      02  FB 0000E        CALLS   #2, a#DS$PRINTF                             ;
    0050    8F          20              6E      00  2C 00015        MOVC5   #0, (SP), #32, #80, a4(R2)                      ; 0313
                                                    04  B2   0001C                                                          ;
                                                    08  BC  94 0001E    CLRB    aPTR                                        ; 0314
                                                        04 00021        RET                                                 ; 0315

; Routine Size:  34 bytes,    Routine Base:  CODE + 003D

;    0316  1                                                                                                                   !G:9
```

```
;   0318  1     %sbttl 'directory for online ODS-1, ODS-2, ANSI directories'    ! begin [27]                    !G:9
;   0319  1     routine VMS_dir =                                                                              !G:9
;   0320  1                                                                                                    !G:9
;   0321  1     !+                                                                                             !G:9
;   0322  1     ! Function                                                                                     !G:9
;   0323  1     !       sends the full file specification of a directory to VMS                                !G:11
;   0324  1     !       (via $parse and $search) and then prints the directory                                 !G:9
;   0325  1     !       (columns or with the /WIDE switch)                                                     !G:9
;   0326  1     !                                                                                              !G:11
;   0327  1     ! Input                                                                                        !G:9
;   0328  1     !       none                                                                                   !G:9
;   0329  1     ! Output                                                                                       !G:9
;   0330  1     !       none                                                                                   !G:9
;   0331  1     !                                                                                              !G:9
;   0332  1     ! Condition Code sent to dsr$completion if the action was not successful                       !G:9
;   0333  1     !-                                                                                             !G:9
;   0334  1                                                                                                    !G:9
;   0335  2     begin                                                                                          !G:11
;   0336  2         local                                                                                      !G:9
;   0337  2             user_spec : bblock [dsc$c_s_bln] initial (0),                                          !G:9
;   0338  2             file_buf : block [48, byte],            ! 39.3;3                                       !G:9
;   0339  2             filename_d : bblock [dsc$c_s_bln] initial (                                            !G:9
;   0340  2                         long (0),                                                                  !G:9
;   0341  2                         long (file_buf)),                                                          !G:9
;   0342  2             line_buf : block [80,byte] initial (%ascii ' '),                                       !G:9
;   0343  2             line_d : bblock [dsc$c_s_bln] initial (                                                !G:9
;   0344  2                         long (80),                                                                 !G:9
;   0345  2                         long (line_buf)),                                                          !G:9
;   0346  2             dev_buf : block [lnm$c_namlength, byte],                                               !G:9
;   0347  2             device_d : bblock [dsc$c_s_bln] initial (                                              !G:9
;   0348  2                         long (0),                                                                  !G:9
;   0349  2                         long (dev_buf)),                                                           !G:9
;   0350  2             files : long initial (0),                                                              !G:9
;   0351  2             total_files : long initial (0),                                                        !G:9
;   0352  2             total_dirs : long initial (0),                                                         !G:9
;   0353  2             status : long initial (0),                                                             !G:9
;   0354  2             stat2 : long initial (0),                                                              !G:9
;   0355  2             def_spec : long initial (%ascid '*.*;*'),                                              !G:9
;   0356  2             ptr : byte initial (0),                                                                !G:9
;   0357  2             same_dir : byte initial (0);                                                           !G:9
;   0358  2                                                                                                    !G:9
;   0359  2         own                                                                                        !G:9
;   0360  2             res_str : bblock [nam$c_maxrss],                ! output from $search                  !G:9
;   0361  2             spec : bblock [nam$c_maxrss],                                                          !G:9
;   0362  2             exp_str : bblock [nam$c_maxrss],                ! expanded string buffer               !G:9
;   0363  2             nam_blk : $nam_decl,                                                                   !G:9
;   0364  2             fab_blk : $fab_decl;                                                                   !G:9
;   0365  2                                                                                                    !G:9
;   0366  2         bind                                                                                       !G:9
;   0367  2             default = .def_spec;                                                                   !G:9
;   0368  2         map                                                                                        !G:9
;   0369  2             default : bblock [dsc$c_s_bln];                                                        !G:9
;   0370  2                                                                                                    !G:9
;   0371  2             ! move user-supplied filespec from ds$gl_clibase to user_spec                          !G:9
;   0372  2             ch$move (.(ds$gl_clibase + 8), .(ds$gl_clibase+12), spec);                             !G:11
;   0373  2             user_spec [dsc$w_length] = .(ds$gl_clibase + 8);                                       !G:11
;   0374  2             user_spec [dsc$a_pointer] = spec;                                                      !G:9
```

```
;    0375  2              !                                                                                          !G:9
;    0376  2              ! initialize the nam and fab blocks                                                        !G:11
; P  0377  2              $nam_init                                                                                  !G:11
; P  0378  2                      (nam = nam_blk,                                                                    !G:9
; P  0379  2                       rsa = res_str,                                                                    !G:9
; P  0380  2                       rss = nam$c_maxrss,                                                               !G:9
; P  0381  2                       esa = exp_str,                                                                    !G:9
;    0382  2                       ess = nam$c_maxrss);                                                              !G:9
; P  0383  2              $fab_init                                                                                  !G:11
; P  0384  2                      (fop = nam,                                                                        !G:9
; P  0385  2                       fab = fab_blk,                                                                    !G:11
; P  0386  2                       nam = nam_blk,                                                                    !G:9
; P  0387  2                       fna = spec,                                                                       !G:11
; P  0388  2                       fns = .user_spec [dsc$w_length],                                                  !G:9
; P  0389  2                       dna = .default [dsc$a_pointer],                                                   !G:9
;    0390  2                       dns = .default [dsc$w_length]);                                                   !G:9
;    0391  2              !                                                                                          !G:9
;    0392  2              ! analize file spec to prepare for wildcard processing to be used in Search service.  If wildcards or !G:11
;    0393  2              ! logical names (including a searchlist) are present in the file spec, RMS allocates internal data !G:11
;    0394  2              ! structures to store the context for subsequent searches.                                 !G:9
;    0395  2              !                                                                                          !G:9
;    0396  2              status = sys$parse (fab_blk);                                                              !G:9
;    0397  2              !                                                                                          !G:9
;    0398  2              if .status eql rms$_suc                                                                    !G:11
;    0399  2              then                                                                                       !G:9
;    0400  3                begin                                                                                    !G:9
;    0401  3                    ! scan a directory file for an entry that matches the file spec specified by the expanded !G:11
;    0402  3                    ! string.  Upon finding a match, RMS returns the file spec in the resultant string.  !G:9
;    0403  3                    stat2 = (sys$search (fab_blk));                    ! this call to $search used initially !G:9
;    0404  3                    !                                                                                    !G:9
;    0405  3                    ch$fill (%c' ',80, .line_d [dsc$a_pointer]);                                         !G:9
;    0406  3                    while .stat2 eql rms$_suc do                                                         !G:9
;    0407  4                     begin                                                                               !G:9
;    0408  4                       if .files eql 0                                  ! beginning of algorithm?         !G:9
;    0409  4                       then header (nam_blk,device_d);                  ! print "Directory blah:[blah]"   !G:9
;    0410  4                       !                                                                                 !G:9
;    0411  4                       ! if resultant file spec from $search does not have the same device:[dir] as the previous !G:11
;    0412  4                       ! previous file spec then print buffer if not empty and print new header, ie Directory blah:[blah] !
;    0413  4                       same_dir = ch$eql (.device_d [dsc$w_length], .device_d [dsc$a_pointer],            !G:9
;    0414  4                                          (.nam_blk[nam$b_dev] + .nam_blk[nam$b_dir]), .nam_blk[nam$l_dev]); !G:9
;    0415  4                       if .same_dir eql 0                               ! new directory, ie searchlist    !G:9
;    0416  5                       then begin                                                                        !G:11
;    0417  5                            if .ptr neq 0 then put_line (line_d, ptr);                                   !G:9
;    0418  5                            if .files neq 1                                                              !G:11
;    0419  6                            then $ds_printf ($ascic ('!/Total of !ZL Files.!/'), .files)                 !G:9
;    0420  5                            else $ds_printf ($ascic ('!/Total of !ZL File.!/'), .files);                 !G:9
;    0421  5                            total_files = .total_files + .files;     ! searchlists, ie multiple directories !G:9
;    0422  5                            files = 0;                                                                   !G:11
;    0423  5                            total_dirs = .total_dirs + 1;            ! searchlists                     !G:9
;    0424  5                            header (nam_blk,device_d);               ! print new heading               !G:9
;    0425  4                        end;                                                                            !G:9
;    0426  4                       !                                                                                !G:9
;    0427  4                       ! move the filename.typ;ver from the resultant string in the name block to filename_d !G:9
;    0428  4                       filename_d [dsc$w_length] = .nam_blk[nam$b_name] + .nam_blk[nam$b_type] + .nam_blk[nam$b_ver]; !G:11
;    0429  4                       ch$move (.filename_d [dsc$w_length],.nam_blk[nam$l_name],.filename_d [dsc$a_pointer]); !G:11
;    0430  4                       !                                                                                !G:9
;    0431  4                       files = .files + 1;                                                              !G:9
```

```
;  0432  4                    if .wide_flag                                                  ! /WIDE switch          !G:9
;  0433  5                    then begin                                                     ! print 1 file per line !G:9
;  0434  5                            ch$move (.filename_d [dsc$w_length], .filename_d [dsc$a_pointer], .line_d [dsc$a_pointer]);
;  0435  5                            put_line (line_d, ptr);                                                        !G:9
;  0436  5                        end                                                                                !G:9
;  0437  4                    else                                                                                   !G:11
;  0438  5                        BEGIN                                                      ! put x (x<=4) filenames in line_d !
;  0439  5                            if (80 - .ptr) gtr .filename_d [dsc$w_length]          ! will the filename fit on the line?
;  0440  6                            then begin                                             ! yes                   !G:9
;  0441  6                                !                                                                          !G:9
;  0442  6                                ! move the filename into line_buf;                                         !G:11
;  0443  6                                ! ptr = ptr + length of filename + length of column, 20 - (length of filename mod 20) !G
;  0444  6                                ! if line_buf is full, then print it                                       !G:9
;  0445  6                                ch$move (.filename_d [dsc$w_length], .filename_d [dsc$a_pointer],          !G:11
;  0446  6                                        .line_d [dsc$a_pointer] + .ptr);                                   !G:9
;  0447  6                                ptr = .ptr + .filename_d [dsc$w_length] + 20 - (.filename_d [dsc$w_length] mod 20); !G:9
;  0448  6                                if .ptr geq 80                                                             !G:11
;  0449  6                                then put_line (line_d, ptr);                                               !G:9
;  0450  6                              end                                                                          !G:9
;  0451  5                            else                                                                           !G:11
;  0452  5                              !                                                                            !G:9
;  0453  5                              ! if the new filename does not fit in the line_buf then print the line_buf, and then !G:9
;  0454  5                              ! move the new filename into line_buf and update the pointer                 !G:9
;  0455  6                              begin                                                                        !G:11
;  0456  6                                put_line (line_d, ptr);                                                    !G:9
;  0457  6                                ch$move (.filename_d [dsc$w_length], .filename_d [dsc$a_pointer],          !G:11
;  0458  6                                        .line_d [dsc$a_pointer]);                                          !G:9
;  0459  6                                ptr = .filename_d [dsc$w_length] + 20 - (.filename_d [dsc$w_length] mod 20); !G:9
;  0460  5                              end;                                                                         !G:9
;  0461  4                        END;                                                 ! end if not wide              !G:9
;  0462  4                    stat2 = (sys$search (fab_blk));                          ! this call to $search is used to loop on !G
;  0463  3                  end;                                                       ! end while sys$search         !G:9
;  0464  3                  if .stat2 eql rms$_nmf                                     ! no more files found          !G:9
;  0465  4                  then  begin                                                ! to get RMS$_NMF, >= 1 file match occurred
;  0466  4                            if .ptr neq 0                                                                  !G:11
;  0467  4                            then put_line (line_d, ptr);                     ! is there still stuff in the line buffer !G
;  0468  4                            if .files neq 1                                                                !G:11
;  0469  5                            then $ds_printf ($ascic ('!/Total of !ZL Files.!/'), .files)                   !G:9
;  0470  4                            else $ds_printf ($ascic ('!/Total of !ZL File.!/'), .files);                   !G:9
;  0471  4                            if .total_dirs neq 0                             ! is it a searchlist?          !G:9
;  0472  5                            then begin                                                                     !G:9
;  0473  5                                  total_files = .total_files + .files;                                     !G:9
;  0474  5                                  total_dirs = .total_dirs + 1;                                            !G:9
;  0475  5                                  $ds_printf ($ascic ('!/Grand total of !ZL directories, !ZL files.!/'), .total_dirs, .
;  0476  4                                end;                                                                       !G:11
;  0477  4                            stat2 = rms$_suc;                                                              !G:9
;  0478  4                            return .stat2;                                                                 !G:9
;  0479  4                        end                                                                                !G:9
;  0480  3                  else return .stat2                                                                       !G:9
;  0481  3              end                                                                                          !G:9
;  0482  2                else return .status;                                           ! is sys$parse status isn't normal !G:9
;  0483  1      end;    ! end begin                                                      ! end [27]                     !G:9
```

.PSECT DATA,NOWRT,NOEXE, SHR,2

```
                                                                  00022          .BLKB   2
                                                  00  00  00  20  00024 P.AAD:   .ASCII  \ \<0><0><0>                                  ;
                                  00  00  00  2A  3B  2A  2E  2A  00028 P.AAF:   .ASCII  \*.*;*\<0><0><0>                              ;
                                                      010E0005    00030 P.AAE:   .LONG   17694725                                      ;
                                                      00000000'   00034          .ADDRESS P.AAF                                        ;
4C  5A  21  20  66  6F  20  6C  61  74  6F  54  2F  21  17  00038 P.AAG:   .ASCII  <23>\!/Total of !ZL Files.!/\                   ;
                        2F  21  2E  73  65  6C  69  46  20  00047                                                                ;
4C  5A  21  20  66  6F  20  6C  61  74  6F  54  2F  21  16  00050 P.AAH:   .ASCII  <22>\!/Total of !ZL File.!/\                    ;
                            2F  21  2E  65  6C  69  46  20  0005F                                                                ;
4C  5A  21  20  66  6F  20  6C  61  74  6F  54  2F  21  17  00067 P.AAI:   .ASCII  <23>\!/Total of !ZL Files.!/\                   ;
                        2F  21  2E  73  65  6C  69  46  20  00076                                                                ;
4C  5A  21  20  66  6F  20  6C  61  74  6F  54  2F  21  16  0007F P.AAJ:   .ASCII  <22>\!/Total of !ZL File.!/\                    ;
                            2F  21  2E  65  6C  69  46  20  0008E                                                                ;
20  6C  61  74  6F  74  20  64  6E  61  72  47  2F  21  2E  00096 P.AAK:   .ASCII  \.!/Grand total of !ZL directories, !ZL f\      ;
72  6F  74  63  65  72  69  64  20  4C  5A  21  20  66  6F  000A5                                                                ;
                    66  20  4C  5A  21  20  2C  73  65  69  000B4                                                                ;
                                2F  21  2E  73  65  6C  69  000BE          .ASCII  \iles.!/\                                     ;

                                                                           .PSECT  WORK,NOEXE,2

                                                                  0000B          .BLKB   1
                                                                  0000C RES_STR:.BLKB   255
                                                                  0010B          .BLKB   1
                                                                  0010C SPEC:    .BLKB   255
                                                                  0020B          .BLKB   1
                                                                  0020C EXP_STR:.BLKB   255
                                                                  0030B          .BLKB   1
                                                                  0030C NAM_BLK:.BLKB   96
                                                                  0036C FAB_BLK:.BLKB   80

                                                                        $RMS_PTR=          NAM_BLK
                                                                        $RMS_PTR=          FAB_BLK


                                                                           .PSECT  CODE,NOWRT,  SHR,2

                                                      0FFC 00000 VMS_DIR:.WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0319
                                      5E      FE54  CE  9E 00002          MOVAB   -428(SP), SP                                  ;
                                              F8  AD  7C 00007          CLRQ    USER_SPEC                                     ; 0335
                                              C0  AD  D4 0000A          CLRL    FILENAME_D                                    ;
                                  C4  AD      C8  AD  9E 0000D          MOVAB   FILE_BUF, FILENAME_D+4                        ;
0050  8F                00 00000000'  EF          04  2C 00012          MOVC5   #4, P.AAD, #0, #80, LINE_BUF                  ; 0342
                                          FF70  CD     0001D                                                                    ;
                              FF68  CD      50  8F  9A 00020          MOVZBL  #80, LINE_D                                   ;
                              FF6C  CD    FF70  CD  9E 00026          MOVAB   LINE_BUF, LINE_D+4                            ;
                                              0C  AE  D4 0002D          CLRL    DEVICE_D                                      ;
                                  10  AE      14  AE  9E 00030          MOVAB   DEV_BUF, DEVICE_D+4                           ;
                                              04  AE  D4 00035          CLRL    TOTAL_FILES                                   ;
                                                  5B  D4 00038          CLRL    TOTAL_DIRS                                    ;
                                                  57  D4 0003A          CLRL    STATUS                                        ;
                                                  59  7C 0003C          CLRQ    FILES                                         ;
                                      56 00000000'  EF  9E 0003E          MOVAB   P.AAE, DEF_SPEC                               ;
                                              08  AE  94 00045          CLRB    PTR                                           ;
                                                  6E  94 00048          CLRB    SAME_DIR                                      ;
          00000000'  EF 00000000G  FF 00000000G  EF  28 0004A          MOVC3   DS$GL_CLIBASE+8, @DS$GL_CLIBASE+12, SPEC      ; 0372
                                  F8  AD 00000000G  EF  B0 0005A          MOVW    DS$GL_CLIBASE+8, USER_SPEC                    ; 0373
                                  FC  AD 00000000'  EF  9E 00062          MOVAB   SPEC, USER_SPEC+4                             ; 0374
```

```
0060   8F          00          6E          00  2C 0006A        MOVC5   #0, (SP), #0, #96, $RMS_PTR                ; 0382
                                 00000000'  EF     00071                                                          ;
                     00000000'  EF     6002  8F  B0 00076        MOVW    #24578, $RMS_PTR                          ;
                     00000000'  EF           01  8E 0007F        MNEGB   #1, $RMS_PTR+2                            ;
                     00000000'  EF 00000000'  EF  9E 00086        MOVAB   RES_STR, $RMS_PTR+4                       ;
                     00000000'  EF           01  8E 00091        MNEGB   #1, $RMS_PTR+10                           ;
                     00000000'  EF 00000000'  EF  9E 00098        MOVAB   EXP_STR, $RMS_PTR+12                      ;
0050   8F          00          6E          00  2C 000A3        MOVC5   #0, (SP), #0, #80, $RMS_PTR                ; 0390
                                 00000000'  EF     000AA                                                          ;
                     00000000'  EF     5003  8F  B0 000AF        MOVW    #20483, $RMS_PTR                          ;
                     00000000'  EF 01000000  8F  D0 000B8        MOVL    #16777216, $RMS_PTR+4                     ;
                     00000000'  EF           02  90 000C3        MOVB    #2, $RMS_PTR+22                           ;
                     00000000'  EF           02  90 000CA        MOVB    #2, $RMS_PTR+31                           ;
                     00000000'  EF 00000000'  EF  9E 000D1        MOVAB   NAM_BLK, $RMS_PTR+40                      ;
                     00000000'  EF 00000000'  EF  9E 000DC        MOVAB   SPEC, $RMS_PTR+44                         ;
                     00000000'  EF       04  A6  D0 000E7        MOVL    4(DEF_SPEC), $RMS_PTR+48                  ;
                     00000000'  EF       F8  AD  90 000EF        MOVB    USER_SPEC, $RMS_PTR+52                    ;
                     00000000'  EF           66  90 000F7        MOVB    (DEF_SPEC), $RMS_PTR+53                   ;
                                 00000000'  EF  9F 000FE        PUSHAB  FAB_BLK                                   ; 0396
                     00000000G  00           01  FB 00104        CALLS   #1, SYS$PARSE                             ;
                                57           50  D0 0010B        MOVL    R0, STATUS                                ;
                     00010001   8F           57  D1 0010E        CMPL    STATUS, #65537                            ; 0398
                                             03  13 00115        BEQL    1$                                        ;
                                           01B1  31 00117        BRW     20$                                       ;
                                 00000000'  EF  9F 0011A 1$:    PUSHAB  FAB_BLK                                   ; 0403
                     00000000G  00           01  FB 00120        CALLS   #1, SYS$SEARCH                            ;
                                5A           50  D0 00127        MOVL    R0, STAT2                                 ;
0050   8F          20          6E          00  2C 0012A        MOVC5   #0, (SP), #32, #80, @LINE_D+4              ; 0405
                                     FF6C  DD     00131                                                          ;
                     00010001   8F           5A  D1 00134 2$:    CMPL    STAT2, #65537                             ; 0406
                                             03  13 0013B        BEQL    3$                                        ;
                                           012C  31 0013D        BRW     14$                                       ;
                                             59  D5 00140 3$:    TSTL    FILES                                     ; 0408
                                             0E  12 00142        BNEQ    4$                                        ;
                                       0C  AE  9F 00144        PUSHAB  DEVICE_D                                  ; 0409
                                 00000000'  EF  9F 00147        PUSHAB  NAM_BLK                                   ;
                         FE4F   CF           02  FB 0014D        CALLS   #2, HEADER                                ;
                                50 00000000'  EF  9A 00152 4$:    MOVZBL  NAM_BLK+57, R0                            ; 0414
                                51 00000000'  EF  9A 00159        MOVZBL  NAM_BLK+58, R1                            ;
                                50           51  C0 00160        ADDL2   R1, R0                                    ;
                                             54  D4 00163        CLRL    R4                                        ; 0413
      50          00       10   BE       0C  AE  2D 00165        CMPC5   DEVICE_D, @DEVICE_D+4, #0, R0, @NAM_BLK+68 ;
                                 00000000'  FF     0016C                                                          ;
                                             02  12 00171        BNEQ    5$                                        ;
                                             54  D6 00173        INCL    R4                                        ;
                                6E           54  90 00175 5$:    MOVB    R4, SAME_DIR                              ;
                                             45  12 00178        BNEQ    9$                                        ; 0415
                                       08  AE  95 0017A        TSTB    PTR                                       ; 0417
                                             0C  13 0017D        BEQL    6$                                        ;
                                       08  AE  9F 0017F        PUSHAB  PTR                                       ;
                                     FF68  CD  9F 00182        PUSHAB  LINE_D                                    ;
                         FE53   CF           02  FB 00186        CALLS   #2, PUT_LINE                              ;
                                01           59  D1 0018B 6$:    CMPL    FILES, #1                                 ; 0418
                                             0A  13 0018E        BEQL    7$                                        ;
                                             59  DD 00190        PUSHL   FILES                                     ; 0419
                                 00000000'  EF  9F 00192        PUSHAB  P.AAG                                     ;
                                             08  11 00198        BRB     8$                                        ;
```

```
                                                  59  DD 0019A 7$:     PUSHL   FILES                                          ; 0420
                                        00000000'  EF  9F 0019C         PUSHAB  P.AAH                                          ;
                        00000000G  9F              02  FB 001A2 8$:     CALLS   #2, @#DS$PRINTF                                ;
                                04  AE              59  C0 001A9         ADDL2   FILES, TOTAL_FILES                             ; 0421
                                                    59  D4 001AD         CLRL    FILES                                          ; 0422
                                                    5B  D6 001AF         INCL    TOTAL_DIRS                                     ; 0423
                                              0C    AE  9F 001B1         PUSHAB  DEVICE_D                                       ; 0424
                                        00000000'  EF  9F 001B4         PUSHAB  NAM_BLK                                        ;
                            FDE2  CF              02  FB 001BA         CALLS   #2, HEADER                                     ;
                                  50 00000000'    EF  9A 001BF 9$:     MOVZBL  NAM_BLK+59, R0                                 ; 0428
                                  51 00000000'    EF  9A 001C6         MOVZBL  NAM_BLK+60, R1                                 ;
                                  50              51  C0 001CD         ADDL2   R1, R0                                         ;
                                  52 00000000'    EF  9A 001D0         MOVZBL  NAM_BLK+61, R2                                 ;
                C0    AD          50              52  A1 001D7         ADDW3   R2, R0, FILENAME_D                             ;
                                  57        C0    AD  3C 001DC         MOVZWL  FILENAME_D, R7                                 ; 0429
                C4    BD 00000000'  FF              57  28 001E0         MOVC3   R7, @NAM_BLK+76, @FILENAME_D+4                 ;
                                                    59  D6 001E9         INCL    FILES                                          ; 0431
                                  09 00000000'    EF  E9 001EB         BLBC    WIDE_FLAG, 10$                                 ; 0432
              FF6C    DD      C4  BD              57  28 001F2         MOVC3   R7, @FILENAME_D+4, @LINE_D+4                   ; 0434
                                                    35  11 001F9         BRB     11$                                            ; 0435
        7E            00          57              01  7A 001FB 10$:    EMUL    #1, R7, #0, -(SP)                              ; 0447
        58            58          8E              14  7B 00200         EDIV    #20, (SP)+, R8, R8                             ;
                                  56        08    AE  9A 00205         MOVZBL  PTR, R6                                        ; 0439
                                  50        B0    A6  9E 00209         MOVAB   -80(R6), R0                                    ;
                                  50              50  CE 0020D         MNEGL   R0, R0                                         ;
                                  57              50  D1 00210         CMPL    R0, R7                                         ;
                                                    29  15 00213         BLEQ    12$                                            ;
              FF6C DD46       C4  BD              57  28 00215         MOVC3   R7, @FILENAME_D+4, @LINE_D+4[R6]               ; 0446
                    50            56              57  C1 0021D         ADDL3   R7, R6, R0                                     ; 0447
                                  50              58  C2 00221         SUBL2   R8, R0                                         ;
                08    AE          50              14  81 00224         ADDB3   #20, R0, PTR                                   ;
                                50  8F        08    AE  91 00229         CMPB    PTR, #80                                       ; 0448
                                                    29  1F 0022E         BLSSU   13$                                            ;
                                              08    AE  9F 00230 11$:    PUSHAB  PTR                                            ; 0449
                                            FF68    CD  9F 00233         PUSHAB  LINE_D                                         ;
                            FDA2  CF              02  FB 00237         CALLS   #2, PUT_LINE                                   ;
                                                    1B  11 0023C         BRB     13$                                            ;
                                              08    AE  9F 0023E 12$:    PUSHAB  PTR                                            ; 0456
                                            FF68    CD  9F 00241         PUSHAB  LINE_D                                         ;
                            FD94  CF              02  FB 00245         CALLS   #2, PUT_LINE                                   ;
              FF6C    DD      C4  BD              57  28 0024A         MOVC3   R7, @FILENAME_D+4, @LINE_D+4                   ; 0458
                                  57              58  C2 00251         SUBL2   R8, R7                                         ; 0459
                08    AE          57              14  81 00254         ADDB3   #20, R7, PTR                                   ;
                                        00000000'  EF  9F 00259 13$:    PUSHAB  FAB_BLK                                        ; 0462
                        00000000G  00              01  FB 0025F         CALLS   #1, SYS$SEARCH                                 ;
                                  5A              50  D0 00266         MOVL    R0, STAT2                                      ;
                                                FEC8  31 00269         BRW     2$                                             ;
                        000182CA  8F              5A  D1 0026C 14$:    CMPL    STAT2, #99018                                  ; 0464
                                                    52  12 00273         BNEQ    19$                                            ;
                                              08    AE  95 00275         TSTB    PTR                                            ; 0466
                                                    0C  13 00278         BEQL    15$                                            ;
                                              08    AE  9F 0027A         PUSHAB  PTR                                            ; 0467
                                            FF68    CD  9F 0027D         PUSHAB  LINE_D                                         ;
                            FD58  CF              02  FB 00281         CALLS   #2, PUT_LINE                                   ;
                                  01              59  D1 00286 15$:    CMPL    FILES, #1                                      ; 0468
                                                    0A  13 00289         BEQL    16$                                            ;
                                                    59  DD 0028B         PUSHL   FILES                                          ; 0469
```

```
                                  00000000'  EF  9F 0028D          PUSHAB  P.AAI                                  ;
                                             08  11 00293          BRB     17$                                    ;
                                             59  DD 00295 16$:     PUSHL   FILES                                  ; 0470
                                  00000000'  EF  9F 00297          PUSHAB  P.AAJ                                  ;
                     00000000G  9F           02  FB 0029D 17$:     CALLS   #2, a#DS$PRINTF                        ;
                                             5B  D5 002A4          TSTL    TOTAL_DIRS                             ; 0471
                                             18  13 002A6          BEQL    18$                                    ;
                           04   AE           59  C0 002A8          ADDL2   FILES, TOTAL_FILES                     ; 0473
                                             5B  D6 002AC          INCL    TOTAL_DIRS                             ; 0474
                                       04    AE  DD 002AE          PUSHL   TOTAL_FILES                            ; 0475
                                             5B  DD 002B1          PUSHL   TOTAL_DIRS                             ;
                                  00000000'  EF  9F 002B3          PUSHAB  P.AAK                                  ;
                     00000000G  9F           03  FB 002B9          CALLS   #3, a#DS$PRINTF                        ;
                                5A 00010001  8F  D0 002C0 18$:     MOVL    #65537, STAT2                          ; 0477
                                50           5A  D0 002C7 19$:     MOVL    STAT2, R0                              ; 0480
                                                 04 002CA          RET                                            ; 0482
                                50           57  D0 002CB 20$:     MOVL    STATUS, R0                             ;
                                                 04 002CE          RET                                            ; 0483

; Routine Size:  719 bytes,    Routine Base:  CODE + 005F


;   0484  1                                                                                                  !G:11
;   0485  1                                                                                                  !G:11
;   0486  1                                                                                                  !G:9
;   0487  1                                                                                                  !G:9
;   0488  1                                                                                                  !G:9
;   0489  1                                                                                                  !G:9
;   0490  1
```

```
;    0492  1     %Sbttl  'Put filename to buffer, if matches pattern'
;    0493  1     ;
;    0494  1     Routine PutBuffer (count, Buffer, Desc) : NoValue =                                          !G:9
;    0495  2         Begin
;    0496  2
;    0497  2     !+
;    0498  2     ! Function:
;    0499  2     !       Includes a filename within the printout buffer, if it matches the
;    0500  2     !       specified pattern, or if there is no pattern.
;    0501  2     !
;    0502  2     !       count           Address of count longword (number of count typed)                    !G:9
;    0503  2     !       Buffer          Address of directory line buffer descriptor
;    0504  2     !       Desc            Address of file name buffer descriptor (at end of
;    0505  2     !                       'Buffer' buffer.
;    0506  2     !
;    0507  2     !-
;    0508  2
;    0509  2         Local                               !                                                    [10]
;    0510  2             Wild;                           ! Dummy for key comparison routine                   [10]
;    0511  2
;    0512  2         Map                                 !                                                    [10]
;    0513  2             Buffer : Ref Block [, Byte],    !                                                    [10]
;    0514  2             Desc : Ref Block [, Byte];      !                                                    [10]
;    0515  2
;    0516  2         If .filespecMatch [Dsc$W_Length] Eql 0      ! If no pattern                              [10]
;    0517  2          Or Dsr$Key_Equal (.Desc, filespecMatch, Wild)      ! .. or pattern matches              [10]
;    0518  2         Then                                !                                                    [10]
;    0519  3             Begin                           !                                                    [10]
;    0520  3             .count = ..count + 1;           ! count this file                                    [10]
;    0521  3
;    0522  3             IF .WIDE_FLAG                                                   ! DIR/WIDE           [17]
;    0523  4             THEN BEGIN                                                      !                    [17]
;    0524  4                     Buffer [Dsc$W_Length] = .Buffer [Dsc$W_Length] + .Desc [Dsc$W_Length];      !  [17]
;    0525  4                     $Ds_PrintF ($Ascic ('!AS!/'), Buffer [Dsc$W_Length]);   ! Print out buffer  [17]
;    0526  4                     Buffer [Dsc$W_Length] = 0;                              ! Reset buffer length [17]
;    0527  4                     Ch$Fill (%C' ', 132, .Buffer [Dsc$A_Pointer])           ! Fill with blanks   [17]
;    0528  4                 END
;    0529  3             ELSE If (Buffer [Dsc$W_Length] = .Buffer [Dsc$W_Length] + .Desc [Dsc$W_Length]) Gtr 60  !  [10]
;    0530  3                     Then                    !                                                    [10]
;    0531  4                         Begin               !                                                    [10]
;    0532  4                         $Ds_PrintF ($Ascic ('!AS!/'), Buffer [Dsc$W_Length]);       ! Print out buffer  [10]
;    0533  4                         Buffer [Dsc$W_Length] = 0;  ! Reset buffer length                        [10]
;    0534  4                         Ch$Fill (%C' ', 80, .Buffer [Dsc$A_Pointer]);       ! Fill with blanks  [10]
;    0535  4                         End                 !                                                    [10]
;    0536  4
;    0537  3             End                             !                                                    [10]
;    0538  3
;    0539  1     End;                                    !                                                    [10]


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                            2F  21  53  41  21  05  000C5 P.AAL:  .ASCII  <5>\!AS!/\                         ;
                            2F  21  53  41  21  05  000CB P.AAM:  .ASCII  <5>\!AS!/\                         ;
```

```
                                                                  .PSECT  CODE,NOWRT,  SHR,2

                                                  01FC 00000 PUTBUFFER:
                                                                  .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ; 0494
                                  58 00000000G  9F  9E 00002      MOVAB   @#DS$PRINTF, R8                              ;
                                  57 00000000'  EF  9E 00009      MOVAB   FILESPECMATCH, R7                            ;
                                  5E            04  C2 00010      SUBL2   #4, SP                                       ;
                                                67  B5 00013      TSTW    FILESPECMATCH                                ; 0516
                                                0F  13 00015      BEQL    1$                                           ;
                                        4080    8F  BB 00017      PUSHR   #^M<R7,SP>                                   ; 0517
                                          0C    AC  DD 0001B      PUSHL   DESC                                         ;
                          0000G   CF            03  FB 0001E      CALLS   #3, DSR$KEY_EQUAL                            ;
                                  4E            50  E9 00023      BLBC    R0, 3$                                       ;
                                          04    BC  D6 00026 1$:  INCL    @COUNT                                       ; 0520
                                  56      08    AC  D0 00029      MOVL    BUFFER, R6                                   ; 0524
                                  1B      09    A7  E9 0002D      BLBC    WIDE_FLAG, 2$                                ; 0522
                                  66      0C    BC  A0 00031      ADDW2   @DESC, (R6)                                  ; 0524
                                                56  DD 00035      PUSHL   R6                                           ; 0525
                                     00000000'  EF  9F 00037      PUSHAB  P.AAL                                        ;
                                  68            02  FB 0003D      CALLS   #2, DS$PRINTF                                ;
                                                66  B4 00040      CLRW    (R6)                                         ; 0526
  0084   8F          20           6E            00  2C 00042      MOVC5   #0, (SP), #32, #132, @4(R6)                  ; 0527
                                          04    B6     00049                                                          ;
                                                    04 0004B      RET                                                  ;
                                  50            66  3C 0004C 2$:  MOVZWL  (R6), R0                                     ; 0529
                                  51      0C    BC  3C 0004F      MOVZWL  @DESC, R1                                    ;
                                  50            51  C0 00053      ADDL2   R1, R0                                       ;
                                  66            50  B0 00056      MOVW    R0, (R6)                                     ;
                                  3C            50  D1 00059      CMPL    R0, #60                                      ;
                                                16  15 0005C      BLEQ    3$                                           ;
                                                56  DD 0005E      PUSHL   R6                                           ; 0532
                                     00000000'  EF  9F 00060      PUSHAB  P.AAM                                        ;
                                  68            02  FB 00066      CALLS   #2, DS$PRINTF                                ;
                                                66  B4 00069      CLRW    (R6)                                         ; 0533
  0050   8F          20           6E            00  2C 0006B      MOVC5   #0, (SP), #32, #80, @4(R6)                   ; 0534
                                          04    B6     00072                                                          ;
                                                    04 00074 3$:  RET                                                  ; 0539

; Routine Size:  117 bytes,    Routine Base:  CODE + 032E
```

```
;   0541  1     %sbttl 'RAD50'
;   0542  1     ;
;   0543  1     routine rad50 (data, address) =
;   0544  2         begin
;   0545  2
;   0546  2     !++
;   0547  2     ! Functional description:
;   0548  2     !       Routine to convert RAD50 to ASCII
;   0549  2     !       writes the converted characters into a byte vector at ADDRESS
;   0550  2     !       Routine value is number of characters that were non blank
;   0551  2     !
;   0552  2     ! Formal parameters:
;   0553  2     !
;   0554  2     !       data    16 bit quantity containing 3 rad50 characters
;   0555  2     !       address address of byte vector to receive the 3 characters in ascii
;   0556  2     !
;   0557  2     ! Completion codes:     none
;   0558  2     !
;   0559  2     !--
;   0560  2
;   0561  2         map
;   0562  2             address : ref vector [3, byte];         ! Map out where to store data
;   0563  2
;   0564  2         local
;   0565  2             ptr;                                    ! Point to first space in string
;   0566  2
;   0567  2         bind
;   0568  2             xlate = uplit byte(%ascii' ABCDEFGHIJKLMNOPQRSTUVWXYZ$.?0123456789')
;   0569  2             : vector [40, byte];
;   0570  2
;   0571  2         address [0] = .xlate [.data/1600];
;   0572  2         address [1] = .xlate [(.data/40) mod 40];
;   0573  2         address [2] = .xlate [.data mod 40];
;   0574  2
;   0575  2         if ch$fail (ptr = ch$find_ch (3, ch$ptr (.address), %c' ')) then 3 else ch$diff (.ptr, ch$ptr (.address))
;   0576  2
;   0577  1         end;


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

4E  4D  4C  4B  4A  49  48  47  46  45  44  43  42  41  20  000D1 P.AAN:  .ASCII  \ ABCDEFGHIJKLMNOPQRSTUVWXYZ$.?0123456789\  ;
3F  2E  24  5A  59  58  57  56  55  54  53  52  51  50  4F  000E0                                                          ;
                    39  38  37  36  35  34  33  32  31  30  000EF                                                          ;

                                                                XLATE=              P.AAN


                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                    000C 00000 RAD50:  .WORD   Save R2,R3                                  ; 0543
                                    53 00000000'  EF  9E 00002         MOVAB   XLATE, R3                                   ;
                                    52        08  AC  D0 00009         MOVL    ADDRESS, R2                                 ; 0571
                        50      04  AC 00000640   8F  C7 0000D         DIVL3   #1600, DATA, R0                             ;
                                    62          6340  90 00016         MOVB    XLATE[R0], (R2)                             ;
                        50      04  AC            28  C7 0001A         DIVL3   #40, DATA, R0                               ; 0572
```

```
        7E          00          50          01  7A 0001F        EMUL    #1, R0, #0, -(SP)                   ;
        50          50          8E          28  7B 00024        EDIV    #40, (SP)+, R0, R0                  ;
                        01      A2        6340  90 00029        MOVB    XLATE[R0], 1(R2)                    ;
        7E          00      04  AC          01  7A 0002E        EMUL    #1, DATA, #0, -(SP)                 ; 0573
        50          50          8E          28  7B 00034        EDIV    #40, (SP)+, R0, R0                  ;
                        02      A2        6340  90 00039        MOVB    XLATE[R0], 2(R2)                    ;
                    62          03          20  3A 0003E        LOCC    #32, #3, (R2)                       ; 0575
                                            02  12 00042        BNEQ    1$                                  ;
                                            51  D4 00044        CLRL    R1                                  ;
                                            51  D5 00046 1$:    TSTL    PTR                                 ;
                                            05  12 00048        BNEQ    2$                                  ;
                                51          03  D0 0004A        MOVL    #3, R1                              ;
                                            03  11 0004D        BRB     3$                                  ;
                                51          52  C2 0004F 2$:    SUBL2   R2, R1                              ;
                                50          51  D0 00052 3$:    MOVL    R1, R0                              ; 0577
                                                04 00055        RET                                         ;

; Routine Size:  86 bytes,    Routine Base:  CODE + 03A3
```

```
;    0579  1      %sbttl 'format_ultrix'
;    0580  1      routine format_ultrix (length, dir, buffer, count) : novalue =
;    0581  1
;    0582  1      !++
;    0583  1      ! Functional description:
;    0584  1      !
;    0585  1      !       This routine will format an ultrix directory record
;    0586  1      !
;    0587  1      !       Record format from $GET
;    0588  1      !       31                                  0
;    0589  1      !       ----------------------------------+
;    0590  1      !       |       INODE NUMBER              |
;    0591  1      !       ----------------------------------|
;    0592  1      !       | Name len    |  Structure len    |
;    0593  1      !       +----------------------------------|
;    0594  1      !       |  file name padded to next       |
;    0595  1      !       . longword boundary               .
;    0596  1      !       .                                 .
;    0597  1      !       .                                 .
;    0598  1      !
;    0599  1      !
;    0600  1      !
;    0601  1      ! Formal parameters:
;    0602  1      !
;    0603  1      !       length  length of record
;    0604  1      !       dir     Address of record
;    0605  1      !       buffer  Address of descriptor of output buffer
;    0606  1      !
;    0607  1      !       count   address of longword to get count of files
;    0608  1      !--
;    0609  1
;    0610  2          begin
;    0611  2
;    0612  2          local
;    0613  2              desc : block [8, byte];                 ! Desc. of area for fao output
;    0614  2
;    0615  2          map
;    0616  2              dir : ref block [, byte],               ! Address of record
;    0617  2              buffer : ref block [8, byte];           ! Address of buffer descriptor
;    0618  2
;    0619  2      IF .WIDE_FLAG
;    0620  3                  THEN BEGIN
;    0621  3                       if (.buffer [dsc$w_length] mod 132) neq 0
;    0622  3                       THEN                                                              !
;    0623  3                            buffer [dsc$w_length] = .buffer [dsc$w_length] + 132 - (.buffer [dsc$w_length] mod 132);
;    0624  3                       desc [dsc$w_length] = 132;                                        !
;    0625  3                       desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];
;    0626  3                       $fao ($ascid ('!AD'), desc, desc, .dir[6,0,16,0], dir[8,0,32,0]);
;    0627  3                       END
;    0628  3                  ELSE BEGIN
;    0629  3                       if (.buffer [dsc$w_length] mod 20) neq 0
;    0630  3                          then
;    0631  3                           buffer [dsc$w_length] = .buffer [dsc$w_length] + 20 - (.buffer [dsc$w_length] mod 20);
;    0632  3
;    0633  3                       desc [dsc$w_length] = 19;
;    0634  3                       desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];
;    0635  4                       $fao ($ascid ('!19<!AD!>'),desc , desc, .dir[6,0,16,0], dir[8,0,32,0] ) !
```

```
;   0636  2                 END;
;   0637  2           PutBuffer (.Count, .Buffer, Desc)
;   0638  2
;   0639  1       end;


                                                                  .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                  44  41  21  000F9 P.AAP:  .ASCII  \!AD\                                  ;
                                    00000003  000FC P.AAO:  .LONG   3                                      ;
                                    00000000' 00100         .ADDRESS P.AAP                                 ;
                  3E  21  44  41  21  3C  39  31  21  00104 P.AAR:  .ASCII  \!19<!AD!>\                            ;
                                                0010D         .BLKB   3
                                    00000009  00110 P.AAQ:  .LONG   9                                      ;
                                    00000000' 00114         .ADDRESS P.AAR                                 ;

                                                                  .EXTRN  SYS$FAO

                                                                  .PSECT  CODE,NOWRT,  SHR,2

                                        001C 00000 FORMAT_ULTRIX:
                                                                  .WORD   Save R2,R3,R4                          ; 0580
                          5E            08  C2 00002            SUBL2   #8, SP                                 ;
                          52       0C   AC  D0 00005            MOVL    BUFFER, R2                             ; 0621
                          50       08   AC  D0 00009            MOVL    DIR, R0                                ; 0626
                          53       08   A0  9E 0000D            MOVAB   8(R0), R3                              ;
                          43 00000000'  EF  E9 00011            BLBC    WIDE_FLAG, 2$                          ; 0619
                          51            62  3C 00018            MOVZWL  (R2), R1                               ; 0621
      7E          00      51            01  7A 0001B            EMUL    #1, R1, #0, -(SP)                      ;
      51          51      8E 00000084   8F  7B 00020            EDIV    #132, (SP)+, R1, R1                    ;
                                        51  D5 00029            TSTL    R1                                     ;
                                        0D  13 0002B            BEQL    1$                                     ;
                          54            62  3C 0002D            MOVZWL  (R2), R4                               ; 0623
                  51      54            51  C3 00030            SUBL3   R1, R4, R1                             ;
                  62      51     0084   8F  A1 00034            ADDW3   #132, R1, (R2)                         ;
                          6E       84   8F  9B 0003A 1$:        MOVZBW  #132, DESC                             ; 0624
                          51            62  3C 0003E            MOVZWL  (R2), R1                               ; 0625
                     04   AE       04 B241  9E 00041            MOVAB   @4(R2)[R1], DESC+4                     ;
                                        53  DD 00047            PUSHL   R3                                     ; 0626
                          7E       06   A0  3C 00049            MOVZWL  6(R0), -(SP)                           ;
                                   08   AE  9F 0004D            PUSHAB  DESC                                   ;
                                   0C   AE  9F 00050            PUSHAB  DESC                                   ;
                             00000000'  EF  9F 00053            PUSHAB  P.AAO                                  ;
                                        3A  11 00059            BRB     4$                                     ;
                          51            62  3C 0005B 2$:        MOVZWL  (R2), R1                               ; 0629
      7E          00      51            01  7A 0005E            EMUL    #1, R1, #0, -(SP)                      ;
      51          51      8E            14  7B 00063            EDIV    #20, (SP)+, R1, R1                     ;
                                        51  D5 00068            TSTL    R1                                     ;
                                        0B  13 0006A            BEQL    3$                                     ;
                          54            62  3C 0006C            MOVZWL  (R2), R4                               ; 0631
                  51      54            51  C3 0006F            SUBL3   R1, R4, R1                             ;
                  62      51            14  A1 00073            ADDW3   #20, R1, (R2)                          ;
                          6E            13  B0 00077 3$:        MOVW    #19, DESC                              ; 0633
                          51            62  3C 0007A            MOVZWL  (R2), R1                               ; 0634
                     04   AE       04 B241  9E 0007D            MOVAB   @4(R2)[R1], DESC+4                     ;
                                        53  DD 00083            PUSHL   R3                                     ; 0635
```

```
                        7E        06   A0  3C 00085        MOVZWL  6(R0), -(SP)                          ;
                                  08   AE  9F 00089        PUSHAB  DESC                                  ;
                                  0C   AE  9F 0008C        PUSHAB  DESC                                  ;
                          00000000'    EF  9F 0008F        PUSHAB  P.AAQ                                 ;
              00000000G  00            05  FB 00095 4$:    CALLS   #5, SYS$FAO                           ;
                                4004   8F  BB 0009C        PUSHR   #^M<R2,SP>                            ; 0637
                                  10   AC  DD 000A0        PUSHL   COUNT                                 ;
                  FE8D   CF            03  FB 000A3        CALLS   #3, PUTBUFFER                         ;
                                           04 000A8        RET                                           ; 0639

; Routine Size:  169 bytes,    Routine Base:  CODE + 03F9


;   0640  1
;   0641  1
```

```
;    0643  1      %sbttl 'format_ods1'
;    0644  1      routine format_ods1 (length, dir, buffer, count) : novalue =
;    0645  1
;    0646  1      !++
;    0647  1      ! Functional description:
;    0648  1      !
;    0649  1      !       This routine will format a ODS-1 directory record for STANDALONE        [27]                    !G:9
;    0650  1      !
;    0651  1      ! Formal parameters:
;    0652  1      !
;    0653  1      !       length  length of record
;    0654  1      !       dir     Address of record
;    0655  1      !       buffer  Address of descriptor of output buffer
;    0656  1      !
;    0657  1      !       count   address of longword to get count of files
;    0658  1      !--
;    0659  1
;    0660  2          begin
;    0661  2
;    0662  2          local
;    0663  2              desc : block [8, byte],                 ! Desc. of area for fao output
;    0664  2              name : vector [16, byte],               ! ASCIC file name
;    0665  2              type : vector [4, byte];                ! ascic file type
;    0666  2
;    0667  2          map
;    0668  2              dir : ref block [, byte],               ! Address of record
;    0669  2              buffer : ref block [8, byte];           ! Address of buffer descriptor
;    0670  2
;    0671  2          if .dir [0, 0, 16, 0] neq 0
;    0672  2          then
;    0673  3              begin
;    0674  3
;    0675  3              if (.buffer [dsc$w_length] mod 20) neq 0
;    0676  3              then
;    0677  3                  buffer [dsc$w_length] = .buffer [dsc$w_length] + 20 - (.buffer [dsc$w_length] mod 20);
;    0678  3
;    0679  3              name [0] = rad50 (.dir [6, 0, 16, 0], name [1]) + rad50 (.dir [8, 0, 16, 0], name [4]) + rad50 (.dir
;    0680  3                  [10, 0, 16, 0], name [7]);
;    0681  3              type [0] = rad50 (.dir [12, 0, 16, 0], type [1]);
;    0682  3              desc [dsc$w_length] = 19;
;    0683  3              desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];
;    0684  3              $fao ($ascid ('!19<!AC.!AC;!UW!>'), desc, desc, name, type, .dir [14, 0, 16, 0]);       !               [10]
;    0685  3              PutBuffer (.Count, .Buffer, Desc)
;    0686  2              end;
;    0687  2
;    0688  1          end;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

57  55  21  3B  43  41  21  2E  43  41  21  3C  39  31  21  00118 P.AAT:  .ASCII  \!19<!AC.!AC;!UW!>\                    ;
                                                    3E  21  00127                                                       ;
                                                            00129          .BLKB   3
                                                00000011    0012C P.AAS:  .LONG   17                                    ;
                                                00000000'   00130          .ADDRESS P.AAT                           ;
```

```
                                                                  .PSECT  CODE,NOWRT,  SHR,2

                                          003C 00000 FORMAT_ODS1:
                                                                  .WORD   Save R2,R3,R4,R5                          ; 0644
                          55      FEFB   CF  9E 00002             MOVAB   RAD50, R5                                 ;
                          5E             1C  C2 00007             SUBL2   #28, SP                                   ;
                          53        08   AC  D0 0000A             MOVL    DIR, R3                                   ; 0671
                                         63  B5 0000E             TSTW    (R3)                                      ;
                                         01  12 00010             BNEQ    1$                                        ;
                                             04 00012             RET                                               ;
                          54        0C   AC  D0 00013 1$:         MOVL    BUFFER, R4                                ; 0675
                          50             64  3C 00017             MOVZWL  (R4), R0                                  ;
     7E          00       50             01  7A 0001A             EMUL    #1, R0, #0, -(SP)                         ;
     50          50       8E             14  7B 0001F             EDIV    #20, (SP)+, R0, R0                        ;
                                         50  D5 00024             TSTL    R0                                        ;
                                         0B  13 00026             BEQL    2$                                        ;
                          51             64  3C 00028             MOVZWL  (R4), R1                                  ; 0677
                 50       51             50  C3 0002B             SUBL3   R0, R1, R0                                ;
                 64       50             14  A1 0002F             ADDW3   #20, R0, (R4)                             ;
                                   05   AE  9F 00033 2$:         PUSHAB  NAME+1                                    ; 0679
                          7E        06   A3  3C 00036             MOVZWL  6(R3), -(SP)                              ;
                          65             02  FB 0003A             CALLS   #2, RAD50                                 ;
                          52             50  D0 0003D             MOVL    R0, R2                                    ;
                                   08   AE  9F 00040             PUSHAB  NAME+4                                    ;
                          7E        08   A3  3C 00043             MOVZWL  8(R3), -(SP)                              ;
                          65             02  FB 00047             CALLS   #2, RAD50                                 ;
                          52             50  C0 0004A             ADDL2   R0, R2                                    ;
                                   0B   AE  9F 0004D             PUSHAB  NAME+7                                    ; 0680
                          7E        0A   A3  3C 00050             MOVZWL  10(R3), -(SP)                             ;
                          65             02  FB 00054             CALLS   #2, RAD50                                 ;
          04   AE         52             50  81 00057             ADDB3   R0, R2, NAME                              ;
                                   01   AE  9F 0005C             PUSHAB  TYPE+1                                    ; 0681
                          7E        0C   A3  3C 0005F             MOVZWL  12(R3), -(SP)                             ;
                          65             02  FB 00063             CALLS   #2, RAD50                                 ;
                          6E             50  90 00066             MOVB    R0, TYPE                                  ;
                     14   AE             13  B0 00069             MOVW    #19, DESC                                 ; 0682
                          50             64  3C 0006D             MOVZWL  (R4), R0                                  ; 0683
                     18   AE        04 B440  9E 00070             MOVAB   @4(R4)[R0], DESC+4                        ;
                          7E        0E   A3  3C 00076             MOVZWL  14(R3), -(SP)                             ; 0684
                                   04   AE  9F 0007A             PUSHAB  TYPE                                      ;
                                   0C   AE  9F 0007D             PUSHAB  NAME                                      ;
                                   20   AE  9F 00080             PUSHAB  DESC                                      ;
                                   24   AE  9F 00083             PUSHAB  DESC                                      ;
                             00000000'  EF  9F 00086             PUSHAB  P.AAS                                     ;
                 00000000G  00          06  FB 0008C             CALLS   #6, SYS$FAO                               ;
                                   14   AE  9F 00093             PUSHAB  DESC                                      ; 0685
                                         54  DD 00096             PUSHL   R4                                        ;
                                   10   AC  DD 00098             PUSHL   COUNT                                     ;
                     8B   A5             03  FB 0009B             CALLS   #3, PUTBUFFER                             ;
                                             04 0009F             RET                                               ; 0688

; Routine Size:  160 bytes,    Routine Base:  CODE + 04A2
```

```
;    0690  1     %sbttl 'format_ods2'
;    0691  1     routine format_ods2 (length, dir, buffer, count) : novalue =
;    0692  1
;    0693  1     !++
;    0694  1     ! Functional description:
;    0695  1     !
;    0696  1     !       This routine will format an ODS-2 directory record for STANDALONE        [27]                    !G:9
;    0697  1     !
;    0698  1     ! Formal parameters:
;    0699  1     !
;    0700  1     !       length  length of record
;    0701  1     !       dir     Address of record
;    0702  1     !       buffer  Address of descriptor of output buffer
;    0703  1     !
;    0704  1     !       count   address of longword to get file count
;    0705  1     !--
;    0706  1
;    0707  2         begin
;    0708  2
;    0709  2         local
;    0710  2             desc : block [8, byte];
;    0711  2
;    0712  2         map
;    0713  2             dir : ref block [, byte],               ! Address of record
;    0714  2             buffer : ref block [8, byte];           ! Address of buffer descriptor
;    0715  2
;    0716  2         bind
;    0717  2             ver_offset = -2 + dir$c_length + ((.dir [-2 + dir$b_namecount] + 1) and not 1);
;    0718  2
;    0719  2         incr index from ver_offset to .length - 1 by 8 do
;    0720  3             begin
;    0721  3             IF .WIDE_FLAG                                                                                   !
;    0722  4                THEN BEGIN                                                                                   !
;    0723  4                     if (.buffer [dsc$w_length] mod 132) neq 0                                               !
;    0724  4                        then                                                                                 !
;    0725  4                         buffer [dsc$w_length] = .buffer [dsc$w_length] + 132 - (.buffer [dsc$w_length] mod 132);    !
;    0726  4                     desc [dsc$w_length] = 132;                                                              !
;    0727  4                     desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];                !
;    0728  5                     $fao ($ascid ('!AC;!UW'), desc, desc, dir [-2 + dir$b_namecount], .dir [.index, 0, 16, 0])  !
;    0729  4                     END                                                                                     !
;    0730  4                ELSE BEGIN                                                                                   !
;    0731  4                     if (.buffer [dsc$w_length] mod 20) neq 0
;    0732  4                        then
;    0733  4                         buffer [dsc$w_length] = .buffer [dsc$w_length] + 20 - (.buffer [dsc$w_length] mod 20);
;    0734  4
;    0735  4                     desc [dsc$w_length] = 19;
;    0736  4                     desc [dsc$a_pointer] = .buffer [dsc$a_pointer] + .buffer [dsc$w_length];
;    0737  5                     $fao ($ascid ('!19<!AC;!UW!>'), desc, desc, dir [-2 + dir$b_namecount], .dir [.index, 0, 16, 0])
;    0738  3                     END;
;    0739  3             PutBuffer (.Count, .Buffer, Desc)
;    0740  2             end;
;    0741  2
;    0742  1         end;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2
```

```
                                57  55  21  3B  43  41  21  00134 P.AAV:  .ASCII  \!AC;!UW\                          ;
                                                            0013B         .BLKB   1
                                                  00000007  0013C P.AAU:  .LONG   7                                  ;
                                                  00000000' 00140         .ADDRESS P.AAV                             ;
     3E  21  57  55  21  3B  43  41  21  3C  39  31  21  00144 P.AAX:  .ASCII  \!19<!AC;!UW!>\                     ;
                                                            00151         .BLKB   3
                                                  0000000D  00154 P.AAW:  .LONG   13                                 ;
                                                  00000000' 00158         .ADDRESS P.AAX                             ;


                                                                          .PSECT  CODE,NOWRT,  SHR,2

                                                     003C 00000 FORMAT_ODS2:
                                                                          .WORD   Save R2,R3,R4,R5                    ; 0691
                                      5E             08  C2 00002         SUBL2   #8, SP                              ;
                                      52        08   AC  7D 00005         MOVQ    DIR, R2                             ; 0717
                                      50        03   A2  9A 00009         MOVZBL  3(R2), R0                           ;
                                                     50  D6 0000D         INCL    R0                                  ;
                                      50             01  8A 0000F         BICB2   #1, R0                              ;
                                      54        04   A0  9E 00012         MOVAB   4(R0), R4                           ; 0719
                        55        04  AC             01  C3 00016         SUBL3   #1, LENGTH, R5                      ;
                                                  00A0  31 0001B         BRW     6$                                  ; 0739
                                      46 00000000'   EF  E9 0001E 1$:    BLBC    WIDE_FLAG, 3$                       ; 0721
                                      50             63  3C 00025         MOVZWL  (R3), R0                            ; 0723
        7E              00            50             01  7A 00028         EMUL    #1, R0, #0, -(SP)                   ;
        50              50            8E 00000084    8F  7B 0002D         EDIV    #132, (SP)+, R0, R0                 ;
                                                     50  D5 00036         TSTL    R0                                  ;
                                                     0D  13 00038         BEQL    2$                                  ;
                                      51             63  3C 0003A         MOVZWL  (R3), R1                            ; 0725
                        50            51             50  C3 0003D         SUBL3   R0, R1, R0                          ;
                        63            50      0084   8F  A1 00041         ADDW3   #132, R0, (R3)                      ;
                                      6E        84   8F  9B 00047 2$:     MOVZBW  #132, DESC                          ; 0726
                                      50             63  3C 0004B         MOVZWL  (R3), R0                            ; 0727
                                04    AE        04 B340  9E 0004E         MOVAB   @4(R3)[R0], DESC+4                  ;
                                                  6442  9F 00054         PUSHAB  (INDEX)[R2]                         ; 0728
                                      7E             9E  3C 00057         MOVZWL  @(SP)+, -(SP)                       ;
                                                03   A2  9F 0005A         PUSHAB  3(R2)                               ;
                                                08   AE  9F 0005D         PUSHAB  DESC                                ;
                                                0C   AE  9F 00060         PUSHAB  DESC                                ;
                                         00000000'   EF  9F 00063         PUSHAB  P.AAU                               ;
                                                     3D  11 00069         BRB     5$                                  ;
                                      50             63  3C 0006B 3$:     MOVZWL  (R3), R0                            ; 0731
        7E              00            50             01  7A 0006E         EMUL    #1, R0, #0, -(SP)                   ;
        50              50            8E             14  7B 00073         EDIV    #20, (SP)+, R0, R0                  ;
                                                     50  D5 00078         TSTL    R0                                  ;
                                                     0B  13 0007A         BEQL    4$                                  ;
                                      51             63  3C 0007C         MOVZWL  (R3), R1                            ; 0733
                        50            51             50  C3 0007F         SUBL3   R0, R1, R0                          ;
                        63            50             14  A1 00083         ADDW3   #20, R0, (R3)                       ;
                                      6E             13  B0 00087 4$:     MOVW    #19, DESC                           ; 0735
                                      50             63  3C 0008A         MOVZWL  (R3), R0                            ; 0736
                                04    AE        04 B340  9E 0008D         MOVAB   @4(R3)[R0], DESC+4                  ;
                                                  6442  9F 00093         PUSHAB  (INDEX)[R2]                         ; 0737
                                      7E             9E  3C 00096         MOVZWL  @(SP)+, -(SP)                       ;
                                                03   A2  9F 00099         PUSHAB  3(R2)                               ;
```

```
                                  08   AE  9F 0009C        PUSHAB  DESC                                  ;
                                  0C   AE  9F 0009F        PUSHAB  DESC                                  ;
                          00000000'    EF  9F 000A2        PUSHAB  P.AAW                                 ;
                 00000000G  00         05  FB 000A8 5$:    CALLS   #5, SYS$FAO                           ;
                                4008   8F  BB 000AF        PUSHR   #^M<R3,SP>                            ; 0739
                                  10   AC  DD 000B3        PUSHL   COUNT                                 ;
                     FD31   CF         03  FB 000B6        CALLS   #3, PUTBUFFER                         ;
                            54         08  C0 000BB        ADDL2   #8, INDEX                             ;
                            55         54  D1 000BE 6$:    CMPL    INDEX, R5                             ;
                                       03  14 000C1        BGTR    7$                                    ;
                                     FF58  31 000C3        BRW     1$                                    ;
                                           04 000C6 7$:    RET                                           ; 0742

; Routine Size:  199 bytes,    Routine Base:  CODE + 0542
```

```
;    0744  1     %sbttl 'RT-11 directory'
;    0745  1     routine rt11_direct (q_device, count) =                                                    !G:6
;    0746  2         begin
;    0747  2
;    0748  2         bind
;    0749  2             l = .q_device : word;
;    0750  2
;    0751  2         local
;    0752  2             segment,
;    0753  2             extra_length,
;    0754  2             address_first,
;    0755  2             RPB : Ref Block [, byte],       ! Fake RPB                                         [09]
;    0756  2             segblk : bblock [1024], ! Buffer to read segment
;    0757  2             t_buffer : bblock [80], ! Buffer for directory line
;    0758  2             name : vector [7, byte],        ! Buffer for ASCIC file name
;    0759  2             type : vector [4, byte],        ! Buffer for ASCIC file type
;    0760  2             OutBuffer : BBlock [Dsc$C_S_Bln],       ! Descriptor for buffer                    [10]
;    0761  2             q_buffer : bblock [dsc$c_s_bln];        ! Descriptor for FAO
;    0762  2
;    0763  2         RPB = Dsr$Allocate_Pseudo_RPB (BTD$K_Console, 0, 0, 0, 1, 0);       !                  [09]
;    0764  2         segment = 1;
;    0765  2         OutBuffer [Dsc$W_Length] = 0;
;    0766  2         OutBuffer [Dsc$A_Pointer] = T_Buffer;
;    0767  2         $ds_printf ($ascic ('!/!/Directory !AS!/!/'), .q_device);                              !G:10
;    0768  2         ch$fill (%c' ', 80, t_buffer);
;    0769  2         If .FileSpecMatch[Dsc$W_Length] Neq 0
;    0770  2             Then FileSpecMatch[Dsc$W_Length] = .FileSpecMatch[Dsc$W_Length] - 2;    ! Shorten string by 2, don't    [21]
;    0771  2                                                                                     ! look for version number.      [21]
;    0772  2
;    0773  2         while 1 do
;    0774  3             begin
;    0775  3
;    0776  3             local
;    0777  3                 status;
;    0778  3
;    0779  3             if .segment eql 0 then exitloop;
;    0780  3
;    0781  4             if not (status = boo$qio (segblk,       ! Read directory segment                   [03]
;    0782  4                 1024,                   !  two blocks long                                 [03]
;    0783  4                 4 * .segment*2,         !  from disk logical block                         [03]
;    0784  4                 io$_readlblk,           !                                                  [03]
;    0785  4                 0,                      !  physical address                                [03]
;    0786  4                 .rpb))                  !  address of rpb                                  [03]
;    0787  3             then return .status;        !                                                  [03]
;    0788  3
;    0789  3             segment = .segblk [rt$w_segment];       ! Link to next segment
;    0790  3             extra_length = .segblk [rt$w_extralen]; ! Variable part size
;    0791  3             address_first = segblk [rt$w_first] - rt$_fixed - .extra_length;
;    0792  3
;    0793  3             while 1 do
;    0794  4                 begin
;    0795  4
;    0796  4                 bind
;    0797  4                     entry = (address_first = .address_first + rt$_fixed + .extra_length) : bblock;
;    0798  4
;    0799  4                 if .entry [rt$v_endseg] then exitloop;      ! End of segment
;    0800  4
```

```
;   0801  4                 if .entry [rt$v_perm]                   ! Only list permanent files
;   0802  4                 then
;   0803  5                     begin
;   0804  6                     name [0] = rad50 (.(entry [rt$l_filename])<0, 16>, name [1]) + rad50 (.(entry [rt$l_filename]
;   0805  5                         )<16, 16>, name [4]);
;   0806  5                     type [0] = rad50 (.entry [rt$w_filetype], type [1]);
;   0807  5                     q_buffer [dsc$w_length] = 19;   ! Set up FAO output buffer
;   0808  5                     q_buffer [dsc$a_pointer] = .OutBuffer [Dsc$A_Pointer] + .OutBuffer [Dsc$W_Length];      !       [10]
;   0809  5                     $fao ($ascid ('!19<!AC.!AC!>'), q_buffer, q_buffer, name, type);        !                       [10]
;   0810  5                     PutBuffer (.Count, OutBuffer, Q_Buffer) !                                                       [10]
;   0811  5                     end
;   0812  5
;   0813  3                 end;                                    !                                                           [08]
;   0814  3
;   0815  3                 If .ControlCFlag            !                                                                       [09]
;   0816  3                 Then                        !                                                                       [09]
;   0817  4                     Begin                   !                                                                       [09]
;   0818  4                     Dsr$DeAllocate_Pseudo_RPB (.RPB);       !                                                       [09]
;   0819  4                     Return Rms$_Normal      ! Abort gracefully on ^C                                                [09]
;   0820  3                     End;                    !                                                                       [09]
;   0821  3
;   0822  2             end;
;   0823  2
;   0824  2         if .OutBuffer [Dsc$W_Length] gtr 0 then $ds_printf ($Ascic ('!AS!/'), OutBuffer);   !                       [10]
;   0825  2
;   0826  2         Dsr$DeAllocate_Pseudo_RPB (.RPB);
;   0827  2         rms$_normal
;   0828  1         end;                                            ! of rt11_direct


                                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20  79  72  6F  74  63  65  72  69  44  2F  21  2F  21  15  0015C P.AAY:  .ASCII  <21>\!/!/Directory !AS!/!/\                ;
                                2F  21  2F  21  53  41  21  0016B                                                                ;
        3E  21  43  41  21  2E  43  41  21  3C  39  31  21  00172 P.ABA:  .ASCII  \!19<!AC.!AC!>\                              ;
                                                            0017F         .BLKB   1
                                                    0000000D  00180 P.AAZ:  .LONG   13                                          ;
                                                    00000000' 00184         .ADDRESS P.ABA                                      ;
                                    2F  21  53  41  21  05  00188 P.ABB:  .ASCII  <5>\!AS!/\                                    ;


                                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                        0FFC 00000 RT11_DIRECT:
                                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0745
                                    5B 00000000G  9F  9E 00002          MOVAB   @#DS$PRINTF, R11                            ;
                                    5A 00000000'  EF  9E 00009          MOVAB   FILESPECMATCH, R10                          ;
                                    59 00000000'  EF  9E 00010          MOVAB   P.AAY, R9                                   ;
                                    58     FD7F   CF  9E 00017          MOVAB   RAD50, R8                                   ;
                                    5E     FB94   CE  9E 0001C          MOVAB   -1132(SP), SP                               ;
                                    7E            01  7D 00021          MOVQ    #1, -(SP)                                   ; 0763
                                                  7E  7C 00024          CLRQ    -(SP)                                       ;
                                                  7E  D4 00026          CLRL    -(SP)                                       ;
                                    7E       40   8F  9A 00028          MOVZBL  #64, -(SP)                                  ;
                        00000000G   EF            06  FB 0002C          CALLS   #6, DSR$ALLOCATE_PSEUDO_RPB                 ;
```

```
                                57          50  D0 00033        MOVL    R0, RPB                                  ;
                                56          01  D0 00036        MOVL    #1, SEGMENT                              ; 0764
                                        0C  AE  B4 00039        CLRW    OUTBUFFER                                ; 0765
                        10      AE      1C  AE  9E 0003C        MOVAB   T_BUFFER, OUTBUFFER+4                    ; 0766
                                        04  AC  DD 00041        PUSHL   Q_DEVICE                                 ; 0767
                                            59  DD 00044        PUSHL   R9                                       ;
                                6B          02  FB 00046        CALLS   #2, DS$PRINTF                            ;
  0050   8F         20          6E          00  2C 00049        MOVC5   #0, (SP), #32, #80, T_BUFFER             ; 0768
                                        1C  AE     00050                                                         ;
                                            6A  B5 00052        TSTW    FILESPECMATCH                            ; 0769
                                            03  13 00054        BEQL    1$                                       ;
                                6A          02  A2 00056        SUBW2   #2, FILESPECMATCH                        ; 0770
                                            56  D5 00059 1$:    TSTL    SEGMENT                                  ; 0779
                                            03  12 0005B        BNEQ    2$                                       ;
                                          00A2  31 0005D        BRW     6$                                       ;
                                            57  DD 00060 2$:    PUSHL   RPB                                      ; 0786
                                7E          21  7D 00062        MOVQ    #33, -(SP)                               ; 0781
                  7E            56          01  78 00065        ASHL    #1, SEGMENT, -(SP)                       ; 0783
                                6E          04  C0 00069        ADDL2   #4, (SP)                                 ;
                                7E    0400  8F  3C 0006C        MOVZWL  #1024, -(SP)                             ; 0781
                                      0080  CE  9F 00071        PUSHAB  SEGBLK                                   ;
                  00000000G     EF          06  FB 00075        CALLS   #6, BOO$QIO                              ;
                                01          50  E8 0007C        BLBS    STATUS, 3$                               ;
                                                04 0007F        RET                                              ;
                                56      6E  AE  3C 00080 3$:    MOVZWL  SEGBLK+2, SEGMENT                        ; 0789
                                54      72  AE  3C 00084        MOVZWL  SEGBLK+6, EXTRA_LENGTH                   ; 0790
                                50      68  AE  9E 00088        MOVAB   SEGBLK-4, R0                             ; 0791
                  52            50          54  C3 0008C        SUBL3   EXTRA_LENGTH, R0, ADDRESS_FIRST          ;
                                52    0E A442  9E 00090 4$:    MOVAB   14(EXTRA_LENGTH)[ADDRESS_FIRST], -       ; 0797
                                                                        ADDRESS_FIRST                            ;
                                53          52  D0 00095        MOVL    ADDRESS_FIRST, R3                        ;
                  5F            63          0B  E0 00098        BBS     #11, (R3), 5$                            ; 0799
                  F0            63          0A  E1 0009C        BBC     #10, (R3), 4$                            ; 0801
                                        15  AE  9F 000A0        PUSHAB  NAME+1                                   ; 0804
                                7E      02  A3  3C 000A3        MOVZWL  2(R3), -(SP)                             ;
                                68          02  FB 000A7        CALLS   #2, RAD50                                ;
                                55          50  D0 000AA        MOVL    R0, R5                                   ;
                                        18  AE  9F 000AD        PUSHAB  NAME+4                                   ; 0805
                                7E      04  A3  3C 000B0        MOVZWL  4(R3), -(SP)                             ;
                                68          02  FB 000B4        CALLS   #2, RAD50                                ;
            14    AE            55          50  81 000B7        ADDB3   R0, R5, NAME                             ;
                                        01  AE  9F 000BC        PUSHAB  TYPE+1                                   ; 0806
                                7E      06  A3  3C 000BF        MOVZWL  6(R3), -(SP)                             ;
                                68          02  FB 000C3        CALLS   #2, RAD50                                ;
                                6E          50  90 000C6        MOVB    R0, TYPE                                 ;
                        04      AE          13  B0 000C9        MOVW    #19, Q_BUFFER                            ; 0807
                                50      0C  AE  3C 000CD        MOVZWL  OUTBUFFER, R0                            ; 0808
                        08      AE      10 BE40  9E 000D1        MOVAB   @OUTBUFFER+4[R0], Q_BUFFER+4             ;
                                            5E  DD 000D7        PUSHL   SP                                       ; 0809
                                        18  AE  9F 000D9        PUSHAB  NAME                                     ;
                                        0C  AE  9F 000DC        PUSHAB  Q_BUFFER                                 ;
                                        10  AE  9F 000DF        PUSHAB  Q_BUFFER                                 ;
                                        24  A9  9F 000E2        PUSHAB  P.AAZ                                    ;
                  00000000G     00          05  FB 000E5        CALLS   #5, SYS$FAO                              ;
                                        04  AE  9F 000EC        PUSHAB  Q_BUFFER                                 ; 0810
                                        10  AE  9F 000EF        PUSHAB  OUTBUFFER                                ;
                                        08  AC  DD 000F2        PUSHL   COUNT                                    ;
```

```
                    8B   A8          03  FB 000F5        CALLS   #3, PUTBUFFER                          ;
                                     95  11 000F9        BRB     4$                                     ;
                         11      08  AA  E8 000FB 5$:    BLBS    CONTROLCFLAG, 7$                       ; 0815
                                   FF57  31 000FF        BRW     1$                                     ;
                                 0C  AE  B5 00102 6$:    TSTW    OUTBUFFER                              ; 0824
                                     09  13 00105        BEQL    7$                                     ;
                                 0C  AE  9F 00107        PUSHAB  OUTBUFFER                              ;
                                 2C  A9  9F 0010A        PUSHAB  P.ABB                                  ;
                         6B          02  FB 0010D        CALLS   #2, DS$PRINTF                          ;
                                     57  DD 00110 7$:    PUSHL   RPB                                    ; 0826
                 00000000G  EF       01  FB 00112        CALLS   #1, DSR$DEALLOCATE_PSEUDO_RPB          ;
                         50 00010001 8F  D0 00119        MOVL    #65537, R0                             ; 0828
                                         04 00120        RET                                            ;

; Routine Size:  289 bytes,    Routine Base:  CODE + 0609
```

```
;    0830  1     %sbttl 'ANSI directory'
;    0831  1     routine ansi_direct (filespec, count) =
;    0832  1
;    0833  1     !++
;    0834  1     ! Functional description:
;    0835  1     !
;    0836  1     !       Search through a tape to find all files, and list them on the
;    0837  1     !       console.
;    0838  1     !
;    0839  1     !       If double tape marks are found instead of a file header the
;    0840  1     !       tape is rewound and the search continued until the first file found is
;    0841  1     !       encountered again.
;    0842  1     !
;    0843  1     !       STANDALONE              [27]                                    !G:9
;    0844  1     !                                                                       !G:9
;    0845  1     ! Formal parameters:
;    0846  1     !       filespec        address of filespec block
;    0847  1     !       count           address of longword to get file count
;    0848  1     !
;    0849  1     ! Implicit inputs:
;    0850  1     !       none
;    0851  1     !
;    0852  1     ! Outputs:
;    0853  1     !       count           count of files
;    0854  1     !
;    0855  1     ! Implicit outputs:
;    0856  1     !       none
;    0857  1     !
;    0858  1     ! Routine value:
;    0859  1     !
;    0860  1     !       Standard RMS completion codes
;    0861  1     !
;    0862  1     ! Side effects:
;    0863  1     !
;    0864  1     !       The device and associated channel are accessed
;    0865  1     !
;    0866  1     !--
;    0867  1
;    0868  2         begin
;    0869  2         external routine
;    0870  2             sys$search : addressing_mode (absolute),
;    0871  2             sys$open : addressing_mode (absolute);
;    0872  2
;    0873  2         bind routine
;    0874  2             next_file =
;    0875  2
;    0876  2                 if .dsa$v_user then sys$search else sys$open;
;    0877  2
;    0878  2         local
;    0879  2             filenam : bblock [nam$c_maxrss],
;    0880  2             filedesc : bblock [dsc$c_s_bln],
;    0881  2             res_len,
;    0882  2             status,
;    0883  2             fab : $fab_decl,                        ! FAB for open
;    0884  2             nam : $nam_decl,
;    0885  2             result : bblock [nam$c_maxrss],         ! Space for resultant string
;    0886  2             expand : bblock [nam$c_maxrss],         ! Space for expanded string
```

```
;    0887  2              first_file : bblock [132],              ! First file found
;    0888  2              first_len,
;    0889  2              buffer : vector [132, byte],            ! Space for output
;    0890  2              OutBuffer : BBlock [8];                 ! Descriptor for output                                [10]
;    0891  2
;    0892  2          ch$fill (%c' ', 132, buffer);               ! Clear output buffer
;    0893  2          OutBuffer [Dsc$W_Length] = 0;               !                                                      [10]
;    0894  2          OutBuffer [Dsc$A_Pointer] = Buffer;         !                                                      [10]
;    0895  2          first_len = 132;
;    0896  2          ch$fill (0, 132, first_file);
;    0897  2          $ds_printf ($ascic ('!/!/Directory !AS!/!/'), .filespec);                                          !G:10
;    0898  2          filedesc [dsc$w_length] = nam$c_maxrss;
;    0899  2          filedesc [dsc$a_pointer] = filenam;
;    0900  2          $fao ($ascid ('!AS*.*;*'), filedesc, filedesc, .filespec);
;    0901  2          $fab_init (fab = fab, nam = nam, fop = rwo, fna = filenam, fns = .filedesc [dsc$w_length]);
;    0902  2          $nam_init (nam = nam, ess = nam$c_maxrss, rss = nam$c_maxrss, esa = expand, rsa = result);
;    0903  2
;    0904  2          if .dsa$v_user
;    0905  2          then
;    0906  2                                                      ! Init wildcard context
;    0907  3              begin
;    0908  3
;    0909  3              local
;    0910  3                  status;
;    0911  3
;    0912  3              status = $parse (fab = fab);
;    0913  3
;    0914  3              if not .status then return .status
;    0915  3
;    0916  2              end;
;    0917  2
;    0918  2          while (status = next_file (fab); fab [fab$v_rwo] = 0; .status) do
;    0919  3              begin
;    0920  3
;    0921  3              local
;    0922  3                  nam_len,
;    0923  3                  nam_ptr;
;    0924  3
;    0925  3              if not .dsa$v_user                      ! If we did an OPEN,
;    0926  3              then                                    !  we have to CLOSE, too.
;    0927  4                  begin
;    0928  4                  status = $close (fab = fab);
;    0929  4
;    0930  4                  if not .status then exitloop
;    0931  4
;    0932  3                  end;
;    0933  3
;    0934  3              res_len = .nam [nam$b_rsl];
;    0935  3
;    0936  3              if ch$compare (.first_len, first_file, .res_len, result, 0) eql 0 then exitloop;
;    0937  3
;    0938  3              if .(first_file)<0, 8> eql 0 then ch$move (first_len = .res_len, result, first_file);
;    0939  3
;    0940  3              nam_ptr = ch$find_ch (.res_len, result, %c':') + 1;     ! Look for end of "MTAn:"
;    0941  3              nam_len = .res_len - (.nam_ptr - result);
;    0942  3              FileDesc [Dsc$W_Length] = .Nam_Len;
;    0943  3              FileDesc [Dsc$A_Pointer] = .Nam_Ptr;
```

```
;   0944  3            IF .WIDE_FLAG
;   0945  3               THEN Ch$Copy (.nam_len, .nam_ptr, %c' ', 132, buffer + .OutBuffer [Dsc$W_Length])          !      [17]
;   0946  3               ELSE Ch$Copy (.nam_len, .nam_ptr, %c' ', 19, buffer + .OutBuffer [Dsc$W_Length]);          !      [10]
;   0947  3            PutBuffer (.Count, OutBuffer, FileDesc);          !                                                [10]
;   0948  3            If .ControlCFlag Then Return Rms$_Normal;         ! Abort gracefully on ^C                         [08]
;   0949  3
;   0950  2            end;
;   0951  2
;   0952  2         if .OutBuffer [Dsc$W_Length] gtr 0 then $ds_printf ($Ascic ('!AS!/'), OutBuffer);   !                [10]
;   0953  2
;   0954  2         if .status eql rms$_nmf then status = rms$_normal;
;   0955  2
;   0956  2         .status
;   0957  1         end;                                          ! of ansi_direct


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20  79  72  6F  74  63  65  72  69  44  2F  21  2F  21  15  0018E P.ABC:  .ASCII  <21>\!/!/Directory !AS!/!/\                ;
                                2F  21  2F  21  53  41  21  0019D                                                          ;
                            2A  3B  2A  2E  2A  53  41  21  001A4 P.ABE:  .ASCII  \!AS*.*;*\                                ;
                                                00000008  001AC P.ABD:  .LONG   8                                        ;
                                                00000000' 001B0         .ADDRESS P.ABE                                   ;
                                    2F  21  53  41  21  05  001B4 P.ABF:  .ASCII  <5>\!AS!/\                                ;

                                                                        .EXTRN  SYS$OPEN, SYS$CLOSE

                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                    0FFC 00000 ANSI_DIRECT:
                                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11       ; 0831
                                    5B 00000000'  EF  9E 00002          MOVAB   P.ABC, R11                                 ;
                                    5E     FB38   CE  9E 00009          MOVAB   -1224(SP), SP                              ;
        5A 0000FE03   9F            01            04  EF 0000E          EXTZV   #4, #1, a#^X0000FE03, R10                  ; 0876
                                    09            5A  E9 00017          BLBC    R10, 1$                                    ;
                                    58 00000000G  8F  D0 0001A          MOVL    #SYS$SEARCH, R8                            ;
                                                  07  11 00021          BRB     2$                                         ;
                                    58 00000000G  8F  D0 00023 1$:      MOVL    #SYS$OPEN, R8                              ;
   0084   8F            20          6E            00  2C 0002A 2$:      MOVC5   #0, (SP), #32, #132, BUFFER                ; 0892
                                           08     AE     00031                                                             ;
                                                  6E  B4 00033          CLRW    OUTBUFFER                                  ; 0893
                                04  AE     08     AE  9E 00035          MOVAB   BUFFER, OUTBUFFER+4                        ; 0894
                                    59     84     8F  9A 0003A          MOVZBL  #132, FIRST_LEN                            ; 0895
   0084   8F            00          6E            00  2C 0003E          MOVC5   #0, (SP), #0, #132, FIRST_FILE             ; 0896
                                         008C     CE     00045                                                             ;
                                           04     AC  DD 00048          PUSHL   FILESPEC                                   ; 0897
                                                  5B  DD 0004B          PUSHL   R11                                        ;
                        00000000G   9F            02  FB 0004D          CALLS   #2, a#DS$PRINTF                            ;
                            FEF8    CD     FF     8F  9B 00054          MOVZBW  #255, FILEDESC                             ; 0898
                            FEFC    CD   FF00     CD  9E 0005A          MOVAB   FILENAM, FILEDESC+4                        ; 0899
                                           04     AC  DD 00061          PUSHL   FILESPEC                                   ; 0900
                                         FEF8     CD  9F 00064          PUSHAB  FILEDESC                                   ;
                                         FEF8     CD  9F 00068          PUSHAB  FILEDESC                                   ;
                                           1E     AB  9F 0006C          PUSHAB  P.ABD                                      ;
                        00000000G   00            04  FB 0006F          CALLS   #4, SYS$FAO                                ;
   0050   8F            00          6E            00  2C 00076          MOVC5   #0, (SP), #0, #80, $RMS_PTR                ; 0901
```

```
                                        FEA8   CD      0007D                                                                  ;
                          FEA8   CD     5003   8F   B0 00080        MOVW    #20483, $RMS_PTR                                  ;
                          FEAC   CD       80   8F   9A 00087        MOVZBL  #128, $RMS_PTR+4                                  ;
                          FEBE   CD            02   90 0008D        MOVB    #2, $RMS_PTR+22                                   ;
                          FEC7   CD            02   90 00092        MOVB    #2, $RMS_PTR+31                                   ;
                          FED0   CD     FE48   CD   9E 00097        MOVAB   NAM, $RMS_PTR+40                                  ;
                          FED4   CD     FF00   CD   9E 0009E        MOVAB   FILENAM, $RMS_PTR+44                              ;
                          FEDC   CD     FEF8   CD   90 000A5        MOVB    FILEDESC, $RMS_PTR+52                             ;
0060   8F          00            6E            00   2C 000AC        MOVC5   #0, (SP), #0, #96, $RMS_PTR                       ; 0902
                                        FE48   CD      000B3                                                                  ;
                          FE48   CD     6002   8F   B0 000B6        MOVW    #24578, $RMS_PTR                                  ;
                          FE4A   CD            01   8E 000BD        MNEGB   #1, $RMS_PTR+2                                    ;
                          FE4C   CD     0210   CE   9E 000C2        MOVAB   RESULT, $RMS_PTR+4                                ;
                          FE52   CD            01   8E 000C9        MNEGB   #1, $RMS_PTR+10                                   ;
                          FE54   CD     0110   CE   9E 000CE        MOVAB   EXPAND, $RMS_PTR+12                               ;
                                 0F            5A   E9 000D5        BLBC    R10, 3$                                           ; 0904
                                        FEA8   CD   9F 000D8        PUSHAB  FAB                                               ; 0912
                     00000000G   00            01   FB 000DC        CALLS   #1, SYS$PARSE                                     ;
                                 01            50   E8 000E3        BLBS    STATUS, 3$                                        ; 0914
                                               04      000E6        RET                                                       ;
                                        FEA8   CD   9F 000E7 3$:    PUSHAB  FAB                                               ; 0918
                                 68            01   FB 000EB        CALLS   #1, (R8)                                          ;
                                 56            50   D0 000EE        MOVL    R0, STATUS                                        ;
                          FEAC   CD       80   8F   8A 000F1        BICB2   #128, FAB+4                                       ;
                                 11            56   E9 000F7        BLBC    STATUS, 4$                                        ;
                                 14            5A   E8 000FA        BLBS    R10, 5$                                           ; 0925
                                        FEA8   CD   9F 000FD        PUSHAB  FAB                                               ; 0928
                     00000000G   00            01   FB 00101        CALLS   #1, SYS$CLOSE                                     ;
                                 56            50   D0 00108        MOVL    R0, STATUS                                        ;
                                 03            56   E8 0010B 4$:    BLBS    STATUS, 5$                                        ; 0930
                                            008D   31      0010E        BRW     12$                                               ;
                                 57     FE4B   CD   9A 00111 5$:    MOVZBL  NAM+3, RES_LEN                                    ; 0934
                                 54            01   D0 00116        MOVL    #1, R4                                            ; 0936
      57          00     008C   CE            59   2D 00119        CMPC5   FIRST_LEN, FIRST_FILE, #0, RES_LEN, RESULT        ;
                                        0210   CE      00120                                                                  ;
                                               03   1A 00123        BGTRU   6$                                                ;
                                 54            01   D9 00125        SBWC    #1, R4                                            ;
                                               54   D5 00128 6$:    TSTL    R4                                                ;
                                               72   13 0012A        BEQL    12$                                               ;
                                        008C   CE   95 0012C        TSTB    FIRST_FILE                                        ; 0938
                                               0B   12 00130        BNEQ    7$                                                ;
                                 59            57   D0 00132        MOVL    RES_LEN, FIRST_LEN                                ;
            008C   CE     0210   CE            57   28 00135        MOVC3   RES_LEN, RESULT, FIRST_FILE                       ;
            0210   CE            57            3A   3A 0013D 7$:    LOCC    #58, RES_LEN, RESULT                              ; 0940
                                               02   12 00143        BNEQ    8$                                                ;
                                               51   D4 00145        CLRL    R1                                                ;
                                               51   D6 00147 8$:    INCL    NAM_PTR                                           ;
                                 50     0210   CE   9E 00149        MOVAB   RESULT, R0                                        ; 0941
                                 50            51   C2 0014E        SUBL2   NAM_PTR, R0                                       ;
                     52          50            57   C1 00151        ADDL3   RES_LEN, R0, NAM_LEN                              ;
                          FEF8   CD            52   B0 00155        MOVW    NAM_LEN, FILEDESC                                 ; 0942
                          FEFC   CD            51   D0 0015A        MOVL    NAM_PTR, FILEDESC+4                               ; 0943
                                 50            6E   3C 0015F        MOVZWL  OUTBUFFER, R0                                     ; 0945
                                 0C 00000000'  EF   E9 00162        BLBC    WIDE_FLAG, 9$                                     ; 0944
0084   8F          20            61            52   2C 00169        MOVC5   NAM_LEN, (NAM_PTR), #32, #132, BUFFER[R0]         ; 0945
                                        08 AE40      00170                                                                  ;
                                               08   11 00173        BRB     10$                                               ;
```

```
        13          20          61          52  2C 00175 9$:    MOVC5   NAM_LEN, (NAM_PTR), #32, #19, BUFFER[R0]   ; 0946
                                      08 AE40     0017A                                                          ;
                                    FEF8  CD  9F 0017D 10$:   PUSHAB  FILEDESC                                   ; 0947
                                      04  AE  9F 00181        PUSHAB  OUTBUFFER                                  ;
                                      08  AC  DD 00184        PUSHL   COUNT                                      ;
                        FA78  CF          03  FB 00187        CALLS   #3, PUTBUFFER                              ;
                              03 00000000' EF  E8 0018C        BLBS    CONTROLCFLAG, 11$                          ; 0948
                                        FF51  31 00193        BRW     3$                                         ;
                              50 00010001  8F  D0 00196 11$:   MOVL    #65537, R0                                 ;
                                             04 0019D        RET                                                ;
                                          6E  B5 0019E 12$:   TSTW    OUTBUFFER                                  ; 0952
                                          0C  13 001A0        BEQL    13$                                        ;
                                          5E  DD 001A2        PUSHL   SP                                         ;
                                      26  AB  9F 001A4        PUSHAB  P.ABF                                      ;
                    00000000G  9F         02  FB 001A7        CALLS   #2, a#DS$PRINTF                            ;
                    000182CA   8F         56  D1 001AE 13$:   CMPL    STATUS, #99018                             ; 0954
                                          07  12 001B5        BNEQ    14$                                        ;
                              56 00010001  8F  D0 001B7        MOVL    #65537, STATUS                             ;
                              50          56  D0 001BE 14$:   MOVL    STATUS, R0                                 ; 0957
                                             04 001C1        RET                                                ;

; Routine Size:  450 bytes,    Routine Base:  CODE + 072A
```

```
;    0959  1     %sbttl 'ods directory'
;    0960  1     routine ods_direct (q_filspec, count, ultrix_wcard) =
;    0961  2         begin
;    0962  2
;    0963  2         local
;    0964  2             status,
;    0965  2             fab : $fab_decl,                        ! FAB block to read directory file
;    0966  2             rab : $rab_decl,                        ! RAB block
;    0967  2             buffer : bblock [512],                  ! Buffer to read record into
;    0968  2             t_directory : bblock [8],               ! Buffer for parsed directory
;    0969  2             q_directory : bblock [dsc$c_s_bln],     ! Descriptor of parsed directory
;    0970  2             base_directory : bblock [dsc$c_s_bln],  ! Descriptor of direct                          !
;    0971  2             q_file : bblock [dsc$c_s_bln],          ! Descriptor for file name
;    0972  2             t_file : bblock [512],                  ! Space for file name           [18]
;    0973  2             q_buffer : bblock [dsc$c_s_bln],        ! Descriptor for output buffer
;    0974  2             t_buffer : bblock [132];                ! Buffer to build file list     [18]
;    0975  2
;    0976  2         bind
;    0977  2             base_direct = $ascid ('0,0'),           ! Normal base directory                     !
;    0978  2             q_direct = .q_filspec + dsc$c_s_bln : bblock;   ! Directory descriptor
;    0979  2
;    0980  2         local
;    0981  2             left,
;    0982  2             right,
;    0983  2             p_dot,                                                                              !
;    0984  2             l_dot,                                                                              !
;    0985  2             p_lpart,
;    0986  2             p_rpart,
;    0987  2             l_rpart,
;    0988  2             l_lpart,
;    0989  2             p_comma,
;    0990  2             l_rest,
;    0991  2             start_size :BYTE;                       !                       [22]
;    0992  2
;    0993  2                                                                                         !G:9
;    0994  2         q_file [dsc$w_length] = 512 ;               !                       [18]
;    0995  2         q_file [dsc$a_pointer] = t_file;
;    0996  2
;    0997  2         IF  (.ultrix_flags)<ultrix_v_mode,1>                                ! [22]
;    0998  2             THEN
;    0999  3             begin
;    1000  3             $fao    ($ascid ('!AS!AS!AS!AS'), q_file,q_file,.q_filspec,.q_filspec+8,.q_filspec+16,.q_filspec+24);
;    1001  3             $ds_printf      ($ascic ('!/!/Directory_!AS !/!/'),q_file);             ! [22]          !G:10
;    1002  3             start_size =  .q_file<0,8> ;    ! Save original length of file string [22]
;    1003  3             end_size = q_file = .q_file -.ultrix_wcard;     ! Cut off the wildcarded part       [22]
;    1004  3             END
;    1005  2             ELSE
;    1006  3             BEGIN
;    1007  3         ch$move (8, base_direct, base_directory);   ! Assume normal directory                      !
;    1008  3         p_lpart = .q_direct [dsc$a_pointer] + 1;    ! Point past "["
;    1009  3         l_rest = .q_direct [dsc$w_length] - 2;      ! Exclude brackets
;    1010  3         p_comma = ch$find_ch (.l_rest, .p_lpart, %c',');    ! Find ',' if any
;    1011  3
;    1012  3         if .p_comma neq 0                           ! If there was a comma ("[octal,octal]")
;    1013  3         then
;    1014  4             begin
;    1015  4             p_rpart = .p_comma + 1;
```

```
;    1016  4              l_lpart = .p_comma - .p_lpart;          ! Length of left number
;    1017  4              l_rpart = .l_rest - .l_lpart - 1;       ! Length of right number
;    1018  4              left = scan$numeric (8, .l_lpart, .p_lpart);
;    1019  4              right = scan$numeric (8, .l_rpart, .p_rpart);
;    1020  4              q_directory [dsc$a_pointer] = t_directory;
;    1021  4              q_directory [dsc$w_length] = 8;
;    1022  5              $fao ($ascid ('!3OL!3OL'), q_directory, q_directory, .left, .right)
;    1023  4              end
;    1024  3          else
;    1025  4              begin                                   ! Copy string, without '[]'                    !
;    1026  4              q_directory [dsc$w_length] = .q_direct [dsc$w_length] - 2;                             !
;    1027  4              q_directory [dsc$a_pointer] = .q_direct [dsc$a_pointer] + 1                            !
;    1028  3              end;                                                                                   !
;    1029  3      !
;    1030  3      ! Print out the header
;    1031  3      !
;    1032  3          $ds_printf ($ascic ('!/!/Directory !AS[!AS]!/!/'), .q_filspec, q_directory);       ! Print header  !G:10
;    1033  3      !                                                                                              !
;    1034  3      ! Now, check for subdirectories, and kluge string around                                       !
;    1035  3      !                                                                                              !
;    1036  3          p_dot = ch$find_ch (.l_rest, .p_lpart, %c'.');                                             !
;    1037  3                                                                                                     !
;    1038  3          if not ch$fail (.p_dot)                 ! If there are subdirectories                      !
;    1039  3          then                                                                                       !
;    1040  4              begin                                                                                  !
;    1041  4                                                                                                     !
;    1042  4              local                                                                                  !
;    1043  4                  last_dot;                                                                          !
;    1044  4                                                                                                     !
;    1045  4              while not ch$fail (.p_dot) do       ! Find last dot                                    !
;    1046  5                  begin                                                                              !
;    1047  5                  last_dot = .p_dot;                                                                 !
;    1048  5                  p_dot = ch$find_ch (.l_rest - (.p_dot - .p_lpart) - 1,                             !
;    1049  5                      .p_dot + 1, %c'.')                                                             !
;    1050  4                  end;                                                                               !
;    1051  4                                                                                                     !
;    1052  4              base_directory [dsc$w_length] = .last_dot - .p_lpart;                                  !
;    1053  4              base_directory [dsc$a_pointer] = .p_lpart;                                             !
;    1054  4              q_directory [dsc$w_length] = .l_rest - (.last_dot - .p_lpart) - 1;                     !
;    1055  4              q_directory [dsc$a_pointer] = .last_dot + 1                                            !
;    1056  3              end;                                                                                   !
;    1057  3
;    1058  3      !
;    1059  3      ! Now get the filename to look up
;    1060  3      !
; P  1061  3          $fao ($ascid ('!AS[!AS]!AS.DIR;0'),     ! Create file to look up (directory)               !
;    1062  3              q_file, q_file, .q_filspec, base_directory, q_directory);                               !
;    1063  3
;    1064  2              END;                                                          ! [22]
;    1065  2
;    1066  2          $fab_init (fab = fab, dnm = '.dir', fna = .q_file [dsc$a_pointer], fns = .q_file [dsc$w_length], fac = get);
;    1067  2          $rab_init (rab = rab, fab = fab, ubf = buffer, usz = 512);
;    1068  2
;    1069  3          if not (status = $open (fab = fab))
;    1070  2          then
;    1071  3              return (selectone .status of
;    1072  3                      set
```

```
;    1073  3                    [rms$_fnf] : rms$_dnf;
;    1074  3                    [otherwise] : .status
;    1075  2                    tes);
;    1076  2          IF  (.ultrix_flags)<ultrix_v_mode,1>                              ! [22]
;    1077  2              THEN
;    1078  2              IF .end_size   EQL  .start_size
;    1079  2                 Then filespecmatch[dsc$w_length] = 0; ! If file string was not shorten no matching file given
;    1080  3          if not (status = $connect (rab = rab))
;    1081  2          then
;    1082  3              begin
;    1083  3              $close (fab = fab);
;    1084  3              return .status
;    1085  2              end;
;    1086  2
;    1087  2          ch$fill (%c' ', 132, t_buffer);                    !       [18]
;    1088  2          q_buffer [dsc$w_length] = 0;
;    1089  2          q_buffer [dsc$a_pointer] = t_buffer;
;    1090  2
;    1091  2          while (status = $get (rab = rab)) do
;    1092  3              Begin
;    1093  3
;    1094  3          IF  (.ultrix_flags)<ultrix_v_mode,1>                              ! [22]
;    1095  3              THEN
;    1096  3                      format_ULTRIX (.rab [rab$w_rsz], .rab [rab$l_rbf], q_buffer, .count) ! [22]
;    1097  3              ELSE
;    1098  4              BEGIN
;    1099  4              case .fab [fab$b_rfm] from fab$c_fix to fab$c_var of
;    1100  4                  set
;    1101  4
;    1102  4                  [fab$c_fix] :
;    1103  4                      format_ods1 (.rab [rab$w_rsz], .rab [rab$l_rbf], q_buffer, .count);
;    1104  4
;    1105  4                  [fab$c_var] :
;    1106  4                      format_ods2 (.rab [rab$w_rsz], .rab [rab$l_rbf], q_buffer, .count);
;    1107  4
;    1108  4                  [outrange] :
;    1109  5                      begin
;    1110  5                      $close (fab = fab);
;    1111  5                      return ss$_filestruct           ! Bad file structure level
;    1112  4                      end;
;    1113  4                  tes;
;    1114  4
;    1115  3                  END;                                                  ! [22]
;    1116  3              If .ControlCFlag Then Exitloop          ! Close file and leave on ^C                    [08]
;    1117  3
;    1118  2              End;
;    1119  2
;    1120  2          if .q_buffer [dsc$w_length] neq 0 then $ds_printf ($ascic ('!AS!/'), q_buffer);
;    1121  2
;    1122  2          $close (fab = fab);
;    1123  3          (selectone .status of
;    1124  3              set
;    1125  3              [rms$_eof] : rms$_normal;
;    1126  3              [otherwise] : .status
;    1127  3              tes)
;    1128  1          end;
```

```
                                                                                  .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                                  30  2C  30  001BA P.ABH:  .ASCII  \0,0\                                  ;
                                                              001BD         .BLKB   3
                                                    00000003  001C0 P.ABG:  .LONG   3                                      ;
                                                    00000000' 001C4         .ADDRESS P.ABH                                 ;
                53  41  21  53  41  21  53  41  21  53  41  21  001C8 P.ABJ:  .ASCII  \!AS!AS!AS!AS\                         ;
                                                    0000000C  001D4 P.ABI:  .LONG   12                                     ;
                                                    00000000' 001D8         .ADDRESS P.ABJ                                 ;
5F  79  72  6F  74  63  65  72  69  44  2F  21  2F  21  16  001DC P.ABK:  .ASCII  <22>\!/!/Directory_!AS !/!/\                ;
                            2F  21  2F  21  20  53  41  21  001EB                                                          ;
                            4C  4F  33  21  4C  4F  33  21  001F3 P.ABM:  .ASCII  \!3OL!3OL\                              ;
                                                              001FB         .BLKB   1
                                                    00000008  001FC P.ABL:  .LONG   8                                      ;
                                                    00000000' 00200         .ADDRESS P.ABM                                 ;
20  79  72  6F  74  63  65  72  69  44  2F  21  2F  21  1A  00204 P.ABN:  .ASCII  <26>\!/!/Directory !AS[!AS]!/!/\            ;
            2F  21  2F  21  5D  53  41  21  5B  53  41  21  00213                                                          ;
52  49  44  2E  53  41  21  5D  53  41  21  5B  53  41  21  0021F P.ABP:  .ASCII  \!AS[!AS]!AS.DIR;0\                      ;
                                                      30  3B  0022E                                                          ;
                                                    00000011  00230 P.ABO:  .LONG   17                                     ;
                                                    00000000' 00234         .ADDRESS P.ABP                                 ;
                                                  72  69  64  2E  00238 P.ABQ:  .ASCII  \.dir\                                 ;
                                          2F  21  53  41  21  05  0023C P.ABR:  .ASCII  <5>\!AS!/\                              ;

                                                                  BASE_DIRECT=        P.ABG
                                                                          .EXTRN  SYS$CONNECT, SYS$GET

                                                                          .PSECT  CODE,NOWRT,  SHR,2

                                                      0FFC 00000 ODS_DIRECT:
                                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11       ; 0960
                                  5E      FABC   CE  9E 00002           MOVAB   -1348(SP), SP                            ;
                                  56        04   AC  D0 00007           MOVL    Q_FILSPEC, R6                            ; 0978
                                  57        08   A6  9E 0000B           MOVAB   8(R6), R7                                ;
                          0290    CE      0200   8F  B0 0000F           MOVW    #512, Q_FILE                             ; 0994
                          0294    CE      0090   CE  9E 00016           MOVAB   T_FILE, Q_FILE+4                         ; 0995
                                  4C 00000000G   EF  E9 0001D           BLBC    ULTRIX_FLAGS, 1$                         ; 0997
                                            18   A6  9F 00024           PUSHAB  24(R6)                                   ; 1000
                                            10   A6  9F 00027           PUSHAB  16(R6)                                   ;
                                            08   A6  9F 0002A           PUSHAB  8(R6)                                    ;
                                                 56  DD 0002D           PUSHL   R6                                       ;
                                          02A0   CE  9F 0002F           PUSHAB  Q_FILE                                   ;
                                          02A4   CE  9F 00033           PUSHAB  Q_FILE                                   ;
                                     00000000'   EF  9F 00037           PUSHAB  P.ABI                                    ;
                      00000000G   00             07  FB 0003D           CALLS   #7, SYS$FAO                              ;
                                          0290   CE  9F 00044           PUSHAB  Q_FILE                                   ; 1001
                                     00000000'   EF  9F 00048           PUSHAB  P.ABK                                    ;
                      00000000G   9F             02  FB 0004E           CALLS   #2, a#DS$PRINTF                          ;
                                  6E      0290   CE  90 00055           MOVB    Q_FILE, START_SIZE                       ; 1002
                  50      0290    CE        0C   AC  C3 0005A           SUBL3   ULTRIX_WCARD, Q_FILE, R0                 ; 1003
                          0290    CE             50  D0 00061           MOVL    R0, Q_FILE                               ;
                      00000000'   EF             50  90 00066           MOVB    R0, END_SIZE                             ;
                                                00FA  31 0006D           BRW     9$                                       ;
          0298    CE 00000000'   EF             08  28 00070 1$:       MOVC3   #8, BASE_DIRECT, BASE_DIRECTORY          ; 1007
                  53        04   A7             01  C1 0007A           ADDL3   #1, 4(R7), R3                            ; 1008
```

```
                        52          67  3C 0007F        MOVZWL  (R7), R2                                ; 1009
                        52          02  C2 00082        SUBL2   #2, R2                                  ;
                        57          52  7D 00085        MOVQ    R2, L_REST                              ;
            68          57          2C  3A 00088        LOCC    #44, L_REST, (P_LPART)                  ; 1010
                                    02  12 0008C        BNEQ    2$                                      ;
                                    51  D4 0008E        CLRL    R1                                      ;
                                    51  D5 00090 2$:    TSTL    P_COMMA                                 ; 1012
                                    55  13 00092        BEQL    3$                                      ;
                        5A      01  A1  9E 00094        MOVAB   1(R1), P_RPART                          ; 1015
            50          51          58  C3 00098        SUBL3   P_LPART, P_COMMA, L_LPART               ; 1016
            54          57          50  C3 0009C        SUBL3   L_LPART, L_REST, R4                     ; 1017
                        59      FF  A4  9E 000A0        MOVAB   -1(R4), L_RPART                         ;
                        55          58  D0 000A4        MOVL    P_LPART, R5                             ; 1018
                        54          50  D0 000A7        MOVL    L_LPART, R4                             ;
                        53          08  D0 000AA        MOVL    #8, R3                                  ;
                        00000000G   EF  16 000AD        JSB     SCAN$NUMERIC                            ;
                        5B          50  D0 000B3        MOVL    R0, LEFT                                ;
                        54          59  7D 000B6        MOVQ    L_RPART, R4                             ; 1019
                        53          08  D0 000B9        MOVL    #8, R3                                  ;
                        00000000G   EF  16 000BC        JSB     SCAN$NUMERIC                            ;
                FD60    CD      FD64  CD  9E 000C2      MOVAB   T_DIRECTORY, Q_DIRECTORY+4              ; 1020
                02A0    CE            08  B0 000C9      MOVW    #8, Q_DIRECTORY                         ; 1021
                                    50  DD 000CE        PUSHL   RIGHT                                   ; 1022
                                    5B  DD 000D0        PUSHL   LEFT                                    ;
                            FD5C    CD  9F 000D2        PUSHAB  Q_DIRECTORY                             ;
                            FD5C    CD  9F 000D6        PUSHAB  Q_DIRECTORY                             ;
                        00000000'   EF  9F 000DA        PUSHAB  P.ABL                                   ;
            00000000G   00          05  FB 000E0        CALLS   #5, SYS$FAO                             ;
                                    0A  11 000E7        BRB     4$                                      ;
                02A0    CE          52  B0 000E9 3$:    MOVW    R2, Q_DIRECTORY                         ; 1026
                FD60    CD          53  D0 000EE        MOVL    R3, Q_DIRECTORY+4                       ; 1027
                            02A0    CE  9F 000F3 4$:    PUSHAB  Q_DIRECTORY                             ; 1032
                                    56  DD 000F7        PUSHL   R6                                      ;
                        00000000'   EF  9F 000F9        PUSHAB  P.ABN                                   ;
            00000000G   9F          03  FB 000FF        CALLS   #3, @#DS$PRINTF                         ;
            68          57          2E  3A 00106        LOCC    #46, L_REST, (P_LPART)                  ; 1036
                                    02  12 0010A        BNEQ    5$                                      ;
                                    51  D4 0010C        CLRL    R1                                      ;
                                    51  D5 0010E 5$:    TSTL    P_DOT                                   ; 1038
                                    39  13 00110        BEQL    8$                                      ;
                                    51  D5 00112 6$:    TSTL    P_DOT                                   ; 1045
                                    17  13 00114        BEQL    7$                                      ;
                        52          51  D0 00116        MOVL    P_DOT, LAST_DOT                         ; 1047
            55          58          51  C3 00119        SUBL3   P_DOT, P_LPART, R5                      ; 1048
                        50      FF A547  9E 0011D        MOVAB   -1(R5)[L_REST], R0                      ;
        01  A1          50          2E  3A 00122        LOCC    #46, R0, 1(P_DOT)                       ;
                                    E9  12 00127        BNEQ    6$                                      ;
                                    51  D4 00129        CLRL    R1                                      ;
                                    E5  11 0012B        BRB     6$                                      ;
            50          52          58  C3 0012D 7$:    SUBL3   P_LPART, LAST_DOT, R0                   ; 1052
                0298    CE          50  B0 00131        MOVW    R0, BASE_DIRECTORY                      ;
                029C    CE          58  D0 00136        MOVL    P_LPART, BASE_DIRECTORY+4               ; 1053
            54          57          50  C3 0013B        SUBL3   R0, L_REST, R4                          ; 1054
    02A0    CE          54          01  A3 0013F        SUBW3   #1, R4, Q_DIRECTORY                     ;
                FD60    CD      01  A2  9E 00145        MOVAB   1(R2), Q_DIRECTORY+4                    ; 1055
                            02A0    CE  9F 0014B 8$:    PUSHAB  Q_DIRECTORY                             ; 1062
                            029C    CE  9F 0014F        PUSHAB  BASE_DIRECTORY                          ;
```

```
                                                  56  DD 00153          PUSHL   R6                                          ;
                                            029C  CE  9F 00155          PUSHAB  Q_FILE                                      ;
                                            02A0  CE  9F 00159          PUSHAB  Q_FILE                                      ;
                                        00000000'  EF  9F 0015D          PUSHAB  P.ABO                                       ;
                        00000000G  00             06  FB 00163          CALLS   #6, SYS$FAO                                 ;
  0050   8F          00            6E             00  2C 0016A 9$:      MOVC5   #0, (SP), #0, #80, $RMS_PTR                 ; 1066
                                              B0  AD     00171                                                              ;
                              B0  AD       5003  8F  B0 00173          MOVW    #20483, $RMS_PTR                            ;
                              C6  AD             02  90 00179          MOVB    #2, $RMS_PTR+22                             ;
                              CF  AD             02  90 0017D          MOVB    #2, $RMS_PTR+31                             ;
                              DC  AD       0294  CE  D0 00181          MOVL    Q_FILE+4, $RMS_PTR+44                       ;
                              E0  AD 00000000'  EF  9E 00187          MOVAB   P.ABQ, $RMS_PTR+48                          ;
                              E4  AD       0290  CE  90 0018F          MOVB    Q_FILE, $RMS_PTR+52                         ;
                              E5  AD             04  90 00195          MOVB    #4, $RMS_PTR+53                             ;
  0044   8F          00            6E             00  2C 00199          MOVC5   #0, (SP), #0, #68, $RMS_PTR                 ; 1067
                                            FF6C  CD     001A0                                                              ;
                            FF6C  CD       4401  8F  B0 001A3          MOVW    #17409, $RMS_PTR                            ;
                              8C  AD       0200  8F  B0 001AA          MOVW    #512, $RMS_PTR+32                           ;
                              90  AD       FD6C  CD  9E 001B0          MOVAB   BUFFER, $RMS_PTR+36                         ;
                              A8  AD         B0  AD  9E 001B6          MOVAB   FAB, $RMS_PTR+60                            ;
                                              B0  AD  9F 001BB          PUSHAB  FAB                                         ; 1069
                        00000000G  00             01  FB 001BE          CALLS   #1, SYS$OPEN                                ;
                                   56             50  D0 001C5          MOVL    R0, STATUS                                  ;
                                   11             56  E8 001C8          BLBS    STATUS, 10$                                 ;
                        00018292   8F             56  D1 001CB          CMPL    STATUS, #98962                              ; 1073
                                                  39  12 001D2          BNEQ    12$                                         ;
                                   50 0001C04A    8F  D0 001D4          MOVL    #114762, R0                                 ;
                                                      04 001DB          RET                                                 ;
                                   0F 00000000G   EF  E9 001DC 10$:     BLBC    ULTRIX_FLAGS, 11$                           ; 1076
                                   6E 00000000'   EF  91 001E3          CMPB    END_SIZE, START_SIZE                        ; 1078
                                                  06  12 001EA          BNEQ    11$                                         ;
                                      00000000'   EF  B4 001EC          CLRW    FILESPECMATCH                               ; 1079
                                            FF6C  CD  9F 001F2 11$:     PUSHAB  RAB                                         ; 1080
                        00000000G  00             01  FB 001F6          CALLS   #1, SYS$CONNECT                             ;
                                   56             50  D0 001FD          MOVL    R0, STATUS                                  ;
                                   0D             56  E8 00200          BLBS    STATUS, 13$                                 ;
                                              B0  AD  9F 00203          PUSHAB  FAB                                         ; 1083
                        00000000G  00             01  FB 00206          CALLS   #1, SYS$CLOSE                               ;
                                                00BA  31 0020D 12$:     BRW     22$                                         ; 1084
  0084   8F          20            6E             00  2C 0021C 13$:     MOVC5   #0, (SP), #32, #132, T_BUFFER               ; 1087
                                              04  AE     00217                                                              ;
                                            0088  CE  B4 00219          CLRW    Q_BUFFER                                    ; 1088
                            008C  CE         04  AE  9E 0021D          MOVAB   T_BUFFER, Q_BUFFER+4                        ; 1089
                                            FF6C  CD  9F 00223 14$:     PUSHAB  RAB                                         ; 1091
                        00000000G  00             01  FB 00227          CALLS   #1, SYS$GET                                 ;
                                   56             50  D0 0022E          MOVL    R0, STATUS                                  ;
                                   64             56  E9 00231          BLBC    STATUS, 20$                                 ;
                                   15 00000000G   EF  E9 00234          BLBC    ULTRIX_FLAGS, 15$                           ; 1094
                                              08  AC  DD 0023B          PUSHL   COUNT                                       ; 1096
                                            008C  CE  9F 0023E          PUSHAB  Q_BUFFER                                    ;
                                              94  AD  DD 00242          PUSHL   RAB+40                                      ;
                                   7E         8E  AD  3C 00245          MOVZWL  RAB+34, -(SP)                               ;
                            F8BF  CF             04  FB 00249          CALLS   #4, FORMAT_ULTRIX                           ;
                                                  41  11 0024E          BRB     19$                                         ;
                      01          01         CF  AD  8F 00250 15$:     CASEB   FAB+31, #1, #1                              ; 1099
                                0029           0014     00255 16$:     .WORD   17$-16$,-                                   ;
                                                                                18$-16$                                     ;
```

```
                                  B0   AD  9F 00259          PUSHAB  FAB                                  ; 1110
          00000000G  00              01  FB 0025C          CALLS   #1, SYS$CLOSE                        ;
                     50       08C0   8F  3C 00263          MOVZWL  #2240, R0                            ; 1111
                                         04 00268          RET                                          ;
                                  08   AC  DD 00269 17$:    PUSHL   COUNT                                ; 1103
                                008C   CE  9F 0026C          PUSHAB  Q_BUFFER                             ;
                                  94   AD  DD 00270          PUSHL   RAB+40                               ;
                     7E           8E   AD  3C 00273          MOVZWL  RAB+34, -(SP)                        ;
              F93A   CF              04  FB 00277          CALLS   #4, FORMAT_ODS1                      ;
                                     13  11 0027C          BRB     19$                                  ;
                                  08   AC  DD 0027E 18$:    PUSHL   COUNT                                ; 1106
                                008C   CE  9F 00281          PUSHAB  Q_BUFFER                             ;
                                  94   AD  DD 00285          PUSHL   RAB+40                               ;
                     7E           8E   AD  3C 00288          MOVZWL  RAB+34, -(SP)                        ;
              F9C5   CF              04  FB 0028C          CALLS   #4, FORMAT_ODS2                      ;
                     8B 00000000'    EF  E9 00291 19$:     BLBC    CONTROLCFLAG, 14$                    ; 1116
                                0088   CE  B5 00298 20$:    TSTW    Q_BUFFER                             ; 1120
                                     11  13 0029C          BEQL    21$                                  ;
                                0088   CE  9F 0029E          PUSHAB  Q_BUFFER                             ;
                           00000000'   EF  9F 002A2          PUSHAB  P.ABR                                ;
          00000000G  9F              02  FB 002A8          CALLS   #2, @#DS$PRINTF                      ;
                                  B0   AD  9F 002AF 21$:    PUSHAB  FAB                                  ; 1122
          00000000G  00              01  FB 002B2          CALLS   #1, SYS$CLOSE                        ;
          0001827A   8F              56  D1 002B9          CMPL    STATUS, #98938                       ; 1125
                                     08  12 002C0          BNEQ    22$                                  ;
                     50 00010001     8F  D0 002C2          MOVL    #65537, R0                           ;
                                         04 002C9          RET                                          ;
                     50              56  D0 002CA 22$:     MOVL    STATUS, R0                           ; 1126
                                         04 002CD          RET                                          ; 1128

; Routine Size:  718 bytes,    Routine Base:  CODE + 08EC
```

```
;   1130  1     %sbttl 'Main code'
;   1131  1
;   1132  1     global routine dsx$directory (cli_block) : call_cli novalue =
;   1133  1
;   1134  1     !++
;   1135  1     !
;   1136  1     ! Functional description:
;   1137  1     !       This code just lists the file names and versions in a directory
;   1138  1     !
;   1139  1     ! Inputs:
;   1140  1     !       cli_block               pointer to cli block
;   1141  1     !
;   1142  1     ! Implicit inputs:
;   1143  1     !       none
;   1144  1     !
;   1145  1     ! Outputs:
;   1146  1     !       none
;   1147  1     !
;   1148  1     ! Implicit outputs:
;   1149  1     !       types directory of device
;   1150  1     !
;   1151  1     ! Side effects:
;   1152  1     !       accesses device via RMS
;   1153  1     !
;   1154  1     ! Return status:
;   1155  1     !       none
;   1156  1     !--
;   1157  1
;   1158  2         begin
;   1159  2
;   1160  2         Routine InterceptControlC =
;   1161  3             Begin
;   1162  4             (ControlCFlag = 1)      ! Just set the flag
;   1163  2             End;



                                        0000 00000 INTERCEPTCONTROLC:
                                                        .WORD   Save nothing                                    ; 1160
                00000000'  EF           01  90 00002    MOVB    #1, CONTROLCFLAG                                ; 1162
                           50           01  D0 00009    MOVL    #1, R0                                          ; 1163
                                            04 0000C    RET                                                     ;

; Routine Size:  13 bytes,    Routine Base:  CODE + 0BBA


;   1164  2
;   1165  2         External Routine                                                                    !G:2
;   1166  2             PPA$DIRECTORY : Addressing_Mode (Long_Relative) WEAK,   !                [25]       !G:6
;   1167  2             sys$getdvi : addressing_mode (general); !                                [26] gets information about
;   1168  2                                                                                             !G:6
;   1169  2                                                                                             !G:11
;   1170  2         local                                                                               !G:11
;   1171  2             count,
;   1172  2             length,
```

```
;   2          pointer,
;   2          status,                                          !                      [05]          !G:6
;   2          ppa_stat,                                        !                      [25]          !G:11
;   2          ultrix_wcard :BYTE,                              ! Length of filename if wildcards present [22] !G:6
;   2          FileMatch : Vector [Nam$C_MaxRss, Byte],         !                      [10]          !G:6
;   2          q_fileblock : bblock [dsc$c_s_bln*5];            ! File part descriptors                !G:6
;   2
;   2                                                                                                  !G:2
;   2      map
;   2          cli_block : ref bblock;
;   2
;   2      Bind                          !                                                                [10]
;   2          WildCardMask = $Ascid ('*.*;*') : Block [, Byte];      !                                  [10]
;   2
;   2      ControlCFlag = 0;                          ! Initially clear the flag                          [08]
;   2      $Ds_CntrlC (AstAdr = InterceptControlC);   ! Set Control C handler
;   2
;   2      IF  (.ultrix_flags)<ultrix_v_mode,1>                                  ! [22]
;   2          THEN
;   3          BEGIN
;   3      CH$FILL(0,dsc$c_s_bln*5, Q_FILEBLOCK);                   ! Clear out                [22]
;   3      breakup_file (def$c_ultrix_len, def$q_ultrix, q_fileblock); ! Merge last-chance spec
;   3      breakup_file (.def$q_dev [dsc$w_length], .def$q_dev [dsc$a_pointer], q_fileblock);
;   3                                                 ! Merge default device
;   3      breakup_file (.def$q_dir [dsc$w_length], .def$q_dir [dsc$a_pointer], q_fileblock);
;   3                                                 ! Merge default directory
;   3          END                                                               ! [22]
;   2          ELSE
;   3          BEGIN
;   3      breakup_file (def$c_bkstp_len, def$q_backstop, q_fileblock);        ! Merge last-chance spec
;   3      breakup_file (.def$q_dev [dsc$w_length], .def$q_dev [dsc$a_pointer], q_fileblock);
;   3                                                 ! Merge default device
;   3      breakup_file (.def$q_dir [dsc$w_length], .def$q_dir [dsc$a_pointer], q_fileblock);
;   3                                                 ! Merge default directory
;   3      BreakUp_File (.WildCardMask [Dsc$W_Length], .WildCardMask [Dsc$A_Pointer], Q_FileBlock);    !    [10]
;   2          END;                                                              ! [22]
;   2      breakup_file (.cli_block [cli$w_file_len], .cli_block [cli$l_file_adr], q_fileblock);
;   2                                                 ! Merge operator file spec
;   2      FileSpecMatch [Dsc$W_Length] = 0;          ! Initialize descriptor                             [10]
;   2      FileSpecMatch [Dsc$A_Pointer] = FileMatch;  !                                                  [10]
;   2
;   2      Incr Index From 2 to 4 Do                  ! Merge the final spec                              [10]
;   3          Begin                                  !                                                   [10]
;   3
;   3          Bind                                   !                                                   [10]
;   3              Desc = Q_FileBlock + .Index*8 : Block [, Byte];     !                                  [10]
;   3
;   3          If .Desc [Dsc$W_Length] + .FileSpecMatch [Dsc$W_Length] Gtr Nam$C_MaxRss        !          [10]
;   3          Then                                   !                                                   [10]
;   3              Exitloop;                          !                                                   [10]
;   3
;   3          If .Desc [Dsc$W_Length] Neq 0          !                                                   [10]
;   3           And .Desc [Dsc$A_Pointer] Neq 0       !                                                   [10]
;   3          Then                                   !                                                   [10]
;   4              Begin                              !                                                   [10]
;   4
;   4              Bind                               !                                                   [10]
```

```
;    1230  4                    Destination = .FileSpecMatch [Dsc$W_Length] + .FileSpecMatch [Dsc$A_Pointer];   !                [10]
;    1231  4
;    1232  4                Ch$Move (                                   !                                                    [10]
;    1233  4                    .Desc [Dsc$W_Length],                   !                                                    [10]
;    1234  4                    .Desc [Dsc$A_Pointer],                  !                                                    [10]
;    1235  4                    Destination);                           !                                                    [10]
;    1236  4
;    1237  4                If .Index Eql 4                             ! Make sure Version starts with ';', not '.'         [10]
;    1238  4            -   Then                                        !                                                    [10]
;    1239  4                    Ch$WChar (%C';', Destination);          !                                                    [10]
;    1240  4
;    1241  4                FileSpecMatch [Dsc$W_Length] = .FileSpecMatch [Dsc$W_Length] + .Desc [Dsc$W_Length];        !    [10]
;    1242  4                End                                         !                                                    [10]
;    1243  4
;    1244  2            End;
;    1245  2        If (.ultrix_flags)<ultrix_v_mode,1>                 ! If ultrix get the file name only                   [22]
;    1246  2            THEN
;    1247  3            BEGIN
;    1248  3            LOCAL
;    1249  3              p_nam;
;    1250  3            p_nam = .FileSpecMatch [Dsc$A_Pointer]+.FileSpecMatch [Dsc$W_Length] -1; ! Point to end of string   [22]
;    1251  3            ultrix_wcard = 0;                               ! Initialize length                                  [22]
;    1252  3            WHILE   (.p_nam GTR .FileSpecMatch [Dsc$A_Pointer]) AND (.(.p_nam)<0,8> NEQ %C')' ) DO
;    1253  4              BEGIN
;    1254  4              IF    .(.p_nam)<0,8> EQL %C'/'! Search for last '/'                                               [22]
;    1255  4                THEN
;    1256  5                    BEGIN
;    1257  5                      FileSpecMatch [Dsc$W_Length] = .FileSpecMatch [Dsc$A_Pointer]+.FileSpecMatch [Dsc$W_Length] -1 -.p
;    1258  5                      CH$MOVE(.FileSpecMatch [Dsc$W_Length],.p_nam+1,.FileSpecMatch [Dsc$A_Pointer]); ! Move filename [2
;    1259  5                      EXITLOOP
;    1260  5                    END
;    1261  4                ELSE
;    1262  5                    BEGIN
;    1263  5                    IF .(.p_nam)<0,8> EQL %C'*' OR .(.p_nam)<0,8> EQL %C'%' THEN ultrix_wcard = 1; ! Check for wardcard
;    1264  5                   p_nam = .p_nam -1;                       ! Check previous character                           [22]
;    1265  4                    END;
;    1266  3              END;
;    1267  3              IF  .ultrix_wcard  THEN ultrix_wcard = .filespecmatch[dsc$w_length]+1; ! Set to length of wildcarded name
;    1268  2            END;
;    1269  2
;    1270  2        If Ch$Eql (                                         ! Don't bother matching if it's all wildcarded       [10]
;    1271  2            .FileSpecMatch [Dsc$W_Length],                  !                                                    [10]
;    1272  2            .FileSpecMatch [Dsc$A_Pointer],                 !                                                    [10]
;    1273  2            .WildCardMask [Dsc$W_Length],                   !                                                    [10]
;    1274  2            .WildCardMask [Dsc$A_Pointer],                  !                                                    [10]
;    1275  2            0)                                              !  (0 fill)                                          [10]
;    1276  2        Then                                                !                                                    [10]
;    1277  2            FileSpecMatch [Dsc$W_Length] = 0;               !                                                    [10]
;    1278  2
;    1279  3        begin                                                                                                    !G:11
;    1280  3
;    1281  3        global register
;    1282  3            zero = 0,
;    1283  3            one = 1;
;    1284  3                                                                                                                !G:11
;    1285  3        global                                                                                                   !G:12
;    1286  3            dir_flags : byte;                               ! Flags for directory commands                [12]   !G:12
```

```
;    1287  3                                                                                                  !G:12
;    1288  3          linkage
;    1289  3              jsb_device = jsb (register = 4, register = 5) : global (zero = 0, one = 1)
;    1290  3              nopreserve (2, 3);
;    1291  3                                                                                                  !G:11
;    1292  3                                                                                                  !G:12
;    1293  3          global literal                                                                          !G:12
;    1294  3              dir_v_cmm    = 0,                    ! Flag for directory command in progress      [12]  !G:12
;    1295  3              dir_v_drvclr = 1;                    ! Flag for drive clear allowed. Start of command.[12] !G:12
;    1296  3                                                                                                  !G:12
;    1297  3          literal                                                                                 !G:11
;    1298  3              max_devname_len = 64;                ! size of dev name returned by getdvi         [26]  !G:6
;    1299  3                                                                                                  !G:6
;    1300  3          local                                                                                   !G:11
;    1301  3              console : initial (uplit (%ascii 'CSA')),! console string                          [26]  !G:6
;    1302  3              ptable : ref bblock field ($ds_hpo_f),  ! ptable of target dev                      !G:6
;    1303  3              dollar,                              ! pointer to the "$" in the device name       [26]  !G:6
;    1304  3              len,                                                                                !G:6
;    1305  3              status,                              ! return status                               [26]  !G:6
;    1306  3              dev_class,                           ! i.e. tape or disk                           [26]  !G:6
;    1307  3              class_len,                           ! length of class type                        [26]  !G:6
;    1308  3              dev_len,                             ! length of dev name                          [26]  !G:6
;    1309  3              dev_name : block [max_devname_len,byte], ! name of target dev                      [26]  !G:6
;    1310  3              itmlst : block [28,byte] initial (   ! input to sys$getdvi                         [26]  !G:6
;    1311  3                      word (4),word (DVI$_DEVCLASS),  !                                                  !G:11
;    1312  3                      long (dev_class),               !                                                  !G:11
;    1313  3                      long (class_len),               !                                                  !G:11
;    1314  3                      word (64),word (dvi$_devnam),   !                                                  !G:11
;    1315  3                      long (dev_name),                !                                                  !G:11
;    1316  3                      long (dev_len),                 !                                                  !G:11
;    1317  3                      long (0));                      !                                                  !G:11
;    1318  3                                                                                                  !G:11
;    1319  3          count = 0;                                                  ! Initialize file count !G:6
;    1320  3          If not .Dsa$V_User                                      ! if stand alone          [26] !G:9
;    1321  4          then BEGIN                                                                              !G:9
;    1322  4                  dir_flags<dir_v_cmm,1> = 1;     ! Set for directory command              [12]  !G:12
;    1323  4                  dir_flags<dir_v_drvclr,1> =1;   ! Set for drive clear first time in driver [12]  !G:12
;    1324  4                  if ch$eql(3,.q_fileblock [dsc$a_pointer],3,.console) then          ! if dev is the console [26]
;    1325  5                     (selectone .DSA$GL_SID[PR$V_SID_TYPE] of          ! what kind of system is this?  begin [27] !
;    1326  5                        set                                                                       !G:9
;    1327  5                          [pr$_sid_TYP780,                                ! VAX 11/780            !G:9
;    1328  5                           pr$_sid_TYP750,                                ! VAX 11/750            !G:9
;    1329  5                           pr$_sid_TYP730,                                ! VAX 11/730            !G:9
;    1330  5                           pr$_sid_TYP790]:                               ! VAX 8600              !G:9
;    1331  5                                status = RT11_DIRECT (q_fileblock, count);                         !G:9
;    1332  5                          [pr$_sid_TYP8SS,                                ! VAX SCORPIO           !G:10
;    1333  5                           pr$_sid_TYP8NN,                                ! VAX NAUTILUS          !G:10
;    1334  5                           pr$_sid_TYP8RR]:                               ! VAX RRR [28]          !G:10
;    1335  5                                status = ODS_DIRECT (q_fileblock, count, .ultrix_wcard);           !G:9
;    1336  5                          [pr$_sid_TYPUV]:                               ! microVAX chip         !G:9
;    1337  6                                begin                                                             !G:9
;    1338  6                                    if .DS$GB_SYSTYPE eql PR$_SYS_TYP900 ! ANDROMEDA              !G:9
;    1339  7                                     then begin                                                   !G:9
;    1340  7                                            ppa_stat = PPA$DIRECTORY (q_fileblock);              !G:9
;    1341  7                                            return .ppa_stat                                     !G:9
;    1342  7                                          end                                                    !G:9
;    1343  6                                    else return 0;                                               !G:9
```

```
;   1344  5                                 end;                                                                    !G:9
;   1345  5                             [otherwise]:                                                                !G:11
;   1346  5                                 status = ODS_DIRECT (q_fileblock, count, .ultrix_wcard);                !G:9
;   1347  5                         tes)                                                                            !G:9
;   1348  4                     else                        ! STANDALONE, NON CONSOLE DEVICE              end [27]   !G:9
;   1349  5                         begin                                                                           !G:9
;   1350  5                             len = .q_fileblock [dsc$w_length] - 1;                                      !G:6
;   1351  5                             loc$ptable (.len, .q_fileblock [dsc$a_pointer], -1; ptable);    ! [07]       !G:6
;   1352  5                             if .ptable eql 0                                                ! is device attached? !G:9
;   1353  6                             then $ds_printf ($ascic('!/%DS-F-DEVNATT, Device not attached!/'))          !G:9
;   1354  5                             else                                                                        !G:11
;   1355  6                                 (selectone .(ptable [hp$t_type] + 1)<0, 16> of          ! determine device class !G:
;   1356  6                                         set                                                             !G:11
;   1357  6                                            [%ascii'RP', %ascii'RM', %ascii'RK', %ascii'DI',%ascii'RC',          ! [1
;   1358  6                                             %ascii'R8', %Ascii'RA', %ascii'RL', %Ascii'RX', %Ascii'RD'] :       ! [2
;   1359  6                                                 status = ods_direct (q_fileblock, count, .ultrix_wcard);        ! [2
;   1360  6                                            [%ascii'TA',%ascii'TE',%Ascii 'TK', %Ascii'TU', %ascii'TS'] : !G:11
;   1361  6                                                 status = ansi_direct (q_fileblock, count);                      ! [1
;   1362  6                                            [otherwise] : status = rms$_dev;                                !G:6
;   1363  5                                         tes);                                                           !G:6
;   1364  4                         end;                                                                            !G:10
;   1365  4                     if .status<0,1,1>                                                                   !G:11
;   1366  4                     then $ds_printf ($ascic ('!/Total of !ZL files.!/'), .count);                       !G:11
;   1367  4                     dsr$completion(.status);                                                            !G:11
;   1368  4                     dir_flags=0;                                                    ! clear flags   [12]  !G:12
;   1369  4                 END                                                                 ! END if standalone   !G:9
;   1370  3             else                                                                    ! if in user mode     !G:6
;   1371  4                 BEGIN                                                                                     !G:9
;   1372  4                     status = sys$getdvi(0,0,q_fileblock,itmlst,0,0,0,0);            ! get device info  [26] !G:6
;   1373  4                     if .status eql ss$_normal                                       !                  [26] !G:9
;   1374  5                     then begin                                                                            !G:9
;   1375  5                             dollar = ch$find_ch (.dev_len, dev_name, %ascii '$');! skip past first $, ie $node$generic[2
;   1376  6                             if not (.dollar eql 0)                          ! if we found a $ in device name        [26
;   1377  5                             then (itmlst + 16) = .dollar + 1;               ! move pointer to first char after $node$[26
;   1378  5                             if ch$eql(3,.(itmlst+16),3,.console)            ! if dev is the console                 [26
;   1379  6                             then $ds_printf ($ascic('!/%DS-F-USER, can''t access console in user mode.!/')) ![26] !G:9
;   1380  6                             else if (.dev_class eql DC$_DISK) or (.dev_class eql DC$_TAPE)                !G:11
;   1381  5                             then status = vms_dir ()                ! if dev is a disk or tape             [27] !G:9
;   1382  5                             else $ds_printf ($ascic('!/%DS-F-ILDEVCLS, illegal device class for directory!/'));     ![26
;   1383  5                             dsr$completion (.status);               ! abort if error occurred while listing directory[26
;   1384  5                         end                                         ! end if status is normal                 !G:9
;   1385  4                     else dsr$completion (.status);                  ! if status isn't normal call DSR$COMPLETION     [27
;   1386  3                 END;                                                ! END if user mode                        !G:9
;   1387  3         If .ControlCFlag Then Return;                                                   ! No final printout on ^C !G
;   1388  3         $Ds_CntrlC ();                                                                  ! Return ^C control to kerne
;   1389  2         end;                                                                                              !G:6
;   1390  1         end;                                        ! Of dsx$directory


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                2A  3B  2A  2E  2A  00242 P.ABT:  .ASCII  \*.*;*\                                  ;
                                                    00247         .BLKB   1
                                        00000005    00248 P.ABS:  .LONG   5                                         ;
                                        00000000'   0024C         .ADDRESS P.ABT                                    ;
                                    00  41  53  43  00250 P.ABU:  .ASCII  \CSA\<0>                                  ;
```

```
                                                  0004  00254 P.ABV:  .WORD   4                                          ;
                                                  0004  00256         .WORD   4                                          ;
                                              00000000  00258         .LONG   0                                          ;
                                              00000000  0025C         .LONG   0                                          ;
                                                  0040  00260         .WORD   64                                         ;
                                                  0020  00262         .WORD   32                                         ;
                                              00000000  00264         .LONG   0                                          ;
                                              00000000  00268         .LONG   0                                          ;
                                              00000000  0026C         .LONG   0                                          ;
54  41  4E  56  45  44  2D  46  2D  53  44  25  2F  21  26  00270 P.ABW:  .ASCII  \&!/%DS-F-DEVNATT, Device not attached!/\   ;
61  20  74  6F  6E  20  65  63  69  76  65  44  20  2C  54  0027F                                                     ;
                        2F  21  64  65  68  63  61  74  74  0028E                                                     ;
4C  5A  21  20  66  6F  20  6C  61  74  6F  54  2F  21  17  00297 P.ABX:  .ASCII  <23>\!/Total of !ZL files.!/\              ;
                        2F  21  2E  73  65  6C  69  66  20  002A6                                                     ;
20  2C  52  45  53  55  2D  46  2D  53  44  25  2F  21  32  002AF P.ABY:  .ASCII  \2!/%DS-F-USER, can't access console in u\  ;
6F  63  20  73  73  65  63  63  61  20  74  27  6E  61  63  002BE                                                     ;
                    75  20  6E  69  20  65  6C  6F  73  6E  002CD                                                     ;
                2F  21  2E  65  64  6F  6D  20  72  65  73  002D7         .ASCII  \ser mode.!/\                             ;
43  56  45  44  4C  49  2D  46  2D  53  44  25  2F  21  36  002E2 P.ABZ:  .ASCII  \6!/%DS-F-ILDEVCLS, illegal device class \  ;
76  65  64  20  6C  61  67  65  6C  6C  69  20  2C  53  4C  002F1                                                     ;
                    20  73  73  61  6C  63  20  65  63  69  00300                                                     ;
2F  21  79  72  6F  74  63  65  72  69  64  20  72  6F  66  0030A         .ASCII  \for directory!/\                         ;

                                                                          .PSECT  WORK,NOEXE,2

                                                          003BC DIR_FLAGS::
                                                                          .BLKB   1

                                                                  WILDCARDMASK=       P.ABS
                                                                  DIR_V_CMM==         0
                                                                  DIR_V_DRVCLR==      1
                                                                          .EXTRN  SYS$GETDVI, DS$CNTRLC
                                                                          .WEAK   PPA$DIRECTORY

                                                                          .PSECT  CODE,NOWRT,  SHR,2

                                                  0FFC  00000         .ENTRY  DSX$DIRECTORY, Save R2,R3,R4,R5,R6,R7,R8,-  ; 1132
                                                                              R9,R10,R11                                 ;
                                5B        EE   AF  9E  00002         MOVAB   INTERCEPTCONTROLC, R11                     ;
                                5A 00000000'   EF  9E  00006         MOVAB   WILDCARDMASK+4, R10                        ;
                                59 00000000'   EF  9E  0000D         MOVAB   FILESPECMATCH, R9                          ;
                                5E      FE6C   CE  9E  00014         MOVAB   -404(SP), SP                               ;
                                56             52  D0  00019         MOVL    R2, R6                                     ;
                                          08   A9  94  0001C         CLRB    CONTROLCFLAG                               ; 1187
                                               7E  D4  0001F         CLRL    -(SP)                                      ; 1188
                                               5B  DD  00021         PUSHL   R11                                        ;
                        00000000G  9F          02  FB  00023         CALLS   #2, @#DS$CNTRLC                            ;
                                42 00000000G   EF  E9  0002A         BLBC    ULTRIX_FLAGS, 1$                           ; 1190
        28          00          6E             00  2C  00031         MOVC5   #0, (SP), #0, #40, Q_FILEBLOCK             ; 1193
                                          6C   AE      00036                                                           ;
                                          6C   AE  9F  00038         PUSHAB  Q_FILEBLOCK                                ; 1194
                                   00000000G   EF  9F  0003B         PUSHAB  DEF$Q_ULTRIX                               ;
                                   00000000G   8F  DD  00041         PUSHL   #DEF$C_ULTRIX_LEN                          ;
                            0000G  CF          03  FB  00047         CALLS   #3, BREAKUP_FILE                           ;
                                          6C   AE  9F  0004C         PUSHAB  Q_FILEBLOCK                                ; 1195
                                   00000000G   EF  DD  0004F         PUSHL   DEF$Q_DEV+4                                ;
                                7E 00000000G   EF  3C  00055         MOVZWL  DEF$Q_DEV, -(SP)                           ;
```

```
                0000G  CF            03  FB 0005C        CALLS   #3, BREAKUP_FILE                          ;
                                 6C  AE  9F 00061        PUSHAB  Q_FILEBLOCK                               ; 1197
                          00000000G  EF  DD 00064        PUSHL   DEF$Q_DIR+4                               ;
                       7E 00000000G  EF  3C 0006A        MOVZWL  DEF$Q_DIR, -(SP)                          ;
                                     47  11 00071        BRB     2$                                        ;
                                 6C  AE  9F 00073 1$:    PUSHAB  Q_FILEBLOCK                               ; 1202
                          00000000G  EF  9F 00076        PUSHAB  DEF$Q_BACKSTOP                            ;
                          00000000G  8F  DD 0007C        PUSHL   #DEF$C_BKSTP_LEN                          ;
                0000G  CF            03  FB 00082        CALLS   #3, BREAKUP_FILE                          ;
                                 6C  AE  9F 00087        PUSHAB  Q_FILEBLOCK                               ; 1203
                          00000000G  EF  DD 0008A        PUSHL   DEF$Q_DEV+4                               ;
                       7E 00000000G  EF  3C 00090        MOVZWL  DEF$Q_DEV, -(SP)                          ;
                0000G  CF            03  FB 00097        CALLS   #3, BREAKUP_FILE                          ;
                                 6C  AE  9F 0009C        PUSHAB  Q_FILEBLOCK                               ; 1205
                          00000000G  EF  DD 0009F        PUSHL   DEF$Q_DIR+4                               ;
                       7E 00000000G  EF  3C 000A5        MOVZWL  DEF$Q_DIR, -(SP)                          ;
                0000G  CF            03  FB 000AC        CALLS   #3, BREAKUP_FILE                          ;
                                 6C  AE  9F 000B1        PUSHAB  Q_FILEBLOCK                               ; 1207
                                     6A  DD 000B4        PUSHL   WILDCARDMASK+4                            ;
                       7E        FC  AA  3C 000B6        MOVZWL  WILDCARDMASK, -(SP)                       ;
                0000G  CF            03  FB 000BA 2$:    CALLS   #3, BREAKUP_FILE                          ;
                                 6C  AE  9F 000BF        PUSHAB  Q_FILEBLOCK                               ; 1209
                                 0C  A6  DD 000C2        PUSHL   12(CLI_BLOCK)                             ;
                       7E        08  A6  3C 000C5        MOVZWL  8(CLI_BLOCK), -(SP)                       ;
                0000G  CF            03  FB 000C9        CALLS   #3, BREAKUP_FILE                          ;
                                     69  B4 000CE        CLRW    FILESPECMATCH                             ; 1211
                  04   A9      0094  CE  9E 000D0        MOVAB   FILEMATCH, FILESPECMATCH+4                ; 1212
                       57            02  D0 000D6        MOVL    #2, INDEX                                 ; 1214
                       56       6C AE47  7E 000D9 3$:    MOVAQ   Q_FILEBLOCK[INDEX], R6                    ; 1218
                       50            66  3C 000DE        MOVZWL  (R6), R0                                  ; 1220
                       51            69  3C 000E1        MOVZWL  FILESPECMATCH, R1                         ;
                       50            51  C0 000E4        ADDL2   R1, R0                                    ;
            000000FF   8F            50  D1 000E7        CMPL    R0, #255                                  ;
                                     24  14 000EE        BGTR    6$                                        ;
                                     66  B5 000F0        TSTW    (R6)                                      ; 1224
                                     1C  13 000F2        BEQL    5$                                        ;
                                 04  A6  D5 000F4        TSTL    4(R6)                                     ; 1225
                                     17  13 000F7        BEQL    5$                                        ;
                       58            69  3C 000F9        MOVZWL  FILESPECMATCH, R8                         ; 1230
                       58        04  A9  C0 000FC        ADDL2   FILESPECMATCH+4, R8                       ;
        68        04   B6            66  28 00100        MOVC3   (R6), @4(R6), (R8)                        ; 1232
                       04            57  D1 00105        CMPL    INDEX, #4                                 ; 1237
                                     03  12 00108        BNEQ    4$                                        ;
                       68            3B  90 0010A        MOVB    #59, (R8)                                 ; 1239
                       69            66  A0 0010D 4$:    ADDW2   (R6), FILESPECMATCH                       ; 1241
        C5             57            04  F3 00110 5$:    AOBLEQ  #4, INDEX, 3$                             ;
                       49 00000000G  EF  E9 00114 6$:    BLBC    ULTRIX_FLAGS, 12$                         ; 1245
                       56            69  3C 0011B        MOVZWL  FILESPECMATCH, R6                         ; 1250
                       56        04  A9  C0 0011E        ADDL2   FILESPECMATCH+4, R6                       ;
                                     56  D7 00122        DECL    P_NAM                                     ;
                                     57  94 00124        CLRB    ULTRIX_WCARD                              ; 1251
                  04   A9            56  D1 00126 7$:    CMPL    P_NAM, FILESPECMATCH+4                    ; 1252
                                     31  15 0012A        BLEQ    11$                                       ;
                       29            66  91 0012C        CMPB    (P_NAM), #41                              ;
                                     2C  13 0012F        BEQL    11$                                       ;
                       2F            66  91 00131        CMPB    (P_NAM), #47                              ; 1254
                                     16  12 00134        BNEQ    8$                                        ;
```

```
                                50          69  3C 00136          MOVZWL  FILESPECMATCH, R0                                  ; 1257
                                50      04  A9  C0 00139          ADDL2   FILESPECMATCH+4, R0                                ;
                                50          56  C2 0013D          SUBL2   P_NAM, R0                                          ;
                    69          50          01  A3 00140          SUBW3   #1, R0, FILESPECMATCH                              ;
                04  B9      01  A6          69  28 00144          MOVC3   FILESPECMATCH, 1(P_NAM), @FILESPECMATCH+4          ; 1258
                                            11  11 0014A          BRB     11$                                                ;
                                2A          66  91 0014C 8$:      CMPB    (P_NAM), #42                                       ; 1263
                                            05  13 0014F          BEQL    9$                                                 ;
                                25          66  91 00151          CMPB    (P_NAM), #37                                       ;
                                            03  12 00154          BNEQ    10$                                                ;
                                57          01  90 00156 9$:      MOVB    #1, ULTRIX_WCARD                                   ;
                                            56  D7 00159 10$:     DECL    P_NAM                                              ; 1264
                                            C9  11 0015B          BRB     7$                                                 ;
                                04          57  E9 0015D 11$:     BLBC    ULTRIX_WCARD, 12$                                  ; 1267
                    57          69          01  81 00160          ADDB3   #1, FILESPECMATCH, ULTRIX_WCARD                    ;
    FC   AA         00      04  B9          69  2D 00164 12$:     CMPC5   FILESPECMATCH, @FILESPECMATCH+4, #0, -             ; 1270
                                        00  BA     0016B                  WILDCARDMASK, @WILDCARDMASK+4                      ;
                                            02  12 0016D          BNEQ    13$                                                ;
                                            69  B4 0016F          CLRW    FILESPECMATCH                                      ; 1277
                                56      04  AA  9E 00171 13$:     MOVAB   P.ABU, CONSOLE                                     ; 1279
                10  AE      08  AA          1C  28 00175          MOVC3   #28, P.ABV, ITMLST                                 ; 1317
                            14  AE          6E  9E 0017B          MOVAB   DEV_CLASS, ITMLST+4                                ; 1279
                            18  AE      04  AE  9E 0017F          MOVAB   CLASS_LEN, ITMLST+8                                ;
                            20  AE      2C  AE  9E 00184          MOVAB   DEV_NAME, ITMLST+16                                ;
                            24  AE      08  AE  9E 00189          MOVAB   DEV_LEN, ITMLST+20                                 ;
                                        0C  AE  D4 0018E          CLRL    COUNT                                              ; 1319
                    03 0000FE03  9F         04  E1 00191          BBC     #4, @#^X0000FE03, 14$                              ; 1320
                                        012B  31 00199          BRW     30$                                                ;
                        03BC  C9            03  88 0019C 14$:     BISB2   #3, DIR_FLAGS                                      ; 1322
                    66      70  BE          03  29 001A1          CMPC3   #3, @Q_FILEBLOCK+4, (CONSOLE)                      ; 1324
                                            5B  12 001A6          BNEQ    21$                                                ;
                                50 0000FE17  9F  9A 001A8          MOVZBL  @#^X0000FE17, R0                                   ; 1325
                                            12  15 001AF          BLEQ    15$                                                ; 1327
                                04          50  91 001B1          CMPB    R0, #4                                             ;
                                            0D  1A 001B4          BGTRU   15$                                                ;
                                        0C  AE  9F 001B6          PUSHAB  COUNT                                              ; 1331
                                        70  AE  9F 001B9          PUSHAB  Q_FILEBLOCK                                        ;
                        FA4F  CB            02  FB 001BC          CALLS   #2, RT11_DIRECT                                    ;
                                            1D  11 001C1          BRB     18$                                                ;
                                05          50  91 001C3 15$:     CMPB    R0, #5                                             ; 1332
                                            05  1F 001C6          BLSSU   16$                                                ;
                                06          50  91 001C8          CMPB    R0, #6                                             ;
                                            05  1B 001CB          BLEQU   17$                                                ;
                                11          50  91 001CD 16$:     CMPB    R0, #17                                            ;
                                            11  12 001D0          BNEQ    19$                                                ;
                                7E          57  9A 001D2 17$:     MOVZBL  ULTRIX_WCARD, -(SP)                                ; 1335
                                        10  AE  9F 001D5          PUSHAB  COUNT                                              ;
                                        74  AE  9F 001D8          PUSHAB  Q_FILEBLOCK                                        ;
                        FD32  CB            03  FB 001DB          CALLS   #3, ODS_DIRECT                                     ;
                                        00BC  31 001E0 18$:     BRW     25$                                                ;
                                08          50  91 001E3 19$:     CMPB    R0, #8                                             ; 1336
                                            EA  12 001E6          BNEQ    17$                                                ;
00000000G  8F 00000000G  EF     08          00  ED 001E8          CMPZV   #0, #8, DS$GB_SYSTYPE, #PR$_SYS_TYP900             ; 1338
                                            01  13 001F5          BEQL    20$                                                ;
                                                04 001F7          RET                                                        ;
                                        6C  AE  9F 001F8 20$:     PUSHAB  Q_FILEBLOCK                                        ; 1340
                    00000000G  EF           01  FB 001FB          CALLS   #1, PPA$DIRECTORY                                  ;
```

```
                                                04 00202             RET                                             ; 1343
                                50       6C     AE 3C 00203 21$:     MOVZWL  Q_FILEBLOCK, LEN                        ; 1350
                                                50 D7 00207          DECL    LEN                                     ;
                                52              01 CE 00209          MNEGL   #1, R2                                  ; 1351
                                51       70     AE D0 0020C          MOVL    Q_FILEBLOCK+4, R1                       ;
                                   00000000G    EF 16 00210          JSB     LOC$PTABLE                              ;
                                52              51 D0 00216          MOVL    R1, R2                                  ;
                                                0C 12 00219          BNEQ    22$                                     ; 1352
                                         24     AA 9F 0021B          PUSHAB  P.ABW                                   ; 1353
                        00000000G  9F           01 FB 0021E          CALLS   #1, a#DS$PRINTF                         ;
                                                7B 11 00225          BRB     26$                                     ;
                                50       27     A2 3C 00227 22$:     MOVZWL  39(PTABLE), R0                          ; 1355
                            3852   8F           50 B1 0022B          CMPW    R0, #14418                              ; 1357
                                                A0 13 00230 23$:     BEQL    17$                                     ;
                            4152   8F           50 B1 00232          CMPW    R0, #16722                              ;
                                                99 13 00237          BEQL    17$                                     ;
                            4352   8F           50 B1 00239          CMPW    R0, #17234                              ;
                                                92 13 0023E          BEQL    17$                                     ;
                            4452   8F           50 B1 00240          CMPW    R0, #17490                              ;
                                                8B 13 00245          BEQL    17$                                     ;
                            4944   8F           50 B1 00247          CMPW    R0, #18756                              ;
                                                84 13 0024C          BEQL    17$                                     ;
                            4B52   8F           50 B1 0024E          CMPW    R0, #19282                              ;
                                                DB 13 00253          BEQL    23$                                     ;
                            4C52   8F           50 B1 00255          CMPW    R0, #19538                              ;
                                                D4 13 0025A          BEQL    23$                                     ;
                            4D52   8F           50 B1 0025C          CMPW    R0, #19794                              ;
                                                CD 13 00261          BEQL    23$                                     ;
                            5052   8F           50 B1 00263          CMPW    R0, #20562                              ;
                                                C6 13 00268          BEQL    23$                                     ;
                            5852   8F           50 B1 0026A          CMPW    R0, #22610                              ;
                                                BF 13 0026F          BEQL    23$                                     ;
                            4154   8F           50 B1 00271          CMPW    R0, #16724                              ; 1360
                                                1C 13 00276          BEQL    24$                                     ;
                            4554   8F           50 B1 00278          CMPW    R0, #17748                              ;
                                                15 13 0027D          BEQL    24$                                     ;
                            4B54   8F           50 B1 0027F          CMPW    R0, #19284                              ;
                                                0E 13 00284          BEQL    24$                                     ;
                            5354   8F           50 B1 00286          CMPW    R0, #21332                              ;
                                                07 13 0028B          BEQL    24$                                     ;
                            5554   8F           50 B1 0028D          CMPW    R0, #21844                              ;
                                                10 12 00292          BNEQ    27$                                     ;
                                         0C     AE 9F 00294 24$:     PUSHAB  COUNT                                   ; 1361
                                         70     AE 9F 00297          PUSHAB  Q_FILEBLOCK                             ;
                                FB70    CB       02 FB 0029A          CALLS   #2, ANSI_DIRECT                         ;
                                        54       50 D0 0029F 25$:     MOVL    R0, STATUS                              ;
                                                 07 11 002A2 26$:     BRB     28$                                     ;
                                        54 000184C4  8F D0 002A4 27$:     MOVL    #99524, STATUS                     ; 1362
                                        0D           54 E9 002AB 28$:     BLBC    STATUS, 29$                        ; 1365
                                                 0C  AE DD 002AE          PUSHL   COUNT                              ; 1366
                                                 4B  AA 9F 002B1          PUSHAB  P.ABX                              ;
                        00000000G  9F            02 FB 002B4          CALLS   #2, a#DS$PRINTF                         ;
                                50               54 D0 002BB 29$:     MOVL    STATUS, R0                              ; 1367
                                                0000G 30 002BE          BSBW    DSR$COMPLETION                          ;
                                             03BC   C9 94 002C1          CLRB    DIR_FLAGS                          ; 1368
                                                    60 11 002C5          BRB     38$                                ;
                                                    7E 7C 002C7 30$:     CLRQ    -(SP)                              ; 1372
```

```
                                                 7E  7C 002C9          CLRQ    -(SP)                                                  ;
                                              20 AE  9F 002CB          PUSHAB  ITMLST                                                 ;
                                            0080 CE  9F 002CE          PUSHAB  Q_FILEBLOCK                                            ;
                                                 7E  7C 002D2          CLRQ    -(SP)                                                  ;
                                00000000G     00    08  FB 002D4          CALLS   #8, SYS$GETDVI                                         ;
                                              54    50  D0 002DB          MOVL    R0, STATUS                                             ;
                                              01    54  D1 002DE          CMPL    STATUS, #1                                             ; 1373
                                                    3E  12 002E1          BNEQ    37$                                                    ;
                        2C  AE        08  AE        24  3A 002E3          LOCC    #36, DEV_LEN, DEV_NAME                                 ; 1375
                                                    02  12 002E9          BNEQ    31$                                                    ;
                                                    51  D4 002EB          CLRL    R1                                                     ;
                                                    51  D5 002ED 31$:      TSTL    DOLLAR                                                 ; 1376
                                                    05  13 002EF          BEQL    32$                                                    ;
                                20  AE        01    A1  9E 002F1          MOVAB   1(R1), ITMLST+16                                       ; 1377
                            66  20  BE              03  29 002F6 32$:      CMPC3   #3, @ITMLST+16, (CONSOLE)                              ; 1378
                                                    05  12 002FB          BNEQ    33$                                                    ;
                                              63    AA  9F 002FD          PUSHAB  P.ABY                                                  ; 1379
                                                    18  11 00300          BRB     36$                                                    ;
                                              01    6E  D1 00302 33$:      CMPL    DEV_CLASS, #1                                          ; 1380
                                                    05  13 00305          BEQL    34$                                                    ;
                                              02    6E  D1 00307          CMPL    DEV_CLASS, #2                                          ;
                                                    0A  12 0030A          BNEQ    35$                                                    ;
                                    F4A5      CB    00  FB 0030C 34$:      CALLS   #0, VMS_DIR                                            ; 1381
                                              54    50  D0 00311          MOVL    R0, STATUS                                             ;
                                                    0B  11 00314          BRB     37$                                                    ;
                                            0096 CA  9F 00316 35$:      PUSHAB  P.ABZ                                                  ; 1382
                                00000000G     9F    01  FB 0031A 36$:      CALLS   #1, @#DS$PRINTF                                        ;
                                              50    54  D0 00321 37$:      MOVL    STATUS, R0                                             ; 1385
                                                0000G 30  00324          BSBW    DSR$COMPLETION                                         ;
                                              09    08  A9  E8 00327 38$:      BLBS    CONTROLCFLAG, 39$                              ; 1387
                                                    7E  7C 0032B          CLRQ    -(SP)                                                  ; 1388
                                00000000G     9F    02  FB 0032D          CALLS   #2, @#DS$CNTRLC                                        ;
                                                    04  00334 39$:      RET                                                            ; 1390

; Routine Size:  821 bytes,    Routine Base:  CODE + 0BC7

;   1391  1                                                                                                                !G:11
```

```
;    1393  1     %sbttl 'Master routine'
;    1394  1
;    1395  1     global routine dsv$directory : jsb_fake novalue =
;    1396  2         begin
;    1397  2     !
;    1398  2     ! Initialize context to prevent continuation of diagnostic or script.
;    1399  2     !
;    1400  2         script$flush ();                        ! Flush any active scripts
;    1401  2         ds_cleanup ();                          ! Force abort of diagnostic if any
;    1402  2         init_context ();                        ! Clean up stacks, etc.
;    1403  3         begin
;    1404  3
;    1405  3         builtin
;    1406  3             sp;                                 ! 'Fake' return address to avoid trouble when BLISS does RSB
;    1407  3
;    1408  3         sp = .sp - 4;                           ! Do a PUSHAB L^BEGIN_BLISS
;    1409  3         .sp = begin_bliss                       !  (Set address for RSB)
;    1410  2         end;
;    1411  2         dsx$directory (ds$gl_clibase)
;    1412  1         end;                                    ! of dsv$directory



                                00000000G  EF   16 00000 DSV$DIRECTORY::
                                                         JSB     SCRIPT$FLUSH                    ; 1400
                                00000000G  EF   16 00006         JSB     DS_CLEANUP                      ; 1401
                                00000000G  EF   16 0000C         JSB     INIT_CONTEXT                    ; 1402
                             5E                 04   C2 00012         SUBL2   #4, SP                          ; 1408
                             6E 00000000G       EF   9E 00015         MOVAB   BEGIN_BLISS, (SP)               ; 1409
                             52 00000000G       EF   9E 0001C         MOVAB   DS$GL_CLIBASE, R2               ; 1411
                        FCA3 CF                 00   FB 00023         CALLS   #0, DSX$DIRECTORY               ;
                                                     05 00028         RSB                             ; 1412

; Routine Size:  41 bytes,    Routine Base:  CODE + 0EFC

;    1413  1
;    1414  1     end                                     ! End of module
;    1415  1
;    1416  0     eludom



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  DATA                          793  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                          957  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                         3877  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                       0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)
```

Symbol	Type	Defined	Referenced ...									
$ASCIC	Macro	Lib01	211	302	312	419	420	469	470	475	525	532	767
			824	897	952	1001	1032	1120	1353	1366	1379	1382	
$ASCID	Macro	Lib01	626	635	684	728	737	809	900	977	1000	1022	1061
			1185										
$CLOSE	KeyWMacr	Lib03	928	1083	1110	1122							
$CONNECT	KeyWMacr	Lib03	1080										
$DS_CNTRLC	KeyWMacr	Lib01	1188	1388									
$DS_DSADEF	Macro	Lib01	205										
$DS_HPO_F	Fieldset	Lib01	1302										
$DS_PRINTF	Macro	Lib01	302	312	419	420	469	470	475	525	532	767	824
			897	952	1001	1032	1120	1353	1366	1379	1382		
$FAB_DECL	Macro	Lib03	364	883	965								
$FAB_INIT	KeyWMacr	Lib03	383	901	1066								
$FAO	Macro	Lib03	626	635	684	728	737	809	900	1000	1022	1061	
$GET	KeyWMacr	Lib03	1091										
$NAM_DECL	Macro	Lib03	363	884									
$NAM_INIT	KeyWMacr	Lib03	377	902									
$OPEN	KeyWMacr	Lib03	1069										
$PARSE	KeyWMacr	Lib03	912										
$RAB_DECL	Macro	Lib03	966										
$RAB_INIT	KeyWMacr	Lib03	1067										
$RMS_BITFLD_INI	Macro	Lib03	382	390	901	902	1066	1067					
$RMS_BITS	Macro	Lib03	390	901	1066								
$RMS_CODFLD_INI	Macro	Lib03	390	901	1066	1067							
$RMS_OR	Macro	Lib03	390	901	1066								
$RMS_PTR	Unbound		1067										
	Bind	382	382=										
	Bind	390	390=										
	Bind	901	901=										
	Bind	902	902=										
	Bind	1066	1066=										
	Bind	1067	1067=										
$RMS_VALFLD_INI	Unbound		1067										
	Macro	Lib03	382	390	901	902	1066	1067					
ADDRESS	RoutForm	543	562m	571a=	572a=	573a=	575a.	575.					
ADDRESS_FIRST	Local	754	791=	797=	797.								
ANSI_DIRECT	Routine	831	1361c										
BASE_DIRECT	Bind	977	1007.										
BASE_DIRECTORY	Local	970	1007=	1052=	1053=	1062a							
BBLOCK	Structur	Lib02	254	255	298	311	337	339	343	347	360	361	362
			369	756	757	760	761	797	879	880	885	886	887
			890	967	968	969	970	971	972	973	974	978	1178
			1182	1302									
BEGIN_BLISS	External	248	1409a										
BOO$AL_VECTOR	External	251											
BOO$QIO	ExtRout	258	781c										
BOO$SELECT	External	250											
BREAKUP_FILE	ExtRout	265	1194c	1195c	1197c	1202c	1203c	1205c	1207c	1209c			
BTD$K_CONSOLE	Literal	Lib03	763										
BUFFER	RoutForm	494	513m	524a=	524a.	525.	526a=	527a=	529a=	529a.	532.	533a=	534a=
	RoutForm	580	617m	621a.	623a=	623a.	625a.	629a.	631a=	631a.	634a.	637.	

Symbol	Type	Defined	Referenced ...
	RoutForm	644	669m 675a. 677a= 677a. 683a. 685.
	RoutForm	691	714m 723a. 725a= 725a. 727a. 731a. 733a= 733a. 736a. 739.
	Local	889	892= 894a 945a 946a
	Local	967	1067a
CALL_CLI	Linkage	Lib02	1132
CLASS_LEN	Local	1307	1313a
CLI$L_FILE_ADR	Macro	Lib02	1209
CLI$W_FILE_LEN	Macro	Lib02	1209
CLI_BLOCK	RoutForm	1132	1182m 1209a.
CONSOLE	Local	1301	1301= 1324a. 1378a.
CONTROLCFLAG	Own	222	815. 948. 1116. 1162= 1187= 1387.
COUNT	RoutForm	494	520a= 520a.
	RoutForm	580	637.
	RoutForm	644	685.
	RoutForm	691	739.
	RoutForm	745	810.
	RoutForm	831	947.
	RoutForm	960	1096. 1103. 1106.
	Local	1171	1319= 1331a 1335a 1346a 1359a 1361a 1366.
DATA	RoutForm	543	571. 572. 573.
DC$_DISK	Literal	Lib03	1380
DC$_TAPE	Literal	Lib03	1380
DEF$C_BKSTP_LEN	ExtLit	242	1202a
DEF$C_ULTRIX_LEN	ExtLit	241	1194a
DEF$Q_BACKSTOP	External	252	1202a
DEF$Q_DEV	External	254	1195. 1203.
DEF$Q_DIR	External	255	1197. 1205.
DEF$Q_ULTRIX	External	253	1194a
DEFAULT	Bind	367	369m 390.
DEF_SPEC	Local	355	355= 367.
DESC	RoutForm	494	514m 517. 524a. 529a.
	Local	613	624= 625= 626a 633= 634= 635a 637a
	Local	663	682= 683= 684a 685a
	Local	710	726= 727= 728a 735= 736= 737a 739a
	Bind	1218	1220. 1224. 1225. 1233. 1234a. 1241.
DESTINATION	Bind	1230	1235= 1239=
DEVICE_D	RoutForm	278	295.
	Local	347	347= 409a 413. 413a. 424a
DEV_BUF	Local	346	349a
DEV_CLASS	Local	1306	1312a 1380.
DEV_D	Bind	295	298m 299a= 300. 301= 302a
DEV_LEN	Local	1308	1316a 1375.
DEV_NAME	Local	1309	1315a 1375.
DIR	RoutForm	580	616m 626a. 626aa 635a. 635aa
	RoutForm	644	668m 671a. 679a. 681a. 684a.
	RoutForm	691	713m 717a. 728aa 728a. 737aa 737a.
DIR$B_NAMECOUNT	Macro	Lib03	717 728 737
DIR$C_LENGTH	Literal	Lib03	717
DIR_FLAGS	Global	1286	1322= 1323= 1368=
DIR_V_CMM	GlobLit	1294	1322
DIR_V_DRVCLR	GlobLit	1295	1323
DOLLAR	Local	1303	1375= 1376. 1377.
DS$CNTRLC	ExtRout	1188	1188c 1388e 1388c
DS$GB_SYSTYPE	External	247	1338.
DS$GL_CLIBASE	External	249	372. 372a. 373. 1411a

Symbol	Type	Defined	Referenced ...
DS$PRINTF	ExtRout	302	302c 312e 312c 419e 419c 420e 420c 469e 469c 470e 470c 475e 475c 525e 525c 532e 532c 767e 767c 824e 824c 897e 897c 952e 952c 1001e 1001c 1032e 1032c 1120e 1120c 1353e 1353c 1366e 1366c 1379e 1379c 1382e 1382c
DSA$AL_APTMAIL	Bind	205	205
DSA$GL_APTCOM	Bind	205	
DSA$GL_DEVLEN	Bind	205	
DSA$GL_DEVNAM	Bind	205	
DSA$GL_ERRNO	Bind	205	
DSA$GL_EVENT	Bind	205	
DSA$GL_FLAGS	Bind	205	876 904 925 1320
DSA$GL_MSGPTR	Bind	205	
DSA$GL_MSGTYP	Bind	205	
DSA$GL_PASSES	Bind	205	
DSA$GL_PASSNO	Bind	205	
DSA$GL_SECTNO	Bind	205	
DSA$GL_SID	Bind	205	208m 1325
DSA$GL_SUBTNO	Bind	205	
DSA$GL_TESTNO	Bind	205	
DSA$GL_UNITS	Bind	205	
DSA$V_USER	Macro	Lib01	876 904 925 1320
DSC$A_POINTER	Macro	Lib03	299 300 313 374 389 405 413 429 434 445 446 457 458 527 534 625 634 683 727 736 766 808 894 899 943 995 1008 1020 1027 1053 1055 1066 1089 1195 1197 1203 1205 1207 1212 1225 1230 1234 1250 1252 1257 1258 1272 1274 1324 1351
DSC$C_S_BLN	Literal	Lib03	298 337 339 343 347 369 760 761 880 969 970 971 973 978 1178 1193
DSC$W_LENGTH	Macro	Lib03	301 373 388 390 413 428 429 434 439 445 447 457 459 516 524 525 526 529 532 533 621 623 624 625 629 631 633 634 675 677 682 683 723 725 726 727 731 733 735 736 765 769 770 807 808 824 893 898 901 942 945 946 952 994 1009 1021 1026 1052 1054 1066 1079 1088 1120 1195 1197 1203 1205 1207 1211 1220 1224 1230 1233 1241 1250 1257 1258 1267 1271 1273 1277 1350
DSR$ALLOCATE_PSEUDO_RPB	ExtRout	268	763c
DSR$COMPLETION	ExtRout	270	1367c 1383c 1385c
DSR$DEALLOCATE_PSEUDO_RPB	ExtRout	269	818c 826c
DSR$KEY_EQUAL	ExtRout	266	517c
DSV$DIRECTORY	GlobRout	1395	
DSX$DIRECTORY	GlobRout	1132	1411c
DS_CLEANUP	ExtRout	263	1401c
DVI$_DEVCLASS	Literal	Lib03	1311
DVI$_DEVNAM	Literal	Lib03	1314
END_SIZE	Global	226	1003= 1078.
ENTRY	Bind	797	799. 801. 804. 806.
EXPAND	Local	886	902a
EXP_STR	Own	362	382a
EXTRA_LENGTH	Local	753	790= 791. 797.
FAB	Local	883	901a 912a 918a 918= 928a
	Local	965	1066a 1067a 1069a 1083a 1099. 1110a 1122a
FAB$B_BID	Macro	Lib03	390 901 1066
FAB$B_BKS	Macro	Lib03	390 901 1066
FAB$B_BLN	Macro	Lib03	390 901 1066

Symbol	Type	Defined	Referenced ...									
FAB$B_DNS	Macro	Lib03	390	901	1066							
FAB$B_FAC	Macro	Lib03	390	901	1066							
FAB$B_FNS	Macro	Lib03	390	901	1066							
FAB$B_FSZ	Macro	Lib03	390	901	1066							
FAB$B_JOURNAL	Macro	Lib03	390	901	1066							
FAB$B_ORG	Macro	Lib03	390	901	1066							
FAB$B_RAT	Macro	Lib03	390	901	1066							
FAB$B_RFM	Macro	Lib03	390	901	1066	1099						
FAB$B_RTV	Macro	Lib03	390	901	1066							
FAB$B_RU_FACILITY	Macro	Lib03	390	901	1066							
FAB$B_SHR	Macro	Lib03	390	901	1066							
FAB$C_BID	Literal	Lib03	390	901	1066							
FAB$C_BLN	Literal	Lib03	364	390	883	901	965	1066				
FAB$C_FIX	Literal	Lib03	1099	1102								
FAB$C_VAR	Literal	Lib03	390	901	1066	1099	1105					
FAB$L_ALQ	Macro	Lib03	390	901	1066							
FAB$L_CTX	Macro	Lib03	390	901	1066							
FAB$L_DNA	Macro	Lib03	390	901	1066							
FAB$L_FNA	Macro	Lib03	390	901	1066							
FAB$L_FOP	Macro	Lib03	390	901	1066							
FAB$L_MRN	Macro	Lib03	390	901	1066							
FAB$L_NAM	Macro	Lib03	390	901	1066							
FAB$L_XAB	Macro	Lib03	390	901	1066							
FAB$M_GET	Literal	Lib03	390	901	1066							
FAB$M_NAM	Literal	Lib03	390									
FAB$M_RWO	Literal	Lib03	901									
FAB$V_CHAN_MODE	Macro	Lib03	390	901	1066							
FAB$V_FILE_MODE	Macro	Lib03	390	901	1066							
FAB$V_LNM_MODE	Macro	Lib03	390	901	1066							
FAB$V_RWO	Macro	Lib03	918									
FAB$W_BLS	Macro	Lib03	390	901	1066							
FAB$W_DEQ	Macro	Lib03	390	901	1066							
FAB$W_GBC	Macro	Lib03	390	901	1066							
FAB$W_MRS	Macro	Lib03	390	901	1066							
FAB_BLK	Own	364	390a	396a	403a	462a						
FILEDESC	Local	880	898=	899=	900a	901.	942=	943=	947a			
FILEMATCH	Local	1177	1212a									
FILENAM	Local	879	899a	901a								
FILENAME_D	Local	339	339=	428=	429.	429a=	434.	434a.	439.	445.	445a.	447.	457.
			457a.	459.									
FILES	Local	350	350=	408.	418.	419.	420.	421.	422=	431=	431.	468.	469.
			470.	473.									
FILESPEC	RoutForm	831	897.	900.									
FILESPECMATCH	Own	221	516.	517a	769.	770=	770.	1079=	1211=	1212=	1220.	1230.	1241=
			1241.	1250.	1252.	1257=	1257.	1258.	1258a=	1267.	1271.	1272a.	1277=
FILE_BUF	Local	338	341a									
FIRST_FILE	Local	887	896=	936.	938.	938=						
FIRST_LEN	Local	888	895=	936.	938=							
FORMAT_ODS1	Routine	644	1103c									
FORMAT_ODS2	Routine	691	1106c									
FORMAT_ULTRIX	Routine	580	1096c									
F_LINE	Bind	211										
HEADER	Routine	278	409c	424c								
HP$T_TYPE	Field	Lib01	1355									
INDEX	Local	719	728.	737.								

Symbol	Type	Defined	Referenced ...
	Local	1214	1218. 1237.
INIT_CONTEXT	ExtRout	262	1402c
INTERCEPTCONTROLC	Routine	1160	1188a
IO$_READLBLK	Literal	Lib03	784
ITMLST	Local	1310	1310= 1317= 1372a 1377= 1378a.
JSB_0	Linkage	Lib02	270
JSB_DEVICE	Linkage	1289	
JSB_FAKE	Linkage	214	1395
JSB_LOC$PTABLE	Linkage	Lib02	267
JSB_NONE	Linkage	Lib02	260 261 262 263
JSB_SCAN	Linkage	215	259
KB_CHECK	ExtRout	260	
L	Bind	749	
LAST_DOT	Local	1043	1047= 1052. 1054. 1055.
LEFT	Local	981	1018= 1022.
LEN	Local	1304	1350= 1351.
LENGTH	RoutForm	580	
	RoutForm	644	
	RoutForm	691	719.
	Local	1172	
LIB$SIGNAL	ExtRout	272	
LINE_BUF	Local	342	342= 345a
LINE_D	RoutForm	307	311a 312. 313a=
	Local	343	343= 405a= 417a 434a= 435a 446. 449a 456a 458a= 467a
LNM$C_NAMLENGTH	Literal	Lib03	346
LOC$PTABLE	ExtRout	267	1351c
L_DOT	Local	984	
L_LPART	Local	988	1016= 1017. 1018.
L_REST	Local	990	1009= 1010. 1017. 1036. 1048. 1054.
L_RPART	Local	987	1017= 1019.
MAX_DEVNAME_LEN	Literal	1298	1309
NAM	Bind	294	297a 299. 299a. 300. 300a. 301.
	Local	884	901a 902a 934.
NAM$B_BID	Macro	Lib03	382 902
NAM$B_BLN	Macro	Lib03	382 902
NAM$B_DEV	Macro	Lib03	299 300 301 414
NAM$B_DIR	Macro	Lib03	300 301 414
NAM$B_ESS	Macro	Lib03	382 902
NAM$B_NAME	Macro	Lib03	428
NAM$B_NOP	Macro	Lib03	382 902
NAM$B_RSL	Macro	Lib03	934
NAM$B_RSS	Macro	Lib03	382 902
NAM$B_TYPE	Macro	Lib03	428
NAM$B_VER	Macro	Lib03	428
NAM$C_BID	Literal	Lib03	382 902
NAM$C_BLN	Literal	Lib03	363 382 884 902
NAM$C_MAXRSS	Literal	Lib03	360 361 362 382 879 885 886 898 902 1177 1220
NAM$L_DEV	Macro	Lib03	299 414
NAM$L_DIR	Macro	Lib03	300
NAM$L_ESA	Macro	Lib03	382 902
NAM$L_NAME	Macro	Lib03	429
NAM$L_RLF	Macro	Lib03	382 902
NAM$L_RSA	Macro	Lib03	382 902
NAME	Local	664	679= 679a 680a 684a
	Local	758	804= 804a 805a 809a

Symbol	Type	Defined	Referenced ...								
NAM_BLK	RoutForm	278	294.								
	Own	363	382a	390a	409a	414.	414a.	424a	428.	429a.	
NAM_LEN	Local	922	941=	942.	945.	946.					
NAM_PTR	Local	923	940=	941.	943.	945a.	946a.				
NEXT_FILE	BindRout	874	918c								
ODS_DIRECT	Routine	960	1335c	1346c	1359c						
ONE	GlobReg	1283									
OUTBUFFER	Local	760	765=	766=	808.	810a	824.	824a			
	Local	890	893=	894=	945.	946.	947a	952.	952a		
POINTER	Local	1173									
PPA$DIRECTORY	ExtRout	1166	1340c								
PPA_STAT	Local	1175	1340=	1341.							
PR$V_SID_TYPE	Macro	Lib03	1325								
PR$_SID_TYP730	Literal	Lib03	1329								
PR$_SID_TYP750	Literal	Lib03	1328								
PR$_SID_TYP780	Literal	Lib03	1327								
PR$_SID_TYP790	Literal	Lib03	1330								
PR$_SID_TYP8NN	Literal	Lib03	1333								
PR$_SID_TYP8RR	Literal	234	1334								
PR$_SID_TYP8SS	Literal	Lib03	1332								
PR$_SID_TYPUV	Literal	Lib03	1336								
PR$_SYS_TYP900	ExtLit	243	1338								
PTABLE	Local	1302	1351=	1352.	1355.						
PTR	RoutForm	307	314a=								
	Local	356	356=	417.	417a	435a	439.	446.	447=	447.	448.	449a	456a
			459=	466.	467a								
	Local	565	575=	575.									
PUTBUFFER	Routine	494	637c	685c	739c	810c	947c						
PUT_LINE	Routine	307	417c	435c	449c	456c	467c						
P_COMMA	Local	989	1010=	1012.	1015.	1016.							
P_DOT	Local	983	1036=	1038.	1045.	1047.	1048=	1048.	1049.				
P_LPART	Local	985	1008=	1010a.	1016.	1018.	1036a.	1048.	1052.	1053.	1054.		
P_NAM	Local	1249	1250=	1252.	1252a.	1254a.	1257.	1258.	1263a.	1264=	1264.		
P_RPART	Local	986	1015=	1019.									
QIO$CLEANUP	ExtRout	264											
Q_BUFFER	Local	761	807=	808=	809a	810a							
	Local	973	1088=	1089=	1096a	1103a	1106a	1120.	1120a				
Q_DEVICE	RoutForm	745	749.	767.									
Q_DIRECT	Bind	978	1008.	1009.	1026.	1027.							
Q_DIRECTORY	Local	969	1020=	1021=	1022a	1026=	1027=	1032a	1054=	1055=	1062a		
Q_FILE	Local	971	994=	995=	1000a	1001a	1002.	1003=	1003.	1062a	1066.		
Q_FILEBLOCK	Local	1178	1193=	1194a	1195a	1197a	1202a	1203a	1205a	1207a	1209a	1218a	1324a.
			1331a	1335a	1340a	1346a	1350.	1351.	1359a	1361a	1372a		
Q_FILSPEC	RoutForm	960	978.	1000.	1032.	1062.							
RAB	Local	966	1067a	1080a	1091a	1096.	1103.	1106.					
RAB$B_BID	Macro	Lib03	1067										
RAB$B_BLN	Macro	Lib03	1067										
RAB$B_KRF	Macro	Lib03	1067										
RAB$B_KSZ	Unbound		1067										
RAB$B_MBC	Macro	Lib03	1067										
RAB$B_MBF	Macro	Lib03	1067										
RAB$B_RAC	Macro	Lib03	1067										
RAB$B_TMO	Macro	Lib03	1067										
RAB$C_BID	Literal	Lib03	1067										
RAB$C_BLN	Literal	Lib03	966	1067									

Symbol	Type	Defined	Referenced ...
RAB$L_BKT	Macro	Lib03	1067
RAB$L_CTX	Macro	Lib03	1067
RAB$L_FAB	Macro	Lib03	1067
RAB$L_KBF	Unbound		1067
RAB$L_RBF	Macro	Lib03	1067 1096 1103 1106
RAB$L_RHB	Macro	Lib03	1067
RAB$L_ROP	Macro	Lib03	1067
RAB$L_UBF	Macro	Lib03	1067
RAB$L_XAB	Macro	Lib03	1067
RAB$W_RSZ	Macro	Lib03	1067 1096 1103 1106
RAB$W_USZ	Macro	Lib03	1067
RAD50	Routine	543	679c 681c 804c 806c
RESULT	Local	885	902a 936. 938. 940. 941a
RES_LEN	Local	881	934= 936. 938. 940. 941.
RES_STR	Own	360	382a
RIGHT	Local	982	1019= 1022.
RMS$_DEV	Literal	Lib03	1362
RMS$_DNF	Literal	Lib03	1073
RMS$_EOF	Literal	Lib03	1125
RMS$_FNF	Literal	Lib03	1073
RMS$_NMF	Literal	Lib03	464 954
RMS$_NORMAL	Literal	Lib03	819 827 948 954 1125
RMS$_SUC	Literal	Lib03	398 406 477
RPB	Local	755	763= 786. 818. 826.
RT$L_FILENAME	Macro	Lib02	804
RT$V_ENDSEG	Macro	Lib02	799
RT$V_PERM	Macro	Lib02	801
RT$W_EXTRALEN	Macro	Lib02	790
RT$W_FILETYPE	Macro	Lib02	806
RT$W_FIRST	Macro	Lib02	791
RT$W_SEGMENT	Macro	Lib02	789
RT$_FIXED	Literal	Lib02	791 797
RT1I_DIRECT	Routine	745	1331c
SAME_DIR	Local	357	357= 413= 415.
SCAN$NUMERIC	ExtRout	259	1018c 1019c
SCRIPT$FLUSH	ExtRout	261	1400c
SDL$STARLET_CONCAT	Macro	Lib03	912 928 1069 1080 1083 1091 1110 1122
SDL$STARLET_OPT	Macro	Lib03	912 928 1069 1080 1083 1091 1110 1122
SDL$STARLET_REQ	Macro	Lib03	912 928 1069 1080 1083 1091 1110 1122
SEGBLK	Local	756	781a 789. 790. 791a
SEGMENT	Local	752	764= 779. 783. 789=
SP	Builtin	1406	1408= 1408. 1409a=
SPEC	Own	361	372= 374a 390a
SS$_FILESTRUCT	Literal	Lib03	1111
SS$_NORMAL	Literal	Lib03	1373
START_SIZE	Local	991	1002= 1078.
STAT2	Local	354	354= 403= 406. 462= 464. 477= 478. 480.
STATUS	Local	353	353= 396= 398. 482.
	Local	777	781= 787.
	Local	882	918= 918. 928= 930. 954. 954= 956.
	Local	910	912= 914.
	Local	964	1069= 1071. 1074. 1080= 1084. 1091= 1123. 1126.
	Local	1174	
	Local	1305	1331= 1335= 1346= 1359= 1361= 1362= 1365. 1367. 1372= 1373. 1381= 1383. 1385.

Symbol	Type	Defined	Referenced ...										
SYS$CLOSE	ExtRout	928	928c	1083e	1083c	1110e	1110c	1122e	1122c				
SYS$CONNECT	ExtRout	1080	1080c										
SYS$FAO	ExtRout	626	626c	635e	635c	684e	684c	728e	728c	737e	737c	809e	809c
			900e	900c	1000e	1000c	1022e	1022c	1062e	1062c			
SYS$GET	ExtRout	1091	1091c										
SYS$GETDVI	ExtRout	1167	1372c										
SYS$GETSYI	ExtRout	274											
SYS$OPEN	ExtRout	871	876a	1069e	1069c								
SYS$PARSE	ExtRout	273	396c	912e	912c								
SYS$SEARCH	ExtRout	271	403c	462c	870e	876a							
TOTAL_DIRS	Local	352	352=	423=	423.	471.	474=	474.	475.				
TOTAL_FILES	Local	351	351=	421=	421.	473=	473.	475.					
TYPE	Local	665	681=	681a	684a								
	Local	759	806=	806a	809a								
T_BUFFER	Local	757	766a	768=									
	Local	974	1087=	1089a									
T_DIRECTORY	Local	968	1020a										
T_FILE	Local	972	995a										
ULTRIX_FLAGS	External	246	997.	1076.	1094.	1190.	1245.						
ULTRIX_V_MODE	Literal	233	997	1076	1094	1190	1245						
ULTRIX_WCARD	RoutForm	960	1003.										
	Local	1176	1251=	1263=	1267.	1267=	1335.	1346.	1359.				
USER_SPEC	Local	337	337=	373=	374=	390.							
VER_OFFSET	Bind	717	719										
VMS_DIR	Routine	319	1381c										
WIDE_FLAG	Global	225	432.	522.	619.	721.	944.						
WILD	Local	510	517a										
WILDCARDMASK	Bind	1185	1207.	1273.	1274a.								
XLATE	Bind	568	571.	572.	573.								
ZERO	GlobReg	1282											

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIRECTORY.B32;1
17	Module DIRECTORY	
175	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
177	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
179	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
1416	Eludom DIRECTORY	

```
        KEY TO REFERENCE TYPE FLAGS

            .   Fetch
            =   Store
            c   Routine call
            a   Address use
            @   Indirect use
            e   External, external routine, or external literal declaration
            f   Forward or forward routine declaration
            h   Condition handler enabling
            m   Map declaration
            u   Undeclare declaration


;                              Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                            940        20         2        96          00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                            675        15         2        47          00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2             20450       125         0      1101          00:00.8




;                                   COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:DIRECTORY.B32

; Size:          3877 code + 1750 data bytes
; Run Time:         00:12.6
; Elapsed Time:     00:15.9
; Lines/CPU Min:     6742
; Lexemes/CPU-Min:125557
; Memory Used:  365 pages
; Compilation Complete
```

Table of contents

(1)	176	Libraries and Symbol Definitions
(1)	210	Work Declarations
(1)	222	Data Declarations
(1)	271	Dispatch Routine
(1)	535	DSX$EndPass Routine
(1)	633	DS_Cleanup Routine
(1)	712	DSX$DoSummary and VRSummary Routines
(1)	771	VRShowStatus Routine

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DISPAT.MAR
                        0000     2 ; *4    19-OCT-1987 17:30:33 LI "Fixed the EndPass error message on subtest#"
                        0000     3 ; *3    20-MAY-1986 10:28:40 BERGAZZI "/TIME"
                        0000     4 ; *2    19-MAY-1986 16:41:40 BERGAZZI "/TIME"
                        0000     5 ; *1     6-FEB-1986 15:17:28 DS ""
                        0000     6 ; DEC/CMS REPLACEMENT HISTORY, Element DISPAT.MAR
                        0000     7         .TITLE  DISPAT  *** DISPAT Diagnostic test sequence control
                        0000     8         .IDENT  /10.10-04/                                                    ;G:4
                        0000     9         .NoShow Conditionals
                        0000    10
                        0000    11 ;
                        0000    12 ; Copyright (c) 1977, 1982, 1985, 1986                                      ;G:2
                        0000    13 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000    14 ;
                        0000    15 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    16 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    17 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    18 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    19 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    20 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    21 ; REMAIN IN DEC.
                        0000    22 ;
                        0000    23 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    24 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    25 ; CORPORATION.
                        0000    26 ;
                        0000    27 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    28 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    29
                        0000    30 ;++
                        0000    31 ; FACILITY:
                        0000    32 ;       VAX DIAGNOSTIC SUPERVISOR.
                        0000    33 ;
                        0000    34 ; ABSTRACT:
                        0000    35 ;
                        0000    36 ; ENVIRONMENT:
                        0000    37 ;
                        0000    38 ; AUTHOR:
                        0000    39 ;       KEN CHAPMAN     10-NOV-77       VERSION 01.
                        0000    40 ;
```

```
0000    ; MODIFIED BY:
0000    ;                TOM SOUTTER     14-NOV-77       VERSION 02.
0000    ;       01       DSPR #24 & #25 -- CLEANUP SEQUENCING PROBLEMS.
0000    ;                N.  HOWGATE     16-NOV-77       VERSION 03.
0000    ;       02       DSPR #42 -- APT FUNCTION UPDATE
0000    ;                TOM SOUTTER     08-DEC-77       VERSION 04.
0000    ;       03       TURNED ON EXECUTION INFO. MESSAGES.
0000    ;       04       DSPR #43 -- AUTOMATIC "DOSUMMARY" IN FINAL "ENDPASS".
0000    ;       05       DSPR #64 -- CHECK LOAD FLAG FOR SUMMARY COMMAND.
0000    ;       06       SET EXECUTE FLAG BEFORE CHECKING EXIT TEST STATUS. (KWC)
0000    ;                NICK HOWGATE    30-DEC-77       VERSION 05.
0000    ;       07       DSX$ENDPASS - CHANGE REFERENCE TO DSA$GL_PASSES
0000    ;                Roger Riggs     12-Jun-78       VERSION 08
0000    ;       08       Standardizing primary messages (first pass done/end run)
0000    ;       09       Check flag DS$V_DONFLG in DS$GL_FLAGS so cleanup is done
0000    ;                only once.
0000    ;       10       Fix to use .ENTRY and .VECTOR in proceedures
0000    ;                Roger Riggs     12-Jun-1978
0000    ;       11       Removed deallocate from cleanup code.
0000    ;                Cancel ^O before calling summary  and end pass message
0000    ;                Dave Butenhof   15-feb-80
0000    ;       12       Add hooks for 'ERROR SEARCH' function
0000    ;                Roger Riggs,    12-Mar-1980
0000    ;       13       Added call to INITSCB and reset memory management after cleanup
0000    ;                also call QIO$CLEANUP only if program is SEP_FUNCTIONAL
0000    ;
0000    ;                Dave Butenhof   30-jun-1980, version 5.5
0000    ;       14       Correct word displaced mode to longword displaced
0000    ;       15       Support DS RMS by calling cleanup routine to avoid
0000    ;                cluttering pool space in case of errors
0000    ;
0000    ;                Dave Butenhof, 03-oct-1980, version 6.1
0000    ;       16       Change some more word-relative references to longword relative
0000    ;       17       Clear DS$V_CTRLC bit before doing $DS_CNTRLC call; this is to
0000    ;                prevent probably undesired control C at next call to KB_CHECK,
0000    ;                if control C's were disabled by previous program.
0000    ;                Dave Butenhof, 29-dec-1980, version 6.2
0000    ;       18       More word relative references to longword.  This is getting
0000    ;                quite un-fun.
0000    ;                Dave Butenhof, 17-feb-1981, version 6.3
0000    ;       19       More of same.  ALL DSA$?_? references this time.
0000    ;       20       - dave butenhof, 06-Oct-1981, version 6.5
0000    ;                Add SHOW STATUS command
0000    ;       21       - Dave Butenhof, 02-Nov-1981, version 6.-
0000    ;                Fix truncation errors
```

```
0000    88 ;       22      - Jack Stansbury, 03-Nov-1981, Version 6.-
0000    89 ;               Added calls to QA$Main for the QA enhancements. Also fixed
0000    90 ;               some truncation errors. Also changed PSECTs a bit. Added
0000    91 ;               .LIBRARY statements for $DS, $DIAG, and LIB.
0000    92 ;
0000    93 ;       23      - Dave Butenhof, 16-Nov-1981, version 6.5
0000    94 ;               Modified the text (and corrected a bug) in the SHOW STATUS
0000    95 ;               command.
0000    96 ;
0000    97 ;       24      - Dave Butenhof, 19-Nov-1981, version 6.5
0000    98 ;               Modify to use typecodes on all output messages.
0000    99 ;
0000   100 ;       25      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000   101 ;               Commented out several of the calls to QA_MAIN because it
0000   102 ;               appears as though they will not be needed for the 6.5 DS
0000   103 ;               version. Made several other small code changes to the DISPATCH
0000   104 ;               routine for QA.
0000   105 ;
0000   106 ;       26      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000   107 ;               Changed DSX$ENDPASS routine slightly to accomadate QA.
0000   108 ;
0000   109 ;       27      - Jack Stansbury, 16-Dec-1981, Version 6.6
0000   110 ;               Added the DS_QADEFS macro definition.
0000   111 ;
0000   112 ;       28      - Dave Butenhof, 29-Dec-1981, Version 6.6
0000   113 ;               Print out all error counts on Summary command.
0000   114 ;
0000   115 ;       29      - Jack Stansbury, 6-Jan-1982, Version 6.6
0000   116 ;               Changed EndPass routine slightly to accomodate the
0000   117 ;               new way of implementing QA (with the QA Start Loop).
0000   118 ;
0000   119 ;       30      - Dave Butenhof, 12-Jan-1982, Version 6.6
0000   120 ;               Fix EndPass message to avoid losing end of message on
0000   121 ;               narrow screen.
0000   122 ;
0000   123 ;       31      - Jack Stansbury, 15-Jan-1982, Version 6.6
0000   124 ;               Made changes to the Dispatch routine for the Loop on Subtest
0000   125 ;               QA check routine.
0000   126 ;
0000   127 ;       32      - Jack Stansbury, 9-Mar-1982, Version 6.6
0000   128 ;               Inserted some new DS macros that will help these routines
0000   129 ;               be more understandable.
0000   130 ;
0000   131 ;       33      Marion Baggett, 9-Apr-1982, Version 6.7
0000   132 ;               Added NoShow Conditionals after the .IDENT
0000   133 ;
0000   134 ;       34      Dave Butenhof, 20-Apr-1982, Verison 6.7
0000   135 ;               Add kluge code in DS_CLEANUP; if standalone on 730,
0000   136 ;               reset IDC to prevent extra interrupts.
0000   137 ;
0000   138 ;       35      Jack Stansbury, 13-September-1982, Version 6.9
0000   139 ;               Fixed some truncation errors. I didn't mark those that I fixed.
0000   140 ;
0000   141 ;       36      Jack Stansbury, 15-September-1982, Version 6.9
0000   142 ;               Changed the VRSummary and VRShowStatus routines to call the
0000   143 ;               DSR$CheckLoad routine defined in ShowMem.
0000   144 ;
```

```
0000   145 ;       37      Jack Stansbury, 8-Mar-1983, Version 6.11
0000   146 ;               Changed the VRShowStatus routine to look for the user's PC
0000   147 ;               on the stack (in the CLI args).
0000   148 ;
0000   149 ;       38      Amy Iorio, 7-January-1985, Version 8.0
0000   150 ;               Added error message to be reported when subtest requested
0000   151 ;               which is greater than the number of existing subtests.
0000   152 ;
0000   153 ;       39      Amy Iorio, 9-January-1985, Version 8.0
0000   154 ;               Changed FMTLSTPAS in order to print
0000   155 ;               pass counts greater than 100,000 out.
0000   156 ;
0000   157 ;       40      Domenic Andella, 18-Mar-1985, Version 8.2
0000   158 ;               Add call in DS_CLEANUP to MP$_DEALL_MEMORY
0000   159 ;               to check for and deallocate any mp memory.
0000   160 ;
0000   161 ;       41      Amy Iorio, 8-May-1985, Version 8.2
0000   162 ;               Fix to loop-on-subtest.
0000   163 ;
0000   164 ;       42      Domenic Andella, 11-Dec-1985    Version 9.1
0000   165 ;               Add external symbol MP$_DEALL_MEMORY
0000   166 ;
0000   167 ;       43      Bob Bergazzi    16-May-1986     Version 10.1            ;G:2
0000   168 ;               Cancel /TIME in DS_CLEANUP.                             ;G:2
0000   169 ;                                                                       ;G:4
0000   170 ;               Daniel Li       19-Oct-1987     Version 10.10           ;G:4
0000   171 ;               Commented out move subtno inst. in EndPass routine to print   ;G:4
0000   172 ;               correct subtest# of last test in the error on subtest message. ;G:4
0000   173 ;                                                                       ;G:4
0000   174 ;--
```

```
0000    176          .SBTTL  Libraries and Symbol Definitions
0000    177 ;
0000    178 ; INCLUDE FILES:                                                          ;G:4
0000    179 ;
0000    180
0000    181          .LIBRARY        /SYS$LIBRARY:LIB/       ;                      [22]  ;G:2
0000    182          .LIBRARY        /$DS/                   ;                      [22]  ;G:2
0000    183          .LIBRARY        /$DIAG/                 ;                      [22]  ;G:2
0000    184
0000    185 ;
0000    186 ;       EXTERNAL SYMBOLS
0000    187 ;
0000    188          .EXTRN  MP$_DEALL_MEMORY                ;                      [42]
0000    189
0000    190
0000    191 ;
0000    192 ; MACROS:
0000    193 ;
0000    194
0000    195 ;
0000    196 ; EQUATED SYMBOLS:
0000    197 ;                                                                        ;G:4
0000    198          $PrDef                                  ; Define PR codes      [34]
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    199          $DS_ENVDEF
0000    200          $DS_DSADEF
0000    201          $DS_HDRDEF
0000    202          APTDEF
0000    203          CLIDEF                          ; Cli$L_ definitions           [31]
0000    204          DSFDEF
0000    205          DSPDEF
0000    206          DSQA                            ; DSQA$K constant definitions  [22]
0000    207          DS_QADEFS                       ; Other QA$K constant definitions [25]
0000    208          $DS_TYPEDEF                     ; Define type codes            [24]
```

```
                0000          .SUBTITLE       Work Declarations
            00000000          .PSECT  Work, NoExe, NoShr, Wrt, Long          ;                    [22]
                0000
                0000  ;
                0000  ; OWN STORAGE:
                0000  ;
                0000
        00000004 0000  A_TESTPTR:      .BLKL   1       ; POINTER TO CURRENT TEST CODE.
        00000008 0004  A_DATAPTR:      .BLKL   1       ; POINTER TO CURRENT TEST DATA.
        0000000C 0008  REQ_SUB:        .BLKL   1       ; SUBT. # THAT IS GREATER THAN SUBT. THAT EXIST
                000C
```

```
                                    000C   222          .SUBTITLE       Data Declarations
                                00000000   223          .PSECT  Data, NoExe, Shr, NoWrt, Long          ;                    [22]
                                    0000   224
                                    0000   225          MODNAM  DISPATCH
        48 43 54 41 50 53 49 44 00' 0000       $MODULE:        .ASCIC  \DISPATCH\
                                08  0000
                                    0009   226
                                    0009   227 T_NOSUB:
20 3A 47 4E 49 4E 52 41 57 2F 21 00' 0009   228          .ASCIC \!/WARNING: Specified subtest number (!UW) too large.  !/\-      ;[38
75 73 20 64 65 69 66 69 63 65 70 53  0015
72 65 62 6D 75 6E 20 74 73 65 74 62  0021
6C 20 6F 6F 74 20 29 57 55 21 28 20  002D
57 55 21 2F 21 20 20 2E 65 67 72 61  0039
6E 69 20 73 74 73 65 74 62 75 73 20  0045
65 77 20 57 55 21 20 74 73 65 74 20  0051
2E 64 65 74 75 63 65 78 65 20 65 72  005D
                              2F 21  0069
                                61  0009
                                    006B   229                  \!UW subtests in test !UW were executed.!/\                  ;[38
                                    006B   230 T_NOEXE:
73 65 74 20 6F 4E 20 3F 3F 2F 21 00' 006B   231          .ASCIC  \!/?? No tests in this section.!/\      ;                    [24]
73 20 73 69 68 74 20 6E 69 20 73 74  0077
        2F 21 2E 6E 6F 69 74 63 65  0083
                                20  006B
                                    008C   232
                                    008C   233 FMTPASONE:
20 74 73 72 69 46 20 2E 2E 2F 21 00' 008C   234          .ASCIC  "!/.. First pass done, !UW error!%S detected, time is !%D!/" ;  [30]
21 20 2C 65 6E 6F 64 20 73 73 61 70  0098
20 53 25 21 72 6F 72 72 65 20 57 55  00A4
69 74 20 2C 64 65 74 63 65 74 65 64  00B0
   2F 21 44 25 21 20 73 69 20 65 6D  00BC
                                3A  008C
                                    00C7   235
                                    00C7   236 FMTLSTPAS:
66 6F 20 64 6E 45 20 2E 2E 2F 21 00' 00C7   237          .ASCIC  "!/.. End of run, !UW error!%S detected, pass count is !UL,!/"-;[30]
72 65 20 57 55 21 20 2C 6E 75 72 20  00D3
63 65 74 65 64 20 53 25 21 72 6F 72  00DF
6F 63 20 73 73 61 70 20 2C 64 65 74  00EB
21 2C 4C 55 21 20 73 69 20 74 6E 75  00F7
                                2F  0103
21 20 73 69 20 65 6D 69 74 20 20 20  0104   238                  "   time is !%D!/"                  ;                    [30]
                        2F 21 44 25  0110
                                4C  00C7
```

```
                                                0114   240 FMTSEARCH:
20 72 6F 72 72 45 20 2E 2E 2F 21 00'            0114   241         .ASCIC  "!/.. Error search: new test limit is !UL, time is !%D!/" ;      [30]
20 77 65 6E 20 3A 68 63 72 61 65 73             0120
69 20 74 69 6D 69 6C 20 74 73 65 74             012C
20 65 6D 69 74 20 2C 4C 55 21 20 73             0138
            2F 21 44 25 21 20 73 69             0144
                                  37            0114
                                                014C   242
                                                014C   243 FMTTRACE:
3A 42 55 21 20 74 73 65 54 2F 21 00'            014C   244         .ASCIC  "!/Test !UB: !AC"
                        43 41 21 20             0158
                                  0F            014C
                                                015C   245
                                                015C   246 FMT_SUMMARY:                                  ;                        [28]
6F 20 79 72 61 6D 6D 75 53 2F 21 00'            015C   247         .ascic  "!/Summary of !AC, Rev !UL.!UL:!/" -    ;                        [28]
21 20 76 65 52 20 2C 43 41 21 20 66             0168
         2F 21 3A 4C 55 21 2E 4C 55             0174
6D 61 72 67 6F 72 70 20 4C 5A 21 20             017D   248                 " !ZL program detected error!%S " -     ;                        [28]
72 65 20 64 65 74 63 65 74 65 64 20             0189
               20 53 25 21 72 6F 72             0195
21 20 2C 64 72 61 48 20 4C 5A 21 28             019C   249                 "(!ZL Hard, !ZL Soft, !ZL" -            ;                        [28]
4C 5A 21 20 2C 74 66 6F 53 20 4C 5A             01A8
4C 5A 21 20 2C 6D 65 74 73 79 53 20             01B4   250                 " System, !ZL Device).!/" -             ;                        [28]
   2F 21 2E 29 65 63 69 76 65 44 20             01C0
69 76 72 65 70 75 53 20 4C 5A 21 20             01CB   251                 " !ZL Supervisor detected error!%S.!/!/"        ;                [28]
64 65 74 63 65 74 65 64 20 72 6F 73             01D7
2F 21 2E 53 25 21 72 6F 72 72 65 20             01E3
                              2F 21             01EF
                                  94            015C
```

```
                                        01F1   253 Fmt_ShowStatus_No_PC:                                  ;                    [37]
21 20 66 6F 20 73 75 74 61 74 53 00'    01F1   254         .Ascic  "Status of !AC, Rev !UL.!UL!/" -        ;                    [37]
2E 4C 55 21 20 76 65 52 20 2C 43 41     01FD
                      2F 21 4C 55 21    0209
69 74 63 65 53 5B 5F 21 44 25 21 20     020E   255                 " !%D!_[Section !AC]!_[!ZL error!%S]!/" -       ;          [37]
5A 21 5B 5F 21 5D 43 41 21 20 6E 6F     021A
21 5D 53 25 21 72 6F 72 72 65 20 4C     0226
                                  2F    0232
21 5D 4C 5A 21 20 74 73 65 54 5B 20     0233   256                 " [Test !ZL]!_[Subtest !ZL]!/" -        ;                    [37]
5A 21 20 74 73 65 74 62 75 53 5B 5F     023F
                         2F 21 5D 4C    024B
21 5D 4C 5A 21 20 73 73 61 50 5B 20     024F   257                 " [Pass !ZL]!/"                         ;                    [37]
                                  2F    025B
                                  6A    01F1
                                        025C   258
                                        025C   259 FMT_SHOWSTATUS:
21 20 66 6F 20 73 75 74 61 74 53 00'    025C   260         .ascic  "Status of !AC, Rev !UL.!UL!/" -        ;                    [23]
2E 4C 55 21 20 76 65 52 20 2C 43 41     0268
                      2F 21 4C 55 21    0274
69 74 63 65 53 5B 5F 21 44 25 21 20     0279   261                 " !%D!_[Section !AC]!_[!ZL error!%S]!/" -       ;          [23]
5A 21 5B 5F 21 5D 43 41 21 20 6E 6F     0285
21 5D 53 25 21 72 6F 72 72 65 20 4C     0291
                                  2F    029D
21 5D 4C 5A 21 20 74 73 65 54 5B 20     029E   262                 " [Test !ZL]!_[Subtest !ZL]!/" -        ;                    [23]
5A 21 20 74 73 65 74 62 75 53 5B 5F     02AA
                         2F 21 5D 4C    02B6
21 5D 4C 5A 21 20 73 73 61 50 5B 20     02BA   263                 " [Pass !ZL]!_[User PC !XL]!/"  ;                            [23]
58 21 20 43 50 20 72 65 73 55 5B 5F     02C6
                         2F 21 5D 4C    02D2
                                  79    025C
                                        02D6   264
                                        02D6   265 FMT_NODIAG:
6E 67 61 69 64 20 6F 4E 20 3F 3F 00'    02D6   266         .ascic  '?? No diagnostic running.!/'   ;                            [23]
6E 69 6E 6E 75 72 20 63 69 74 73 6F     02E2
                         2F 21 2E 67    02EE
                                  1B    02D6
                                        02F2   267
                                        02F2   268 T_TSTABO:
   5D 64 65 74 72 6F 62 61 5B 20 00'    02F2   269         .ASCIC  " [aborted]"
                                  0A    02F2
```

```
                    02FD   271          .SBTTL  Dispatch Routine
                00000000   272          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                [22]
                    0000   273 ;++
                    0000   274 ; FUNCTIONAL DESCRIPTION:
                    0000   275 ;
                    0000   276 ;       This routine is called to initiate the test program.
                    0000   277 ;       It takes case of calling initialization sections, individual
                    0000   278 ;       tests and finally cleanup code.
                    0000   279 ;
                    0000   280 ; CALLING SEQUENCE:
                    0000   281 ;
                    0000   282 ;       BSBW    DISPATCH
                    0000   283 ;
                    0000   284 ; INPUT PARAMETERS:
                    0000   285 ;
                    0000   286 ;       DS$GL_FSTTEST   = FIRST TEST NUMBER
                    0000   287 ;       DS$GL_LSTTEST   = NUMBER OF LAST TEST TO EXECUTE
                    0000   288 ;
                    0000   289 ; IMPLICIT INPUTS:      NONE
                    0000   290 ;
                    0000   291 ; OUTPUT PARAMETERS:    NONE
                    0000   292 ;
                    0000   293 ; IMPLICIT OUTPUTS:     NONE
                    0000   294 ;
                    0000   295 ; COMPLETION CODES:     NONE
                    0000   296 ;
                    0000   297 ; SIDE EFFECTS:         NONE
                    0000   298 ;--
```

```
                                  0000   300 DISPATCH::
          00000000'EF    D4       0000   301         CLRL    L^DS$GL_ERRCNT          ; START WITH NO ERRORS
          00000000'EF    D4       0006   302         Clrl    L^Ds$GL_HardErr_Count   ; Clear out error counters          [28]
          00000000'EF    D4       000C   303         Clrl    L^Ds$GL_SoftErr_Count   ;                                   [28]
          00000000'EF    D4       0012   304         Clrl    L^Ds$GL_SysErr_Count    ;                                   [28]
          00000000'EF    D4       0018   305         Clrl    L^Ds$GL_DevErr_Count    ;                                   [28]
          00000000'EF    D4       001E   306         Clrl    L^Ds$GL_ErrSup_Count    ;                                   [28]
          0000FE54'EF    D4       0024   307         CLRL    L^DSA$GL_PASSNO         ; START ON FIRST PASS
                                  002A   308         Set_Pass0                       ; Indicate first pass ever          [32]
     0000FE00'EF    1D    E2       002A               BBSS    #DSA$V_Pass0, -         ; Branch if PASS0 bit is set to     [29]
                    00             0031                         L^DSA$GL_Flags, 30000$        ; ...
                                  0032       30000$:                                 ; ... here. Set bit regardless.     [29]
                                  0032   309
                                  0032   310 FIRSTTEST:
                                  0032   311
                                  0032   312         ;+
                                  0032   313         ; Perform the initialization code for the diagnostic.
                                  0032   314         ; Test 0, Subtest 1 => initialization code
                                  0032   315         ; Test 0, Subtest 2 => cleanup code
                                  0032   316         ;-
                                  0032   317
          0000FE50'EF    D4       0032   318         CLRL    L^DSA$GL_TESTNO         ; TEST 0,
     0000FE4C'EF    01    D0       0038   319         MOVL    #1, L^DSA$GL_SUBTNO     ; SUBTEST 1 INDICATES INIT.
                                  003F   320
                                  003F   321         QA_MAIN Dispat_Before_Init      ; Call QA$Main                      [22]
                    13    DD       003F               PUSHL   #DSQA$K_Dispat_Before_Init
     00000000'9F    01    FB       0041               CALLS   #1, @#QA$MAIN
                                  0048   322
     0000023C'FF    00    FB       0048   323         CALLS   #0, @L$A_ICP            ; Execute initialization code.
0000FE50'EF   00000000'EF   D0       004F   324         MOVL    L^DS$GL_FSTTEST, -      ; LOAD NUMBER OF FIRST TEST
                                  005A   325                         L^DSA$GL_TESTNO
                                  005A   326
                                  005A   327         QA_MAIN Dispat_After_Init       ; Call QA$Main                      [22]
                    14    DD       005A               PUSHL   #DSQA$K_Dispat_After_Init
     00000000'9F    01    FB       005C               CALLS   #1, @#QA$MAIN
                                  0063   328
          0000FE4C'EF    D4       0063   329         CLRL    L^DSA$GL_SUBTNO         ; RESET SUBTEST NUMBER.
     0000FE00'EF    1D    E5       0069   330         BBCC    #DSA$V_PASS0, -         ; Clear first pass flag.
                    07             0070   331                         L^DSA$GL_FLAGS, CALLTEST
     0000FE54'EF    01    D0       0071   332         MOVL    #1, L^DSA$GL_PASSNO     ; START ON FIRST PASS
```

```
                                  0078   334 CALLTEST:
                                  0078   335
                                  0078   336          ;+
                                  0078   337          ; Set up everything to call a test in the diagnostic.
                                  0078   338          ;-
                                  0078   339
                                  0078   340          QA_MAIN Dispat_CallTest          ; Call QA$Main                          [22]
                        15   DD   0078                PUSHL   #DSQA$K_Dispat_CallTest
        00000000'9F     01   FB   007A                CALLS   #1, @#QA$MAIN
                                  0081   341          Clear_ErrFlg                     ; Clear the error flag                  [32]
        00000000'EF     04   E4   0081                BBSC    #DS$V_ErrFlg, -          ; Branch if ERRFLG bit is set to        [29]
                        00        0088                          L^DS$GL_Flags, 30001$ ; ...                                   [29]
                                  0089       30001$:                                   ; ... here. Clear bit regardless.       [29]
                                  0089   342
                                  0089   343          ;+
                                  0089   344          ; Compute the starting address of the next test to execute.
                                  0089   345          ; Use the DSA$GL_TESTNO to compute the address. If no more tests,
                                  0089   346          ; go to 130$ label.
                                  0089   347          ;-
                                  0089   348
  5C    0000FE50'EF     01   C3   0089   349          SUBL3   #1, L^DSA$GL_TESTNO, AP          ; GET INDEX INTO DISPATCH TBL
                  5C    18   C4   0091   350          MULL2   #DSP$K_SIZE, AP
     5C   00000218'FF4C   9E   0094   351          MOVAB   @L$A_DTP[AP], AP                 ; GET ADDRESS OF DISP TBL ENTRY.
   00000000'EF    0C AC   D0   009C   352          MOVL    DSP$A_TEST(AP), -                ; GET TEST CODE POINTER.
                                  00A4   353                  L^A_TESTPTR
                        03   12   00A4   354          BNEQ    20$                              ; If not equal to zero, branch
                      01DF   31   00A6   355          BRW     130$                             ; Was equal to zero
                                  00A9   356
   00 BC    0000FE50'EF     91   00A9   357 20$:     CMPB    L^DSA$GL_TESTNO, -               ; CHECK TEST NUMBER.
                                  00B1   358                  @DSP$A_BASE(AP)
                        14   13   00B1   359          BEQL    30$                              ; If they are equal, branch
                                  00B3   360          ERRSUP_S                                 ; Otherwise, error in DS
            00000000'EF     DF   00B3                         PUSHAL  $MODULE
                        00   DD   00B9                         PUSHL   #0
                        01   DD   00BB                         PUSHL   #$ER
        00000000'EF     03   FB   00BD                         CALLS   #3, DS_ERRSUP
                      01F0   31   00C4   361          BRW     DISPATCH_X
                                  00C7   362
 03 10 AC    0000FE10'EF     E0   00C7   363 30$:     BBS     L^DSA$GL_SECTNO, -               ; Check for right section
                                  00D0   364                  DSP$Q_SECTION(AP), 35$
                      010A   31   00D0   365          BRW     115$                             ; Branch if bit clear
```

```
                                  00D3          ;+
                                  00D3          ; If TRACE flag is set, print the trace message.
                                  00D3          ;-
                                  00D3
                                  00D3  35$:    Br_If_Not_Trace 40$                     ; If not tracing, branch        [32]
        0000FE00'EF    0A    E1   00D3          BBC     #DSA$V_Trace, -                 ; If bit not set, branch        [29]
                       19         00DA                    L^DSA$GL_Flags, 40$           ;                               [29]
                                  00DB          $PRINT  #ds$k_type_program_info,-       ; Print info message            [24]
                                  00DB                    #ds$k_printf, -               ; .. use PRINTF                 [24]
                                  00DB                    L^FMTTRACE, -                 ; .. Trace message              [24]
                                  00DB                    @DSP$A_BASE(AP), -            ; .. test number                [24]
                                  00DB                    DSP$A_MSG(AP)                 ; .. and test name              [24]
                    04 AC    DD   00DB                  PUSHL   DSP$A_MSG(AP)
                    00 BC    DD   00DE                  PUSHL   @DSP$A_BASE(AP)
              0000014C'EF    9F   00E1                  PUSHAB  L^FMTTRACE
                   7E   01    9A   00E7                  movzbl  #ds$k_printf, -(sp)
                   7E   17    98   00EA                  cvtbl   #ds$k_type_program_info, -(sp)
        00000000'GF    05    FB   00ED                  CALLS   #$$N+2, G^DSX$PRINT
                                  00F4
     00000004'EF    08 AC    D0   00F4  40$:    MOVL    DSP$A_DATA(AP), -               ; GET DATA TABLE ENTRY.
                                  00FC                    L^A_DATAPTR
                                  00FC
              00000000'EF    D4   00FC  50$:    CLRL    L^DS$GA_CHKLPPC                 ; CLEAR THE CHECK_LOOP P.C.
              0000FE4C'EF    D4   0102          CLRL    L^DSA$GL_SUBTNO                 ; START WITH SUBTEST ZERO
                                  0108          Clear_Subt                              ; Clear the subtest flag        [32]
        00000000'EF    0E    E4   0108          BBSC    #DS$V_Subt, -           ; Branch if SUBT bit is set to          [29]
                       00         010F                    L^DS$GL_Flags, 30002$ ; ...                                   [29]
                                  0110      30002$:                             ; ... here. Clear bit regardless.       [29]
                                  0110
                                  0110          ;+
                                  0110          ; Call the test that is numbered DSA$GL_TESTNO.
                                  0110          ;-
                                  0110
 00000000'FF   00000004'FF    FA   0110          CALLG   @L^A_DATAPTR, -                 ; CALL THE TEST.
                                  011B                  @L^A_TESTPTR
              0000FE50'EF    D1   011B          CMPL    L^DSA$GL_TESTNO,-               ; If test = last save the subtest #
              00000000'EF         0121                    L^DS$GL_LSTTEST
                       0B    12   0126          BNEQ    60$                             ; Don't save subtest #
              0000FE4C'EF    D0   0128          MOVL    L^DSA$GL_SUBTNO,-               ; Keep number of subt. saved
              00000008'EF         012E                    L^REQ_SUB
                                  0133          ;+
                                  0133          ; The test will come back to here assuming there were
                                  0133          ; no strange escapes from the test.
                                  0133          ;-
                                  0133
                                  0133  60$:    Set_ExeTst                              ; Set execute flag              [32]
        00000000'EF    12    E2   0133          BBSS    #DS$V_ExeTst, -         ; Branch if EXETST bit is set to        [29]
                       00         013A                    L^DS$GL_Flags, 30003$ ; ...                                   [29]
                                  013B      30003$:                             ; ... here. Set bit regardless.         [29]
                                  013B          Br_If_Not_ErrFlg        100$            ; Branch if no errors           [32]
        00000000'EF    04    E1   013B          BBC     #DS$V_ErrFlg, -                 ; If bit not set, branch        [29]
                       5A         0142                    L^DS$GL_Flags, 100$           ;                               [29]
                                  0143          Br_If_Not_Search        100$            ; Branch if not searching       [32]
        0000FE00'EF    0E    E1   0143          BBC     #DSA$V_Search, -                ; If bit not set, branch        [29]
                       52         014A                    L^DSA$GL_Flags, 100$          ;                               [29]
```

```
                                014B   405          ;+
                                014B   406          ; This section is for performing SEARCHing using the search flag.
                                014B   407          ;-
                                014B   408
                                014B   409          Clear_ErrFlg                            ; Clear the error flag          [32]
          00000000'EF   04   E4 014B                BBSC    #DS$V_ErrFlg, -                 ; Branch if ERRFLG bit is set to [29]
                        00      0152                  L^DS$GL_Flags, 30004$ ; ...           [29]
                                0153          30004$:                                     ; ... here. Clear bit regardless. [29]
 00000000'EF   0000FE50'EF   D0 0153   410          MOVL    L^DSA$GL_TESTNO,L^DS$GL_LSTTEST ; Copy test number to limit
 FFFFFFFF 8F   00000000'EF   F1 015E   411          ACBL    L^DS$GL_FSTTEST,#-1, -          ; Decrement last test, and be
               00000000'EF      0169
                        0012    016E   412                    L^DS$GL_LSTTEST,80$          ;   sure it's in range--else bad
 0000FE08'EF   0000FE54'EF   D0 0170   413          MOVL    L^DSA$GL_PASSNO,L^DSA$GL_PASSES ; Pretend we just did last pass
          000002B8'EF   00   FB 017B   414          CALLS   #0,L^DSX$ENDPASS                ; Now 'fake' an EOP to stop testing
                                0182   415                                                  ; DSX$ENDPASS will not return here!
                                0182   416
                                0182   417 80$:     $PRINT  #ds$k_type_program_info, -      ; Print search message          [24]
                                0182   418                    #ds$k_printf, -              ; .. use PRINTF                 [24]
                                0182   419                    L^FMTSEARCH, -               ; .. SEARCH message             [24]
                                0182   420                    L^DS$GL_LSTTEST, -           ; .. test number                [24]
                                0182   421                    #0                           ; .. and current time           [24]
                        00      0182                  PUSHL   #0
               00000000'EF      0184                  PUSHL   L^DS$GL_LSTTEST
               00000114'EF      018A                  PUSHAB  L^FMTSEARCH
                     7E   01   0190                  movzbl  #ds$k_printf, -(sp)
                     7E   17   0193                  cvtbl   #ds$k_type_program_info, -(sp)
          00000000'GF   05   FB 0196                  CALLS   #$$N+2, G^DSX$PRINT
```

```
                          019D   423          ;+
                          019D   424          ; If the test did an ABORT, R0 will be clear. If it is clear and the
                          019D   425          ; trace flag is set, print a message saying that the test was aborted
                          019D   426          ; and go on with the next test in the sequence.
                          019D   427          ;-
                          019D   428
                    1D 50 E8  019D   429 100$:   BLBS    R0, 110$                        ; If success, branch
                          01A0   430           Br_If_Not_Trace 115$                    ; If not Tracing, branch around [32]
          0000FE00'EF   0A E1  01A0             BBC     #DSA$V_Trace, -                 ; If bit not set, branch        [29]
                       35      01A7               L^DSA$GL_Flags, 115$                 ;                               [29]
                          01A8   431           $print  #ds$k_type_abort_test,-         ; Print abort test message      [24]
                          01A8   432                   #ds$k_printi, -                 ; .. use PRINTI (was TYPMSG)    [24]
                          01A8   433                   L^T_TSTABO                      ; .. ABORT text                 [24]
              000002F2'EF   9F  01A8                   PUSHAB  L^T_TSTABO
                    7E   00 9A  01AE                   movzbl  #ds$k_printi, -(sp)
                    7E   13 98  01B1                   cvtbl   #ds$k_type_abort_test, -(sp)
          00000000'GF   03 FB  01B4                   CALLS   #$$N+2, G^DSX$PRINT
                       20 11   01BB   434           BRB     115$
                          01BD   435
                          01BD   436          ;+
                          01BD   437          ; Check to see if the same test should be called again with
                          01BD   438          ; another data argument list. If so, compute address of new
                          01BD   439          ; data and go back to call the same test again.
                          01BD   440          ;-
                          01BD   441
          50   00000004'EF  D0  01BD   442 110$:   MOVL    L^A_DATAPTR, R0                 ; GET POINTER TO ARG LIST
                    51   60 D0  01C4   443           MOVL    (R0),R1                     ; GET COUNT OF ARGS IN LIST
                       14 13   01C7   444           BEQL    115$                        ; END OF DATA LIST, DO NEXT TEST
     00000004'EF   04 A041 DE  01C9   445           MOVAL   4(R0)[R1], L^A_DATAPTR      ; POINT PAST ARG LIST
                00000004'FF  D5  01D2   446           TSTL    @L^A_DATAPTR                ; END OF LIST OF ARG LISTS?
                       03 13   01D8   447           BEQL    115$                        ; Yes--next test
                    FF1F 31    01DA   448           BRW     50$                         ; Repeat test with next arg list
```

```
                                  01DD          ;+                                                                  [25]
                                  01DD          ; Check to see if QA is running AND if several cases of check      [25]
                                  01DD          ; routines are currently executing. If both are true, perform special [25]
                                  01DD          ; code to accomplish the checks.                                    [25]
                                  01DD          ;-
                                  01DD
                                  01DD  115$:   Br_If_QA        117$                    ; If running QA, branch      [32]
          0000FE00'EF   0F   E0   01DD          BBS     #DSA$V_QA, -                    ; If bit set, branch         [29]
                        03        01E4                    L^DSA$GL_Flags, 117$          ;                            [29]
                      0092   31   01E5          BRW     125$                            ; QA is not set, branch around. [31]
                                  01E8
          00000000'EF   03   E1   01E8  117$:   BBC     S^#QA$K_Loop_On_Subtest, -      ; If the Loop on Subtest check [31]
                        5B        01EF                    L^QA$AOB_Check_State, 122$    ; is not running, branch.    [31]
                                  01F0
                                  01F0          ;+
                                  01F0          ; This is for the Loop on Subtest check only.                       [31]
                                  01F0          ; We have finished running one test and all its subtests. Reset the [31]
                                  01F0          ; subtest number, and add one to both the first and last test numbers. [31]
                                  01F0          ; Make these changes in both the CLI data vector and the DS (DSA)   [31]
                                  01F0          ; data areas.                                                       [31]
                                  01F0          ;-
                                  01F0
          51    0000FE50'EF   D0   01F0          MOVL    L^DSA$GL_TestNo, R1             ; Get the current test number. [31]
          50    0000FE08'EF   D0   01F7          MOVL    L^DSA$GL_PASSES, R0             ; Save the Passes count.      [31]
          0000FE08'EF   01   D0   01FE          MOVL    #1, L^DSA$GL_PASSES             ; Set to 1 in case we take the [31]
                                  0205                                                  ; branch to 130$.             [31]
  0002 51 01    00000000'EF   3D   0205          ACBW    L^QA$W_Save_Last, #1, R1, 121$  ; Branch if (R1 + 1) LEQ the [31]
                                  020F                                                  ; saved last test number.     [31]
                        77   11   020F          BRB     130$                            ; Otherwise, done all the tests.[31]
                                  0211
          0000FE08'EF   50   D0   0211  121$:   MOVL    R0, L^DSA$GL_PASSES             ; Restore Passes count.       [31]
          50    00000000'EF   DE   0218          MOVAL   L^DS$GL_CLIBASE, R0             ; Address the CLI data vector. [31]
          00000028 E0   01   D0   021F          MOVL    #1, L^CLI$L_SUBT (R0)           ; Set the Subtest number to 1. [31]
          00000000'EF   01   D0   0226          MOVL    #1, L^DS$GL_SUBTEST             ; Ditto.                      [31]
                                  022D
          00000020 E0   51   D0   022D          MOVL    R1, L^CLI$L_TEST (R0)           ; Increment First test.       [31]
          00000000'EF   51   D0   0234          MOVL    R1, L^DS$GL_FSTTEST             ; Ditto.                      [31]
          00000024 E0   51   D0   023B          MOVL    R1, L^CLI$L_LAST (R0)           ; Increment Last test.        [31]
          00000000'EF   51   D0   0242          MOVL    R1, L^DS$GL_LSTTEST             ; Ditto.                      [31]
                        2F   11   0249          BRB     125$                            ; Go execute another test.    [31]
```

```
                00000000'EF   02   E1  024B   489 122$:   BBC     S^#QA$K_Loop_On_Test, -          ; If the Loop on Test check is  [25]
                              0B       0252   490             L^QA$AOB_Check_State, 123$       ; not running, branch           [25]
                00000000'EF   04   E0  0253   491         BBS     S^#QA$K_Loop_Test_Done, -        ; If done looping on this test, [25]
                              1F       025A   492             L^QA$AOB_Flags, 125$             ; branch                        [25]
                           FE1A   31   025B   493         BRW     CallTest                         ; Not done looping => run again [25]
                                       025E   494
                00000000'EF   04   E1  025E   495 123$:   BBC     S^#QA$K_Run_Backwards, -         ; If the Run Backwards check is [25]
                              14       0265   496                L^QA$AOB_Check_State, 125$    ; not running, branch           [25]
                                       0266   497
    FFFFFFFF 8F   00000000'EF   F1  0266   498 124$:   ACBL    L^DS$GL_FSTTEST,   -             ; Subtract one from DSA$GL_TESTNO,[2
                  0000FE50'EF      0271
                                   0276   499               #-1,               -             ; compare that to DS$GL_FSTTEST,[25]
                           FE00    0276   500               L^DSA$GL_TESTNO, -               ; if the former is greater than [25]
                                   0278   501               CALLTEST                         ; or equal to the latter, branch.[25
                             0E    11  0278   502         BRB     130$                             ; Otherwise, we are done Run    [25]
                                       027A   503                                          ; Backwards check.              [25]
                                       027A   504         ;+                                                           [25]
                                       027A   505         ; This is the normal way to advance through the tests.       [25]
                                       027A   506         ; Add one to the DSA$GL_TESTNO, if it is less than or equal to the  [25]
                                       027A   507         ; DS$GL_LSTTEST, branch back to CALLTEST and call the next test.    [25]
                                       027A   508         ;-                                                           [25]
                                       027A   509
0000FE50'EF   01   00000000'EF   F1  027A   510 125$:   ACBL    L^DS$GL_LSTTEST, #1, -
                           FDF0    0286   511               L^DSA$GL_TESTNO, CALLTEST        ; Do next test until reaches LSTTEST
```

```
                         0288   513          ;+
                         0288   514          ; Come to here if there are no more tests to execute in the sequence.
                         0288   515          ; Go back to FIRSTTEST and perform the initialization code again (and
                         0288   516          ; perhaps call DSX$ENDPASS in the init code).
                         0288   517          ;-
                         0288   518
                         0288   519 130$:    Br_If_Not_ExeTst        140$              ; Branch if no test executed.   [32]
00000000'EF   12   E1    0288               BBC     #DS$V_ExeTst, -                   ; If bit not set, branch        [29]
                14       028F                 L^DS$GL_Flags, 140$                     ;                               [29]
                         0290   520          Clear_ExeTst                             ; Clear the execute flag        [32]
00000000'EF   12   E4    0290               BBSC    #DS$V_ExeTst, -           ; Branch if EXETST bit is set to      [29]
                00       0297                 L^DS$GL_Flags, 30005$ ; ...             [29]
                         0298       30005$:                         ; ... here. Clear bit regardless.             [29]
                         0298   521          QA_MAIN Dispat_Done_Tests                ; Call QA$Main.                 [31]
                18   DD  0298               PUSHL   #DSQA$K_Dispat_Done_Tests
00000000'9F     01   FB  029A               CALLS   #1, a#QA$MAIN
              FD8E   31  02A1   522          BRW     FIRSTTEST                        ; GO RUN ANOTHER PASS
                         02A4   523
                         02A4   524          ;+
                         02A4   525          ; If there were no tests executed in this section, come to here.
                         02A4   526          ;-
                         02A4   527
                         02A4   528 140$:    $print  #ds$k_type_no_tests, -           ; No tests in section           [24]
                         02A4   529                    #ds$k_printi, -               ; .. use PRINTI (was TYPMSG)    [24]
                         02A4   530                    L^T_NOEXE                     ; .. 'no tests in section'      [24]
      0000006B'EF   9F   02A4                       PUSHAB  L^T_NOEXE
            7E   00   9A   02AA                       movzbl  #ds$k_printi, -(sp)
            7E   12   98   02AD                       cvtbl   #ds$k_type_no_tests, -(sp)
00000000'GF     03   FB   02B0                       CALLS   #$$N+2, G^DSX$PRINT
                         02B7   531
                         02B7   532 DISPATCH_X:
                   05    02B7   533          RSB                                      ; RETURN TO COMMAND MODE
```

```
                        02B8    535          .SBTTL  DSX$EndPass Routine
                        02B8    536 ;++
                        02B8    537 ; FUNCTIONAL DESCRIPTION:
                        02B8    538 ;
                        02B8    539 ;       This routine is called to mark the end of a 'PASS'.
                        02B8    540 ;       If enough passes have been run execute user summary and cleanup code.
                        02B8    541 ;
                        02B8    542 ; CALLING SEQUENCE:
                        02B8    543 ;
                        02B8    544 ;       CALLx   #0,a#DSX$ENDPASS
                        02B8    545 ;
                        02B8    546 ; INPUT PARAMETERS:     NONE
                        02B8    547 ;
                        02B8    548 ; IMPLICIT INPUTS:      NONE
                        02B8    549 ;
                        02B8    550 ; OUTPUT PARAMETERS:    NONE
                        02B8    551 ;
                        02B8    552 ; IMPLICIT OUTPUTS:     NONE                                                    ;G:4
                        02B8    553 ;
                        02B8    554 ; COMPLETION CODES:     NONE
                        02B8    555 ;
                        02B8    556 ; SIDE EFFECTS:         NONE
                        02B8    557 ;--
```

```
                     0004  02B8   559 .ENTRY  DSX$ENDPASS,^M<R2>
                           02BA   560
                           02BA   561         QA_MAIN Dispat_DSX$EndPass_Bgn  ; Call QA$Main                          [22]
                       01   DD  02BA                PUSHL   #DSQA$K_Dispat_DSX$EndPass_Bgn
          00000000'9F  01   FB  02BC                CALLS   #1, a#QA$MAIN
                           02C3   562
               0000FE08'EF  D5  02C3   563         TSTL    L^DSA$GL_PASSES         ; If passes = 0, => run for infinity
                       03  12  02C9   564         BNEQ    1$                      ; If = 0, branch                 [38]
                     00CA  31  02CB   565         BRW     10$                     ; extended branch                [38]
                           02CE   566
    0000FE54'EF 0000FE08'EF  D1  02CE   567 1$:     CMPL    L^DSA$GL_PASSES, -      ; Compare passes and passno
                           02D9   568                   L^DSA$GL_PASSNO
                       03  1B  02D9   569         BLEQU   2$                      ; If passes > passno, branch     [38]
                     00BA  31  02DB   570         BRW     10$                     ; extended branch                [38]
                           02DE   571
                           02DE   572                                             ; If passno >= passes, continue on [29]
                           02DE   573 2$:     Clear_CtrlO                         ; Clear ^O flag                  [32]
          00000000'EF  10   E4  02DE                BBSC    #DS$V_CtrlO, -          ; Branch if CTRLO bit is set to
                       00       02E5                   L^DS$GL_Flags, 30006$ ; ...
                           02E6       30006$:                                     ; ... here. Clear bit regardless.
                           02E6   574         $DS_SUMMARY_S                       ; Execute diagnostics summary section
          00000000'9F  00   FB  02E6                CALLS   #0, a#DS$SUMMARY
               000003D2'EF  16  02ED   575         JSB     DS_CLEANUP              ; Execute diagnostics cleanup section
                           02F3   576         Clear_CtrlO                         ; Clear ^O flag                  [32]
          00000000'EF  10   E4  02F3                BBSC    #DS$V_CtrlO, -          ; Branch if CTRLO bit is set to
                       00       02FA                   L^DS$GL_Flags, 30007$ ; ...
                           02FB       30007$:                                     ; ... here. Clear bit regardless.
               00000000'EF  16  02FB   577         JSB     INIT_CONTEXT            ; Reset stacks and processor mode [26]
                           0301   578
               00000000'EF  D5  0301   579         TSTL    L^DS$GL_SUBTEST         ; If subt. = 0 no size error    [38]
                           42  13  0307   580         BEQL    4$                      ;                                [38]
                           0309   581
          51   00000254'EF  D0  0309   582         MOVL    L^L$A_TSTCNT,R1         ; If last test in diagnostic..   [41]
          00000000'EF  61   D1  0310   583         CMPL    (R1),L^DS$GL_LSTTEST    ; see if subtests exist          [41]
                           0D  12  0317   584         BNEQ    3$                      ; print as is if not last test   [41]
    00000000'EF 0000FE4C'EF  D1  0319   585         CMPL    L^DSA$GL_SUBTNO,L^DS$GL_SUBTEST; mov correct tests executed into pri
                           25  1E  0324   586         BGEQU   4$                      ; don't print if 0 executed      [41]
                           0326   587         ;MOVL   L^DSA$GL_SUBTNO,L^REQ_SUB;      [41]
                           0326   588                                             ; commented out this incorrect move instruct
                           0326   589
                           0326   590 3$:     $PRINT  #ds$k_type_program_end,-; Print 'error on subtest #'      [38]
                           0326   591                   #ds$k_printf, -         ; .. use PRINTF                  [38]
                           0326   592                   L^T_NOSUB,-             ; .. format text                 [38]
                           0326   593                   L^DS$GL_SUBTEST,-       ; .. number specified            [38]
                           0326   594                   L^REQ_SUB,-             ; .. number of subtests present  [38]
                           0326   595                   L^DS$GL_LSTTEST         ; .. last test executed          [38]
               00000000'EF  DD  0326                PUSHL   L^DS$GL_LSTTEST
               00000008'EF  DD  032C                PUSHL   L^REQ_SUB
               00000000'EF  DD  0332                PUSHL   L^DS$GL_SUBTEST
               00000009'EF  9F  0338                PUSHAB  L^T_NOSUB
                      7E  01  9A  033E                movzbl  #ds$k_printf, -(sp)
                      7E  10  98  0341                cvtbl   #ds$k_type_program_end, -(sp)
          00000000'GF  06   FB  0344                CALLS   #$$N+2, G^DSX$PRINT
                           034B   596
                           034B   597 4$:     $PRINT  #ds$k_type_program_end,-; Print 'run finished'           [24]
                           034B   598                   #ds$k_printf, -         ; .. use PRINTF                  [24]
                           034B   599                   L^FMTLSTPAS, -          ; .. format text                 [24]
```

```
                                034B   600                L^DS$GL_ERRCNT, -      ; .. count of error                [24]
                                034B   601                L^DSA$GL_PASSNO, -     ; .. pass count                    [24]
                                034B   602                #0                     ; .. current time                  [24]
                      00   DD   034B                     PUSHL   #0
            0000FE54'EF   DD   034D                     PUSHL   L^DSA$GL_PASSNO
            00000000'EF   DD   0353                     PUSHL   L^DS$GL_ERRCNT
            000000C7'EF   9F   0359                     PUSHAB  L^FMTLSTPAS
                7E   01   9A   035F                     movzbl  #ds$k_printf, -(sp)
                7E   10   98   0362                     cvtbl   #ds$k_type_program_end, -(sp)
        00000000'GF   06   FB   0365                     CALLS   #$$N+2, G^DSX$PRINT
                                036C   603
```

```
                                036C   605          QA_MAIN Dispat_DSX$EndPass_Mid  ; Call QA$Main                          [29]
                        00   DD 036C                PUSHL   #DSQA$K_Dispat_DSX$EndPass_Mid
        00000000'9F     01   FB 036E                CALLS   #1, a#QA$MAIN
                                0375   606          Br_If_Not_QA    5$              ; If not running QA, branch around       [32]
        0000FE00'EF     0F   E1 0375                BBC     #DSA$V_QA, -                    ; If bit not set, branch         [29]
                        06      037C                          L^DSA$GL_Flags, 5$            ;                                [29]
             00000000'EF    17 037D   607          JMP     VRRestart                ; Running QA: loop back to VRRestart     [29]
                                0383   608                                          ; for the next QA check.                 [29]
                                0383   609 5$:                                      ;                                        [29]
                                0383   610          Set_CmdFlg                      ; Set command mode                       [32]
        00000000'EF     07   E2 0383                BBSS    #DS$V_CmdFlg, -          ; Branch if CMDFLG bit is set to         [29]
                        00      038A                          L^DS$GL_Flags, 30008$ ; ...                                    [29]
                                038B       30008$:                                  ; ... here. Set bit regardless.          [29]
        0000FE40'EF     0A   D0 038B   611          MOVL    #APM$_DONE, -            ; INDICATE PROCESSING COMPLETE
                                0392   612                    L^DSA$GL_MSGTYP
             00000000'EF    17 0392   613          JMP     BEGIN                    ; Clean up the world and go back to CLI
                                0398   614
        0000FE54'EF     01   D1 0398   615 10$:     CMPL    #1, L^DSA$GL_PASSNO      ; IS THIS THE FIRST PASS?
                        1B   12 039F   616          BNEQ    20$                      ; SKIP IF NOT
                                03A1   617
                                03A1   618          $PRINT  #ds$k_type_first_pass, -; Print first pass message               [24]
                                03A1   619                    #ds$k_printf, -        ; .. use PRINTF                          [24]
                                03A1   620                    L^FMTPASONE, -         ; .. format                              [24]
                                03A1   621                    L^DS$GL_ERRCNT, -      ; .. error count so far                  [24]
                                03A1   622                    #0                     ; .. current time                        [24]
                        00   DD 03A1                        PUSHL   #0
             00000000'EF    DD 03A3                        PUSHL   L^DS$GL_ERRCNT
             0000008C'EF    9F 03A9                        PUSHAB  L^FMTPASONE
                  7E    01   9A 03AF                        movzbl  #ds$k_printf, -(sp)
                  7E    11   98 03B2                        cvtbl   #ds$k_type_first_pass, -(sp)
        00000000'GF     05   FB 03B5                        CALLS   #$$N+2, G^DSX$PRINT
                                03BC   623
             0000FE54'EF    D6 03BC   624 20$:     INCL    L^DSA$GL_PASSNO          ; COUNT THIS PASS
                                03C2   625
                                03C2   626 DSX$ENDPASS_X:
                                03C2   627
                                03C2   628          QA_MAIN Dispat_DSX$EndPass_End  ; Call QA$Main                          [22]
                        02   DD 03C2                PUSHL   #DSQA$K_Dispat_DSX$EndPass_End
        00000000'9F     01   FB 03C4                CALLS   #1, a#QA$MAIN
                                03CB   629
             00000000'EF    16 03CB   630          JSB     KB_CHECK                 ; CHECK KEYBOARD STATUS.
                            04 03D1   631          RET
```

```
                        03D2    633          .SBTTL  DS_Cleanup Routine
                        03D2    634 ;**
                        03D2    635 ; FUNCTIONAL DESCRIPTION:
                        03D2    636 ;
                        03D2    637 ;       EXECUTE USER'S CLEANUP CODE AND DEALLOCATE DEVICES.
                        03D2    638 ;
                        03D2    639 ; CALLING SEQUENCE:
                        03D2    640 ;
                        03D2    641 ;       BSBW    DS_CLEANUP
                        03D2    642 ;
                        03D2    643 ; INPUT PARAMETERS:     NONE
                        03D2    644 ;
                        03D2    645 ; IMPLICIT INPUTS:
                        03D2    646 ;
                        03D2    647 ;       L$A_CCP         = USER'S CLEANUP ROUTINE ADDRESS.
                        03D2    648 ;       L$L_UNIT        = MAXIMUM NUMBER OF UNITS.
                        03D2    649 ;
                        03D2    650 ; OUTPUT PARAMETERS:    NONE
                        03D2    651 ;
                        03D2    652 ; IMPLICIT OUTPUTS:     NONE
                        03D2    653 ;
                        03D2    654 ; COMPLETION CODES:     NONE
                        03D2    655 ;
                        03D2    656 ; SIDE EFFECTS:         NONE
                        03D2    657 ;--
```

```
                                  03D2   659 DS_CLEANUP::
                                  03D2   660         Br_If_StrFlg    1$                  ; If started, continue on                [32]
        00000000'EF    02    E0   03D2               BBS     #DS$V_StrFlg, -                     ; If bit set, branch              [29]
                       03         03D9                         L^DS$GL_Flags, 1$                 ;                                 [29]
                     00C9    31   03DA   661         BRW     30$                         ; Exit if not started                    [22]
                                  03DD   662
                                  03DD   663 1$:     Br_If_Not_DonFlg        2$          ; If not done already, branch around     [32]
        00000000'EF    0D    E1   03DD               BBC     #DS$V_DonFlg, -                     ; If bit not set, branch          [29]
                       03         03E4                         L^DS$GL_Flags, 2$                 ;                                 [29]
                     00BE    31   03E5   664         BRW     30$                         ; Exit if cleanup already done           [22]
                                  03E8   665
                                  03E8   666 2$:     Set_DonFlg                          ; Indicate that cleanup has been done    [32]
        00000000'EF    0D    E2   03E8               BBSS    #DS$V_DonFlg, -             ; Branch if DONFLG bit is set to         [29]
                       00         03EF                         L^DS$GL_Flags, 30009$ ; ...                                      [29]
                                  03F0       30009$:                                     ; ... here. Set bit regardless.          [29]
                       07    BB   03F0   667         PUSHR   #^M<R0,R1,R2>               ; SAVE REGISTERS.
                                  03F2   668
                                  03F2   669         QA_MAIN Dispat_DS_Cleanup_Bgn       ;                                        [22]
                       03    DD   03F2               PUSHL   #DSQA$K_Dispat_DS_Cleanup_Bgn
        00000000'9F    01    FB   03F4               CALLS   #1, @#QA$MAIN
                                  03FB   670
                                  03FB   671         Clear_CtrlC                         ; Clear ^C pending bit to avoid ^C       [32]
        00000000'EF    00    E4   03FB               BBSC    #DS$V_CtrlC, -              ; Branch if CTRLC bit is set to          [29]
                       00         0402                         L^DS$GL_Flags, 30010$ ; ...                                      [29]
                                  0403       30010$:                                     ; ... here. Clear bit regardless.        [29]
                                  0403   672                                             ; ... on cleanup.                [32]  ;G:2
                                  0403   673         $DS_CNTRLC_S                        ; DELETE USER CONTROL-C HANDLER
                       00    DD   0403                       PUSHL   #0
                       00    DD   0405                       PUSHL   #0
        00000000'9F    02    FB   0407                       CALLS   #2, @#DS$CNTRLC
                                  040E   674         $CANTIM_S                           ; CANCEL ALL TIMERS
                       7E    7C   040E                       CLRQ    -(SP)
        00000000'GF    02    FB   0410                       CALLS   #2,G^SYS$CANTIM
00000000'EF   02000000 8F    CA   0417   675         BICL2   #DS$M_TIMED,DS$GL_FLAGS     ; Cancel /TIME                   [43]  ;G:2
              0000FE50'EF    D4   0422   676         CLRL    L^DSA$GL_TESTNO             ; TEST 0,
        0000FE4C'EF    02    D0   0428   677         MOVL    #2, L^DSA$GL_SUBTNO         ; SUBTEST 2 INDICATES CLEANUP.
                                  042F   678
        00000240'FF    00    FB   042F   679         CALLS   #0, @L$A_CCP                ; EXECUTE USER'S CLEANUP.
```

```
                                  0436   681          Br_If_User      20$                ; Branch if user mode          [32]  ;G:2
        0000FE00'EF    1C    E0   0436                BBS     #DSA$V_User, -                    ; If bit set, branch          [28]
                       56         043D                          L^DSA$GL_Flags, 20$             ;                             [28]
        03   00000000'EF    91    043E   682          CmpB    L^Exe$GB_CpuType, -       ; Check if CPU is                     [34]
                                  0445   683                   #Pr$_Sid_Typ730          ; .. a Nebula                         [34]
                       1C    12   0445   684          BNeq    3$                        ; Skip if not                         [34]
        50   00F26200'EF    9E    0447   685          MovAB   Ioc$K_IOSpace!^XF26200, R0        ; Address of IDC              [34]
        51   00000004'EF    DE    044E   686          MovAL   L^Scb_Base+4, R1          ; Get mchk handler address            [34]
                       61    DD   0455   687          PushL   (R1)                      ; Save old one                        [34]
        61   00000000'EF    9E    0457   688          MovAB   L^Tst$MChk, (R1)          ; Use common cheap one                [34]
                       60    D4   045E   689          ClrL    (R0)                      ; Clear register                      [34]
                       61  8ED0   0460   690          PopL    (R1)                      ; Restore old handler                 [34]
                                  0463   691
  02   00000204'EF   02    00    ED   0463   692 3$:     CMPZV   #0,#2,L$L_ENVIRON, -
                                  046C   693                   #SEP_FUNCTIONAL          ; SEP_FUNCTIONAL
                       07    12   046C   694          BNEQ    5$                        ; Branch if not
        00000000'EF    6C    FA   046E   695          CALLG   (AP), L^QIO$CLEANUP       ; CLEAN AFTER QIO IF NECESSARY
                                  0475   696
        00000000'EF    00    FB   0475   697 5$:     CALLS   #0, L^DS$INITSCB          ; Re-initialize the SCB too.
             00000000'EF    16    047C   698          JSB     L^RMS$CLEANUP             ; Clean up RMS storage
        50   00000000'EF    9A    0482   699          MOVZBL  L^DS$GB_MM_ENB, R0        ; Get state of memory managment
                       03    13   0489   700          BEQL    10$                       ; Branch if not enabled
                  50   01    D0   048B   701          MOVL    #1,R0                     ; Set to enable
                                  048E   702
             00000000'EF    16    048E   703 10$:    JSB     L^DSR$MMENABLE            ; Set MM according to R0
                                  0494   704
                                  0494   705 20$:    QA_MAIN Dispat_DS_Cleanup_End     ;                                     [22]
                       04    DD   0494                PUSHL   #DSQA$K_Dispat_DS_Cleanup_End
        00000000'9F    01    FB   0496                CALLS   #1, @#QA$MAIN
                                  049D   706
        00000000'EF    00    FB   049D   707          CALLS   #0,MP$_DEALL_MEMORY       ; Deallocate any MP memory            [40]
                       07    BA   04A4   708          POPR    #^M<R0,R1,R2>             ; RESTORE REGISTERS.                  [22]
                                  04A6   709
                       05         04A6   710 30$:    RSB                               ; RETURN.
```

```
                        .SBTTL  DSX$DoSummary and VRSummary Routines
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;
                ; CALLING SEQUENCE:
                ;
                ;       CALLS   #0,DS$SUMMARY   -or-
                ;       BSBW    VRSUMMARY               from command scanner
                ;
                ; INPUT PARAMETERS:     NONE
                ;
                ; IMPLICIT INPUTS:      NONE
                ;
                ; OUTPUT PARAMETERS:    NONE
                ;
                ; IMPLICIT OUTPUTS:     NONE
                ;
                ; COMPLETION CODES:     NONE
                ;
                ; SIDE EFFECTS:         NONE
                ;--
```

```
                     0000  04A7    735 .ENTRY  DSX$DOSUMMARY,^M<>
          00000000'EF  0E   90  04A9    736         Movb    #Ds$K_Type_Summary, -           ; Set Type code to              [28]
                            04B0    737                   L^Ds$GB_TypeCode              ; .. Summary                    [28]
                       71   10  04B0    738         Bsbb    Display_Summary               ; Type out Summary stuff        [28]
                00000000'EF  16  04B2    739         JSB     KB_CHECK
                             04  04B8    740         RET
                             04B9    741
                             04B9    742 VRSUMMARY::
                00000000'EF  16  04B9    743         Jsb     DSR$CheckLoad                 ; Check to see if a program is  [36]
                      4D 50  E9  04BF    744         Blbc    R0, 10$                       ; . . . loaded. If not, branch  [36]
                             04C2    745
                             04C2    746         $Print  #-ds$k_type_summary, -          ; Summary type (sticky)         [28]
                             04C2    747                   #ds$k_printf, -               ; .. use PRINTF                 [28]
                             04C2    748                   L^Fmt_Summary, -              ; .. format string              [28]
                             04C2    749                   L^I$a_name, -                 ; .. diagnostic name            [28]
                             04C2    750                   L^I$l_rev, -                  ; .. revision level             [28]
                             04C2    751                   L^I$l_update, -               ; .. update level               [28]
                             04C2    752                   L^ds$gl_errcnt, -             ; .. error counter              [28]
                             04C2    753                   L^Ds$GL_HardErr_Count, -      ; .. Hard error counter         [28]
                             04C2    754                   L^Ds$GL_SoftErr_Count, -      ; .. Soft error counter         [28]
                             04C2    755                   L^Ds$GL_SysErr_Count, -       ; .. System error counter       [28]
                             04C2    756                   L^Ds$GL_DevErr_Count, -       ; .. Device error counter       [28]
                             04C2    757                   L^Ds$GL_ErrSup_Count          ; .. ErrSup counter             [28]
          00000000'EF  DD  04C2                     PUSHL   L^Ds$GL_ErrSup_Count
          00000000'EF  DD  04C8                     PUSHL   L^Ds$GL_DevErr_Count
          00000000'EF  DD  04CE                     PUSHL   L^Ds$GL_SysErr_Count
          00000000'EF  DD  04D4                     PUSHL   L^Ds$GL_SoftErr_Count
          00000000'EF  DD  04DA                     PUSHL   L^Ds$GL_HardErr_Count
          00000000'EF  DD  04E0                     PUSHL   L^ds$gl_errcnt
          00000210'EF  DD  04E6                     PUSHL   L^I$l_update
          0000020C'EF  DD  04EC                     PUSHL   L^I$l_rev
          00000208'EF  DD  04F2                     PUSHL   L^I$a_name
          0000015C'EF  9F  04F8                     PUSHAB  L^Fmt_Summary
              7E   01  9A  04FE                     movzbl  #ds$k_printf, -(sp)
            7E   F2 8F  98  0501                     cvtbl   #-ds$k_type_summary, -(sp)
00000000'GF      0C  FB  0505                     CALLS   #$$N+2, G^DSX$PRINT
                 15  10  050C    758         Bsbb    Display_Summary               ; Type out Summary stuff        [28]
                     05  050E    759         Rsb                                   ;                               [28]
```

```
                                 050F   761 10$:    $Print  #ds$k_type_command_err, -       ; Command error                  [28]
                                 050F   762                    #ds$k_printf, -           ; .. use PRINTF                  [28]
                                 050F   763                    L^fmt_nodiag              ; Print 'no diagnostic loaded'   [28]
                000002D6'EF   9F 050F                         PUSHAB  L^fmt_nodiag
                          7E   01 9A 0515                     movzbl  #ds$k_printf, -(sp)
                          7E   15 98 0518                     cvtbl   #ds$k_type_command_err, -(sp)
              00000000'GF   03   FB 051B                     CALLS   #$$N+2, G^DSX$PRINT
                                 05 0522   764          Rsb                               ;                                [28]
                                 0523   765
                                 0523   766 Display_Summary:                              ; Execute Summary command        [28]
              00000244'FF   00   FB 0523   767          Calls   #0, @L$A_Repp             ; Call user report code.         [28]
                   00000000'EF   94 052A   768          Clrb    L^Ds$GB_TypeCode          ; Clear sticky type code         [28]
                                 05 0530   769          Rsb                               ;                                [28]
```

```
0531    771          .SBTTL  VRShowStatus Routine
0531    772 ;++
0531    773 ; Functional Description:
0531    774 ;       Execute a SHOW STATUS command (print test, section, etc).
0531    775 ;
0531    776 ; Calling sequence:
0531    777 ;
0531    778 ;       BSBW    VRSHOWSTATUS
0531    779 ;
0531    780 ; Input parameters: none
0531    781 ;
0531    782 ; Implicit inputs: none
0531    783 ;
0531    784 ; Outputs: none
0531    785 ;
0531    786 ; Implicit outputs: none
0531    787 ;
0531    788 ; Completion codes: none
0531    789 ;
0531    790 ;--
```

```
                                0531   VRShowStatus::                                  ;                                [20]
          00000000'EF    16     0531           Jsb     DSR$CheckLoad                   ; Check to see if a program is   [36]
                 03 50   E8     0537           Blbs    R0, 5$                          ; ... loaded. If so, skip over   [37]
                  00AF   31     053A           Brw     100$                            ; Not loaded, print error mess.  [37]
                                053D
           50   3C AC    D0     053D   5$:     MovL    60(AP), R0                      ; Get the PC from the CLI args   [37]
      50   00010000 8F   D1     0541           CmpL    #^X00010000, R0                 ; Is this a real user PC?        [37]
                   07   14     0548           Bgtr    10$                             ; Yes, branch around             [37]
                                054A
      00000000'EF   00   FB     054A           Calls   #0, L^DSR$Find_User_PC          ; Get last user PC from stack    [37]
                                0551
   51   0000FE10'EF   01   C1   0551   10$:    addl3   #1, dsa$gl_sectno, r1           ; Get offset in section table    [23]
      51   00000250'FF41   D0   0559           movl    @l$a_secnam[r1], r1             ; Get ASCIC name address         [23]
                                0561
              50   00   D1     0561           CmpL    #0, R0                          ; Is this really a valid PC?     [37]
                   44   13     0564           Beql    50$                             ; No, branch                     [37]
                                0566
                                0566           $print  #ds$k_type_command_out, -       ; Command output message         [24]
                                0566                     #ds$k_printf, -               ; .. use PRINTF                  [24]
                                0566                     L^fmt_showstatus, -           ; .. format string               [24]
                                0566                     l$a_name, -                   ; .. diagnostic name             [20]
                                0566                     l$l_rev, -                    ; .. revision level              [23]
                                0566                     l$l_update, -                 ; .. update level                [23]
                                0566                     #0, -                         ; .. current time                [23]
                                0566                     r1, -                         ; .. address of section name     [20]
                                0566                     ds$gl_errcnt, -               ; .. error counter               [23]
                                0566                     dsa$gl_testno, -              ; .. test number                 [20]
                                0566                     dsa$gl_subtno, -              ; .. subtest number              [20]
                                0566                     dsa$gl_passno, -              ; .. pass number                 [20]
                                0566                     r0                            ; .. User PC of ^C               [20]
                   50   DD     0566                   PUSHL   r0
          0000FE54'EF   DD     0568                   PUSHL   dsa$gl_passno
          0000FE4C'EF   DD     056E                   PUSHL   dsa$gl_subtno
          0000FE50'EF   DD     0574                   PUSHL   dsa$gl_testno
          00000000'EF   DD     057A                   PUSHL   ds$gl_errcnt
                   51   DD     0580                   PUSHL   r1
                   00   DD     0582                   PUSHL   #0
          00000210'EF   DD     0584                   PUSHL   l$l_update
          0000020C'EF   DD     058A                   PUSHL   l$l_rev
          00000208'EF   DD     0590                   PUSHL   l$a_name
          0000025C'EF   9F     0596                   PUSHAB  L^fmt_showstatus
              7E   01   9A     059C                   movzbl  #ds$k_printf, -(sp)
              7E   16   98     059F                   cvtbl   #ds$k_type_command_out, -(sp)
      00000000'GF   0D   FB     05A2                   CALLS   #$$N+2, G^DSX$PRINT
                        05     05A9           rsb                                     ;                                [20]
                                05AA
                                05AA   50$:    $print  #ds$k_type_command_out, -       ; Command output message         [24]
                                05AA                     #ds$k_printf, -               ; .. use PRINTF                  [24]
                                05AA                     L^Fmt_ShowStatus_No_PC, -     ; .. format string               [24]
                                05AA                     l$a_name, -                   ; .. diagnostic name             [20]
                                05AA                     l$l_rev, -                    ; .. revision level              [23]
                                05AA                     l$l_update, -                 ; .. update level                [23]
                                05AA                     #0, -                         ; .. current time                [23]
                                05AA                     r1, -                         ; .. address of section name     [20]
                                05AA                     ds$gl_errcnt, -               ; .. error counter               [23]
                                05AA                     dsa$gl_testno, -              ; .. test number                 [20]
                                05AA                     dsa$gl_subtno, -              ; .. subtest number              [20]
```

```
                          05AA   835                 dsa$gl_passno                  ; .. pass number                [37]
      0000FE54'EF   DD    05AA                     PUSHL   dsa$gl_passno
      0000FE4C'EF   DD    05B0                     PUSHL   dsa$gl_subtno
      0000FE50'EF   DD    05B6                     PUSHL   dsa$gl_testno
      00000000'EF   DD    05BC                     PUSHL   ds$gl_errcnt
               51   DD    05C2                     PUSHL   r1
               00   DD    05C4                     PUSHL   #0
      00000210'EF   DD    05C6                     PUSHL   l$l_update
      0000020C'EF   DD    05CC                     PUSHL   l$l_rev
      00000208'EF   DD    05D2                     PUSHL   l$a_name
      000001F1'EF   9F    05D8                     PUSHAB  L^Fmt_ShowStatus_No_PC
          7E   01   9A    05DE                     movzbl  #ds$k_printf, -(sp)
          7E   16   98    05E1                     cvtbl   #ds$k_type_command_out, -(sp)
  00000000'GF   0C   FB    05E4                     CALLS   #$$N+2, G^DSX$PRINT
                    05    05EB   836         rsb                                    ;                               [20]
                          05EC   837
                          05EC   838 100$:   $print  #ds$k_type_command_err, -      ; Command error                 [24]
                          05EC   839                 #ds$k_printf, -               ; .. use PRINTF                 [24]
                          05EC   840                 L^fmt_nodiag                  ; Print 'no diagnostic running' [20]
      000002D6'EF   9F    05EC                     PUSHAB  L^fmt_nodiag
          7E   01   9A    05F2                     movzbl  #ds$k_printf, -(sp)
          7E   15   98    05F5                     cvtbl   #ds$k_type_command_err, -(sp)
  00000000'GF   03   FB    05F8                     CALLS   #$$N+2, G^DSX$PRINT
                    05    05FF   841         rsb                                    ;                               [20]
                          0600   842         .END
```

```
$$N                             = 00000001    D
$$T1                            = 00000001    D
$ER                             = 00000002    D
$MODULE                           00000000 R  D  03
APC$_ABORT                      = 00000006    D
APC$_CONTINUE                   = 00000009    D
APC$_DUMP                       = 00000003    D
APC$_NOP                        = 00000000    D
APC$_START                      = 00000005    D
APC$_ZERO                       = 00000004    D
APM$_ABTDON                     = 00000006    D
APM$_DEVERR                     = 00000002    D
APM$_DONE                       = 0000000A    D
APM$_EXCEPT                     = 0000000B    D
APM$_HARDERR                    = 00000003    D
APM$_MORE                       = 00000008    D
APM$_NOMES                      = 00000000    D
APM$_PREPERR                    = 0000000C    D
APM$_PRGERR                     = 00000005    D
APM$_SOFTERR                    = 00000004    D
APM$_SPOOL                      = 00000009    D
APM$_SYSERR                     = 00000001    D
A_DATAPTR                         00000004 R  D  02
A_TESTPTR                         00000000 R  D  02
BEGIN                             ********    X  04
BIT...                          = 00000004    D
CALLTEST                          00000078 R  D  04
CEP_FUNCTIONAL                  = 00000000    D
CEP_REPAIR                      = 00000001    D
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        0000044C    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_FLAGS2                      00000444    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEMP_BASE                   00000448    D
CLI$L_TEST                        00000020    D
CLI$M_START_OR_RUN              = 00000008    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BASE_QUAL                 = 00000002    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NEXT                      = 00000001    D
CLI$V_NOALLOC                   = 00000000    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_START_OR_RUN              = 00000003    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
DISPATCH                          00000000 RG D  04
DISPATCH_X                        000002B7 R  D  04
DISPLAY_SUMMARY                   00000523 R  D  04
DS$CNTRLC                         ********    X  04
DS$GA_CHKLPPC                     ********    X  04
DS$GB_MM_ENB                      ********    X  04
DS$GB_TYPECODE                    ********    X  04
DS$GL_CLIBASE                     ********    X  04
DS$GL_DEVERR_COUNT                ********    X  04
DS$GL_ERRCNT                      ********    X  04
DS$GL_ERRSUP_COUNT                ********    X  04
DS$GL_FLAGS                       ********    X  04
DS$GL_FSTTEST                     ********    X  04
DS$GL_HARDERR_COUNT               ********    X  04
DS$GL_LSTTEST                     ********    X  04
DS$GL_SOFTERR_COUNT               ********    X  04
DS$GL_SUBTEST                     ********    X  04
DS$GL_SYSERR_COUNT                ********    X  04
DS$INITSCB                        ********    X  04
DS$K_BBC                        = 0000001D  G D
DS$K_BBCC                       = 00000021  G D
DS$K_BBCCI                      = 00000023  G D
DS$K_BBCS                       = 0000001F  G D
DS$K_BBS                        = 0000001C  G D
DS$K_BBSC                       = 00000020  G D
DS$K_BBSS                       = 0000001E  G D
DS$K_BBSSI                      = 00000022  G D
DS$K_BCC_M                      = 0000001B  G D
DS$K_BCS_M                      = 0000001A  G D
```

```
DS$K_BEQLU_B                    = 00000005  G D      DS$K_TYPE_PARAM_ERROR                = 0000001C    D
DS$K_BEQLU_M                    = 00000011  G D      DS$K_TYPE_PROGRAM_END                = 00000010    D
DS$K_BEQL_B                     = 00000004  G D      DS$K_TYPE_PROGRAM_INFO               = 00000017    D
DS$K_BEQL_M                     = 00000010  G D      DS$K_TYPE_PROGRAM_START              = 0000000F    D
DS$K_BERROR                     = 00000026  G D      DS$K_TYPE_QIO_INVADP                 = 00000024    D
DS$K_BGEQU_B                    = 00000007  G D      DS$K_TYPE_QIO_NODRIVER               = 00000022    D
DS$K_BGEQU_M                    = 00000013  G D      DS$K_TYPE_QIO_WRONGVER               = 00000023    D
DS$K_BGEQ_B                     = 00000006  G D      DS$K_TYPE_SCRIPT_ECHO                = 00000021    D
DS$K_BGEQ_M                     = 00000012  G D      DS$K_TYPE_SCRIPT_PNF                 = 0000001E    D
DS$K_BGTRU_B                    = 00000009  G D      DS$K_TYPE_SCRIPT_PROMPT              = 00000020    D
DS$K_BGTRU_M                    = 00000015  G D      DS$K_TYPE_SCRIPT_SKIP                = 0000001F    D
DS$K_BGTR_B                     = 00000008  G D      DS$K_TYPE_SEQUENCE_ERROR             = 00000019    D
DS$K_BGTR_M                     = 00000014  G D      DS$K_TYPE_START_ERR                  = 00000018    D
DS$K_BLBC                       = 00000025  G D      DS$K_TYPE_START_LIST                 = 00000025    D
DS$K_BLBS                       = 00000024  G D      DS$K_TYPE_SUMMARY                    = 0000000E    D
DS$K_BLEQU_B                    = 00000003  G D      DS$K_TYPE_USER_PROMPT                = 00000002    D
DS$K_BLEQU_M                    = 0000000F  G D      DS$M_ABRTFLG                         = 00000040    D
DS$K_BLEQ_B                     = 00000002  G D      DS$M_BADTIME                         = 00100000    D
DS$K_BLEQ_M                     = 0000000E  G D      DS$M_BATCH                           = 00400000    D
DS$K_BLSSU_B                    = 00000001  G D      DS$M_BRKCLR                          = 00001000    D
DS$K_BLSSU_M                    = 0000000D  G D      DS$M_BRKPT                           = 00000800    D
DS$K_BLSS_B                     = 00000000  G D      DS$M_CHARFLG                         = 00000100    D
DS$K_BLSS_M                     = 0000000C  G D      DS$M_CMDFLG                          = 00000080    D
DS$K_BNEQU_B                    = 0000000B  G D      DS$M_CTRLC                           = 00000001    D
DS$K_BNEQU_M                    = 00000017  G D      DS$M_CTRLO                           = 00010000    D
DS$K_BNEQ_B                     = 0000000A  G D      DS$M_DEVFLG                          = 00000200    D
DS$K_BNEQ_M                     = 00000016  G D      DS$M_DISABLCC                        = 01000000    D
DS$K_BNERROR                    = 00000027  G D      DS$M_DONFLG                          = 00002000    D
DS$K_BVC_M                      = 00000019  G D      DS$M_ERRFLG                          = 00000010    D
DS$K_BVS_M                      = 00000018  G D      DS$M_EXCEPT                          = 00080000    D
DS$K_DS_STARTING_LOCATION       = 00010000    D      DS$M_EXETST                          = 00040000    D
DS$K_PRINTB                     = 00000002    D      DS$M_HLTFLG                          = 00000008    D
DS$K_PRINTF                     = 00000001    D      DS$M_LODFLG                          = 00000002    D
DS$K_PRINTI                     = 00000000    D      DS$M_MEMMGT                          = 00008000    D
DS$K_PRINTX                     = 00000003    D      DS$M_NI                              = 04000000    D
DS$K_TYPE_ABORT_PROGRAM         = 00000014    D      DS$M_OUTPUT                          = 00800000    D
DS$K_TYPE_ABORT_TEST            = 00000013    D      DS$M_RUBFLG                          = 00000020    D
DS$K_TYPE_COMMAND_ERR           = 00000015    D      DS$M_SCRIPT                          = 00200000    D
DS$K_TYPE_COMMAND_OUT           = 00000016    D      DS$M_SETIMR                          = 02000000    D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A    D      DS$M_STRFLG                          = 00000004    D
DS$K_TYPE_DS_PROMPT             = 00000001    D      DS$M_SUBT                            = 00004000    D
DS$K_TYPE_DS_START              = 0000001D    D      DS$M_SYSFLG                          = 00000400    D
DS$K_TYPE_ERRDEV                = 00000008    D      DS$M_TIMED                           = 02000000    D
DS$K_TYPE_ERRHARD               = 00000006    D      DS$M_TIMED_OUT                       = 08000000    D
DS$K_TYPE_ERROR_BODY            = 00000009    D      DS$M_TIMRON                          = 00020000    D
DS$K_TYPE_ERROR_END             = 0000000A    D      DS$SUMMARY                             ********    X   04
DS$K_TYPE_ERRPREP               = 0000001B    D      DS$V_ABRTFLG                         = 00000006    D
DS$K_TYPE_ERRSOFT               = 00000007    D      DS$V_BADTIME                         = 00000014    D
DS$K_TYPE_ERRSUP                = 00000004    D      DS$V_BATCH                           = 00000016    D
DS$K_TYPE_ERRSYS                = 00000005    D      DS$V_BRKCLR                          = 0000000C    D
DS$K_TYPE_ERR_HALT              = 0000000D    D      DS$V_BRKPT                           = 0000000B    D
DS$K_TYPE_EXCEPTION             = 0000000C    D      DS$V_CHARFLG                         = 00000008    D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D      DS$V_CMDFLG                          = 00000007    D
DS$K_TYPE_FIRST_PASS            = 00000011    D      DS$V_CTRLC                           = 00000000    D
DS$K_TYPE_GENERAL               = 00000000    D      DS$V_CTRLO                           = 00000010    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D      DS$V_DEVFLG                          = 00000009    D
DS$K_TYPE_NO_TESTS              = 00000012    D      DS$V_DISABLCC                        = 00000018    D
```

```
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSA$V_PASS0                     = 0000001D    D
DSA$V_QA                        = 0000000F    D
DSA$V_SEARCH                    = 0000000E    D
DSA$V_TRACE                     = 0000000A    D
DSA$V_USER                      = 0000001C    D
DSP$A_BASE                        00000000    D
DSP$A_DATA                        00000008    D
DSP$A_MSG                         00000004    D
DSP$A_TEST                        0000000C    D
DSP$K_SIZE                      = 00000018    D
DSP$Q_SECTION                     00000010    D
DSQA$K_DISPAT_AFTER_INIT        = 00000014    D
DSQA$K_DISPAT_BEFORE_INIT       = 00000013    D
DSQA$K_DISPAT_CALLTEST          = 00000015    D
DSQA$K_DISPAT_DONE_TESTS        = 00000018    D
DSQA$K_DISPAT_DSX$ENDPASS_BGN   = 00000001    D
DSQA$K_DISPAT_DSX$ENDPASS_END   = 00000002    D
DSQA$K_DISPAT_DSX$ENDPASS_MID   = 00000000    D
DSQA$K_DISPAT_DS_CLEANUP_BGN    = 00000003    D
DSQA$K_DISPAT_DS_CLEANUP_END    = 00000004    D
DSQA$K_DUMMY_1                  = 00000007    D
DSQA$K_ERROR_ERROR_BGN          = 00000006    D
DSQA$K_ERROR_ERROR_END          = 00000005    D
DSQA$K_KERNEL_INIT_CONTEXT      = 00000008    D
DSQA$K_LOOP_DSX$BGNSUB          = 00000009    D
DSQA$K_LOOP_DSX$CKLOOP          = 0000000B    D
DSQA$K_LOOP_DSX$ENDSUB          = 0000000A    D
DSQA$K_PARAM_DSX$GETADDRESS     = 0000000E    D
DSQA$K_PARAM_DSX$GETDATA        = 0000000C    D
DSQA$K_PARAM_DSX$GETLOGICAL     = 0000000F    D
DSQA$K_PARAM_DSX$GETSTRING      = 00000010    D
DSQA$K_PARAM_DSX$GETVIELD       = 0000000D    D
DSQA$K_QA_DSX$BRANCH            = 00000011    D
DSQA$K_START_BEFORE_HEADER      = 00000017    D
DSQA$K_START_VRRESTART          = 00000012    D
DSQA$K_START_VRSTART            = 00000016    D
DSR$CHECKLOAD                     ********    X   04
DSR$FIND_USER_PC                  ********    X   04
DSR$MMENABLE                      ********    X   04
DSX$DOSUMMARY                     000004A7 RG D   04
DSX$ENDPASS                       000002B8 RG D   04
DSX$ENDPASS_X                     000003C2 R  D   04
DSX$PRINT                         ********   X    04
DS_CLEANUP                        000003D2 RG D   04
DS_ERRSUP                         ********   X    04
ENV$M_DOMAIN                    = 00000002    D
ENV$M_LEVEL                     = 00000001    D
ENV$M_SUPER                     = 000003FC    D
ENV$S_DOMAIN                    = 00000001    D
ENV$S_LEVEL                     = 00000001    D
ENV$S_SUPER                     = 00000008    D
ENV$V_DOMAIN                    = 00000001    D
ENV$V_LEVEL                     = 00000000    D
ENV$V_SUPER                     = 00000002    D
ENV$_CPU                        = 00000000    D
ENV$_FUNCTIONAL                 = 00000000    D
ENV$_REPAIR                     = 00000001    D
ENV$_SUPER                      = 00000001    D
ENV$_SYSTEM                     = 00000001    D
EXE$GB_CPUTYPE                    ********   X    04
FALSE                           = 00000000    D
FIRSTTEST                         00000032 R  D   04
FMTLSTPAS                         000000C7 R  D   03
FMTPASONE                         0000008C R  D   03
FMTSEARCH                         00000114 R  D   03
FMTTRACE                          0000014C R  D   03
FMT_NODIAG                        000002D6 R  D   03
FMT_SHOWSTATUS                    0000025C R  D   03
FMT_SHOWSTATUS_NO_PC              000001F1 R  D   03
FMT_SUMMARY                       0000015C R  D   03
INIT_CONTEXT                      ********   X    04
IOC$K_IOSPACE                     ********   X    04
KB_CHECK                          ********   X    04
L$A_CCP                           00000240    D
L$A_DEVP                          0000021C    D
L$A_DREG                          00000224    D
L$A_DTP                           00000218    D
L$A_ICP                           0000023C    D
L$A_LASTADR                       00000214    D
```

```
L$A_NAME                        00000208    D
L$A_REPP                        00000244    D
L$A_SECNAM                      00000250    D
L$A_STATAB                      00000248    D
L$A_TSTCNT                      00000254    D
L$L_ENVIRON                     00000204    D
L$L_ERRTYP                      0000024C    D
L$L_HEADLENGTH                  00000200    D
L$L_REV                         0000020C    D
L$L_UNIT                        00000220    D
L$L_UNUSED                      00000228    D
L$L_UPDATE                      00000210    D
MP$_DEALL_MEMORY                ********   X   00
OFF                           = 00000000    D
ON                            = 00000001    D
PR$_SID_TYP730                = 00000003    D
QA$AOB_CHECK_STATE              ********   X   04
QA$AOB_FLAGS                    ********   X   04
QA$K_ABORT_NOW                = 00000007    D
QA$K_APT_PHASE_ONE            = 0000000B    D
QA$K_APT_PHASE_TWO            = 0000000C    D
QA$K_BAD_ADDRESS              = 00000009    D
QA$K_CONTROL_C_CONT           = 0000000A    D
QA$K_DEALLOCATION             = 0000000E    D
QA$K_ERROR_PHASE_ONE          = 00000005    D
QA$K_ERROR_PHASE_THREE        = 00000007    D
QA$K_ERROR_PHASE_TWO          = 00000006    D
QA$K_FAILURE                  = 00000000    D
QA$K_FIRST_BRANCH_CODE        = 00000000    D
QA$K_FIRST_TIME               = 00000005    D
QA$K_INIT_CONTEXT_DONE        = 00000003    D
QA$K_LAST_BRANCH_CODE         = 00000025    D
QA$K_LOOP_ON_SUBTEST          = 00000003    D
QA$K_LOOP_ON_TEST             = 00000002    D
QA$K_LOOP_TEST_DONE           = 00000004    D
QA$K_MEMORY_MANAGE            = 0000000D    D
QA$K_MULTIPLE_PASS            = 00000001    D
QA$K_NODEF                    = 00000000    D
QA$K_NORMAL_START             = 00000000    D
QA$K_NO_HEADER                = 00000006    D
QA$K_NUMBER_OF_CHECKS         = 0000000F    D
QA$K_NUMBER_QA_FLAGS          = 00000008    D
QA$K_NUMB_ROUTINE_STATES      = 00000004    D
QA$K_QA_DEBUG                 = 00000001    D
QA$K_QA_DONE                  = 00000002    D
QA$K_RUN_BACKWARDS            = 00000004    D
QA$K_SECTION                  = 00000008    D
QA$K_SUCCESS                  = 00000001    D
QA$MAIN                         ********   X   04
QA$V_CHECK_DONE               = 00000003    D
QA$V_DOING_CLEANUP            = 00000002    D
QA$V_ERROR_FOUND              = 00000001    D
QA$V_INIT_DONE                = 00000000    D
QA$W_SAVE_LAST                  ********   X   04
QIO$CLEANUP                     ********   X   04
REQ_SUB                         00000008 R  D   02
RMS$CLEANUP                     ********   X   04
SCB_BASE                        ********   X   04
SEP_FUNCTIONAL                = 00000002    D
SEP_REPAIR                    = 00000003    D
SIZ...                        = 00000001    D
SYS$CANTIM                      ********  GX   04
TRUE                          = 00000001    D
TST$MCHK                        ********   X   04
T_NOEXE                         0000006B R  D   03
T_NOSUB                         00000009 R  D   03
T_TSTABO                        000002F2 R  D   03
VRRESTART                       ********   X   04
VRSHOWSTATUS                    00000531 RG D   04
VRSUMMARY                       000004B9 RG D   04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000000C (12.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	000002FD (765.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000600 (1536.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                      VALUE       DEFINITION      REFERENCES...
------                      -----       ----------      -------------
$$N                        =00000001    840    (1)      376    (1)      421    (1)    #-433    (1)    #-530    (1)
                                                        595    (1)      602    (1)      622    (1)      757    (1)
                                                      #-763    (1)      821    (1)      835    (1)    #-840    (1)
$$T1                       =00000001    674    (1)      674    (1)
$ER                        =00000002    360    (1)    #-360    (1)
$MODULE                     00000000-R  225    (1)      360    (1)
APC$_ABORT                 =00000006    202    (1)
APC$_CONTINUE              =00000009    202    (1)
APC$_DUMP                  =00000003    202    (1)
APC$_NOP                   =00000000    202    (1)
APC$_START                 =00000005    202    (1)
APC$_ZERO                  =00000004    202    (1)
APM$_ABTDON                =00000006    202    (1)
APM$_DEVERR                =00000002    202    (1)
APM$_DONE                  =0000000A    202    (1)    #-611    (1)
APM$_EXCEPT                =0000000B    202    (1)
APM$_HARDERR               =00000003    202    (1)
APM$_MORE                  =00000008    202    (1)
APM$_NOMES                 =00000000    202    (1)
APM$_PREPERR               =0000000C    202    (1)
APM$_PRGERR                =00000005    202    (1)
APM$_SOFTERR               =00000004    202    (1)
APM$_SPOOL                 =00000009    202    (1)
APM$_SYSERR                =00000001    202    (1)
A_DATAPTR                   00000004-R  218    (1)    #-379    (1)      389    (1)    #-442    (1)    #-445    (1)
                                                      #-446    (1)
A_TESTPTR                   00000000-R  217    (1)    #-353    (1)      390    (1)
BEGIN                       00000000-XR                 613    (1)
BIT...                     =00000004    208    (1)      199    (1)      204    (1)      206    (1)      207    (1)
                                                        208    (1)
CALLTEST                    00000078-R  334    (1)    #-331    (1)    #-493    (1)    #-501    (1)    #-511    (1)
CEP_FUNCTIONAL             =00000000    199    (1)
CEP_REPAIR                 =00000001    199    (1)
CLI$K_BUFSIZ               =00000100    203    (1)      203    (1)
CLI$K_SIZE                  0000044C    203    (1)
CLI$L_ADDRESS               00000018    203    (1)
CLI$L_COMMAND               00000004    203    (1)
CLI$L_DATA                  0000001C    203    (1)
CLI$L_FLAGS                 00000000    203    (1)
CLI$L_FLAGS2                00000444    203    (1)
CLI$L_LAST                  00000024    203    (1)    #-485    (1)
CLI$L_NEXT                  00000030    203    (1)
CLI$L_PASS                  0000002C    203    (1)
CLI$L_SUBT                  00000028    203    (1)    #-480    (1)
CLI$L_TEMP_BASE             00000448    203    (1)
CLI$L_TEST                  00000020    203    (1)    #-483    (1)
CLI$M_START_OR_RUN         =00000008    203    (1)
CLI$Q_BUFQWD                00000034    203    (1)
CLI$Q_FILE                  00000008    203    (1)
CLI$Q_SECTION               00000010    203    (1)
```

```
CLI$Q_TIME                  0000043C     203    (1)
CLI$T_BUFFER                0000003C     203    (1)
CLI$V_ADAPTER              =00000018     203    (1)
CLI$V_ADR                  =0000000B     203    (1)
CLI$V_ASCII                =00000013     203    (1)
CLI$V_BASE_QUAL            =00000002     203    (1)
CLI$V_BREAK                =0000000A     203    (1)
CLI$V_BRIEF                =0000001B     203    (1)
CLI$V_BYTE                 =0000000D     203    (1)
CLI$V_CLEAR                =00000002     203    (1)
CLI$V_DEC                  =00000010     203    (1)
CLI$V_DEFAULT              =0000000C     203    (1)
CLI$V_DEPOSIT              =00000019     203    (1)
CLI$V_EVENT                =00000008     203    (1)
CLI$V_EXAM                 =00000005     203    (1)
CLI$V_FLAGS                =00000009     203    (1)
CLI$V_HEX                  =00000012     203    (1)
CLI$V_KERNEL               =00000017     203    (1)
CLI$V_LOAD                 =00000006     203    (1)
CLI$V_LONG                 =0000000F     203    (1)
CLI$V_NEXT                 =00000001     203    (1)
CLI$V_NOALLOC              =00000000     203    (1)
CLI$V_NOTNUF               =00000001     203    (1)
CLI$V_OCT                  =00000011     203    (1)
CLI$V_PREG                 =0000001A     203    (1)
CLI$V_QA                   =00000007     203    (1)
CLI$V_QACKLOOPLOOPS        =0000001C     203    (1)
CLI$V_QAERRORPRINTS        =0000001B     203    (1)
CLI$V_QAMULTIPLEPASS       =0000001F     203    (1)
CLI$V_QASUBTESTLOOPS       =0000001E     203    (1)
CLI$V_QATESTLOOPS          =0000001D     203    (1)
CLI$V_REG                  =00000014     203    (1)
CLI$V_REQUIRED             =00000000     203    (1)
CLI$V_RUN                  =00000015     203    (1)
CLI$V_SET                  =00000003     203    (1)
CLI$V_SHOW                 =00000004     203    (1)
CLI$V_START_OR_RUN         =00000003     203    (1)      203    (1)
CLI$V_VALSEC               =00000016     203    (1)
CLI$V_WORD                 =0000000E     203    (1)
DISPATCH                    00000000-R   300    (1)
DISPATCH_X                  000002B7-R   532    (1)    #-361    (1)
DISPLAY_SUMMARY             00000523-R   766    (1)    #-738    (1)    #-758    (1)
DS$CNTRLC                   00000000-XR                 673    (1)
DS$GA_CHKLPPC               00000000-XR               #-381    (1)
DS$GB_MM_ENB                00000000-XR               #-699    (1)
DS$GB_TYPECODE              00000000-XR               #-737    (1)    #-768    (1)
DS$GL_CLIBASE               00000000-XR                 479    (1)
DS$GL_DEVERR_COUNT          00000000-XR               #-305    (1)    #-757    (1)
DS$GL_ERRCNT                00000000-XR               #-301    (1)    #-602    (1)    #-622    (1)    #-757    (1)
                                                      #-821    (1)    #-835    (1)
DS$GL_ERRSUP_COUNT          00000000-XR               #-306    (1)    #-757    (1)
DS$GL_FLAGS                 00000000-XR                 341    (1)      383    (1)      401    (1)      402    (1)
                                                        409    (1)      519    (1)      520    (1)      573    (1)
                                                        576    (1)      610    (1)      660    (1)      663    (1)
                                                        666    (1)      671    (1)    #-675    (1)
DS$GL_FSTTEST               00000000-XR               #-324    (1)    #-411    (1)    #-484    (1)    #-498    (1)
DS$GL_HARDERR_COUNT         00000000-XR               #-302    (1)    #-757    (1)
```

```
DS$GL_LSTTEST                   00000000-XR                  #-392    (1)    #-410    (1)    #-412    (1)    #-421    (1)
                                                             #-486    (1)    #-510    (1)    #-583    (1)    #-595    (1)
DS$GL_SOFTERR_COUNT             00000000-XR                  #-303    (1)    #-757    (1)
DS$GL_SUBTEST                   00000000-XR                  #-481    (1)    #-579    (1)    #-585    (1)    #-595    (1)
DS$GL_SYSERR_COUNT              00000000-XR                  #-304    (1)    #-757    (1)
DS$INITSCB                      00000000-XR                   697     (1)
DS$K_BBC                       =0000001D      207    (1)
DS$K_BBCC                      =00000021      207    (1)
DS$K_BBCCI                     =00000023      207    (1)
DS$K_BBCS                      =0000001F      207    (1)
DS$K_BBS                       =0000001C      207    (1)
DS$K_BBSC                      =00000020      207    (1)
DS$K_BBSS                      =0000001E      207    (1)
DS$K_BBSSI                     =00000022      207    (1)
DS$K_BCC_M                     =0000001B      207    (1)
DS$K_BCS_M                     =0000001A      207    (1)
DS$K_BEQLU_B                   =00000005      207    (1)
DS$K_BEQLU_M                   =00000011      207    (1)
DS$K_BEQL_B                    =00000004      207    (1)
DS$K_BEQL_M                    =00000010      207    (1)
DS$K_BERROR                    =00000026      207    (1)
DS$K_BGEQU_B                   =00000007      207    (1)
DS$K_BGEQU_M                   =00000013      207    (1)
DS$K_BGEQ_B                    =00000006      207    (1)
DS$K_BGEQ_M                    =00000012      207    (1)
DS$K_BGTRU_B                   =00000009      207    (1)
DS$K_BGTRU_M                   =00000015      207    (1)
DS$K_BGTR_B                    =00000008      207    (1)
DS$K_BGTR_M                    =00000014      207    (1)
DS$K_BLBC                      =00000025      207    (1)      207    (1)
DS$K_BLBS                      =00000024      207    (1)
DS$K_BLEQU_B                   =00000003      207    (1)
DS$K_BLEQU_M                   =0000000F      207    (1)
DS$K_BLEQ_B                    =00000002      207    (1)
DS$K_BLEQ_M                    =0000000E      207    (1)
DS$K_BLSSU_B                   =00000001      207    (1)
DS$K_BLSSU_M                   =0000000D      207    (1)
DS$K_BLSS_B                    =00000000      207    (1)      207    (1)
DS$K_BLSS_M                    =0000000C      207    (1)
DS$K_BNEQU_B                   =0000000B      207    (1)
DS$K_BNEQU_M                   =00000017      207    (1)
DS$K_BNEQ_B                    =0000000A      207    (1)
DS$K_BNEQ_M                    =00000016      207    (1)
DS$K_BNERROR                   =00000027      207    (1)
DS$K_BVC_M                     =00000019      207    (1)
DS$K_BVS_M                     =00000018      207    (1)
DS$K_DS_STARTING_LOCATION      =00010000      207    (1)
DS$K_PRINTB                    =00000002      208    (1)
DS$K_PRINTF                    =00000001      208    (1)  #-376    (1)    #-421    (1)    #-595    (1)    #-602    (1)
                                                           #-622    (1)    #-757    (1)    #-763    (1)    #-821    (1)
                                                           #-835    (1)    #-840    (1)
DS$K_PRINTI                    =00000000      208    (1)  #-433    (1)    #-530    (1)
DS$K_PRINTX                    =00000003      208    (1)
DS$K_TYPE_ABORT_PROGRAM        =00000014      208    (1)
DS$K_TYPE_ABORT_TEST           =00000013      208    (1)  #-433    (1)
DS$K_TYPE_COMMAND_ERR          =00000015      208    (1)  #-763    (1)    #-840    (1)
DS$K_TYPE_COMMAND_OUT          =00000016      208    (1)  #-821    (1)    #-835    (1)
```

```
DS$K_TYPE_CRD_AUTOTEST          =0000001A     208    (1)
DS$K_TYPE_DS_PROMPT             =00000001     208    (1)
DS$K_TYPE_DS_START              =0000001D     208    (1)
DS$K_TYPE_ERRDEV                =00000008     208    (1)
DS$K_TYPE_ERRHARD               =00000006     208    (1)
DS$K_TYPE_ERROR_BODY            =00000009     208    (1)
DS$K_TYPE_ERROR_END             =0000000A     208    (1)
DS$K_TYPE_ERRPREP               =0000001B     208    (1)
DS$K_TYPE_ERRSOFT               =00000007     208    (1)
DS$K_TYPE_ERRSUP                =00000004     208    (1)
DS$K_TYPE_ERRSYS                =00000005     208    (1)
DS$K_TYPE_ERR_HALT              =0000000D     208    (1)
DS$K_TYPE_EXCEPTION             =0000000C     208    (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     208    (1)
DS$K_TYPE_FIRST_PASS            =00000011     208    (1)    #-622    (1)
DS$K_TYPE_GENERAL               =00000000     208    (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     208    (1)
DS$K_TYPE_NO_TESTS              =00000012     208    (1)    #-530    (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     208    (1)
DS$K_TYPE_PROGRAM_END           =00000010     208    (1)    #-595    (1)    #-602    (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     208    (1)    #-376    (1)    #-421    (1)
DS$K_TYPE_PROGRAM_START         =0000000F     208    (1)
DS$K_TYPE_QIO_INVADP            =00000024     208    (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     208    (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     208    (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     208    (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     208    (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     208    (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     208    (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     208    (1)
DS$K_TYPE_START_ERR             =00000018     208    (1)
DS$K_TYPE_START_LIST            =00000025     208    (1)
DS$K_TYPE_SUMMARY               =0000000E     208    (1)    #-736    (1)    #-757    (1)
DS$K_TYPE_USER_PROMPT           =00000002     208    (1)
DS$M_ABRTFLG                    =00000040     204    (1)
DS$M_BADTIME                    =00100000     204    (1)
DS$M_BATCH                      =00400000     204    (1)
DS$M_BRKCLR                     =00001000     204    (1)
DS$M_BRKPT                      =00000800     204    (1)
DS$M_CHARFLG                    =00000100     204    (1)
DS$M_CMDFLG                     =00000080     204    (1)
DS$M_CTRLC                      =00000001     204    (1)
DS$M_CTRLO                      =00010000     204    (1)
DS$M_DEVFLG                     =00000200     204    (1)
DS$M_DISABLCC                   =01000000     204    (1)
DS$M_DONFLG                     =00002000     204    (1)
DS$M_ERRFLG                     =00000010     204    (1)
DS$M_EXCEPT                     =00080000     204    (1)
DS$M_EXETST                     =00040000     204    (1)
DS$M_HLTFLG                     =00000008     204    (1)
DS$M_LODFLG                     =00000002     204    (1)
DS$M_MEMMGT                     =00008000     204    (1)
DS$M_NI                         =04000000     204    (1)
DS$M_OUTPUT                     =00800000     204    (1)
DS$M_RUBFLG                     =00000020     204    (1)
DS$M_SCRIPT                     =00200000     204    (1)
DS$M_SETIMR                     =02000000     204    (1)
```

```
DS$M_STRFLG                =00000004     204    (1)
DS$M_SUBT                  =00004000     204    (1)
DS$M_SYSFLG                =00000400     204    (1)
DS$M_TIMED                 =02000000     204    (1)    #-675    (1)
DS$M_TIMED_OUT             =08000000     204    (1)
DS$M_TIMRON                =00020000     204    (1)
DS$SUMMARY                  00000000-XR                 574    (1)
DS$V_ABRTFLG               =00000006     204    (1)
DS$V_BADTIME               =00000014     204    (1)
DS$V_BATCH                 =00000016     204    (1)
DS$V_BRKCLR                =0000000C     204    (1)
DS$V_BRKPT                 =0000000B     204    (1)
DS$V_CHARFLG               =00000008     204    (1)
DS$V_CMDFLG                =00000007     204    (1)    #-610    (1)
DS$V_CTRLC                 =00000000     204    (1)    #-671    (1)
DS$V_CTRLO                 =00000010     204    (1)    #-573    (1)    #-576    (1)
DS$V_DEVFLG                =00000009     204    (1)
DS$V_DISABLCC              =00000018     204    (1)
DS$V_DONFLG                =0000000D     204    (1)    #-663    (1)    #-666    (1)
DS$V_ERRFLG                =00000004     204    (1)    #-341    (1)    #-402    (1)    #-409    (1)
DS$V_EXCEPT                =00000013     204    (1)
DS$V_EXETST                =00000012     204    (1)    #-401    (1)    #-519    (1)    #-520    (1)
DS$V_HLTFLG                =00000003     204    (1)
DS$V_LODFLG                =00000001     204    (1)
DS$V_MEMMGT                =0000000F     204    (1)
DS$V_NI                    =0000001A     204    (1)
DS$V_OUTPUT                =00000017     204    (1)
DS$V_RUBFLG                =00000005     204    (1)
DS$V_SCRIPT                =00000015     204    (1)
DS$V_SETIMR                =00000019     204    (1)
DS$V_STRFLG                =00000002     204    (1)    #-660    (1)
DS$V_SUBT                  =0000000E     204    (1)    #-383    (1)
DS$V_SYSFLG                =0000000A     204    (1)
DS$V_TIMED                 =00000019     204    (1)
DS$V_TIMED_OUT             =0000001B     204    (1)
DS$V_TIMRON                =00000011     204    (1)
DSA$GL_FLAGS                0000FE00                   308    (1)      331    (1)      371    (1)      403    (1)
                                                       430    (1)      456    (1)      606    (1)      681    (1)
DSA$GL_MSGTYP               0000FE40                  #-612    (1)
DSA$GL_PASSES               0000FE08                  #-413    (1)    #-471    (1)    #-472    (1)    #-478    (1)
                                                     #-563    (1)    #-567    (1)
DSA$GL_PASSNO               0000FE54                  #-307    (1)    #-332    (1)    #-413    (1)    #-568    (1)
                                                     #-602    (1)    #-615    (1)    #-624    (1)    #-821    (1)
                                                     #-835    (1)
DSA$GL_SECTNO               0000FE10                  #-363    (1)    #-803    (1)
DSA$GL_SUBTNO               0000FE4C                  #-319    (1)    #-329    (1)    #-382    (1)    #-394    (1)
                                                     #-585    (1)    #-677    (1)    #-821    (1)    #-835    (1)
DSA$GL_TESTNO               0000FE50                  #-318    (1)    #-325    (1)    #-349    (1)    #-357    (1)
                                                     #-391    (1)    #-410    (1)    #-470    (1)    #-500    (1)
                                                     #-511    (1)    #-676    (1)    #-821    (1)    #-835    (1)
DSA$V_PASS0                =0000001D                  #-308    (1)    #-330    (1)
DSA$V_QA                   =0000000F                  #-456    (1)    #-606    (1)
DSA$V_SEARCH               =0000000E                  #-403    (1)
DSA$V_TRACE                =0000000A                  #-371    (1)    #-430    (1)
DSA$V_USER                 =0000001C                  #-681    (1)
DSP$A_BASE                  00000000     205    (1)    #-358    (1)    #-376    (1)
DSP$A_DATA                  00000008     205    (1)    #-378    (1)
```

```
DSP$A_MSG                       00000004     205    (1)     #-376    (1)
DSP$A_TEST                      0000000C     205    (1)     #-352    (1)
DSP$K_SIZE                     =00000018     205    (1)     #-350    (1)
DSP$Q_SECTION                   00000010     205    (1)       364    (1)
DSQA$K_DISPAT_AFTER_INIT       =00000014     206    (1)     #-327    (1)
DSQA$K_DISPAT_BEFORE_INIT      =00000013     206    (1)     #-321    (1)
DSQA$K_DISPAT_CALLTEST         =00000015     206    (1)     #-340    (1)
DSQA$K_DISPAT_DONE_TESTS       =00000018     206    (1)     #-521    (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN  =00000001     206    (1)     #-561    (1)
DSQA$K_DISPAT_DSX$ENDPASS_END  =00000002     206    (1)     #-628    (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID  =00000000     206    (1)     #-605    (1)
DSQA$K_DISPAT_DS_CLEANUP_BGN   =00000003     206    (1)     #-669    (1)
DSQA$K_DISPAT_DS_CLEANUP_END   =00000004     206    (1)     #-705    (1)
DSQA$K_DUMMY_1                 =00000007     206    (1)
DSQA$K_ERROR_ERROR_BGN         =00000006     206    (1)
DSQA$K_ERROR_ERROR_END         =00000005     206    (1)
DSQA$K_KERNEL_INIT_CONTEXT     =00000008     206    (1)
DSQA$K_LOOP_DSX$BGNSUB         =00000009     206    (1)
DSQA$K_LOOP_DSX$CKLOOP         =0000000B     206    (1)
DSQA$K_LOOP_DSX$ENDSUB         =0000000A     206    (1)
DSQA$K_PARAM_DSX$GETADDRESS    =0000000E     206    (1)
DSQA$K_PARAM_DSX$GETDATA       =0000000C     206    (1)
DSQA$K_PARAM_DSX$GETLOGICAL    =0000000F     206    (1)
DSQA$K_PARAM_DSX$GETSTRING     =00000010     206    (1)
DSQA$K_PARAM_DSX$GETVIELD      =0000000D     206    (1)
DSQA$K_QA_DSX$BRANCH           =00000011     206    (1)
DSQA$K_START_BEFORE_HEADER     =00000017     206    (1)
DSQA$K_START_VRRESTART         =00000012     206    (1)
DSQA$K_START_VRSTART           =00000016     206    (1)
DSR$CHECKLOAD                   00000000-XR                743    (1)      793    (1)
DSR$FIND_USER_PC                00000000-XR                801    (1)
DSR$MMENABLE                    00000000-XR                703    (1)
DSX$DOSUMMARY                   000004A7-R   735    (1)
DSX$ENDPASS                     000002B8-R   559    (1)     414    (1)
DSX$ENDPASS_X                   000003C2-R   626    (1)
DSX$PRINT                       00000000-XR                376    (1)      421    (1)      433    (1)      530    (1)
                                                           595    (1)      602    (1)      622    (1)      757    (1)
                                                           763    (1)      821    (1)      835    (1)      840    (1)
DS_CLEANUP                      000003D2-R   659    (1)     575    (1)
DS_ERRSUP                       00000000-XR                360    (1)
ENV$M_DOMAIN                   =00000002     199    (1)
ENV$M_LEVEL                    =00000001     199    (1)
ENV$M_SUPER                    =000003FC     199    (1)
ENV$S_DOMAIN                   =00000001     199    (1)
ENV$S_LEVEL                    =00000001     199    (1)
ENV$S_SUPER                    =00000008     199    (1)
ENV$V_DOMAIN                   =00000001     199    (1)     199    (1)
ENV$V_LEVEL                    =00000000     199    (1)     199    (1)
ENV$V_SUPER                    =00000002     199    (1)
ENV$_CPU                       =00000000     199    (1)     199    (1)
ENV$_FUNCTIONAL                =00000000     199    (1)     199    (1)
ENV$_REPAIR                    =00000001     199    (1)     199    (1)
ENV$_SUPER                     =00000001     199    (1)
ENV$_SYSTEM                    =00000001     199    (1)     199    (1)
EXE$GB_CPUTYPE                  00000000-XR              #-682    (1)
FALSE                          =00000000     207    (1)
FIRSTTEST                       00000032-R   310    (1)   #-522    (1)
```

FMTLSTPAS	000000C7-R	236	(1)	602	(1)				
FMTPASONE	0000008C-R	233	(1)	622	(1)				
FMTSEARCH	00000114-R	240	(1)	421	(1)				
FMTTRACE	0000014C-R	243	(1)	376	(1)				
FMT_NODIAG	000002D6-R	265	(1)	763	(1)	840	(1)		
FMT_SHOWSTATUS	0000025C-R	259	(1)	821	(1)				
FMT_SHOWSTATUS_NO_PC	000001F1-R	253	(1)	835	(1)				
FMT_SUMMARY	0000015C-R	246	(1)	757	(1)				
INIT_CONTEXT	00000000-XR			577	(1)				
IOC$K_IOSPACE	00000000-XR			685	(1)				
KB_CHECK	00000000-XR			630	(1)	739	(1)		
L$A_CCP	00000240			679	(1)				
L$A_DTP	00000218			351	(1)				
L$A_ICP	0000023C			323	(1)				
L$A_NAME	00000208			#-757	(1)	#-821	(1)	#-835	(1)
L$A_REPP	00000244			767	(1)				
L$A_SECNAM	00000250			#-804	(1)				
L$A_TSTCNT	00000254			#-582	(1)				
L$L_ENVIRON	00000204			692	(1)				
L$L_REV	0000020C			#-757	(1)	#-821	(1)	#-835	(1)
L$L_UPDATE	00000210			#-757	(1)	#-821	(1)	#-835	(1)
MP$_DEALL_MEMORY	00000000-XR			188	(1)	707	(1)		
OFF	=00000000	207	(1)						
ON	=00000001	207	(1)						
PR$_SID_TYP730	=00000003			#-683	(1)				
QA$AOB_CHECK_STATE	00000000-XR			460	(1)	490	(1)	496	(1)
QA$AOB_FLAGS	00000000-XR			492	(1)				
QA$K_ABORT_NOW	=00000007	207	(1)						
QA$K_APT_PHASE_ONE	=0000000B	207	(1)						
QA$K_APT_PHASE_TWO	=0000000C	207	(1)						
QA$K_BAD_ADDRESS	=00000009	207	(1)						
QA$K_CONTROL_C_CONT	=0000000A	207	(1)						
QA$K_DEALLOCATION	=0000000E	207	(1)						
QA$K_ERROR_PHASE_ONE	=00000005	207	(1)						
QA$K_ERROR_PHASE_THREE	=00000007	207	(1)						
QA$K_ERROR_PHASE_TWO	=00000006	207	(1)						
QA$K_FAILURE	=00000000	207	(1)						
QA$K_FIRST_BRANCH_CODE	=00000000	207	(1)						
QA$K_FIRST_TIME	=00000005	207	(1)						
QA$K_INIT_CONTEXT_DONE	=00000003	207	(1)						
QA$K_LAST_BRANCH_CODE	=00000025	207	(1)						
QA$K_LOOP_ON_SUBTEST	=00000003	207	(1)	#-459	(1)				
QA$K_LOOP_ON_TEST	=00000002	207	(1)	#-489	(1)				
QA$K_LOOP_TEST_DONE	=00000004	207	(1)	#-491	(1)				
QA$K_MEMORY_MANAGE	=0000000D	207	(1)						
QA$K_MULTIPLE_PASS	=00000001	207	(1)						
QA$K_NODEF	=00000000	207	(1)						
QA$K_NORMAL_START	=00000000	207	(1)						
QA$K_NO_HEADER	=00000006	207	(1)						
QA$K_NUMBER_OF_CHECKS	=0000000F	207	(1)						
QA$K_NUMBER_QA_FLAGS	=00000008	207	(1)						
QA$K_NUMB_ROUTINE_STATES	=00000004	207	(1)						
QA$K_QA_DEBUG	=00000001	207	(1)						
QA$K_QA_DONE	=00000002	207	(1)						
QA$K_RUN_BACKWARDS	=00000004	207	(1)	#-495	(1)				
QA$K_SECTION	=00000008	207	(1)						
QA$K_SUCCESS	=00000001	207	(1)						

```
QA$MAIN                         00000000-XR                     321     (1)     327     (1)     340     (1)     521     (1)
                                                                561     (1)     605     (1)     628     (1)     669     (1)
                                                                705     (1)
QA$V_CHECK_DONE                 =00000003       207     (1)
QA$V_DOING_CLEANUP              =00000002       207     (1)
QA$V_ERROR_FOUND                =00000001       207     (1)
QA$V_INIT_DONE                  =00000000       207     (1)
QA$W_SAVE_LAST                   00000000-XR                  #-474     (1)
QIO$CLEANUP                      00000000-XR                    695     (1)
REQ_SUB                          00000008-R     219     (1)  #-395     (1)  #-595     (1)
RMS$CLEANUP                      00000000-XR                    698     (1)
SCB_BASE                         00000000-XR                    686     (1)
SEP_FUNCTIONAL                  =00000002       199     (1)  #-693     (1)
SEP_REPAIR                      =00000003       199     (1)
SIZ...                          =00000001       204     (1)    199     (1)      204     (1)
SYS$CANTIM                       00000000-XR                    674     (1)
TRUE                            =00000001       207     (1)
TST$MCHK                         00000000-XR                    688     (1)
T_NOEXE                          0000006B-R     230     (1)    530     (1)
T_NOSUB                          00000009-R     227     (1)    595     (1)
T_TSTABO                         000002F2-R     268     (1)    433     (1)
VRRESTART                        00000000-XR                    607     (1)
VRSHOWSTATUS                     00000531-R     792     (1)
VRSUMMARY                        000004B9-R     742     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CANTIM_S	1	674 (1)	674 (1)
$D1_PRINT_S	1	376 (1)	376 (1) 421 (1) 433 (1) 530 (1)
			595 (1) 602 (1) 622 (1) 757 (1)
			763 (1) 821 (1) 835 (1) 840 (1)
$DEF	1	208 (1)	
$DEFINI	1	198 (1)	198 (1) 200 (1) 201 (1)
$DS_CNTRLC_S	1	673 (1)	673 (1)
$DS_DSADEF	5	200 (1)	200 (1)
$DS_ENVDEF	2	199 (1)	199 (1)
$DS_HDRDEF	2	201 (1)	201 (1)
$DS_QADEF	2	207 (1)	207 (1)
$DS_SUMMARY_S	1	574 (1)	574 (1)
$DS_TYPEDEF	4	208 (1)	208 (1)
$EQU	1	208 (1)	199 (1) 206 (1) 207 (1) 208 (1)
$EQULS1	1	208 (1)	199 (1) 206 (1) 207 (1) 208 (1)
$EQULST	1	199 (1)	199 (1) 206 (1) 207 (1) 208 (1)
$GBLINI	2		199 (1) 204 (1) 206 (1) 207 (1)
			208 (1)
$PRDEF	5	198 (1)	198 (1)
$PRINT	2	372 (1)	372 (1) 417 (1) 431 (1) 528 (1)
			590 (1) 597 (1) 618 (1) 746 (1)
			761 (1) 809 (1) 824 (1) 838 (1)
$PUSHADR	1	360 (1)	360 (1) 673 (1)
$PUSHTWO	1	674 (1)	674 (1)
$VIELD	1	199 (1)	199 (1) 204 (1)
$VIELD1	1	208 (1)	199 (1) 204 (1)
APTDEF	1	202 (1)	202 (1)
BR_IF_NOT_DONFLG	1	663 (1)	663 (1)
BR_IF_NOT_ERRFLG	1	402 (1)	402 (1)
BR_IF_NOT_EXETST	1	519 (1)	519 (1)
BR_IF_NOT_QA	1	606 (1)	606 (1)
BR_IF_NOT_SEARCH	1	403 (1)	403 (1)
BR_IF_NOT_TRACE	1	371 (1)	371 (1) 430 (1)
BR_IF_QA	1	456 (1)	456 (1)
BR_IF_STRFLG	1	660 (1)	660 (1)
BR_IF_USER	1	681 (1)	681 (1)
CLEAR_CTRLC	1	671 (1)	671 (1)
CLEAR_CTRLO	1	573 (1)	573 (1) 576 (1)
CLEAR_ERRFLG	1	341 (1)	341 (1) 409 (1)
CLEAR_EXETST	1	520 (1)	520 (1)
CLEAR_SUBT	1	383 (1)	383 (1)
CLIDEF	4	203 (1)	203 (1)
DSFDEF	3	204 (1)	204 (1)
DSPDEF	1	205 (1)	205 (1)
DSQA	2	206 (1)	206 (1)
DS_QADEFS	6	207 (1)	207 (1)
ERRSUP_S	1	360 (1)	360 (1)
MODNAM	1	225 (1)	225 (1)
QA_MAIN	1	321 (1)	321 (1) 327 (1) 340 (1) 521 (1)
			561 (1) 605 (1) 628 (1) 669 (1)

				705	(1)
SET_CMDFLG	1	610	(1)	610	(1)
SET_DONFLG	1	666	(1)	666	(1)
SET_EXETST	1	401	(1)	401	(1)
SET_PASS0	1	308	(1)	308	(1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ACBL	00F1	411 (1) 498 (1) 510 (1)
ACBW	003D	474 (1)
ADDL3	00C1	803 (1)
BBC	00E1	371 (1) 402 (1) 403 (1) 430 (1) 459 (1) 489 (1) 495 (1)
		519 (1) 606 (1) 663 (1)
BBCC	00E5	330 (1)
BBS	00E0	363 (1) 456 (1) 491 (1) 660 (1) 681 (1)
BBSC	00E4	341 (1) 383 (1) 409 (1) 520 (1) 573 (1) 576 (1) 671 (1)
BBSS	00E2	308 (1) 401 (1) 610 (1) 666 (1)
BEQL	0013	359 (1) 444 (1) 447 (1) 580 (1) 700 (1) 807 (1)
BGEQU	001E	586 (1)
BGTR	0014	799 (1)
BICL2	00CA	675 (1)
BLBC	00E9	744 (1)
BLBS	00E8	429 (1) 794 (1)
BLEQU	001B	569 (1)
BNEQ	0012	354 (1) 393 (1) 564 (1) 584 (1) 616 (1) 684 (1) 694 (1)
BRB	0011	434 (1) 476 (1) 487 (1) 502 (1)
BRW	0031	355 (1) 361 (1) 365 (1) 448 (1) 457 (1) 493 (1) 522 (1)
		565 (1) 570 (1) 661 (1) 664 (1) 795 (1)
BSBB	0010	738 (1) 758 (1)
CALLG	00FA	389 (1) 695 (1)
CALLS	00FB	321 (1) 323 (1) 327 (1) 340 (1) 360 (1) 376 (1) 414 (1)
		421 (1) 433 (1) 521 (1) 530 (1) 561 (1) 574 (1) 595 (1)
		602 (1) 605 (1) 622 (1) 628 (1) 669 (1) 673 (1) 674 (1)
		679 (1) 697 (1) 705 (1) 707 (1) 757 (1) 763 (1) 767 (1)
		801 (1) 821 (1) 835 (1) 840 (1)
CLRB	0094	768 (1)
CLRL	00D4	301 (1) 302 (1) 303 (1) 304 (1) 305 (1) 306 (1) 307 (1)
		318 (1) 329 (1) 381 (1) 382 (1) 676 (1) 689 (1)
CLRQ	007C	674 (1)
CMPB	0091	357 (1) 682 (1)
CMPL	00D1	391 (1) 567 (1) 583 (1) 585 (1) 615 (1) 798 (1) 806 (1)
CMPZV	00ED	692 (1)
CVTBL	0098	376 (1) 421 (1) 433 (1) 530 (1) 595 (1) 602 (1) 622 (1)
		757 (1) 763 (1) 821 (1) 835 (1) 840 (1)
INCL	00D6	624 (1)
JMP	0017	607 (1) 613 (1)
JSB	0016	575 (1) 577 (1) 630 (1) 698 (1) 703 (1) 739 (1) 743 (1)
		793 (1)
MOVAB	009E	351 (1) 685 (1) 688 (1)
MOVAL	00DE	445 (1) 479 (1) 686 (1)
MOVB	0090	736 (1)
MOVL	00D0	319 (1) 324 (1) 332 (1) 352 (1) 378 (1) 394 (1) 410 (1)
		413 (1) 442 (1) 443 (1) 470 (1) 471 (1) 472 (1) 478 (1)
		480 (1) 481 (1) 483 (1) 484 (1) 485 (1) 486 (1) 582 (1)
		611 (1) 677 (1) 701 (1) 797 (1) 804 (1)
MOVZBL	009A	376 (1) 421 (1) 433 (1) 530 (1) 595 (1) 602 (1) 622 (1)
		699 (1) 757 (1) 763 (1) 821 (1) 835 (1) 840 (1)
MULL2	00C4	350 (1)

```
POPL            8ED0    690     (1)
POPR            00BA    708     (1)
PUSHAB          009F    376     (1)     421     (1)     433     (1)     530     (1)     595     (1)     602     (1)     622     (1)
                        757     (1)     763     (1)     821     (1)     835     (1)     840     (1)
PUSHAL          00DF    360     (1)
PUSHL           00DD    321     (1)     327     (1)     340     (1)     360     (1)     376     (1)     421     (1)     521     (1)
                        561     (1)     595     (1)     602     (1)     605     (1)     622     (1)     628     (1)     669     (1)
                        673     (1)     687     (1)     705     (1)     757     (1)     821     (1)     835     (1)
PUSHR           00BB    667     (1)
RET             0004    631     (1)     740     (1)
RSB             0005    533     (1)     710     (1)     759     (1)     764     (1)     769     (1)     822     (1)     836     (1)
                        841     (1)
SUBL3           00C3    349     (1)
TSTL            00D5    446     (1)     563     (1)     579     (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	225	(1)	228	(1)	231	(1)	234	(1)	237	(1)	241	(1)	244	(1)	247	(1)
	254	(1)	260	(1)	266	(1)	269	(1)								
.BLKL	203	(1)	205	(1)	217	(1)	218	(1)	219	(1)						
.END	842	(1)														
.ENDC	199	(1)	204	(1)	206	(1)	207	(1)	208	(1)	321	(1)	327	(1)	340	(1)
	360	(1)	376	(1)	421	(1)	433	(1)	521	(1)	530	(1)	561	(1)	595	(1)
	602	(1)	605	(1)	622	(1)	628	(1)	669	(1)	673	(1)	674	(1)	705	(1)
	757	(1)	763	(1)	821	(1)	835	(1)	840	(1)						
.ENDM	198	(1)	199	(1)	200	(1)	201	(1)	202	(1)	203	(1)	204	(1)	205	(1)
	206	(1)	207	(1)	208	(1)	225	(1)	308	(1)	321	(1)	341	(1)	360	(1)
	371	(1)	372	(1)	376	(1)	383	(1)	401	(1)	402	(1)	403	(1)	456	(1)
	519	(1)	520	(1)	573	(1)	574	(1)	606	(1)	610	(1)	660	(1)	663	(1)
	666	(1)	671	(1)	673	(1)	674	(1)	681	(1)						
.ENDR	199	(1)	204	(1)	206	(1)	207	(1)	208	(1)	376	(1)	421	(1)	433	(1)
	530	(1)	595	(1)	602	(1)	622	(1)	757	(1)	763	(1)	821	(1)	835	(1)
	840	(1)														
.ENTRY	559	(1)	735	(1)												
.EXTRN	188	(1)														
.GLOBL	674	(1)														
.IDENT	8	(1)														
.IF	199	(1)	204	(1)	206	(1)	207	(1)	208	(1)	321	(1)	327	(1)	340	(1)
	360	(1)	376	(1)	421	(1)	433	(1)	521	(1)	530	(1)	561	(1)	595	(1)
	602	(1)	605	(1)	622	(1)	628	(1)	669	(1)	673	(1)	674	(1)	705	(1)
	757	(1)	763	(1)	821	(1)	835	(1)	840	(1)						
.IFF	199	(1)	204	(1)	206	(1)	207	(1)	208	(1)	360	(1)	673	(1)	674	(1)
.IF_FALSE	321	(1)	327	(1)	340	(1)	521	(1)	561	(1)	605	(1)	628	(1)	669	(1)
	705	(1)														
.IIF	199	(1)	204	(1)	206	(1)	207	(1)	208	(1)	360	(1)				
.IRP	199	(1)	204	(1)	206	(1)	207	(1)	208	(1)	376	(1)	421	(1)	433	(1)
	530	(1)	595	(1)	602	(1)	622	(1)	757	(1)	763	(1)	821	(1)	835	(1)
	840	(1)														
.LIBRARY	181	(1)	182	(1)	183	(1)										
.LIST	200	(1)	201	(1)	203	(1)	205	(1)								
.MACRO	199	(1)	204	(1)	206	(1)	207	(1)	208	(1)						
.MDELETE	199	(1)	207	(1)												
.NLIST	200	(1)	201	(1)	203	(1)	205	(1)								
.NOCROSS	198	(1)	200	(1)	201	(1)										
.NOSHOW	9	(1)														
.PAGE	175	(1)	209	(1)	221	(1)	239	(1)	252	(1)	270	(1)	299	(1)	333	(1)
	366	(1)	404	(1)	41	(1)	422	(1)	449	(1)	488	(1)	512	(1)	534	(1)
	558	(1)	604	(1)	632	(1)	658	(1)	680	(1)	711	(1)	734	(1)	760	(1)
	770	(1)	791	(1)	87	(1)										
.PSECT	203	(1)	205	(1)	211	(1)	223	(1)	272	(1)						
.RESTORE	198	(1)	200	(1)	201	(1)	203	(1)	205	(1)						
.SAVE	198	(1)	200	(1)	201	(1)	203	(1)	205	(1)						
.SBTTL	176	(1)	271	(1)	535	(1)	633	(1)	712	(1)	771	(1)				
.SUBTITLE	210	(1)	222	(1)												
.TITLE	7	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	12	#-349	(1)	#-350	(1)	351	(1)	#-352	(1)	#-358	(1)	364	(1)
		#-376	(1)	#-378	(1)	695	(1)	#-797	(1)				
R0	22	#-429	(1)	#-442	(1)	#-443	(1)	445	(1)	#-471	(1)	#-478	(1)
		#-479	(1)	#-480	(1)	#-483	(1)	#-485	(1)	#-667	(1)	#-685	(1)
		#-689	(1)	#-699	(1)	#-701	(1)	#-708	(1)	#-744	(1)	#-794	(1)
		#-797	(1)	#-798	(1)	#-806	(1)	#-821	(1)				
R1	21	#-443	(1)	445	(1)	#-470	(1)	#-474	(1)	#-483	(1)	#-484	(1)
		#-485	(1)	#-486	(1)	#-582	(1)	#-583	(1)	#-667	(1)	#-686	(1)
		#-687	(1)	#-688	(1)	#-690	(1)	#-708	(1)	#-803	(1)	#-804	(1)
		#-821	(1)	#-835	(1)								
R2	3	559	(1)	#-667	(1)	#-708	(1)						
SP	25	#-376	(1)	#-421	(1)	#-433	(1)	#-530	(1)	#-595	(1)	#-602	(1)
		#-622	(1)	#-674	(1)	#-757	(1)	#-763	(1)	#-821	(1)	#-835	(1)
		#-840	(1)										

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.21
Command processing	888	00:00:00.22	00:00:00.74
Pass 1	939	00:00:03.00	00:00:06.52
Symbol table sort	0	00:00:00.15	00:00:00.14
Pass 2	58	00:00:00.81	00:00:01.66
Symbol table output	2	00:00:00.07	00:00:00.25
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	8	00:00:00.44	00:00:00.75
Assembler run totals	1921	00:00:04.72	00:00:10.28

The working set limit was 2900 pages.
172435 bytes (337 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 568 non-local and 48 local symbols.
842 source lines were read in Pass 1, producing 62 object records in Pass 2.
119 pages of virtual memory were used to define 48 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              8
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6               28
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       9
TOTALS (all libraries)                                45

813 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DISPAT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(2) 49 DECLARATIONS
(3) 135 RL01/2 Bootstrap driver code

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DLBTDRIVR.MAR
0000     2 ; *1     6-FEB-1986 15:17:30 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element DLBTDRIVR.MAR
0000     4         .TITLE  DLBTDRIVR - RL01/2 BOOT DRIVER
0000     5         .IDENT  'V02-002'
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1979, 1980                                                 *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    12 ;*                                                                           *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    18 ;*  TRANSFERRED.                                                             *
0000    19 ;*                                                                           *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    22 ;*  CORPORATION.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28 ;
0000    29
0000    30 ;++
0000    31 ; FACILITY:     BOOTS
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       This module contains the bootstrap device driver for
0000    35 ;       the RL01/2  disks.
0000    36 ;
0000    37 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    38 ;
0000    39 ; AUTHOR:  Steve Beckhardt,     CREATION DATE:  31-Oct-1979
0000    40 ;  (Original author:  Charlie Franks)
0000    41 ;
0000    42 ; MODIFIED BY:
0000    43 ;
0000    44 ;       02-02   CAS0001         C.A. Samuelson          30-Apr-1980
0000    45 ;                               Change interface to BOOTDRIVR for purge of UBA datapath
0000    46 ;
0000    47 ;--
```

```
                     0000     49          .SBTTL  DECLARATIONS
                     0000     50 ;
                     0000     51 ; INCLUDE FILES:
                     0000     52 ;
                     0000     53
                     0000     54          $BTDDEF                             ; Boot device types
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     55          $IODEF                              ; I/O function codes
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     56          $RPBDEF                             ; RPB offsets
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     57          $SSDEF                              ; Status codes
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     58          $UBADEF                             ; UBA definitions
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     59          $UBIDEF                             ; 11/750 UBA definitions
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     60
                     0000     61 ;
                     0000     62 ; MACROS:
                     0000     63 ;
                     0000     64
                     0000     65 ;
                     0000     66 ; EQUATED SYMBOLS:
                     0000     67 ;
                     0000     68 ;
                     0000     69 ; RL11/RL02 CONTROLLER REGISTER OFFSETS
                     0000     70 ;
                     0000     71
                     0000     72          $DEFINI RL                          ;START OF REGISTER DEFINITIONS
                     0000                 .SAVE   LOCAL_BLOCK
                     0000     73
                     0000     74 $DEF     RL_CS           .BLKW   1           ;CONTROL STATUS REGISTER (CSR)
                     0000                 .IIF    NB,RL_CS,       RL_CS:
           00000002  0000                 .IIF    NB,,BLKW,       .BLKW   1
                     0002     75          _VIELD  RL_CS,0,<-                  ;START OF CSR BIT DEFINITIONS
                     0002     76                  <DRDY,,M>,-                 ; DRIVER READY
                     0002     77                  <FCODE,3>,-                 ; FUNCTION CODE
                     0002     78                  <XBA,2>,-                   ; BUS ADDRESS EXTENSION BITS
                     0002     79                  <IE,,M>,-                   ; INTERRUPT ENABLE
                     0002     80                  <CRDY,,M>,-                 ; CONTROLLER READY
                     0002     81                  <DS,2>,-                    ; DRIVE SELECT
                     0002     82                  <OPI,,M>,-                  ; OPERATION INCOMPLETE
                     0002     83                  <CRC,,M>,-                  ; DATA CRC OR HEADER CRC
                     0002     84                  <DLT,,M>,-                  ; DATA LATE OR HEADER NOT FOUND
                     0002     85                  <NXM,,M>,-                  ; NON-EXISTENT MEMORY
                     0002     86                  <DE,,M>,-                   ; DRIVE ERROR
                     0002     87                  <CE,,M>-                    ; COMPOSITE ERROR
                     0002     88          >                                   ;END CSR BIT DEFINITIONS
                     0002     89
                     0002     90 $DEF     RL_BA           .BLKW   1           ;BUS ADDRESS REGISTER (BAR)
```

```
                0002                .IIF    NB,RL_BA,       RL_BA:
        00000004 0002                .IIF    NB,.BLKW,       .BLKW   1
                0004    91
                0004    92 $DEF     RL_DA           .BLKW   1       ;DISK ADDRESS REGISTER (DAR)
                0004                .IIF    NB,RL_DA,       RL_DA:
        00000006 0004                .IIF    NB,.BLKW,       .BLKW   1
                0006    93          _VIELD  RL_DA,0,<-              ;START OF DAR BIT DEFINITIONS
                0006    94                  <MRK,,M>,-              ; MARK (ALWAYS 1)
                0006    95                  <STS,,M>,-              ; GET STATUS
                0006    96                  <,1>,-                  ; RESERVED BIT
                0006    97                  <RST,,M>,-              ; RESET
                0006    98                  <,12>-                  ; RESERVED BITS
                0006    99          >                               ;END OF DAR BIT DEFINITIONS
                0006   100
                0006   101 $DEF     RL_MP           .BLKW   1       ;MULTIPURPOSE REGISTER (MPR)
                0006                .IIF    NB,RL_MP,       RL_MP:
        00000008 0006                .IIF    NB,.BLKW,       .BLKW   1
                0008   102          _VIELD  RL_MP,0,<-              ;START OF MPR BIT DEFINITIONS
                0008   103                  <STA,3>,-               ; DRIVE STATE
                0008   104                  <BH,,M>,-               ; BRUSH HOME
                0008   105                  <HO,,M>,-               ; HEADS OUT
                0008   106                  <CO,,M>,-               ; COVER OPEN
                0008   107                  <HS,,M>,-               ; HEAD SELECT
                0008   108                  <TYP,,M>,-              ; DRIVE TYPE
                0008   109                  <DSE,,M>,-              ; DRIVE SELECT ERROR
                0008   110                  <VC,,M>,-               ; VOLUME CHECK
                0008   111                  <WGE,,M>,-              ; WRITE GATE ERROR
                0008   112                  <SPE,,M>,-              ; SPIN ERROR
                0008   113                  <SKTO,,M>,-             ; SEEK TIME OUT
                0008   114                  <WL,,M>,-               ; WRITE LOCK
                0008   115                  <CHE,,M>,-              ; CURRENT HEAD ERROR
                0008   116                  <WDE,,M>-               ; WRITE DATA ERROR
                0008   117          >                               ;END OF MPR BIT DEFINITIONS
                0008   118
                0008   119          $DEFEND RL                      ;END RL11/RL02 REGISTER DEFINITIONS
                0000                .RESTORE
                0000   120
                0000   121
                0000   122 ;
                0000   123 ; OWN STORAGE:
                0000   124 ;
                0000   125
                0000   126 ;
                0000   127 ; Boot driver table entry
                0000   128 ;
                0000   129
                0000   130          $BOOT_DRIVER    DEVTYPE = BTD$K_DL,-    ; Device type (DL)
                0000   131                          SIZE = DL_DRVSIZ,-      ; Driver size
                0000   132                          ADDR = DL_DRIVER,-      ; Driver address
                0000   133                          DRIVRNAME = DLNAME      ; Driver file name
            00000000                .PSECT  BOOTDRIVR_4
           FFFF 0000                .WORD   -1
           0002 0002                .WORD   BTD$K_DL
        00000000 0004                .LONG   0
        00000120' 0008               .LONG   DL_DRVSIZ
        00000000' 000C               .LONG   DL_DRIVER-$TABLE
        00000000 0010                .LONG   0
```

```
                    00000113' 0014          .LONG   DLNAME-DL_DRIVER
                          00000000          .PSECT  BOOTDRIVR_2
```

```
                                0000          .SBTTL  RL01/2 Bootstrap driver code
                                0000
                                0000  ;++
                                0000  ;
                                0000  ; Inputs:
                                0000  ;
                                0000  ;       R3      - base address of adapter's register space
                                0000  ;       R5      - LBN FOR CURRENT PIECE OF TRANSFER
                                0000  ;       R6      - contains 0
                                0000  ;       R7      - address of device's CSR
                                0000  ;       R8      - SIZE OF TRANSFER IN BYTES
                                0000  ;       R9      - address of the RPB
                                0000  ;       R10     - starting address of transfer (byte offset in first
                                0000  ;                 page ORed with starting map register number)
                                0000  ;
                                0000  ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                                0000  ;       BUF(AP) - Buffer address
                                0000  ;       SIZE(AP)- Size of transfer
                                0000  ;
                                0000  ; Implicit inputs:
                                0000  ;
                                0000  ;       RPB$W_UNIT      - RPB field containing boot device unit number
                                0000  ;
                                0000  ; Outputs:
                                0000  ;
                                0000  ;       R0 - status code
                                0000  ;
                                0000  ;               SS$_NORMAL      - successful transfer
                                0000  ;               SS$_CTRLERR     - fatal controller error
                                0000  ;
                                0000  ;       R3 - must be preserved
                                0000  ;
                                0000  ;       This routine destroys R1, R2, R4, R5, R6. Within the
                                0000  ;       routine, register usage is as follows:
                                0000
                                0000  ;--
                                0000
                    00000004    0000  BUF = 4
                    00000008    0000  SIZE = 8
                    00000010    0000  FUNC = 16
                                0000
                                0000  DL_DRIVER:
                                0000
                                0000  ;
                                0000  ; RESET DRIVE AND WAIT FOR IT TO SPIN UP.
                                0000  ;
                                0000
                    50    D4    0000          CLRL    R0                      ; CLEAR R0
50    02    08    64 A9    F0    0002          INSV    RPB$W_UNIT(R9),#8,#2,R0 ; GET UNIT NUMBER
            04 A7    0B    B0    0008          MOVW    #RL_DA_M_RST!-          ; PUT RESET & GET STATUS IN DAR
                                000C                  RL_DA_M_STS!-
                                000C                  RL_DA_M_MRK,RL_DA(R7)
      67    50    04    A9    000C  10$:    BISW3   #4,R0,RL_CS(R7)         ; EXECUTE DRIVE RESET
                  00F8    30    0010          BSBW    READY                   ; WAIT FOR CONTROLLER READY
            06 A7    1D    B3    0013          BITW    #RL_MP_M_HO!-           ; HEADS,BRUSHES,STATE OK?
                                0017                  RL_MP_M_BH!5,RL_MP(R7)  ; ... (5 = SEEK LINEAR MODE STATE)
                    F3    13    0017          BEQL    10$                     ; BRANCH IF NOT: WAIT FOR DRIVE TO SPIN UP
```

```
                    67   01   B3  0019   192          BITW    #RL_CS_M_DRDY,RL_CS(R7) ; IS DRIVE READY?
                         EE   13  001C   193          BEQL    10$                     ; IF NOT, BRANCH TO WAIT FOR DRIVE READY
                                 001E   194
                                 001E   195 ;
                                 001E   196 ; FIND CURRENT DISK ADDRESS, CALCULATE CYLINDER DIFFERENCE, AND SEEK
                                 001E   197 ; DESIRED CYLINDER
                                 001E   198 ;
                                 001E   199
                67   50   08    A9  001E   200 20$:     BISW3   #8,R0,RL_CS(R7)         ; EXECUTE READ HEADER
                          00E6   30  0022   201          BSBW    READY                   ; WAIT FOR CONTROLLER READY
                             03   18  0025   202          BGEQ    30$                     ; BRANCH IF NO ERROR READING HEADER
                          00C8   31  0027   203          BRW     100$                    ; OTHERWISE, BRANCH TO ERROR HANDLING
          51    06 A7    3F    AB  002A   204 30$:     BICW3   #^077,RL_MP(R7),R1      ; GET CURRENT CYL & SURFACE
                                 002F   205
                                 002F   206 ;
                                 002F   207 ; NOW CONVERT LOGICAL TO PHYSICAL
                                 002F   208 ;
                                 002F   209
                    55   02   C4  002F   210          MULL    #2,R5                   ; CONVERT LOGICAL BLOCKS TO SECTORS
                         56   D4  0032   211          CLRL    R6                      ; CLEAR HIGH PART OF DIVIDEND
   54   56   55   00000050 8F    7B  0034   212          EDIV    #80,R5,R6,R4            ; R6 = DESIRED CYL = LBN/(SECTORS/CYL)
                                 003D   213                                          ; R4 = REMAINING SECTORS
                         55   D4  003D   214          CLRL    R5                      ; CLEAR HIGH PART OF DIVIDEND
             54   55   54   28  7B  003F   215          EDIV    #40,R4,R5,R4            ; R5 = DESIRED SURFACE = R5/(SECT/SUR)
                                 0044   216                                          ; R4 = DESIRED SECTOR
             56   56   07   78  0044   217          ASHL    #7,R6,R6                ; SHIFT DESIRED CYLINDER INTO R6<15:7>
             56   01   06   55  F0  0048   218          INSV    R5,#6,#1,R6             ; INSERT DESIRED SURFACE BIT INTO R6<6>
                    51   56   B1  004D   219          CMPW    R6,R1                   ; IS A SEEK NEEDED?
                         2E   13  0050   220          BEQL    50$                     ; BRANCH IF NOT.
                                 0052   221
                                 0052   222 ;
                                 0052   223 ; NEED TO PERFORM A SEEK.
                                 0052   224 ;
                                 0052   225
          51   007F 8F    AA  0052   226          BICW    #^O177,R1               ; ISOLATE CURRENT CYLINDER IN R1
   52     56   007F 8F    AB  0057   227 35$:     BICW3   #^O177,R6,R2            ; ISOLATE DESIRED CYLINDER IN R2
                51   52   A2  005D   228          SUBW    R2,R1                   ; SUBTRACT DESIRED FROM ACTUAL
                     08   13  0060   229          BEQL    40$                     ; BRANCH IF CURRENT = DESIRED CYLINDER
                     06   1E  0062   230          BCC     40$                     ; BRANCH IF CURRENT >= DESIRED CYLINDER
                51   51   AE  0064   231          MNEGW   R1,R1                   ; ACTUAL<DESIRED, MAKE POSITIVE DIFF
                51   04   A8  0067   232          BISW    #4,R1                   ; SET SIGN FOR MOVE TO CENTER OF DISK
   51   01   04   55   F0  006A   233 40$:     INSV    R5,#4,#1,R1             ; INSERT SURFACE BIT IN R1<4>
     04 A7   51   01   A9  006F   234          BISW3   #RL_DA_M_MRK,R1,-       ; SET MARKER AND LOAD DIFFERENCE
                                 0074   235                  RL_DA(R7)          ; WORD.
          67   50   06   A9  0074   236          BISW3   #6,R0,RL_CS(R7)         ; EXECUTE SEEK FUNCTION
                    0090   30  0078   237          BSBW    READY                   ; WAIT FOR CONTROLLER READY OR ERROR
                       03   18  007B   238          BGEQ    50$                     ; BRANCH IF NO ERROR DURING SEEK
                    0072   31  007D   239          BRW     100$                    ; OTHERWISE, BRANCH DUE TO ERROR
                                 0080   240
                                 0080   241 ;
                                 0080   242 ; SEEK, IF ANY, IS COMPLETE. EXECUTE TRANSFER FUNCTION
                                 0080   243 ;
                                 0080   244
   56   06   00   54   F0  0080   245 50$:     INSV    R4,#0,#6,R6             ; MERGE SECTOR WITH CYLINDER AND SURFACE
                                 0085   246                                          ; CYL=R6<15:7> SUR=R6<6> SEC=R6<5:0>
        04 A7   56   B0  0085   247          MOVW    R6,RL_DA(R7)            ; SET DESIRED DISK ADDRESS
        02 A7   5A   B0  0089   248          MOVW    R10,RL_BA(R7)           ; SET BUFFER ADDRESS
```

```
                52   58   B0  008D          MOVW    R8,R2                    ; GET WORKING COPY OF BYTES LEFT TO XFER
                                0090                                          ; AND ASSUME THIS IS LAST TRANSFER
          51    28   54   A3  0090          SUBW3   R4,#40,R1                ; R1 = SECTORS LEFT ON SURFACE
          51    0100 8F   A4  0094          MULW    #256,R1                  ; CONVERT TO BYTES LEFT ON SURFACE
                51   52   B1  0099          CMPW    R2,R1                    ; ARE ADDITIONAL TRANSFERS REQUIRED?
                     03   1B  009C          BLEQU   60$                      ; BRANCH IF ANSWER IS NO.
                52   51   B0  009E          MOVW    R1,R2                    ; SET BYTE COUNT FOR THIS TRANSFER
                52   02   A6  00A1  60$:    DIVW    #2,R2                    ; CALCULATE TRANSFER WORD COUNT
             06 A7   52   AE  00A4          MNEGW   R2,RL_MP(R7)             ; SET NEG TRANSFER WORD COUNT
                51   0C   B0  00A8          MOVW    #^XC,R1                  ; ASSUME READ FUNCTION
             20   10 AC   D1  00AB          CMPL    FUNC(AP),#IO$_WRITELBLK  ; IS IT A WRITE FUNCTION?
                     03   12  00AF          BNEQ    70$                      ; BRANCH IF NOT
                51   0A   B0  00B1          MOVW    #^XA,R1                  ; SET WRITE FUNCTION CODE
          67    51   50   A9  00B4  70$:    BISW3   R0,R1,RL_CS(R7)          ; MERGE UNIT # WITH FUNCTION AND EXECUTE
                   0050   30  00B8          BSBW    READY                    ; WAIT FOR CONTROLLER READY OR ERROR
                     35   19  00BB          BLSS    100$                     ; BRANCH ON ERROR
                52   02   A4  00BD  80$:    MULW    #2,R2                    ; FIND BYTES TRANSFERRED THIS TIME
                58   52   A2  00C0          SUBW    R2,R8                    ; UPDATE BYTES LEFT TO TRANSFER
                     1C   13  00C3          BEQL    90$                      ; BRANCH IF TRANSFER IS COMPLETE
                                00C5
                                00C5  ;
                                00C5  ; UPDATE PARAMETERS FOR NEXT TRANSFER
                                00C5  ;
                                00C5
  51   04 A7   007F 8F   AB  00C5          BICW3   #^O177,RL_DA(R7),R1      ; UPDATE CURRENT CYLINDER IN R1<15:7>
       56   04 A7   3F   A9  00CC          BISW3   #^O77,RL_DA(R7),R6       ; SET SECTOR BITS TO 1'S AND -
                     56   B6  00D1          INCW    R6                       ; UPDATE DESIRED DISK ADDRESS IN R6
     55   56   01   06   EF  00D3          EXTZV   #6,#1,R6,R5              ; UPDATE DESIRED SURFACE IN R5
                     54   D4  00D8          CLRL    R4                       ; UPDATE DESIRED SECTOR IN R4
             5A   02 A7   B0  00DA          MOVW    RL_BA(R7),R10            ; UPDATE DESIRED BUFFER ADDRESS
                   FF76   31  00DE          BRW     35$                      ; LOOP FOR NEXT TRANSFER
                                00E1
                                00E1  ;
                                00E1  ; TRANSFER COMPLETE - RETURN
                                00E1  ;
                                00E1
             51   08 AC   3C  00E1  90$:    MOVZWL  SIZE(AP),R1              ; SET TOTAL BYTES TRANSFERRED
                     07   12  00E5          BNEQ    95$                      ; BRANCH IF ORIGINAL SIZE WAS TRANSFERRED
       51   00008000 8F   D0  00E7          MOVL    #^X8000,R1               ; ELSE SIZE WAS FORCED TO 64K
                50   01   3C  00EE  95$:    MOVZWL  #SS$_NORMAL,R0           ; SET COMPLETION CODE
                          05  00F1          RSB                              ; AND RETURN
                                00F2
                                00F2  ;
                                00F2  ; RETRY ERROR
                                00F2  ;
                                00F2
             58   08 AC   3C  00F2  100$:   MOVZWL  SIZE(AP),R8              ; RESTORE BYTE SIZE OF TRANSFER IN R8
                     07   12  00F6          BNEQ    110$                     ; BRANCH IF SIZE WAS LEGAL
       58   00001F40 8F   D0  00F8          MOVL    #8000,R8                 ; ELSE FORCE TO 64K SIZE
  5A   04 AC   09   00   EF  00FF  110$:   EXTZV   #0,#9,BUF(AP),R10        ; RESTORE BYTE OFFSET IN R10
          50   0054 8F   3C  0105          MOVZWL  #SS$_CTRLERR,R0          ; SET FATAL CONTROLLER ERROR
                          05  010A          RSB                              ; AND ATTEMPT RETRY
                                010B
                                010B
                                010B  ;
                                010B  ; SUBROUTINE TO WAIT FOR CONTROLLER READY OR ERROR
                                010B  ;
```

```
                                    010B   306
                                    010B   307 READY:                                  ;
                 67   8080 8F   B3  010B   308         BITW    #^X8080,(R7)            ;CONTROLLER READY OR ERROR?
                           F9   13  0110   309         BEQL    READY                   ;IF EQL NO
                                05  0112   310         RSB                             ;
                                    0113   311
58 45 2E 52 45 56 49 52 44 4C 44 00' 0113   312 DLNAME: .ASCIC  /DLDRIVER.EXE/          ; Driver file name
                                45  011F
                                0C  0113
                                    0120   313
                         00000120   0120   314 DL_DRVSIZ=.-DL_DRIVER
                                    0120   315
                                    0120   316         .END
```

```
$TABLE                          = 00000000 R  D  02
BTD$K_DL                        = 00000002    D
BUF                             = 00000004    D
DLNAME                            00000113 R  D  03
DL_DRIVER                         00000000 R  D  03
DL_DRVSIZ                       = 00000120    D
FUNC                            = 00000010    D
IO$_WRITELBLK                   = 00000020    D
READY                             0000010B R  D  03
RL_BA                             00000002    D
RL_CS                             00000000    D
RL_CS_M_DRDY                    = 00000001    D
RL_DA                             00000004    D
RL_DA_M_MRK                     = 00000001    D
RL_DA_M_RST                     = 00000008    D
RL_DA_M_STS                     = 00000002    D
RL_MP                             00000006    D
RL_MP_M_BH                      = 00000008    D
RL_MP_M_HO                      = 00000010    D
RPB$W_UNIT                      = 00000064    D
SIZ...                          = 00000001    D
SIZE                            = 00000008    D
SS$_CTRLERR                     = 00000054    D
SS$_NORMAL                      = 00000001    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000008 (8.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000120 (288.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	133 (2)	133 (2)
BTD$K_DL	=00000002		133 (2)
BUF	=00000004	172 (3)	298 (3)
DLNAME	00000113-R	312 (3)	133 (2)
DL_DRIVER	00000000-R	176 (3)	133 (2) 314 (3)
DL_DRVSIZ	=00000120	314 (3)	133 (2)
FUNC	=00000010	174 (3)	#-259 (3)
IO$_WRITELBLK	=00000020		#-259 (3)
READY	0000010B-R	307 (3)	#-188 (3) #-201 (3) #-237 (3) #-263 (3)
			#-309 (3)
RL_BA	00000002		#-248 (3) #-278 (3)
RL_CS	00000000		#-187 (3) #-192 (3) #-200 (3) #-236 (3)
			#-262 (3)
RL_CS_M_DRDY	=00000001		#-192 (3)
RL_DA	00000004		#-186 (3) #-235 (3) #-247 (3) #-273 (3)
			#-274 (3)
RL_DA_M_MRK	=00000001		#-186 (3) #-234 (3)
RL_DA_M_RST	=00000008		#-184 (3)
RL_DA_M_STS	=00000002		#-185 (3)
RL_MP	00000006		#-190 (3) #-204 (3) #-257 (3)
RL_MP_M_BH	=00000008		#-190 (3)
RL_MP_M_HO	=00000010		#-189 (3)
RPB$W_UNIT	=00000064		#-183 (3)
SIZE	=00000008	173 (3)	#-285 (3) #-295 (3)
SS$_CTRLERR	=00000054		#-299 (3)
SS$_NORMAL	=00000001		#-288 (3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	130 (2)	130 (2)
$BTDDEF	2	54 (2)	54 (2)
$DEFINI	1	54 (2)	54 (2) 55 (2) 56 (2) 57 (2) 58 (2) 59 (2) 72 (2)
$IODEF	15	55 (2)	55 (2)
$RPBDEF	5	56 (2)	56 (2)
$SSDEF	21	57 (2)	57 (2)
$UBADEF	6	58 (2)	58 (2)
$UBIDEF	2	59 (2)	59 (2)

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ASHL            0078      217    (3)
BCC             001E      230    (3)
BEQL            0013      191    (3)    193    (3)    220    (3)    229    (3)    267    (3)    309    (3)
BGEQ            0018      202    (3)    238    (3)
BICW            00AA      226    (3)
BICW3           00AB      204    (3)    227    (3)    273    (3)
BISW            00A8      232    (3)
BISW3           00A9      187    (3)    200    (3)    234    (3)    236    (3)    262    (3)    274    (3)
BITW            00B3      189    (3)    192    (3)    308    (3)
BLEQU           001B      254    (3)
BLSS            0019      264    (3)
BNEQ            0012      260    (3)    286    (3)    296    (3)
BRW             0031      203    (3)    239    (3)    279    (3)
BSBW            0030      188    (3)    201    (3)    237    (3)    263    (3)
CLRL            00D4      182    (3)    211    (3)    214    (3)    277    (3)
CMPL            00D1      259    (3)
CMPW            00B1      219    (3)    253    (3)
DIVW            00A6      256    (3)
EDIV            007B      212    (3)    215    (3)
EXTZV           00EF      276    (3)    298    (3)
INCW            00B6      275    (3)
INSV            00F0      183    (3)    218    (3)    233    (3)    245    (3)
MNEGW           00AE      231    (3)    257    (3)
MOVL            00D0      287    (3)    297    (3)
MOVW            00B0      184    (3)    247    (3)    248    (3)    249    (3)    255    (3)    258    (3)    261    (3)
                          278    (3)
MOVZWL          003C      285    (3)    288    (3)    295    (3)    299    (3)
MULL            00C4      210    (3)
MULW            00A4      252    (3)    265    (3)
RSB             0005      289    (3)    300    (3)    310    (3)
SUBW            00A2      228    (3)    266    (3)
SUBW3           00A3      251    (3)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...													
.ASCIC	312	(3)												
.END	316	(3)												
.ENDC	133	(2)												
.ENDM	130	(2)	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)
.IDENT	5	(1)												
.IF	133	(2)												
.IFF	133	(2)												
.LONG	133	(2)												
.NOCROSS	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	72	(2)
.PSECT	133	(2)												
.RESTORE	119	(2)	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)
.SAVE	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	72	(2)
.SBTTL	135	(3)	49	(2)										
.TITLE	4	(1)												
.WORD	133	(2)												

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	4	#-259 (3) #-285 (3) #-295 (3) 298 (3)
R0	8	#-182 (3) 183 (3) #-187 (3) #-200 (3) #-236 (3) #-262 (3) #-288 (3) #-299 (3)
R1	19	#-204 (3) #-219 (3) #-226 (3) #-228 (3) #-231 (3) #-232 (3) 233 (3) #-234 (3) #-251 (3) #-252 (3) #-253 (3) #-255 (3) #-258 (3) #-261 (3) #-262 (3) #-273 (3) #-285 (3) #-287 (3)
R10	3	#-248 (3) #-278 (3) #-298 (3)
R2	9	#-227 (3) #-228 (3) #-249 (3) #-253 (3) #-255 (3) #-256 (3) #-257 (3) #-265 (3) #-266 (3)
R4	6	#-212 (3) #-215 (3) #-245 (3) #-251 (3) #-277 (3)
R5	7	#-210 (3) #-212 (3) #-214 (3) #-215 (3) #-218 (3) #-233 (3) #-276 (3)
R6	12	#-211 (3) #-212 (3) #-217 (3) 218 (3) #-219 (3) #-227 (3) 245 (3) #-247 (3) #-274 (3) #-275 (3) 276 (3)
R7	16	#-186 (3) #-187 (3) #-190 (3) #-192 (3) #-200 (3) #-204 (3) #-235 (3) #-236 (3) #-247 (3) #-248 (3) #-257 (3) #-262 (3) #-273 (3) #-274 (3) #-278 (3) #-308 (3)
R8	4	#-249 (3) #-266 (3) #-295 (3) #-297 (3)
R9	1	#-183 (3)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.19
Command processing	889	00:00:00.27	00:00:00.74
Pass 1	644	00:00:02.05	00:00:03.07
Symbol table sort	0	00:00:00.33	00:00:00.32
Pass 2	62	00:00:00.59	00:00:01.08
Symbol table output	2	00:00:00.01	00:00:00.06
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	5	00:00:00.08	00:00:00.11
Assembler run totals	1628	00:00:03.37	00:00:05.58

The working set limit was 2400 pages.
118003 bytes (231 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1080 non-local and 13 local symbols.
316 source lines were read in Pass 1, producing 71 object records in Pass 2.
40 pages of virtual memory were used to define 15 macros.

```
                          +----------------------------+
                          ! Macro library statistics !
                          +----------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                1
SYS$COMMON:[SYSLIB]LIB.MLB;2                           4
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       6
TOTALS (all libraries)                                11

1117 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DLBTDRIVR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

Table of contents

(2) 49 DECLARATIONS
(3) 186 RK06/7 Bootstrap driver code
(4) 332 ECC - PERFORM ECC ERROR CORRECTION

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DMBTDRIVR.MAR
0000     2 ; *1     6-FEB-1986 15:17:32 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element DMBTDRIVR.MAR
0000     4         .TITLE  DMBTDRIVR - RK06/7 BOOT DRIVER
0000     5         .IDENT  'V02-002'
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1979, 1980                                                 *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    12 ;*                                                                           *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    18 ;*  TRANSFERRED.                                                             *
0000    19 ;*                                                                           *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    22 ;*  CORPORATION.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28 ;
0000    29
0000    30 ;++
0000    31 ; FACILITY:     BOOTS
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       This module contains the bootstrap device driver for the
0000    35 ;       RK06/7 disks.
0000    36 ;
0000    37 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    38 ;
0000    39 ; AUTHOR:  Steve Beckhardt,     CREATION DATE:  1-Nov-1979
0000    40 ;  (Original author:  Carol Peters)
0000    41 ;
0000    42 ; MODIFIED BY:
0000    43 ;
0000    44 ;       02-02   CAS0001         C.A. Samuelson          30-Apr-1980
0000    45 ;                               Change interface to BOOTDRIVR for purge of UBA datapath
0000    46 ;
0000    47 ;--
```

```
                        0000     49          .SBTTL  DECLARATIONS
                        0000     50 ;
                        0000     51 ; INCLUDE FILES:
                        0000     52 ;
                        0000     53
                        0000     54          $BTDDEF                         ; Boot device types
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     55          $IODEF                          ; I/O function codes
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     56          $PRDEF                          ; Processor registers
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     57          $RPBDEF                         ; RPB offsets
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     58          $SSDEF                          ; Status codes
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     59          $UBADEF                         ; UBA definitions
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     60          $UBIDEF                         ; 11/750 UBA definitions
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     61
                        0000     62 ;
                        0000     63 ; MACROS:
                        0000     64 ;
                        0000     65
                        0000     66 ;
                        0000     67 ; EQUATED SYMBOLS:
                        0000     68 ;
                        0000     69 ;
                        0000     70 ; RK611/RK06 CONTROLLER REGISTER OFFSETS
                        0000     71 ;
                        0000     72
                        0000     73          $DEFINI RK
                        0000                 .SAVE   LOCAL_BLOCK
                        0000     74
                        0000     75 $DEF     RK_CS1              .BLKW   1       ;CONTROL STATUS REGISTER 1
                        0000                 .IIF    NB,RK_CS1,      RK_CS1:
                        0000                 .IIF    NB,,BLKW,       .BLKW   1
                        0002     76          _VIELD  RK_CS1,0,<-                 ; CONTROL STATUS REGISTER 1 FIELD DEFINITION
                        0002     77                  <GO,,M>,-                   ; GO BIT
                        0002     78                  <FCODE,4>,-                 ; FUNCTION CODE
                        0002     79                  <DPPE,,M>,-                 ; DATA PATH PURGE ERROR
                        0002     80                  <IE,,M>,-                   ; INTERRUPT ENABLE
                        0002     81                  <RDY,,M>,-                  ; CONTROLLER READY
                        0002     82                  <MEX,2>,-                   ; MEMORY EXTENSION BITS
                        0002     83                  <CDT,,M>,-                  ; CONTROLLER DRIVE TYPE
                        0002     84                  <CTO,,M>,-                  ; CONTROLLER TIME OUT
                        0002     85                  <CFMT,,M>,-                 ; CONTROLLER FORMAT ERROR
                        0002     86                  <SPAR,,M>,-                 ; SERIAL BUS PARITY ERROR
                        0002     87                  <DI,,M>,-                   ; DRIVE INTERRUPT
                        0002     88                  <CERR,,M>-                  ; CONTROLLER ERROR
```

```
                 0002    89          >                                   ;
                 0002    90 $DEF     RK_WC            .BLKW   1         ;WORD COUNT REGISTER
                 0002                .IIF    NB,RK_WC,       RK_WC:
        00000004 0002                .IIF    NB,.BLKW,       .BLKW   1
                 0004    91 $DEF     RK_BA            .BLKW   1         ;BUFFER ADDRESS REGISTER
                 0004                .IIF    NB,RK_BA,       RK_BA:
        00000006 0004                .IIF    NB,.BLKW,       .BLKW   1
                 0006    92 $DEF     RK_DA            .BLKW   1         ;DESIRED SECTOR/TRACK ADDRESS REGISTER
                 0006                .IIF    NB,RK_DA,       RK_DA:
        00000008 0006                .IIF    NB,.BLKW,       .BLKW   1
                 0008    93          _VIELD  RK_DA,0,<-                ; DESIRED ADDRESS FIELD DEFINITIONS
                 0008    94                  <SA,5>,-                  ; DESIRED SECTOR ADDRESS
                 0008    95                  <,3>,-                    ; RESERVED BITS
                 0008    96                  <TA,3>-                   ; DESIRED TRACK ADDRESS
                 0008    97          >                                   ;
                 0008    98 $DEF     RK_CS2           .BLKW   1         ;CONTROL STATUS REGISTER 2
                 0008                .IIF    NB,RK_CS2,      RK_CS2:
        0000000A 0008                .IIF    NB,.BLKW,       .BLKW   1
                 000A    99          _VIELD  RK_CS2,0,<-               ; CONTROL STATUS REGISTER 2 FIELD DEFINITION
                 000A   100                  <DS,3>,-                  ; DRIVE SELECT
                 000A   101                  <RLS,,M>,-                ; RELEASE DRIVE
                 000A   102                  <BAI,,M>,-                ; BUFFER ADDRESS INCREMENT INHIBIT
                 000A   103                  <SCLR,,M>,-               ; SUBSYSTEM CLEAR
                 000A   104                  <IR,,M>,-                 ; INPUT READY
                 000A   105                  <OR,,M>,-                 ; OUTPUT READY
                 000A   106                  <UFE,,M>,-                ; UNIT FIELD ERROR
                 000A   107                  <MDS,,M>,-                ; MULTIPLE DRIVE SELECT
                 000A   108                  <PGE,,M>,-                ; PROGRAMMING ERROR
                 000A   109                  <NEM,,M>,-                ; NONEXISTENT MEMORY
                 000A   110                  <NED,,M>,-                ; NONEXISTENT DRIVE
                 000A   111                  <UPE,,M>,-                ; UNIBUS PARITY ERROR
                 000A   112                  <WCE,,M>,-                ; WRITE CHECK ERROR
                 000A   113                  <DLT,,M>-                 ; DATA LATE ERROR
                 000A   114          >                                   ;
                 000A   115 $DEF     RK_DS            .BLKW   1         ;DRIVE STATUS REGISTER
                 000A                .IIF    NB,RK_DS,       RK_DS:
        0000000C 000A                .IIF    NB,.BLKW,       .BLKW   1
                 000C   116          _VIELD  RK_DS,0,<-                ; DRIVE STATUS REGISTER BIT DEFINTIONS
                 000C   117                  <DRA,,M>,-                ; DRIVE AVAILABLE
                 000C   118                  <,1>,-                    ; RESERVED BIT
                 000C   119                  <OFST,,M>,-               ; DRIVE OFFSET
                 000C   120                  <ACLO,,M>,-               ; DRIVE AC LOW
                 000C   121                  <DCLO,,M>,-               ; DRIVE DC LOW
                 000C   122                  <DROT,,M>,-               ; DRIVE OFF TRACK
                 000C   123                  <VV,,M>,-                 ; VOLUME VALID
                 000C   124                  <DRDY,,M>,-               ; DRIVE READY
                 000C   125                  <DDT,,M>,-                ; DRIVE DRIVE TYPE
                 000C   126                  <,2>,-                    ; RESERVED BITS
                 000C   127                  <WRL,,M>,-                ; DRIVE WRITE LOCKED
                 000C   128                  <,1>,-                    ; RESERVED BIT
                 000C   129                  <PIP,,M>,-                ; POSITIONING IN PROGRESS
                 000C   130                  <DSC,,M>,-                ; DRIVE STATUS CHANGE
                 000C   131                  <SVAL,,M>-                ; DRIVE STATUS VALID
                 000C   132          >                                   ;
                 000C   133 $DEF     RK_ER            .BLKW   1         ;ERROR REGISTER
                 000C                .IIF    NB,RK_ER,       RK_ER:
        0000000E 000C                .IIF    NB,.BLKW,       .BLKW   1
```

```
                     000E   134          _VIELD  RK_ER,0,<-                ; ERROR REGISTER BIT DEFINITIONS
                     000E   135                  <ILF,,M>,-                ; ILLEGAL FUNCTION
                     000E   136                  <SKI,,M>,-                ; SEEK INCOMPLETE
                     000E   137                  <NXF,,M>,-                ; NONEXECUTABLE FUNCTION
                     000E   138                  <DRPAR,,M>,-              ; DRIVE PARITY ERROR
                     000E   139                  <FMTE,,M>,-               ; FORMAT ERROR
                     000E   140                  <DTYE,,M>,-               ; DRIVE TYPE ERROR
                     000E   141                  <ECH,,M>,-                ; ECC HARD ERROR
                     000E   142                  <BSE,,M>,-                ; BAD SECTOR ERROR
                     000E   143                  <HVRC,,M>,-               ; HEADER VRC ERROR
                     000E   144                  <COE,,M>,-                ; CYLINDER OVERFLOW ERROR
                     000E   145                  <IDAE,,M>,-               ; INVALID DISK ADDRESS ERROR
                     000E   146                  <WLE,,M>,-                ; WRITE LOCK ERROR
                     000E   147                  <DTE,,M>,-                ; DRIVE TIMING ERROR
                     000E   148                  <OPI,,M>,-                ; OPERATION INCOMPLETE
                     000E   149                  <UNS,,M>,-                ; DRIVE UNSAFE
                     000E   150                  <DCK,,M>-                 ; DATA CHECK ERROR
                     000E   151          >                                 ;
                     000E   152 $DEF     RK_AS               .BLKW   1     ;ATTENTION SUMMARY/OFFSET REGISTER
                     000E               .IIF     NB,RK_AS,      RK_AS:
           00000010  000E               .IIF     NB,,BLKW,      .BLKW   1
                     0010   153          _VIELD  RK_AS,0,<-                ; ATTENTION SUMMARY/OFFSET REGISTER FIELDS
                     0010   154                  <OF,7>,-                  ; DRIVE OFFSET
                     0010   155                  <,1>,-                    ; RESERVED BIT
                     0010   156                  <ATTN,8,M>-               ; DRIVE ATTENTION SUMMARY
                     0010   157          >                                 ;
                     0010   158 $DEF     RK_DC               .BLKW   1     ;DESIRED CYLINDER ADDRESS
                     0010               .IIF     NB,RK_DC,      RK_DC:
           00000012  0010               .IIF     NB,,BLKW,      .BLKW   1
                     0012   159 $DEF     RK_SPR              .BLKW   1     ;UNUSED REGISTER
                     0012               .IIF     NB,RK_SPR,     RK_SPR:
           00000014  0012               .IIF     NB,,BLKW,      .BLKW   1
                     0014   160 $DEF     RK_DB               .BLKW   1     ;DATA BUFFER REGISTER
                     0014               .IIF     NB,RK_DB,      RK_DB:
           00000016  0014               .IIF     NB,,BLKW,      .BLKW   1
                     0016   161 $DEF     RK_MR1              .BLKW   1     ;MAINTENANCE REGISTER 1
                     0016               .IIF     NB,RK_MR1,     RK_MR1:
           00000018  0016               .IIF     NB,,BLKW,      .BLKW   1
                     0018   162          _VIELD  RK_MR1,0,<<MS,3>>         ; MAINTENANCE REGISTER 1 FIELD DEFINITION
                     0018   163 $DEF     RK_EC1              .BLKW   1     ;ECC POSITION REGISTER
                     0018               .IIF     NB,RK_EC1,     RK_EC1:
           0000001A  0018               .IIF     NB,,BLKW,      .BLKW   1
                     001A   164          _VIELD  RK_EC1,0,<<EPS,13>>       ; ECC POSITION FIELD
                     001A   165 $DEF     RK_EC2              .BLKW   1     ;ECC PATTERN REGISTER
                     001A               .IIF     NB,RK_EC2,     RK_EC2:
           0000001C  001A               .IIF     NB,,BLKW,      .BLKW   1
                     001C   166          _VIELD  RK_EC2,0,<<EPT,11>>       ; ECC PATTERN FIELD
                     001C   167 $DEF     RK_MR2              .BLKW   1     ;MAINTENANCE REGISTER 2
                     001C               .IIF     NB,RK_MR2,     RK_MR2:
           0000001E  001C               .IIF     NB,,BLKW,      .BLKW   1
                     001E   168 $DEF     RK_MR3              .BLKW   1     ;MAINTENANCE REGISTER 3
                     001E               .IIF     NB,RK_MR3,     RK_MR3:
           00000020  001E               .IIF     NB,,BLKW,      .BLKW   1
                     0020   169
                     0020   170          $DEFEND RK
                     0000               .RESTORE
                     0000   171
```

```
                        0000   172
                        0000   173 ;
                        0000   174 ; OWN STORAGE:
                        0000   175 ;
                        0000   176
                        0000   177 ;
                        0000   178 ; Boot driver table entry
                        0000   179 ;
                        0000   180
                        0000   181          $BOOT_DRIVER    DEVTYPE = BTD$K_DM,-     ; Device type (DM)
                        0000   182                          SIZE = DM_DRVSIZ,-       ; Driver size
                        0000   183                          ADDR = DM_DRIVER,-       ; Driver address
                        0000   184                          DRIVRNAME = DMNAME       ; Driver name
                  00000000                      .PSECT  BOOTDRIVR_4
                      FFFF  0000                .WORD   -1
                      0001  0002                .WORD   BTD$K_DM
                  00000000  0004                .LONG   0
                  00000123' 0008                .LONG   DM_DRVSIZ
                  00000000' 000C                .LONG   DM_DRIVER-$TABLE
                  00000000  0010                .LONG   0
                  00000116' 0014                .LONG   DMNAME-DM_DRIVER
                        00000000                .PSECT  BOOTDRIVR_2
```

```
                        0000   186          .SBTTL  RK06/7 Bootstrap driver code
                        0000   187
                        0000   188 ;++
                        0000   189 ;
                        0000   190 ; Inputs:
                        0000   191 ;
                        0000   192 ;       R3      - base address of adapter's register space
                        0000   193 ;       R5      - LBN FOR CURRENT PIECE OF TRANSFER
                        0000   194 ;       R6      - contains 0
                        0000   195 ;       R7      - address of the device's CSR
                        0000   196 ;       R8      - SIZE OF TRANSFER IN BYTES
                        0000   197 ;       R9      - address of the RPB
                        0000   198 ;       R10     - starting address of transfer (byte offset in first
                        0000   199 ;                 page ORed with starting map register number)
                        0000   200 ;       R11     - LBN at start of transfer
                        0000   201 ;
                        0000   202 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                        0000   203 ;       BUF(AP) - Buffer address
                        0000   204 ;       SIZE(AP)- Size of transfer in bytes
                        0000   205 ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
                        0000   206 ;
                        0000   207 ; Implicit inputs:
                        0000   208 ;
                        0000   209 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                        0000   210 ;
                        0000   211 ; Outputs:
                        0000   212 ;
                        0000   213 ;       R0 - status code
                        0000   214 ;
                        0000   215 ;               SS$_NORMAL      - successful transfer
                        0000   216 ;               SS$_NOSUCHDEV   - unsupported device
                        0000   217 ;               SS$_CTRLERR     - fatal controller error
                        0000   218 ;
                        0000   219 ;       R3 - must be preserved
                        0000   220 ;
                        0000   221 ;       This routine destroys R1, R2, R4, R5, R6. Within the
                        0000   222 ;       routine, register usage is as follows:
                        0000   223 ;
                        0000   224 ;               R0      - mapping enabled flag
                        0000   225 ;                         device unit number
                        0000   226 ;                         status code
                        0000   227 ;               R1      - device function code
                        0000   228 ;               R2      - drive type according to device register
                        0000   229 ;               R4      - used in logical to physical calculation
                        0000   230 ;               R5      - used in logical to physical calculation
                        0000   231 ;               R6      - used in logical to physical calculation
                        0000   232 ;                         transfer word count
                        0000   233 ;
                        0000   234 ;--
00000004                0000   235 BUF = 4
00000008                0000   236 SIZE = 8
0000000C                0000   237 LBN = 12
00000010                0000   238 FUNC = 16
00000014                0000   239 MODE = 20
                        0000   240
                        0000   241 DM_DRIVER:                           ; RK06/7 device driver.
                        0000   242
```

```
                                  0000  ;
                                  0000  ; Translate the I/O function code into a device-dependent function
                                  0000  ; code for this disk.
                                  0000  ;
                                  0000
                 51    11    D0   0000  10$:    MOVL    #17,R1                    ; ASSUME READ
              20    10 AC    D1   0003          CMPL    FUNC(AP),#IO$_WRITELBLK   ; CHECK FOR WRITE FUNCTION
                       03    12   0007          BNEQ    20$                       ; NO, DO READ
                 51    13    D0   0009          MOVL    #19,R1                    ; SET WRITE FUNCTION CODE
                                  000C
                                  000C  ;
                                  000C  ; Clear controller and drive status. Confirm that the drive exists and
                                  000C  ; is ready for a transfer.
                                  000C  ;
                                  000C
              50    64 A9    3C   000C  20$:    MOVZWL  RPB$W_UNIT(R9),R0         ; GET UNIT NUMBER
              08 A7    20    B0   0010          MOVW    #RK_CS2_M_SCLR,RK_CS2(R7) ; CLEAR CONTROLLER AND ALL DRIVES
                     0092    30   0014          BSBW    READY                     ; WAIT FOR CONTROLLER READY
              08 A7    50    B0   0017          MOVW    R0,RK_CS2(R7)             ; SET DRIVE NUMBER
                                  001B
                                  001B  30$:                                      ; Clear drive and find type.
                 67    05    B0   001B          MOVW    #5,RK_CS1(R7)             ; Clear drive.
                     0088    30   001E          BSBW    READY                     ; Wait for controller ready.
                       52    D4   0021          CLRL    R2                        ; Assume RK06 drive.
        0A A7   0100 8F    B3   0023          BITW    #RK_DS_M_DDT,RK_DS(R7)    ; Is drive RK07?
                       05    13   0029          BEQL    33$                       ; No. Branch.
           52   0400 8F    3C   002B          MOVZWL  #RK_CS1_M_CDT,R2          ; Yes. Set drive code.
                                  0030
                                  0030  33$:                                      ; Check for existence of drive.
        08 A7   1000 8F    B3   0030          BITW    #RK_CS2_M_NED,RK_CS2(R7); Does drive exist?
                       06    13   0036          BEQL    35$                       ; Yes. Branch.
           50   0908 8F    3C   0038          MOVZWL  #SS$_NOSUCHDEV,R0         ; No.  Exit with error
                             05   003D          RSB
                                  003E
                                  003E  35$:                                      ; Clear drive and acknowledge.
                 8000 8F    B0   003E          MOVW    #RK_CS1_M_CERR,-          ; Clear controller error
                       67         0042                  RK_CS1(R7)                ; status.
              08 A7    50    B0   0043          MOVW    R0,RK_CS2(R7)             ; Set unit number code again.
           67    52    05    A9   0047          BISW3   #5,R2,RK_CS1(R7)          ; Clear drive.
                     005B    30   004B          BSBW    READY                     ; Wait for controller ready.
        0A A7   0080 8F    B3   004E          BITW    #RK_DS_M_DRDY,RK_DS(R7)   ; Is the drive ready?
                       E8    13   0054          BEQL    35$                       ; No. Clear drive again.
           67    52    03    A9   0056          BISW3   #3,R2,RK_CS1(R7)          ; Acknowledge pack and set
                                  005A                                            ; volume valid.
                       4D    10   005A          BSBB    READY                     ; Wait for controller ready.
                                  005C
                                  005C  ;
                                  005C  ; Compute the cylinder, track, and sector addresses. Load device
                                  005C  ; registers with transfer parameters and start transfer.
                                  005C  ;
                                  005C
                       56    D4   005C          CLRL    R6                        ; Clear register for EDIV.
 55    54    55   00000042 8F    7B   005E          EDIV    #22*3,R5,R4,R5            ; COMPUTE DESIRED CYLINDER
              10 A7    54    B0   0067          MOVW    R4,RK_DC(R7)              ; AND SET IN DEVICE
        56    55    55    16   7B   006B          EDIV    #22,R5,R5,R6              ; CALCULATE DESIRED TRACK/SECTOR
        56    08    08    55   F0   0070          INSV    R5,#8,#8,R6               ; MERGE TRACK AND SECTOR
              06 A7    56    B0   0075          MOVW    R6,RK_DA(R7)              ; SET DESIRED TRACK SECTOR
```

```
                04 A7   5A   B0  0079   300          MOVW    R10,RK_BA(R7)           ; SET STARTING BUFFER ADDRESS
              56   58   02   C7  007D   301          DIVL3   #2,R8,R6                ; COMPUTE WORD COUNT
                02 A7   56   AE  0081   302          MNEGW   R6,RK_WC(R7)            ; SET NUMBER OF WORDS TO TRANSFER
              67   52   51   A9  0085   303          BISW3   R1,R2,RK_CS1(R7)        ; Start disk function.
                        1E   10  0089   304          BSBB    READY                   ; WAIT FOR CONTROLLER READY
                             008B   305
                             008B   306 ;
                             008B   307 ; Transfer is complete. See whether the transfer completed without
                             008B   308 ; error. If not, prepare input registers for the ECC correction routine
                             008B   309 ; and branch to that routine.
                             008B   310 ;
                             008B   311
                   50   01   3C  008B   312          MOVZWL  #SS$_NORMAL,R0          ; ASSUME NORMAL COMPLETION
                        67   B5  008E   313          TSTW    RK_CS1(R7)              ; CHECK COMPOSITE ERROR
                        01   19  0090   314          BLSS    50$                     ; CONTINUE IF NO ERROR
                             05  0092   315          RSB                             ;
                50   0C A7   B0  0093   316 50$:     MOVW    RK_ER(R7),R0            ; GET ERROR STATUS
                51   02 A7   32  0097   317          CVTWL   RK_WC(R7),R1            ; GET NEGATED COUNT REMAINING
                   51   02   C4  009B   318          MULL    #2,R1                   ; CONVERT TO BYTES
                55   18 A7   3C  009E   319          MOVZWL  RK_EC1(R7),R5           ; GET POSITION OF ERROR
                56   1A A7   B0  00A2   320          MOVW    RK_EC2(R7),R6           ; GET PATTERN
                     0008    31  00A6   321          BRW     ECC                     ; AND ATTEMPT ECC CORRECTION
                             00A9   322
                             00A9   323 ;
                             00A9   324 ; SUBROUTINE TO WAIT FOR CONTROLLER READY OR ERROR
                             00A9   325 ;
                             00A9   326
                             00A9   327 READY:                                      ;
67            8080 8F   B3  00A9   328          BITW    #^X8080,(R7)            ;CONTROLLER READY OR ERROR?
                        F9   13  00AE   329          BEQL    READY                   ;IF EQL NO
                             05  00B0   330          RSB                             ;
```

```
                                00B1   332          .SBTTL  ECC - PERFORM ECC ERROR CORRECTION
                                00B1   333
                                00B1   334 ;++
                                00B1   335 ;
                                00B1   336 ; Functional description:
                                00B1   337 ;
                                00B1   338 ;       ATTEMPT ECC ERROR CORRECTION
                                00B1   339 ;
                                00B1   340 ; INPUTS:
                                00B1   341 ;
                                00B1   342 ;       R0      - RK_ER/RP_ER1  ERROR STATUS REGISTER
                                00B1   343 ;       R1      - NEGATIVE BYTE COUNT REMAINING
                                00B1   344 ;       R5      - ECC POSITION
                                00B1   345 ;       R6      - ECC PATTERN
                                00B1   346 ;       R8      - BYTE COUNT REMAINING AT START OF LAST TRANSFER
                                00B1   347 ;       R10     - starting address of transfer
                                00B1   348 ;       R11     - BLOCK NUMBER AT START OF TRANSFER
                                00B1   349 ;
                                00B1   350 ; Outputs:
                                00B1   351 ;
                                00B1   352 ;--
                                00B1   353
                                00B1   354 ECC:                                 ; ATTEMPT ECC CORRECTION
                                00B1   355
                                00B1   356 ;
                                00B1   357 ; Don't attempt an ECC correction if any of the following conditions apply:
                                00B1   358 ;
                                00B1   359 ;       the error was not a data check
                                00B1   360 ;       the error was a hard ECC error
                                00B1   361 ;       the transfer mode does not match the map-enabled position, i.e.,
                                00B1   362 ;               transfer is virtual, and mapping is not enabled, or v.v.
                                00B1   363 ;       no bytes have been transferred yet
                                00B1   364 ;
                                00B1   365
          5B 50    0F    E1   00B1   366          BBC     #RK_ER_V_DCK,R0,RETRY   ; NOT DATA CHECK, RETRY
          57 50    06    E0   00B5   367          BBS     #RK_ER_V_ECH,R0,RETRY   ; HARD ECC ERROR, RETRY
             50    38    DB   00B9   368          MFPR    #PR$_MAPEN,R0           ; GET MAP ENABLE STATE
          14 AC    50    D1   00BC   369          CMPL    R0,MODE(AP)             ; SAME AS I/O MODE
                   4E    12   00C0   370          BNEQ    RETRY                   ; NO, CANT DO SIMPLE ECC
             51    58    C0   00C2   371          ADDL    R8,R1                   ; COMPUTE BYTES TRANSFERRED
                   49    13   00C5   372          BEQL    RETRY                   ; NONE, RETRY
                               00C7   373
                               00C7   374 ;
                               00C7   375 ; Appears to be a correctable data check. Attempt the correction.
                               00C7   376 ;
                               00C7   377
       50    51    F7    8F   00C7   378          ASHL    #-9,R1,R0               ; CONVERT TO PAGE COUNT
             5B    50    C0   00CC   379          ADDL    R0,R11                  ; UPDATE BLOCK NUMBER
             5A    51    C0   00CF   380          ADDL    R1,R10                  ; UPDATE BYTE ADDRESS
             58    51    C2   00D2   381          SUBL    R1,R8                   ; DECREASE BYTES REMAINING
                   55    D7   00D5   382          DECL    R5                      ; MAKE POSITION 1 ORIGIN
    50    0200 C8    9E   00D7   383          MOVAB   512(R8),R0              ; REMAINING BYTES AT START OF BAD SECTOR
52    08 AC    50    C3   00DC   384          SUBL3   R0,SIZE(AP),R2          ; BYTES TRANSFERRED AT START OF ERROR SECTOR
             50    08    C4   00E1   385          MULL    #8,R0                   ; CONVERT BYTE COUNT TO BIT COUNT
             50    55    C2   00E4   386          SUBL    R5,R0                   ; COMPUTE CORRECTION FIELD WIDTH
                   19    15   00E7   387          BLEQ    20$                     ; BR IF NO CORRECTION NEEDED
             0D    50    D1   00E9   388          CMPL    R0,#RK_EC1_S_EPS        ; MINIMUM OF 13 AND BUFFER REMAINING
```

```
                        03   15  00EC          BLEQ    10$                     ; KEEP MINIMUM VALUE
                    50  0D   D0  00EE          MOVL    #RK_EC1_S_EPS,R0        ; LIMIT FIELD TO RK_EC1_S_EPS BITS
     51   04 BC42    50  55   EF  00F1  10$:    EXTZV   R5,R0,@BUF(AP)[R2],R1   ; GET FIELD TO BE CORRECTED
                    51  56   CC  00F8          XORL    R6,R1                   ; APPLY CORRECTION CODE
     04 BC42   50   55  51   F0  00FB          INSV    R1,R5,R0,@BUF(AP)[R2]   ; AND STORE IN BUFFER
                                0102
                                0102  ;
                                0102  ; If the transfer is not complete, branch back to retry it.
                                0102  ;
                                0102
                        58   D5  0102  20$:    TSTL    R8                      ; CHECK FOR COUNT REMAINING
                        06   15  0104          BLEQ    30$                     ; NONE, EXIT
                    55  5B   D0  0106          MOVL    R11,R5                  ; GET WORKING COPY OF LBN
                      FEF4   31  0109          BRW     DM_DRIVER               ; CONTINUE TRANSFER
                                010C
                                010C  ;
                                010C  ; Transfer is complete. Return with success status code.
                                010C  ;
                                010C
                    50  01   3C  010C  30$:    MOVZWL  #SS$_NORMAL,R0          ; SET COMPLETION CODE
                             05  010F          RSB                             ; AND RETURN
                                0110
                                0110  ;
                                0110  ; No ECC correction was possible.  Return and retry.
                                0110  ;
                                0110
                                0110  RETRY:
                50   0054 8F  3C  0110          MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                             05  0115          RSB                             ; Return to BOOTDRIVR
                                0116
58 45 2E 52 45 56 49 52 44 4D 44 00'  0116  DMNAME: .ASCIC  /DMDRIVER.EXE/          ; Driver filename
                             45  0122
                             0C  0116
                                0123
                       00000123  0123  DM_DRVSIZ=.-DM_DRIVER
                                0123
                                0123          .END
```

```
$TABLE                          = 00000000 R  D  02
BTD$K_DM                        = 00000001    D
BUF                             = 00000004    D
DMNAME                            00000116 R  D  03
DM_DRIVER                         00000000 R  D  03
DM_DRVSIZ                       = 00000123    D
ECC                               000000B1 R  D  03
FUNC                            = 00000010    D
IO$_WRITELBLK                   = 00000020    D
LBN                             = 0000000C    D
MODE                            = 00000014    D
PR$_MAPEN                       = 00000038    D
READY                             000000A9 R  D  03
RETRY                             00000110 R  D  03
RK_AS                             0000000E    D
RK_BA                             00000004    D
RK_CS1                            00000000    D
RK_CS1_M_CDT                    = 00000400    D
RK_CS1_M_CERR                   = 00008000    D
RK_CS2                            00000008    D
RK_CS2_M_NED                    = 00001000    D
RK_CS2_M_SCLR                   = 00000020    D
RK_DA                             00000006    D
RK_DB                             00000014    D
RK_DC                             00000010    D
RK_DS                             0000000A    D
RK_DS_M_DDT                     = 00000100    D
RK_DS_M_DRDY                    = 00000080    D
RK_EC1                            00000018    D
RK_EC1_S_EPS                    = 0000000D    D
RK_EC2                            0000001A    D
RK_ER                             0000000C    D
RK_ER_V_DCK                     = 0000000F    D
RK_ER_V_ECH                     = 00000006    D
RK_MR1                            00000016    D
RK_MR2                            0000001C    D
RK_MR3                            0000001E    D
RK_SPR                            00000012    D
RK_WC                             00000002    D
RPB$W_UNIT                      = 00000064    D
SIZ...                          = 0000000B    D
SIZE                            = 00000008    D
SS$_CTRLERR                     = 00000054    D
SS$_NORMAL                      = 00000001    D
SS$_NOSUCHDEV                   = 00000908    D
```

```
                                  +----------------+
                                  ! Psect synopsis !
                                  +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000020  (   32.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
BOOTDRIVR_4                     00000018  (   24.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
BOOTDRIVR_2                     00000123  (  291.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	184 (2)	184 (2)
BTD$K_DM	=00000001		184 (2)
BUF	=00000004	235 (3)	391 (4) 393 (4)
DMNAME	00000116-R	419 (4)	184 (2)
DM_DRIVER	00000000-R	241 (3)	184 (2) #-402 (4) 421 (4)
DM_DRVSIZ	=00000123	421 (4)	184 (2)
ECC	000000B1-R	354 (4)	#-321 (3)
FUNC	=00000010	238 (3)	#-249 (3)
IO$_WRITELBLK	=00000020		#-249 (3)
LBN	=0000000C	237 (3)	
MODE	=00000014	239 (3)	#-369 (4)
PR$_MAPEN	=00000038		#-368 (4)
READY	000000A9-R	327 (3)	#-260 (3) #-265 (3) #-282 (3) #-287 (3)
			#-304 (3) #-329 (3)
RETRY	00000110-R	415 (4)	#-366 (4) #-367 (4) #-370 (4) #-372 (4)
RK_BA	00000004		#-300 (3)
RK_CS1	00000000		#-264 (3) #-279 (3) #-281 (3) #-285 (3)
			#-303 (3) #-313 (3)
RK_CS1_M_CDT	=00000400		#-269 (3)
RK_CS1_M_CERR	=00008000		#-278 (3)
RK_CS2	00000008		#-259 (3) #-261 (3) #-272 (3) #-280 (3)
RK_CS2_M_NED	=00001000		#-272 (3)
RK_CS2_M_SCLR	=00000020		#-259 (3)
RK_DA	00000006		#-299 (3)
RK_DC	00000010		#-296 (3)
RK_DS	0000000A		#-267 (3) #-283 (3)
RK_DS_M_DDT	=00000100		#-267 (3)
RK_DS_M_DRDY	=00000080		#-283 (3)
RK_EC1	00000018		#-319 (3)
RK_EC1_S_EPS	=0000000D		#-388 (4) #-390 (4)
RK_EC2	0000001A		#-320 (3)
RK_ER	0000000C		#-316 (3)
RK_ER_V_DCK	=0000000F		#-366 (4)
RK_ER_V_ECH	=00000006		#-367 (4)
RK_WC	00000002		#-302 (3) #-317 (3)
RPB$W_UNIT	=00000064		#-258 (3)
SIZE	=00000008	236 (3)	#-384 (4)
SS$_CTRLERR	=00000054		#-416 (4)
SS$_NORMAL	=00000001		#-312 (3) #-408 (4)
SS$_NOSUCHDEV	=00000908		#-274 (3)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	181 (2)	181 (2)
$BTDDEF	2	54 (2)	54 (2)
$DEFINI	1	54 (2)	54 (2) 55 (2) 56 (2) 57 (2) 58 (2) 59 (2) 60 (2) 73 (2)
$IODEF	15	55 (2)	55 (2)
$PRDEF	5	56 (2)	56 (2)
$RPBDEF	5	57 (2)	57 (2)
$SSDEF	21	58 (2)	58 (2)
$UBADEF	6	59 (2)	59 (2)
$UBIDEF	2	60 (2)	60 (2)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	371 (4) 379 (4) 380 (4)
ASHL	0078	378 (4)
BBC	00E1	366 (4)
BBS	00E0	367 (4)
BEQL	0013	268 (3) 273 (3) 284 (3) 329 (3) 372 (4)
BISW3	00A9	281 (3) 285 (3) 303 (3)
BITW	00B3	267 (3) 272 (3) 283 (3) 328 (3)
BLEQ	0015	387 (4) 389 (4) 400 (4)
BLSS	0019	314 (3)
BNEQ	0012	250 (3) 370 (4)
BRW	0031	321 (3) 402 (4)
BSBB	0010	287 (3) 304 (3)
BSBW	0030	260 (3) 265 (3) 282 (3)
CLRL	00D4	266 (3) 294 (3)
CMPL	00D1	249 (3) 369 (4) 388 (4)
CVTWL	0032	317 (3)
DECL	00D7	382 (4)
DIVL3	00C7	301 (3)
EDIV	007B	295 (3) 297 (3)
EXTZV	00EF	391 (4)
INSV	00F0	298 (3) 393 (4)
MFPR	00DB	368 (4)
MNEGW	00AE	302 (3)
MOVAB	009E	383 (4)
MOVL	00D0	248 (3) 251 (3) 390 (4) 401 (4)
MOVW	00B0	259 (3) 261 (3) 264 (3) 278 (3) 280 (3) 296 (3) 299 (3) 300 (3) 316 (3) 320 (3)
MOVZWL	003C	258 (3) 269 (3) 274 (3) 312 (3) 319 (3) 408 (4) 416 (4)
MULL	00C4	318 (3) 385 (4)
RSB	0005	275 (3) 315 (3) 330 (3) 409 (4) 417 (4)
SUBL	00C2	381 (4) 386 (4)
SUBL3	00C3	384 (4)
TSTL	00D5	399 (4)
TSTW	00B5	313 (3)
XORL	00CC	392 (4)

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	419	(4)														
.END	423	(4)														
.ENDC	184	(2)														
.ENDM	181	(2)	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	60	(2)
.IDENT	5	(1)														
.IF	184	(2)														
.IFF	184	(2)														
.LONG	184	(2)														
.NOCROSS	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	60	(2)	73	(2)
.PSECT	184	(2)														
.RESTORE	170	(2)	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	60	(2)
.SAVE	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	60	(2)	73	(2)
.SBTTL	186	(3)	332	(4)	49	(2)										
.TITLE	4	(1)														
.WORD	184	(2)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	5	#-249	(3)	#-369	(4)	#-384	(4)	391	(4)	393	(4)		
R0	22	#-258	(3)	#-261	(3)	#-274	(3)	#-280	(3)	#-312	(3)	#-316	(3)
		366	(4)	367	(4)	#-368	(4)	#-369	(4)	#-378	(4)	#-379	(4)
		#-383	(4)	#-384	(4)	#-385	(4)	#-386	(4)	#-388	(4)	#-390	(4)
		#-391	(4)	#-393	(4)	#-408	(4)	#-416	(4)				
R1	12	#-248	(3)	#-251	(3)	#-303	(3)	#-317	(3)	#-318	(3)	#-371	(4)
		#-378	(4)	#-380	(4)	#-381	(4)	#-391	(4)	#-392	(4)	#-393	(4)
R10	2	#-300	(3)	#-380	(4)								
R11	2	#-379	(4)	#-401	(4)								
R2	8	#-266	(3)	#-269	(3)	#-281	(3)	#-285	(3)	#-303	(3)	#-384	(4)
		391	(4)	393	(4)								
R4	2	#-295	(3)	#-296	(3)								
R5	11	#-295	(3)	#-297	(3)	#-298	(3)	#-319	(3)	#-382	(4)	#-386	(4)
		#-391	(4)	#-393	(4)	#-401	(4)						
R6	8	#-294	(3)	#-297	(3)	298	(3)	#-299	(3)	#-301	(3)	#-302	(3)
		#-320	(3)	#-392	(4)								
R7	21	#-259	(3)	#-261	(3)	#-264	(3)	#-267	(3)	#-272	(3)	#-279	(3)
		#-280	(3)	#-281	(3)	#-283	(3)	#-285	(3)	#-296	(3)	#-299	(3)
		#-300	(3)	#-302	(3)	#-303	(3)	#-313	(3)	#-316	(3)	#-317	(3)
		#-319	(3)	#-320	(3)	#-328	(3)						
R8	5	#-301	(3)	#-371	(4)	#-381	(4)	383	(4)	#-399	(4)		
R9	1	#-258	(3)										

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.04	00:00:00.20
Command processing	877	00:00:00.27	00:00:00.75
Pass 1	724	00:00:02.47	00:00:03.48
Symbol table sort	0	00:00:00.35	00:00:00.36
Pass 2	93	00:00:00.70	00:00:01.13
Symbol table output	2	00:00:00.03	00:00:00.13
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	4	00:00:00.08	00:00:00.12
Assembler run totals	1726	00:00:03.95	00:00:06.18

The working set limit was 2400 pages.
143654 bytes (281 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1266 non-local and 9 local symbols.
423 source lines were read in Pass 1, producing 80 object records in Pass 2.
45 pages of virtual memory were used to define 16 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                1
SYS$COMMON:[SYSLIB]LIB.MLB;2                           4
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       7
TOTALS (all libraries)                                12

1226 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DMBTDRIVR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

Table of contents

(2) 66 DECLARATIONS
(3) 216 RB730:RB02/RB80 Bootstrap driver code

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DQBTDRIVR.MAR
0000     2 ; *1     6-FEB-1986 15:17:35 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element DQBTDRIVR.MAR
0000     4         .TITLE  DQBTDRIVR - RB730:RB02/RB80 BOOT DRIVER
0000     5         .IDENT  'V02-001'
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1981                                                       *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    12 ;*                                                                           *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    18 ;*  TRANSFERRED.                                                             *
0000    19 ;*                                                                           *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    22 ;*  CORPORATION.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28 ;
0000    29
0000    30 ;++
0000    31 ; FACILITY:     BOOTS
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       This module contains the bootstrap device driver for
0000    35 ;       the RB02 and RB80 disks on the RB730 controller.
0000    36 ;
0000    37 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    38 ;
0000    39 ; AUTHOR:  Greg Robert, CREATION DATE:  09-Jun-1981
0000    40 ;
0000    41 ; NOTE: The RB730 controller is supported by host microcode which is
0000    42 ;       activated each time a device register is accessed, and for
0000    43 ;       each block transferred.  Because of this, registers must be
0000    44 ;       handled in strict accordance with the RB730 software specification.
0000    45 ;       The following special registers protocols are significant:
0000    46 ;
0000    47 ;               DAR --  When the DAR is loaded some drive functions
0000    48 ;                       are initiated.  Consequently the function code
0000    49 ;                       must be loaded into the CSR (with CRDY set)
0000    50 ;                       prior to loading the DAR for seeks and transfers.
0000    51 ;                       For data transfers, the BAR and BCR must also
0000    52 ;                       be loaded prior to loading the DAR.
0000    53 ;
0000    54 ;               MPR --  The microcode keeps an internal disk address
0000    55 ;                       register for computing cylinder difference words
0000    56 ;                       for RB02 seek commands.  If this internal register
0000    57 ;                       falls out of sync with the actual disk address
```

```
0000    58 ;                    it can be reset by executing a "read header" command.
0000    59 ;                    Note that the header must actually be read (thru
0000    60 ;                    the MPR) in order to effect the reset.
0000    61 ;
0000    62 ; MODIFIED BY:
0000    63 ;
0000    64 ;--
```

```
                     0000     66          .SBTTL  DECLARATIONS
                     0000     67 ;
                     0000     68 ; INCLUDE FILES:
                     0000     69 ;
                     0000     70          .library        /sys$library:lib/
                     0000     71          .library        /$ds/
                     0000     72          .library        /$diag/
                     0000     73
                     0000     74          $BTDDEF                         ; Boot device types
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     75          $IODEF                          ; I/O function codes
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     76          $PRDEF                          ; Processor Register definitions
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     77          $RPBDEF                         ; RPB offsets
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     78          $SSDEF                          ; Status codes
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     79          $UBADEF                         ; UBA definitions
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     80          $UBIDEF                         ; 11/750 UBA definitions
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     81
                     0000     82 ;
                     0000     83 ; MACROS:
                     0000     84 ;
                     0000     85
                     0000     86 ;
                     0000     87 ; EQUATED SYMBOLS:
                     0000     88 ;
                     0000     89
                     0000     90 ;
                     0000     91 ; RB730:/RB02/RB80 REGISTER OFFSETS FROM CSR ADDRESS
                     0000     92 ;
                     0000     93          $DEFINI RB                      ; START OF REGISTER DEFINITIONS
                     0000                 .SAVE   LOCAL_BLOCK
                     0000     94
                     0000     95 $DEF     RB_CS           .BLKL   1       ;CONTROL STATUS REGISTER (CSR)
                     0000                 .IIF    NB,RB_CS,       RB_CS:
                00000004     0000                 .IIF    NB,,BLKL,       .BLKL   1
                     0004     96          _VIELD  RB_CS,0,<-              ;START OF CSR BIT DEFINITIONS
                     0004     97                  <DRDY,,M>,-             ; DRIVE READY
                     0004     98                  <FCODE,3>,-             ; FUNCTION CODE
                     0004     99                  <,2>,-                  ; RESERVED BITS
                     0004    100                  <IE,,M>,-               ; INTERRUPT ENABLE
                     0004    101                  <CRDY,,M>,-             ; CONTROLLER READY
                     0004    102                  <DS,2>,-                ; DRIVE SELECT
                     0004    103                  <OPI,,M>,-              ; OPERATION INCOMPLETE
                     0004    104                  <CRC,,M>,-              ; DATA CRC OR HEADER CRC OR DATA ECC
                     0004    105                  <DLT,,M>,-              ; DATA LATE OR HEADER NOT FOUND
```

```
                0004                    <NXM,,M>,-                  ; NON-EXISTENT MEMORY
                0004                    <DE,,M>,-                   ; DRIVE ERROR
                0004                    <CE,,M>,-                   ; COMPOSITE ERROR
                0004                    <ATN,4>,-                   ; DRIVE ATTENTION BITS
                0004                    <ECC,2>,-                   ; ECC STATUS
                0004                    <SSEI,,M>,-                 ; SKIP SECTOR ERROR INHIBIT
                0004                    <SSE,,M>,-                  ; SKIP SECTOR ERROR
                0004                    <IR,,M>,-                   ; IDC INTERRUPT REQUEST
                0004                    <MTN,,M>,-                  ; MAINTENANCE MODE
                0004                    <TYP,,M>,-                  ; DRIVE TYPE 1=RB80, 0=RB02
                0004                    <ASSI,,M>,-                 ; AUTOMATIC SKIP SECTOR INHIBIT
                0004                    <TOI,,M>,-                  ; TIME OUT INHIBIT (U-DIAG'S)
                0004                    <FMT,,M>,-                  ; R80 FORMAT CONTROL
                0004                    <,2>-                       ; RESERVED BITS
                0004            >                                   ;END CSR BIT DEFINITIONS
                0004
                0004    $DEF    RB_BA           .BLKL   1           ;BUS ADDRESS REGISTER (BAR)
                0004            .IIF    NB,RB_BA,       RB_BA:
        00000008 0004           .IIF    NB,.BLKL,       .BLKL   1
                0008
                0008    $DEF    RB_BC           .BLKL   1           ;BYTE COUNT REGISTER (BCR)
                0008            .IIF    NB,RB_BC,       RB_BC:
        0000000C 0008           .IIF    NB,.BLKL,       .BLKL   1
                000C
                000C    $DEF    RB_DA           .BLKL   1           ;DISK ADDRESS REGISTER (DAR)
                000C            .IIF    NB,RB_DA,       RB_DA:
        00000010 000C           .IIF    NB,.BLKL,       .BLKL   1
                0010            _VIELD  RB_DA,0,<-                  ;START OF DAR BIT DEFINITIONS
                0010                    <SEC,8>,-                   ; SECTOR
                0010                    <TRK,8>,-                   ; TRACK
                0010                    <CYL,16>-                   ; CYLINDER
                0010            >                                   ;END OF DAR BIT DEFINITIONS
                0010
                0010    $DEF    RB_MP           .BLKL   1           ;MULTIPURPOSE REGISTER (MPR)
                0010            .IIF    NB,RB_MP,       RB_MP:
        00000014 0010           .IIF    NB,.BLKL,       .BLKL   1
                0014            _VIELD  RB_MP,0,<-                  ;RB02 STATUS WORD DEFINITIONS
                0014                    <STA,3>,-                   ; DRIVE STATE
                0014                    <BH,,M>,-                   ; BRUSH HOME
                0014                    <HO,,M>,-                   ; HEADS OUT
                0014                    <CO,,M>,-                   ; COVER OPEN
                0014                    <HS,,M>,-                   ; HEAD SELECT
                0014                    <,1>,-                      ; RESERVED
                0014                    <DSE,,M>,-                  ; DRIVE SELECT ERROR
                0014                    <VC,,M>,-                   ; VOLUME CHECK
                0014                    <WGE,,M>,-                  ; WRITE GATE ERROR
                0014                    <SPE,,M>,-                  ; SPIN ERROR
                0014                    <SKTO,,M>,-                 ; SEEK TIME OUT
                0014                    <WL,,M>,-                   ; WRITE LOCK
                0014                    <CHE,,M>,-                  ; CURRENT HEAD ERROR
                0014                    <WDE,,M>-                   ; WRITE DATA ERROR
                0014            >                                   ;
                0014            _VIELD  RB_MP,0,<-                  ;GET STATUS COMMAND DEFINITIONS
                0014                    <MRK,,M>,-                  ; MARK (ALWAYS 1)
                0014                    <STS,,M>,-                  ; GET STATUS
                0014                    <,1>,-                      ; RESERVED
                0014                    <RST,,M>,-                  ; RESET
```

```
                     0014   155          >
                     0014   156          _VIELD  RB_MP,0,<-                ;RB80 STATUS WORD DEFINITIONS
                     0014   157                  <SEC,5>,-                 ; CURRENT RB80 SECTOR
                     0014   158                  <,3>,-                    ; RESERVED
                     0014   159                  <FLT,,M>,-                ; DRIVE FAULT
                     0014   160                  <PLGV,,M>,-               ; PLUG VALID
                     0014   161                  <SKE,,M>,-                ; SEEK ERROR
                     0014   162                  <ONCY,,M>,-               ; ON CYLINDER
                     0014   163                  <DRDY,,M>,-               ; DRIVE READY
                     0014   164                  <WTP,,M>,-                ; WRITE PROTECT
                     0014   165                  <,2>,-                    ; RESERVED
                     0014   166          >                                 ;END MPR BIT DEFINITIONS
                     0014   167
                     0014   168 $DEF     RB_EC1          .BLKL   1         ;ECC PATTERN REGISTER (EPAR)
                     0014           .IIF    NB,RB_EC1,      RB_EC1:
                     0014           .IIF    NB,,BLKL,       .BLKL   1
00000018             0018   169          _VIELD  RB_EC1,0,<-               ;START OF EC1 BIT DEFINITIONS
                     0018   170                  <POS,11>,-                ; STARTING BIT POSITION OF ECC ERROR
                     0018   171                  <,21>-                    ; RESERVED
                     0018   172          >                                 ;END EC1 BIT DEFINITIONS
                     0018   173
                     0018   174 $DEF     RB_EC2          .BLKL   1         ;ECC POSITION REGISTER (EPOR)
                     0018           .IIF    NB,RB_EC2,      RB_EC2:
0000001C             0018           .IIF    NB,,BLKL,       .BLKL   1
                     001C   175          _VIELD  RB_EC2,0,<-               ;START OF EC2 BIT DEFINITIONS
                     001C   176                  <PAT,11>,-                ; PATTERN OF ECC ERROR BURST
                     001C   177                  <,21>-                    ; RESERVED
                     001C   178          >                                 ;END EC2 BIT DEFINITIONS
                     001C   179
                     001C   180 $DEF     RB_CMD          .BLKL   1         ;AUXILLARY COMMAND REGISTER
                     001C           .IIF    NB,RB_CMD,      RB_CMD:
00000020             001C           .IIF    NB,,BLKL,       .BLKL   1
                     0020   181          _VIELD  RB_CMD,0,<-               ;START OF CMD BIT DEFINITIONS
                     0020   182                  <INIT,32>,-               ; SUBSYSTEM CLEAR <-- -1
                     0020   183          >                                 ;END CMD BIT DEFINITIONS
                     0020   184
                     0020   185          $DEFEND RB                        ;END RB730:RB80/RB02 REGISTER DEFS
                     0000           .RESTORE
                     0000   186
                     0000   187 ;
                     0000   188 ; HARDWARE FUNCTION CODES
                     0000   189 ;
                     0000   190
00000000             0000   191 F_NOP=0*2                                  ;NO OPERATION
00000006             0000   192 F_SEEK=3*2                                 ;SEEK CYLINDER
00000008             0000   193 F_READHEAD=4*2                             ;READ HEADER
00000002             0000   194 F_WRITECHECK=1*2                           ;WRITE CHECK
0000000A             0000   195 F_WRITEDATA=5*2                           ;WRITE DATA
0000000C             0000   196 F_READDATA=6*2                            ;READ DATA
00000004             0000   197 F_GETSTATUS=2*2                           ;GET STATUS
00000003             0000   198 BTD$K_DQ = 3
00000003             0000   199 PR$_SID_TYP730 = 3
                     0000   200
                     0000   201 ;
                     0000   202 ; OWN STORAGE:
                     0000   203 ;
                     0000   204
```

```
                        0000   205
                        0000   206 ;
                        0000   207 ; Boot driver table entry
                        0000   208 ;
                        0000   209
                        0000   210          $BOOT_DRIVER    CPUTYPE = PR$_SID_TYP730,- ; Must be VAX 11/730
                        0000   211                          DEVTYPE = BTD$K_DQ,-     ; Device type (DQ)
                        0000   212                          SIZE = DQ_DRVSIZ,-       ; Driver size
                        0000   213                          ADDR = DQ_DRIVER,-       ; Driver address
                        0000   214                          DRIVRNAME = DQNAME       ; Driver file name
                      00000000                     .PSECT  BOOTDRIVR_4
                    0003  0000                     .WORD   PR$_SID_TYP730
                    0003  0002                     .WORD   BTD$K_DQ
                00000000  0004                     .LONG   0
                00000135' 0008                     .LONG   DQ_DRVSIZ
                00000000' 000C                     .LONG   DQ_DRIVER-$TABLE
                00000000  0010                     .LONG   0
                00000128' 0014                     .LONG   DQNAME-DQ_DRIVER
                      00000000                     .PSECT  BOOTDRIVR_2
```

```
                                .SBTTL  RB730:RB02/RB80 Bootstrap driver code

                        ;++
                        ;
                        ; Inputs:
                        ;
                        ;       R3      - base address of adapter's register space
                        ;       R5      - LBN FOR CURRENT PIECE OF TRANSFER
                        ;       R6      - contains 0
                        ;       R7      - address of device's CSR
                        ;       R8      - SIZE OF TRANSFER IN BYTES
                        ;       R9      - address of the RPB
                        ;       R10     - starting address of transfer (byte offset in first
                        ;                 page ORed with starting map register number)
                        ;
                        ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                        ;       BUF(AP) - Buffer address
                        ;       SIZE(AP)- Size of transfer
                        ;
                        ; Implicit inputs:
                        ;
                        ;       RPB$W_UNIT      - RPB field containing boot device unit number
                        ;
                        ; Outputs:
                        ;
                        ;       R0 - status code
                        ;
                        ;               SS$_NORMAL      - successful transfer
                        ;               SS$_CTRLERR     - fatal controller error
                        ;
                        ;       R3 - must be preserved
                        ;
                        ;       This routine destroys R1, R2, R4, R5, R6.
                        ;
                        ;
                        ; Register useage during this routine:
                        ;
                        ;       R0 - scratch
                        ;       R1 - byte count for current transfer / scratch
                        ;       R2 - scratch
                        ;       R3 - adaptor base address
                        ;       R4 - unit number
                        ;       R5 - logical block number for current transfer
                        ;       R6 - disk address for current transfer / scratch
                        ;       R7 - device CSR
                        ;       R8 - remaining byte count
                        ;--

00000004                BUF = 4
00000008                SIZE = 8
00000010                FUNC = 16

                        DQ_DRIVER:

                        ;
                        ; COMPUTE ADDRESS OF DEVICE CSR
                        ;
```

```
                57   0200 C3   DE  0000   273          MOVAL   ^X200(R3),R7            ; COMPUTE CORRECT DEVICE CSR
                                     0005   274
                                     0005   275 ;
                                     0005   276 ; POSITION AND STORE THE UNIT NUMBER IN R4 FOR GLOBAL USE
                                     0005   277 ;
                          54   D4   0005   278          CLRL    R4                      ; CLEAR R4
        54   02   08   64 A9   F0   0007   279          INSV    RPB$W_UNIT(R9),#8,#2,R4 ; GET UNIT NUMBER
                                     000D   280
                                     000D   281 ;
                                     000D   282 ; RESET DRIVE AND GET STATUS
                                     000D   283 ;
                1C A7    01   CE   000D   284 10$:    MNEGL   #1,RB_CMD(R7)           ; INITIALIZE SUBSYSTEM
                                  D0 0011   285          MOVL    #RB_MP_M_STS-           ; PUT GET STATUS
                                     0012   286                  !RB_MP_M_RST-           ; ... AND WITH RESET
                                     0012   287                  !RB_MP_M_MRK,-          ; ... AND MARK BIT
                10 A7    0B          0012   288                  RB_MP(R7)               ; ... INTO DAR
                            54   C9   0015   289          BISL3   R4,-                    ; MERGE UNIT NUMBER
                                     0017   290                  #F_GETSTATUS,-          ; ... AND FUNCTION
                   67    04          0017   291                  RB_CS(R7)               ; ... INTO CSR
                      00F9   30      0019   292          BSBW    READY                   ; WAIT FOR DRIVE AND CONTROLLER READY
                50    10 A7   D0     001C   293          MOVL    RB_MP(R7),R0            ; FETCH STATUS WORD
           67   04000000 8F   D3     0020   294          BITL    #RB_CS_M_TYP,RB_CS(R7)  ; IS THIS AN RB80?
                            15   12   0027   295          BNEQ    15$                     ; BRANCH IF SO
                                     0029   296
                                     0029   297 ;
                                     0029   298 ; CHECK RB02 STATUS
                                     0029   299 ;
     1D   50    05    00   ED   0029   300          CMPZV   #0,#5,R0,-              ; TEST BITS 04:00 OF STATUS FOR
                                     002E   301                  #RB_MP_M_HO-            ; ... HEADS OUT
                                     002E   302                  !RB_MP_M_BH-            ; ... BRUSHES HOME
                                     002E   303                  !5                      ; ... SEEK LINEAR MODE (READY TO GO)
                            DD   12  002E   304          BNEQ    10$                     ; LOOP IF NOT READY
                                     0030   305
                                     0030   306 ;
                                     0030   307 ; READ A HEADER TO MAKE SURE MICROCODE IS SYNCRONIZED WITH CURRENT
                                     0030   308 ; DISK POSITION
                                     0030   309 ;
                            54   C9   0030   310          BISL3   R4,-                    ; MERGE UNIT NUMBER
                                     0032   311                  #F_READHEAD,-           ; ... AND FUNCTION
                   67    08          0032   312                  RB_CS(R7)               ; ... INTO CSR
                      00DE   30      0034   313          BSBW    READY                   ; WAIT FOR DRIVE AND CONTROLLER READY
           10 A7    10 A7   D1     0037   314          CMPL    RB_MP(R7),RB_MP(R7)     ; READ THE HEADER (UCODE DOESN'T LOOK
                                     003C   315                                          ; ...AT IT UNLESS WE ACCESS IT)
                            07   11  003C   316          BRB     20$                     ; CONTINUE IN COMMON
                                     003E   317
                                     003E   318
                                     003E   319 ;
                                     003E   320 ; CHECK RB80 STATUS
                                     003E   321 ;
     1A   50    05    08   ED   003E   322 15$:    CMPZV   #8,#5,R0,-              ; TEST BITS 08:12 OF STATUS FOR
                                     0043   323                  #<RB_MP_M_DRDY-         ; ...DRIVE READY
                                     0043   324                  !RB_MP_M_ONCY-          ; ...ON CYLINDER
                                     0043   325                  !RB_MP_M_PLGV>-         ; ...PLUG VALID
                                     0043   326                  @-8                     ; ...SHIFT TO LOW BYTE
                            C8   12  0043   327 18$:    BNEQ    10$                     ; IF NOT, BRANCH TO WAIT FOR IT
                                     0045   328
                                     0045   329
```

```
                                0045   330 ;
                                0045   331 ; NOW CONVERT LOGICAL TO PHYSICAL -- THE COMPUTED DISK ADDRESS HAS THE
                                0045   332 ; FORM OF A LONG WORD WITH LOW BYTE=SECTOR, NEXT BYTE=TRACK, AND HIGH
                                0045   333 ; WORD = CYLINDER.
                                0045   334 ;
                                0045   335 ;                I)      CYLINDER = LBN / BLOCKS_PER_CYLINDER
                                0045   336 ;               II)      TRACK = REMAINDER(I) / BLOCKS_PER_TRACK
                                0045   337 ;              III)      RB02_SECTOR = REMAINDER(II) * 2
                                0045   338 ;               IV)      RB80_SECTOR = REMAINDER(II)
                                0045   339 ;
                                0045   340
                          51    D4    0045   341 20$:    CLRL    R1                          ; CLEAR HIGH PART OF DIVIDEND
                          56    D4    0047   342         CLRL    R6                          ; CLEAR HIGH PART OF DIVIDEND
          67    04000000    8F    D3    0049   343         BITL    #RB_CS_M_TYP,RB_CS(R7)      ; IS THIS AN RB80?
                          13    12    0050   344         BNEQ    25$                         ; BRANCH IF SO
                                0052   345
                                0052   346 ;
                                0052   347 ;COMPUTE RB02 DISK ADDRESS
                                0052   348 ;
     50   52   55    28    7B    0052   349         EDIV    #40/2*2,R5,R2,R0            ; R2 = DESIRED CYL, R0 = REMAINING BLKS
     56   50   50    14    7B    0057   350         EDIV    #40/2,R0,R0,R6              ; R0 = DESIRED TRK, R6 = REMAINING BLKS
               56    02    C4    005C   351         MULL    #2,R6                       ; 2 SECTORS PER BLOCK
          51   28    56    C3    005F   352         SUBL3   R6,#40,R1                   ; R1 = 256 BYTE SECTORS LEFT ON SURFACE
                     15    11    0063   353         BRB     27$                         ; CONTINUE IN COMMON
                                0065   354
                                0065   355 ;
                                0065   356 ;COMPUTE RB80 DISK ADDRESS INSTEAD
                                0065   357 ;
50   52   55    000001B2    8F    7B    0065   358 25$:    EDIV    #31*14,R5,R2,R0             ; R2 = DESIRED CYL, R0 = REMAINING BLKS
     56   50    50    1F    7B    006E   359         EDIV    #31,R0,R0,R6                ; R0 = DESIRED TRK, R6 = REMAINING BLKS
          51   1F    56    C3    0073   360         SUBL3   R6,#31,R1                   ; R1 = 512 BYTE SECTORS LEFT ON SURFACE
               51    02    C4    0077   361         MULL    #2,R1                       ; R1 = 256 BYTE SECTORS LEFT ON SURFACE
                                007A   362
                                007A   363 ;
                                007A   364 ;IF FINAL TRANSFER USE REMAINING BYTE COUNT, ELSE USE REMAINDER OF TRACK
                                007A   365 ;
     51    00000100    8F    C4    007A   366 27$:    MULL    #256,R1                     ; R1 = BYTES LEFT ON SURFACE
               51    58    D1    0081   367         CMPL    R8,R1                       ; ARE ADDITIONAL TRANSFERS REQUIRED?
                     03    1A    0084   368         BGTRU   28$                         ; BRANCH IF ANSWER YES
               51    58    D0    0086   369         MOVL    R8,R1                       ; SET BYTE COUNT FOR FINAL TRANSFER
                                0089   370
                                0089   371 ;
                                0089   372 ;FORM FULL DISK ADDRESS (CYL, TRK, SEC)
                                0089   373 ;
     56   10   10    52    F0    0089   374 28$:    INSV    R2,#16,#16,R6               ; MOVE CYLINDER INTO HIGH WORD
     56   08   08    50    F0    008E   375         INSV    R0,#8,#8,R6                 ; MOVE TRACK INTO SECOND BYTE
                                0093   376
                                0093   377 ;
                                0093   378 ; PERFORM SEEK -- NOTE: FOR RB730 SEEKS AND TRANSFERS, THE COMMAND
                                0093   379 ; MUST BE LOADED INTO THE CSR WITH CONTROLLER READY BIT SET, BEFORE
                                0093   380 ; WRITING TO THE DISK ADDRESS REGISTER !!!
                                0093   381 ;
                          54    C9    0093   382         BISL3   R4, -                       ; MERGE UNIT NUMBER
                                0095   383                 #F_SEEK-                    ; ... FUNCTION AND
                                0095   384                 !RB_CS_M_CRDY,-             ; ... SUPPRESS EXECUTION
          67    00000086    8F        0095   385                 RB_CS(R7)               ; ... INTO CSR
                 0C   A7    56    D0    009B   386         MOVL    R6,RB_DA(R7)                ; LOAD DISK ADDRESS IN DAR
```

```
                67   00000080 8F    CA  009F   387          BICL    #RB_CS_M_CRDY,RB_CS(R7) ; INITIATE THE FUNCTION
                           006C    30  00A6   388          BSBW    READY                   ; WAIT FOR CONTROLLER AND DRIVE READY
                             51    19  00A9   389          BLSS    100$                    ; BRANCH IF ERROR
                                       00AB   390
                                       00AB   391
                                       00AB   392 ;
                                       00AB   393 ; SEEK IS COMPLETE -- EXECUTE TRANSFER FUNCTION
                                       00AB   394 ;
                     52    8C 8F    9A  00AB   395 60$:     MOVZBL  #F_READDATA-            ; ASSUME READ FUNCTION
                                       00AF   396                  !RB_CS_M_CRDY,R2        ; ... WITH CONTROLLER READY
                     20    10 AC    D1  00AF   397          CMPL    FUNC(AP),#IO$_WRITELBLK ; IS IT A WRITE FUNCTION?
                                04    12  00B3   398          BNEQ    70$                     ; BRANCH IF NOT
                     52    8A 8F    9A  00B5   399          MOVZBL  #F_WRITEDATA-           ; SET WRITE FUNCTION CODE
                                       00B9   400                  !RB_CS_M_CRDY,R2        ; ... WITH CONTROLLER READY
                                       00B9   401
                                       00B9   402 ;
                                       00B9   403 ; NOTE: THE DEVICE REGISTERS MUST BE LOADED IN THE PRESCRIBED ORDER!
                                       00B9   404 ;       1) FUNCTION CODE LOADED INTO CSR WITH CRDY SET
                                       00B9   405 ;       2) BYTE COUNT AND MEMORY ADDRESS (THESE TWO ARE REVERSABLE)
                                       00B9   406 ;       3) DISK ADDRESS (FIRST HARDWARE SILO LOADED AT THIS TIME)
                                       00B9   407 ;       4) CRDY CLEARED (SECOND SILO LOADED, XFER BEGINS)
                                       00B9   408 ;
                   67   52    54    C9  00B9   409 70$:     BISL3   R4,R2,-                 ; MERGE UNIT NUMBER AND FUNCTION
                                       00BD   410                  RB_CS(R7)               ;...INTO CSR
                     08 A7    51    CE  00BD   411          MNEGL   R1,RB_BC(R7)            ; SET NEG TRANSFER BYTE COUNT
                     04 A7    5A    D0  00C1   412          MOVL    R10,RB_BA(R7)           ; SET BUFFER ADDRESS
                     0C A7    56    D0  00C5   413          MOVL    R6,RB_DA(R7)            ; SET DESIRED DISK ADDRESS
                67   00000080 8F    CA  00C9   414          BICL    #RB_CS_M_CRDY,RB_CS(R7) ; INITIATE THE FUNCTION
                           0042    30  00D0   415          BSBW    READY                   ; WAIT FOR CONTROLLER AND DRIVE READY
                             27    19  00D3   416          BLSS    100$                    ; BRANCH ON ERROR
                                       00D5   417
                                       00D5   418 ;
                                       00D5   419 ; UPDATE BUFFER ADDRESS, BLOCK NUMBER, AND BYTES REMAINING
                                       00D5   420 ; FOR NEXT TRANSFER
                                       00D5   421 ;
                   5A    04 A7    D0  00D5   422          MOVL    RB_BA(R7),R10           ; UPDATE BUFFER ADDRESS
                      58   51    C2  00D9   423          SUBL    R1,R8                   ; UPDATE BYTES LEFT TO TRANSFER
                             0D    1B  00DC   424          BLEQU   90$                     ; BRANCH IF TRANSFER COMPLETE
                51   00000200 8F    C6  00DE   425          DIVL    #512,R1                 ; COMPUTE BLOCKS TRANSFERRED
                      55   51    C0  00E5   426          ADDL    R1,R5                   ; UPDATE LOGICAL BLOCK NUMBER
                           FF5A    31  00E8   427          BRW     20$                     ; CONTINUE TRANSFER
                                       00EB   428
                                       00EB   429 ;
                                       00EB   430 ; TRANSFER COMPLETE - RETURN
                                       00EB   431 ;
                                       00EB   432
                   51    08 AC    3C  00EB   433 90$:     MOVZWL  SIZE(AP),R1             ; SET TOTAL BYTES TRANSFERRED
                             07    12  00EF   434          BNEQ    95$                     ; BRANCH IF ORIGINAL SIZE WAS TRANSFERRED
                51   00008000 8F    D0  00F1   435          MOVL    #^X8000,R1              ; ELSE SIZE WAS FORCED TO 64K
                      50   01    3C  00F8   436 95$:     MOVZWL  #SS$_NORMAL,R0          ; SET COMPLETION CODE
                                05    00FB   437          RSB                             ; AND RETURN
                                       00FC   438
                                       00FC   439 ;
                                       00FC   440 ; RETRY ERROR
                                       00FC   441 ;
                                       00FC   442
                   58    08 AC    3C  00FC   443 100$:    MOVZWL  SIZE(AP),R8             ; RESTORE BYTE SIZE OF TRANSFER IN R8
```

```
                        07   12  0100          BNEQ    110$                    ; BRANCH IF SIZE WAS LEGAL
          58   00001F40 8F   D0  0102          MOVL    #8000,R8                ; ELSE FORCE TO 64K SIZE
     5A   04 AC    09   00   EF  0109  110$:   EXTZV   #0,#9,BUF(AP),R10       ; RESTORE BYTE OFFSET IN R10
               50   0054 8F   3C  010F          MOVZWL  #SS$_CTRLERR,R0         ; SET FATAL CONTROLLER ERROR
                             05  0114          RSB                             ; AND ATTEMPT RETRY
                                 0115
                                 0115
                                 0115  ;
                                 0115  ; SUBROUTINE TO WAIT FOR CONTROLLER AND DRIVE READY OR ERROR
                                 0115  ;
                                 0115
                                 0115  READY:
                             D3  0115          BITL    #RB_CS_M_CRDY-          ; CONTROLLER READY
                                 0116                  !RB_CS_M_CE,-           ; ... OR ERROR?
          67   00008080 8F       0116                  RB_CS(R7)               ; ... DEVICE CSR
                        F7   13  011C          BEQL    READY                   ; LOOP UNTIL CRDY OR ERROR
                             D3  011E          BITL    #RB_CS_M_DRDY-          ; DRIVE READY
                                 011F                  !RB_CS_M_CE,-           ; ... OR ERROR?
          67   00008001 8F       011F                  RB_CS(R7)               ; ... DEVICE CSR
                        EE   13  0125          BEQL    READY                   ; LOOP UNTIL DRDY OR ERROR
                             05  0127          RSB                             ;
                                 0128
58 45 2E 52 45 56 49 52 44 51 44 00'  0128  DQNAME: .ASCIC  /DQDRIVER.EXE/     ; Driver file name
                             45  0134
                             0C  0128
                                 0135
                       00000135  0135  DQ_DRVSIZ=.-DQ_DRIVER
                                 0135
                                 0135          .END
```

```
$TABLE                          = 00000000 R   D  02
BTD$K_DQ                        = 00000003     D
BUF                             = 00000004     D
DQNAME                            00000128 R   D  03
DQ_DRIVER                         00000000 R   D  03
DQ_DRVSIZ                       = 00000135     D
FUNC                            = 00000010     D
F_GETSTATUS                     = 00000004     D
F_NOP                           = 00000000     D
F_READDATA                      = 0000000C     D
F_READHEAD                      = 00000008     D
F_SEEK                          = 00000006     D
F_WRITECHECK                    = 00000002     D
F_WRITEDATA                     = 0000000A     D
IO$_WRITELBLK                   = 00000020     D
PR$_SID_TYP730                  = 00000003     D
RB_BA                             00000004     D
RB_BC                             00000008     D
RB_CMD                            0000001C     D
RB_CS                             00000000     D
RB_CS_M_CE                      = 00008000     D
RB_CS_M_CRDY                    = 00000080     D
RB_CS_M_DRDY                    = 00000001     D
RB_CS_M_TYP                     = 04000000     D
RB_DA                             0000000C     D
RB_EC1                            00000014     D
RB_EC2                            00000018     D
RB_MP                             00000010     D
RB_MP_M_BH                      = 00000008     D
RB_MP_M_DRDY                    = 00001000     D
RB_MP_M_HO                      = 00000010     D
RB_MP_M_MRK                     = 00000001     D
RB_MP_M_ONCY                    = 00000800     D
RB_MP_M_PLGV                    = 00000200     D
RB_MP_M_RST                     = 00000008     D
RB_MP_M_STS                     = 00000002     D
READY                             00000115 R   D  03
RPB$W_UNIT                      = 00000064     D
SIZ...                          = 00000020     D
SIZE                            = 00000008     D
SS$_CTRLERR                     = 00000054     D
SS$_NORMAL                      = 00000001     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000020 (32.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000135 (309.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	214 (2)	214 (2)
BTD$K_DQ	=00000003	198 (2)	214 (2)
BUF	=00000004	264 (3)	446 (3)
DQNAME	00000128-R	466 (3)	214 (2)
DQ_DRIVER	00000000-R	268 (3)	214 (2) 468 (3)
DQ_DRVSIZ	=00000135	468 (3)	214 (2)
FUNC	=00000010	266 (3)	#-397 (3)
F_GETSTATUS	=00000004	197 (2)	#-290 (3)
F_NOP	=00000000	191 (2)	
F_READDATA	=0000000C	196 (2)	#-396 (3)
F_READHEAD	=00000008	193 (2)	#-311 (3)
F_SEEK	=00000006	192 (2)	#-384 (3)
F_WRITECHECK	=00000002	194 (2)	
F_WRITEDATA	=0000000A	195 (2)	#-400 (3)
IO$_WRITELBLK	=00000020		#-397 (3)
PR$_SID_TYP730	=00000003	199 (2)	214 (2)
RB_BA	00000004		#-412 (3) #-422 (3)
RB_BC	00000008		#-411 (3)
RB_CMD	0000001C		#-284 (3)
RB_CS	00000000		#-291 (3) #-294 (3) #-312 (3) #-343 (3)
			#-385 (3) #-387 (3) #-410 (3) #-414 (3)
			#-458 (3) #-462 (3)
RB_CS_M_CE	=00008000		#-457 (3) #-461 (3)
RB_CS_M_CRDY	=00000080		#-384 (3) #-387 (3) #-396 (3) #-400 (3)
			#-414 (3) #-457 (3)
RB_CS_M_DRDY	=00000001		#-461 (3)
RB_CS_M_TYP	=04000000		#-294 (3) #-343 (3)
RB_DA	0000000C		#-386 (3) #-413 (3)
RB_MP	00000010		#-288 (3) #-293 (3) #-314 (3)
RB_MP_M_BH	=00000008		#-303 (3)
RB_MP_M_DRDY	=00001000		#-324 (3)
RB_MP_M_HO	=00000010		#-302 (3)
RB_MP_M_MRK	=00000001		#-287 (3)
RB_MP_M_ONCY	=00000800		#-325 (3)
RB_MP_M_PLGV	=00000200		#-325 (3)
RB_MP_M_RST	=00000008		#-287 (3)
RB_MP_M_STS	=00000002		#-286 (3)
READY	00000115-R	455 (3)	#-292 (3) #-313 (3) #-388 (3) #-415 (3)
			#-459 (3) #-463 (3)
RPB$W_UNIT	=00000064		#-279 (3)
SIZE	=00000008	265 (3)	#-433 (3) #-443 (3)
SS$_CTRLERR	=00000054		#-447 (3)
SS$_NORMAL	=00000001		#-436 (3)

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO            SIZE          DEFINITION       REFERENCES...
-----            ----          ----------       -------------
$BOOT_DRIVER     1             210     (2)      210     (2)
$BTDDEF          2             74      (2)      74      (2)
$DEFINI          1             74      (2)      74      (2)      75      (2)      76      (2)      77      (2)      78      (2)
                                                79      (2)      80      (2)      93      (2)
$IODEF           15            75      (2)      75      (2)
$PRDEF           5             76      (2)      76      (2)
$RPBDEF          5             77      (2)      77      (2)
$SSDEF           21            78      (2)      78      (2)
$UBADEF          6             79      (2)      79      (2)
$UBIDEF          2             80      (2)      80      (2)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	426	(3)												
BEQL	0013	459	(3)	463	(3)										
BGTRU	001A	368	(3)												
BICL	00CA	387	(3)	414	(3)										
BISL3	00C9	289	(3)	310	(3)	382	(3)	409	(3)						
BITL	00D3	294	(3)	343	(3)	456	(3)	460	(3)						
BLEQU	001B	424	(3)												
BLSS	0019	389	(3)	416	(3)										
BNEQ	0012	295	(3)	304	(3)	327	(3)	344	(3)	398	(3)	434	(3)	444	(3)
BRB	0011	316	(3)	353	(3)										
BRW	0031	427	(3)												
BSBW	0030	292	(3)	313	(3)	388	(3)	415	(3)						
CLRL	00D4	278	(3)	341	(3)	342	(3)								
CMPL	00D1	314	(3)	367	(3)	397	(3)								
CMPZV	00ED	300	(3)	322	(3)										
DIVL	00C6	425	(3)												
EDIV	007B	349	(3)	350	(3)	358	(3)	359	(3)						
EXTZV	00EF	446	(3)												
INSV	00F0	279	(3)	374	(3)	375	(3)								
MNEGL	00CE	284	(3)	411	(3)										
MOVAL	00DE	273	(3)												
MOVL	00D0	285	(3)	293	(3)	369	(3)	386	(3)	412	(3)	413	(3)	422	(3)
		435	(3)	445	(3)										
MOVZBL	009A	395	(3)	399	(3)										
MOVZWL	003C	433	(3)	436	(3)	443	(3)	447	(3)						
MULL	00C4	351	(3)	361	(3)	366	(3)								
RSB	0005	437	(3)	448	(3)	464	(3)								
SUBL	00C2	423	(3)												
SUBL3	00C3	352	(3)	360	(3)										

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	466	(3)														
.END	470	(3)														
.ENDC	214	(2)														
.ENDM	210	(2)	74	(2)	75	(2)	76	(2)	77	(2)	78	(2)	79	(2)	80	(2)
.IDENT	5	(1)														
.IF	214	(2)														
.IFF	214	(2)														
.LIBRARY	70	(2)	71	(2)	72	(2)										
.LONG	214	(2)														
.NOCROSS	74	(2)	75	(2)	76	(2)	77	(2)	78	(2)	79	(2)	80	(2)	93	(2)
.PSECT	214	(2)														
.RESTORE	185	(2)	74	(2)	75	(2)	76	(2)	77	(2)	78	(2)	79	(2)	80	(2)
.SAVE	74	(2)	75	(2)	76	(2)	77	(2)	78	(2)	79	(2)	80	(2)	93	(2)
.SBTTL	216	(3)	66	(2)												
.TITLE	4	(1)														
.WORD	214	(2)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	4	#-397 (3) #-433 (3) #-443 (3) 446 (3)
R0	12	#-293 (3) 300 (3) 322 (3) #-349 (3) #-350 (3) #-358 (3)
		#-359 (3) #-375 (3) #-436 (3) #-447 (3)
R1	13	#-341 (3) #-352 (3) #-360 (3) #-361 (3) #-366 (3) #-367 (3)
		#-369 (3) #-411 (3) #-423 (3) #-425 (3) #-426 (3) #-433 (3)
		#-435 (3)
R10	3	#-412 (3) #-422 (3) #-446 (3)
R2	6	#-349 (3) #-358 (3) #-374 (3) #-396 (3) #-400 (3) #-409 (3)
R3	1	273 (3)
R4	6	#-278 (3) 279 (3) #-289 (3) #-310 (3) #-382 (3) #-409 (3)
R5	3	#-349 (3) #-358 (3) #-426 (3)
R6	10	#-342 (3) #-350 (3) #-351 (3) #-352 (3) #-359 (3) #-360 (3)
		374 (3) 375 (3) #-386 (3) #-413 (3)
R7	21	#-273 (3) #-284 (3) #-288 (3) #-291 (3) #-293 (3) #-294 (3)
		#-312 (3) #-314 (3) #-343 (3) #-385 (3) #-386 (3) #-387 (3)
		#-410 (3) #-411 (3) #-412 (3) #-413 (3) #-414 (3) #-422 (3)
		#-458 (3) #-462 (3)
R8	5	#-367 (3) #-369 (3) #-423 (3) #-443 (3) #-445 (3)
R9	1	#-279 (3)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.33
Command processing	888	00:00:00.20	00:00:00.89
Pass 1	781	00:00:02.56	00:00:04.13
Symbol table sort	0	00:00:00.36	00:00:00.36
Pass 2	118	00:00:00.73	00:00:01.18
Symbol table output	2	00:00:00.02	00:00:00.11
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	5	00:00:00.08	00:00:00.12
Assembler run totals	1820	00:00:03.99	00:00:07.13

The working set limit was 2400 pages.
140009 bytes (274 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1238 non-local and 13 local symbols.
470 source lines were read in Pass 1, producing 79 object records in Pass 2.
45 pages of virtual memory were used to define 16 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                1
SYS$COMMON:[SYSLIB]LIB.MLB;2                           4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       7
TOTALS (all libraries)                              12

1226 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DQBTDRIVR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

Table of contents

(2) 43 DECLARATIONS
(3) 110 DR$INT - DR32 INTERRUPT HANDLER

```
                        1 ; DEC/CMS REPLACEMENT HISTORY, Element DRINT.MAR
                        2 ; *1     6-FEB-1986 15:17:37 DS ""
                        3 ; DEC/CMS REPLACEMENT HISTORY, Element DRINT.MAR
                        4         .TITLE  DRINT - DR32 interrupt handler
                        5
                        6         .IDENT  /V01/
00000000                7         .PSECT  SEP, WRT, SHR, EXE, LONG
                        8
                        9 ;
                       10 ;                                    COPYRIGHT (c) 1978, 1980 BY
                       11 ;            DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                       12 ;
                       13 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
                       14 ; ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
                       15 ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
                       16 ; COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
                       17 ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
                       18 ; TRANSFERRED.
                       19 ;
                       20 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
                       21 ; AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
                       22 ; CORPORATION.
                       23 ;
                       24 ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
                       25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                       26 ;
                       27
                       28 ;++
                       29 ; FACILITY:     EXECUTIVE
                       30 ;
                       31 ; ABSTRACT:
                       32 ;       THIS IS A DUMMY DR32 INTERRUPT HANDLER WHICH IS USED UNTIL
                       33 ;       THE REAL DR32 DRIVER IS LOADED.
                       34 ;
                       35 ; ENVIRONMENT:  KERNEL MODE, NON-PAGED
                       36 ;
                       37 ; AUTHOR:  STEVE BECKHARDT,     CREATION DATE:  7-MAR-1979
                       38 ;
                       39 ; MODIFIED BY:
                       40 ;
                       41 ;--
```

```
                        0000     43          .SBTTL  DECLARATIONS
                        0000     44 ;
                        0000     45 ; INCLUDE FILES:
                        0000     46 ;
                        0000     47
                        0000     48 ;
                        0000     49 ; MACROS:
                        0000     50 ;
                        0000     51
                        0000     52          $IDBDEF                         ; IDB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,IDB$L_CSR,   IDB$L_CSR:
00000004                0000                         .BLKL   1
                        0004                 .IIF    NB,IDB$L_OWNER, IDB$L_OWNER:
00000008                0004                         .BLKL   1
                        0008                 .IIF    NB,IDB$W_SIZE,  IDB$W_SIZE:
0000000A                0008                         .BLKW   1
                        000A                 .IIF    NB,IDB$B_TYPE,  IDB$B_TYPE:
0000000B                000A                         .BLKB   1
0000000C                000B                         .BLKB   1
                        000C                 .IIF    NB,IDB$W_UNITS, IDB$W_UNITS:
0000000E                000C                         .BLKW   1
00000010                000E                         .BLKW   1
                        0010                 .IIF    NB,IDB$L_ADP,   IDB$L_ADP:
00000014                0010                         .BLKL   1
                        0014                 .IIF    NB,IDB$L_UCBLST,        IDB$L_UCBLST:
00000034                0014                         .BLKL   8
                        0034                 .IIF    NB,IDB$C_LENGTH,        IDB$C_LENGTH:
                        0034                 .IIF    NB,IDB$K_LENGTH,        IDB$K_LENGTH:
                00000000                     .RESTORE
                        0000     53
                        0000     54 ;
                        0000     55 ; EQUATED SYMBOLS:
                        0000     56 ;
                        0000     57
                        0000     58          ;
                        0000     59          ; DR32 DCR REGISTER DEFINITIONS
                        0000     60          ;
                        0000     61
                        0000     62          $DEFINI DR
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                0034                 .=0
                        0000     63 $DEF     DR_DCR          .BLKL   1       ; DR32 CONTROL REGISTER
                        0000                 .IIF    NB,DR_DCR,      DR_DCR:
00000004                0000                 .IIF    NB,,BLKL,       .BLKL   1
                        0004     64          _VIELD  DR_DCR,0,<-
                        0004     65                  <ADPTYP,8>,-            ; ADAPTER TYPE
                        0004     66                  <ID2ERR,,M>,-           ; ID2 ERROR
                        0004     67                  <ID2TOS,2>,-            ; ID2 TIME-OUT STATUS
                        0004     68                  <,1>,-                  ; RESERVED
                        0004     69                  <ID1ERR,,M>,-           ; ID1 ERROR
                        0004     70                  <ID1TOS,2>,-            ; ID1 TIME-OUT STATUS
                        0004     71                  <RDS,,M>,-              ; READ DATA SUBSTITUTE
                        0004     72                  <CRD,,M>,-              ; CORRECTED READ DATA
                        0004     73                  <DCRHLT,,M>,-           ; DCR HALT
                        0004     74                  <DCRABT,,M>,-           ; DCR ABORT INTERRUPT
```

```
                     0004    75                 <PKTINT,,M>,-           ; PACKET INTERRUPT
                     0004    76                 <INTENB,,M>,-           ; INTERRUPT ENABLE
                     0004    77                 <,1>,-                  ; RESERVED
                     0004    78                 <PWR_UP,,M>,-           ; ADAPTER POWER UP
                     0004    79                 <PWR_DN,,M>,-           ; ADAPTER POWER DOWN
                     0004    80                 <EXTABT,,M>,-           ; EXTERNAL ABORT
                     0004    81                 <,1>,-                  ; RESERVED
                     0004    82                 <IMPDEP,6>,-            ; IMPLEMENTATION DEPENDENT BITS
                     0004    83             >
                     0004    84
                     0004    85 ; DCR CONTROL FIELD A CODES (USED WHEN WRITING TO DCR)
                     0004    86
            00000100 0004    87 DCR_K_CLRPWRUP=^X100
            00000200 0004    88 DCR_K_CLRPWRDN=^X200                    ; CLEAR POWER DOWN
            00000300 0004    89 DCR_K_CLREXTABT=^X300                   ; CLEAR EXTERNAL ABORT
            00000400 0004    90 DCR_K_CLRABTINT=^X400                   ; CLEAR ABORT INTERRUPT
            00000500 0004    91 DCR_K_CLRINTENB=^X500                   ; CLEAR INTERRUPT ENABLE
            00000600 0004    92 DCR_K_SETINTENB=^X600                   ; SET INTERRUPT ENABLE
            00000700 0004    93 DCR_K_CLRHLT=^X700                      ; CLEAR HALT
                     0004    94
                     0004    95 ; DCR CONTROL FIELD B CODES (USED WHEN WRITING TO DCR)
                     0004    96
            00001000 0004    97 DCR_K_CLRCRD=^X1000                     ; CLEAR CRD
            00002000 0004    98 DCR_K_SETEXTABT=^X2000                  ; SET EXTERNAL ABORT
            00003000 0004    99 DCR_K_CLRPKTINT=^X3000                  ; CLEAR PACKET INTERRUPT
            00004000 0004   100 DCR_K_RESET=^X4000                      ; RESET
            00005000 0004   101 DCR_K_SETOSQTST=^X5000                  ; SET OSEQ TEST
            00006000 0004   102 DCR_K_CLROSQTST=^X6000                  ; CLEAR OSEQ TEST
                     0004   103         $DEFEND DR
                 00000000           .RESTORE
                     0000   104
                     0000   105 ;
                     0000   106 ; OWN STORAGE:
                     0000   107 ;
                     0000   108
```

```
                        0000    110          .SBTTL  DR$INT - DR32 INTERRUPT HANDLER
                        0000    111 ;++
                        0000    112 ; FUNCTIONAL DESCRIPTION:
                        0000    113 ;
                        0000    114 ;       THIS MODULE IS A DUMMY DR32 INTERRUPT HANDLER WHICH IS USED
                        0000    115 ;       UNTIL THE REAL DR32 DRIVER IS LOADED.
                        0000    116 ;       THIS ROUTINE ALSO CONTAINS A DUMMY DR32 CONTROLLER INITIALIZATION
                        0000    117 ;       ENTRY POINT.
                        0000    118 ;
                        0000    119 ; CALLING SEQUENCE:
                        0000    120 ;
                        0000    121 ;       JSB FROM INTERRUPT VECTOR IN CRB
                        0000    122 ;
                        0000    123 ; INPUT PARAMETERS:
                        0000    124 ;
                        0000    125 ;       NONE
                        0000    126 ;
                        0000    127 ; IMPLICIT INPUTS:
                        0000    128 ;
                        0000    129 ;       THE STACK ON ENTRY IS AS FOLLOWS:
                        0000    130 ;
                        0000    131 ;                 0(SP)           ADDRESS OF IDB ADDRESS
                        0000    132 ;      4(SP) - 16(SP)           SAVED R2 - R5
                        0000    133 ;                20(SP)           INTERRUPT PC
                        0000    134 ;                24(SP)           INTERRUPT PSL
                        0000    135 ;
                        0000    136 ; OUTPUT PARAMETERS:
                        0000    137 ;
                        0000    138 ;       NONE
                        0000    139 ;
                        0000    140 ; IMPLICIT OUTPUTS:
                        0000    141 ;
                        0000    142 ;       NONE
                        0000    143 ;
                        0000    144 ; COMPLETION CODES:
                        0000    145 ;
                        0000    146 ;       NONE
                        0000    147 ;
                        0000    148 ; SIDE EFFECTS:
                        0000    149 ;
                        0000    150 ;       INTERRUPTS ARE DISABLED ON THE DR32
                        0000    151 ;
                        0000    152 ;--
                        0000    153
                        0000    154 DR$INT::
     53    9E    D0     0000    155          MOVL    @(SP)+,R3                       ; GET ADDRESS OF IDB
     54    63    D0     0003    156          MOVL    IDB$L_CSR(R3),R4                ; GET ADDRESS OF FIRST CSR
64   0100  8F    3C     0006    157          MOVZWL  #DCR_K_CLRPWRUP,DR_DCR(R4)      ; CLEAR POWER UP
64   0200  8F    3C     000B    158          MOVZWL  #DCR_K_CLRPWRDN,DR_DCR(R4)      ; CLEAR POWER DOWN
     52    8E    7D     0010    159          MOVQ    (SP)+,R2                        ; RESTORE REGISTERS
     54    8E    7D     0013    160          MOVQ    (SP)+,R4
                 02     0016    161          REI
                        0017    162
                        0017    163
                        0017    164 DR$INITIAL::                                    ; CONTROLLER INITIALIZATION
                        0017    165
                        0017    166
```

```
                64   4000 8F    3C  0017   167          MOVZWL  #DCR_K_RESET,(R4)              ; RESET DR (R4 POINTS TO FIRST CSR)
                                 05  001C   168          RSB
                                     001D   169
                                     001D   170
                                     001D   171
                                     001D   172          .END
```

```
DCR_K_CLRPWRDN  =  00000200     D
DCR_K_CLRPWRUP  =  00000100     D
DCR_K_RESET     =  00004000     D
DR$INITIAL         00000017 RG  D  01
DR$INT             00000000 RG  D  01
DR_DCR             00000000     D
IDB$B_TYPE         0000000A     D
IDB$C_LENGTH       00000034     D
IDB$K_LENGTH       00000034     D
IDB$L_ADP          00000010     D
IDB$L_CSR          00000000     D
IDB$L_OWNER        00000004     D
IDB$L_UCBLST       00000014     D
IDB$W_SIZE         00000008     D
IDB$W_UNITS        0000000C     D
SIZ...          =  00000006     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
SEP	0000001D (29.)	01 (1.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
ABS	00000034 (52.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
DCR_K_CLRPWRDN	=00000200		#-158 (3)
DCR_K_CLRPWRUP	=00000100		#-157 (3)
DCR_K_RESET	=00004000		#-167 (3)
DR$INITIAL	00000017-R	164 (3)	
DR$INT	00000000-R	154 (3)	
DR_DCR	00000000		#-157 (3) #-158 (3)
IDB$L_CSR	00000000		#-156 (3)

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEFINI         1           52     (2)      52     (2)      62     (2)
$IDBDEF         1           52     (2)      52     (2)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...					
MOVL	00D0	155	(3)	156	(3)		
MOVQ	007D	159	(3)	160	(3)		
MOVZWL	003C	157	(3)	158	(3)	167	(3)
REI	0002	161	(3)				
RSB	0005	168	(3)				

Directives Cross Reference

DIRECTIVE	REFERENCES...			
.END	172	(3)		
.ENDM	52	(2)		
.IDENT	6	(1)		
.NOCROSS	52	(2)	62	(2)
.PSECT	7	(1)		
.RESTORE	103	(2)	52	(2)
.SAVE	52	(2)	62	(2)
.SBTTL	110	(3)	43	(2)
.TITLE	4	(1)		

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
R2	1	#-159 (3)
R3	2	#-155 (3) #-156 (3)
R4	5	#-156 (3) #-157 (3) #-158 (3) #-160 (3) #-167 (3)
SP	3	#-155 (3) #-159 (3) #-160 (3)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.04	00:00:00.27
Command processing	891	00:00:00.20	00:00:00.69
Pass 1	338	00:00:00.32	00:00:01.00
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	28	00:00:00.11	00:00:00.24
Symbol table output	2	00:00:00.00	00:00:00.00
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	4	00:00:00.02	00:00:00.06
Assembler run totals	1289	00:00:00.71	00:00:02.28

The working set limit was 1900 pages.
13087 bytes (26 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 57 non-local and 0 local symbols.
172 source lines were read in Pass 1, producing 17 object records in Pass 2.
14 pages of virtual memory were used to define 9 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	5

87 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DRINT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element DSLOAD.B32
;    0002  0      ! *2    27-MAR-1987 16:53:36 BARANSKI "remove zero address argument check"
;    0003  0      ! *1     6-FEB-1986 15:17:49 DS ""
;    0004  0      ! DEC/CMS REPLACEMENT HISTORY, Element DSLOAD.B32
;    0005  0      %title '*** DSLOAD DS$LOAD routine'
;    0006  0      MODULE DSLOAD (                                 !Routines to handle DS$LOAD
;    0007  0                     IDENT = '10.8-12',                                          !G:2
;    0008  0                     ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE)
;    0009  0                     ) =
;    0010  0      !
;    0011  0      ! COPYRIGHT (c) 1980, 1981,1984
;    0012  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0013  0      !
;    0014  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0015  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0016  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0017  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0018  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0019  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0020  0      ! REMAIN IN DEC.
;    0021  0      !
;    0022  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0023  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0024  0      ! CORPORATION.
;    0025  0      !
;    0026  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0027  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0028  0
;    0029  0      !++
;    0030  0      ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0031  0      !
;    0032  0      ! ABSTRACT:
;    0033  0      !
;    0034  0      !       This module contains the routines necessary to handle
;    0035  0      !       the DS$LOAD routines.
;    0036  0      !
;    0037  0      ! AUTHOR:       Roger Riggs, CREATION DATE: 10-November-1979
;    0038  0      !                                                                           !G:2
;    0039  0      ! MODIFIED BY:
;    0040  0      !
;    0041  0      !               Dave Butenhof, 8-apr-1981, version 6.3
;    0042  0      !       01      In user mode, magtapes will only load one magtape block,
;    0043  0      !               irregardless of USZ field size, on $READ.  Therefore, loop
;    0044  0      !               on $READ until RMS$_EOF found, to make sure entire file is
;    0045  0      !               read.  Also, use $SPACE if VBN arg is specified, so it'll
;    0046  0      !               work on magtapes.
;    0047  0      !                                                                           !G:2
;    0048  0      !               Dave Butenhof, 02-Jun-1981, version 6.3                     !G:2
;    0049  0      !       02      Change directory specs of libraries for flexibility
;    0050  0      !
;    0051  0      !       03      Jack Stansbury, 28-Oct-1981, Version 6.5                    !G:2
;    0052  0      !               Changed PSECTs from SEP to WORK, CODE, and DATA.
;    0053  0      !
;    0054  0      !       04      Jack Stansbury, 10-Apr-1982, Version 6.7
;    0055  0      !               Changed the location of the following assignment statement:
;    0056  0      !                       addr = .address;
;    0057  0      !               in the routine, since if the return length is requested
```

```
;    0058  0     !                (ACTUALCOUNT() GEQ 5), the retlen is calculated to be the
;    0059  0     !                maximum of the actual loaded size (addr - address) or the
;    0060  0     !                XAB size. However, if the Length parameter is 0, the assignment
;    0061  0     !                statement above was not getting done, and the aboce calculation
;    0062  0     !                would have been wrong!
;    0063  0     !
;    0064  0     !        05      Richard Brown, 15-June-82,  Version 6.8
;    0065  0     !                Addition of code to differentiate between those diagnostics
;    0066  0     !                which have an image header and those which do not.  If the
;    0067  0     !                header is present, it is stored in a separate buffer area.
;    0068  0     !
;    0069  0     !        06      Richard Brown, 4-August-82, Version 6.9
;    0070  0     !                Change to allow loading files bigger than 64k.
;    0071  0     !
;    0072  0     !        07      Marion Baggett, 24-Sept-82,  Version 6.9
;    0073  0     !                Set addressing mode to fix trucation errors.
;    0074  0     !
;    0075  0     !        08      R.E.Muse, 15-Feb_84, version 6.14
;    0076  0     !                Added check of image header size for those diagnostics
;    0077  0     !                with new images headers created by VMS version 4.
;    0078  0     !
;    0079  0     !        09      Richard Brown, 25-Jun-84, Version 7.0
;    0080  0     !                Changed value of image header size checked for, since
;    0081  0     !                it changed between field test 1 and field test 2 of VMS V4.
;    0082  0     !
;    0083  0     !        10      Ted Salem, 14-Feb-85, Version 8.1
;    0084  0     !                Made changes in order to down-line load over the NI.
;    0085  0     !
;    0086  0     !        11      Jim Baranski, 24-May-85, Version 8.2
;    0087  0     !                Added check for address parameters equal to zero.
;    0088  0     !                                                                          !G:2
;    0089  0     !        12      Jim Baranski, 24-Mar-87, Version 10.8                     !G:2
;    0090  0     !                Remove check for address parameters equal to zero.        !G:2
;    0091  0     !                Incompatible with many diagnostics.                       !G:2
;    0092  0     !--
;    0093  0
;    0094  1     BEGIN
;    0095  1     !
;    0096  1     ! INCLUDE FILES:
;    0097  1     !
;    0098  1
;    0099  1     LIBRARY '$DIAG';                !
;    0100  1
;    0101  1     LIBRARY '$DS';                  !
;    0102  1
;    0103  1     LIBRARY 'SYS$LIBRARY:LIB';
;    0104  1
;    0105  1     $DS_DSADEF
;    0106  1
;    0107  1     !
;    0108  1     ! TABLE OF CONTENTS:
;    0109  1     !
;    0110  1
;    0111  1     FORWARD ROUTINE
;    0112  1         DSX$LOAD;
;    0113  1
;    0114  1     !
```

```
;    0115  1      ! EXTERNAL REFERENCES:
;    0116  1      !
;    0117  1
;    0118  1      EXTERNAL
;    0119  1
;    0120  1              DS$GL_FLAGS ;
;    0121  1
;    0122  1      !
;    0123  1      ! PSECT DEFINITIONS:
;    0124  1      !
;    0125  1
;    0126  1      PSECT                                                          !
;    0127  1          Plit    = Data (NoExecute, Share, NoWrite,                 !
;    0128  1                           Addressing_Mode (Long_Relative));         !
;    0129  1
;    0130  1      PSECT                                                          !
;    0131  1          Code    = Code (Execute, Share, NoWrite);                  !
;    0132  1
;    0133  1      PSECT                                                          !
;    0134  1          Own     = Work (NoExecute, NoShare, Write,                 !
;    0135  1                          Addressing_Mode (Long_Relative));         !
;    0136  1
;    0137  1      PSECT                                                          !
;    0138  1          Global  = Work (NoExecute, NoShare, Write,                 !
;    0139  1                          Addressing_Mode (Long_Relative));         !
;    0140  1
;    0141  1      BIND                                                           ![05]
;    0142  1              FMT_BAD_HDR =                                          ![05]
;    0143  1       UPLIT (%ASCIC 'Routine DSX$LOAD has detected non-standard diag. or image header.!/Assuming no image header.!/');![0
;    0144  1
;    0145  1      EXTERNAL
;    0146  1              IMAGE_HEADER,                                          ![05]
;    0147  1              IMAGE_HEADER_END;                                      ![05]
;    0148  1
```

```
;    0150  1      %SBTTL 'DSX$LOAD routine'
;    0151  1
;    0152  1      global routine dsx$load (file, default, length, address, retlen, retrec, vbn) =
;    0153  1
;    0154  1      !++
;    0155  1      ! Functional descrition:
;    0156  1      !
;    0157  1      !       This routine opens and read portions of the specified file using RMS
;    0158  1      !       into the buffer specified.
;    0159  1      !
;    0160  1      ! Formal parameters:
;    0161  1      !
;    0162  1      !       file    Address of a quadword descriptor of the file name to be used
;    0163  1      !       default Address of a quadword descriptor of the file name defaults
;    0164  1      !       length  Length of the area to receive the file in bytes.
;    0165  1      !       address Address of the buffer to receive the file
;    0166  1      !       retlen  Address of a longword to receive the length of the file (bytes)
;    0167  1      !       retrec  Address of a longword to receive record attributes of the file
;    0168  1      !       vbn     Virtual block of the file to start reading.
;    0169  1      !
;    0170  1      ! Side effects:
;    0171  1      !
;    0172  1      !       The current load media is used to locate the file
;    0173  1      !
;    0174  1      ! Completion codes:
;    0175  1      !
;    0176  1      !       Any RMS or system service return
;    0177  1      !--
;    0178  1
;    0179  2          begin
;    0180  2
;    0181  2          local
;    0182  2              xab_size,                   ! size of the image loaded                  [10]
;    0183  2              addr,                       ! address at end of load                    [01]
;    0184  2              status,                     ! Status of RMS calls
;    0185  2              len ;
;    0186  2
;    0187  2          map
;    0188  2              file : ref vector [2, long];        ! Length and address of file name
;    0189  2
;    0190  2          builtin
;    0191  2              actualcount;
;    0192  2
;    0193  2          label
;    0194  2              read;
;    0195  2
;    0196  2
;    0197  2          addr = .address;                ! Initial address to load                   [04]
;    0198  2
;    0199  2          IF NOT .DS$V_NI THEN            ! If not over the NI                        [10]
;    0200  2
;    0201  3          BEGIN
;    0202  3
;    0203  3          local
;    0204  3              fab : $fab_decl,            ! FAB for file access
;    0205  3              rab : $rab_decl,            ! RAB for file access
;    0206  3              xab : $xabfhc_decl;         ! XAB for file access
```

```
;    0207  3
;    0208  3          map
;    0209  3              default : ref vector [2, long];          ! Length and address of default file name
;    0210  3
; P  0211  3          $fab_init (fab = fab, fna = .file [1], fns = .file [0], dna = .default [1], dns = .default [0],
;    0212  3              xab = xab, fac = (get, bio));
;    0213  3          $xabfhc_init (xab = xab);
;    0214  3          $rab_init (rab = rab, fab = fab, rac = seq);
;    0215  3
;    0216  3          if not (status = $open (fab = fab)) then return .status;
;    0217  3
;    0218  3          XAB_SIZE = .xab [xab$l_ebk]^9 + .xab [xab$w_ffb] - 512 ;                                    ![10]
;    0219  3
;    0220  3
;    0221  3          if actualcount () geq 6
;    0222  3           then
;    0223  4              begin
;    0224  4
;    0225  4              map
;    0226  4                  retrec : ref block [, byte];
;    0227  4
;    0228  4                  retrec [0, 0, 16, 0] = .fab [fab$w_mrs];
;    0229  4                  retrec [2, 0, 8, 0] = .fab [fab$b_rfm];
;    0230  4                  retrec [3, 0, 8, 0] = .fab [fab$b_fsz];
;    0231  3              end;
;    0232  3
;    0233  3
;    0234  3          if .length neq 0
;    0235  3          then
;    0236  3      read :
;    0237  4              begin
;    0238  4
;    0239  4              local                                   !                                              [01]
;    0240  4                  first;                              !                                              [01]
;    0241  4
;    0242  4              if not (status = $connect (rab = rab)) then leave read;
;    0243  4
;    0244  4              if actualcount () geq 7                 ! If VBN specified                             [01]
;    0245  4              then                                    !                                              [01]
;    0246  4                  if .dsa$v_user                      ! if in user mode                              [01]
;    0247  4                      and (.fab [fab$l_dev] and dev$m_sqd) neq 0      ! and magtape                  [01]
;    0248  4                  then                                !                                              [01]
;    0249  5                      begin                           ! space forward.                               [01]
;    0250  5
;    0251  5                      local                           !                                              [01]
;    0252  5                          blsize;                     ! size of magtape block                        [01]
;    0253  5
;    0254  5                      blsize = .fab [fab$w_bls]/512;  ! Size of block in pages                       [01]
;    0255  5                      if ((.vbn - 1)/.blsize)*.blsize neq .vbn - 1    ! If can't access              [01]
;    0256  5                      then                            !                                              [01]
;    0257  5                          return ds$_error;           ! return fatal error                           [01]
;    0258  5
;    0259  5                      rab [rab$l_bkt] = (.vbn - 1)/.blsize;   ! Number of tape blocks to skip       [01]
;    0260  5                      status = $space (rab = rab);    ! do it                                        [01]
;    0261  5
;    0262  5                      if not .status then leave read  !                                              [01]
;    0263  5                      end                             !                                              [01]
```

```
;  0264  4                 else                                  !                                                  [01]
;  0265  4                     rab [rab$l_bkt] = .vbn;           !                                                  [01]
;  0266  4
;  0267  4             len = .length;                            ! Initial (max) length to load                     [01]
;  0268  4             first = 1;                                ! First time                                       [01]
;  0269  4
;  0270  4             while 1 do                                ! Loop on $READ until done                         [01]
;  0271  5                 begin                                 !                                                  [01]
;  0272  5                 rab [rab$l_ubf] = .addr;
;  0273  5                 if .len gtr %X'FE00'                  !                                                  [06]
;  0274  5                 then                                  !                                                  [06]
;  0275  5                     rab [rab$w_usz] = %X'FE00'        !                                                  [06]
;  0276  5                 else                                  !                                                  [06]
;  0277  5                     rab [rab$w_usz] = .len;
;  0278  5                 status = $read (rab = rab);
;  0279  5                 rab [rab$l_bkt] = 0;                  ! Use NBP for future reads (if any)                [01]
;  0280  5                 addr = .addr + .rab [rab$w_rsz];      ! Set to read next section (or EOF)                [01]
;  0281  5
;  0282  5                 if not .status then exitloop;         ! Leave if no good                                 [01]
;  0283  5
;  0284  5                 len = .len - .rab [rab$w_rsz];        !                                                  [01]
;  0285  5
;  0286  5                 first = 0;                            ! not first loop anymore                           [01]
;  0287  5
;  0288  5                 if .len eql 0                         ! If no more room,                                 [01]
;  0289  5                 then                                  !                                                  [01]
;  0290  6                     begin                             !                                                  [01]
;  0291  6                     status = rms$_normal;             !  that's OK                                       [01]
;  0292  6                     exitloop                          !                                                  [01]
;  0293  6                     end                               !                                                  [01]
;  0294  4                 end;                                  !                                                  [01]
;  0295  4
;  0296  4
;  0297  4             if not .first and .status eql rms$_eof    ! If end of file, but on loop                      [01]
;  0298  4             then                                      !                                                  [01]
;  0299  4                 status = rms$_normal;                 !                                                  [01]
;  0300  4
;  0301  4             $disconnect (rab = rab);
;  0302  4
;  0303  4
;  0304  3             END ;                     ! end if .length neq 0
;  0305  3
;  0306  3           $close (fab = fab);
;  0307  3
;  0308  3           END                         ! end if not NI, but continue else section
;  0309  3
;  0310  2         ELSE                          ! Load across the NI                          ![10]
;  0311  2
;  0312  2     !       Load file across the network:                                           ![10]
;  0313  2     !
;  0314  2     !       a) First determine if the filename has an extension, if it does not     ![10]
;  0315  2     !          then add the default extension.                                      ![10]
;  0316  2     !
;  0317  2     !       b) Copy the filename to another buffer called FILENAME.                 ![10]
;  0318  2     !
;  0319  2     !       c) Call the Load_self routine to invoke the network facilities.         ![10]
;  0320  2     !
```

```
;    0321  2
;    0322  3                    BEGIN                                                          ![10]
;    0323  3
;    0324  3                    OWN  PTR_BEG, PTR_COM, PTR_DIF, VER_LENGTH,                    ![10]
;    0325  3
;    0326  3                         FILE2     : VECTOR[2],          ! QUAD. of new string  ![10]
;    0327  3
;    0328  3                         FILE_NAME   : VECTOR[32,BYTE],  ! string of new file name  ![10]
;    0329  3                         EXTENSION : VECTOR[4,BYTE]  ;                             ![10]
;    0330  3
;    0331  3                    BIND LENGTH2  = FILE2[0] ,          ! length to pass to LOAD.  ![10]
;    0332  3                         ADDRESS2 = FILE2[1],           ! address to pass to LOAD. ![10]
;    0333  3
;    0334  3                         LENGTH1   = FILE[0],           ! length of original length ![10]
;    0335  3                         ADDRESS1  = FILE[1]  ;         ! address of original file name ![10]
;    0336  3
;    0337  3
;    0338  3                    EXTERNAL ROUTINE    LOAD_SELF : WEAK ;                                  ![10]
;    0339  3
;    0340  3                    LENGTH2 = .LENGTH1;                                            ![10]
;    0341  3                    ADDRESS2 = FILE_NAME ;                                         ![10]
;    0342  3
;    0343  3                    EXTENSION = %ASCII'.EXE' ;                                     ![10]
;    0344  3
;    0345  3                    IF CH$FAIL( CH$FIND_CH( .LENGTH1, CH$PTR(.ADDRESS1), %C'.' ) )  ![10]
;    0346  3
;    0347  3                    THEN                                                           ![10]
;    0348  3
;    0349  3     !                An extension does not exist so we have to add the default    ![10]
;    0350  3
;    0351  4                    BEGIN                                                          ![10]
;    0352  4
;    0353  4     !              First, search for a semi-colon                                 ![10]
;    0354  4
;    0355  4                    PTR_COM = CH$FIND_CH( .LENGTH1, CH$PTR(.ADDRESS1),%C';' ) ;    ![10]
;    0356  4
;    0357  4                    IF CH$FAIL( .PTR_COM )                                         ![10]
;    0358  4
;    0359  4                    THEN                                                           ![10]
;    0360  4
;    0361  4     !                no semi-colon therefore just add the extention               ![10]
;    0362  4
;    0363  4                      CH$COPY(.LENGTH1,CH$PTR(.ADDRESS1),4,CH$PTR(EXTENSION),0,    ![10]
;    0364  4                              .LENGTH1+4,CH$PTR(FILE_NAME) )                       ![10]
;    0365  4
;    0366  4                    ELSE                                                           ![10]
;    0367  4
;    0368  5                      BEGIN                                                        ![10]
;    0369  5
;    0370  5     !                semi-colon was found so add extention and version          ![10]
;    0371  5
;    0372  5                      PTR_BEG = CH$PTR(.ADDRESS1) ;                                ![10]
;    0373  5
;    0374  5                      PTR_DIF = .PTR_COM - .PTR_BEG ;                              ![10]
;    0375  5
;    0376  5                      VER_LENGTH =  .LENGTH1 - .PTR_DIF  ;                         ![10]
;    0377  5
```

```
;    0378  5                        CH$COPY(.PTR_DIF ,.PTR_BEG , 4,CH$PTR(EXTENSION),          ![10]
;    0379  5                                .VER_LENGTH, .PTR_COM  ,0,                         ![10]
;    0380  5                                .LENGTH1+4,CH$PTR(FILE_NAME) ) ;                   ![10]
;    0381  5
;    0382  4                        END ;                                                      ![10]
;    0383  4
;    0384  4                        ! update the length                                        ![10]
;    0385  4
;    0386  4                      LENGTH2 = .LENGTH1 + 4 ;                                     ![10]
;    0387  4
;    0388  4                      END                                                        ![10]
;    0389  4
;    0390  3                    ELSE                                                         ![10]
;    0391  3
;    0392  3                      CH$MOVE(.LENGTH1,CH$PTR(.ADDRESS1),CH$PTR(FILE_NAME) ) ;   ![10]
;    0393  3
;    0394  3                  LOAD_SELF( FILE2, .ADDRESS, XAB_SIZE ) ;                         ![10]
;    0395  3
;    0396  2                END;                                                             ![10]
;    0397  2
;    0398  2
;    0399  2                ! If we are loading  something at address 200                    ![05]
;    0400  2                ! -- i., e., a diagnostic program load, then see if there        ![05]
;    0401  2                ! is an image header.  (Do this by checking the first            ![05]
;    0402  2                ! longword loaded.) If the header exists, move                   ![05]
;    0403  2                ! it to its own buffer space and move the rest of what           ![05]
;    0404  2                ! was read in down one page, so that the actual image will       ![05]
;    0405  2                ! reside at 200.  If no header, the first record of the          ![05]
;    0406  2                ! image should be the "diagnostic header".  If the first         ![05]
;    0407  2                ! location of this header is not what the routine expects,       ![05]
;    0408  2                ! the operator is warned of the discrepancy.                     ![05]
;    0409  2                ! The reason for this is that if the image header or diagnostic  ![05]
;    0410  2                ! header size is changed, this code will also have to be         ![05]
;    0411  2                ! changed, and the warning will be a reminder that the change    ![05]
;    0412  2                ! must be made.  Note that if either of the header sizes         ![05]
;    0413  2                ! is changed, the routine will assume there is no image header.  ![05]
;    0414  2
;    0415  2                IF .ADDRESS EQL %X'200'                                          ![05]
;    0416  2                THEN                                                             ![05]
;    0417  3                   IF (..ADDRESS EQL %X'00300084')
;    0418  3                   or (..ADDRESS EQL %X'003000A8')                  ![08][09]
;    0419  2                        THEN                                                     ![05]
;    0420  3                                BEGIN                                            ![05]
;    0421  3                                LOCAL PTR1, PTR2;                                ![05]
;    0422  3
;    0423  3                                ! Move the image header to its                   ![05]
;    0424  3                                ! storage area for use by the                    ![05]
;    0425  3                                ! debugger.                                      ![05]
;    0426  3
;    0427  3                                PTR1 = IMAGE_HEADER;                             ![05]
;    0428  3                                PTR2 = .ADDRESS;                                 ![05]
;    0429  3                                WHILE .PTR1 LSSU IMAGE_HEADER_END                ![05]
;    0430  3                                DO                                               ![05]
;    0431  4                                        BEGIN                                    ![05]
;    0432  4                                        .PTR1 = ..PTR2;                          ![05]
;    0433  4                                        PTR1 = .PTR1 + 4;                        ![05]
;    0434  4                                        PTR2 = .PTR2 + 4;                        ![05]
```

```
;   0435  3                                 END;                                    ![05]
;   0436  3
;   0437  3                         ! Now everything already loaded                 ![05]
;   0438  3                         ! must be moved down one page,                  ![05]
;   0439  3                         ! so that the diag. header                      ![05]
;   0440  3                         ! starts at 200.                                ![05]
;   0441  3
;   0442  3                         PTR1 = .ADDRESS;                                ![05]
;   0443  3                         PTR2 = .ADDRESS + %X'200';                      ![05]
;   0444  3                         WHILE .PTR2 LSSU .ADDR DO                       ![05]
;   0445  4                                 BEGIN                                   ![05]
;   0446  4                                 .PTR1 = ..PTR2;                         ![05]
;   0447  4                                 PTR1 = .PTR1 + 4;                       ![05]
;   0448  4                                 PTR2 = .PTR2 + 4;                       ![05]
;   0449  3                                 END;                                    ![05]
;   0450  3                         ADDR = .ADDR - %X'200';                         ![05]
;   0451  3                         LEN = .LEN + %X'200';                           ![05]
;   0452  3                         END                                             ![05]
;   0453  2                 ELSE IF ..ADDRESS NEQ %X'00000058'
;   0454  2                  THEN
;   0455  2                         $DS_PRINTF (FMT_BAD_HDR);
;   0456  2
;   0457  2         if actualcount () geq 5 then .retlen = max ( .XAB_SIZE , .addr - .address);
;   0458  2                 ! Bigger of XAB size and actually loaded size           [01]
;   0459  2
;   0460  2         .status
;   0461  2
;   0462  1       end ;                         ! end routine


                                                                        .TITLE  DSLOAD *** DSLOAD DS$LOAD routine
                                                                        .IDENT  \10.8-12\

                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

4F  4C  24  58  53  44  20  65  6E  69  74  75  6F  52  5E  00000 P.AAA:  .ASCII  \^Routine DSX$LOAD has detected non-stand\  ;
64  65  74  63  65  74  65  64  20  73  61  68  20  44  41  0000F                                                            ;
                    64  6E  61  74  73  2D  6E  6F  6E  20  0001E                                                            ;
6D  69  20  72  6F  20  2E  67  61  69  64  20  64  72  61  00028         .ASCII  \ard diag. or image header.!/Assuming no \  ;
73  41  2F  21  2E  72  65  64  61  65  68  20  65  67  61  00037                                                            ;
                    20  6F  6E  20  67  6E  69  6D  75  73  00046                                                            ;
2F  21  2E  72  65  64  61  65  68  20  65  67  61  6D  69  00050         .ASCII  \image header.!/\<0>                        ;
                                                        00  0005F                                                            ;

                                                                        .PSECT  WORK,NOEXE,2

                                                        00000 PTR_BEG:.BLKB   4
                                                        00004 PTR_COM:.BLKB   4
                                                        00008 PTR_DIF:.BLKB   4
                                                        0000C VER_LENGTH:
                                                                        .BLKB   4
                                                        00010 FILE2:  .BLKB   8
                                                        00018 FILE_NAME:
                                                                        .BLKB   32
                                                        00038 EXTENSION:
                                                                        .BLKB   4
```

```
                                                DSA$AL_APTMAIL=     65024
                                                DSA$GL_FLAGS=       65024
                                                DSA$GL_APTCOM=      65028
                                                DSA$GL_PASSES=      65032
                                                DSA$GL_UNITS=       65036
                                                DSA$GL_SECTNO=      65040
                                                DSA$GL_SID=         65044
                                                DSA$GL_MSGTYP=      65088
                                                DSA$GL_ERRNO=       65092
                                                DSA$GL_EVENT=       65096
                                                DSA$GL_SUBTNO=      65100
                                                DSA$GL_TESTNO=      65104
                                                DSA$GL_PASSNO=      65108
                                                DSA$GL_DEVLEN=      65112
                                                DSA$GL_DEVNAM=      65116
                                                DSA$GL_MSGPTR=      65128
                                                FMT_BAD_HDR=        P.AAA
                                                LENGTH2=            FILE2
                                                ADDRESS2=           FILE2+4
                                                        .EXTRN  DS$GL_FLAGS, IMAGE_HEADER
                                                        .EXTRN  IMAGE_HEADER_END
                                                        .EXTRN  SYS$OPEN, SYS$CONNECT
                                                        .EXTRN  SYS$SPACE, SYS$READ
                                                        .EXTRN  SYS$DISCONNECT, SYS$CLOSE
                                                        .EXTRN  DS$PRINTF
                                                        .WEAK   LOAD_SELF

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                0FFC 00000      .ENTRY  DSX$LOAD, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0152
                                                                        R11                                         ;
                                        5E      FF3C    CE  9E 00002    MOVAB   -196(SP), SP                        ;
                                        56      10      AC  D0 00007    MOVL    ADDRESS, ADDR                       ; 0197
                                        57      04      AC  D0 0000B    MOVL    FILE, R7                            ; 0212
                        03 00000000G    EF              02  E1 0000F    BBC     #2, DS$GL_FLAGS+3, 1$               ; 0199
                                                    016D    31 00017    BRW     14$                                 ;
0050    8F              00              6E              00  2C 0001A 1$: MOVC5   #0, (SP), #0, #80, $RMS_PTR         ; 0212
                                                74      AE      00021                                               ;
                                        74      AE      5003    8F  B0 00023    MOVW    #20483, $RMS_PTR            ;
                                        C6      AD              22  90 00029    MOVB    #34, $RMS_PTR+22            ;
                                        CF      AD              02  90 0002D    MOVB    #2, $RMS_PTR+31             ;
                                        D4      AD      04      AE  9E 00031    MOVAB   XAB, $RMS_PTR+36            ;
                                        DC      AD      04      A7  D0 00036    MOVL    4(R7), $RMS_PTR+44          ;
                                                50      08      AC  D0 0003B    MOVL    DEFAULT, R0                 ;
                                        E0      AD      04      A0  D0 0003F    MOVL    4(R0), $RMS_PTR+48          ;
                                        E4      AD              67  90 00044    MOVB    (R7), $RMS_PTR+52           ;
                                        E5      AD              60  90 00048    MOVB    (R0), $RMS_PTR+53           ;
        2C              00              6E              00  2C 0004C    MOVC5   #0, (SP), #0, #44, $RMS_PTR         ; 0213
                                                04      AE      00051                                               ;
                                        04      AE      2C1D    8F  B0 00053    MOVW    #11293, $RMS_PTR            ;
0044    8F              00              6E              00  2C 00059    MOVC5   #0, (SP), #0, #68, $RMS_PTR         ; 0214
                                                30      AE      00060                                               ;
                                        30      AE      4401    8F  B0 00062    MOVW    #17409, $RMS_PTR            ;
                                          4E    AE      94 00068    CLRB    $RMS_PTR+30                         ;
                                        6C      AE      74  AE  9E 0006B    MOVAB   FAB, $RMS_PTR+60            ;
                                                74      AE  9F 00070    PUSHAB  FAB                             ; 0216
                    00000000G   00              01  FB 00073    CALLS   #1, SYS$OPEN                        ;
```

```
                    54          50  D0 0007A        MOVL    R0, STATUS                          ;
                    03          54  E8 0007D        BLBS    STATUS, 2$                          ;
                              028A  31 00080        BRW     31$                                 ;
      50       14   AE          09  78 00083 2$:    ASHL    #9, XAB+16, R0                      ; 0218
                    51      18  AE  3C 00088        MOVZWL  XAB+20, R1                          ;
                    6E    FE00 C140 9E 0008C        MOVAB   -512(R1)[R0], XAB_SIZE              ;
                    06          6C  91 00092        CMPB    (AP), #6                            ; 0221
                                12  1F 00095        BLSSU   3$                                  ;
                    50      18  AC  D0 00097        MOVL    RETREC, R0                          ; 0228
                    60      E6  AD  B0 0009B        MOVW    FAB+54, (R0)                        ;
                02  A0      CF  AD  90 0009F        MOVB    FAB+31, 2(R0)                       ; 0229
                03  A0      EF  AD  90 000A4        MOVB    FAB+63, 3(R0)                       ; 0230
                            0C  AC  D5 000A9 3$:    TSTL    LENGTH                              ; 0234
                                5A  13 000AC        BEQL    5$                                  ;
                            30  AE  9F 000AE        PUSHAB  RAB                                 ; 0242
          00000000G 00          01  FB 000B1        CALLS   #1, SYS$CONNECT                     ;
                    54          50  D0 000B8        MOVL    R0, STATUS                          ;
                    4A          54  E9 000BB        BLBC    STATUS, 5$                          ;
                    07          6C  91 000BE        CMPB    (AP), #7                            ; 0244
                                4C  1F 000C1        BLSSU   7$                                  ;
      3F 0000FE03   9F          04  E1 000C3        BBC     #4, @#^X0000FE03, 6$                ; 0246
      3A       F0   AD          05  E1 000CB        BBC     #5, FAB+64, 6$                      ; 0247
                    52      EC  AD  3C 000D0        MOVZWL  FAB+60, BLSIZE                      ; 0254
                    52 00000200 8F  C6 000D4        DIVL2   #512, BLSIZE                        ;
      50       1C   AC          01  C3 000DB        SUBL3   #1, VBN, R0                         ; 0255
      51            50          52  C7 000E0        DIVL3   BLSIZE, R0, R1                      ;
                    52          51  C4 000E4        MULL2   R1, R2                              ;
                    50          52  D1 000E7        CMPL    R2, R0                              ;
                                08  13 000EA        BEQL    4$                                  ;
                    50 00660002 8F  D0 000EC        MOVL    #6684674, R0                        ; 0257
                                    04 000F3        RET                                         ;
                68  AE          51  D0 000F4 4$:    MOVL    R1, RAB+56                          ; 0259
                            30  AE  9F 000F8        PUSHAB  RAB                                 ; 0260
          00000000G 00          01  FB 000FB        CALLS   #1, SYS$SPACE                       ;
                    54          50  D0 00102        MOVL    R0, STATUS                          ;
                    07          54  E8 00105        BLBS    STATUS, 7$                          ; 0262
                                70  11 00108 5$:    BRB     13$                                 ;
                68  AE      1C  AC  D0 0010A 6$:    MOVL    VBN, RAB+56                         ; 0265
                    53      0C  AC  D0 0010F 7$:    MOVL    LENGTH, LEN                         ; 0267
                    52          01  D0 00113        MOVL    #1, FIRST                           ; 0268
                54  AE          56  D0 00116 8$:    MOVL    ADDR, RAB+36                        ; 0272
          0000FE00  8F          53  D1 0011A        CMPL    LEN, #65024                         ; 0273
                                08  15 00121        BLEQ    9$                                  ;
                50  AE    FE00  8F  B0 00123        MOVW    #-512, RAB+32                       ; 0275
                                04  11 00129        BRB     10$                                 ;
                50  AE          53  B0 0012B 9$:    MOVW    LEN, RAB+32                         ; 0277
                            30  AE  9F 0012F 10$:   PUSHAB  RAB                                 ; 0278
          00000000G 00          01  FB 00132        CALLS   #1, SYS$READ                        ;
                    54          50  D0 00139        MOVL    R0, STATUS                          ;
                            68  AE  D4 0013C        CLRL    RAB+56                              ; 0279
                    50      52  AE  3C 0013F        MOVZWL  RAB+34, R0                          ; 0280
                    56          50  C0 00143        ADDL2   R0, ADDR                            ;
                    14          54  E9 00146        BLBC    STATUS, 11$                         ; 0282
                    50      52  AE  3C 00149        MOVZWL  RAB+34, R0                          ; 0284
                    53          50  C2 0014D        SUBL2   R0, LEN                             ;
                                52  D4 00150        CLRL    FIRST                               ; 0286
                                53  D5 00152        TSTL    LEN                                 ; 0288
```

```
                                          C0  12 00154          BNEQ    8$                                          ;
                          54 00010001     8F  D0 00156          MOVL    #65537, STATUS                              ; 0291
                          10              52  E8 0015D 11$:     BLBS    FIRST, 12$                                  ; 0297
                 0001827A 8F              54  D1 00160          CMPL    STATUS, #98938                              ;
                                          07  12 00167          BNEQ    12$                                         ;
                          54 00010001     8F  D0 00169          MOVL    #65537, STATUS                              ; 0299
                                    30    AE  9F 00170 12$:     PUSHAB  RAB                                         ; 0301
                 00000000G 00             01  FB 00173          CALLS   #1, SYS$DISCONNECT                          ;
                                    74    AE  9F 0017A 13$:     PUSHAB  FAB                                         ; 0306
                 00000000G 00             01  FB 0017D          CALLS   #1, SYS$CLOSE                               ;
                                        00FA  31 00184          BRW     22$                                         ;
                 00000000' EF             67  D0 00187 14$:     MOVL    (R7), LENGTH2                               ; 0340
                 00000000' EF 00000000'   EF  9E 0018E          MOVAB   FILE_NAME, ADDRESS2                         ; 0341
                 00000000' EF 4558452E    8F  D0 00199          MOVL    #1163412782, EXTENSION                      ; 0343
                          52        04    A7  D0 001A4          MOVL    4(R7), R2                                   ; 0345
              62          67              2E  3A 001A8          LOCC    #46, (R7), (R2)                             ;
                                          02  12 001AC          BNEQ    15$                                         ;
                                          51  D4 001AE          CLRL    R1                                          ;
                                          51  D5 001B0 15$:     TSTL    R1                                          ;
                                          03  13 001B2          BEQL    16$                                         ;
                                        00B0  31 001B4          BRW     20$                                         ;
              62          67              3B  3A 001B7 16$:     LOCC    #59, (R7), (R2)                             ; 0355
                                          02  12 001BB          BNEQ    17$                                         ;
                                          51  D4 001BD          CLRL    R1                                          ;
                 00000000' EF             51  D0 001BF 17$:     MOVL    R1, PTR_COM                                 ;
                          59 00000000'    EF  D0 001C6          MOVL    PTR_COM, R9                                 ; 0357
                                          28  12 001CD          BNEQ    18$                                         ;
              58          67              04  C1 001CF          ADDL3   #4, (R7), R8                                ; 0364
                          5A              58  D0 001D3          MOVL    R8, R10                                     ;
                          5B 00000000'    EF  9E 001D6          MOVAB   FILE_NAME, R11                              ;
 5A           00          62              67  2C 001DD          MOVC5   (R7), (R2), #0, R10, (R11)                  ;
                                          6B     001E2                                                              ;
                                          79  18 001E3          BGEQ    19$                                         ;
                          5B              67  C0 001E5          ADDL2   (R7), R11                                   ;
                          5A              67  C2 001E8          SUBL2   (R7), R10                                   ;
 5A           00 00000000' EF             04  2C 001EB          MOVC5   #4, EXTENSION, #0, R10, (R11)               ;
                                          6B     001F4                                                              ;
                                          67  11 001F5          BRB     19$                                         ; 0363
                 00000000' EF             52  D0 001F7 18$:     MOVL    R2, PTR_BEG                                 ; 0372
    00000000' EF          59 00000000'    EF  C3 001FE          SUBL3   PTR_BEG, R9, PTR_DIF                        ; 0374
    00000000' EF          67 00000000'    EF  C3 0020A          SUBL3   PTR_DIF, (R7), VER_LENGTH                   ; 0376
              58          67              04  C1 00216          ADDL3   #4, (R7), R8                                ; 0380
                          5A              58  D0 0021A          MOVL    R8, R10                                     ;
                          57 00000000'    EF  9E 0021D          MOVAB   FILE_NAME, R7                               ;
 5A           00 00000000' FF 00000000'   EF  2C 00224          MOVC5   PTR_DIF, @PTR_BEG, #0, R10, (R7)            ;
                                          67     00231                                                              ;
                                          2A  18 00232          BGEQ    19$                                         ;
                          57 00000000'    EF  C0 00234          ADDL2   PTR_DIF, R7                                 ;
                          5A 00000000'    EF  C2 0023B          SUBL2   PTR_DIF, R10                                ;
 5A           00 00000000' EF             04  2C 00242          MOVC5   #4, EXTENSION, #0, R10, (R7)                ;
                                          67     0024B                                                              ;
                                          10  18 0024C          BGEQ    19$                                         ;
                          57              04  C0 0024E          ADDL2   #4, R7                                      ;
                          5A              04  C2 00251          SUBL2   #4, R10                                     ;
 5A           00          69 00000000'    EF  2C 00254          MOVC5   VER_LENGTH, (R9), #0, R10, (R7)             ;
                                          67     0025D                                                              ;
                 00000000' EF             58  D0 0025E 19$:     MOVL    R8, LENGTH2                                 ; 0386
```

```
                                                08  11  00265          BRB     21$                                     ;
                00000000'  EF            62     67  28  00267  20$:    MOVC3   (R7), (R2), FILE_NAME                   ; 0392
                                                5E  DD  0026F  21$:    PUSHL   SP                                      ; 0394
                                          10    AC  DD  00271          PUSHL   ADDRESS                                 ;
                                    00000000'   EF  9F  00274          PUSHAB  FILE2                                   ;
                        00000000G   EF          03  FB  0027A          CALLS   #3, LOAD_SELF                           ;
                        00000200    8F        10    AC  D1  00281  22$:    CMPL    ADDRESS, #512                   ; 0415
                                                6A  12  00289          BNEQ    29$                                     ;
                        00300084    8F        10    BC  D1  0028B          CMPL    @ADDRESS, #3145860              ; 0417
                                                0A  13  00293          BEQL    23$                                     ;
                        003000A8    8F        10    BC  D1  00295          CMPL    @ADDRESS, #3145896              ; 0418
                                                3F  12  0029D          BNEQ    28$                                     ;
                                    51  00000000G   EF  9E  0029F  23$:    MOVAB   IMAGE_HEADER, PTR1              ; 0427
                                    50        10    AC  D0  002A6          MOVL    ADDRESS, PTR2                   ; 0428
                                    52  00000000G   EF  9E  002AA  24$:    MOVAB   IMAGE_HEADER_END, R2            ; 0429
                                    52              51  D1  002B1          CMPL    PTR1, R2                        ;
                                                05  1E  002B4          BGEQU   25$                                     ;
                                    81              80  D0  002B6          MOVL    (PTR2)+, (PTR1)+                ; 0432
                                                EF  11  002B9          BRB     24$                                     ; 0434
                                    51        10    AC  D0  002BB  25$:    MOVL    ADDRESS, PTR1                   ; 0442
                        50        10    AC  00000200  8F  C1  002BF          ADDL3   #512, ADDRESS, PTR2             ; 0443
                                    56              50  D1  002C8  26$:    CMPL    PTR2, ADDR                      ; 0444
                                                05  1E  002CB          BGEQU   27$                                     ;
                                    81              80  D0  002CD          MOVL    (PTR2)+, (PTR1)+                ; 0446
                                                F6  11  002D0          BRB     26$                                     ; 0448
                                    56      FE00    C6  9E  002D2  27$:    MOVAB   -512(R6), ADDR                  ; 0450
                                    53      0200    C3  9E  002D7          MOVAB   512(R3), LEN                    ; 0451
                                                17  11  002DC          BRB     29$                                     ;
                        00000058    8F        10    BC  D1  002DE  28$:    CMPL    @ADDRESS, #88                   ; 0453
                                                0D  13  002E6          BEQL    29$                                     ;
                                    00000000'   EF  9F  002E8          PUSHAB  FMT_BAD_HDR                             ; 0455
                        00000000G   9F          01  FB  002EE          CALLS   #1, @#DS$PRINTF                         ;
                                    05          6C  91  002F5  29$:    CMPB    (AP), #5                        ; 0457
                                                13  1F  002F8          BLSSU   31$                                     ;
                                    56        10    AC  C2  002FA          SUBL2   ADDRESS, R6                     ;
                                    50              6E  D0  002FE          MOVL    XAB_SIZE, R0                    ;
                                    56              50  D1  00301          CMPL    R0, R6                          ;
                                                03  18  00304          BGEQ    30$                                     ;
                                    50              56  D0  00306          MOVL    R6, R0                          ;
                              14    BC              50  D0  00309  30$:    MOVL    R0, @RETLEN                     ;
                                    50              54  D0  0030D  31$:    MOVL    STATUS, R0                      ; 0462
                                                    04  00310          RET                                             ;

; Routine Size:  785 bytes,    Routine Base:  CODE + 0000

;   0463  1
;   0464  1      END
;   0465  0      ELUDOM



;                                   PSECT SUMMARY
;
```

```
;       Name                    Bytes                   Attributes
;
;  DATA                           96  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                           60  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          785  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



Symbol                Type  Defined  Referenced ...
--------------------- ------- ----- ---------------
$CLOSE                KeyWMacr Lib03   306
$CONNECT              KeyWMacr Lib03   242
$DISCONNECT           KeyWMacr Lib03   301
$DS_DSADEF            Macro    Lib01   105
$DS_PRINTF            Macro    Lib01   455
$FAB_DECL             Macro    Lib03   204
$FAB_INIT             KeyWMacr Lib03   211
$OPEN                 KeyWMacr Lib03   216
$RAB_DECL             Macro    Lib03   205
$RAB_INIT             KeyWMacr Lib03   214
$READ                 KeyWMacr Lib03   278
$RMS_BITFLD_INI       Macro    Lib03   212      214
$RMS_BITS             Macro    Lib03   212
$RMS_CODFLD_INI       Macro    Lib03   212      214
$RMS_OR               Macro    Lib03   212
$RMS_PTR              Unbound          214
                      Bind       212   212=
                      Bind       213   213=
                      Bind       214   214=
$RMS_VALFLD_INI       Unbound          214
                      Macro    Lib03   212      213      214
$SPACE                KeyWMacr Lib03   260
$XABFHC_DECL          Macro    Lib03   206
$XABFHC_INIT          KeyWMacr Lib03   213
ACTUALCOUNT           Builtin    191   221      244      457
ADDR                  Local      183   197=     272.     280=     280.     444.     450=     450.     457.
ADDRESS               RoutForm   152   197.     394.     415.     417a.    418a.    428.     442.     443.     453a.    457.
ADDRESS1              Bind       335   345a.    355a.    363a.    372.     392a.
ADDRESS2              Bind       332   341=
AP                    Builtin    221   221a.
                      Builtin    244   244a.
                      Builtin    457   457a.
BLSIZE                Local      252   254=     255.     259.
DEFAULT               RoutForm   152   209m     212a.
DEV$M_SQD             Literal  Lib03   247
DS$GL_FLAGS           External   120   199.
DS$PRINTF             ExtRout    455   455c
DS$V_NI               Macro    Lib02   199
DS$_ERROR             Literal  Lib01   257
DSA$AL_APTMAIL        Bind       105   105
DSA$GL_APTCOM         Bind       105
DSA$GL_DEVLEN         Bind       105
DSA$GL_DEVNAM         Bind       105
DSA$GL_ERRNO          Bind       105
DSA$GL_EVENT          Bind       105
```

Symbol	Type	Defined	Referenced ...							
DSA$GL_FLAGS	Bind	105	246							
DSA$GL_MSGPTR	Bind	105								
DSA$GL_MSGTYP	Bind	105								
DSA$GL_PASSES	Bind	105								
DSA$GL_PASSNO	Bind	105								
DSA$GL_SECTNO	Bind	105								
DSA$GL_SID	Bind	105								
DSA$GL_SUBTNO	Bind	105								
DSA$GL_TESTNO	Bind	105								
DSA$GL_UNITS	Bind	105								
DSA$V_USER	Macro	Lib01	246							
DSX$LOAD	GlobRout	152	112f							
EXTENSION	Own	329	343=	363.	378.					
FAB	Local	204	212a	214a	216a	228.	229.	230.	247.	254.	306a
FAB$B_BID	Macro	Lib03	212							
FAB$B_BKS	Macro	Lib03	212							
FAB$B_BLN	Macro	Lib03	212							
FAB$B_DNS	Macro	Lib03	212							
FAB$B_FAC	Macro	Lib03	212							
FAB$B_FNS	Macro	Lib03	212							
FAB$B_FSZ	Macro	Lib03	212	230						
FAB$B_JOURNAL	Macro	Lib03	212							
FAB$B_ORG	Macro	Lib03	212							
FAB$B_RAT	Macro	Lib03	212							
FAB$B_RFM	Macro	Lib03	212	229						
FAB$B_RTV	Macro	Lib03	212							
FAB$B_RU_FACILITY	Macro	Lib03	212							
FAB$B_SHR	Macro	Lib03	212							
FAB$C_BID	Literal	Lib03	212							
FAB$C_BLN	Literal	Lib03	204	212						
FAB$C_VAR	Literal	Lib03	212							
FAB$L_ALQ	Macro	Lib03	212							
FAB$L_CTX	Macro	Lib03	212							
FAB$L_DEV	Macro	Lib03	247							
FAB$L_DNA	Macro	Lib03	212							
FAB$L_FNA	Macro	Lib03	212							
FAB$L_FOP	Macro	Lib03	212							
FAB$L_MRN	Macro	Lib03	212							
FAB$L_NAM	Macro	Lib03	212							
FAB$L_XAB	Macro	Lib03	212							
FAB$M_BIO	Literal	Lib03	212							
FAB$M_GET	Literal	Lib03	212							
FAB$V_CHAN_MODE	Macro	Lib03	212							
FAB$V_FILE_MODE	Macro	Lib03	212							
FAB$V_LNM_MODE	Macro	Lib03	212							
FAB$W_BLS	Macro	Lib03	212	254						
FAB$W_DEQ	Macro	Lib03	212							
FAB$W_GBC	Macro	Lib03	212							
FAB$W_MRS	Macro	Lib03	212	228						
FILE	RoutForm	152	188m	212a.	334.	335aa				
FILE2	Own	326	331a	332a	394a					
FILE_NAME	Own	328	341a	364=	380=	392=				
FIRST	Local	240	268=	286=	297.					
FMT_BAD_HDR	Bind	142	455a							
IMAGE_HEADER	External	146	427a							

Symbol	Type	Defined	Referenced ...
IMAGE_HEADER_END	External	147	429
LEN	Local	185	267= 273. 277. 284= 284. 288. 451= 451.
LENGTH	RoutForm	152	234. 267.
LENGTH1	Bind	334	340. 345. 355. 363. 364. 376. 380. 386. 392.
LENGTH2	Bind	331	340= 386=
LOAD_SELF	ExtRout	338	394c
PTR1	Local	421	427= 429. 432a= 433= 433. 442= 446a= 447= 447.
PTR2	Local	421	428= 432a. 434= 434. 443= 444. 446a. 448= 448.
PTR_BEG	Own	324	372= 374. 378a.
PTR_COM	Own	324	355= 357. 374. 379a.
PTR_DIF	Own	324	374= 376. 378.
RAB	Local	205	214a 242a 259= 260a 265= 272= 275= 277= 278a 279= 280. 284. 301a
RAB$B_BID	Macro	Lib03	214
RAB$B_BLN	Macro	Lib03	214
RAB$B_KRF	Macro	Lib03	214
RAB$B_KSZ	Unbound		214
RAB$B_MBC	Macro	Lib03	214
RAB$B_MBF	Macro	Lib03	214
RAB$B_RAC	Macro	Lib03	214
RAB$B_TMO	Macro	Lib03	214
RAB$C_BID	Literal	Lib03	214
RAB$C_BLN	Literal	Lib03	205 214
RAB$C_SEQ	Literal	Lib03	214
RAB$L_BKT	Macro	Lib03	214 259 265 279
RAB$L_CTX	Macro	Lib03	214
RAB$L_FAB	Macro	Lib03	214
RAB$L_KBF	Unbound		214
RAB$L_RBF	Macro	Lib03	214
RAB$L_RHB	Macro	Lib03	214
RAB$L_ROP	Macro	Lib03	214
RAB$L_UBF	Macro	Lib03	214 272
RAB$L_XAB	Macro	Lib03	214
RAB$W_RSZ	Macro	Lib03	214 280 284
RAB$W_USZ	Macro	Lib03	214 275 277
READ	Label	194	236 242 262
RETLEN	RoutForm	152	457a=
RETREC	RoutForm	152	226m 228a= 229a= 230a=
RMS$_EOF	Literal	Lib03	297
RMS$_NORMAL	Literal	Lib03	291 299
SDL$STARLET_CONCAT	Macro	Lib03	216 242 260 278 301 306
SDL$STARLET_OPT	Macro	Lib03	216 242 260 278 301 306
SDL$STARLET_REQ	Macro	Lib03	216 242 260 278 301 306
STATUS	Local	184	216= 216. 242= 260= 262. 278= 282. 291= 297. 299= 460.
SYS$CLOSE	ExtRout	306	306c
SYS$CONNECT	ExtRout	242	242c
SYS$DISCONNECT	ExtRout	301	301c
SYS$OPEN	ExtRout	216	216c
SYS$READ	ExtRout	278	278c
SYS$SPACE	ExtRout	260	260c
VBN	RoutForm	152	255. 259. 265.
VER_LENGTH	Own	324	376= 379.
XAB	Local	206	212a 213a 218.
XAB$B_BLN	Macro	Lib03	213
XAB$B_COD	Macro	Lib03	213

```
Symbol                Type  Defined  Referenced ...
------------------ -------- ----- ----------------
XAB$C_FHC           Literal  Lib03   213
XAB$C_FHCLEN        Literal  Lib03   206     213
XAB$L_EBK           Macro    Lib03   218
XAB$L_NXT           Macro    Lib03   213
XAB$W_FFB           Macro    Lib03   218
XAB_SIZE            Local      182   218=    394a    457.


                CROSS REFERENCE MAP

Line #      Event          File ...
-----  --------------  ---------
    1  Source (start)   DISK$DS:[DS.LOCAL.RELEASE.WORK]DSLOAD.B32;1
    6  Module DSLOAD
   99  Library #1       DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
  101  Library #2       DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
  103  Library #3       SYS$COMMON:[SYSLIB]LIB.L32;2
  465  Eludom DSLOAD


        KEY TO REFERENCE TYPE FLAGS

            .   Fetch
            =   Store
            c   Routine call
            a   Address use
            @   Indirect use
            e   External, external routine, or external literal declaration
            f   Forward or forward routine declaration
            h   Condition handler enabling
            m   Map declaration
            u   Undeclare declaration


;                              Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                            940         4         0        96          00:00.1
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                            675         1         0        47          00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2             20450        85         0      1101          00:00.8




;                                   COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:DSLOAD.B32
```

```
;   0466  0
; Size:          785 code + 156 data bytes
; Run Time:        00:03.8
; Elapsed Time:    00:05.8
; Lines/CPU Min:    7436
; Lexemes/CPU-Min:104218
; Memory Used:  260 pages
; Compilation Complete
```

Table of contents

(1) 96 Libraries and Macros
(1) 102 The Dispatch Vectors

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element DSVECS.MAR
0000     2 ; *1     6-FEB-1986 15:17:55 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element DSVECS.MAR
0000     4         .Title  DSVECS  *** DSVECS Module
0000     5         .Ident  /01-10/
0000     6         .NoShow Conditionals
0000     7
0000     8 ;++
0000     9 ; COPYRIGHT (C) 1977,1980,1985
0000    10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    11 ;
0000    12 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    13 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    14 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    15 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    16 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    17 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    18 ; REMAIN IN DEC.
0000    19 ;
0000    20 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    21 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    22 ; CORPORATION.
0000    23 ;
0000    24 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    26 ;--
```

```
                        0000     28 ;**
                        0000     29 ; Facility:
                        0000     30 ;
                        0000     31 ;       VAX Diagnostic Supervisor (part of the Resident-DS image).
                        0000     32 ;
                        0000     33 ; Abstract:
                        0000     34 ;
                        0000     35 ;       This module contains the dispatch vectors that the CRD-Loadable
                        0000     36 ;       routines need to access certain Resident-DS locations and routines.
                        0000     37 ;
                        0000     38 ; Environment:
                        0000     39 ;
                        0000     40 ;       Not applicable
                        0000     41 ;
                        0000     42 ; Author:
                        0000     43 ;
                        0000     44 ;       Jack Stansbury
                        0000     45 ;
                        0000     46 ; Date Written:
                        0000     47 ;
                        0000     48 ;       March 30, 1982
                        0000     49 ;
                        0000     50 ; Modifications:
                        0000     51 ;
                        0000     52 ;       01      Jack Stansbury, Version 1.0, June 22, 1982
                        0000     53 ;               Added two more vectors.
                        0000     54 ;
                        0000     55 ;       02      Jack Stansbury, Version 1.1, July 24, 1982
                        0000     56 ;               Added the CRD$GB_Menu_OnLine_Debug byte. If == 1, then
                        0000     57 ;               on-line CRD command is allowed. If == 0, the CRD command
                        0000     58 ;               is disallowed. It is == 0 by default.
                        0000     59 ;
                        0000     60 ;       03      Jack Stansbury, Version 1.1, July 27, 1982
                        0000     61 ;               Added the DS$GA_DS_Ctrl_C_First (formerly DS$GA_DSCntrlC) and
                        0000     62 ;               the DS$GA_DS_Ctrl_C_Second longwords to the dispatch list.
                        0000     63 ;
                        0000     64 ;       04      Jack Stansbury, Version 1.1, August 19, 1982
                        0000     65 ;               Added the Begin and DS$GL_CRD_Debug vectors. Also changed the
                        0000     66 ;               version number bytes from 1 to 2.
                        0000     67 ;
                        0000     68 ;       05      Jack Stansbury, Version 1.1, September 10, 1982
                        0000     69 ;               Added the Boot$L_R5 address. This is used by Menu Test to allow
                        0000     70 ;               it to determine how CRD was invoked.
                        0000     71 ;
                        0000     72 ;       06      Jack Stansbury, Version 1.1, September 12, 1982
                        0000     73 ;               Took out the one added in 05 above, and added the
                        0000     74 ;               DSR$Check_MenuTest_Set and the DSR$Check_AutoTest_Set routine
                        0000     75 ;               addresses.
                        0000     76 ;
                        0000     77 ;       07      Jack Stansbury, Version 1.1, October 26, 1982
                        0000     78 ;               Added the DS$GL_CRD_Flags longword. This is set up now as a
                        0000     79 ;               'flags' longword. It is now used for CRD_Debug (bit 1) and
                        0000     80 ;               for disabling ^C echoing (bit 0).
                        0000     81 ;
                        0000     82 ;       08      Jack Stansbury, Version ?.?, March 3, 1983
                        0000     83 ;               Added a number of new (empty for now) dispatch vectors. These
                        0000     84 ;               empty ones are to allow the debug vectors (in CRD) to be big
```

```
0000    85 ;                enough to pass enough useful information back to the
0000    86 ;                DSR$Unload_CRD routine.
0000    87 ;
0000    88 ;        09      John Ciukaj     Version 6.11    4-April 1983
0000    89 ;                - Changed name of CRD Online debug vector
0000    90 ;                - Changed reference names to DSR$Check.. routines
0000    91 ;
0000    92 ;        10      Ron Parker      20-FEB-1985
0000    93 ;                - Added address of Ptable descriptor table
0000    94 ;--
```

```
                0000     96          .SubTitle       Libraries and Macros
                0000     97
                0000     98          .Library        /Sys$Library:Lib/       ; Use Lib last
                0000     99          .Library        /$DS/                   ; Use DS.Mar second
                0000    100          .Library        /$Diag/                 ; Use Diag.Mar first
```

```
                     0000   102          .SubTitle       The Dispatch Vectors
                 00000000   103          .Psect          Work, NoShr, NoExe, Wrt, Long
                     0000   104
                     0000   105 DS$GL_CRD_Flags::                          ; Analguous to the DS flags longword
            00000000 0000   106          .Long   0                         ; all bits = 0 by default       [07]
                     0004   107
                     0004   108          ;+
                     0004   109          ; This longword is used for special debug of CRD by CRD Command
                     0004   110          ;-
                     0004   111
                     0004   112 CRD$GL_CRD_Command_Debug::                 ;
            00000000 0004   113          .Long   0                         ; = 0 by default                [09]
                     0008   114
                     0008   115          ;+
                     0008   116          ; These four bytes are for verification purposes. See the
                     0008   117          ; Exchange_Dispat_Vectors routine in the CRDLoad.B32 module for more
                     0008   118          ; of an explanation of these bytes.
                     0008   119          ;-
                     0008   120
                     0008   121 DS$AL_DS_Dispatch_Vectors::
                  19 0008   122          .Byte   25                        ; The # of dispatch vectors     [08]
                  0A 0009   123          .Byte   10                        ; The positive version number   [10]
                  F6 000A   124          .Byte   -10                       ; The negative version number   [10]
                  00 000B   125          .Byte   0                         ; The null byte
                     000C   126
                     000C   127          ;+
                     000C   128          ; These are the actual dispatch vectors. The first vector is the one that
                     000C   129          ; will contain the address of the routine in the CRD-Loadable image that
                     000C   130          ; will be called by the Print routine. Thus, the global label. The dispatch
                     000C   131          ; vectors now contain those routines and locations internal to the
                     000C   132          ; Resident-DS that the CRD routines will have to access. These vectors
                     000C   133          ; will be exchanged with another like-set of vectors that are part of the
                     000C   134          ; CRD-Loadable image.
                     000C   135          ;-
                     000C   136
                     000C   137 DS$GA_CRD_DS_Interface::                   ; Talk to the CRD image
            00000000' 000C   138          .Address        DSV$Exit          ; To exit the Resident-DS
                     0010   139
            00000000' 0010   140          .Address        DSX$Print         ; The DS Print routine
                     0014   141
            00000000' 0014   142          .Address        DS$GL_Flags       ; The DS flags
                     0018   143
            00000000' 0018   144          .Address        DS$GA_DS_Ctrl_C_First ; The DS's 1st Control-C  [03]
                     001C   145                                            ; . . . handler                 [03]
                     001C   146
            00000000' 001C   147          .Address        DS$GA_PTable      ; The list of addresses of
                     0020   148                                            ; ... PTables
                     0020   149
            00000000' 0020   150          .Address        Exe$AloNonPaged   ; Allocate a block of Non-Paged
                     0024   151                                            ; ... pool
                     0024   152
                     0024   153 DS$GA_User_Error_Text::                    ; The address of the user's
            00000024' 0024   154          .Address        DS$GA_User_Error_Text ; ... MsgAdr (from the ERRxxxx
                     0028   155                                            ; ... macro) will be put here.
                     0028   156
            00000000' 0028   157          .Address        Scan$Numeric      ; The address of the            [01]
                     002C   158                                            ; ... Scan$Numeric routine in   [01]
```

```
                 002C                                                  ; ... the PARSE module.          [01]
                 002C
        00000000' 002C          .Address        Exe$DeaNonPaged         ; The address of the Deallocate [01]
                 0030                                                  ; ... Non-Paged pool routine.   [01]
                 0030
        00000000' 0030          .Address        DS$GA_DS_Ctrl_C_Second  ; The DS's 2nd Control-C        [03]
                 0034                                                  ; . . . handler                 [03]
                 0034
        00000000' 0034          .Address        Begin                   ; The Begin label in KERNEL     [04]
                 0038
                 0038  DS$GL_CRD_Debug::                                ; Make it global to be in MAP   [04]
        00000000  0038          .Long           0                       ; If bit (1) = 1, CRD will print[07]
                 003C                                                  ; . . . debug statements. If    [07]
                 003C                                                  ; . . . bit (1) = 0, CRD won't  [07]
                 003C                                                  ; . . . print them.             [07]
                 003C                                                  ; Bit (0) is for disabling ^C   [07]
                 003C                                                  ; . . . echoing.                [07]
                 003C
        00000000' 003C          .Address        DSR$Check_MenuTest_Off_Set
                 0040                                                  ; Return a 1 iff
                 0040                                                  ; CRD MENU Offline is active    [09]
                 0040
        00000000' 0040          .Address        DSR$Check_AutoTest_Off_Set
                 0044                                                  ; Return a 1 iff
                 0044                                                  ; CRD AUTO Offline is active    [09]
                 0044
        00000000' 0044          .Address        DS$GA_PTDESC
                 0048                                                  ; Address of Ptable descriptor
                 0048                                                  ; table                         [10]
                 0048
        00000000' 0048          .Address        DS$ATTACH
                 004C                                                  ; Address of attach routine
                 004C
                 004C          ;+
                 004C          ; The above vectors are the ones that actually contain the VDS          [08]
                 004C          ; addresses that CRD must know about. The ones below are unused except  [08]
                 004C          ; by the Debug_Vector. Note: the total number of these vectors must     [08]
                 004C          ; agree with the CRD$K_Total_Numb_Vectors constant in the CRD.R32       [08]
                 004C          ; library module.                                                       [08]
                 004C          ;-
                 004C
        00000070  004C          .BlkL   <25-16>                         ; CRD$K_Total_Numb_Vectors = 25 [10]
                 0070                                                  ; # of vectors above = 16 [10]
                 0070          .End
```

```
BEGIN                           ********    X   01
CRD$GL_CRD_COMMAND_DEBUG        00000004 RG  D  01
DS$AL_DS_DISPATCH_VECTORS       00000008 RG  D  01
DS$ATTACH                       ********    X   01
DS$GA_CRD_DS_INTERFACE          0000000C RG  D  01
DS$GA_DS_CTRL_C_FIRST           ********    X   01
DS$GA_DS_CTRL_C_SECOND          ********    X   01
DS$GA_PTABLE                    ********    X   01
DS$GA_PTDESC                    ********    X   01
DS$GA_USER_ERROR_TEXT           00000024 RG  D  01
DS$GL_CRD_DEBUG                 00000038 RG  D  01
DS$GL_CRD_FLAGS                 00000000 RG  D  01
DS$GL_FLAGS                     ********    X   01
DSR$CHECK_AUTOTEST_OFF_SET      ********    X   01
DSR$CHECK_MENUTEST_OFF_SET      ********    X   01
DSV$EXIT                        ********    X   01
DSX$PRINT                       ********    X   01
EXE$ALONONPAGED                 ********    X   01
EXE$DEANONPAGED                 ********    X   01
SCAN$NUMERIC                    ********    X   01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
WORK	00000070 (112.)	01 (1.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BEGIN	00000000-XR		167 (1)
CRD$GL_CRD_COMMAND_DEBUG	00000004-R	112 (1)	
DS$AL_DS_DISPATCH_VECTORS	00000008-R	121 (1)	
DS$ATTACH	00000000-XR		189 (1)
DS$GA_CRD_DS_INTERFACE	0000000C-R	137 (1)	
DS$GA_DS_CTRL_C_FIRST	00000000-XR		144 (1)
DS$GA_DS_CTRL_C_SECOND	00000000-XR		164 (1)
DS$GA_PTABLE	00000000-XR		147 (1)
DS$GA_PTDESC	00000000-XR		185 (1)
DS$GA_USER_ERROR_TEXT	00000024-R	153 (1)	154 (1)
DS$GL_CRD_DEBUG	00000038-R	169 (1)	
DS$GL_CRD_FLAGS	00000000-R	105 (1)	
DS$GL_FLAGS	00000000-XR		142 (1)
DSR$CHECK_AUTOTEST_OFF_SET	00000000-XR		181 (1)
DSR$CHECK_MENUTEST_OFF_SET	00000000-XR		177 (1)
DSV$EXIT	00000000-XR		138 (1)
DSX$PRINT	00000000-XR		140 (1)
EXE$ALONONPAGED	00000000-XR		150 (1)
EXE$DEANONPAGED	00000000-XR		161 (1)
SCAN$NUMERIC	00000000-XR		157 (1)

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE      REFERENCES...
---------      -------------
.ADDRESS       138    (1)    140    (1)    142    (1)    144    (1)    147    (1)    150    (1)    154    (1)    157    (1)
               161    (1)    164    (1)    167    (1)    177    (1)    181    (1)    185    (1)    189    (1)
.BLKL          200    (1)
.BYTE          122    (1)    123    (1)    124    (1)    125    (1)
.END           202    (1)
.IDENT         5      (1)
.LIBRARY       100    (1)    98     (1)    99     (1)
.LONG          106    (1)    113    (1)    170    (1)
.NOSHOW        6      (1)
.PAGE          101    (1)    27     (1)    95     (1)
.PSECT         103    (1)
.SUBTITLE      102    (1)    96     (1)
.TITLE         4      (1)

                                      +-------------------------+
                                      ! Performance indicators !
                                      +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     22    00:00:00.01    00:00:00.20
Command processing                881    00:00:00.26    00:00:00.65
Pass 1                            125    00:00:00.17    00:00:00.49
Symbol table sort                   0    00:00:00.00    00:00:00.00
Pass 2                              8    00:00:00.08    00:00:00.24
Symbol table output                 0    00:00:00.01    00:00:00.01
Psect synopsis output               0    00:00:00.00    00:00:00.00
Cross-reference output              1    00:00:00.02    00:00:00.06
Assembler run totals             1037    00:00:00.55    00:00:01.65

The working set limit was 2400 pages.
1888 bytes (4 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 20 non-local and 0 local symbols.
202 source lines were read in Pass 1, producing 12 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.
```

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       0
TOTALS (all libraries)                                 0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:DSVECS.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 213 Libraries and Equated Symbols
(1) 241 Data Psect Declarations
(1) 253 DS Entry Points
(1) 1848 DS$SRVDISP_TABLE - Dispatch table for VDS services
(1) 1912 DSX$SERVICE_DISPATCHER - Service Dispatch Routine
(1) 1960 Resrvd_Entry Routine - For entry errors

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ENTRY.MAR
                        0000     2 ; *2    25-APR-1986 13:37:27 SHELHAMER "<no reason given>"
                        0000     3 ; *1     6-FEB-1986 15:18:04 DS ""
                        0000     4 ; DEC/CMS REPLACEMENT HISTORY, Element ENTRY.MAR
                        0000     5         .TITLE  ENTRY   *** ENTRY DS service entry vectors
                        0000     6         .IDENT  /10-43/                                                ;G:2
                        0000     7         .NoShow Conditionals    ;                                       [28]
                        0000     8
                        0000     9 ;++
                        0000    10 ; COPYRIGHT (c) 1977, 1982, 1983, 1984, 1985
                        0000    11 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000    12 ;
                        0000    13 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    14 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    15 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    16 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    17 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    18 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    19 ; REMAIN IN DEC.
                        0000    20 ;
                        0000    21 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    22 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    23 ; CORPORATION.
                        0000    24 ;
                        0000    25 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    26 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    27
                        0000    28 ;++
                        0000    29 ; FACILITY:
                        0000    30 ;
                        0000    31 ;       VAX DIAGNOSTIC SUPERVISOR.
                        0000    32 ;
                        0000    33 ; ABSTRACT:
                        0000    34 ;
                        0000    35 ;       NONE
                        0000    36 ;
                        0000    37 ; ENVIRONMENT:
                        0000    38 ;
                        0000    39 ;       CEP AND SEP.
                        0000    40 ;
                        0000    41 ; AUTHOR:
                        0000    42 ;
                        0000    43 ;       KEN CHAPMAN     31-OCT-77       VERSION 01.
```

```
0000     45 ;
0000     46 ; MODIFIED BY:
0000     47 ;                    NICK HOWGATE    15-NOV-77       VERSION 02
0000     48 ;         01         CHANGE ENTRY VECTOR DISPATCH FROM 'BRW' TO 'JMP'
0000     49 ;
0000     50 ;                    KEN CHAPMAN     30-NOV-77       VERSION 03
0000     51 ;         02         ADDED SYS$WAKE VECTOR.  REMOVED SYS$CRETVA VECTOR.
0000     52 ;
0000     53 ;                    NICK HOWGATE    15-DEC-77       VERSION 04.
0000     54 ;         03         ADDED SYS$SETPRT ENTRY MASK.
0000     55 ;
0000     56 ;                    N. HOWGATE      21-FEB-78       VERSION 05 (ESSAA-4.00).
0000     57 ;         04         CHANGED ENTRY MASKS FOR SYS$ASSIGN, SYS$DASSGN, SYS$CANCEL,
0000     58 ;                    SYS$GTCHAN, SYS$ALLOC, SYS$DALLOC.
0000     59 ;
0000     60 ;                    KEN CHAPMAN     27-MAR-78       VERSION 06 (ESSAA-4.01).
0000     61 ;         05         CHANGED ENTRY MASKS FOR SYS$EXPREG, SYS$CNTREG, DS$GETBUF,
0000     62 ;                    AND DS$RELBUF.
0000     63 ;         06         ADDED NEW ENTRY NAMES: DS$SUMMARY, DS$CANWAIT, DS$ASKDATA,
0000     64 ;                    DS$ASKVLD, DS$ASKLGCL, DS$ASKSTR, DS$SHOCHAN.
0000     65 ;         07         ADDED '+2' TO SKIP OVER LOCAL ENTRY MASKS IN SOME CASES.
0000     66 ;
0000     67 ;                    Roger Riggs                 12-Jun-1978
0000     68 ;         08         Changed GETMEM, RELMEM, MOVVRT, MOVPHY to be reserved.
0000     69 ;                    previously they were dummy routines in MEMMGT.
0000     70 ;                    Also changed EXPREG, CNTREG, MMOFF, MMON, and SETPRT to
0000     71 ;                    use .VECTOR pseudo op.
0000     72 ;         09         Changed names and entry vectors of:
0000     73 ;                    RGETDATA        DSX$GETDATA
0000     74 ;                    RGETADDRESS     DSX$GETADDRESS
0000     75 ;                    RGETVIELD       DSX$GETVIELD
0000     76 ;                    RGETLOGICAL     DSX$GETLOGICAL
0000     77 ;                    RGETSTRING      DSX$GETSTRING
0000     78 ;                    RGPHARD         DSX$GPHARD
0000     79 ;                    RBREAK
0000     80 ;         10         Added DS$PRINTSIG entry point
0000     81 ;
0000     82 ;                    Roger Riggs, 14-Jan-1979        Version 5.2
0000     83 ;         11         Activated DS$PROBE entry point at 101A0
0000     84 ;
0000     85 ;         12         David R. Butenhof  06-feb-80
0000     86 ;                    Add entry point for EXE$SETAST system call
0000     87 ;
0000     88 ;                    Roger Riggs, 24-MAR-1980
0000     89 ;         13         Added dummy entry points for SYS$LKWSET, SYS$SETPRI,
0000     90 ;                    SYS$SETRWM, SYS$SETSWM, SYS$ULWSET
0000     91 ;
0000     92 ;                    Roger Riggs, 27-May-1980, Version 5.4
0000     93 ;         14         Changed entry points for QIO routines to use .VECTOR
0000     94 ;                    and JMP +2
0000     95 ;
0000     96 ;                    Dave Butenhof   9-jun-1980, Version 5.4
0000     97 ;         15         Added entry vectors for VMS V2.0 driver support routines:
0000     98 ;                    IOC$LOADUBAMAPA, IOC$GETBUTE, IOC$PUTBYTE, IOC$INITBUFWIND,
0000     99 ;                    IOC$MOVFRUSER2, IOC$MOVFRUSER1, IOC$MOVTOUSER2, IOC$MOVTOUSER1.
0000    100 ;
0000    101 ;                    Dave Butenhof   18-jun-1980,    version 5.5
```

```
                        0000    102 ;       16      Add driver entry vectors for SCH$LOCKW and SCH$UNLOCK, for
                        0000    103 ;               use by drivers doing buffered I/O.
                        0000    104 ;
                        0000    105 ;               Roger Riggs, 22-July-1980, Version 5.5
                        0000    106 ;       17      Added entry points for ???
                        0000    107 ;               Moved RMS$GL_ABSTIM, RMS$GQ_SYSTIME here from clock.
                        0000    108 ;
                        0000    109 ;               Roger Riggs, 27-Aug-1980
                        0000    110 ;       18      Added SYS$GET, SYS$READ, SYS$OPEN, SYS$CLOSE, SYS$CONNECT,
                        0000    111 ;               SYS$DISCONNECT for subset of RMS services.
```

```
0000    113 ;
0000    114 ;           Dave Butenhof, 30-sep-1980, Version 6.1
0000    115 ;       19  Added entry point for DS$ATTACH--user-callable ATTACH
0000    116 ;           command processing
0000    117 ;       20  Added entry vectors for COM$DRVDEALMEM, EXE$INSIOQ,
0000    118 ;           IOC$PTETOPFN, and moved EXE$GQ_SYSTIME here from CLOCK to
0000    119 ;           make it accessible to drivers.
0000    120 ;       21  Added entry point for DS$HELP--user-callable HELP command
0000    121 ;           processing.
0000    122 ;       22  Add entry point for IOC$MOVFRUSER
0000    123 ;
0000    124 ;           Dave Butenhof, 22-apr-1981, version 6.4
0000    125 ;       23  Add entry points for IOC$ALLOSPT, EXE$MAXACMODE, and
0000    126 ;           EXE$READLOCK.  These were specified in release notes
0000    127 ;           of version 6.3, but somehow the updates were lost.
0000    128 ;           [also, define all libraries in source]
0000    129 ;
0000    130 ;       24  - dave butenhof, 11-Sep-1981, version 6.5
0000    131 ;           Fix entries for event flag services.  They are all done
0000    132 ;           as jumps directly to label: fix to allow for entry mask
0000    133 ;           on routine.
0000    134 ;
0000    135 ;       25  - Jack Stansbury, 23-Oct-1981, Version 6.-
0000    136 ;           Fixed a truncation error in the CODE Psect, and
0000    137 ;           another possibly future one in the same Psect.
0000    138 ;
0000    139 ;       26  - Jack Stansbury, 23-Oct-1981, Version 6.-
0000    140 ;           Added DS$BRANCH entry point, by putting it between
0000    141 ;           DS$SUMMARY and DS$BGNSUB, and by taking out the blank
0000    142 ;           one after DS$ASKSTR.
0000    143 ;
0000    144 ;       27  - Jack Stansbury, 16-Nov-1981, Version 6.-
0000    145 ;           Changed the code at the DS$BREAK entry point. Put in a
0000    146 ;           JMP to the label FAKE_JUMP, which is in the routine
0000    147 ;           RESRVD_ENTRY. This was necessitated by the code at
0000    148 ;           DS$BREAK occupying too many bytes for the entry point.
0000    149 ;
0000    150 ;       28  Marion Baggett, 12-Apr-1982,    Version 6.7
0000    151 ;           Added entry point DSX$ERRPREP
0000    152 ;
0000    153 ;       29  Richard Brown, 27-May-82, Version 6.8
0000    154 ;           Added entry point DS$SETPRIEXV.
0000    155 ;
0000    156 ;       30  Richard Brown, 24-June-82, Version 6.8
0000    157 ;           Added entry point DS$MAPDBGBLOCK.
0000    158 ;
0000    159 ;       31  Richard Brown, 4-August-82, Version 6.9
0000    160 ;           Added entry point DS$FREEDBGSYM.
0000    161 ;
0000    162 ;       32  Bob Bergazzi    23-Nov-82       Version 6.10
0000    163 ;           Added entry point DS$LOADPCS.
0000    164 ;
0000    165 ;       33  Bob Bergazzi    May 16,1983     Version 6.11
0000    166 ;           Changed the order of the .LIB statements.
0000    167 ;
0000    168 ;       34  Richard Brown   22-June-83      Version 6.12
0000    169 ;           Added entry point DS$GETTERM.
```

```
                        0000    170 ;
                        0000    171 ;       35      Richard Brown   12-July-84      Version 7.0
                        0000    172 ;               Added entry point DS$SERVICE_DISPATCHER, to handle
                        0000    173 ;               all future additional services, since empty spots
                        0000    174 ;               in the dispatch vector area are in short supply.
                        0000    175 ;               Added DS$SRVDISP_TABLE, used by DS$SERVICE_DISPATCHER.
                        0000    176 ;               Added DSX$MOUNT and DSX$DISMOUNT to DS$SRVDISP_TABLE.
                        0000    177 ;               Changed DATA psect alignment from Byte to Quad.
                        0000    178 ;
                        0000    179 ;       36      Jim Shelhamer   6-Aug-1984      Version 7.1
                        0000    180 ;               Added entry point DS$MEMSIZE, to return the physical
                        0000    181 ;               memory size.
                        0000    182 ;
                        0000    183 ;       37      Richard Brown   20-Sep-1984     Version 7.1
                        0000    184 ;               Removed DSX$MOUNT and DSX$DISMOUNT.
                        0000    185 ;
                        0000    186 ;       38      Domenic Andella 30-Nov-1984     Version 8.0
                        0000    187 ;               Added new VDS multiprocessing service dispatch
                        0000    188 ;               vectors to DS$SRVDISP_TABLE.
                        0000    189 ;
                        0000    190 ;       39      Domenic Andella 29-Jan-1985     Version 8.1
                        0000    191 ;               Added new service dispatch vectors for
                        0000    192 ;               $DS_GETSYSBUF and $DS_RELSYSBUF
                        0000    193 ;
                        0000    194 ;       40      Amy Iorio       15-Feb-1985     Version 8.1
                        0000    195 ;               Added new service dispath vector for
                        0000    196 ;               DS$_PRINTREV.
                        0000    197 ;
                        0000    198 ;       41      Domenic Andella 26-Feb-85       Version 8.1
                        0000    199 ;               Establish WEAK Externals for DSX$RELSYSBUF, and
                        0000    200 ;               DSX$GETSYSBUF pending replacement of BUFFER module.
                        0000    201 ;               Replace .vectors with .word for each.
                        0000    202 ;
                        0000    203 ;       42      Ted Salem       16-Dec-85       Version 9.2
                        0000    204 ;               Establish WEAK Externals for DSX$interlock_test, and
                        0000    205 ;               DSX$set and clear interlock. Added new service
                        0000    206 ;               dispatch vectors for each.
                        0000    207 ;
                        0000    208 ;       43      Jim Shelhamer   24-Apr-86       Version 10.0                    ;G:2
                        0000    209 ;               Add new service, DS$PPSCB, which returns the base               ;G:2
                        0000    210 ;               address of the PP's SCB on 8200 AP systems.                     ;G:2
                        0000    211 ;--
```

```
                        0000    213          .SBTTL  Libraries and Equated Symbols
                        0000    214 ;
                        0000    215 ; INCLUDE FILES:
                        0000    216 ;
                        0000    217          .LIBRARY        "SYS$LIBRARY:LIB"       ;                       [33]
                        0000    218          .LIBRARY        "$DS"                   ;                       [33]
                        0000    219          .LIBRARY        "$DIAG"                 ;                       [33]
                        0000    220
                        0000    221 ;
                        0000    222 ; MACROS:
                        0000    223 ;
                        0000    224
                        0000    225 ;
                        0000    226 ; Weak Externals
                        0000    227 ;
                        0000    228
                        0000    229          .WEAK   DSX$GETSYSBUF, DSX$RELSYSBUF
                        0000    230          .WEAK   DSX$_INTERLOCK_TEST     ; Entry for the Interlock test  [42]
                        0000    231          .WEAK   DSX$_SET_INTERLOCK      ; Entry for misc. interlocking  [42]
                        0000    232          .WEAK   DSX$_CLEAR_INTERLOCK    ; Entry for misc. interlocking  [42]
                        0000    233
                        0000    234
                        0000    235 ;
                        0000    236 ; EQUATED SYMBOLS:
                        0000    237 ;
0000000D                0000    238          CR      = 13
0000000A                0000    239          LF      = 10
```

```
                                    0000   241          .SUBTITLE       Data Psect Declarations
                                    0000   242
                                    0000   243 ;
                                    0000   244 ; OWN STORAGE:
                                    0000   245 ;
                                00000000   246          .PSECT  DATA, SHR, NOEXE, NOWRT, quad                    ; [35]
                                    0000   247
                                    0000   248          MODNAM  ENTRY
                  59 52 54 4E 45 00'0000       $MODULE:        .ASCIC  \ENTRY\
                                 05 0000
                                    0006   249
                                    0006   250 T_RESRVD:
56 52 45 53 45 52 20 3F 3F 0A 0D 00'0006   251          .ASCIC  <CR><LF>\?? RESERVED DIAGNOSTIC/SYSTEM SERVICE CALL.\
49 54 53 4F 4E 47 41 49 44 20 44 45 0012
52 45 53 20 4D 45 54 53 59 53 2F 43 001E
      2E 4C 4C 41 43 20 45 43 49 56 002A
                                 2D 0006
```

```
                     253          .SBTTL  DS Entry Points
                     254 ;++
                     255 ; FUNCTIONAL DESCRIPTION:
                     256 ;
                     257 ;       THIS SECTION PROVIDES A WELL DEFINED LINK BETWEEN THE DIAGNOSTIC
                     258 ;       SUPERVISOR AND THE DIAGNOSTIC PROGRAM.  EACH ROUTINE STARTS ON A
                     259 ;       FIXED QUADWORD BOUNDRY WHICH CONTAINS THE APPROPRIATE LINK TO
                     260 ;       THE BODY OF THE ROUTINE.  THERE ARE FOUR SETS OF LINK VECTORS,
                     261 ;       ONE FOR EACH TYPE OF PROGRAM:  CEP FUNCTIONAL, CEP REPAIR,
                     262 ;       SEP FUNCTIONAL, SEP REPAIR.
                     263 ;
                     264 ; CALLING SEQUENCE:
                     265 ;
                     266 ;       ALL ROUTINES ARE CALLED VIA THE STANDARD VAX CALL PROCEDURE,
                     267 ;       I.E., CALLG AND CALLS INSTRUCTIONS.
                     268 ;
                     269 ; INPUT PARAMETERS:
                     270 ;
                     271 ;       SEE SPECIFIC ROUTINE.
                     272 ;
                     273 ; IMPLICIT INPUTS:
                     274 ;
                     275 ;       SEE SPECIFIC ROUTINE.
                     276 ;
                     277 ; OUTPUT PARAMETERS:
                     278 ;
                     279 ;       SEE SPECIFIC ROUTINE.
                     280 ;
                     281 ; IMPLICIT OUTPUTS:
                     282 ;
                     283 ;       SEE SPECIFIC ROUTINE.
                     284 ;
                     285 ; COMPLETION CODES:
                     286 ;
                     287 ;       SEE SPECIFIC ROUTINE.
                     288 ;
                     289 ; SIDE EFFECTS:
                     290 ;
                     291 ;       SEE SPECIFIC ROUTINE.
                     292 ;
                     293 ;--
```

```
                  00000000   295          .PSECT  $BASE, SHR, NOEXE, NOWRT, QUAD
                      0000   296
                      0000   297 DS$ENTRY:                                  ; Address 10000 (X)
          01E9   31   0000   298          BRW     LEAP_BOOT                 ; STAND ALONE ENTRY POINT.
                      0003   299
                      0003   300          .ALIGN  LONG
                      0004   301 DS$APT_ENTRY:
          01ED   31   0004   302          BRW     LEAP_APT                  ; APT ENTRY POINT
                      0007   303
                      0007   304          .ALIGN  QUAD
                      0008   305 ; RESERVED ENTRY POINT
                 0000 0008   306          .WORD   ^M<>                      ; ENTRY MASK.
     0000001D'EF   17 000A   307          JMP     RESRVD_ENTRY
```

```
                     0010   309 ;+
                     0010   310 ; 4.1.11        INITIALIZE CODE SERVICES.
                     0010   311 ;-
                     0010   312         .ALIGN  QUAD
                     0010   313 DS$ENDPASS::                          ; END OF PASS ENTRY POINT.
               0000' 0010   314         .VECTOR DSX$ENDPASS
00000002'EF    17    0012   315         JMP     DSX$ENDPASS+2         ;
                     0018   316
                     0018   317         .ALIGN  QUAD
                     0018   318 DS$GPHARD::                           ; GET POINTER TO HARDWARE PARAMETER TABLE.
               0000' 0018   319         .VECTOR DSX$GPHARD
00000002'EF    17    001A   320         JMP     DSX$GPHARD+2
                     0020   321
                     0020   322 ;+
                     0020   323 ; 4.1.12        CLEANUP CODE CALL.
                     0020   324 ;-
                     0020   325         .ALIGN  QUAD
                     0020   326 DS$ABORT::                            ; ABORT PROGRAM.
               0000' 0020   327         .VECTOR DSX$ABORT
00000002'EF    17    0022   328         JMP     DSX$ABORT+2
                     0028   329
                     0028   330 ;+
                     0028   331 ; 4.1.13        SUMMARY REPORT CALL.
                     0028   332 ;-
                     0028   333         .ALIGN  QUAD
                     0028   334 DS$SUMMARY::
                     0028   335 DS$DOSUMMARY::                        ; EXECUTE SUMMARY REPORT ROUTINE.
               0000' 0028   336         .VECTOR DSX$DOSUMMARY
00000002'EF    17    002A   337         JMP     DSX$DOSUMMARY+2
```

```
                     0030   339 ;+
                     0030   340 ; 4.2.1          PROGRAM CONTROL SERVICES.
                     0030   341 ;-
                     0030   342          .ALIGN  QUAD
                     0030   343 DS$BGNSUB::                          ; BEGIN SUBTEST ENTRY POINT.
                0000' 0030   344          .VECTOR DSX$BGNSUB
    00000002'EF   17 0032   345          JMP     DSX$BGNSUB+2
                     0038   346
                     0038   347          .ALIGN  QUAD
                     0038   348 DS$ENDSUB::                          ; END SUBTEST ENTRY POINT.
                0000' 0038   349          .VECTOR DSX$ENDSUB
    00000002'EF   17 003A   350          JMP     DSX$ENDSUB+2
                     0040   351
                     0040   352          .ALIGN  QUAD
                     0040   353 DS$CKLOOP::                          ; CHECK LOOP ENRTY POINT.
                0000' 0040   354          .VECTOR DSX$CKLOOP
    00000002'EF   17 0042   355          JMP     DSX$CKLOOP+2
                     0048   356
                     0048   357          .ALIGN  QUAD
                     0048   358 DS$INLOOP::                          ; IN LOOP ENTRY POINT.
                0000' 0048   359          .VECTOR DSX$INLOOP
    00000002'EF   17 004A   360          JMP     DSX$INLOOP+2
                     0050   361
                     0050   362          .ALIGN  QUAD
                     0050   363 DS$ESCAPE::                          ; ESCAPE ENTRY POINT.
                0000' 0050   364          .VECTOR DSX$ESCAPE
    00000002'EF   17 0052   365          JMP     DSX$ESCAPE+2
                     0058   366
                     0058   367          .ALIGN  QUAD
                     0058   368 DS$BREAK::                           ; BREAK FOR DYNAMIC SERVICES.
                0000 0058   369          .WORD   ^M<>                ; ENTRY MASK.
    00000032'EF   17 005A   370          JMP     FAKE_JUMP           ; Go to Fake_Jump, which JSB's to KB_CHECK,
                     0060   371                                      ; and then RETurns.
                     0060   372
                     0060   373          .ALIGN  QUAD
                     0060   374 DS$WAITMS::                          ; WAIT MILLISECONDS ENTRY POINT.
                0000' 0060   375          .VECTOR DSX$WAITMS
    00000002'EF   17 0062   376          JMP     DSX$WAITMS+2
                     0068   377
                     0068   378          .ALIGN  QUAD
                     0068   379 DS$WAITUS::                          ; WAIT MICROSECONDS ENTRY POINT.
                0000' 0068   380          .VECTOR DSX$WAITUS
    00000002'EF   17 006A   381          JMP     DSX$WAITUS+2
                     0070   382
                     0070   383          .ALIGN  QUAD
                     0070   384 DS$CANWAIT::
                     0070   385 DS$ABORTWAIT::                       ; CANCEL WAIT ENTRY POINT.
                0000' 0070   386          .VECTOR DSX$CANWAIT
    00000002'EF   17 0072   387          JMP     DSX$CANWAIT+2
                     0078   388
                     0078   389          .ALIGN  QUAD
                     0078   390 DS$CNTRLC::
                0000' 0078   391          .VECTOR DSX$CNTRLC
    00000002'EF   17 007A   392          JMP     DSX$CNTRLC+2
```

```
                     0080   394 ;+
                     0080   395 ; 4.2.2          MESSAGE HANDLER SERVICES.
                     0080   396 ;-
                     0080   397          .ALIGN  QUAD
                     0080   398 DS$ASKDATA::
                     0080   399 DS$GETDATA::                          ; GET DATA ENTRY POINT.
              0000'  0080   400          .VECTOR DSX$GETDATA
00000002'EF    17    0082   401          JMP     DSX$GETDATA+2
                     0088   402
                     0088   403          .ALIGN  QUAD
                     0088   404 DS$ASKVLD::
                     0088   405 DS$GETVIELD::                         ; GET A VIELD ENTRY POINT.
              0000'  0088   406          .VECTOR DSX$GETVIELD
00000002'EF    17    008A   407          JMP     DSX$GETVIELD+2
                     0090   408
                     0090   409          .ALIGN  QUAD
                     0090   410 DS$ASKADR::
                     0090   411 DS$GETADDRESS::                       ; GET AN ADDRESS ENTRY POINT.
              0000'  0090   412          .VECTOR DSX$GETADDRESS
00000002'EF    17    0092   413          JMP     DSX$GETADDRESS+2
                     0098   414
                     0098   415          .ALIGN  QUAD
                     0098   416 DS$ASKLGCL::
                     0098   417 DS$GETLOGICAL::                       ; GET A LOGICAL DECISION ENTRY POINT.
              0000'  0098   418          .VECTOR DSX$GETLOGICAL
00000002'EF    17    009A   419          JMP     DSX$GETLOGICAL+2
                     00A0   420
                     00A0   421          .ALIGN  QUAD
                     00A0   422 DS$ASKSTR::
                     00A0   423 DS$GETSTRING::                       ; GET AN ASCII STRING ENTRY POINT.
              0000'  00A0   424          .VECTOR DSX$GETSTRING
00000002'EF    17    00A2   425          JMP     DSX$GETSTRING+2
                     00A8   426
                     00A8   427
                     00A8   428          .ALIGN  QUAD                ;                                [26]
                     00A8   429 DS$BRANCH::                          ;                                [26]
              0000'  00A8   430          .VECTOR DSX$BRANCH          ; QA branch routine              [26]
00000002'EF    17    00AA   431          JMP     DSX$BRANCH+2        ;                                [26]
                     00B0   432                                      ;                                [26]
                     00B0   433                                      ;                                [26]
                     00B0   434
                     00B0   435          .ALIGN  QUAD
                     00B0   436 DS$CVTREG::                          ; CONVERT REGISTER BITS TO ASCII.
              0000'  00B0   437          .VECTOR DSX$CVTREG,^M<>
00000002'EF    17    00B2   438          JMP     DSX$CVTREG+2
                     00B8   439
                     00B8   440          .ALIGN  QUAD
                     00B8   441 DS$PARSE::                           ; PARSE AN ASCII STRING.
              0000'  00B8   442          .VECTOR DSX$PARSE
00000002'EF    17    00BA   443          JMP     DSX$PARSE+2
```

```
                        00C0   445          .ALIGN  QUAD
                        00C0   446 DS$ERRSYS::                          ; SYSTEM FATAL ERROR ENTRY POINT.
                0000'   00C0   447          .VECTOR DSX$ERRSYS
00000002'EF       17    00C2   448          JMP     DSX$ERRSYS+2
                        00C8   449
                        00C8   450          .ALIGN  QUAD
                        00C8   451 DS$ERRDEV::                          ; DEVICE FATAL ERROR ENTRY POINT.
                0000'   00C8   452          .VECTOR DSX$ERRDEV
00000002'EF       17    00CA   453          JMP     DSX$ERRDEV+2
                        00D0   454
                        00D0   455          .ALIGN  QUAD
                        00D0   456 DS$ERRHARD::                         ; HARD ERROR ENTRY POINT.
                0000'   00D0   457          .VECTOR DSX$ERRHARD
00000002'EF       17    00D2   458          JMP     DSX$ERRHARD+2
                        00D8   459
                        00D8   460          .ALIGN  QUAD
                        00D8   461 DS$ERRSOFT::                         ; SOFT ERROR ENTRY POINT.
                0000'   00D8   462          .VECTOR DSX$ERRSOFT
00000002'EF       17    00DA   463          JMP     DSX$ERRSOFT+2
                        00E0   464
                        00E0   465          .ALIGN  QUAD
                        00E0   466 DS$PRINTB::                          ; PRINT BASIC MESSAGE ENTRY POINT.
                0000'   00E0   467          .VECTOR DSX$PRINTB,^M<>
00000002'EF       17    00E2   468          JMP     DSX$PRINTB+2
                        00E8   469
                        00E8   470          .ALIGN  QUAD
                        00E8   471 DS$PRINTX::                          ; PRINT EXTENDED MESSAGE ENTRY POINT.
                0000'   00E8   472          .VECTOR DSX$PRINTX,^M<>
00000002'EF       17    00EA   473          JMP     DSX$PRINTX+2
                        00F0   474
                        00F0   475          .ALIGN  QUAD
                        00F0   476 DS$PRINTF::                          ; PRINT FORCED MESSAGE ENTRY POINT.
                0000'   00F0   477          .VECTOR DSX$PRINTF,^M<>
00000002'EF       17    00F2   478          JMP     DSX$PRINTF+2
                        00F8   479
                        00F8   480          .ALIGN  QUAD
                        00F8   481 DS$PRINTS::                          ; PRINT STATISTICAL MESSAGE ENTRY POINT.
                0000'   00F8   482          .VECTOR DSX$PRINTS,^M<>
00000002'EF       17    00FA   483          JMP     DSX$PRINTS+2
```

```
                          0100   485          .ALIGN  QUAD
                          0100   486 DS$PRINTSIG::                          ; Print signal array routine
                    0000' 0100   487          .VECTOR DSX$PRINTSIG,^M<>
          00000002'EF   17  0102   488          JMP     DSX$PRINTSIG+2
                          0108   489
                          0108   490          .ALIGN  QUAD
                          0108   491 DS$ERRPREP::                           ; ERROR IN DEVICE PREPARATION.          [28]
                    0000' 0108   492          .VECTOR DSX$ERRPREP           ;                                       [28]
          00000002'EF   17  010A   493          JMP     DSX$ERRPREP+2         ;                                       [28]
                          0110   494
                          0110   495          .ALIGN  QUAD
                          0110   496 DS$SETPRIEXV::                          ; SET PRIMARY EXCEPTION VECTOR  [29]
                    0000' 0110   497          .VECTOR DSX$SETPRIEXV,^M<>    ;                               [29]
          00000002'EF   17  0112   498          JMP     DSX$SETPRIEXV+2       ;                               [29]
                          0118   499
                          0118   500          .ALIGN  QUAD
                          0118   501 DS$MAPDBGBLOCK::                        ; MAP BLOCK OF MEMORY FOR DEBUGGER[30]
                    0000' 0118   502          .VECTOR DSX$MAPDBGBLOCK,^M<>  ;                                       [30]
          00000002'EF   17  011A   503          JMP     DSX$MAPDBGBLOCK+2     ;                                       [30]
                          0120   504
```

```
                     0120   506 ;+
                     0120   507 ; 4.2.3          MEMORY MANAGEMENT SERVICES.
                     0120   508 ;-
                     0120   509          .ALIGN  QUAD
                     0120   510 DS$GETBUF::                          ; GET BUFFER ENTRY POINT.
              0000'  0120   511          .VECTOR DSX$GETBUF
00000002'EF     17   0122   512          JMP     DSX$GETBUF+2        ;
                     0128   513
                     0128   514          .ALIGN  QUAD
                     0128   515 DS$RELBUF::                          ; RELEASE BUFFER ENTRY POINT.
              0000'  0128   516          .VECTOR DSX$RELBUF
00000002'EF     17   012A   517          JMP     DSX$RELBUF+2        ;
                     0130   518
                     0130   519          .ALIGN  QUAD
                     0130   520 ; RESERVED ENTRY POINT                ; GET PHYSICAL MEMORY ENTRY POINT.
              0000   0130   521          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF     17   0132   522          JMP     RESRVD_ENTRY        ;
                     0138   523
                     0138   524          .ALIGN  QUAD
                     0138   525 ; RESERVED ENTRY POINT                ; RELEASE PHYSICAL MEMORY ENTRY POINT.
              0000   0138   526          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF     17   013A   527          JMP     RESRVD_ENTRY        ;
                     0140   528
                     0140   529          .ALIGN  QUAD
                     0140   530 ; RESERVED ENTRY POINT                ; RELOCATE PROGRAM ENTRY POINT.
              0000   0140   531          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF     17   0142   532          JMP     RESRVD_ENTRY        ;
                     0148   533
                     0148   534          .ALIGN  QUAD
                     0148   535 ; RESERVED ENTRY POINT                ; PHYSICAL RELOCATE ENTRY POINT.
              0000   0148   536          .WORD   ^M<>                ; ENTRY MASK
0000001D'EF     17   014A   537          JMP     RESRVD_ENTRY        ;
                     0150   538
                     0150   539          .ALIGN  QUAD
                     0150   540 DS$MMON::                            ; MEMORY MANAGEMENT ON ENTRY POINT.
              0000'  0150   541          .VECTOR DSX$MMON,^M<>
00000002'EF     17   0152   542          JMP     DSX$MMON+2          ;
                     0158   543
                     0158   544          .ALIGN  QUAD
                     0158   545 DS$MMOFF::                           ; MEMORY MANAGEMENT OFF ENTRY POINT.
              0000'  0158   546          .VECTOR DSX$MMOFF,^M<>
00000002'EF     17   015A   547          JMP     DSX$MMOFF+2         ;
```

```
                     0160   549 ;+
                     0160   550 ; 4.2.4          I/O SERVICES.
                     0160   551 ;-
                     0160   552         .ALIGN  QUAD
                     0160   553 DS$SETVEC::                          ; SET VECTOR ENTRY POINT.
               0000' 0160   554         .VECTOR DSX$SETVEC
00000002'EF     17   0162   555         JMP     DSX$SETVEC+2
                     0168   556
                     0168   557         .ALIGN  QUAD
                     0168   558 DS$CLRVEC::                          ; CLEAR VECTOR ENTRY POINT.
               0000' 0168   559         .VECTOR DSX$CLRVEC
00000002'EF     17   016A   560         JMP     DSX$CLRVEC+2
                     0170   561
                     0170   562         .ALIGN  QUAD
                     0170   563 DS$INITSCB::                         ; INITIALIZE SYSTEM CONTROL BLOCK ENTRY POIN
               0000' 0170   564         .VECTOR DSX$INITSCB
00000002'EF     17   0172   565         JMP     DSX$INITSCB+2
                     0178   566
                     0178   567         .ALIGN  QUAD
                     0178   568 DS$SETIPL::                          ; SET INTERRUPT PRIORITY LEVEL ENTRY POINT.
               0000' 0178   569         .VECTOR DSX$SETIPL
00000002'EF     17   017A   570         JMP     DSX$SETIPL+2
                     0180   571
                     0180   572         .ALIGN  QUAD
                     0180   573 DS$CHANNEL::                        ; CHANNEL SERVICES ENTRY POINT.
               0000' 0180   574         .VECTOR DSX$CHANNEL
00000002'EF     17   0182   575         JMP     DSX$CHANNEL+2
                     0188   576
                     0188   577         .ALIGN  QUAD
                     0188   578 DS$SETMAP::                         ; SET CHANNEL MAPPING ENTRY POINT.
               0000' 0188   579         .VECTOR DSX$SETMAP
00000002'EF     17   018A   580         JMP     DSX$SETMAP+2
                     0190   581
                     0190   582         .ALIGN  QUAD
                     0190   583 DS$SHOCHAN::
                     0190   584 DS$SHOWCHAN::                       ; SHOW CHANNEL STATUS ENTRY POINT.
               0000' 0190   585         .VECTOR DSX$SHOWCHAN
00000002'EF     17   0192   586         JMP     DSX$SHOWCHAN+2
                     0198   587
                     0198   588         .ALIGN  QUAD
                     0198   589 DS$LOAD::                          ; ^X10198
               0000' 0198   590         .VECTOR DSX$LOAD
00000002'EF     17   019A   591         JMP     DSX$LOAD+2
                     01A0   592
                     01A0   593         .ALIGN  QUAD
                     01A0   594 DS$PROBE::                         ; Probe accessibility
               0000' 01A0   595         .VECTOR DSX$PROBE
00000002'EF     17   01A2   596         JMP     DSX$PROBE+2
                     01A8   597
                     01A8   598         .ALIGN  QUAD
                     01A8   599 DS$ATTACH::                        ; Process ATTACH command line
               0000' 01A8   600         .VECTOR DSX$ATTACH
00000002'EF     17   01AA   601         JMP     DSX$ATTACH+2
                     01B0   602         .ALIGN  QUAD
                     01B0   603 DS$HELP::                          ; Process HELP command line
               0000' 01B0   604         .VECTOR DSX$HELP
00000002'EF     17   01B2   605         JMP     DSX$HELP+2
```

```
                    01B8   607          .ALIGN  QUAD
                    01B8   608 DS$FREEDBGSYM::                          ; FREE MEMORY ASSIGNED TO SYMBOL TABLES[31]
                0000' 01B8   609          .VECTOR DSX$FREEDBGSYM,^M<>    ;                                        [31]
    00000002'EF   17  01BA   610          JMP     DSX$FREEDBGSYM+2       ;                                        [31]
                    01C0   611
                    01C0   612          .ALIGN  QUAD
                    01C0   613 DS$LOADPCS::                             ; LOAD THE PCS INTO THE COMET            [32]
                0000' 01C0   614          .VECTOR DSX$LOADPCS,^M<>       ;                                        [32]
    00000002'EF   17  01C2   615          JMP     DSX$LOADPCS+2          ;                                        [32]
                    01C8   616
                    01C8   617          .ALIGN  QUAD
                    01C8   618 DS$GETTERM::                             ; Get terminal characteristics           [34]
                0000' 01C8   619          .VECTOR DSX$GETTERM,^M<>       ;                                        [34]
    00000002'EF   17  01CA   620          JMP     DSX$GETTERM+2          ;                                        [34]
                    01D0   621
                    01D0   622          .ALIGN  QUAD
                    01D0   623 DS$SERVICE_DISPATCHER::                  ;                                        [35]
                0000' 01D0   624          .VECTOR DSX$SERVICE_DISPATCHER ;                                        [35]
    00000002'EF   17  01D2   625          JMP     DSX$SERVICE_DISPATCHER+2; Common dispatcher for all             [35]
                    01D8   626                                          ; future additional services.            [35]
                    01D8   627          .ALIGN  QUAD
                    01D8   628 DS$MEMSIZE::                             ; Get memory size                        [36]
                0000' 01D8   629          .VECTOR DSX$MEMSIZE,^M<>       ;                                        [36]
    00000002'EF   17  01DA   630          JMP     DSX$MEMSIZE+2          ;                                        [36]
                    01E0   631
                    01E0   632
                    01E0   633          .ALIGN  QUAD
                    01E0   634 ; RESERVED ENTRY POINT.
                0000  01E0   635          .WORD   ^M<>                   ; ENTRY MASK.
    0000001D'EF   17  01E2   636          JMP     RESRVD_ENTRY
                    01E8   637
                    01E8   638          .ALIGN  QUAD
                    01E8   639 ; RESERVED ENTRY POINT.
                0000  01E8   640          .WORD   ^M<>                   ; ENTRY MASK.
              0E  11  01EA   641          BRB     RESRVD                 ; LEAPFROG TO RESERVED ENTRY
                    01EC   642 LEAP_BOOT:
    00000000'EF   17  01EC   643          JMP     BOOT                   ; LEAPFROG TO BOOT
                    01F2   644
                    01F2   645 ; RESERVED_ENTRY+2      (SAVE MASK IS HIGH ORDER BITS OF BOOT RELATIVE ADDRESS)
              06  11  01F2   646          BRB     RESRVD                 ; LEAPFROG TO RESERVED_ENTRY
                    01F4   647
                    01F4   648 LEAP_APT:
    00000000'EF   17  01F4   649          JMP     APT                    ; LEAPFROG TO APT ENTRY POINT
                    01FA   650 ; RESERVED_ENTRY+2      (SAVE MASK IS HIGH ORDER BITS OF APT RELATIVE ADDRESS)
    0000001D'EF   17  01FA   651 RESRVD: JMP     RESRVD_ENTRY
```

```
                        0200   653 ;
                        0200   654 ; STARLET SYSTEM SERVICE VECTORS.
                        0200   655 ;
                        0200   656          .ALIGN  QUAD
                        0200   657 DS$AQ_SYSSRV::
                        0200   658 SYS$QIOW::                                ; ^X80000000
                0000'   0200   659          .VECTOR EXE$QIO
00000002'9F     16      0202   660          JSB     @#EXE$QIO+2
        00000204        0208   661 EXE$QIO2=.-4                             ; FUDGE ADDRESS OF EXE$QIO+2, SEE SYS$QIO
      03 50     E9      0208   662          BLBC    R0, 10$                 ; SKIP WAIT IF QIO ERROR.
       026C'    31      020B   663          BRW     SYS$WAITFR+2            ; WAIT FOR EVENT FLAG.
                04      020E   664 10$:     RET
                        020F   665
                        020F   666          .ALIGN  QUAD
                        0210   667 SYS$CALL_HANDL::
 61   04 AE     FA      0210   668          CALLG   4(SP),(R1)              ; CALL EXCEPTION HANDLER
                05      0214   669          RSB
                        0215   670
                        0215   671          .ALIGN  QUAD
                        0218   672 ; RESERVED ENTRY POINT.
                0000    0218   673          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17      021A   674          JMP     RESRVD_ENTRY
                        0220   675
                        0220   676          .ALIGN  QUAD
                        0220   677 ; RESERVED ENTRY POINT.
                0000    0220   678          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17      0222   679          JMP     RESRVD_ENTRY
                        0228   680
                        0228   681          .ALIGN  QUAD
                        0228   682 ; RESERVED ENTRY POINT.
                0000    0228   683          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17      022A   684          JMP     RESRVD_ENTRY
                        0230   685
                        0230   686          .ALIGN  QUAD
                        0230   687 ; RESERVED ENTRY POINT.
                0000    0230   688          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF     17      0232   689          JMP     RESRVD_ENTRY
                        0238   690
                        0238   691          .ALIGN  QUAD
                        0238   692 SYS$ALLOC::                              ; ^X80000038
                0000'   0238   693          .VECTOR EXE$ALLOC
00000002'EF     17      023A   694          JMP     EXE$ALLOC+2
```

```
                        0240   696          .ALIGN  QUAD
                        0240   697 ; RESERVED ENTRY POINT.                    ; ^X80000040
                0000    0240   698          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    0242   699          JMP     RESRVD_ENTRY
                        0248   700
                        0248   701          .ALIGN  QUAD
                        0248   702 SYS$ASCTIM::                              ; ^X80000048
                007C    0248   703          .WORD   ^M<R2,R3,R4,R5,R6>       ; ENTRY MASK.
    00000002'EF   17    024A   704          JMP     EXE$ASCTIM+2
                        0250   705
                        0250   706          .ALIGN  QUAD
                        0250   707 SYS$ASSIGN::                              ; ^X80000050
                0000'   0250   708          .VECTOR EXE$ASSIGN
    00000002'EF   17    0252   709          JMP     EXE$ASSIGN+2
                        0258   710
                        0258   711          .ALIGN  QUAD
                        0258   712 SYS$BINTIM::                              ; ^X80000058
                01FC    0258   713          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8>; ENTRY MASK.
    00000002'EF   17    025A   714          JMP     EXE$BINTIM+2
                        0260   715
                        0260   716          .ALIGN  QUAD
                        0260   717 SYS$CANCEL::                              ; ^X80000060
                0000'   0260   718          .VECTOR EXE$CANCEL
    00000002'EF   17    0262   719          JMP     EXE$CANCEL+2
                        0268   720
                        0268   721          .ALIGN  QUAD
                        0268   722 SYS$CANTIM::                              ; ^X80000068
                0000'   0268   723          .VECTOR EXE$CANTIM
    00000002'EF   17    026A   724          JMP     EXE$CANTIM+2
                        0270   725
                        0270   726          .ALIGN  QUAD
                        0270   727 ; RESERVED ENTRY POINT.
                0000    0270   728          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    0272   729          JMP     RESRVD_ENTRY
                        0278   730
                        0278   731          .ALIGN  QUAD
                        0278   732 ; RESERVED ENTRY POINT.
                0000    0278   733          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17    027A   734          JMP     RESRVD_ENTRY
```

```
                        0280   736          .ALIGN  QUAD
                        0280   737 ; RESERVED ENTRY POINT.                    ; ^X80000080
                0000    0280   738          .WORD   ^M<>                     ; ENTRY MASK.
0000001D'EF      17     0282   739          JMP     RESRVD_ENTRY
                        0288   740
                        0288   741          .ALIGN  QUAD
                        0288   742 ; RESERVED ENTRY POINT.
                0000    0288   743          .WORD   ^M<>                     ; ENTRY MASK.
0000001D'EF      17     028A   744          JMP     RESRVD_ENTRY
                        0290   745
                        0290   746          .ALIGN  QUAD
                        0290   747 ; RESERVED ENTRY POINT.
                0000    0290   748          .WORD   ^M<>                     ; ENTRY MASK.
0000001D'EF      17     0292   749          JMP     RESRVD_ENTRY
                        0298   750
                        0298   751          .ALIGN  QUAD
                        0298   752 SYS$CLREF::                               ; ^X80000098
                0000'   0298   753          .vector exe$clref
00000002'EF      17     029A   754          jmp     exe$clref+2
                        02A0   755
                        02A0   756          .ALIGN  QUAD
                        02A0   757 SYS$CNTREG::                             ; ^X800000A0
                0000'   02A0   758          .VECTOR EXE$CNTREG,^M<>
00000002'EF      17     02A2   759          JMP     EXE$CNTREG+2
                        02A8   760
                        02A8   761          .ALIGN  QUAD
                        02A8   762 ; RESERVED ENTRY POINT.
                0000    02A8   763          .WORD   ^M<>                     ; ENTRY MASK.
0000001D'EF      17     02AA   764          JMP     RESRVD_ENTRY
                        02B0   765
                        02B0   766          .ALIGN  QUAD
                        02B0   767 ; RESERVED ENTRY POINT.
                0000    02B0   768          .WORD   ^M<>                     ; ENTRY MASK.
0000001D'EF      17     02B2   769          JMP     RESRVD_ENTRY
                        02B8   770
                        02B8   771          .ALIGN  QUAD
                        02B8   772 ; RESERVED ENTRY POINT.
                0000    02B8   773          .WORD   ^M<>                     ; ENTRY MASK.
0000001D'EF      17     02BA   774          JMP     RESRVD_ENTRY
```

```
                        02C0   776          .ALIGN  QUAD
                        02C0   777 ; RESERVED ENTRY POINT.                  ; ^X800000C0
                  0000  02C0   778          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  02C2   779          JMP     RESRVD_ENTRY
                        02C8   780
                        02C8   781          .ALIGN  QUAD
                        02C8   782 ; RESERVED ENTRY POINT
                  0000  02C8   783          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  02CA   784          JMP     RESRVD_ENTRY
                        02D0   785
                        02D0   786          .ALIGN  QUAD
                        02D0   787 ; RESERVED ENTRY POINT.
                  0000  02D0   788          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  02D2   789          JMP     RESRVD_ENTRY
                        02D8   790
                        02D8   791          .ALIGN  QUAD
                        02D8   792 SYS$DALLOC::                             ; ^X800000D8
                  0000' 02D8   793          .VECTOR EXE$DALLOC
      00000002'EF   17  02DA   794          JMP     EXE$CANCEL+2
                        02E0   795
                        02E0   796          .ALIGN  QUAD
                        02E0   797 SYS$DASSGN::                             ; ^X800000E0
                  0000' 02E0   798          .VECTOR EXE$DASSGN
      00000002'EF   17  02E2   799          JMP     EXE$DASSGN+2
                        02E8   800
                        02E8   801          .ALIGN  QUAD
                        02E8   802 ; RESERVED ENTRY POINT.
                  0000  02E8   803          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  02EA   804          JMP     RESRVD_ENTRY
                        02F0   805
                        02F0   806          .ALIGN  QUAD
                        02F0   807 ; RESERVED ENTRY POINT.
                  0000  02F0   808          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  02F2   809          JMP     RESRVD_ENTRY
                        02F8   810
                        02F8   811          .ALIGN  QUAD
                        02F8   812 ; RESERVED ENTRY POINT.
                  0000  02F8   813          .WORD   ^M<>                    ; ENTRY MASK.
      0000001D'EF   17  02FA   814          JMP     RESRVD_ENTRY
```

```
                        0300   816          .ALIGN  QUAD
                        0300   817 ; RESERVED ENTRY POINT.                    ; ^X80001000
                0000    0300   818          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0302   819          JMP     RESRVD_ENTRY
                        0308   820
                        0308   821          .ALIGN  QUAD
                        0308   822 ; RESERVED ENTRY POINT.
                0000    0308   823          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     030A   824          JMP     RESRVD_ENTRY
                        0310   825
                        0310   826          .ALIGN  QUAD
                        0310   827 ; RESERVED ENTRY POINT
                0000    0310   828          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0312   829          JMP     RESRVD_ENTRY
                        0318   830
                        0318   831          .ALIGN  QUAD
                        0318   832 ; RESERVED ENTRY POINT.
                0000    0318   833          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     031A   834          JMP     RESRVD_ENTRY
                        0320   835
                        0320   836          .ALIGN  QUAD
                        0320   837 ; RESERVED ENTRY POINT.
                0000    0320   838          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0322   839          JMP     RESRVD_ENTRY
                        0328   840
                        0328   841          .ALIGN  QUAD
                        0328   842 ; RESERVED ENTRY POINT.
                0000    0328   843          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     032A   844          JMP     RESRVD_ENTRY
                        0330   845
                        0330   846          .ALIGN  QUAD
                        0330   847 ; RESERVED ENTRY POINT.
                0000    0330   848          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0332   849          JMP     RESRVD_ENTRY
                        0338   850
                        0338   851          .ALIGN  QUAD
                        0338   852 ; RESERVED ENTRY POINT.
                0000    0338   853          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     033A   854          JMP     RESRVD_ENTRY
```

```
                        0340   856          .ALIGN  QUAD
                        0340   857 SYS$EXIT::                                ; ^X80000140
                   0000 0340   858          .WORD   ^M<>                     ; ENTRY MASK.
                     04 0342   859          RET
                        0343   860
                        0343   861          .ALIGN  QUAD
                        0348   862 SYS$EXPREG::                              ; ^X80000148
                   0000'0348   863          .VECTOR EXE$EXPREG,^M<>
          00000002'EF   17 034A   864          JMP     EXE$EXPREG+2
                        0350   865
                        0350   866          .ALIGN  QUAD
                        0350   867 SYS$FAO::                                 ; ^X80000150
                   0FFC 0350   868          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; ENTRY MASK.
          00000002'EF   17 0352   869          JMP     EXE$FAO+2
                        0358   870
                        0358   871          .ALIGN  QUAD
                        0358   872 SYS$FAOL::                                ; ^X80000158
                   0FFC 0358   873          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; ENTRY MASK.
          00000002'EF   17 035A   874          JMP     EXE$FAOL+2
                        0360   875
                        0360   876          .ALIGN  QUAD
                        0360   877 ; RESERVED ENTRY POINT.
                   0000 0360   878          .WORD   ^M<>                     ; ENTRY MASK.
          0000001D'EF   17 0362   879          JMP     RESRVD_ENTRY
                        0368   880
                        0368   881          .ALIGN  QUAD
                        0368   882 ; RESERVED ENTRY POINT.
                   0000 0368   883          .WORD   ^M<>                     ; ENTRY MASK.
          0000001D'EF   17 036A   884          JMP     RESRVD_ENTRY
                        0370   885
                        0370   886          .ALIGN  QUAD
                        0370   887 ; RESERVED ENTRY POINT.
                   0000 0370   888          .WORD   ^M<>                     ; ENTRY MASK.
          0000001D'EF   17 0372   889          JMP     RESRVD_ENTRY
                        0378   890
                        0378   891          .ALIGN  QUAD
                        0378   892 SYS$GETTIM::                              ; ^X80000178
                   0000 0378   893          .WORD   ^M<>                     ; ENTRY MASK.
          00000002'EF   17 037A   894          JMP     EXE$GETTIM+2
```

```
                        0380   896          .ALIGN  QUAD
                        0380   897 SYS$GTCHAN::                        ; ^X80000180
                  0000' 0380   898          .VECTOR EXE$GTCHAN
00000002'EF         17  0382   899          JMP     EXE$GTCHAN+2
                        0388   900
                        0388   901          .ALIGN  QUAD
                        0388   902 SYS$HIBER::                         ; ^X80000188
                  0000' 0388   903          .VECTOR EXE$HIBER
00000002'EF         17  038A   904          JMP     EXE$HIBER+2
                        0390   905
                        0390   906          .ALIGN  QUAD
                        0390   907 ; RESERVED ENTRY POINT.
                  0000  0390   908          .WORD   ^M<>                ; ENTRY MASK.
0000001D'EF         17  0392   909          JMP     RESRVD_ENTRY
                        0398   910
                        0398   911          .ALIGN  QUAD
                        0398   912 ; RESERVED ENTRY POINT.
                  0000  0398   913          .WORD   ^M<>                ; ENTRY MASK.
0000001D'EF         17  039A   914          JMP     RESRVD_ENTRY
                        03A0   915
                        03A0   916          .ALIGN  QUAD
                        03A0   917 SYS$LKWSET::                        ; ^X800001A0
                  0000  03A0   918          .WORD   0                   ; Entry mask
            50    01    D0   03A2   919          MOVL    #1,R0               ; Indicate success
                        04   03A5   920          RET                         ; Return
                        03A6   921
                        03A6   922          .ALIGN  QUAD
                        03A8   923 ; RESERVED ENTRY POINT.
                  0000  03A8   924          .WORD   ^M<>                ; ENTRY MASK.
0000001D'EF         17  03AA   925          JMP     RESRVD_ENTRY
                        03B0   926
                        03B0   927          .ALIGN  QUAD
                        03B0   928 ; RESERVED ENTRY POINT.
                  0000  03B0   929          .WORD   ^M<>                ; ENTRY MASK.
0000001D'EF         17  03B2   930          JMP     RESRVD_ENTRY
                        03B8   931
                        03B8   932          .ALIGN  QUAD
                        03B8   933 SYS$NUMTIM::                        ; ^X800001B8
                  00FC  03B8   934          .WORD   ^M<R2,R3,R4,R5,R6,R7>   ; ENTRY MASK.
00000002'EF         17  03BA   935          JMP     EXE$NUMTIM+2
```

```
                        03C0   937          .ALIGN  QUAD
                        03C0   938 ; RESERVED ENTRY POINT.                  ; ^X800001C0
                  0000  03C0   939          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03C2   940          JMP     RESRVD_ENTRY
                        03C8   941
                        03C8   942          .ALIGN  QUAD
                        03C8   943 SYS$QIO::                               ; ^X800001C8
                  0000' 03C8   944          .VECTOR EXE$QIO
          FE36 DF   16  03CA   945          JSB     @EXE$QIO2
                    04  03CE   946          RET
                        03CF   947
                        03CF   948          .ALIGN  QUAD
                        03D0   949 SYS$READEF::                            ; ^X800001D0
                  0000' 03D0   950          .vector exe$readef
      00000002'EF   17  03D2   951          jmp     exe$readef+2
                        03D8   952
                        03D8   953          .ALIGN  QUAD
                        03D8   954 ; RESERVED ENTRY POINT.
                  0000  03D8   955          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03DA   956          JMP     RESRVD_ENTRY
                        03E0   957
                        03E0   958          .ALIGN  QUAD
                        03E0   959 ; RESERVED ENTRY POINT.
                  0000  03E0   960          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03E2   961          JMP     RESRVD_ENTRY
                        03E8   962
                        03E8   963          .ALIGN  QUAD
                        03E8   964 ; RESERVED ENTRY POINT.
                  0000  03E8   965          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03EA   966          JMP     RESRVD_ENTRY
                        03F0   967
                        03F0   968          .ALIGN  QUAD
                        03F0   969 ; RESERVED ENTRY POINT.
                  0000  03F0   970          .WORD   ^M<>                     ; ENTRY MASK.
      0000001D'EF   17  03F2   971          JMP     RESRVD_ENTRY
                        03F8   972
                        03F8   973          .ALIGN  QUAD
                        03F8   974 SYS$SETAST::                            ; ^X800001F8
                  0000' 03F8   975          .VECTOR EXE$SETAST               ; Entry mask
      00000002'EF   17  03FA   976          JMP     EXE$SETAST+2
```

```
                        0400   978          .ALIGN  QUAD
                        0400   979 SYS$SETEF::                          ; ^X80000200
                0000'   0400   980          .vector exe$setef
    00000002'EF   17    0402   981          jmp     exe$setef+2
                        0408   982
                        0408   983          .ALIGN  QUAD
                        0408   984 ; RESERVED ENTRY POINT.
                0000    0408   985          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17    040A   986          JMP     RESRVD_ENTRY
                        0410   987
                        0410   988          .ALIGN  QUAD
                        0410   989 ; RESERVED ENTRY POINT.
                0000    0410   990          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17    0412   991          JMP     RESRVD_ENTRY
                        0418   992
                        0418   993          .ALIGN  QUAD
                        0418   994 ; RESERVED ENTRY POINT.
                0000    0418   995          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17    041A   996          JMP     RESRVD_ENTRY
                        0420   997
                        0420   998          .ALIGN  QUAD
                        0420   999 SYS$SETIMR::                         ; ^X80000220
                0000'   0420  1000          .VECTOR EXE$SETIMR
    00000002'EF   17    0422  1001          JMP     EXE$SETIMR+2
                        0428  1002
                        0428  1003          .ALIGN  QUAD
                        0428  1004 SYS$SETPRI::                         ; ^X80000228
                0000    0428  1005          .WORD   0                    ; Entry mask
        50  01    D0    042A  1006          MOVL    #1,R0                ; Indicate success
              04      042D  1007          RET                          ; Return
                      042E  1008
                      042E  1009          .ALIGN  QUAD
                      0430  1010 SYS$SETPRT::                          ; ^X80000230
                0000' 0430  1011          .VECTOR EXE$SETPRT,^M<>      ; ENTRY MASK.
    00000002'EF   17  0432  1012          JMP     EXE$SETPRT+2
                      0438  1013
                      0438  1014          .ALIGN  QUAD
                      0438  1015 SYS$SETRWM::                          ; ^X80000238
                0000  0438  1016          .WORD   0                    ; Entry mask
        50  01    D0  043A  1017          MOVL    #1,R0                ; Indicate success
              04    043D  1018          RET                          ; Return
```

```
                        043E          .ALIGN  QUAD
                        0440  ; RESERVED ENTRY POINT.                    ; ^X80000240
                  0000  0440          .WORD   ^M<>                       ; ENTRY MASK.
0000001D'EF         17  0442          JMP     RESRVD_ENTRY
                        0448
                        0448          .ALIGN  QUAD
                        0448  ; RESERVED ENTRY POINT.
                  0000  0448          .WORD   ^M<>                       ; ENTRY MASK.
0000001D'EF         17  044A          JMP     RESRVD_ENTRY
                        0450
                        0450          .ALIGN  QUAD
                        0450  ; RESERVED ENTRY POINT.
                  0000  0450          .WORD   ^M<>                       ; ENTRY MASK.
0000001D'EF         17  0452          JMP     RESRVD_ENTRY
                        0458
                        0458          .ALIGN  QUAD
                        0458  SYS$TRNLOG::                               ; ^X80000258
                  0000  0458          .WORD   ^M<>                       ; ENTRY MASK.
0000001D'EF         17  045A          JMP     RESRVD_ENTRY
                        0460
                        0460          .ALIGN  QUAD
                        0460  SYS$ULKPAG::                               ; ^X80000260
                  0000  0460          .WORD   0                          ; Entry mask
      50    01      D0  0462          MOVL    #1,R0                      ; Indicate success
                    04  0465          RET                                ; Return
                        0466
                        0466          .ALIGN  QUAD
                        0468  SYS$ULWSET::                               ; ^X80000268
                  0000  0468          .WORD   0                          ; Entry mask
      50    01      D0  046A          MOVL    #1,R0                      ; Indicate success
                    04  046D          RET                                ; Return
                        046E
                        046E          .ALIGN  QUAD
                        0470  SYS$UNWIND::
                  0000' 0470          .VECTOR EXE$UNWIND
00000002'EF         17  0472          JMP     EXE$UNWIND+2
                        0478
                        0478          .ALIGN  QUAD
                        0478  SYS$WAITFR::                               ; ^X80000278
                  0000' 0478          .vector exe$waitfr
00000002'EF         17  047A          jmp     exe$waitfr+2
```

```
                     0480  1062          .ALIGN  QUAD
                     0480  1063 SYS$WAKE::                          ; ^X80000280
                0000' 0480  1064          .VECTOR EXE$WAKE
    00000002'EF   17  0482  1065          JMP     EXE$WAKE+2
                     0488  1066
                     0488  1067          .ALIGN  QUAD
                     0488  1068 SYS$WFLAND::                        ; ^X80000288
                0000' 0488  1069          .vector exe$wfland
    00000002'EF   17  048A  1070          jmp     exe$wfland+2
                     0490  1071
                     0490  1072          .ALIGN  QUAD
                     0490  1073 SYS$WFLOR::                         ; ^X80000290
                0000' 0490  1074          .vector exe$wflor
    00000002'EF   17  0492  1075          jmp     exe$wflor+2
                     0498  1076
                     0498  1077          .ALIGN  QUAD
                     0498  1078 ; RESERVED ENTRY POINT.
                0000 0498  1079          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17  049A  1080          JMP     RESRVD_ENTRY
                     04A0  1081
                     04A0  1082          .ALIGN  QUAD
                     04A0  1083 ; RESERVED ENTRY POINT.
                0000 04A0  1084          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17  04A2  1085          JMP     RESRVD_ENTRY
                     04A8  1086
                     04A8  1087          .ALIGN  QUAD
                     04A8  1088 ; RESERVED ENTRY POINT.
                0000 04A8  1089          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17  04AA  1090          JMP     RESRVD_ENTRY
                     04B0  1091
                     04B0  1092          .ALIGN  QUAD
                     04B0  1093 ; RESERVED ENTRY POINT.
                0000 04B0  1094          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17  04B2  1095          JMP     RESRVD_ENTRY
                     04B8  1096
                     04B8  1097          .ALIGN  QUAD
                     04B8  1098 ; RESERVED ENTRY POINT.
                0000 04B8  1099          .WORD   ^M<>                 ; ENTRY MASK.
    0000001D'EF   17  04BA  1100          JMP     RESRVD_ENTRY
```

```
                        04C0  1102          .ALIGN  QUAD
                        04C0  1103 ; RESERVED ENTRY POINT.                  ; ^X800002C0
                0000    04C0  1104          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04C2  1105          JMP     RESRVD_ENTRY
                        04C8  1106
                        04C8  1107          .ALIGN  QUAD
                        04C8  1108 SYS$GETCHN::                             ; ^X800002C8
                0000'   04C8  1109          .VECTOR EXE$GTCHAN
    00000002'EF   17    04CA  1110          JMP     EXE$GTCHAN+2
                        04D0  1111
                        04D0  1112          .ALIGN  QUAD
                        04D0  1113 ; RESERVED ENTRY POINT.
                0000    04D0  1114          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04D2  1115          JMP     RESRVD_ENTRY
                        04D8  1116
                        04D8  1117          .ALIGN  QUAD
                        04D8  1118 ; RESERVED ENTRY POINT.
                0000    04D8  1119          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04DA  1120          JMP     RESRVD_ENTRY
                        04E0  1121
                        04E0  1122          .ALIGN  QUAD
                        04E0  1123 ; RESERVED ENTRY POINT.
                0000    04E0  1124          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04E2  1125          JMP     RESRVD_ENTRY
                        04E8  1126
                        04E8  1127          .ALIGN  QUAD
                        04E8  1128 ; RESERVED ENTRY POINT.
                0000    04E8  1129          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04EA  1130          JMP     RESRVD_ENTRY
                        04F0  1131
                        04F0  1132          .ALIGN  QUAD
                        04F0  1133 ; RESERVED ENTRY POINT.
                0000    04F0  1134          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04F2  1135          JMP     RESRVD_ENTRY
                        04F8  1136
                        04F8  1137          .ALIGN  QUAD
                        04F8  1138 ; RESERVED ENTRY POINT.
                0000    04F8  1139          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    04FA  1140          JMP     RESRVD_ENTRY
```

```
                     0500  1142          .ALIGN  QUAD
                     0500  1143 ; RESERVED ENTRY POINT.                    ; ^X80000300
                0000 0500  1144          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 0502  1145          JMP     RESRVD_ENTRY
                     0508  1146
                     0508  1147          .ALIGN  QUAD
                     0508  1148 ; RESERVED ENTRY POINT.
                0000 0508  1149          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 050A  1150          JMP     RESRVD_ENTRY
                     0510  1151
                     0510  1152          .ALIGN  QUAD
                     0510  1153 ; RESERVED ENTRY POINT.
                0000 0510  1154          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 0512  1155          JMP     RESRVD_ENTRY
                     0518  1156
                     0518  1157          .ALIGN  QUAD
                     0518  1158 ; RESERVED ENTRY POINT.
                0000 0518  1159          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 051A  1160          JMP     RESRVD_ENTRY
                     0520  1161
                     0520  1162          .ALIGN  QUAD
                     0520  1163 ; RESERVED ENTRY POINT.
                0000 0520  1164          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 0522  1165          JMP     RESRVD_ENTRY
                     0528  1166
                     0528  1167          .ALIGN  QUAD
                     0528  1168 ; RESERVED ENTRY POINT.
                0000 0528  1169          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 052A  1170          JMP     RESRVD_ENTRY
                     0530  1171
                     0530  1172          .ALIGN  QUAD
                     0530  1173 ; RESERVED ENTRY POINT.
                0000 0530  1174          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 0532  1175          JMP     RESRVD_ENTRY
                     0538  1176
                     0538  1177          .ALIGN  QUAD
                     0538  1178 ; RESERVED ENTRY POINT.
                0000 0538  1179          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17 053A  1180          JMP     RESRVD_ENTRY
```

```
                     0540  1182          .ALIGN  QUAD
                     0540  1183 ; RESERVED ENTRY POINT                    ; ^X80000340
                0000 0540  1184          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 0542  1185          JMP     RESRVD_ENTRY
                     0548  1186
                     0548  1187          .ALIGN  QUAD
                     0548  1188 ; RESERVED ENTRY POINT.
                0000 0548  1189          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 054A  1190          JMP     RESRVD_ENTRY
                     0550  1191
                     0550  1192          .ALIGN  QUAD
                     0550  1193 ; RESERVED ENTRY POINT.
                0000 0550  1194          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 0552  1195          JMP     RESRVD_ENTRY
                     0558  1196
                     0558  1197          .ALIGN  QUAD
                     0558  1198 ; RESERVED ENTRY POINT.
                0000 0558  1199          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 055A  1200          JMP     RESRVD_ENTRY
                     0560  1201
                     0560  1202          .ALIGN  QUAD
                     0560  1203 ; RESERVED ENTRY POINT.
                0000 0560  1204          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 0562  1205          JMP     RESRVD_ENTRY
                     0568  1206
                     0568  1207          .ALIGN  QUAD
                     0568  1208 ; RESERVED ENTRY POINT.
                0000 0568  1209          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 056A  1210          JMP     RESRVD_ENTRY
                     0570  1211
                     0570  1212          .ALIGN  QUAD
                     0570  1213 ; RESERVED ENTRY POINT.
                0000 0570  1214          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 0572  1215          JMP     RESRVD_ENTRY
                     0578  1216
                     0578  1217          .ALIGN  QUAD
                     0578  1218 ; RESERVED ENTRY POINT.
                0000 0578  1219          .WORD   ^M<>                     ; ENTRY MASK.
    0000001D'EF   17 057A  1220          JMP     RESRVD_ENTRY
```

```
                     0580  1222          .ALIGN  QUAD
                     0580  1223 SYS$GET::                                  ; ^X80000380
               0000' 0580  1224          .VECTOR RMS$GET
00000002'EF      17  0582  1225          JMP     RMS$GET+2
                     0588  1226
                     0588  1227          .ALIGN  QUAD
                     0588  1228 ; RESERVED ENTRY POINT.
               0000  0588  1229          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17  058A  1230          JMP     RESRVD_ENTRY
                     0590  1231
                     0590  1232          .ALIGN  QUAD
                     0590  1233 SYS$READ::
               0000' 0590  1234          .VECTOR RMS$READ
00000002'EF      17  0592  1235          JMP     RMS$READ+2                ; ^X80000390
                     0598  1236
                     0598  1237          .ALIGN  QUAD
                     0598  1238 ; RESERVED ENTRY POINT.
               0000  0598  1239          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17  059A  1240          JMP     RESRVD_ENTRY
                     05A0  1241
                     05A0  1242          .ALIGN  QUAD
                     05A0  1243 ; RESERVED ENTRY POINT.
               0000  05A0  1244          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17  05A2  1245          JMP     RESRVD_ENTRY
                     05A8  1246
                     05A8  1247          .ALIGN  QUAD
                     05A8  1248 ; RESERVED ENTRY POINT.
               0000  05A8  1249          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17  05AA  1250          JMP     RESRVD_ENTRY
                     05B0  1251
                     05B0  1252          .ALIGN  QUAD
                     05B0  1253 ; RESERVED ENTRY POINT.
               0000  05B0  1254          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17  05B2  1255          JMP     RESRVD_ENTRY
                     05B8  1256
                     05B8  1257          .ALIGN  QUAD
                     05B8  1258 SYS$CLOSE::
               0000' 05B8  1259          .VECTOR RMS$CLOSE
00000002'EF      17  05BA  1260          JMP     RMS$CLOSE+2
```

```
                        05C0  1262          .ALIGN  QUAD
                        05C0  1263 SYS$CONNECT::                            ; ^X800003C0
                0000'   05C0  1264          .VECTOR RMS$CONNECT
    00000002'EF   17    05C2  1265          JMP     RMS$CONNECT+2
                        05C8  1266
                        05C8  1267          .ALIGN  QUAD
                        05C8  1268 ; RESERVED ENTRY POINT.
                0000    05C8  1269          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05CA  1270          JMP     RESRVD_ENTRY
                        05D0  1271
                        05D0  1272          .ALIGN  QUAD
                        05D0  1273 SYS$DISCONNECT::                                 ; ^X800003D0
                0000'   05D0  1274          .VECTOR RMS$DISCONNECT
    00000002'EF   17    05D2  1275          JMP     RMS$DISCONNECT+2
                        05D8  1276
                        05D8  1277          .ALIGN  QUAD
                        05D8  1278 ; RESERVED ENTRY POINT.
                0000    05D8  1279          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05DA  1280          JMP     RESRVD_ENTRY
                        05E0  1281
                        05E0  1282          .ALIGN  QUAD
                        05E0  1283 ; RESERVED ENTRY POINT.
                0000    05E0  1284          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05E2  1285          JMP     RESRVD_ENTRY
                        05E8  1286
                        05E8  1287          .ALIGN  QUAD
                        05E8  1288 ; RESERVED ENTRY POINT.
                0000    05E8  1289          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05EA  1290          JMP     RESRVD_ENTRY
                        05F0  1291
                        05F0  1292          .ALIGN  QUAD
                        05F0  1293 ; RESERVED ENTRY POINT.
                0000    05F0  1294          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05F2  1295          JMP     RESRVD_ENTRY
                        05F8  1296
                        05F8  1297          .ALIGN  QUAD
                        05F8  1298 ; RESERVED ENTRY POINT.
                0000    05F8  1299          .WORD   ^M<>                    ; ENTRY MASK.
    0000001D'EF   17    05FA  1300          JMP     RESRVD_ENTRY
```

```
                     0600  1302          .ALIGN  QUAD
                     0600  1303 ; RESERVED ENTRY POINT.                    ; ^X80000400
                0000 0600  1304          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0602  1305          JMP     RESRVD_ENTRY
                     0608  1306
                     0608  1307          .ALIGN  QUAD
                     0608  1308 SYS$OPEN::                                 ; ^X80000408
                0000'0608  1309          .VECTOR RMS$OPEN
    00000002'EF   17 060A  1310          JMP     RMS$OPEN+2
                     0610  1311
                     0610  1312
                     0610  1313          .ALIGN  QUAD
                     0610  1314 ; RESERVED ENTRY POINT.
                0000 0610  1315          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0612  1316          JMP     RESRVD_ENTRY
                     0618  1317
                     0618  1318          .ALIGN  QUAD
                     0618  1319 ; RESERVED ENTRY POINT.
                0000 0618  1320          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 061A  1321          JMP     RESRVD_ENTRY
                     0620  1322
                     0620  1323          .ALIGN  QUAD
                     0620  1324 ; RESERVED ENTRY POINT.
                0000 0620  1325          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0622  1326          JMP     RESRVD_ENTRY
                     0628  1327
                     0628  1328          .ALIGN  QUAD
                     0628  1329 ; RESERVED ENTRY POINT.
                0000 0628  1330          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 062A  1331          JMP     RESRVD_ENTRY
                     0630  1332
                     0630  1333          .ALIGN  QUAD
                     0630  1334 ; RESERVED ENTRY POINT.
                0000 0630  1335          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 0632  1336          JMP     RESRVD_ENTRY
                     0638  1337
                     0638  1338          .ALIGN  QUAD
                     0638  1339 ; RESERVED ENTRY POINT.
                0000 0638  1340          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17 063A  1341          JMP     RESRVD_ENTRY
```

```
                        0640  1343          .ALIGN  QUAD
                        0640  1344 ; RESERVED ENTRY POINT.                    ; ^X80000440
                0000    0640  1345          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0642  1346          JMP     RESRVD_ENTRY
                        0648  1347
                        0648  1348          .ALIGN  QUAD
                        0648  1349 ; RESERVED ENTRY POINT.
                0000    0648  1350          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     064A  1351          JMP     RESRVD_ENTRY
                        0650  1352
                        0650  1353          .ALIGN  QUAD
                        0650  1354 ; RESERVED ENTRY POINT.
                0000    0650  1355          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0652  1356          JMP     RESRVD_ENTRY
                        0658  1357
                        0658  1358          .ALIGN  QUAD
                        0658  1359 ; RESERVED ENTRY POINT.
                0000    0658  1360          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     065A  1361          JMP     RESRVD_ENTRY
                        0660  1362
                        0660  1363          .ALIGN  QUAD
                        0660  1364 ; RESERVED ENTRY POINT.
                0000    0660  1365          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0662  1366          JMP     RESRVD_ENTRY
                        0668  1367
                        0668  1368          .ALIGN  QUAD
                        0668  1369 ; RESERVED ENTRY POINT.
                0000    0668  1370          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     066A  1371          JMP     RESRVD_ENTRY
                        0670  1372
                        0670  1373          .ALIGN  QUAD
                        0670  1374 ; RESERVED ENTRY POINT.
                0000    0670  1375          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0672  1376          JMP     RESRVD_ENTRY
                        0678  1377
                        0678  1378          .ALIGN  QUAD
                        0678  1379 ; RESERVED ENTRY POINT.
                0000    0678  1380          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     067A  1381          JMP     RESRVD_ENTRY
```

```
                     0680  1383          .ALIGN  QUAD
                     0680  1384 ; RESERVED ENTRY POINT.                    ; ^X80000480
                0000 0680  1385          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  0682  1386          JMP     RESRVD_ENTRY
                     0688  1387
                     0688  1388          .ALIGN  QUAD
                     0688  1389 ; RESERVED ENTRY POINT.
                0000 0688  1390          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  068A  1391          JMP     RESRVD_ENTRY
                     0690  1392
                     0690  1393          .ALIGN  QUAD
                     0690  1394 ; RESERVED ENTRY POINT.
                0000 0690  1395          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  0692  1396          JMP     RESRVD_ENTRY
                     0698  1397
                     0698  1398          .ALIGN  QUAD
                     0698  1399 ; RESERVED ENTRY POINT.
                0000 0698  1400          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  069A  1401          JMP     RESRVD_ENTRY
                     06A0  1402
                     06A0  1403          .ALIGN  QUAD
                     06A0  1404 ; RESERVED ENTRY POINT.
                0000 06A0  1405          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  06A2  1406          JMP     RESRVD_ENTRY
                     06A8  1407
                     06A8  1408          .ALIGN  QUAD
                     06A8  1409 ; RESERVED ENTRY POINT.
                0000 06A8  1410          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  06AA  1411          JMP     RESRVD_ENTRY
                     06B0  1412
                     06B0  1413          .ALIGN  QUAD
                     06B0  1414 ; RESERVED ENTRY POINT.
                0000 06B0  1415          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  06B2  1416          JMP     RESRVD_ENTRY
                     06B8  1417
                     06B8  1418          .ALIGN  QUAD
                     06B8  1419 ; RESERVED ENTRY POINT.
                0000 06B8  1420          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17  06BA  1421          JMP     RESRVD_ENTRY
```

```
                        06C0  1423          .ALIGN  QUAD
                        06C0  1424 ; RESERVED ENTRY POINT.                  ; ^X800004C0
                0000    06C0  1425          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06C2  1426          JMP     RESRVD_ENTRY
                        06C8  1427
                        06C8  1428          .ALIGN  QUAD
                        06C8  1429 ; RESERVED ENTRY POINT.
                0000    06C8  1430          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06CA  1431          JMP     RESRVD_ENTRY
                        06D0  1432
                        06D0  1433          .ALIGN  QUAD
                        06D0  1434 ; RESERVED ENTRY POINT.
                0000    06D0  1435          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06D2  1436          JMP     RESRVD_ENTRY
                        06D8  1437
                        06D8  1438          .ALIGN  QUAD
                        06D8  1439 ; RESERVED ENTRY POINT.
                0000    06D8  1440          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06DA  1441          JMP     RESRVD_ENTRY
                        06E0  1442
                        06E0  1443          .ALIGN  QUAD
                        06E0  1444 ; RESERVED ENTRY POINT.
                0000    06E0  1445          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06E2  1446          JMP     RESRVD_ENTRY
                        06E8  1447
                        06E8  1448          .ALIGN  QUAD
                        06E8  1449 ; RESERVED ENTRY POINT.
                0000    06E8  1450          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06EA  1451          JMP     RESRVD_ENTRY
                        06F0  1452
                        06F0  1453          .ALIGN  QUAD
                        06F0  1454 ; RESERVED ENTRY POINT.
                0000    06F0  1455          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06F2  1456          JMP     RESRVD_ENTRY
                        06F8  1457
                        06F8  1458          .ALIGN  QUAD
                        06F8  1459 ; RESERVED ENTRY POINT.
                0000    06F8  1460          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     06FA  1461          JMP     RESRVD_ENTRY
```

```
                        0700  1463          .ALIGN  QUAD
                        0700  1464 ; RESERVED ENTRY POINT.                    ; ^X80000500
                0000    0700  1465          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0702  1466          JMP     RESRVD_ENTRY
                        0708  1467
                        0708  1468          .ALIGN  QUAD
                        0708  1469 ; RESERVED ENTRY POINT.
                0000    0708  1470          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    070A  1471          JMP     RESRVD_ENTRY
                        0710  1472
                        0710  1473          .ALIGN  QUAD
                        0710  1474 ; RESERVED ENTRY POINT.
                0000    0710  1475          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0712  1476          JMP     RESRVD_ENTRY
                        0718  1477
                        0718  1478          .ALIGN  QUAD
                        0718  1479 ; RESERVED ENTRY POINT.
                0000    0718  1480          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    071A  1481          JMP     RESRVD_ENTRY
                        0720  1482
                        0720  1483          .ALIGN  QUAD
                        0720  1484 ; RESERVED ENTRY POINT.
                0000    0720  1485          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0722  1486          JMP     RESRVD_ENTRY
                        0728  1487
                        0728  1488          .ALIGN  QUAD
                        0728  1489 ; RESERVED ENTRY POINT.
                0000    0728  1490          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    072A  1491          JMP     RESRVD_ENTRY
                        0730  1492
                        0730  1493          .ALIGN  QUAD
                        0730  1494 ; RESERVED ENTRY POINT.
                0000    0730  1495          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    0732  1496          JMP     RESRVD_ENTRY
                        0738  1497
                        0738  1498          .ALIGN  QUAD
                        0738  1499 ; RESERVED ENTRY POINT.
                0000    0738  1500          .WORD   ^M<>                      ; ENTRY MASK.
    0000001D'EF   17    073A  1501          JMP     RESRVD_ENTRY
```

```
                        0740  1503          .ALIGN  QUAD
                        0740  1504 ; RESERVED ENTRY POINT.                    ; ^X80000540
                0000    0740  1505          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0742  1506          JMP     RESRVD_ENTRY
                        0748  1507
                        0748  1508          .ALIGN  QUAD
                        0748  1509 ; RESERVED ENTRY POINT.
                0000    0748  1510          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     074A  1511          JMP     RESRVD_ENTRY
                        0750  1512
                        0750  1513          .ALIGN  QUAD
                        0750  1514 ; RESERVED ENTRY POINT.
                0000    0750  1515          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0752  1516          JMP     RESRVD_ENTRY
                        0758  1517
                        0758  1518          .ALIGN  QUAD
                        0758  1519 ; RESERVED ENTRY POINT.
                0000    0758  1520          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     075A  1521          JMP     RESRVD_ENTRY
                        0760  1522
                        0760  1523          .ALIGN  QUAD
                        0760  1524 ; RESERVED ENTRY POINT.
                0000    0760  1525          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0762  1526          JMP     RESRVD_ENTRY
                        0768  1527
                        0768  1528          .ALIGN  QUAD
                        0768  1529 ; RESERVED ENTRY POINT.
                0000    0768  1530          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     076A  1531          JMP     RESRVD_ENTRY
                        0770  1532
                        0770  1533          .ALIGN  QUAD
                        0770  1534 ; RESERVED ENTRY POINT.
                0000    0770  1535          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     0772  1536          JMP     RESRVD_ENTRY
                        0778  1537
                        0778  1538          .ALIGN  QUAD
                        0778  1539 ; RESERVED ENTRY POINT.
                0000    0778  1540          .WORD   ^M<>                      ; ENTRY MASK.
0000001D'EF      17     077A  1541          JMP     RESRVD_ENTRY
```

```
                                0780          .ALIGN  QUAD
                                0780  ; RESERVED ENTRY POINT.                    ; ^X80000580
                        0000    0780          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    0782          JMP     RESRVD_ENTRY
                                0788
                                0788          .ALIGN  QUAD
                                0788  ; RESERVED ENTRY POINT.
                        0000    0788          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    078A          JMP     RESRVD_ENTRY
                                0790
                                0790          .ALIGN  QUAD
                                0790  ; RESERVED ENTRY POINT.
                        0000    0790          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    0792          JMP     RESRVD_ENTRY
                                0798
                                0798          .ALIGN  QUAD
                                0798  ; RESERVED ENTRY POINT.
                        0000    0798          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    079A          JMP     RESRVD_ENTRY
                                07A0
                                07A0          .ALIGN  QUAD
                                07A0  ; RESERVED ENTRY POINT.
                        0000    07A0          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    07A2          JMP     RESRVD_ENTRY
                                07A8
                                07A8          .ALIGN  QUAD
                                07A8  ; RESERVED ENTRY POINT.
                        0000    07A8          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    07AA          JMP     RESRVD_ENTRY
                                07B0
                                07B0          .ALIGN  QUAD
                                07B0  ; RESERVED ENTRY POINT.
                        0000    07B0          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    07B2          JMP     RESRVD_ENTRY
                                07B8
                                07B8          .ALIGN  QUAD
                                07B8  ; RESERVED ENTRY POINT.
                        0000    07B8          .WORD   ^M<>                       ; ENTRY MASK.
        0000001D'EF       17    07BA          JMP     RESRVD_ENTRY
```

```
                        07C0  1583          .ALIGN  QUAD
                        07C0  1584 ; RESERVED ENTRY POINT.                  ; ^X80005C0
                0000    07C0  1585          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07C2  1586          JMP     RESRVD_ENTRY
                        07C8  1587
                        07C8  1588          .ALIGN  QUAD
                        07C8  1589 ; RESERVED ENTRY POINT.
                0000    07C8  1590          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07CA  1591          JMP     RESRVD_ENTRY
                        07D0  1592
                        07D0  1593          .ALIGN  QUAD
                        07D0  1594 ; RESERVED ENTRY POINT.
                0000    07D0  1595          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07D2  1596          JMP     RESRVD_ENTRY
                        07D8  1597
                        07D8  1598          .ALIGN  QUAD
                        07D8  1599 ; RESERVED ENTRY POINT.
                0000    07D8  1600          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07DA  1601          JMP     RESRVD_ENTRY
                        07E0  1602
                        07E0  1603          .ALIGN  QUAD
                        07E0  1604 ; RESERVED ENTRY POINT.
                0000    07E0  1605          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07E2  1606          JMP     RESRVD_ENTRY
                        07E8  1607
                        07E8  1608          .ALIGN  QUAD
                        07E8  1609 ; RESERVED ENTRY POINT.
                0000    07E8  1610          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07EA  1611          JMP     RESRVD_ENTRY
                        07F0  1612
                        07F0  1613          .ALIGN  QUAD
                        07F0  1614 ; RESERVED ENTRY POINT.
                0000    07F0  1615          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07F2  1616          JMP     RESRVD_ENTRY
                        07F8  1617
                        07F8  1618          .ALIGN  QUAD
                        07F8  1619 ; RESERVED ENTRY POINT.
                0000    07F8  1620          .WORD   ^M<>                    ; ENTRY MASK.
0000001D'EF      17     07FA  1621          JMP     RESRVD_ENTRY
                        0800  1622
                        0800  1623          .ALIGN  QUAD
            000007FF    0800  1624 DS$AQ_SSEND      == . - 1
            00000600    0800  1625 DS$K_SSSIZE      == . - DS$AQ_SYSSRV
```

```
                        0800  1627 ;++
                        0800  1628 ; FUNCTIONAL DESCRIPTION:
                        0800  1629 ;
                        0800  1630 ;       These entry points are fixed to allow VMS device drivers
                        0800  1631 ;       a static program interface to the common utility and
                        0800  1632 ;       support routines.  Note that the absolute addresses
                        0800  1633 ;       of these entry points are defined only in $QIOVEC_DEF in
                        0800  1634 ;       DS.MLB.
                        0800  1635 ;
                        0800  1636 ;--
                        0800  1637 SYS$GL_OPRMBX::
                00000000  0800  1638         .LONG   0                       ; ^X10800
                        0804  1639 MMG$GL_SPTBASE::
                00000000  0804  1640         .LONG   0                       ; ^X10804
                        0808  1641 MMG$GL_P0BASE::
                00000000' 0808  1642         .LONG   P0PT_BASE               ; ^X10808
                        080C  1643 SCH$GL_PCBVEC::
                00000000' 080C  1644         .ADDRESS        DS$GL_PCBVEC    ; Address of PCB vector
                00000000  0810  1645 EXE$GL_ABSTIM:: .LONG   0                       ; EXE$GL_ABSTIM, time in secs
                00000000  0814  1646                .LONG   0                       ; Unused
                00000818' 0818  1647 IOC$GL_PSFL::   .LONG   IOC$GL_PSFL
                00000818' 081C  1648 IOC$GL_PSBL::   .LONG   IOC$GL_PSFL
                        0820  1649         .ALIGN  QUAD
    00000000'9F   17    0820  1650         JMP     a#ACP$ACCESS
                        0826  1651         .ALIGN  QUAD
    00000000'9F   17    0828  1652         JMP     a#ACP$DEACCESS
                        082E  1653         .ALIGN  QUAD
    00000000'9F   17    0830  1654         JMP     a#ACP$MODIFY
                        0836  1655         .ALIGN  QUAD
    00000000'9F   17    0838  1656         JMP     a#ACP$MOUNT
                        083E  1657         .ALIGN  QUAD
    00000000'9F   17    0840  1658         JMP     a#ACP$READBLK
                        0846  1659         .ALIGN  QUAD
    00000000'9F   17    0848  1660         JMP     a#ACP$WRITEBLK
                        084E  1661         .ALIGN  QUAD
    00000000'9F   17    0850  1662         JMP     a#ERL$DEVICERR
                        0856  1663         .ALIGN  QUAD
    00000000'9F   17    0858  1664         JMP     a#ERL$DEVICTMO
                        085E  1665         .ALIGN  QUAD
    00000000'9F   17    0860  1666         JMP     a#EXE$ABORTIO
                        0866  1667         .ALIGN  QUAD
    00000000'9F   17    0868  1668         JMP     a#EXE$IOFORK
                        086E  1669         .ALIGN  QUAD
    00000000'9F   17    0870  1670         JMP     a#EXE$ONEPARM
                        0876  1671         .ALIGN  QUAD
    00000000'9F   17    0878  1672         JMP     a#EXE$PWRTIMCHK
                        087E  1673         .ALIGN  QUAD
    00000000'9F   17    0880  1674         JMP     a#EXE$SENSEMODE
                        0886  1675         .ALIGN  QUAD
    00000000'9F   17    0888  1676         JMP     a#EXE$SETCHAR
                        088E  1677         .ALIGN  QUAD
    00000000'9F   17    0890  1678         JMP     a#EXE$SETMODE
                        0896  1679         .ALIGN  QUAD
    00000000'9F   17    0898  1680         JMP     a#EXE$SNDEVMSG
                        089E  1681         .ALIGN  QUAD
    00000000'9F   17    08A0  1682         JMP     a#EXE$ZEROPARM
                        08A6  1683         .ALIGN  QUAD
```

```
00000000'9F   17  08A8  1684          JMP     a$IOC$APPLYECC
                  08AE  1685          .ALIGN  QUAD
00000000'9F   17  08B0  1686          JMP     a$IOC$CANCELIO
                  08B6  1687          .ALIGN  QUAD
00000000'9F   17  08B8  1688          JMP     a$IOC$DIAGBUFILL
                  08BE  1689          .ALIGN  QUAD
00000000'9F   17  08C0  1690          JMP     a$IOC$LOADMBAMAP
                  08C6  1691          .ALIGN  QUAD
00000000'9F   17  08C8  1692          JMP     a$IOC$LOADUBAMAP
                  08CE  1693          .ALIGN  QUAD
00000000'9F   17  08D0  1694          JMP     a$IOC$MOVTOUSER
                  08D6  1695          .ALIGN  QUAD
00000000'9F   17  08D8  1696          JMP     a$IOC$PURGDATAP
                  08DE  1697          .ALIGN  QUAD
00000000'9F   17  08E0  1698          JMP     a$IOC$RELCHAN
                  08E6  1699          .ALIGN  QUAD
00000000'9F   17  08E8  1700          JMP     a$IOC$RELDATAP
                  08EE  1701          .ALIGN  QUAD
00000000'9F   17  08F0  1702          JMP     a$IOC$RELMAPREG
                  08F6  1703          .ALIGN  QUAD
00000000'9F   17  08F8  1704          JMP     a$IOC$RELSCHAN
                  08FE  1705          .ALIGN  QUAD
00000000'9F   17  0900  1706          JMP     a$IOC$REQCOM
                  0906  1707          .ALIGN  QUAD
00000000'9F   17  0908  1708          JMP     a$IOC$REQDATAP
                  090E  1709          .ALIGN  QUAD
00000000'9F   17  0910  1710          JMP     a$IOC$REQMAPREG
                  0916  1711          .ALIGN  QUAD
00000000'9F   17  0918  1712          JMP     a$IOC$REQPCHANH
                  091E  1713          .ALIGN  QUAD
00000000'9F   17  0920  1714          JMP     a$IOC$REQPCHANL
                  0926  1715          .ALIGN  QUAD
00000000'9F   17  0928  1716          JMP     a$IOC$REQSCHANL
                  092E  1717          .ALIGN  QUAD
00000000'9F   17  0930  1718 IOC$RTN::JMP     a$IOC$RETURN          ; Defined external for QIO
                  0936  1719          .ALIGN  QUAD
00000000'9F   17  0938  1720          JMP     a$IOC$SENSEDISK
                  093E  1721          .ALIGN  QUAD
00000000'9F   17  0940  1722          JMP     a$IOC$UPDATRANSP
                  0946  1723          .ALIGN  QUAD
00000000'9F   17  0948  1724          JMP     a$IOC$WFIKPCH
                  094E  1725          .ALIGN  QUAD
00000000'9F   17  0950  1726          JMP     a$IOC$WFIRLCH
                  0956  1727          .ALIGN  QUAD
00000000'9F   17  0958  1728          JMP     a$MT$CHECK_ACCESS
                  095E  1729          .ALIGN  QUAD
00000000'9F   17  0960  1730          JMP     a$EXE$ALLOCIRP
                  0966  1731          .ALIGN  QUAD
00000000'9F   17  0968  1732          JMP     a$EXE$ALONONPAGED
                  096E  1733          .ALIGN  QUAD
00000000'9F   17  0970  1734          JMP     a$EXE$DEANONPAGED
                  0976  1735          .ALIGN  QUAD
00000000'9F   17  0978  1736          JMP     a$EXE$FINISHIOC
                  097E  1737          .ALIGN  QUAD
00000000'9F   17  0980  1738          JMP     a$EXE$FORK
                  0986  1739          .ALIGN  QUAD
00000000'9F   17  0988  1740          JMP     a$EXE$MODIFYLOCKR
```

```
                    098E  1741          .ALIGN  QUAD
00000000'9F   17    0990  1742          JMP     a$EXE$QIODRVPKT
                    0996  1743          .ALIGN  QUAD
00000000'9F   17    0998  1744          JMP     a$EXE$WRITELOCK
                    099E  1745          .ALIGN  QUAD
00000000'9F   17    09A0  1746          JMP     a$SCH$POSTEF
                    09A6  1747          .ALIGN  QUAD
00000000'9F   17    09A8  1748          JMP     a$SCH$QAST
                    09AE  1749          .ALIGN  QUAD
00000000'9F   17    09B0  1750          JMP     a$EXE$WRITECHK
                    09B6  1751          .ALIGN  QUAD
00000000'9F   17    09B8  1752          JMP     a$EXE$READLOCKR
                    09BE  1753          .ALIGN  QUAD
00000000'9F   17    09C0  1754          JMP     a$EXE$WRITELOCKR
                    09C6  1755          .ALIGN QUAD
00000000'9F   17    09C8  1756          JMP     a$IOC$INITIATE
                    09CE  1757          .ALIGN  QUAD
00000000'9F   17    09D0  1758          JMP     a$EXE$INSERTIRP
                    09D6  1759          .ALIGN QUAD
00000000'9F   17    09D8  1760          JMP     a$EXE$QIORETURN
                    09DE  1761          .ALIGN   QUAD
00000000'9F   17    09E0  1762          JMP     a$IOC$REQDATAPNW
                    09E6  1763          .ALIGN  QUAD
00000000'9F   17    09E8  1764          JMP     a$COM$POST
                    09EE  1765          .ALIGN  QUAD
00000000'9F   17    09F0  1766          JMP     a$IOC$ALOUBAMAP
                    09F6  1767          .ALIGN  QUAD
00000000'9F   17    09F8  1768          JMP     a$IOC$ALTUBAMAP
                    09FE  1769          .ALIGN  QUAD
00000000'9F   17    0A00  1770          JMP     a$ERL$RELEASEMB
                    0A06  1771          .ALIGN  QUAD
00000000'9F   17    0A08  1772          JMP     a$IOC$ALOUBAMAPN
                    0A0E  1773          .ALIGN  QUAD
00000000'9F   17    0A10  1774          JMP     a$EXE$ALLOCBUF
                    0A16  1775          .ALIGN  QUAD
00000000'9F   17    0A18  1776          JMP     a$EXE$BUFQUOPRC
                    0A1E  1777          .ALIGN  QUAD
00000000'9F   17    0A20  1778          JMP     a$EXE$BUFFRQUOTA
                    0A26  1779          .ALIGN  QUAD
00000000'9F   17    0A28  1780          JMP     a$EXE$FINISHIO
                    0A2E  1781          .ALIGN  QUAD
00000000'9F   17    0A30  1782          JMP     a$EXE$READ
                    0A36  1783          .ALIGN  QUAD
00000000'9F   17    0A38  1784          JMP     a$EXE$READCHK
                    0A3E  1785          .ALIGN  QUAD
00000000'9F   17    0A40  1786          JMP     a$EXE$READCHKR
                    0A46  1787          .ALIGN  QUAD
00000000'9F   17    0A48  1788          JMP     a$EXE$WRITE
                    0A4E  1789          .ALIGN  QUAD
00000000'9F   17    0A50  1790          JMP     a$EXE$WRITECHKR
                    0A56  1791          .ALIGN  QUAD
00000000'9F   17    0A58  1792          JMP     a$EXE$MODIFY
                    0A5E  1793          .ALIGN  QUAD
00000000'9F   17    0A60  1794          JMP     a$EXE$MODIFYLOCK
                    0A66  1795          .ALIGN  QUAD
00000000'9F   17    0A68  1796          JMP     a$EXE$CARRIAGE
                    0A6E  1797          .ALIGN  QUAD
```

```
                00000000'9F   17  0A70  1798          JMP     a#IOC$REQSCHANH
                                  0A76  1799          .ALIGN QUAD
                00000000'9F   17  0A78  1800          JMP     a#IOC$LOADUBAMAPA
                                  0A7E  1801          .ALIGN QUAD
                00000000'9F   17  0A80  1802          JMP     a#IOC$GETBYTE
                                  0A86  1803          .ALIGN QUAD
                00000000'9F   17  0A88  1804          JMP     a#IOC$PUTBYTE
                                  0A8E  1805          .ALIGN QUAD
                00000000'9F   17  0A90  1806          JMP     a#IOC$INITBUFWIND
                                  0A96  1807          .ALIGN QUAD
                00000000'9F   17  0A98  1808          JMP     a#IOC$MOVFRUSER2
                                  0A9E  1809          .ALIGN QUAD
                00000000'9F   17  0AA0  1810          JMP     a#IOC$MOVFRUSER1
                                  0AA6  1811          .ALIGN QUAD
                00000000'9F   17  0AA8  1812          JMP     a#IOC$MOVTOUSER2
                                  0AAE  1813          .ALIGN QUAD
                00000000'9F   17  0AB0  1814          JMP     a#IOC$MOVTOUSER1
                                  0AB6  1815          .ALIGN  QUAD
                00000000'9F   17  0AB8  1816          JMP     a#SCH$LOCKW                     ; Lock "MUTEX" for write access
                                  0ABE  1817          .ALIGN  QUAD
                00000000'9F   17  0AC0  1818          JMP     a#SCH$UNLOCK                    ; Unlock "MUTEX"
                                  0AC6  1819          .ALIGN  QUAD
                00000000'9F   17  0AC8  1820          JMP     a#EXE$ALTQUEPKT
                                  0ACE  1821          .ALIGN  QUAD
                00000000'9F   17  0AD0  1822          JMP     a#COM$SETATTNAST
                                  0AD6  1823          .ALIGN  QUAD
                00000000'9F   17  0AD8  1824          JMP     a#COM$DELATTNAST
                                  0ADE  1825          .ALIGN  QUAD
                00000000'9F   17  0AE0  1826          JMP     a#COM$FLUSHATTNS
                                  0AE6  1827          .ALIGN  QUAD
                00000000'9F   17  0AE8  1828          JMP     a#COM$DRVDEALMEM
                                  0AEE  1829          .ALIGN  QUAD
                00000000'9F   17  0AF0  1830          JMP     a#EXE$INSIOQ
                                  0AF6  1831          .ALIGN  QUAD
                00000000'9F   17  0AF8  1832          JMP     a#IOC$PTETOPFN
                                  0AFE  1833          .ALIGN  QUAD
                                  0B00  1834 EXE$GQ_SYSTIME::
            00000000 00000000     0B00  1835          .QUAD   0                       ; **DATA** System time quadword
                                  0B08  1836          .ALIGN  QUAD
                00000000'9F   17  0B08  1837          JMP     a#IOC$MOVFRUSER
                                  0B0E  1838          .ALIGN  QUAD
                00000000'9F   17  0B10  1839          JMP     a#IOC$ALLOSPT                       ;                       [23]
                                  0B16  1840          .ALIGN  QUAD                        ;                       [23]
                00000000'9F   17  0B18  1841          JMP     a#EXE$MAXACMODE             ;                       [23]
                                  0B1E  1842          .ALIGN  QUAD                        ;                       [23]
                00000000'9F   17  0B20  1843          JMP     a#EXE$READLOCK              ;                       [23]
                                  0B26  1844          .ALIGN  QUAD                        ;                       [23]
                                  0B28  1845
                                  0B28  1846
```

```
                     0B28  1848 .sbttl DS$SRVDISP_TABLE - Dispatch table for VDS services
                 00000034  1849 .psect DATA, SHR, NOEXE, NOWRT, QUAD
                     0034  1850
                     0034  1851 ; This is the dispatch table for all future additional VDS services.    [35]
                     0034  1852
                     0034  1853                    .ALIGN QUAD                 ;                       [35]
                     0038  1854 DS$SRVDISP_TABLE::                            ;                       [35]
                     0038  1855
                     0038  1856 ;[37]    .VECTOR DSX$MOUNT                   ;                       [35]
                     0038  1857 ;[37]    JMP     DSX$MOUNT+2                 ;                       [35]
                     0038  1858
                     0038  1859 ;[37]    .VECTOR DSX$DISMOUNT                ;                       [35]
                     0038  1860 ;[37]    JMP     DSX$DISMOUNT+2              ;                       [35]
                     0038  1861
                     0038  1862 ;                                                                      [38]
                     0038  1863 ; The following four dispatch vectors jump to module MPCOMMON.B32 to   [38]
                     0038  1864 ; perform multi-processing services                                    [38]
                     0038  1865 ;                                                                      [38]
                     0038  1866
              0000'  0038  1867          .VECTOR DSX$BOOTATTACHED            ;                       [38]
00000002'EF    17    003A  1868          JMP     DSX$BOOTATTACHED+2          ;                       [38]
                     0040  1869
              0000'  0040  1870          .VECTOR DSX$STARTATTACHED           ;                       [38]
00000002'EF    17    0042  1871          JMP     DSX$STARTATTACHED+2         ;                       [38]
                     0048  1872
              0000'  0048  1873          .VECTOR DSX$HALTATTACHED            ;                       [38]
00000002'EF    17    004A  1874          JMP     DSX$HALTATTACHED+2          ;                       [38]
                     0050  1875
              0000'  0050  1876          .VECTOR DSX$SHOWIDLE                ;                       [38]
00000002'EF    17    0052  1877          JMP     DSX$SHOWIDLE+2              ;                       [38]
                     0058  1878
                     0058  1879 ;
                     0058  1880 ; Add dispatch vectors for new VDS internal GETSYSBUF and RELSYSBUF     begin [39]
                     0058  1881 ; services
                     0058  1882 ;
                     0058  1883
              001C   0058  1884          .WORD   ^M<R2,R3,R4>                ; Entry mask            [41]
00000002'EF    17    005A  1885          JMP     DSX$GETSYSBUF+2
                     0060  1886
              000C   0060  1887          .WORD   ^M<R2,R3>                   ; Entry mask            [41]
00000002'EF    17    0062  1888          JMP     DSX$RELSYSBUF+2                                     ;end [39]
                     0068  1889
              0000'  0068  1890          .VECTOR DSX$PRINTREV                ; Vector for PRINTREV macro [40]
00000002'EF    17    006A  1891          JMP     DSX$PRINTREV+2
                     0070  1892
                     0070  1893          .ALIGN  QUAD
              0000'  0070  1894          .VECTOR DSX$_INTERLOCK_TEST         ; Entry for the Interlock test  [42]
00000002'EF    17    0072  1895          JMP     DSX$_INTERLOCK_TEST+2
                     0078  1896
                     0078  1897          .ALIGN  QUAD
              0000'  0078  1898          .VECTOR DSX$_SET_INTERLOCK          ; Entry for misc. interlocking  [42]
00000002'EF    17    007A  1899          JMP     DSX$_SET_INTERLOCK+2
                     0080  1900
                     0080  1901          .ALIGN  QUAD
              0000'  0080  1902          .VECTOR DSX$_CLEAR_INTERLOCK        ; Entry for misc. interlocking  [42]
00000002'EF    17    0082  1903          JMP     DSX$_CLEAR_INTERLOCK+2
                     0088  1904
```

```
                          0088  1905          .ALIGN  QUAD                  ;                          [43]  ;G:2
                     0000' 0088  1906          .VECTOR DSX$PPSCB             ;                          [43]  ;G:2
          00000002'EF   17  008A  1907          JMP     DSX$PPSCB+2           ;                          [43]  ;G:2
                          0090  1908                                                                           ;G:2
                          0090  1909 DS$SRVDISP_TABLE_END:                    ;                          [35]
                          0090  1910
```

```
                     0090  1912 .SBTTL DSX$SERVICE_DISPATCHER - Service Dispatch Routine
                 00000000  1913 .psect code, shr, exe, nowrt, byte
                     0000  1914
                     0000  1915 ;++                                                                    [35]
                     0000  1916 ; Functional Description:                                             [35]
                     0000  1917 ;                                                                     [35]
                     0000  1918 ;       Calls the proper service via the service dispatch table.      [35]
                     0000  1919 ;                                                                     [35]
                     0000  1920 ; Inputs:                                                            [35]
                     0000  1921 ;                                                                     [35]
                     0000  1922 ;       .................................                             [35]
                     0000  1923 ;       :       No. of args.            :<=AP                       [35]
                     0000  1924 ;       :.............................:                               [35]
                     0000  1925 ;       :Code indicating type of service:                           [35]
                     0000  1926 ;       :.............................:                               [35]
                     0000  1927 ;       :       Service routine arg1    :                           [35]
                     0000  1928 ;       :.............................:                               [35]
                     0000  1929 ;       /                             /                             [35]
                     0000  1930 ;       /                             /                             [35]
                     0000  1931 ;       :.............................:                               [35]
                     0000  1932 ;       :       Service routine argN    :                           [35]
                     0000  1933 ;       :.............................:                               [35]
                     0000  1934 ;                                                                     [35]
                     0000  1935 ;                                                                     [35]
                     0000  1936 ; Implicit Inputs:                                                   [35]
                     0000  1937 ;                                                                     [35]
                     0000  1938 ; Outputs:                                                           [35]
                     0000  1939 ;                                                                     [35]
                     0000  1940 ; Implicit Outputs:                                                  [35]
                     0000  1941 ;                                                                     [35]
                     0000  1942 ; Side Effects:                                                      [35]
                     0000  1943 ;                                                                     [35]
                     0000  1944 ; Return Status:                                                     [35]
                     0000  1945 ;                                                                     [35]
                     0000  1946 ;--                                                                  [35]
                     0000  1947
                0000 0000  1948 .ENTRY DSX$SERVICE_DISPATCHER, ^M<>     ;                           [35]
                     0002  1949
 7E   04 AC   08   C5 0002  1950         MULL3   #8,4(AP),-(SP)          ; Calc. offset into table.  [35]
7E   00000038'EF   9E 0007  1951         MOVAB   DS$SRVDISP_TABLE,-(SP)  ; Get dispatch table address. [35]
    04 AE   6E   C0 000E  1952         ADDL    (SP),4(SP)              ; Add offset.               [35]
 04 AC   6C   01   C3 0012  1953         SUBL3   #1,(AP),4(AP)           ; Create true arg. list for [35]
                     0017  1954                                         ;   service routine by getting [35]
                     0017  1955                                         ;   rid of service code arg. [35]
   04 BE   04 AC   FA 0017  1956         CALLG   4(AP),@4(SP)            ; Call the proper service.  [35]
                04 001C  1957         RET                             ; Return.                   [35]
                     001D  1958
```

```
                        001D  1960          .SBTTL  Resrvd_Entry Routine - For entry errors
                       0000001D  1961          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                        001D  1962 ;++
                        001D  1963 ; FUNCTIONAL DESCRIPTION:
                        001D  1964 ;
                        001D  1965 ;
                        001D  1966 ; CALLING SEQUENCE:
                        001D  1967 ;
                        001D  1968 ;       NONE
                        001D  1969 ;
                        001D  1970 ; INPUT PARAMETERS:
                        001D  1971 ;
                        001D  1972 ;       NONE
                        001D  1973 ;
                        001D  1974 ; IMPLICIT INPUTS:
                        001D  1975 ;
                        001D  1976 ;       NONE
                        001D  1977 ;
                        001D  1978 ; OUTPUT PARAMETERS:
                        001D  1979 ;
                        001D  1980 ;       NONE
                        001D  1981 ;
                        001D  1982 ; IMPLICIT OUTPUTS:
                        001D  1983 ;
                        001D  1984 ;       NONE
                        001D  1985 ;
                        001D  1986 ; COMPLETION CODES:
                        001D  1987 ;
                        001D  1988 ;       NONE
                        001D  1989 ;
                        001D  1990 ; SIDE EFFECTS:
                        001D  1991 ;
                        001D  1992 ;       NONE
                        001D  1993 ;
                        001D  1994 ;--
```

```
                                  001D  1996 RESRVD_ENTRY:
                                  001D  1997         ERRSUP_S        MSGADR=L^T_RESRVD       ;                                [25]
          00000000'EF    DF       001D                       PUSHAL  $MODULE
          00000006'EF    DF       0023                       PUSHAL  L^T_RESRVD
                    01   DD       0029                       PUSHL   #$ER
     00000000'EF    03   FB       002B                       CALLS   #3, DS_ERRSUP
                                  0032  1998
                                  0032  1999 FAKE_JUMP:                                      ;                                [27]
          00000000'EF    16       0032  2000         JSB     KB_CHECK                        ; Check keyboard status          [25]
                         04       0038  2001         RET
                                  0039  2002
                                  0039  2003         .END
```

```
$ER                             = 00000002     D
$MODULE                           00000000 R   D   01
ACP$ACCESS                        ********     X   02
ACP$DEACCESS                      ********     X   02
ACP$MODIFY                        ********     X   02
ACP$MOUNT                         ********     X   02
ACP$READBLK                       ********     X   02
ACP$WRITEBLK                      ********     X   02
APT                               ********     X   02
BOOT                              ********     X   02
COM$DELATTNAST                    ********     X   02
COM$DRVDEALMEM                    ********     X   02
COM$FLUSHATTNS                    ********     X   02
COM$POST                          ********     X   02
COM$SETATTNAST                    ********     X   02
CR                              = 0000000D     D
DS$ABORT                          00000020 RG  D   02
DS$ABORTWAIT                      00000070 RG  D   02
DS$APT_ENTRY                      00000004 R   D   02
DS$AQ_SSEND                     = 000007FF RG  D   02
DS$AQ_SYSSRV                      00000200 RG  D   02
DS$ASKADR                         00000090 RG  D   02
DS$ASKDATA                        00000080 RG  D   02
DS$ASKLGCL                        00000098 RG  D   02
DS$ASKSTR                         000000A0 RG  D   02
DS$ASKVLD                         00000088 RG  D   02
DS$ATTACH                         000001A8 RG  D   02
DS$BGNSUB                         00000030 RG  D   02
DS$BRANCH                         000000A8 RG  D   02
DS$BREAK                          00000058 RG  D   02
DS$CANWAIT                        00000070 RG  D   02
DS$CHANNEL                        00000180 RG  D   02
DS$CKLOOP                         00000040 RG  D   02
DS$CLRVEC                         00000168 RG  D   02
DS$CNTRLC                         00000078 RG  D   02
DS$CVTREG                         000000B0 RG  D   02
DS$DOSUMMARY                      00000028 RG  D   02
DS$ENDPASS                        00000010 RG  D   02
DS$ENDSUB                         00000038 RG  D   02
DS$ENTRY                          00000000 R   D   02
DS$ERRDEV                         000000C8 RG  D   02
DS$ERRHARD                        000000D0 RG  D   02
DS$ERRPREP                        00000108 RG  D   02
DS$ERRSOFT                        000000D8 RG  D   02
DS$ERRSYS                         000000C0 RG  D   02
DS$ESCAPE                         00000050 RG  D   02
DS$FREEDBGSYM                     000001B8 RG  D   02
DS$GETADDRESS                     00000090 RG  D   02
DS$GETBUF                         00000120 RG  D   02
DS$GETDATA                        00000080 RG  D   02
DS$GETLOGICAL                     00000098 RG  D   02
DS$GETSTRING                      000000A0 RG  D   02
DS$GETTERM                        000001C8 RG  D   02
DS$GETVIELD                       00000088 RG  D   02
DS$GL_PCBVEC                      ********       X   02
DS$GPHARD                         00000018 RG  D   02
DS$HELP                           000001B0 RG  D   02
DS$INITSCB                        00000170 RG  D   02
DS$INLOOP                         00000048 RG  D   02
DS$K_SSSIZE                     = 00000600   G  D
DS$LOAD                           00000198 RG  D   02
DS$LOADPCS                        000001C0 RG  D   02
DS$MAPDBGBLOCK                    00000118 RG  D   02
DS$MEMSIZE                        000001D8 RG  D   02
DS$MMOFF                          00000158 RG  D   02
DS$MMON                           00000150 RG  D   02
DS$PARSE                          000000B8 RG  D   02
DS$PRINTB                         000000E0 RG  D   02
DS$PRINTF                         000000F0 RG  D   02
DS$PRINTS                         000000F8 RG  D   02
DS$PRINTSIG                       00000100 RG  D   02
DS$PRINTX                         000000E8 RG  D   02
DS$PROBE                          000001A0 RG  D   02
DS$RELBUF                         00000128 RG  D   02
DS$SERVICE_DISPATCHER             000001D0 RG  D   02
DS$SETIPL                         00000178 RG  D   02
DS$SETMAP                         00000188 RG  D   02
DS$SETPRIEXV                      00000110 RG  D   02
DS$SETVEC                         00000160 RG  D   02
DS$SHOCHAN                        00000190 RG  D   02
DS$SHOWCHAN                       00000190 RG  D   02
DS$SRVDISP_TABLE                  00000038 RG  D   01
DS$SRVDISP_TABLE_END              00000090 R   D   01
DS$SUMMARY                        00000028 RG  D   02
DS$WAITMS                         00000060 RG  D   02
DS$WAITUS                         00000068 RG  D   02
DSX$ABORT                         ********       X   02
DSX$ATTACH                        ********       X   02
DSX$BGNSUB                        ********       X   02
DSX$BOOTATTACHED                  ********       X   01
DSX$BRANCH                        ********       X   02
DSX$CANWAIT                       ********       X   02
DSX$CHANNEL                       ********       X   02
DSX$CKLOOP                        ********       X   02
DSX$CLRVEC                        ********       X   02
DSX$CNTRLC                        ********       X   02
DSX$CVTREG                        ********       X   02
DSX$DOSUMMARY                     ********       X   02
DSX$ENDPASS                       ********       X   02
DSX$ENDSUB                        ********       X   02
DSX$ERRDEV                        ********       X   02
DSX$ERRHARD                       ********       X   02
DSX$ERRPREP                       ********       X   02
DSX$ERRSOFT                       ********       X   02
DSX$ERRSYS                        ********       X   02
DSX$ESCAPE                        ********       X   02
DSX$FREEDBGSYM                    ********       X   02
DSX$GETADDRESS                    ********       X   02
DSX$GETBUF                        ********       X   02
DSX$GETDATA                       ********       X   02
DSX$GETLOGICAL                    ********       X   02
DSX$GETSTRING                     ********       X   02
DSX$GETSYSBUF                     ********W GX   00
DSX$GETTERM                       ********       X   02
```

DSX$GETVIELD	********	X	02	EXE$DEANONPAGED	********	X	02
DSX$GPHARD	********	X	02	EXE$EXPREG	********	X	02
DSX$HALTATTACHED	********	X	01	EXE$FAO	********	X	02
DSX$HELP	********	X	02	EXE$FAOL	********	X	02
DSX$INITSCB	********	X	02	EXE$FINISHIO	********	X	02
DSX$INLOOP	********	X	02	EXE$FINISHIOC	********	X	02
DSX$LOAD	********	X	02	EXE$FORK	********	X	02
DSX$LOADPCS	********	X	02	EXE$GETTIM	********	X	02
DSX$MAPDBGBLOCK	********	X	02	EXE$GL_ABSTIM	00000810 RG	D	02
DSX$MEMSIZE	********	X	02	EXE$GQ_SYSTIME	00000B00 RG	D	02
DSX$MMOFF	********	X	02	EXE$GTCHAN	********	X	02
DSX$MMON	********	X	02	EXE$HIBER	********	X	02
DSX$PARSE	********	X	02	EXE$INSERTIRP	********	X	02
DSX$PPSCB	********	X	01	EXE$INSIOQ	********	X	02
DSX$PRINTB	********	X	02	EXE$IOFORK	********	X	02
DSX$PRINTF	********	X	02	EXE$MAXACMODE	********	X	02
DSX$PRINTREV	********	X	01	EXE$MODIFY	********	X	02
DSX$PRINTS	********	X	02	EXE$MODIFYLOCK	********	X	02
DSX$PRINTSIG	********	X	02	EXE$MODIFYLOCKR	********	X	02
DSX$PRINTX	********	X	02	EXE$NUMTIM	********	X	02
DSX$PROBE	********	X	02	EXE$ONEPARM	********	X	02
DSX$RELBUF	********	X	02	EXE$PWRTIMCHK	********	X	02
DSX$RELSYSBUF	********W	GX	00	EXE$QIO	********	X	02
DSX$SERVICE_DISPATCHER	00000000 RG	D	03	EXE$QIO2	= 00000204 R	D	02
DSX$SETIPL	********	X	02	EXE$QIODRVPKT	********	X	02
DSX$SETMAP	********	X	02	EXE$QIORETURN	********	X	02
DSX$SETPRIEXV	********	X	02	EXE$READ	********	X	02
DSX$SETVEC	********	X	02	EXE$READCHK	********	X	02
DSX$SHOWCHAN	********	X	02	EXE$READCHKR	********	X	02
DSX$SHOWIDLE	********	X	01	EXE$READEF	********	X	02
DSX$STARTATTACHED	********	X	01	EXE$READLOCK	********	X	02
DSX$WAITMS	********	X	02	EXE$READLOCKR	********	X	02
DSX$WAITUS	********	X	02	EXE$SENSEMODE	********	X	02
DSX$_CLEAR_INTERLOCK	********W	GX	00	EXE$SETAST	********	X	02
DSX$_INTERLOCK_TEST	********W	GX	00	EXE$SETCHAR	********	X	02
DSX$_SET_INTERLOCK	********W	GX	00	EXE$SETEF	********	X	02
DS_ERRSUP	********	X	03	EXE$SETIMR	********	X	02
ERL$DEVICERR	********	X	02	EXE$SETMODE	********	X	02
ERL$DEVICTMO	********	X	02	EXE$SETPRT	********	X	02
ERL$RELEASEMB	********	X	02	EXE$SNDEVMSG	********	X	02
EXE$ABORTIO	********	X	02	EXE$UNWIND	********	X	02
EXE$ALLOC	********	X	02	EXE$WAITFR	********	X	02
EXE$ALLOCBUF	********	X	02	EXE$WAKE	********	X	02
EXE$ALLOCIRP	********	X	02	EXE$WFLAND	********	X	02
EXE$ALONONPAGED	********	X	02	EXE$WFLOR	********	X	02
EXE$ALTQUEPKT	********	X	02	EXE$WRITE	********	X	02
EXE$ASCTIM	********	X	02	EXE$WRITECHK	********	X	02
EXE$ASSIGN	********	X	02	EXE$WRITECHKR	********	X	02
EXE$BINTIM	********	X	02	EXE$WRITELOCK	********	X	02
EXE$BUFFRQUOTA	********	X	02	EXE$WRITELOCKR	********	X	02
EXE$BUFQUOPRC	********	X	02	EXE$ZEROPARM	********	X	02
EXE$CANCEL	********	X	02	FAKE_JUMP	00000032 R	D	03
EXE$CANTIM	********	X	02	IOC$ALLOSPT	********	X	02
EXE$CARRIAGE	********	X	02	IOC$ALOUBAMAP	********	X	02
EXE$CLREF	********	X	02	IOC$ALOUBAMAPN	********	X	02
EXE$CNTREG	********	X	02	IOC$ALTUBAMAP	********	X	02
EXE$DASSGN	********	X	02	IOC$APPLYECC	********	X	02

IOC$CANCELIO	********	X	02	SCH$UNLOCK	********	X	02
IOC$DIAGBUFILL	********	X	02	SYS$ALLOC	00000238	RG D	02
IOC$GETBYTE	********	X	02	SYS$ASCTIM	00000248	RG D	02
IOC$GL_PSBL	0000081C	RG D	02	SYS$ASSIGN	00000250	RG D	02
IOC$GL_PSFL	00000818	RG D	02	SYS$BINTIM	00000258	RG D	02
IOC$INITBUFWIND	********	X	02	SYS$CALL_HANDL	00000210	RG D	02
IOC$INITIATE	********	X	02	SYS$CANCEL	00000260	RG D	02
IOC$LOADMBAMAP	********	X	02	SYS$CANTIM	00000268	RG D	02
IOC$LOADUBAMAP	********	X	02	SYS$CLOSE	000005B8	RG D	02
IOC$LOADUBAMAPA	********	X	02	SYS$CLREF	00000298	RG D	02
IOC$MOVFRUSER	********	X	02	SYS$CNTREG	000002A0	RG D	02
IOC$MOVFRUSER1	********	X	02	SYS$CONNECT	000005C0	RG D	02
IOC$MOVFRUSER2	********	X	02	SYS$DALLOC	000002D8	RG D	02
IOC$MOVTOUSER	********	X	02	SYS$DASSGN	000002E0	RG D	02
IOC$MOVTOUSER1	********	X	02	SYS$DISCONNECT	000005D0	RG D	02
IOC$MOVTOUSER2	********	X	02	SYS$EXIT	00000340	RG D	02
IOC$PTETOPFN	********	X	02	SYS$EXPREG	00000348	RG D	02
IOC$PURGDATAP	********	X	02	SYS$FAO	00000350	RG D	02
IOC$PUTBYTE	********	X	02	SYS$FAOL	00000358	RG D	02
IOC$RELCHAN	********	X	02	SYS$GET	00000580	RG D	02
IOC$RELDATAP	********	X	02	SYS$GETCHN	000004C8	RG D	02
IOC$RELMAPREG	********	X	02	SYS$GETTIM	00000378	RG D	02
IOC$RELSCHAN	********	X	02	SYS$GL_OPRMBX	00000800	RG D	02
IOC$REQCOM	********	X	02	SYS$GTCHAN	00000380	RG D	02
IOC$REQDATAP	********	X	02	SYS$HIBER	00000388	RG D	02
IOC$REQDATAPNW	********	X	02	SYS$LKWSET	000003A0	RG D	02
IOC$REQMAPREG	********	X	02	SYS$NUMTIM	000003B8	RG D	02
IOC$REQPCHANH	********	X	02	SYS$OPEN	00000608	RG D	02
IOC$REQPCHANL	********	X	02	SYS$QIO	000003C8	RG D	02
IOC$REQSCHANH	********	X	02	SYS$QIOW	00000200	RG D	02
IOC$REQSCHANL	********	X	02	SYS$READ	00000590	RG D	02
IOC$RETURN	********	X	02	SYS$READEF	000003D0	RG D	02
IOC$RTN	00000930	RG D	02	SYS$SETAST	000003F8	RG D	02
IOC$SENSEDISK	********	X	02	SYS$SETEF	00000400	RG D	02
IOC$UPDATRANSP	********	X	02	SYS$SETIMR	00000420	RG D	02
IOC$WFIKPCH	********	X	02	SYS$SETPRI	00000428	RG D	02
IOC$WFIRLCH	********	X	02	SYS$SETPRT	00000430	RG D	02
KB_CHECK	********	X	03	SYS$SETRWM	00000438	RG D	02
LEAP_APT	000001F4	R D	02	SYS$TRNLOG	00000458	RG D	02
LEAP_BOOT	000001EC	R D	02	SYS$ULKPAG	00000460	RG D	02
LF	= 0000000A	D		SYS$ULWSET	00000468	RG D	02
MMG$GL_P0BASE	00000808	RG D	02	SYS$UNWIND	00000470	RG D	02
MMG$GL_SPTBASE	00000804	RG D	02	SYS$WAITFR	00000478	RG D	02
MT$CHECK_ACCESS	********	X	02	SYS$WAKE	00000480	RG D	02
P0PT_BASE	********	X	02	SYS$WFLAND	00000488	RG D	02
RESRVD	000001FA	R D	02	SYS$WFLOR	00000490	RG D	02
RESRVD_ENTRY	0000001D	R D	03	T_RESRVD	00000006	R D	01
RMS$CLOSE	********	X	02				
RMS$CONNECT	********	X	02				
RMS$DISCONNECT	********	X	02				
RMS$GET	********	X	02				
RMS$OPEN	********	X	02				
RMS$READ	********	X	02				
SCH$GL_PCBVEC	0000080C	RG D	02				
SCH$LOCKW	********	X	02				
SCH$POSTEF	********	X	02				
SCH$QAST	********	X	02				

+-----------------+
! Psect synopsis !
+-----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
DATA	00000090 (144.)	01 (1.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC QUAD
$BASE	00000B28 (2856.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC QUAD
CODE	00000039 (57.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL                          VALUE        DEFINITION      REFERENCES...
------                          -----        ----------      -------------
$ER                            =00000002     1997   (1)     #-1997   (1)
$MODULE                         00000000-R   248    (1)       1997   (1)
ACP$ACCESS                      00000000-XR                   1650   (1)
ACP$DEACCESS                    00000000-XR                   1652   (1)
ACP$MODIFY                      00000000-XR                   1654   (1)
ACP$MOUNT                       00000000-XR                   1656   (1)
ACP$READBLK                     00000000-XR                   1658   (1)
ACP$WRITEBLK                    00000000-XR                   1660   (1)
APT                             00000000-XR                   649    (1)
BOOT                            00000000-XR                   643    (1)
COM$DELATTNAST                  00000000-XR                   1824   (1)
COM$DRVDEALMEM                  00000000-XR                   1828   (1)
COM$FLUSHATTNS                  00000000-XR                   1826   (1)
COM$POST                        00000000-XR                   1764   (1)
COM$SETATTNAST                  00000000-XR                   1822   (1)
CR                             =0000000D     238    (1)       251    (1)
DS$ABORT                        00000020-R   326    (1)
DS$ABORTWAIT                    00000070-R   385    (1)
DS$APT_ENTRY                    00000004-R   301    (1)
DS$AQ_SSEND                    =000007FF-R   1624   (1)
DS$AQ_SYSSRV                    00000200-R   657    (1)       1625   (1)
DS$ASKADR                       00000090-R   410    (1)
DS$ASKDATA                      00000080-R   398    (1)
DS$ASKLGCL                      00000098-R   416    (1)
DS$ASKSTR                       000000A0-R   422    (1)
DS$ASKVLD                       00000088-R   404    (1)
DS$ATTACH                       000001A8-R   599    (1)
DS$BGNSUB                       00000030-R   343    (1)
DS$BRANCH                       000000A8-R   429    (1)
DS$BREAK                        00000058-R   368    (1)
DS$CANWAIT                      00000070-R   384    (1)
DS$CHANNEL                      00000180-R   573    (1)
DS$CKLOOP                       00000040-R   353    (1)
DS$CLRVEC                       00000168-R   558    (1)
DS$CNTRLC                       00000078-R   390    (1)
DS$CVTREG                       000000B0-R   436    (1)
DS$DOSUMMARY                    00000028-R   335    (1)
DS$ENDPASS                      00000010-R   313    (1)
DS$ENDSUB                       00000038-R   348    (1)
DS$ENTRY                        00000000-R   297    (1)
DS$ERRDEV                       000000C8-R   451    (1)
DS$ERRHARD                      000000D0-R   456    (1)
DS$ERRPREP                      00000108-R   491    (1)
DS$ERRSOFT                      000000D8-R   461    (1)
DS$ERRSYS                       000000C0-R   446    (1)
DS$ESCAPE                       00000050-R   363    (1)
DS$FREEDBGSYM                   000001B8-R   608    (1)
DS$GETADDRESS                   00000090-R   411    (1)
DS$GETBUF                       00000120-R   510    (1)
DS$GETDATA                      00000080-R   399    (1)
```

```
DS$GETLOGICAL                   00000098-R   417    (1)
DS$GETSTRING                    000000A0-R   423    (1)
DS$GETTERM                      000001C8-R   618    (1)
DS$GETVIELD                     00000088-R   405    (1)
DS$GL_PCBVEC                    00000000-XR                1644   (1)
DS$GPHARD                       00000018-R   318    (1)
DS$HELP                         000001B0-R   603    (1)
DS$INITSCB                      00000170-R   563    (1)
DS$INLOOP                       00000048-R   358    (1)
DS$K_SSSIZE                     =00000600    1625   (1)
DS$LOAD                         00000198-R   589    (1)
DS$LOADPCS                      000001C0-R   613    (1)
DS$MAPDBGBLOCK                  00000118-R   501    (1)
DS$MEMSIZE                      000001D8-R   628    (1)
DS$MMOFF                        00000158-R   545    (1)
DS$MMON                         00000150-R   540    (1)
DS$PARSE                        000000B8-R   441    (1)
DS$PRINTB                       000000E0-R   466    (1)
DS$PRINTF                       000000F0-R   476    (1)
DS$PRINTS                       000000F8-R   481    (1)
DS$PRINTSIG                     00000100-R   486    (1)
DS$PRINTX                       000000E8-R   471    (1)
DS$PROBE                        000001A0-R   594    (1)
DS$RELBUF                       00000128-R   515    (1)
DS$SERVICE_DISPATCHER           000001D0-R   623    (1)
DS$SETIPL                       00000178-R   568    (1)
DS$SETMAP                       00000188-R   578    (1)
DS$SETPRIEXV                    00000110-R   496    (1)
DS$SETVEC                       00000160-R   553    (1)
DS$SHOCHAN                      00000190-R   583    (1)
DS$SHOWCHAN                     00000190-R   584    (1)
DS$SRVDISP_TABLE                00000038-R   1854   (1)       1951   (1)
DS$SRVDISP_TABLE_END            00000090-R   1909   (1)
DS$SUMMARY                      00000028-R   334    (1)
DS$WAITMS                       00000060-R   374    (1)
DS$WAITUS                       00000068-R   379    (1)
DSX$ABORT                       00000000-XR                328    (1)
DSX$ATTACH                      00000000-XR                601    (1)
DSX$BGNSUB                      00000000-XR                345    (1)
DSX$BOOTATTACHED                00000000-XR                1868   (1)
DSX$BRANCH                      00000000-XR                431    (1)
DSX$CANWAIT                     00000000-XR                387    (1)
DSX$CHANNEL                     00000000-XR                575    (1)
DSX$CKLOOP                      00000000-XR                355    (1)
DSX$CLRVEC                      00000000-XR                560    (1)
DSX$CNTRLC                      00000000-XR                392    (1)
DSX$CVTREG                      00000000-XR                438    (1)
DSX$DOSUMMARY                   00000000-XR                337    (1)
DSX$ENDPASS                     00000000-XR                315    (1)
DSX$ENDSUB                      00000000-XR                350    (1)
DSX$ERRDEV                      00000000-XR                453    (1)
DSX$ERRHARD                     00000000-XR                458    (1)
DSX$ERRPREP                     00000000-XR                493    (1)
DSX$ERRSOFT                     00000000-XR                463    (1)
DSX$ERRSYS                      00000000-XR                448    (1)
DSX$ESCAPE                      00000000-XR                365    (1)
DSX$FREEDBGSYM                  00000000-XR                610    (1)
```

```
DSX$GETADDRESS                  00000000-XR                     413     (1)
DSX$GETBUF                      00000000-XR                     512     (1)
DSX$GETDATA                     00000000-XR                     401     (1)
DSX$GETLOGICAL                  00000000-XR                     419     (1)
DSX$GETSTRING                   00000000-XR                     425     (1)
DSX$GETSYSBUF                   00000000-XR                     1885    (1)     229     (1)
DSX$GETTERM                     00000000-XR                     620     (1)
DSX$GETVIELD                    00000000-XR                     407     (1)
DSX$GPHARD                      00000000-XR                     320     (1)
DSX$HALTATTACHED                00000000-XR                     1874    (1)
DSX$HELP                        00000000-XR                     605     (1)
DSX$INITSCB                     00000000-XR                     565     (1)
DSX$INLOOP                      00000000-XR                     360     (1)
DSX$LOAD                        00000000-XR                     591     (1)
DSX$LOADPCS                     00000000-XR                     615     (1)
DSX$MAPDBGBLOCK                 00000000-XR                     503     (1)
DSX$MEMSIZE                     00000000-XR                     630     (1)
DSX$MMOFF                       00000000-XR                     547     (1)
DSX$MMON                        00000000-XR                     542     (1)
DSX$PARSE                       00000000-XR                     443     (1)
DSX$PPSCB                       00000000-XR                     1907    (1)
DSX$PRINTB                      00000000-XR                     468     (1)
DSX$PRINTF                      00000000-XR                     478     (1)
DSX$PRINTREV                    00000000-XR                     1891    (1)
DSX$PRINTS                      00000000-XR                     483     (1)
DSX$PRINTSIG                    00000000-XR                     488     (1)
DSX$PRINTX                      00000000-XR                     473     (1)
DSX$PROBE                       00000000-XR                     596     (1)
DSX$RELBUF                      00000000-XR                     517     (1)
DSX$RELSYSBUF                   00000000-XR                     1888    (1)     229     (1)
DSX$SERVICE_DISPATCHER          00000000-R      1948    (1)     625     (1)
DSX$SETIPL                      00000000-XR                     570     (1)
DSX$SETMAP                      00000000-XR                     580     (1)
DSX$SETPRIEXV                   00000000-XR                     498     (1)
DSX$SETVEC                      00000000-XR                     555     (1)
DSX$SHOWCHAN                    00000000-XR                     586     (1)
DSX$SHOWIDLE                    00000000-XR                     1877    (1)
DSX$STARTATTACHED               00000000-XR                     1871    (1)
DSX$WAITMS                      00000000-XR                     376     (1)
DSX$WAITUS                      00000000-XR                     381     (1)
DSX$_CLEAR_INTERLOCK            00000000-XR                     1903    (1)     232     (1)
DSX$_INTERLOCK_TEST             00000000-XR                     1895    (1)     230     (1)
DSX$_SET_INTERLOCK              00000000-XR                     1899    (1)     231     (1)
DS_ERRSUP                       00000000-XR                     1997    (1)
ERL$DEVICERR                    00000000-XR                     1662    (1)
ERL$DEVICTMO                    00000000-XR                     1664    (1)
ERL$RELEASEMB                   00000000-XR                     1770    (1)
EXE$ABORTIO                     00000000-XR                     1666    (1)
EXE$ALLOC                       00000000-XR                     694     (1)
EXE$ALLOCBUF                    00000000-XR                     1774    (1)
EXE$ALLOCIRP                    00000000-XR                     1730    (1)
EXE$ALONONPAGED                 00000000-XR                     1732    (1)
EXE$ALTQUEPKT                   00000000-XR                     1820    (1)
EXE$ASCTIM                      00000000-XR                     704     (1)
EXE$ASSIGN                      00000000-XR                     709     (1)
EXE$BINTIM                      00000000-XR                     714     (1)
EXE$BUFFRQUOTA                  00000000-XR                     1778    (1)
```

EXE$BUFQUOPRC	00000000-XR			1776	(1)		
EXE$CANCEL	00000000-XR			719	(1)	794	(1)
EXE$CANTIM	00000000-XR			724	(1)		
EXE$CARRIAGE	00000000-XR			1796	(1)		
EXE$CLREF	00000000-XR			754	(1)		
EXE$CNTREG	00000000-XR			759	(1)		
EXE$DASSGN	00000000-XR			799	(1)		
EXE$DEANONPAGED	00000000-XR			1734	(1)		
EXE$EXPREG	00000000-XR			864	(1)		
EXE$FAO	00000000-XR			869	(1)		
EXE$FAOL	00000000-XR			874	(1)		
EXE$FINISHIO	00000000-XR			1780	(1)		
EXE$FINISHIOC	00000000-XR			1736	(1)		
EXE$FORK	00000000-XR			1738	(1)		
EXE$GETTIM	00000000-XR			894	(1)		
EXE$GL_ABSTIM	00000810-R	1645	(1)				
EXE$GQ_SYSTIME	00000B00-R	1834	(1)				
EXE$GTCHAN	00000000-XR			1110	(1)	899	(1)
EXE$HIBER	00000000-XR			904	(1)		
EXE$INSERTIRP	00000000-XR			1758	(1)		
EXE$INSIOQ	00000000-XR			1830	(1)		
EXE$IOFORK	00000000-XR			1668	(1)		
EXE$MAXACMODE	00000000-XR			1841	(1)		
EXE$MODIFY	00000000-XR			1792	(1)		
EXE$MODIFYLOCK	00000000-XR			1794	(1)		
EXE$MODIFYLOCKR	00000000-XR			1740	(1)		
EXE$NUMTIM	00000000-XR			935	(1)		
EXE$ONEPARM	00000000-XR			1670	(1)		
EXE$PWRTIMCHK	00000000-XR			1672	(1)		
EXE$QIO	00000000-XR			660	(1)		
EXE$QIO2	=00000204-R	661	(1)	945	(1)		
EXE$QIODRVPKT	00000000-XR			1742	(1)		
EXE$QIORETURN	00000000-XR			1760	(1)		
EXE$READ	00000000-XR			1782	(1)		
EXE$READCHK	00000000-XR			1784	(1)		
EXE$READCHKR	00000000-XR			1786	(1)		
EXE$READEF	00000000-XR			951	(1)		
EXE$READLOCK	00000000-XR			1843	(1)		
EXE$READLOCKR	00000000-XR			1752	(1)		
EXE$SENSEMODE	00000000-XR			1674	(1)		
EXE$SETAST	00000000-XR			976	(1)		
EXE$SETCHAR	00000000-XR			1676	(1)		
EXE$SETEF	00000000-XR			981	(1)		
EXE$SETIMR	00000000-XR			1001	(1)		
EXE$SETMODE	00000000-XR			1678	(1)		
EXE$SETPRT	00000000-XR			1012	(1)		
EXE$SNDEVMSG	00000000-XR			1680	(1)		
EXE$UNWIND	00000000-XR			1055	(1)		
EXE$WAITFR	00000000-XR			1060	(1)		
EXE$WAKE	00000000-XR			1065	(1)		
EXE$WFLAND	00000000-XR			1070	(1)		
EXE$WFLOR	00000000-XR			1075	(1)		
EXE$WRITE	00000000-XR			1788	(1)		
EXE$WRITECHK	00000000-XR			1750	(1)		
EXE$WRITECHKR	00000000-XR			1790	(1)		
EXE$WRITELOCK	00000000-XR			1744	(1)		
EXE$WRITELOCKR	00000000-XR			1754	(1)		

```
EXE$ZEROPARM                    00000000-XR                     1682    (1)
FAKE_JUMP                       00000032-R    1999    (1)       370     (1)
IOC$ALLOSPT                     00000000-XR                     1839    (1)
IOC$ALOUBAMAP                   00000000-XR                     1766    (1)
IOC$ALOUBAMAPN                  00000000-XR                     1772    (1)
IOC$ALTUBAMAP                   00000000-XR                     1768    (1)
IOC$APPLYECC                    00000000-XR                     1684    (1)
IOC$CANCELIO                    00000000-XR                     1686    (1)
IOC$DIAGBUFILL                  00000000-XR                     1688    (1)
IOC$GETBYTE                     00000000-XR                     1802    (1)
IOC$GL_PSBL                     0000081C-R    1648    (1)
IOC$GL_PSFL                     00000818-R    1647    (1)       1647    (1)       1648    (1)
IOC$INITBUFWIND                 00000000-XR                     1806    (1)
IOC$INITIATE                    00000000-XR                     1756    (1)
IOC$LOADMBAMAP                  00000000-XR                     1690    (1)
IOC$LOADUBAMAP                  00000000-XR                     1692    (1)
IOC$LOADUBAMAPA                 00000000-XR                     1800    (1)
IOC$MOVFRUSER                   00000000-XR                     1837    (1)
IOC$MOVFRUSER1                  00000000-XR                     1810    (1)
IOC$MOVFRUSER2                  00000000-XR                     1808    (1)
IOC$MOVTOUSER                   00000000-XR                     1694    (1)
IOC$MOVTOUSER1                  00000000-XR                     1814    (1)
IOC$MOVTOUSER2                  00000000-XR                     1812    (1)
IOC$PTETOPFN                    00000000-XR                     1832    (1)
IOC$PURGDATAP                   00000000-XR                     1696    (1)
IOC$PUTBYTE                     00000000-XR                     1804    (1)
IOC$RELCHAN                     00000000-XR                     1698    (1)
IOC$RELDATAP                    00000000-XR                     1700    (1)
IOC$RELMAPREG                   00000000-XR                     1702    (1)
IOC$RELSCHAN                    00000000-XR                     1704    (1)
IOC$REQCOM                      00000000-XR                     1706    (1)
IOC$REQDATAP                    00000000-XR                     1708    (1)
IOC$REQDATAPNW                  00000000-XR                     1762    (1)
IOC$REQMAPREG                   00000000-XR                     1710    (1)
IOC$REQPCHANH                   00000000-XR                     1712    (1)
IOC$REQPCHANL                   00000000-XR                     1714    (1)
IOC$REQSCHANH                   00000000-XR                     1798    (1)
IOC$REQSCHANL                   00000000-XR                     1716    (1)
IOC$RETURN                      00000000-XR                     1718    (1)
IOC$RTN                         00000930-R    1718    (1)
IOC$SENSEDISK                   00000000-XR                     1720    (1)
IOC$UPDATRANSP                  00000000-XR                     1722    (1)
IOC$WFIKPCH                     00000000-XR                     1724    (1)
IOC$WFIRLCH                     00000000-XR                     1726    (1)
KB_CHECK                        00000000-XR                     2000    (1)
LEAP_APT                        000001F4-R    648     (1)     #-302    (1)
LEAP_BOOT                       000001EC-R    642     (1)     #-298    (1)
LF                              =0000000A     239     (1)       251    (1)
MMG$GL_POBASE                   00000808-R    1641    (1)
MMG$GL_SPTBASE                  00000804-R    1639    (1)
MT$CHECK_ACCESS                 00000000-XR                     1728    (1)
POPT_BASE                       00000000-XR                     1642    (1)
RESRVD                          000001FA-R    651     (1)     #-641    (1)     #-646    (1)
RESRVD_ENTRY                    0000001D-R    1996    (1)       1023    (1)       1028    (1)       1033    (1)       1038    (1)
                                                                1080    (1)       1085    (1)       1090    (1)       1095    (1)
                                                                1100    (1)       1105    (1)       1115    (1)       1120    (1)
                                                                1125    (1)       1130    (1)       1135    (1)       1140    (1)
```

				1145	(1)	1150	(1)	1155	(1)	1160	(1)
				1165	(1)	1170	(1)	1175	(1)	1180	(1)
				1185	(1)	1190	(1)	1195	(1)	1200	(1)
				1205	(1)	1210	(1)	1215	(1)	1220	(1)
				1230	(1)	1240	(1)	1245	(1)	1250	(1)
				1255	(1)	1270	(1)	1280	(1)	1285	(1)
				1290	(1)	1295	(1)	1300	(1)	1305	(1)
				1316	(1)	1321	(1)	1326	(1)	1331	(1)
				1336	(1)	1341	(1)	1346	(1)	1351	(1)
				1356	(1)	1361	(1)	1366	(1)	1371	(1)
				1376	(1)	1381	(1)	1386	(1)	1391	(1)
				1396	(1)	1401	(1)	1406	(1)	1411	(1)
				1416	(1)	1421	(1)	1426	(1)	1431	(1)
				1436	(1)	1441	(1)	1446	(1)	1451	(1)
				1456	(1)	1461	(1)	1466	(1)	1471	(1)
				1476	(1)	1481	(1)	1486	(1)	1491	(1)
				1496	(1)	1501	(1)	1506	(1)	1511	(1)
				1516	(1)	1521	(1)	1526	(1)	1531	(1)
				1536	(1)	1541	(1)	1546	(1)	1551	(1)
				1556	(1)	1561	(1)	1566	(1)	1571	(1)
				1576	(1)	1581	(1)	1586	(1)	1591	(1)
				1596	(1)	1601	(1)	1606	(1)	1611	(1)
				1616	(1)	1621	(1)	307	(1)	522	(1)
				527	(1)	532	(1)	537	(1)	636	(1)
				651	(1)	674	(1)	679	(1)	684	(1)
				689	(1)	699	(1)	729	(1)	734	(1)
				739	(1)	744	(1)	749	(1)	764	(1)
				769	(1)	774	(1)	779	(1)	784	(1)
				789	(1)	804	(1)	809	(1)	814	(1)
				819	(1)	824	(1)	829	(1)	834	(1)
				839	(1)	844	(1)	849	(1)	854	(1)
				879	(1)	884	(1)	889	(1)	909	(1)
				914	(1)	925	(1)	930	(1)	940	(1)
				956	(1)	961	(1)	966	(1)	971	(1)
				986	(1)	991	(1)	996	(1)		
RMS$CLOSE	00000000-XR			1260	(1)						
RMS$CONNECT	00000000-XR			1265	(1)						
RMS$DISCONNECT	00000000-XR			1275	(1)						
RMS$GET	00000000-XR			1225	(1)						
RMS$OPEN	00000000-XR			1310	(1)						
RMS$READ	00000000-XR			1235	(1)						
SCH$GL_PCBVEC	0000080C-R	1643	(1)								
SCH$LOCKW	00000000-XR			1816	(1)						
SCH$POSTEF	00000000-XR			1746	(1)						
SCH$QAST	00000000-XR			1748	(1)						
SCH$UNLOCK	00000000-XR			1818	(1)						
SYS$ALLOC	00000238-R	692	(1)								
SYS$ASCTIM	00000248-R	702	(1)								
SYS$ASSIGN	00000250-R	707	(1)								
SYS$BINTIM	00000258-R	712	(1)								
SYS$CALL_HANDL	00000210-R	667	(1)								
SYS$CANCEL	00000260-R	717	(1)								
SYS$CANTIM	00000268-R	722	(1)								
SYS$CLOSE	000005B8-R	1258	(1)								
SYS$CLREF	00000298-R	752	(1)								
SYS$CNTREG	000002A0-R	757	(1)								
SYS$CONNECT	000005C0-R	1263	(1)								

SYS$DALLOC	000002D8-R	792	(1)		
SYS$DASSGN	000002E0-R	797	(1)		
SYS$DISCONNECT	000005D0-R	1273	(1)		
SYS$EXIT	00000340-R	857	(1)		
SYS$EXPREG	00000348-R	862	(1)		
SYS$FAO	00000350-R	867	(1)		
SYS$FAOL	00000358-R	872	(1)		
SYS$GET	00000580-R	1223	(1)		
SYS$GETCHN	000004C8-R	1108	(1)		
SYS$GETTIM	00000378-R	892	(1)		
SYS$GL_OPRMBX	00000800-R	1637	(1)		
SYS$GTCHAN	00000380-R	897	(1)		
SYS$HIBER	00000388-R	902	(1)		
SYS$LKWSET	000003A0-R	917	(1)		
SYS$NUMTIM	000003B8-R	933	(1)		
SYS$OPEN	00000608-R	1308	(1)		
SYS$QIO	000003C8-R	943	(1)		
SYS$QIOW	00000200-R	658	(1)		
SYS$READ	00000590-R	1233	(1)		
SYS$READEF	000003D0-R	949	(1)		
SYS$SETAST	000003F8-R	974	(1)		
SYS$SETEF	00000400-R	979	(1)		
SYS$SETIMR	00000420-R	999	(1)		
SYS$SETPRI	00000428-R	1004	(1)		
SYS$SETPRT	00000430-R	1010	(1)		
SYS$SETRWM	00000438-R	1015	(1)		
SYS$TRNLOG	00000458-R	1036	(1)		
SYS$ULKPAG	00000460-R	1041	(1)		
SYS$ULWSET	00000468-R	1047	(1)		
SYS$UNWIND	00000470-R	1053	(1)		
SYS$WAITFR	00000478-R	1058	(1)	#-663	(1)
SYS$WAKE	00000480-R	1063	(1)		
SYS$WFLAND	00000488-R	1068	(1)		
SYS$WFLOR	00000490-R	1073	(1)		
T_RESRVD	00000006-R	250	(1)	1997	(1)

```
                                        +-------------------------+
                                        ! Macros Cross Reference !
                                        +-------------------------+

MACRO            SIZE          DEFINITION       REFERENCES...
-----            ----          ----------       -------------
$PUSHADR         1             1997   (1)       1997   (1)
ERRSUP_S         1             1997   (1)       1997   (1)
MODNAM           1             248    (1)       248    (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	1952	(1)												
BLBC	00E9	662	(1)												
BRB	0011	641	(1)	646	(1)										
BRW	0031	298	(1)	302	(1)	663	(1)								
CALLG	00FA	1956	(1)	668	(1)										
CALLS	00FB	1997	(1)												
JMP	0017	1001	(1)	1012	(1)	1023	(1)	1028	(1)	1033	(1)	1038	(1)	1055	(1)
		1060	(1)	1065	(1)	1070	(1)	1075	(1)	1080	(1)	1085	(1)	1090	(1)
		1095	(1)	1100	(1)	1105	(1)	1110	(1)	1115	(1)	1120	(1)	1125	(1)
		1130	(1)	1135	(1)	1140	(1)	1145	(1)	1150	(1)	1155	(1)	1160	(1)
		1165	(1)	1170	(1)	1175	(1)	1180	(1)	1185	(1)	1190	(1)	1195	(1)
		1200	(1)	1205	(1)	1210	(1)	1215	(1)	1220	(1)	1225	(1)	1230	(1)
		1235	(1)	1240	(1)	1245	(1)	1250	(1)	1255	(1)	1260	(1)	1265	(1)
		1270	(1)	1275	(1)	1280	(1)	1285	(1)	1290	(1)	1295	(1)	1300	(1)
		1305	(1)	1310	(1)	1316	(1)	1321	(1)	1326	(1)	1331	(1)	1336	(1)
		1341	(1)	1346	(1)	1351	(1)	1356	(1)	1361	(1)	1366	(1)	1371	(1)
		1376	(1)	1381	(1)	1386	(1)	1391	(1)	1396	(1)	1401	(1)	1406	(1)
		1411	(1)	1416	(1)	1421	(1)	1426	(1)	1431	(1)	1436	(1)	1441	(1)
		1446	(1)	1451	(1)	1456	(1)	1461	(1)	1466	(1)	1471	(1)	1476	(1)
		1481	(1)	1486	(1)	1491	(1)	1496	(1)	1501	(1)	1506	(1)	1511	(1)
		1516	(1)	1521	(1)	1526	(1)	1531	(1)	1536	(1)	1541	(1)	1546	(1)
		1551	(1)	1556	(1)	1561	(1)	1566	(1)	1571	(1)	1576	(1)	1581	(1)
		1586	(1)	1591	(1)	1596	(1)	1601	(1)	1606	(1)	1611	(1)	1616	(1)
		1621	(1)	1650	(1)	1652	(1)	1654	(1)	1656	(1)	1658	(1)	1660	(1)
		1662	(1)	1664	(1)	1666	(1)	1668	(1)	1670	(1)	1672	(1)	1674	(1)
		1676	(1)	1678	(1)	1680	(1)	1682	(1)	1684	(1)	1686	(1)	1688	(1)
		1690	(1)	1692	(1)	1694	(1)	1696	(1)	1698	(1)	1700	(1)	1702	(1)
		1704	(1)	1706	(1)	1708	(1)	1710	(1)	1712	(1)	1714	(1)	1716	(1)
		1718	(1)	1720	(1)	1722	(1)	1724	(1)	1726	(1)	1728	(1)	1730	(1)
		1732	(1)	1734	(1)	1736	(1)	1738	(1)	1740	(1)	1742	(1)	1744	(1)
		1746	(1)	1748	(1)	1750	(1)	1752	(1)	1754	(1)	1756	(1)	1758	(1)
		1760	(1)	1762	(1)	1764	(1)	1766	(1)	1768	(1)	1770	(1)	1772	(1)
		1774	(1)	1776	(1)	1778	(1)	1780	(1)	1782	(1)	1784	(1)	1786	(1)
		1788	(1)	1790	(1)	1792	(1)	1794	(1)	1796	(1)	1798	(1)	1800	(1)
		1802	(1)	1804	(1)	1806	(1)	1808	(1)	1810	(1)	1812	(1)	1814	(1)
		1816	(1)	1818	(1)	1820	(1)	1822	(1)	1824	(1)	1826	(1)	1828	(1)
		1830	(1)	1832	(1)	1837	(1)	1839	(1)	1841	(1)	1843	(1)	1868	(1)
		1871	(1)	1874	(1)	1877	(1)	1885	(1)	1888	(1)	1891	(1)	1895	(1)
		1899	(1)	1903	(1)	1907	(1)	307	(1)	315	(1)	320	(1)	328	(1)
		337	(1)	345	(1)	350	(1)	355	(1)	360	(1)	365	(1)	370	(1)
		376	(1)	381	(1)	387	(1)	392	(1)	401	(1)	407	(1)	413	(1)
		419	(1)	425	(1)	431	(1)	438	(1)	443	(1)	448	(1)	453	(1)
		458	(1)	463	(1)	468	(1)	473	(1)	478	(1)	483	(1)	488	(1)
		493	(1)	498	(1)	503	(1)	512	(1)	517	(1)	522	(1)	527	(1)
		532	(1)	537	(1)	542	(1)	547	(1)	555	(1)	560	(1)	565	(1)
		570	(1)	575	(1)	580	(1)	586	(1)	591	(1)	596	(1)	601	(1)
		605	(1)	610	(1)	615	(1)	620	(1)	625	(1)	630	(1)	636	(1)
		643	(1)	649	(1)	651	(1)	674	(1)	679	(1)	684	(1)	689	(1)
		694	(1)	699	(1)	704	(1)	709	(1)	714	(1)	719	(1)	724	(1)
		729	(1)	734	(1)	739	(1)	744	(1)	749	(1)	754	(1)	759	(1)

		764	(1)	769	(1)	774	(1)	779	(1)	784	(1)	789	(1)	794	(1)
		799	(1)	804	(1)	809	(1)	814	(1)	819	(1)	824	(1)	829	(1)
		834	(1)	839	(1)	844	(1)	849	(1)	854	(1)	864	(1)	869	(1)
		874	(1)	879	(1)	884	(1)	889	(1)	894	(1)	899	(1)	904	(1)
		909	(1)	914	(1)	925	(1)	930	(1)	935	(1)	940	(1)	951	(1)
		956	(1)	961	(1)	966	(1)	971	(1)	976	(1)	981	(1)	986	(1)
		991	(1)	996	(1)										
JSB	0016	2000	(1)	660	(1)	945	(1)								
MOVAB	009E	1951	(1)												
MOVL	00D0	1006	(1)	1017	(1)	1043	(1)	1049	(1)	919	(1)				
MULL3	00C5	1950	(1)												
PUSHAL	00DF	1997	(1)												
PUSHL	00DD	1997	(1)												
RET	0004	1007	(1)	1018	(1)	1044	(1)	1050	(1)	1957	(1)	2001	(1)	664	(1)
		859	(1)	920	(1)	946	(1)								
RSB	0005	669	(1)												
SUBL3	00C3	1953	(1)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	1644	(1)														
.ALIGN	1003	(1)	1009	(1)	1014	(1)	1020	(1)	1025	(1)	1030	(1)	1035	(1)	1040	(1)
	1046	(1)	1052	(1)	1057	(1)	1062	(1)	1067	(1)	1072	(1)	1077	(1)	1082	(1)
	1087	(1)	1092	(1)	1097	(1)	1102	(1)	1107	(1)	1112	(1)	1117	(1)	1122	(1)
	1127	(1)	1132	(1)	1137	(1)	1142	(1)	1147	(1)	1152	(1)	1157	(1)	1162	(1)
	1167	(1)	1172	(1)	1177	(1)	1182	(1)	1187	(1)	1192	(1)	1197	(1)	1202	(1)
	1207	(1)	1212	(1)	1217	(1)	1222	(1)	1227	(1)	1232	(1)	1237	(1)	1242	(1)
	1247	(1)	1252	(1)	1257	(1)	1262	(1)	1267	(1)	1272	(1)	1277	(1)	1282	(1)
	1287	(1)	1292	(1)	1297	(1)	1302	(1)	1307	(1)	1313	(1)	1318	(1)	1323	(1)
	1328	(1)	1333	(1)	1338	(1)	1343	(1)	1348	(1)	1353	(1)	1358	(1)	1363	(1)
	1368	(1)	1373	(1)	1378	(1)	1383	(1)	1388	(1)	1393	(1)	1398	(1)	1403	(1)
	1408	(1)	1413	(1)	1418	(1)	1423	(1)	1428	(1)	1433	(1)	1438	(1)	1443	(1)
	1448	(1)	1453	(1)	1458	(1)	1463	(1)	1468	(1)	1473	(1)	1478	(1)	1483	(1)
	1488	(1)	1493	(1)	1498	(1)	1503	(1)	1508	(1)	1513	(1)	1518	(1)	1523	(1)
	1528	(1)	1533	(1)	1538	(1)	1543	(1)	1548	(1)	1553	(1)	1558	(1)	1563	(1)
	1568	(1)	1573	(1)	1578	(1)	1583	(1)	1588	(1)	1593	(1)	1598	(1)	1603	(1)
	1608	(1)	1613	(1)	1618	(1)	1623	(1)	1649	(1)	1651	(1)	1653	(1)	1655	(1)
	1657	(1)	1659	(1)	1661	(1)	1663	(1)	1665	(1)	1667	(1)	1669	(1)	1671	(1)
	1673	(1)	1675	(1)	1677	(1)	1679	(1)	1681	(1)	1683	(1)	1685	(1)	1687	(1)
	1689	(1)	1691	(1)	1693	(1)	1695	(1)	1697	(1)	1699	(1)	1701	(1)	1703	(1)
	1705	(1)	1707	(1)	1709	(1)	1711	(1)	1713	(1)	1715	(1)	1717	(1)	1719	(1)
	1721	(1)	1723	(1)	1725	(1)	1727	(1)	1729	(1)	1731	(1)	1733	(1)	1735	(1)
	1737	(1)	1739	(1)	1741	(1)	1743	(1)	1745	(1)	1747	(1)	1749	(1)	1751	(1)
	1753	(1)	1755	(1)	1757	(1)	1759	(1)	1761	(1)	1763	(1)	1765	(1)	1767	(1)
	1769	(1)	1771	(1)	1773	(1)	1775	(1)	1777	(1)	1779	(1)	1781	(1)	1783	(1)
	1785	(1)	1787	(1)	1789	(1)	1791	(1)	1793	(1)	1795	(1)	1797	(1)	1799	(1)
	1801	(1)	1803	(1)	1805	(1)	1807	(1)	1809	(1)	1811	(1)	1813	(1)	1815	(1)
	1817	(1)	1819	(1)	1821	(1)	1823	(1)	1825	(1)	1827	(1)	1829	(1)	1831	(1)
	1833	(1)	1836	(1)	1838	(1)	1840	(1)	1842	(1)	1844	(1)	1853	(1)	1893	(1)
	1897	(1)	1901	(1)	1905	(1)	300	(1)	304	(1)	312	(1)	317	(1)	325	(1)
	333	(1)	342	(1)	347	(1)	352	(1)	357	(1)	362	(1)	367	(1)	373	(1)
	378	(1)	383	(1)	389	(1)	397	(1)	403	(1)	409	(1)	415	(1)	421	(1)
	428	(1)	435	(1)	440	(1)	445	(1)	450	(1)	455	(1)	460	(1)	465	(1)
	470	(1)	475	(1)	480	(1)	485	(1)	490	(1)	495	(1)	500	(1)	509	(1)
	514	(1)	519	(1)	524	(1)	529	(1)	534	(1)	539	(1)	544	(1)	552	(1)
	557	(1)	562	(1)	567	(1)	572	(1)	577	(1)	582	(1)	588	(1)	593	(1)
	598	(1)	602	(1)	607	(1)	612	(1)	617	(1)	622	(1)	627	(1)	633	(1)
	638	(1)	656	(1)	666	(1)	671	(1)	676	(1)	681	(1)	686	(1)	691	(1)
	696	(1)	701	(1)	706	(1)	711	(1)	716	(1)	721	(1)	726	(1)	731	(1)
	736	(1)	741	(1)	746	(1)	751	(1)	756	(1)	761	(1)	766	(1)	771	(1)
	776	(1)	781	(1)	786	(1)	791	(1)	796	(1)	801	(1)	806	(1)	811	(1)
	816	(1)	821	(1)	826	(1)	831	(1)	836	(1)	841	(1)	846	(1)	851	(1)
	856	(1)	861	(1)	866	(1)	871	(1)	876	(1)	881	(1)	886	(1)	891	(1)
	896	(1)	901	(1)	906	(1)	911	(1)	916	(1)	922	(1)	927	(1)	932	(1)
	937	(1)	942	(1)	948	(1)	953	(1)	958	(1)	963	(1)	968	(1)	973	(1)
	978	(1)	983	(1)	988	(1)	993	(1)	998	(1)						
.ASCIC	248	(1)	251	(1)												
.END	2003	(1)														
.ENDC	1997	(1)														
.ENDM	1997	(1)	248	(1)												

```
.ENTRY          1948  (1)
.IDENT          6     (1)
.IF             1997  (1)
.IFF            1997  (1)
.IIF            1997  (1)
.LIBRARY        217   (1)  218   (1)  219   (1)
.LONG           1638  (1)  1640  (1)  1642  (1)  1645  (1)  1646  (1)  1647  (1)  1648  (1)
.NOSHOW         7     (1)
.PAGE           1019  (1)  1061  (1)  1101  (1)  112   (1)  1141  (1)  1181  (1)  1221  (1)  1261  (1)
                1301  (1)  1342  (1)  1382  (1)  1422  (1)  1462  (1)  1502  (1)  1542  (1)  1582  (1)
                1626  (1)  1847  (1)  1911  (1)  1959  (1)  1995  (1)  212   (1)  240   (1)  252   (1)
                294   (1)  308   (1)  338   (1)  393   (1)  44    (1)  444   (1)  484   (1)  505   (1)
                548   (1)  606   (1)  652   (1)  695   (1)  735   (1)  775   (1)  815   (1)  855   (1)
                895   (1)  936   (1)  977   (1)
.PSECT          1849  (1)  1913  (1)  1961  (1)  246   (1)  295   (1)
.QUAD           1835  (1)
.SBTTL          1848  (1)  1912  (1)  1960  (1)  213   (1)  253   (1)
.SUBTITLE       241   (1)
.TITLE          5     (1)
.VECTOR         1000  (1)  1011  (1)  1054  (1)  1059  (1)  1064  (1)  1069  (1)  1074  (1)  1109  (1)
                1224  (1)  1234  (1)  1259  (1)  1264  (1)  1274  (1)  1309  (1)  1867  (1)  1870  (1)
                1873  (1)  1876  (1)  1890  (1)  1894  (1)  1898  (1)  1902  (1)  1906  (1)  314   (1)
                319   (1)  327   (1)  336   (1)  344   (1)  349   (1)  354   (1)  359   (1)  364   (1)
                375   (1)  380   (1)  386   (1)  391   (1)  400   (1)  406   (1)  412   (1)  418   (1)
                424   (1)  430   (1)  437   (1)  442   (1)  447   (1)  452   (1)  457   (1)  462   (1)
                467   (1)  472   (1)  477   (1)  482   (1)  487   (1)  492   (1)  497   (1)  502   (1)
                511   (1)  516   (1)  541   (1)  546   (1)  554   (1)  559   (1)  564   (1)  569   (1)
                574   (1)  579   (1)  585   (1)  590   (1)  595   (1)  600   (1)  604   (1)  609   (1)
                614   (1)  619   (1)  624   (1)  629   (1)  659   (1)  693   (1)  708   (1)  718   (1)
                723   (1)  753   (1)  758   (1)  793   (1)  798   (1)  863   (1)  898   (1)  903   (1)
                944   (1)  950   (1)  975   (1)  980   (1)
.WEAK           229   (1)  230   (1)  231   (1)  232   (1)
.WORD           1005  (1)  1016  (1)  1022  (1)  1027  (1)  1032  (1)  1037  (1)  1042  (1)  1048  (1)
                1079  (1)  1084  (1)  1089  (1)  1094  (1)  1099  (1)  1104  (1)  1114  (1)  1119  (1)
                1124  (1)  1129  (1)  1134  (1)  1139  (1)  1144  (1)  1149  (1)  1154  (1)  1159  (1)
                1164  (1)  1169  (1)  1174  (1)  1179  (1)  1184  (1)  1189  (1)  1194  (1)  1199  (1)
                1204  (1)  1209  (1)  1214  (1)  1219  (1)  1229  (1)  1239  (1)  1244  (1)  1249  (1)
                1254  (1)  1269  (1)  1279  (1)  1284  (1)  1289  (1)  1294  (1)  1299  (1)  1304  (1)
                1315  (1)  1320  (1)  1325  (1)  1330  (1)  1335  (1)  1340  (1)  1345  (1)  1350  (1)
                1355  (1)  1360  (1)  1365  (1)  1370  (1)  1375  (1)  1380  (1)  1385  (1)  1390  (1)
                1395  (1)  1400  (1)  1405  (1)  1410  (1)  1415  (1)  1420  (1)  1425  (1)  1430  (1)
                1435  (1)  1440  (1)  1445  (1)  1450  (1)  1455  (1)  1460  (1)  1465  (1)  1470  (1)
                1475  (1)  1480  (1)  1485  (1)  1490  (1)  1495  (1)  1500  (1)  1505  (1)  1510  (1)
                1515  (1)  1520  (1)  1525  (1)  1530  (1)  1535  (1)  1540  (1)  1545  (1)  1550  (1)
                1555  (1)  1560  (1)  1565  (1)  1570  (1)  1575  (1)  1580  (1)  1585  (1)  1590  (1)
                1595  (1)  1600  (1)  1605  (1)  1610  (1)  1615  (1)  1620  (1)  1884  (1)  1887  (1)
                306   (1)  369   (1)  521   (1)  526   (1)  531   (1)  536   (1)  635   (1)  640   (1)
                673   (1)  678   (1)  683   (1)  688   (1)  698   (1)  703   (1)  713   (1)  728   (1)
                733   (1)  738   (1)  743   (1)  748   (1)  763   (1)  768   (1)  773   (1)  778   (1)
                783   (1)  788   (1)  803   (1)  808   (1)  813   (1)  818   (1)  823   (1)  828   (1)
                833   (1)  838   (1)  843   (1)  848   (1)  853   (1)  858   (1)  868   (1)  873   (1)
                878   (1)  883   (1)  888   (1)  893   (1)  908   (1)  913   (1)  918   (1)  924   (1)
                929   (1)  934   (1)  939   (1)  955   (1)  960   (1)  965   (1)  970   (1)  985   (1)
                990   (1)  995   (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	4	%-1950	(1)	%-1953	(1)	1956	(1)						
R0	6	%-1006	(1)	%-1017	(1)	%-1043	(1)	%-1049	(1)	%-662	(1)	%-919	(1)
R1	1	668	(1)										
R10	2	868	(1)	873	(1)								
R11	2	868	(1)	873	(1)								
R2	7	1884	(1)	1887	(1)	703	(1)	713	(1)	868	(1)	873	(1)
		934	(1)										
R3	7	1884	(1)	1887	(1)	703	(1)	713	(1)	868	(1)	873	(1)
		934	(1)										
R4	6	1884	(1)	703	(1)	713	(1)	868	(1)	873	(1)	934	(1)
R5	5	703	(1)	713	(1)	868	(1)	873	(1)	934	(1)		
R6	5	703	(1)	713	(1)	868	(1)	873	(1)	934	(1)		
R7	4	713	(1)	868	(1)	873	(1)	934	(1)				
R8	3	713	(1)	868	(1)	873	(1)						
R9	2	868	(1)	873	(1)								
SP	6	%-1950	(1)	%-1951	(1)	%-1952	(1)	1956	(1)	668	(1)		

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.21
Command processing	882	00:00:00.28	00:00:00.63
Pass 1	530	00:00:01.64	00:00:02.46
Symbol table sort	0	00:00:00.08	00:00:00.07
Pass 2	62	00:00:00.78	00:00:01.69
Symbol table output	2	00:00:00.06	00:00:00.17
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	9	00:00:00.41	00:00:00.65
Assembler run totals	1511	00:00:03.28	00:00:05.89

The working set limit was 2400 pages.
26937 bytes (53 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 332 non-local and 1 local symbols.
2003 source lines were read in Pass 1, producing 52 object records in Pass 2.
3 pages of virtual memory were used to define 3 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                2
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       1
TOTALS (all libraries)                                 3

18 GETS were required to define 3 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ENTRY.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

Table of contents

(1)   207   Libraries, Macros, Equated Symbols
(1)   234   Work Psect Declarations
(1)   256   Data Psect Declarations
(1)   490   DSX$ErrSoft Routine
(1)   542   DSX$ErrHard Routine
(1)   594   DSX$ErrPrep Routine
(1)   646   DSX$ErrDev Routine
(1)   701   DSX$ErrSys Routine
(1)   756   RRptError Routine
(1)   1006   DS_TestID Routine
(1)   1032   DS_ErrSup Routine
(1)   1158   DSR$Completion Routine

[17]

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ERROR.MAR
0000     2 ; *5     1-SEP-1987 13:31:10 MEIER "CHANGES TO DS"
0000     3 ; *4    30-MAR-1987 16:19:00 MARTIN "FIXED A COMMENT"
0000     4 ; *3    26-JAN-1987 08:06:00 GEARIN "PRV ADDED TO insufficient privilege ERROR MESSA
0000     5 ; *2    23-JAN-1987 15:47:05 GEARIN "CHANGED ERROR CODE MESSAGES"
0000     6 ; *1     6-FEB-1986 15:18:15 DS ""
0000     7 ; DEC/CMS REPLACEMENT HISTORY, Element ERROR.MAR
0000     8         .Title  ERROR   *** ERROR Error routines
0000     9         .Ident  /10-5/                                                  ;G:5
0000    10         .NoShow Conditionals
0000    11
0000    12 ;++
0000    13 ; Copyright (c) 1977, 1983, 1984, 1985, 1986, 1987                     ;G:2
0000    14 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    15 ;
0000    16 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    17 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    18 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    19 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    20 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    21 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    22 ; REMAIN IN DEC.
0000    23 ;
0000    24 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    25 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    26 ; CORPORATION.
0000    27 ;
0000    28 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    29 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    30 ;--
```

```
                        32 ;++
                        33 ; FACILITY:
                        34 ;
                        35 ;       VAX Diagnostic Supervisor
                        36 ;
                        37 ; ABSTRACT:
                        38 ;
                        39 ; ENVIRONMENT:
                        40 ;
                        41 ; AUTHOR:
                        42 ;
                        43 ;       Tom Soutter     16-NOV-77       VERSION 01
                        44 ;
                        45 ; MODIFIED BY:
                        46 ;                       TOM SOUTTER     08-DEC-77       VERSION 02 (ESSAA-3.05)
                        47 ;       01              DSPR #20:  "ERROR HALT AT ..." MESSAGE INSERTED.
                        48 ;
                        49 ;                       KEN CHAPMAN     30-DEC-77       VERSION 03 (ESSAA-3.06)
                        50 ;       02              PASS AP TO USER'S ERROR ROUTINE.
                        51 ;                       NICK HOWGATE    06-DEC-78       VERSION 04 (ESSAA-5.01)
                        52 ;       03              EMULATE CHARACTER STRING INSTRUCTIONS
                        53 ;       04              Removed DSA$V_ERR_LOOP, use DS$INLOOP instead
                        54 ;
                        55 ;                       Dave Butenhof, 10-sep-1980, Version 6.0
                        56 ;       05              Add error text to support RMS error codes used within
                        57 ;                       supervisor.
                        58 ;
                        59 ;                       Dave Butenhof, 30-sep-1980, Version 6.1
                        60 ;       06              Change forced word-relative references to longword-relative,
                        61 ;                       since supervisor has grown.
                        62 ;
                        63 ;       07              - Jack Stansbury, 21-Oct-1981, Version 6.-
                        64 ;                       Added calls to QA$MAIN for the QA enhancement.
                        65 ;                       Also added .LIBRARY statements for $DS and $DIAG.
                        66 ;                       Also changed some of the error messages to lower case.
                        67 ;
                        68 ;       08              - Jack Stansbury, 22-Oct-1981, Version 6.-
                        69 ;                       Fixed truncation errors.
                        70 ;
                        71 ;       09              - Jack Stansbury, 12-Nov-1981, Version 6.-
                        72 ;                       Took out references to HALTD and HALTI. Changed them to HALT.
                        73 ;
                        74 ;       10              - Dave Butenhof, 17-Nov-1981, Version 6.5
                        75 ;                       Add 'end of error' message, and typecodes
                        76 ;
                        77 ;       11              - Jack Stansbury, 21-Nov-1981, Version 6.5
                        78 ;                       Commented out several of the calls to QA_MAIN because it
                        79 ;                       appears as though they will not be needed for the 6.5 DS version.
                        80 ;
                        81 ;       12              - Dave Butenhof, 25-Nov-1981, version 6.5
                        82 ;                       Added error text for error '54', 'Fatal controller error',
                        83 ;                       as it is appallingly common.
                        84 ;
                        85 ;       13              - Dave Butenhof, 15-Dec-1981, Version 6.6
                        86 ;                       Fix error trailer message to only type out if IE1 isn't
                        87 ;                       set.
                        88 ;
```

```
                                ;       14      - Jack Stansbury, 17-Dec-1981, Version ?.?
                                ;               Added all the registers to the register masks on each of the
                                ;               error routines. These are for the QA routines.
                                ;
                                ;       15      - Dave Butenhof, 29-Dec-1981, Version 6.6
                                ;               Use new error counters for each class of error (hard,
                                ;               soft, etc.).
                                ;
                                ;       16      - Jack Stansbury, 25-Jan-1982, Version 6.6
                                ;               Discovered that the error number is truncated to a word in the
                                ;               RRprError error reporting routine even though the error number
                                ;               that is passed to the routine is a longword in length. This is
                                ;               most likely because of APT running on PDP-11's, which operate
                                ;               on word lengths. So, I changed one instruction to zero out the
                                ;               top word in DSA$GL_ERRNO when the error number is moved to this
                                ;               longword.
                                ;
                                ;       17      Marion Baggett, 2-apr-1982, Version 6.7
                                ;               added the PREPERR error routine for reporting errors caused by a
                                ;               testing starting and the hardware not being properly set up.
                                ;
                                ;       18      Marion Baggett, 9-Apr-1982, Version 6.7
                                ;               Change DS$PRINTF to DSX$PRINT for type coding
                                ;
                                ;       19      Jack Stansbury, 10-Apr-1982, Version 6.7
                                ;               Changed some of the BBC instructions to the new DS macros such
                                ;               as Br_If_ ....
                                ;
                                ;       20      Jack Stansbury, 11-May-1982, Version 6.8
                                ;               Added code to store the MsgAdr parameter from the $DS_ERRxxx
                                ;               macros into a location for use by the CRD routines.
                                ;
                                ;       21      Jack Stansbury, 21-Dec-1982, Version 6.11
                                ;               Printed out some more flags, locations, etc. in the ErrSup
                                ;               routine.
                                ;
                                ;       22      Jack Stansbury, 03-Mar-1983, Version 6.11
                                ;               Changed the code that sets the DS$GA_User_Error_Text so that
                                ;               this location is set ONLY when CRD is running. This MUST be
                                ;               true or else the CRD dispatch vectors will get screwed up.
                                ;               The condition was this: if a diagnostic was run that issued
                                ;               an error (prep, sys, dev, hard, or soft), then if CRD was run,
                                ;               and one of the diags run by CRD encountered a prep error, the
                                ;               path from the MEN$Preparation_Error_Text routine to the address
                                ;               of the user's error text would be lost, and an exception would
                                ;               result.
                                ;
                                ;       23      John Ciukaj     4-Apr-1983      Version 6.11
                                ;               - Changed all references to CRD Bits in DSA Flags
                                ;                 distinguishing between Online and Offline.
                                ;
                                ;       24      Bob Bergazzi    24-Apr-1984     Version 7.0
                                ;               Fixed edit # 21, FMTERRSUP1, which pushed contents instead of the
                                ;               address of DS$GT_VDS_VERSION on the stack.
                                ;
                                ;       25      Amy Iorio       18-Mar-1985     Version 8.2
                                ;               Put in hardware/firmware error reporting.
```

```
0000    146 ;
0000    147 ;       26      Ted Salem       8-April-1985    Version 8.2
0000    148 ;               Added interlocks to RRPTERROR and DS_ERRSUP routines
0000    149 ;
0000    150 ;       27      Ted Salem       10-April-1985   Version 8.2
0000    151 ;               Fixed bug in DS_ERRSUP routine.
0000    152 ;
0000    153 ;       28      Amy Iorio       17-May-1985     Version 8.2
0000    154 ;               Enhancement to Printrev; Patch revision level.
0000    155 ;
0000    156 ;       29      Amy Iorio       18-June-1985    Version 8.3
0000    157 ;               Fix to Printrev; actual revision level zero recognized.
0000    158 ;
0000    159 ;       30      Domenic Andella 12-July-1985    Version 8.3
0000    160 ;               Modified routine RRPTERROR to solve mp interlocking
0000    161 ;               code problem where the saved_FP's pc must be used
0000    162 ;               instead of the currect FP to compensate for additional
0000    163 ;               call frame generation.
0000    164 ;
0000    165 ;       31      Amy Iorio       15-July-1985    Version 8.3
0000    166 ;               Branch around revision levels if no prlink given
0000    167 ;               deleted so that revision levels print with or with out
0000    168 ;               prlink address.
0000    169 ;
0000    170 ;       32      Domenic Andella 7-August-1985   Version 8.3
0000    171 ;               Comment out modification 30 until interlocking
0000    172 ;               code is reinstated.
0000    173 ;
0000    174 ;       33      Ted Salem       9-Oct-1985      Version 9.0
0000    175 ;               Changed the format for the print service if the
0000    176 ;               revision level refers to harware.
0000    177 ;
0000    178 ;       34      Ted Salem       18_Nov_1985     Version 9.0
0000    179 ;               Fixed problem in RRPTERROR. Wrong log unit
0000    180 ;               number was being used to get device name .
0000    181 ;               Also, fixed incorrect checking of rev. type.
0000    182 ;
0000    183 ;       35      Domenic Andella 17-Dec-1985     Version 9.1
0000    184 ;               Modify routine RRPTERROR such that if cpu
0000    185 ;               is a Scorpio or Nautilus, frame pointer will be
0000    186 ;               altered to accomodate interlocking. See edit 30
0000    187 ;
0000    188 ;       36      M. Baggett      16-Jan-1986     Version 9.1
0000    189 ;               Added error message for invalid load for cpu specific
0000    190 ;               supervisor.
0000    191 ;
0000    192 ;       37      M. Baggett      17-Jan-1986     Version 9.1
0000    193 ;               Change error code from DS$_NOSUPPORT to DS$_UNSUPPDEV
0000    194 ;                                                                   ;G:2
0000    195 ;       3       David Gearin    23-Jan-1987     Version 10.6        ;G:5
0000    196 ;               Change error message for RMS$PRV from "Unknown error" to    ;G:2
0000    197 ;               "PRV, Insufficient privilege or file protection violation".  ;G:2
0000    198 ;                                                                   ;G:4
0000    199 ;       4       Ingrid Martin   30-March-1987   Version 10.7        ;G:5
0000    200 ;               Just updated a comment:  BR_IF_MP checks if the machine     ;G:4
0000    201 ;               is Scorpio, Nautilus, RRR or LLL.                   ;G:4
0000    202 ;                                                                   ;G:5
```

```
                                0000   203 ;       5       Stephen Meier   1-Sept-1987     Version 10.10                 ;G:5
                                0000   204 ;               Recompiled to include changes to BR_IF_MP                     ;G:5
                                0000   205 ;--
```

```
                     0000   207          .SBTTL  Libraries, Macros, Equated Symbols
                     0000   208 ;
                     0000   209 ; INCLUDE FILES:
                     0000   210 ;
                     0000   211          .LIBRARY        /SYS$LIBRARY:LIB/        ;                    [17]
                     0000   212          .LIBRARY        /$DS/                    ;                    [07]
                     0000   213          .LIBRARY        /$DIAG/                  ;                    [07]
                     0000   214
                     0000   215 ;
                     0000   216 ; EQUATED SYMBOLS:
                     0000   217 ;
                     0000   218
00000003             0000   219          BPTCODE     = 03                    ; BREAKPOINT OPCODE
00000020             0000   220          TESTID_LEN = 32                     ; TEXT BUFFER FOR TEST/SUBTEST MESSAGE
                     0000   221
                     0000   222          APTDEF                              ; APT MESSAGE CODE DEFINITIONS
                     0000   223          DSFDEF                              ; CONTROL FLAG DEFINITIONS
                     0000   224          DSQA                                ; QA routine name constants          [07]
                     0000   225          $DS_CFDEF
                     0000   226          $DS_DSDEF
                     0000   227          $DS_DSADEF                          ; CONTROL FLAG DEFINITIONS
                     0000   228          $DS_ERRDEF
                     0000   229          $DS_HDRDEF                          ; DIAGNOSTIC PROGRAM HEADER SYMBOLS
                     0000   230          $DS_HPODEF                          ; P-TABLE OFFSET DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             FE70                .=0
                     0000                .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
00000008             0000                .IIF    NB,..BLKQ,      .BLKQ   1
                     0008                .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
0000000A             0008                .IIF    NB,..BLKW,      .BLKW   1
                     000A                .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
0000000B             000A                .IIF    NB,..BLKB,      .BLKB   1
                     000B                .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
0000000C             000B                .IIF    NB,..BLKB,      .BLKB   1
                     000C                .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
00000018             000C                .IIF    NB,..BLKB,      .BLKB   12
                     0018                .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
0000001C             0018                .IIF    NB,..BLKL,      .BLKL   1
                     001C                .IIF    NB,HP$A_DVA,    HP$A_DVA:
00000020             001C                .IIF    NB,..BLKL,      .BLKL   1
                     0020                .IIF    NB,HP$A_LINK,   HP$A_LINK:
00000024             0020                .IIF    NB,..BLKL,      .BLKL   1
                     0024                .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
00000026             0024                .IIF    NB,..BLKW,      .BLKW   1
                     0026                .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
00000032             0026                .IIF    NB,..BLKB,      .BLKB   12
                     0032                .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                     0000                .RESTORE
                     0000   231          $RMSDEF                             ; RMS return code definitions
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             FE70                .=0
                     0000                .RESTORE
                     0000   232          $ds_typedef                         ; Define error typecodes             [10]
```

```
                                0000   234          .Subtitle       Work Psect Declarations
                                0000   235 ;
                                0000   236 ; OWN STORAGE:
                                0000   237 ;
                                0000   238
                             00000000  239          .Psect  Work, Noshr, Noexe, Wrt, Long
                                0000   240
                                0000   241 Q_BUFQWD:                                ; QUADWORD DESCRIPTOR
                    00000008    0000   242          .BLKQ   1                       ;  FOR MISC. STRINGS
                                0008   243 DS$GQ_TESTID::
                    00000010    0008   244          .BLKQ   1                       ; STRING LENGTH & POINTER
                                0010   245 L_OFFSET:                               ; OFFSET INTO LOGICAL UNIT DATA [25]
                    00000014    0010   246          .BLKL   1
                                0014   247 REVERROR:                               ; HOLDS REVISION TYPE [25]
                    00000018    0014   248          .BLKL   1
                                0018   249 PRINT:                                  ; FLAG TO PRINT MESSAGE [28]
                    0000001C    0018   250          .BLKL   1
                                001C   251 MODULE:                                 ; HOLDS MODULE NUMBER TO PRINT[ATI]
                    00000020    001C   252          .BLKL   1
                                0020   253 T_TESTID:
                    00000040    0020   254          .BLKB   TESTID_LEN              ; BUFFER FOR TEST/SUBTEST MESSAGE
```

```
                                    0040   256          .Subtitle       Data Psect Declarations
                                00000000   257          .Psect  Data, Shr, Noexe, Nowrt, Byte
                                    0000   258
                                    0000   259 MSGBELL:
                                07  0000   260          .ASCII  <7>                     ; ASCII "BELL"
                                    0001   261 HARDER:
                                06  0001   262          .byte   ds$k_type_errhard       ; Type code byte                [10]
   72 6F 72 72 65 20 64 72 61 48 00' 0002   263          .ASCIC  .Hard error.
                                0A  0002
                                    000D   264 SOFTER:
                                07  000D   265          .byte   ds$k_type_errsoft       ; Type code byte                [10]
   72 6F 72 72 65 20 74 66 6F 53 00' 000E   266          .ASCIC  .Soft error.
                                0A  000E
                                    0019   267 DEVFER:
                                08  0019   268          .byte   ds$k_type_errdev        ; Type code byte                [10]
61 74 61 66 20 65 63 69 76 65 44 00' 001A   269          .ASCIC  .Device fatal error.    ;                               [10]
                72 6F 72 72 65 20 6C  0026
                                12  001A
                                    002D   270 PREPER:
                                1B  002D   271          .byte   ds$K_type_errprep       ; Type code byte                [17]
20 74 6F 6E 20 65 63 69 76 65 44 00' 002E   272          .ASCIC  .Device not prepared error.     ;                       [17]
72 72 65 20 64 65 72 61 70 65 72 70  003A
                            72 6F    0046
                                19  002E
                                    0048   273 SYSFER:
                                05  0048   274          .byte   ds$k_type_errsys        ; Type code byte                [10]
61 74 61 66 20 6D 65 74 73 79 53 00' 0049   275          .ASCIC  .System fatal error.    ;                               [10]
                72 6F 72 72 65 20 6C  0055
                                12  0049
                                    005C   276 ersuper:                                 ; Supervisor error text         [10]
65 64 20 65 72 61 77 74 66 6F 53 00' 005C   277          .ascic  .Software detected error.       ;                       [10]
72 6F 72 72 65 20 64 65 74 63 65 74  0068
                                17  005C
                                    0074   278 ANULL:
                                00' 0074   279          .ASCIC  ..
                                00  0074
                                    0075   280 QUNKDEV:
77 6F 6E 6B 6E 55 0000007D'010E0000' 0075   281          .ASCID  .Unknown device.
            65 63 69 76 65 64 20 6E  0083
                                    008B   282 QINITSEC:
61 69 74 69 6E 49 00000093'010E0000' 008B   283          .ASCID  .Initialization section.
63 65 73 20 6E 6F 69 74 61 7A 69 6C  0099
                        6E 6F 69 74  00A5
                                    00A9   284 QCLEANSEC:
75 6E 61 65 6C 43 000000B1'010E0000' 00A9   285          .ASCID  .Cleanup section.
         6E 6F 69 74 63 65 73 20 70  00B7
                                    00C0   286 HARDREV:                                ;                               [25][33]
20 3D 65 72 61 77 64 72 61 48 20 00' 00C0   287          .ASCIC " Hardware= !AC;"
                        3B 43 41 21  00CC
                                0F  00C0
                                    00D0   288 FIRMREV:                                ;                               [25]
20 3D 65 72 61 77 6D 72 69 46 20 00' 00D0   289          .ASCIC " Firmware= !ZL;"
                        3B 4C 5A 21  00DC
                                0F  00D0
                                    00E0   290 PATCHREV:                               ;                               [28]
4C 5A 21 20 3D 68 63 74 61 50 20 00' 00E0   291          .ASCIC " Patch= !ZL;"
                                3B  00EC
```

```
                                 0C  00E0
                                     00ED   292 QFMTTESTID:
21 20 74 73 65 74 000000F5'010E0000' 00ED   293          .ASCID  .test !UL, subtest !UL.
20 74 73 65 74 62 75 73 20 2C 4C 55  00FB
                              4C 55 21  0107
```

```
                                        010A   295 FMTERRHLT:
6F 20 74 6C 61 48 20 2E 2E 2F 21 00'    010A   296         .ASCIC  "!/.. Halt on error at PC !XL(X)!/"     ;                    [07]
50 20 74 61 20 72 6F 72 72 65 20 6E     0116
      2F 21 29 58 28 4C 58 21 20 43     0122
                                  21    010A
                                        012C   297 FMTERRHDR:
20 2A 2A 2A 2A 2A 2A 2A 2A 2F 21 00'    012C   298         .ASCIC  \!/********  !AC - !UL.!UL  ********\ -
21 2E 4C 55 21 20 2D 20 43 41 21 20     0138
2A 2A 2A 2A 2A 2A 2A 2A 20 20 4C 55     0144
2C 4C 55 21 20 73 73 61 50 20 2F 21     0150   299                 "!/ Pass !UL, !AS, error !UL, !%D"-     ;                    [07]
20 72 6F 72 72 65 20 2C 53 41 21 20     015C
            44 25 21 20 2C 4C 55 21     0168
65 6C 69 68 77 20 43 41 21 20 2F 21     0170   300                 "!/ !AC while testing !AS: !AC !/"              ;
53 41 21 20 67 6E 69 74 73 65 74 20     017C
            2F 21 20 43 41 21 20 3A     0188
                                  63    012C
                                        0190   301 linefeed:                                                ;                    [28]
                            2F 21 00'   0190   302         .ASCIC  \!/\
                                  02    0190
                                        0193   303 FMTREVERROR:                                            ;                    [25]
20 6E 6F 69 73 69 76 65 52 2F 21 00'    0193   304         .ASCIC  \!/Revision level(s) of !AS:!/\
20 66 6F 20 29 73 28 6C 65 76 65 6C     019F
                  2F 21 3A 53 41 21     01AB
                                  1D    0193
                                        01B1   305 FMTMODULE:
   3A 29 43 41 21 28 09 09 2F 21 00'    01B1   306         .ASCIC  \!/              (!AC):\
                                  0A    01B1
                                        01BC   307 FMTNO_MOD:
20 20 20 20 20 20 20 09 09 2F 21 00'    01BC   308         .ASCIC  \!/                      \
                                  20    01C8
                                  0C    01BC
                                        01C9   309 fmtenderror:
20 2A 2A 2A 2A 2A 2A 2A 2A 2F 21 00'    01C9   310         .ascic  \!/********  End of !AC number !UL ********!/!/\        ;       [10]
20 43 41 21 20 66 6F 20 64 6E 45 20     01D5
2A 20 4C 55 21 20 72 65 62 6D 75 6E     01E1
   2F 21 2F 21 2A 2A 2A 2A 2A 2A 2A     01ED
                                  2E    01C9
                                        01F8   311 FMTERRSUP:
61 77 74 66 6F 53 20 3F 3F 2F 21 00'    01F8   312         .ASCIC  '!/?? Software detected error in "!AC", error #!UL, !%D'-  ;    [07]
20 64 65 74 63 65 74 65 64 20 65 72     0204
41 21 22 20 6E 69 20 72 6F 72 72 65     0210
21 23 20 72 6F 72 72 65 20 2C 22 43     021C
               44 25 21 20 2C 4C 55     0228
            2F 21 43 41 21 20 2F 21     022F   313                 "!/ !AC!/"
                                  3E    01F8
                                        0237   314 FmtErrSup1:                                            ;                    [21]
67 61 6C 46 5F 4C 47 24 41 53 44 00'    0237   315         .Ascic  'DSA$GL_Flags: !XL(X)  DS$GL_Flags: !XL(X)  Version: !AC!/' ;   [21]
44 20 20 29 58 28 4C 58 21 20 3A 73     0243
20 3A 73 67 61 6C 46 5F 4C 47 24 53     024F
73 72 65 56 20 20 29 58 28 4C 58 21     025B
      2F 21 43 41 21 20 3A 6E 6F 69     0267
                                  39    0237
                                        0271   316
                                        0271   317 FMTERRSUP2:
20 20 4C 58 21 20 3A 43 50 5F 21 00'    0271   318         .ASCIC  "!_PC: !XL  PSL: !XL!/"-
      2F 21 4C 58 21 20 3A 4C 53 50     027D
52 20 20 4C 58 21 20 3A 30 52 5F 21     0287   319                 "!_R0: !XL  R1:  !XL  R2:  !XL  R3:  !XL!/"-
```

```
3A 32 52 20 20 4C 58 21 20 20 3A 31   0293
20 20 3A 33 52 20 20 4C 58 21 20 20   029F
                  2F 21 4C 58 21      02AB
52 20 20 4C 58 21 20 3A 34 52 5F 21   02B0    320                 "!_R4: !XL  R5:  !XL  R6:  !XL  R7:  !XL!/"-
3A 36 52 20 20 4C 58 21 20 20 3A 35   02BC
20 20 3A 37 52 20 20 4C 58 21 20 20   02C8
                  2F 21 4C 58 21      02D4
52 20 20 4C 58 21 20 3A 38 52 5F 21   02D9    321                 "!_R8: !XL  R9:  !XL  R10: !XL  R11: !XL!/"
30 31 52 20 20 4C 58 21 20 20 3A 39   02E5
20 3A 31 31 52 20 20 4C 58 21 20 3A   02F1
                  2F 21 4C 58 21      02FD
                              90      0271
                                      0302    322 FMTERRSUP3:
39 28 37 21 4C 58 21 3A 4C 58 21 00'  0302    323         .ASCIC  "!XL:!XL!7(9XL)!/"
                  2F 21 29 4C 58      030E
                              10      0302
```

```
                                0313   325 T_ERROR_MASK:
21 2D 44 41 21 2D 43 41 21 2F 21 00' 0313   326          .ASCIC  "!/!AC-!AD-!AC (R0=!XL)!/"
21 29 4C 58 21 3D 30 52 28 20 43 41  031F
                                2F   032B
                                18   0313
                                     032C   327 UTILITY_CODES:
                          00000004' 032C   328          .LONG   <10$-.>/4
                          00000000  0330   329          .LONG   0
                          00000001  0334   330          .LONG   1
                          00000066  0338   331          .LONG   ^X66
                                     033C   332 10$:
                                     033C   333 UTILITY_TEXT:
                               0000  033C   334          .WORD   0
                               0006' 033E   335          .WORD   10$-.
                               000C' 0340   336          .WORD   20$-.
                               000F' 0342   337          .WORD   30$-.
                  4D 45 54 53 59 53 25 00' 0344   338 10$:    .ASCIC  "%SYSTEM"
                                07   0344
                     53 4D 52 25 00' 034C   339 20$:    .ASCIC  "%RMS"
                                04   034C
                  47 41 49 44 25 00' 0351   340 30$:    .ASCIC  "%DIAG"
                                05   0351
                                     0357   341
                                     0357   342 ERROR_CODES:
                          0000002F' 0357   343          .LONG   <10$-.>/4
                          00000000' 035B   344          .LONG   SS$_ACCVIO          ; Access violation
                          00000000' 035F   345          .LONG   SS$_NOSUCHDEV       ; No such device
                          00000000' 0363   346          .LONG   SS$_NOSUCHFILE      ; No such file
                          00000000' 0367   347          .long   ss$_ctrlerr         ; Fatal controller error          [12]
                          00018298  036B   348          .LONG   RMS$_PRV&^XFFFFFFF8 ; RMS insufficient privilege      [38]
                          00018290  036F   349          .LONG   RMS$_FNF&^XFFFFFFF8 ; RMS no such file
                          000184C0  0373   350          .LONG   RMS$_DEV&^XFFFFFFF8 ; RMS bad device
                          0001C048  0377   351          .LONG   RMS$_DNF&^XFFFFFFF8 ; RMS directory not found
                          0001C000  037B   352          .LONG   RMS$_ACC&^XFFFFFFF8 ; RMS file access error
                          00018490  037F   353          .LONG   RMS$_CCR&^XFFFFFFF8 ; RMS connect error
                          000184D0  0383   354          .LONG   RMS$_DME&^XFFFFFFF8 ; RMS pool allocation error
                          00018278  0387   355          .LONG   RMS$_EOF&^XFFFFFFF8 ; RMS end of file
                          00018508  038B   356          .LONG   RMS$_FAB&^XFFFFFFF8 ; RMS invalid FAB
                          00018560  038F   357          .LONG   RMS$_IFI&^XFFFFFFF8 ; RMS invalid internal file index
                          00018570  0393   358          .LONG   RMS$_IOP&^XFFFFFFF8 ; RMS illegal operation
                          00018580  0397   359          .LONG   RMS$_ISI&^XFFFFFFF8 ; RMS invalid RAB/FAB pair
                          00018638  039B   360          .LONG   RMS$_RAB&^XFFFFFFF8 ; RMS invalid RAB
                          0001C0F0  039F   361          .LONG   RMS$_RER&^XFFFFFFF8 ; RMS file read error
                          000181A8  03A3   362          .LONG   RMS$_RTB&^XFFFFFFF8 ; RMS record truncation warning
                          00018658  03A7   363          .LONG   RMS$_RFA&^XFFFFFFF8 ; RMS invalid RFA
                          00018578  03AB   364          .LONG   RMS$_IRC&^XFFFFFFF8 ; RMS invalid record
                          00018528  03AF   365          .LONG   RMS$_FNM&^XFFFFFFF8 ; FNM error in filename
                          00660008  03B3   366          .LONG   DS$_OVERFLOW        ; Field overflow
                          00660010  03B7   367          .LONG   DS$_NULLSTR         ; Null input string
                          00660018  03BB   368          .LONG   DS$_ILLCHAR         ; Illegal character
                          00660020  03BF   369          .LONG   DS$_PROGERR         ; Programmer error
                          00660028  03C3   370          .LONG   DS$_TRUNCATE        ; Data truncated
                          00660038  03C7   371          .LONG   DS$_IVVECT          ; Invalid vector
                          00660040  03CB   372          .LONG   DS$_IVADDR          ; Invalid vector address
                          00660048  03CF   373          .LONG   DS$_VASFULL         ; virtural address space full
                          00660050  03D3   374          .LONG   DS$_INSFMEM         ; Insufficient memory
                          00660060  03D7   375          .LONG   DS$_IHWE            ; Initial channel hardware error
```

```
00660068  03DB   376          .LONG   DS$_FHWE                ; Final channel hardware error
00660070  03DF   377          .LONG   DS$_LOGIC               ; Program logic error
00660078  03E3   378          .LONG   DS$_ILLPAGCNT           ; Illegal page count
00660080  03E7   379          .LONG   DS$_FRAGBUF             ; Buffer is fragmented
00660088  03EB   380          .LONG   DS$_MCHK                ; Machine check
00660090  03EF   381          .LONG   DS$_KRNLSTK             ; Kernel stack not valid
00660098  03F3   382          .LONG   DS$_POWER               ; Power fail interrupt
006600A0  03F7   383          .LONG   DS$_TRANSL              ; Translation not valid
006600B0  03FB   384          .LONG   DS$_NOTIMP              ; Not implemented
006600B8  03FF   385          .LONG   DS$_IPL2HI              ; IPL too high
006600C0  0403   386          .LONG   DS$_ICERR               ; Interval clock error
006600D0  0407   387          .LONG   DS$_ARITH               ; Arithmetic trap
006600D8  040B   388          .LONG   DS$_UNEXPINT            ; Unexpected interrupt
00660158  040F   389          .LONG   DS$_UNSUPPDEV           ; load device not support under this cpu[27]
          0413   390 10$:
          0413   391 ERROR_TEXT:
005E'     0413   392          .WORD   0$-.                     ; Unknown error
00F8'     0415   393          .WORD   60$-.                    ; ^X0C Access violation
0068'     0417   394          .WORD   10$-.
0080'     0419   395          .WORD   20$-.
04CC'     041B   396          .word   440$-.                   ; Fatal controller error                [12]
051E'     041D   397          .WORD   510$-.                   ;                                       [38]
0093'     041F   398          .WORD   30$-.
00A5'     0421   399          .WORD   40$-.
00D1'     0423   400          .WORD   50$-.
02C6'     0425   401          .WORD   300$-.
02E0'     0427   402          .WORD   310$-.
02F6'     0429   403          .WORD   320$-.
0312'     042B   404          .WORD   330$-.
032A'     042D   405          .WORD   340$-.
034F'     042F   406          .WORD   350$-.
037F'     0431   407          .WORD   360$-.
03B4'     0433   408          .WORD   370$-.
03E6'     0435   409          .WORD   380$-.
040B'     0437   410          .WORD   390$-.
041E'     0439   411          .WORD   400$-.
0444'     043B   412          .WORD   410$-.
046B'     043D   413          .WORD   420$-.
0491'     043F   414          .WORD   430$-.
00E2'     0441   415          .WORD   70$-.
00EF'     0443   416          .WORD   80$-.
00FF'     0445   417          .WORD   90$-.
010F'     0447   418          .WORD   100$-.
011E'     0449   419          .WORD   110$-.
012B'     044B   420          .WORD   120$-.
0138'     044D   421          .WORD   130$-.
014D'     044F   422          .WORD   140$-.
0167'     0451   423          .WORD   150$-.
0179'     0453   424          .WORD   160$-.
0196'     0455   425          .WORD   170$-.
01B1'     0457   426          .WORD   180$-.
01C3'     0459   427          .WORD   190$-.
01D4'     045B   428          .WORD   200$-.
01E7'     045D   429          .WORD   210$-.
01F3'     045F   430          .WORD   220$-.
0208'     0461   431          .WORD   230$-.
021B'     0463   432          .WORD   240$-.
```

```
                                022F' 0465    433          .WORD   250$-.
                                023D' 0467    434          .WORD   260$-.
                                0248' 0469    435          .WORD   270$-.
                                025B' 046B    436          .WORD   280$-.
                                0269' 046D    437          .WORD   290$-.
                                0498' 046F    438          .WORD   500$-.
72 72 65 20 6E 77 6F 6E 6B 6E 55 00' 0471    439 0$:      .ASCIC  "Unknown error"
                                72 6F 047D
                                   0D 0471
20 2C 56 45 44 48 43 55 53 4F 4E 00' 047F    440 10$:     .ASCIC  "NOSUCHDEV, No such device"
69 76 65 64 20 68 63 75 73 20 6F 4E  048B
                                65 63  0497
                                   19  047F
2C 45 4C 49 46 48 43 55 53 4F 4E 00' 0499    441 20$:     .ASCIC  "NOSUCHFILE, No such file"
6C 69 66 20 68 63 75 73 20 6F 4E 20  04A5
                                   65  04B1
                                   18  0499
6E 20 65 6C 69 66 20 2C 46 4E 46 00' 04B2    442 30$:     .ASCIC  "FNF, file not found"
            64 6E 75 6F 66 20 74 6F  04BE
                                   13  04B2
65 64 20 64 61 62 20 2C 56 45 44 00' 04C6    443 40$:     .ASCIC  "DEV, bad device, or inappropriate device type"
61 6E 69 20 72 6F 20 2C 65 63 69 76  04D2
64 20 65 74 61 69 72 70 6F 72 70 70  04DE
      65 70 79 74 20 65 63 69 76 65  04EA
                                   2D  04C6
74 63 65 72 69 64 20 2C 46 4E 44 00' 04F4    444 50$:     .ASCIC  "DNF, directory not found"
6E 75 6F 66 20 74 6F 6E 20 79 72 6F  0500
                                   64  050C
                                   18  04F4
73 73 65 63 63 61 20 2C 43 43 41 00' 050D    445 60$:     .ASCIC  "ACC, access violation"
      6E 6F 69 74 61 6C 6F 69 76 20  0519
                                   15  050D
66 72 65 76 6F 20 64 6C 65 69 46 00' 0523    446 70$:     .ASCIC  "Field overflow"
                                77 6F 6C  052F
                                   0E  0523
20 74 75 70 6E 69 20 6C 6C 75 4E 00' 0532    447 80$:     .ASCIC  "Null input string"
                  67 6E 69 72 74 73  053E
                                   11  0532
61 68 63 20 6C 61 67 65 6C 6C 49 00' 0544    448 90$:     .ASCIC  "Illegal character"
                  72 65 74 63 61 72  0550
                                   11  0544
20 72 65 6D 6D 61 72 67 6F 72 50 00' 0556    449 100$:    .ASCIC  "Programmer error"
                     72 6F 72 72 65  0562
                                   10  0556
61 63 6E 75 72 74 20 61 74 61 44 00' 0567    450 110$:    .ASCIC  "Data truncated"
                              64 65 74  0573
                                   0E  0567
63 65 76 20 64 69 6C 61 76 6E 49 00' 0576    451 120$:    .ASCIC  "Invalid vector"
                              72 6F 74  0582
                                   0E  0576
63 65 76 20 64 69 6C 61 76 6E 49 00' 0585    452 130$:    .ASCIC  "Invalid vector address"
   73 73 65 72 64 64 61 20 72 6F 74  0591
                                   16  0585
64 61 20 6C 61 72 75 74 72 69 56 00' 059C    453 140$:    .ASCIC  "Virtural address space full"
20 65 63 61 70 73 20 73 73 65 72 64  05A8
                        6C 6C 75 66  05B4
                                   1B  059C
```

```
6E 65 69 63 69 66 66 75 73 6E 49 00' 05B8    454 150$:   .ASCIC  "Insufficient memory"
               79 72 6F 6D 65 6D 20 74  05C4
                                  13  05B8
61 68 63 20 6C 61 69 74 69 6E 49 00' 05CC    455 160$:   .ASCIC  "Initial channel hardware error"
72 61 77 64 72 61 68 20 6C 65 6E 6E  05D8
                  72 6F 72 72 65 20 65  05E4
                                  1E  05CC
6E 6E 61 68 63 20 6C 61 6E 69 46 00' 05EB    456 170$:   .ASCIC  "Final channel hardware error"
20 65 72 61 77 64 72 61 68 20 6C 65  05F7
                     72 6F 72 72 65  0603
                                  1C  05EB
67 6F 6C 20 6D 61 72 67 6F 72 50 00' 0608    457 180$:   .ASCIC  "Program logic error"
               72 6F 72 72 65 20 63 69  0614
                                  13  0608
67 61 70 20 6C 61 67 65 6C 6C 49 00' 061C    458 190$:   .ASCIC  "Illegal page count"
                  74 6E 75 6F 63 20 65  0628
                                  12  061C
66 20 73 69 20 72 65 66 66 75 42 00' 062F    459 200$:   .ASCIC  "Buffer is fragmented"
            64 65 74 6E 65 6D 67 61 72  063B
                                  14  062F
65 68 63 20 65 6E 69 68 63 61 4D 00' 0644    460 210$:   .ASCIC  "Machine check"
                                  6B 63  0650
                                  0D  0644
63 61 74 73 20 6C 65 6E 72 65 4B 00' 0652    461 220$:   .ASCIC  "Kernel stack not valid"
   64 69 6C 61 76 20 74 6F 6E 20 6B  065E
                                  16  0652
20 6C 69 61 66 20 72 65 77 6F 50 00' 0669    462 230$:   .ASCIC  "Power fail interrupt"
            74 70 75 72 72 65 74 6E 69  0675
                                  14  0669
6E 6F 69 74 61 6C 73 6E 61 72 54 00' 067E    463 240$:   .ASCIC  "Translation not valid"
      64 69 6C 61 76 20 74 6F 6E 20  068A
                                  15  067E
65 6D 65 6C 70 6D 69 20 74 6F 4E 00' 0694    464 250$:   .ASCIC  "Not implemented"
                        64 65 74 6E  06A0
                                  0F  0694
67 69 68 20 6C 6F 74 20 4C 50 49 00' 06A4    465 260$:   .ASCIC  "IPL too high"
                                  68  06B0
                                  0C  06A4
6C 63 20 6C 61 76 72 65 74 6E 49 00' 06B1    466 270$:   .ASCIC  "Interval clock error"
            72 6F 72 72 65 20 6B 63 6F  06BD
                                  14  06B1
20 63 69 74 65 6D 68 74 69 72 41 00' 06C6    467 280$:   .ASCIC  "Arithmetic trap"
                        70 61 72 74  06D2
                                  0F  06C6
20 64 65 74 63 65 70 78 65 6E 55 00' 06D6    468 290$:   .ASCIC  "Unexpected interrupt"
            74 70 75 72 72 65 74 6E 69  06E2
                                  14  06D6
69 66 20 50 43 41 20 2C 43 43 41 00' 06EB    469 300$:   .ASCIC  "ACC, ACP file access failed"
61 66 20 73 73 65 63 63 61 20 65 6C  06F7
                        64 65 6C 69  0703
                                  1B  06EB
74 6F 6E 6E 61 63 20 2C 52 43 43 00' 0707    470 310$:   .ASCIC  "CCR, cannot connect RAB"
42 41 52 20 74 63 65 6E 6E 6F 63 20  0713
                                  17  0707
69 6D 61 6E 79 64 20 2C 45 4D 44 00' 071F    471 320$:   .ASCIC  "DME, dynamic memory exhausted"
68 78 65 20 79 72 6F 6D 65 6D 20 63  072B
                     64 65 74 73 75 61  0737
```

```
                                                  1D  071F
66 6F 20 64 6E 65 20 2C 46 4F 45  00' 073D    472 330$:   .ASCIC  "EOF, end of file detected"
74 63 65 74 65 64 20 65 6C 69 66  20  0749
                                  64 65  0755
                                  19  073D
69 6C 61 76 6E 69 20 2C 42 41 46  00' 0757    473 340$:   .ASCIC  "FAB, invalid FAB or FAB not accessible"
42 41 46 20 72 6F 20 42 41 46 20  64  0763
69 73 73 65 63 63 61 20 74 6F 6E  20  076F
                                  65 6C 62  077B
                                  26  0757
69 6C 61 76 6E 69 20 2C 49 46 49  00' 077E    474 350$:   .ASCIC  "IFI, invalid internal file identifier (IFI) value"
66 20 6C 61 6E 72 65 74 6E 69 20  64  078A
69 66 69 74 6E 65 64 69 20 65 6C  69  0796
6C 61 76 20 29 49 46 49 28 20 72  65  07A2
                                  65 75  07AE
                                  31  077E
74 61 72 65 70 6F 20 2C 50 4F 49  00' 07B0    475 360$:   .ASCIC  "IOP, operation invalid for file organization or device"
20 64 69 6C 61 76 6E 69 20 6E 6F  69  07BC
67 72 6F 20 65 6C 69 66 20 72 6F  66  07C8
72 6F 20 6E 6F 69 74 61 7A 69 6E  61  07D4
                  65 63 69 76 65  64  07E0
                                  36  07B0
69 6C 61 76 6E 69 20 2C 49 53 49  00' 07E7    476 370$:   .ASCIC  "ISI, invalid internal stream identifier (ISI) value"
73 20 6C 61 6E 72 65 74 6E 69 20  64  07F3
69 74 6E 65 64 69 20 6D 61 65 72  74  07FF
76 20 29 49 53 49 28 20 72 65 69  66  080B
                              65 75 6C  61  0817
                                  33  07E7
69 6C 61 76 6E 69 20 2C 42 41 52  00' 081B    477 380$:   .ASCIC  "RAB, invalid RAB or RAB not accessible"
42 41 52 20 72 6F 20 42 41 52 20  64  0827
69 73 73 65 63 63 61 20 74 6F 6E  20  0833
                                  65 6C 62  083F
                                  26  081B
72 20 65 6C 69 66 20 2C 52 45 52  00' 0842    478 390$:   .ASCIC  "RER, file read error"
         72 6F 72 72 65 20 64 61  65  084E
                                  14  0842
64 72 6F 63 65 72 20 2C 42 54 52  00' 0857    479 400$:   .ASCIC  "RTB, record too large for user's buffer"
66 20 65 67 72 61 6C 20 6F 6F 74  20  0863
75 62 20 73 27 72 65 73 75 20 72  6F  086F
                              72 65 66  66  087B
                                  27  0857
69 6C 61 76 6E 69 20 2C 41 46 52  00' 087F    480 410$:   .ASCIC  "RFA, invalid record's file address (RFA)"
66 20 73 27 64 72 6F 63 65 72 20  64  088B
20 73 73 65 72 64 64 61 20 65 6C  69  0897
                        29 41 46  52  28  08A3
                                  28  087F
61 67 65 6C 6C 69 20 2C 43 52 49  00' 08A8    481 420$:   .ASCIC  "IRC, illegal record encountered in file"
63 6E 65 20 64 72 6F 63 65 72 20  6C  08B4
20 6E 69 20 64 65 72 65 74 6E 75  6F  08C0
                              65 6C 69  66  08CC
                                  27  08A8
20 72 6F 72 72 65 20 2C 4D 4E 46  00' 08D0    482 430$:   .ASCIC  "FNM, error in filename"
   65 6D 61 6E 65 6C 69 66 20 6E  69  08DC
                                  16  08D0
61 66 20 2C 52 52 45 4C 52 54 43  00' 08E7    483 440$:   .ascic  "CTRLERR, fatal controller error"       ;                    [12]
6C 6C 6F 72 74 6E 6F 63 20 6C 61  74  08F3
            72 6F 72 72 65 20 72  65  08FF
```

```
                                        1F  08E7
65 64 20 64 61 6F 6C 20 65 68 54 00'    0907    484 500$:   .ASCIC  "The load device is not supported in this Supervisor"   ;       [37]
20 74 6F 6E 20 73 69 20 65 63 69 76     0913
6E 69 20 64 65 74 72 6F 70 70 75 73     091F
76 72 65 70 75 53 20 73 69 68 74 20     092B
                     72 6F 73 69        0937
                                 33     0907
66 66 75 73 6E 49 20 2C 56 52 50 00'    093B    485 510$:   .ASCIC  "PRV, Insufficient privilege or file protection violation"  ;   [38]
69 76 69 72 70 20 74 6E 65 69 63 69     0947
65 6C 69 66 20 72 6F 20 65 67 65 6C     0953
20 6E 6F 69 74 63 65 74 6F 72 70 20     095F
         6E 6F 69 74 61 6C 6F 69 76     096B
                                 38     093B
                                        0974    486
                                        0974    487 SEVERITY:
                        3F 46 45 57     0974    488         .ASCII  "WEF?"
```

```
                     0978   490          .SBTTL  DSX$ErrSoft Routine
                 00000000   491          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000   492 ;++
                     0000   493 ; FUNCTIONAL DESCRIPTION:
                     0000   494 ;
                     0000   495 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
                     0000   496 ;       "ERRSOFT" CALL TO THE OPERATOR AND A.P.T..
                     0000   497 ;
                     0000   498 ; CALLING SEQUENCE:
                     0000   499 ;
                     0000   500 ;       NONE
                     0000   501 ;
                     0000   502 ; INPUT PARAMETERS:
                     0000   503 ;
                     0000   504 ;       ERR$_NUM(AP)     = ERROR NUMBER
                     0000   505 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
                     0000   506 ;       ERR$_MSGADR(AP)  = POINTER TO ASCII STRING
                     0000   507 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
                     0000   508 ;
                     0000   509 ; IMPLICIT INPUTS:
                     0000   510 ;
                     0000   511 ;       NONE
                     0000   512 ;
                     0000   513 ; OUTPUT PARAMETERS:
                     0000   514 ;
                     0000   515 ;       NONE
                     0000   516 ;
                     0000   517 ; IMPLICIT OUTPUTS:
                     0000   518 ;
                     0000   519 ;       NONE
                     0000   520 ;
                     0000   521 ; COMPLETION CODES:
                     0000   522 ;
                     0000   523 ;       NONE
                     0000   524 ;
                     0000   525 ; SIDE EFFECTS:
                     0000   526 ;
                     0000   527 ;       NONE
                     0000   528 ;
                     0000   529 ;--
```

```
                    0FFC  0000  .ENTRY  DSX$ERRSOFT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;                    [14]
                          0002
        00000000'EF   D6  0002          Incl    L^Ds$GL_SoftErr_Count   ; Count this soft error              [15]
        0000000D'EF   DF  0008          PUSHAL  L^SOFTER                ; INDICATE ERROR TYPE                [08]
                00C0  30  000E          BSBW    RRPTERROR               ; CALL ERROR REPORTING SUBRTN
   0000FE40'EF    04  D0  0011          MOVL    #APM$_SOFTERR, -        ; Indicate error type in mailbox
                          0018                    L^DSA$GL_MSGTYP
                          0018          Br_If_Not_APT   10$             ; If not running from APT, branch    [19]
   0000FE00'EF    1F  E1  0018          BBC     #DSA$V_Apt, -                   ; If bit not set, branch
                  06      001F                    L^DSA$GL_Flags, 10$             ;
        00000000'EF   16  0020          JSB     L^APT_MSG               ; Else, transfer control to APT      [08]
                          0026
                      04  0026  10$:    RET
                          0027          .SBTTL  DSX$ErrHard Routine
                          0027  ;++
                          0027  ; FUNCTIONAL DESCRIPTION:
                          0027  ;
                          0027  ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
                          0027  ;       "ERRHARD" CALL TO THE OPERATOR AND A.P.T..
                          0027  ;
                          0027  ; CALLING SEQUENCE:
                          0027  ;
                          0027  ;       NONE
                          0027  ;
                          0027  ; INPUT PARAMETERS:
                          0027  ;
                          0027  ;       ERR$_NUM(AP)     = ERROR NUMBER
                          0027  ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
                          0027  ;       ERR$_MSGADR(AP)  = POINTER TO ASCII STRING
                          0027  ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
                          0027  ;
                          0027  ; IMPLICIT INPUTS:
                          0027  ;
                          0027  ;       NONE
                          0027  ;
                          0027  ; OUTPUT PARAMETERS:
                          0027  ;
                          0027  ;       NONE
                          0027  ;
                          0027  ; IMPLICIT OUTPUTS:
                          0027  ;
                          0027  ;       NONE
                          0027  ;
                          0027  ; COMPLETION CODES:
                          0027  ;
                          0027  ;       NONE
                          0027  ;
                          0027  ; SIDE EFFECTS:
                          0027  ;
                          0027  ;       NONE
                          0027  ;
                          0027  ;--
```

```
                    0FFC  0027   582 .ENTRY  DSX$ERRHARD,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;                    [14]
                          0029   583
        00000000'EF   D6  0029   584         Incl    L^Ds$GL_HardErr_Count   ; Count this hard error              [15]
        00000001'EF   DF  002F   585         PUSHAL  L^HARDER                ; INDICATE ERROR TYPE                [08]
                0099  30  0035   586         BSBW    RRPTERROR               ; CALL ERROR REPORTING SUBRTN
   0000FE40'EF   03   D0  0038   587         MOVL    #APM$_HARDERR, -        ; Indicate error type in mailbox
                          003F   588                   L^DSA$GL_MSGTYP
                          003F   589         Br_If_Not_APT   10$             ; If not running from APT, branch    [19]
   0000FE00'EF   1F   E1  003F               BBC     #DSA$V_Apt, -                   ; If bit not set, branch
                 06       0046                         L^DSA$GL_Flags, 10$           ;
        00000000'EF   16  0047   590         JSB     L^APT_MSG               ; Else, transfer control to APT      [08]
                          004D   591
                      04  004D   592 10$:    RET
```

```
004E          .SBTTL  DSX$ErrPrep Routine                                    [17]
004E    ;++                                                                  [17]
004E    ; FUNCTIONAL DESCRIPTION:                                            [17]
004E    ;                                                                    [17]
004E    ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN            [17]
004E    ;       "ERRPREP" CALL TO THE OPERATOR AND A.P.T..                   [17]
004E    ;                                                                    [17]
004E    ; CALLING SEQUENCE:                                                  [17]
004E    ;                                                                    [17]
004E    ;       NONE                                                         [17]
004E    ;                                                                    [17]
004E    ; INPUT PARAMETERS:                                                  [17]
004E    ;                                                                    [17]
004E    ;       ERR$_NUM(AP)     = ERROR NUMBER                              [17]
004E    ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER                       [17]
004E    ;       ERR$_MSGADR(AP)  = POINTER TO ASCII STRING                   [17]
004E    ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE                      [17]
004E    ;                                                                    [17]
004E    ; IMPLICIT INPUTS:                                                   [17]
004E    ;                                                                    [17]
004E    ;       NONE                                                         [17]
004E    ;                                                                    [17]
004E    ; OUTPUT PARAMETERS:                                                 [17]
004E    ;                                                                    [17]
004E    ;       NONE                                                         [17]
004E    ;                                                                    [17]
004E    ; IMPLICIT OUTPUTS:                                                  [17]
004E    ;                                                                    [17]
004E    ;       NONE                                                         [17]
004E    ;                                                                    [17]
004E    ; COMPLETION CODES:                                                  [17]
004E    ;                                                                    [17]
004E    ;       NONE                                                         [17]
004E    ;                                                                    [17]
004E    ; SIDE EFFECTS:                                                      [17]
004E    ;                                                                    [17]
004E    ;       NONE                                                         [17]
004E    ;                                                                    [17]
004E    ;--                                                                  [17]
```

```
                      0FFC  004E   634 .ENTRY  DSX$ERRPREP,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;                    [17]
                            0050   635
          00000000'EF   D6  0050   636         Incl    L^Ds$GL_PrepErr_Count   ; Count this device preparation  error  [17]
          0000002D'EF   DF  0056   637         PUSHAL  L^PREPER                ; INDICATE ERROR TYPE                  [17]
                0072    30  005C   638         BSBW    RRPTERROR               ; CALL ERROR REPORTING SUBRTN          [17]
     0000FE40'EF    0C   D0  005F   639         MOVL    #APM$_PREPERR, -        ; Indicate error type in mailbox       [17]
                            0066   640                   L^DSA$GL_MSGTYP
                            0066   641         Br_If_Not_APT   10$             ; If not running from APT, branch      [19]
     0000FE00'EF    1F   E1  0066               BBC     #DSA$V_Apt, -                   ; If bit not set, branch
                      06        006D                     L^DSA$GL_Flags, 10$             ;
          00000000'EF   16  006E   642         JSB     L^APT_MSG               ; Else, transfer control to APT        [17]
                            0074   643
                      04    0074   644 10$:    RET
```

```
0075    646          .SBTTL  DSX$ErrDev Routine
0075    647 ;++
0075    648 ; FUNCTIONAL DESCRIPTION:
0075    649 ;
0075    650 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
0075    651 ;       "ERRDEV" CALL TO THE OPERATOR AND A.P.T..
0075    652 ;
0075    653 ; CALLING SEQUENCE:
0075    654 ;
0075    655 ;       NONE
0075    656 ;
0075    657 ; INPUT PARAMETERS:
0075    658 ;
0075    659 ;       ERR$_NUM(AP)     = ERROR NUMBER
0075    660 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
0075    661 ;       ERR$_MSGADR(AP)  = POINTER TO ASCII STRING
0075    662 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
0075    663 ;
0075    664 ; IMPLICIT INPUTS:
0075    665 ;
0075    666 ;       NONE
0075    667 ;
0075    668 ; OUTPUT PARAMETERS:
0075    669 ;
0075    670 ;       NONE
0075    671 ;
0075    672 ; IMPLICIT OUTPUTS:
0075    673 ;
0075    674 ;       NONE
0075    675 ;
0075    676 ; COMPLETION CODES:
0075    677 ;
0075    678 ;       NONE
0075    679 ;
0075    680 ; SIDE EFFECTS:
0075    681 ;
0075    682 ;       NONE
0075    683 ;
0075    684 ;--
```

```
                     0FFC  0075   686 .ENTRY  DSX$ERRDEV,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>  ;                [14]
                           0077   687
          00000000'EF  D6  0077   688         Incl    L^Ds$GL_DevErr_Count    ; Count this device error        [15]
          00000019'EF  DF  007D   689         PUSHAL  L^DEVFER                ; INDICATE ERROR TYPE            [08]
    00000000'EF   09   E2  0083   690         BBSS    #DS$V_DEVFLG, -         ; FLAG DEVICE-FATAL              [08]
                  00       008A   691                   L^DS$GL_FLAGS,10$
                           008B   692
                  44   10  008B   693 10$:    BSBB    RRPTERROR               ; CALL ERROR REPORTING SUBRTN    [10]
    0000FE40'EF   02   D0  008D   694         MOVL    #APM$_DEVERR, -         ; Indicate error type in mailbox
                           0094   695                   L^DSA$GL_MSGTYP
                           0094   696         Br_If_Not_APT   20$             ; If not running from APT, branch [19]
    0000FE00'EF   1F   E1  0094               BBC     #DSA$V_Apt, -                   ; If bit not set, branch
                  06       009B                   L^DSA$GL_Flags, 20$             ;
          00000000'EF  16  009C   697         JSB     L^APT_MSG               ; Else, transfer control to APT  [08]
                           00A2   698
                  04       00A2   699 20$:    RET
```

```
00A3    701          .SBTTL  DSX$ErrSys Routine
00A3    702 ;++
00A3    703 ; FUNCTIONAL DESCRIPTION:
00A3    704 ;
00A3    705 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
00A3    706 ;       "ERRSYS" CALL TO THE OPERATOR AND A.P.T..
00A3    707 ;
00A3    708 ; CALLING SEQUENCE:
00A3    709 ;
00A3    710 ;       NONE
00A3    711 ;
00A3    712 ; INPUT PARAMETERS:
00A3    713 ;
00A3    714 ;       ERR$_NUM(AP)     = ERROR NUMBER
00A3    715 ;       ERR$_UNIT(AP)    = LOGICAL UNIT NUMBER
00A3    716 ;       ERR$_MSGADR(AP)  = POINTER TO ASCII STRING
00A3    717 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
00A3    718 ;
00A3    719 ; IMPLICIT INPUTS:
00A3    720 ;
00A3    721 ;       NONE
00A3    722 ;
00A3    723 ; OUTPUT PARAMETERS:
00A3    724 ;
00A3    725 ;       NONE
00A3    726 ;
00A3    727 ; IMPLICIT OUTPUTS:
00A3    728 ;
00A3    729 ;       NONE
00A3    730 ;
00A3    731 ; COMPLETION CODES:
00A3    732 ;
00A3    733 ;       NONE
00A3    734 ;
00A3    735 ; SIDE EFFECTS:
00A3    736 ;
00A3    737 ;       NONE
00A3    738 ;
00A3    739 ;--
```

```
                    0FFC  00A3   741 .ENTRY  DSX$ERRSYS,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>  ;                    [14]
                          00A5   742
        00000000'EF   D6  00A5   743         Incl    L^Ds$GL_SysErr_Count    ; Count this system error            [15]
        00000048'EF   DF  00AB   744         PUSHAL  L^SYSFER                ; INDICATE ERROR TYPE                [08]
  00000000'EF    0A   E2  00B1   745         BBSS    #DS$V_SYSFLG, -         ; FLAG SYSTEM-FATAL                  [08]
                 00        00B8   746                   L^DS$GL_FLAGS, 10$
                           00B9   747
                 16   10   00B9   748 10$:    BSBB    RRPTERROR               ; CALL ERROR REPORTING SUBRTN        [10]
  0000FE40'EF    01   D0   00BB   749         MOVL    #APM$_SYSERR, -         ; Indicate error type in mailbox
                           00C2   750                   L^DSA$GL_MSGTYP
                           00C2   751         Br_If_Not_APT   20$             ; If not running from APT, branch    [19]
  0000FE00'EF    1F   E1   00C2               BBC     #DSA$V_Apt, -                   ; If bit not set, branch
                 06        00C9                         L^DSA$GL_Flags, 20$           ;
        00000000'EF   16   00CA   752         JSB     L^APT_MSG               ; Else, transfer control to APT      [08]
                           00D0   753
                      04   00D0   754 20$:    RET
```

```
00D1    756          .SBTTL  RRptError Routine
00D1    757 ;++
00D1    758 ; FUNCTIONAL DESCRIPTION:
00D1    759 ;
00D1    760 ;       THIS ROUTINE REPORTS THE ERROR MESSAGE TO THE OPERATOR,
00D1    761 ;       IT IS REACHED VIA "BSBW" FROM A SUPERVISOR ROUTINE.
00D1    762 ;
00D1    763 ; CALLING SEQUENCE:
00D1    764 ;
00D1    765 ;       NONE
00D1    766 ;
00D1    767 ; INPUT PARAMETERS:
00D1    768 ;
00D1    769 ;       4(SP)             = POINTER TO ERROR TYPE STRING
00D1    770 ;       ERR$_NUM(AP)      = ERROR NUMBER
00D1    771 ;       ERR$_UNIT(AP)     = LOGICAL UNIT NUMBER
00D1    772 ;       ERR$_MSGADR(AP)   = POINTER TO ASCII STRING
00D1    773 ;       ERR$_POINTER(AP)= LINK TO PRINT ROUTINE
00D1    774 ;
00D1    775 ; IMPLICIT INPUTS:
00D1    776 ;
00D1    777 ;       NONE
00D1    778 ;
00D1    779 ; OUTPUT PARAMETERS:
00D1    780 ;
00D1    781 ;       NONE
00D1    782 ;
00D1    783 ; IMPLICIT OUTPUTS:
00D1    784 ;
00D1    785 ;       NONE
00D1    786 ;
00D1    787 ; COMPLETION CODES:
00D1    788 ;
00D1    789 ;       NORMAL RETURN IF "HALT" ON ERROR IS NOT SET
00D1    790 ;       GO TO COMMAND MODE IF "HALT" IS SET (IMMEDIATELY AFTER RETURN)
00D1    791 ;
00D1    792 ; SIDE EFFECTS:
00D1    793 ;
00D1    794 ;       NONE
00D1    795 ;
00D1    796 ;--
```

```
                                  00D1   798 RRPTERROR:                                          ;
          00000018'EF    D4       00D1   799          CLRL    PRINT                              ; Make sure flag is clear
          00000000'EF    D6       00D7   800          INCL    L^DS$GL_ERRCNT                     ; KEEP TOTAL OF ERRORS
                                  00DD   801          Set_ErrFlg                                 ; Set the error flag
     00000000'EF    04    E2       00DD               BBSS    #DS$V_ErrFlg, -          ; Branch if ERRFLG bit is set to        [29]
                    00             00E4                         L^DS$GL_Flags, 30000$ ; ...                                      [29]
                                  00E5       30000$:                                  ; ... here. Set bit regardless.         [29]
          00000000'EF    D4       00E5   802          CLRL    L^DS$GA_CHKLPPC                    ; CLEAR THE CHECK_LOOP P.C.
          0000FE68'EF    7C       00EB   803          CLRQ    L^DSA$GQ_MSGPTR                    ; Clear message descriptor
  0000FE44'EF    04 AC    3C       00F1   804          MOVZWL  ERR$_NUM(AP), L^DSA$GL_ERRNO       ; Save error number-only a word
                                  00F9   805          Br_If_Not_Bell  20$                        ; Skip if "BELL" flag not set
     0000FE00'EF    03    E1       00F9               BBC     #DSA$V_Bell, -                     ; If bit not set, branch        [29]
                    25             0100                         L^DSA$GL_Flags, 20$                ;                               [29]
                                  0101   806          $QIOW_S ,L^DS$GW_TTOUT, -                  ; "DING"
                                  0101   807                       #IO$_WRITEVBLK,,,, -
                                  0101   808                       L^MSGBELL, #1
                    7E    7C       0101                         CLRQ    -(SP)
                    7E    7C       0103                         CLRQ    -(SP)
                    01    DD       0105                         PUSHL   #1
          00000000'EF    DF       0107                         PUSHAL  L^MSGBELL
                    7E    7C       010D                         CLRQ    -(SP)
                    00    DD       010F                         PUSHL   #0
        7E    0000'8F    3C       0111                         MOVZWL  #IO$_WRITEVBLK,-(SP)
     7E   00000000'EF    3C       0116                         MOVZWL  L^DS$GW_TTOUT,-(SP)
                    00    DD       011D                         PUSHL   #0
     00000000'GF    0C    FB       011F                         CALLS   #12,G^SYS$QIOW
                                  0126   809
                                  0126   810 20$:     $DS_GPHARD_S  ERR$_UNIT(AP), -(SP)         ; GET ADDRESS OF UUT'S P-TABLE
                    7E    DF       0126                         PUSHAL  -(SP)
                 08 AC    DD       0128                         PUSHL   ERR$_UNIT(AP)
     00000000'9F    02    FB       012B                         CALLS   #2, @#DS$GPHARD
                51    8E    D0       0132   811          MOVL    (SP)+, R1                          ;  INTO R1
                 13 50    E9       0135   812          BLBC    R0, 30$                            ; IF NON-EXISTENT, FAKE IT
  00000000'EF    61    01    C3       0138   813          SUBL3   #1, HP$Q_DEVICE(R1), L^Q_BUFQWD    ; REDUCE COUNT BY ONE
 00000004'EF    04 A1    01    C1       0140   814          ADDL3   #1, HP$Q_DEVICE+4(R1), -           ; ADJUST POINTER (SKIP "_")
                                  0149   815                       L^Q_BUFQWD+4                     ;
                    0B    11       0149   816          BRB     40$                                ; AND PROCEED
                                  014B   817
00000000'EF    00000075'EF    7D       014B   818 30$:     MOVQ    L^QUNKDEV, L^Q_BUFQWD              ; ELSE USE "UNK" MSG
                                  0156   819
0000FE58'EF    00000000'EF    3C       0156   820 40$:     MOVZWL  L^Q_BUFQWD,L^DSA$GL_DEVLEN         ; Dev name string len to mailbox
                    07    BB       0161   821          PUSHR   #^M<R0,R1,R2>                      ; CLOBBERED BY "MOVC3"
     50   00000000'EF    7D       0163   822          MOVQ    Q_BUFQWD,R0                        ; Get length/address of name
                50    50    3C       016A   823          movzwl  r0, r0                             ; Zero extend length
     52   0000FE5C'EF    9E       016D   824          MOVAB   L^DSA$GT_DEVNAM,R2
                                  0174   825
                82    81    90       0174   826 45$:     MOVB    (R1)+,(R2)+
                 FA 50    F5       0177   827          SOBGTR  R0,45$
                    07    BA       017A   828          POPR    #^M<R0,R1,R2>                      ; RESTORE CLOBBERED REGISTERS
```

```
                                017C   830          Br_If_IE1       97$                 ; Inhibit level-1 flag set?
0000FE00'EF     04     E0       017C                BBS     #DSA$V_IE1, -                       ; If bit set, branch          [29]
                7C              0183                          L^DSA$GL_Flags, 97$               ;                             [29]
                                0184   831                                                 ;
            028C       30       0184   832 50$:      BSBW    DS_TESTID                   ; GET TEST/SUBTEST MESSAGE.
      50    0C BC      DE       0187   833           MOVAL   @ERR$_MSGADR(AP),R0         ; GET ADDRESS OF CALLER'S MSG
                07     12       018B   834           BNEQ    90$                         ; PROCEED IF SPECIFIED
50    00000074'EF      DE       018D   835           MOVAL   L^ANULL,R0                  ; Else, use a null string
                                0194   836
      51    FC BD      9A       0194   837 90$:      movzbl  @-4(fp), r1                 ; Get error type code
            FC AD      D6       0198   838           incl    -4(fp)                      ; And increment address past type
                                019B   839
                                019B   840           ;+
                                019B   841           ; Now, check to see if CRD is running. If so, load the address of the
                                019B   842           ; user's error text into the User_Error_Text location for use by CRD.
                                019B   843           ;-
                                019B   844
                01     BB       019B   845           PushR   #^M<R0>                     ; Save R0 because it's clobbered
      00000000'EF      16       019D   846           Jsb     L^DSR$Check_MenuTest_Off_Set; Check if Menu Test  Offline bit is set
                                01A3   847                                                 ;
            09 50      E8       01A3   848           Blbs    R0, 93$                     ; If Menu Test running, branch
      00000000'EF      16       01A6   849           Jsb     L^DSR$Check_AutoTest_Off_Set; Check if Auto Test Offlinebit is set
                                01AC   850                                                 ;
            0B 50      E9       01AC   851           Blbc    R0, 95$                     ; No Auto Test, no Menu Test => branch
                                01AF   852
                01     BA       01AF   853 93$:      PopR    #^M<R0>                     ; Restore R0 - MsgAdr parameter
00000000'EF     50     D0       01B1   854           MovL    R0, -                       ; Move the MsgAdr parameter into a
                                01B8   855                          L^DS$GA_User_Error_Text ; ... location for CRD.
                02     11       01B8   856           Brb     96$                         ; Go print it
                                01BA   857
                01     BA       01BA   858 95$:      PopR    #^M<R0>                     ; Restore R0
                                01BC   859
                                01BC   860 96$:      $PRINT  r1, -                       ; Print the error message:
                                01BC   861                          #DS$K_PrintF, -           ; ... printf
                                01BC   862                          L^FMTERRHDR, -            ; ... format string
                                01BC   863                          @#L$A_NAME, -             ; ... diag name
                                01BC   864                          @#L$L_REV, -              ; ... diag rev level
                                01BC   865                          @#L$L_UPDATE, -           ; ... diag update level
                                01BC   866                          L^DSA$GL_PASSNO, -        ; ... pass number of the diagnostic
                                01BC   867                          #DS$GQ_TESTID, -          ; ... ???
                                01BC   868                          L^DSA$GL_ERRNO, -         ; ... error number
                                01BC   869                          #0, -                     ; LINE 2
                                01BC   870                          -4(FP), -                 ;
                                01BC   871                          #Q_BUFQWD, -              ; LINE 3
                                01BC   872                          R0                        ; ... MsgAdr - user's error text
                50     DD       01BC                          PUSHL   R0
      00000000'8F      DD       01BE                          PUSHL   #Q_BUFQWD
            FC AD      DD       01C4                          PUSHL   -4(FP)
                00     DD       01C7                          PUSHL   #0
      0000FE44'EF      DD       01C9                          PUSHL   L^DSA$GL_ERRNO
      00000008'8F      DD       01CF                          PUSHL   #DS$GQ_TESTID
      0000FE54'EF      DD       01D5                          PUSHL   L^DSA$GL_PASSNO
      00000210 9F      DD       01DB                          PUSHL   @#L$L_UPDATE
      0000020C 9F      DD       01E1                          PUSHL   @#L$L_REV
      00000208 9F      DD       01E7                          PUSHL   @#L$A_NAME
      0000012C'EF      9F       01ED                          PUSHAB  L^FMTERRHDR
          7E    01     9A       01F3                          movzbl  #DS$K_PrintF, -(sp)
```

```
                7E   51   98  01F6                          cvtbl   r1, -(sp)
      00000000'GF   0D   FB  01F9                          CALLS   #$$N+2, G^DSX$PRINT
                                0200   873
                10 AC   D5  0200   874 97$:    TSTL    ERR$_POINTER(AP)          ; IS THERE A PRINT LINK ?
                   03   12  0203   875         BNEQ    98$                       ; NO, SKIP
                 001E   31  0205   876         BRW     100$                      ; Print revision levels w/ or w/out prlink
      00000000'EF   09   90  0208   877 98$:    movb    #ds$k_type_error_body, -; Set type code to error body
                                020F   878                   L^ds$gb_typecode      ;
            10 BC   6C   FA  020F   879         CALLG   (AP), @ERR$_POINTER(AP) ; CALL USER'S PRINT ROUTINE
                                0213   880         $print  #ds$k_type_command_out,-
                                0213   881                 #ds$k_printf,-
                                0213   882                 linefeed
          00000190'EF   9F  0213                          PUSHAB  linefeed
                7E   01   9A  0219                          movzbl  #ds$k_printf, -(sp)
                7E   16   98  021C                          cvtbl   #ds$k_type_command_out, -(sp)
      00000000'GF   03   FB  021F                          CALLS   #$$N+2, G^DSX$PRINT
                                0226   883 100$:   Br_If_IE1       102$             ; Don't print revision level if IE1
      0000FE00'EF   04   E0  0226                 BBS     #DSA$V_IE1, -                  ; If bit set, branch          [29]
                   04          022D                           L^DSA$GL_Flags, 102$         ;                             [29]
                   04   BB  022E   884         PUSHR           #^M<R2>           ; USED FOR DEVREV$T
                   03   11  0230   885         BRB             103$
                 012F   31  0232   886 102$:   BRW             119$              ; Extend Br_If_IE1
                                0235   887
   00000044 8F   08 AC   C5  0235   888 103$:   MULL3           ERR$_UNIT(AP),#68,L_OFFSET      ; Get address of logunit's d
          00000010'EF         023D
      52  00000000'EF   DE  0242   889         MOVAL           DEVREV$T,R2                     ; TO ADD OFFSET TO
      52  00000010'EF   C0  0249   890         ADDL2           L_OFFSET,R2                     ; Move to address
            52   04   C0  0250   891         ADDL2           #4,R2
      00000010'EF   04   D0  0253   892         MOVL            #4,L_OFFSET                     ; Counter of module numbers
                   62   D5  025A   893 104$:   TSTL            (R2)                            ; Does Module # exist?
                   12   12  025C   894         BNEQ            105$                            ; Yes, than print
                04 A2   D5  025E   895         TSTL            4(R2)                           ; Does Hard rev exist?
                   0D   12  0261   896         BNEQ            105$                            ; Yes, than print
                08 A2   D5  0263   897         TSTL            8(R2)                           ; Does Firm rev exist?
                   08   12  0266   898         BNEQ            105$                            ; Yes, than print
                0C A2   D5  0268   899         TSTL            12(R2)                          ; Does Patch rev exist?
                   03   12  026B   900         BNEQ            105$                            ; Yes, than print
                 00CE   31  026D   901         BRW             115$                            ; Else, do not print
                                0270   902
          00000018'EF   D5  0270   903 105$:   TSTL            PRINT
                   20   12  0276   904         BNEQ            106$
                                0278   905         $print          #ds$k_type_command_out,-        ; Print message
                                0278   906                         #ds$k_printf,-                  ; ...use printf to type it
                                0278   907                         fmtreverror,-                   ; ...error message
                                0278   908                         #q_bufqwd                       ; ...device name
          00000000'8F   DD  0278                          PUSHL   #q_bufqwd
          00000193'EF   9F  027E                          PUSHAB  fmtreverror
                7E   01   9A  0284                          movzbl  #ds$k_printf, -(sp)
                7E   16   98  0287                          cvtbl   #ds$k_type_command_out, -(sp)
      00000000'GF   04   FB  028A                          CALLS   #$$N+2, G^DSX$PRINT
      00000018'EF   01   D0  0291   909         MOVL            #1,PRINT
      0000001C'EF   62   D0  0298   910 106$:   MOVL            (R2),MODULE                     ; to print module name
          0000001C'EF   D5  029F   911         TSTL            module
                   1B   13  02A5   912         BEQL            107$
                                02A7   913         $print          #ds$k_type_command_out,-        ; Print message
                                02A7   914                         #ds$k_printf,-                  ; ...use printf to type it
                                02A7   915                         fmtmodule,-                     ; ...error message
```

```
                                02A7   916                          module                          ; ...module name
        0000001C'EF    DD      02A7                        PUSHL   module
        000001B1'EF    9F      02AD                        PUSHAB  fmtmodule
            7E   01    9A      02B3                        movzbl  #ds$k_printf, -(sp)
            7E   16    98      02B6                        cvtbl   #ds$k_type_command_out, -(sp)
    00000000'GF   04    FB      02B9                        CALLS   #$$N+2, G^DSX$PRINT
                 13    11      02C0   917          BRB             110$                            ; Next chekc
                                02C2   918 107$:    $print          #ds$k_type_command_out,-
                                02C2   919                          #ds$k_printf,-
                                02C2   920                          fmtno_mod
        000001BC'EF    9F      02C2                        PUSHAB  fmtno_mod
            7E   01    9A      02C8                        movzbl  #ds$k_printf, -(sp)
            7E   16    98      02CB                        cvtbl   #ds$k_type_command_out, -(sp)
    00000000'GF   03    FB      02CE                        CALLS   #$$N+2, G^DSX$PRINT
            52   04    C0      02D5   921 110$:    ADDL2           #4,R2
                 62    D5      02D8   922          TSTL            (R2)
                 15    13      02DA   923          BEQL            108$
                                02DC   924
                                02DC   925
                                02DC   926          $print          #ds$k_type_command_out,-        ; print "hardware"
                                02DC   927                          #ds$k_printf,-
                                02DC   928                          hardrev,-
                                02DC   929                          R2                              ; address of ascic string
                 52    DD      02DC                        PUSHL   R2
        000000C0'EF    9F      02DE                        PUSHAB  hardrev
            7E   01    9A      02E4                        movzbl  #ds$k_printf, -(sp)
            7E   16    98      02E7                        cvtbl   #ds$k_type_command_out, -(sp)
    00000000'GF   04    FB      02EA                        CALLS   #$$N+2, G^DSX$PRINT
                                02F1   930
            52   04    C0      02F1   931 108$:    ADDL2           #4,R2
                 62    D5      02F4   932          TSTL            (R2)
                 15    13      02F6   933          BEQL            111$
                                02F8   934
                                02F8   935
                                02F8   936
                                02F8   937 109$:    $print          #ds$k_type_command_out,-        ; print "firmware"
                                02F8   938                          #ds$k_printf,-
                                02F8   939                          firmrev,-
                                02F8   940                          (R2)
                 62    DD      02F8                        PUSHL   (R2)
        000000D0'EF    9F      02FA                        PUSHAB  firmrev
            7E   01    9A      0300                        movzbl  #ds$k_printf, -(sp)
            7E   16    98      0303                        cvtbl   #ds$k_type_command_out, -(sp)
    00000000'GF   04    FB      0306                        CALLS   #$$N+2, G^DSX$PRINT
                                030D   941
            52   04    C0      030D   942 111$:    ADDL2           #4,R2
                 62    D5      0310   943          TSTL            (R2)
                 15    13      0312   944          BEQL            114$                            ; Don't print message
                                0314   945
                                0314   946
                                0314   947 113$:    $print          #ds$k_type_command_out,-        ; print "patch"
                                0314   948                          #ds$k_printf,-
                                0314   949                          patchrev,-
                                0314   950                          (R2)
                 62    DD      0314                        PUSHL   (R2)
        000000E0'EF    9F      0316                        PUSHAB  patchrev
            7E   01    9A      031C                        movzbl  #ds$k_printf, -(sp)
```

```
                7E   16   98  031F                        cvtbl   #ds$k_type_command_out, -(sp)
      00000000'GF    04   FB  0322                        CALLS   #$$N+2, G^DSX$PRINT
                                0329    951
                52   04   C0  0329    952 114$:   ADDL2           #4,R2                           ; Point to next module
      00000010'EF    01   C2  032C    953         SUBL2           #1,L_OFFSET                     ; Check for four module numb
           00000010'EF    D5  0333    954         TSTL            L_OFFSET
                     03   13  0339    955         BEQL            115$
                   FF1C   31  033B    956         BRW             104$
                     04   BA  033E    957 115$:   POPR            #^M<R2>                         ; DELETED BY DEVREV
                                0340    958
                                0340    959 117$:   Br_If_IE1       119$                            ; Don't print trailer if IE1
      0000FE00'EF    04   E0  0340                BBS     #DSA$V_IE1, -                   ; If bit set, branch                [29]
                     1C         0347                  L^DSA$GL_Flags, 119$              ;                                   [29]
                                0348    960         $print  #ds$k_type_error_end, -                 ; Print out end-of-error mes
                                0348    961                 #ds$k_printf, -                     ; .. use PRINTF to type it
                                0348    962                 fmtenderror, -                      ; .. 'end-of-error' format
                                0348    963                 -4(fp), -                           ; .. Type of error
                                0348    964                 L^dsa$gl_errno                      ; .. and error number
           0000FE44'EF    DD   0348                PUSHL   L^dsa$gl_errno
                 FC AD    DD  034E                PUSHL   -4(fp)
           000001C9'EF    9F  0351                PUSHAB  fmtenderror
                7E   01   9A  0357                movzbl  #ds$k_printf, -(sp)
                7E   0A   98  035A                cvtbl   #ds$k_type_error_end, -(sp)
      00000000'GF    05   FB  035D                CALLS   #$$N+2, G^DSX$PRINT
                                0364    965
                                0364    966 119$:   Br_If_Halt      120$                            ; Halt on error detection?
      0000FE00'EF    00   E0  0364                BBS     #DSA$V_Halt, -                  ; If bit set, branch                [29]
                     03         036B                  L^DSA$GL_Flags, 120$              ;                                   [29]
                   009A   31   036C    967         BRW     RRPTERRORX                              ; NO HALT(S), RETURN
                                036F    968                                                         ;
                                036F    969 120$:   Set_Halt                                        ; YES HALT, SET INTERNAL FLA
      0000FE00'EF    00   E2  036F                BBSS    #DSA$V_Halt, -          ; Branch if Halt bit is set to          [29]
                     00         0376                  L^DSA$GL_Flags, 30001$    ; ...
                                0377            30001$:                         ; ... here. Set bit regardless.         [29]
                                0377    970
                                0377    971         BR_IF_MP 130$                                   ; Check if Scorpio, Nautilus
05000000 8F    00000000'8F    D1  0377                CMPL    #DS$GK_SID,#^X05000000          ; Branch if Scorpio [61]
                     2A   13   0382                BEQL    130$                            ; Yes, branch
06000000 8F    00000000'8F    D1  0384                CMPL    #DS$GK_SID,#^X06000000          ; Branch if Nautilus [61]
                     1D   13   038F                BEQL    130$                            ; Yes, branch
11000000 8F    00000000'8F    D1  0391                CMPL    #DS$GK_SID,#^X11000000          ; Branch if RRR [69]            ;G:20
                     10   13   039C                BEQL    130$                                                            ;G:18
^A000000 8F    00000000'8F    D1  039E                CMPL    #DS$GK_SID,#^X0A000000          ; Branch if LLL [69]            ;G:20
                     03   13   03A9                BEQL    130$                                                            ;G:22
                   0031   31   03AB    972         BRW     140$                                    ; Handle for non-mp systems
                                03AE    973
                                03AE    974 ;++
                                03AE    975 ; NOTE: For mp systems,
                                03AE    976 ;       the saved_FP must be used to get the return pc since the current call frame
                                03AE    977 ;       built as a result of mp interlocking features.
                                03AE    978 ;--
                                03AE    979
                5A   0C AD    D0  03AE    980 130$:   MOVL    12(FP),R10                              ; Get saved Frame Pointer.
                5B   10 AA    D0  03B2    981         MOVL    16(R10),R11                             ; Get return pc
      00000000'EF    5B   D0  03B6    982         MOVL    R11,L^DS$AA_BPTADDR                     ; Save Return 'PC'
      00000000'EF    6B   90  03BD    983         MOVB    (R11),L^DS$AB_BPTINST                   ; SAVE THE OPCODE LOCATED TH
                6B   03   90  03C4    984         MOVB    #BPTCODE,(R11)                          ; CAUSE A BREAK TO THE "CLI"
```

```
                                03C7   985          $print  #ds$k_type_err_halt, -                  ; Print out HALT ON ERROR me
                                03C7   986                    #ds$k_printf, -                       ; .. Use Printf to type it
                                03C7   987                    l^fmterrhlt, -                        ; .. 'Halt on error' message
                                03C7   988                    R11                                   ; .. display "halt pc"
                     5B   DD    03C7                        PUSHL   R11
          0000010A'EF    9F    03C9                        PUSHAB  l^fmterrhlt
                7E   01   9A    03CF                        movzbl  #ds$k_printf, -(sp)
                7E   0D   98    03D2                        cvtbl   #ds$k_type_err_halt, -(sp)
      00000000'GF    04   FB    03D5                        CALLS   #$$N+2, G^DSX$PRINT
                                03DC   989
                   002A   31    03DC   990          BRW     RRPTERRORX                            ; [35]
                                03DF   991
                                03DF   992 ; The following code will be executed for non mpsystems [35]
                                03DF   993
   00000000'EF   10 BD   DE    03DF   994 140$:    MOVAL   @CF$L_PC(FP),L^DS$AA_BPTADDR           ; Save Return 'PC'
   00000000'EF   10 BD   90    03E7   995          MOVB    @CF$L_PC(FP),L^DS$AB_BPTINST           ; SAVE THE OPCODE LOCATED TH
          10 BD   03    90    03EF   996          MOVB    #BPTCODE,@CF$L_PC(FP)                  ; CAUSE A BREAK TO THE "CLI"
                                03F3   997          $print  #ds$k_type_err_halt, -                  ; Print out HALT ON ERROR me
                                03F3   998                    #ds$k_printf, -                       ; .. Use Printf to type it
                                03F3   999                    l^fmterrhlt, -                        ; .. 'Halt on error' message
                                03F3  1000                    CF$L_PC(FP)                           ; .. display "halt pc"
                10 AD   DD    03F3                        PUSHL   CF$L_PC(FP)
          0000010A'EF    9F    03F6                        PUSHAB  l^fmterrhlt
                7E   01   9A    03FC                        movzbl  #ds$k_printf, -(sp)
                7E   0D   98    03FF                        cvtbl   #ds$k_type_err_halt, -(sp)
      00000000'GF    04   FB    0402                        CALLS   #$$N+2, G^DSX$PRINT
                                0409  1001
                                0409  1002 RRPTERRORX:                                            ;
                                0409  1003          QA_MAIN Error_Error_End                       ;
                     05   DD    0409                PUSHL   #DSQA$K_Error_Error_End
      00000000'9F    01   FB    040B                CALLS   #1, @#QA$MAIN
                          05    0412  1004          RSB                                           ; RETURN TO THE DIAGNOSTIC
```

```
                                0413  1006          .SBTTL  DS_TestID Routine
                                0413  1007 ;+
                                0413  1008 ; FUNCTIONAL DESCRIPTION:
                                0413  1009 ;
                                0413  1010 ;       DETERMINES IF IN INIT OR CLEANUP CODE AND FORMS THE PROPER MESSAGE.
                                0413  1011 ;
                                0413  1012 ;--
                                0413  1013 DS_TESTID::
              0000FE50'EF   D5  0413  1014          TSTL    L^DSA$GL_TESTNO                 ; Test 0 ?
                         2C   12  0419  1015          BNEQ    70$                     ; NO, PROCEED WITH STANDARD MSG
                                041B  1016                                          ;
           01 0000FE4C'EF   D1  041B  1017          CMPL    L^DSA$GL_SUBTNO,#1              ; Test 0 & subtest 1 ?
                         0D   12  0422  1018          BNEQ    60$                     ; NO, TRY NEXT
00000008'EF   0000008B'EF   7D  0424  1019          MOVQ    L^QINITSEC,L^DS$GQ_TESTID       ; Yes, use init section msg     [08]
                         4D   11  042F  1020          BRB     80$                     ;  AND PROCEED
           02 0000FE4C'EF   D1  0431  1021 60$:     CMPL    L^DSA$GL_SUBTNO,#2              ; Test 0 & subtest 2 ?
                         0D   12  0438  1022          BNEQ    70$                     ; NO, TRY NEXT
00000008'EF   000000A9'EF   7D  043A  1023          MOVQ    L^QCLEANSEC,L^DS$GQ_TESTID      ; Yes, use clean section msg    [08]
                         37   11  0445  1024          BRB     80$                     ;  AND PROCEED
           00000008'EF   20   3C  0447  1025 70$:     MOVZWL  #TESTID_LEN,L^DS$GQ_TESTID      ; SET UP QUADWORD DESCRIPTOR    [08]
0000000C'EF   00000020'EF   DE  044E  1026          MOVAL   L^T_TESTID,L^DS$GQ_TESTID+4     ;  FOR "TEST/SUBTEST" STRING    [08]
                                0459  1027          $FAO_S  L^QFMTTESTID,L^DS$GQ_TESTID,-   ; Convert test and subtest    [08]
                                0459  1028                  L^DS$GQ_TESTID,-                ;   numbers                     [08]
                                0459  1029                  L^DSA$GL_TESTNO,L^DSA$GL_SUBTNO ;  into ascii for output
              0000FE4C'EF   DD  0459                        PUSHL   L^DSA$GL_SUBTNO
              0000FE50'EF   DD  045F                        PUSHL   L^DSA$GL_TESTNO
              00000008'EF   7F  0465                        PUSHAQ  L^DS$GQ_TESTID
              00000008'EF   3F  046B                        PUSHAW  L^DS$GQ_TESTID
              000000ED'EF   7F  0471                        PUSHAQ  L^QFMTTESTID
           00000000'GF   05   FB  0477                      CALLS   #$$T2,G^SYS$FAO
                         05   047E  1030 80$:     RSB                                 ; RETURN.
```

```
                        1032          .SBTTL  DS_ErrSup Routine
                        1033 ;++
                        1034 ; FUNCTIONAL DESCRIPTION:
                        1035 ;
                        1036 ;       THIS ROUTINE LOGS AND REPORTS THE OCCURENCE OF AN
                        1037 ;       "ERRSUP" CALL TO THE OPERATOR AND A.P.T..
                        1038 ;
                        1039 ; CALLING SEQUENCE:
                        1040 ;
                        1041 ;       CALLG   ARGLIST, a#DS_ERRSUP
                        1042 ;         OR
                        1043 ;       CALLS   #3, a#DS_ERRSUP
                        1044 ;
                        1045 ; INPUT PARAMETERS:
                        1046 ;
                        1047 ;        4(AP) ERROR NUMBER
                        1048 ;        8(AP) POINTER TO COUNTED ASCII ERROR MESSAGE STRING
                        1049 ;       12(AP) POINTER TO COUNTED ASCII MODULE NAME STRING
                        1050 ;
                        1051 ; IMPLICIT INPUTS:
                        1052 ;
                        1053 ;       NONE
                        1054 ;
                        1055 ; OUTPUT PARAMETERS:
                        1056 ;
                        1057 ;       NONE
                        1058 ;
                        1059 ; IMPLICIT OUTPUTS:
                        1060 ;
                        1061 ;       NONE
                        1062 ;
                        1063 ; COMPLETION CODES:
                        1064 ;
                        1065 ;       NONE
                        1066 ;
                        1067 ; SIDE EFFECTS:
                        1068 ;
                        1069 ;       NONE
                        1070 ;
                        1071 ;--
```

```
                        0FFC  047F  1073 .ENTRY  DS_ERRSUP,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>   ;                          [14]
                              0481  1074
                0FFF 8F   BB  0481  1075         PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     7E   DC  0485  1076         MOVPSL  -(SP)                    ; Save PSL for error typeout
                  10 AD   DD  0487  1077         PUSHL   16(FP)                   ; Push return address for typeout
                              048A  1078
     0000FE44'EF   04 AC  D0  048A  1079         MOVL    4(AP),L^DSA$GL_ERRNO     ; Error number to mailbox
            50    0C AC  D0  0492  1080         MOVL    12(AP),R0                ; Get address of ASCIC module name
                51    80  9A  0496  1081         MOVZBL  (R0)+,R1                 ; Get length of string
     0000FE58'EF    51  D0  0499  1082         MOVL    R1,L^DSA$GL_DEVLEN       ; Move string length to mailbox
     52   0000FE5C'EF    9E  04A0  1083         MOVAB   L^DSA$GT_DEVNAM,R2
                              04A7  1084
                82    80  90  04A7  1085 15$:    MOVB    (R0)+,(R2)+
                  FA 51   F5  04AA  1086         SOBGTR  R1,15$
                              04AD  1087
            52    08 BC  DE  04AD  1088         MOVAL   @8(AP),R2                ; GET OPTIONAL MSG ADDRESS
                     07   12  04B1  1089         BNEQ    10$                      ; SKIP IF VALID
     52   00000074'EF    DE  04B3  1090         MOVAL   ANULL,R2                 ; ELSE FAKE A MESSAGE
                              04BA  1091
                              04BA  1092 10$:    $print  #ds$k_type_errsup, -     ; Print Supervisor error                [10]
                              04BA  1093                   #ds$k_printf, -        ; .. use PRINTF                         [10]
                              04BA  1094                   L^FMTERRSUP, -         ; .. Format                             [10]
                              04BA  1095                   12(AP), -              ; .. force the message out              [08]
                              04BA  1096                   4(AP), -               ; .. number                             [10]
                              04BA  1097                   #0, -                  ; .. current time                       [10]
                              04BA  1098                   R2                     ;                                       [08]
                     52   DD  04BA                        PUSHL   R2
                     00   DD  04BC                        PUSHL   #0
                  04 AC   DD  04BE                        PUSHL   4(AP)
                  0C AC   DD  04C1                        PUSHL   12(AP)
          000001F8'EF     9F  04C4                        PUSHAB  L^FMTERRSUP
                7E    01  9A  04CA                        movzbl  #ds$k_printf, -(sp)
                7E    04  98  04CD                        cvtbl   #ds$k_type_errsup, -(sp)
     00000000'GF    07   FB  04D0                        CALLS   #$$N+2, G^DSX$PRINT
                              04D7  1099
          00000271'EF     9F  04D7  1100         PUSHAB  L^FMTERRSUP2             ; Push address of register text         [08]
                7E    01  9A  04DD  1101         movzbl  #ds$k_printf, -(sp)      ; .. push PRINTF code                   [10]
          FFFFFFF7 8F     DD  04E0  1102         pushl   #-ds$k_type_error_body   ; .. and type code (sticky)             [10]
     00000000'EF    11   FB  04E6  1103         CALLS   #17,DSX$PRINT            ; Print the error                       [10]
                              04ED  1104
          00000000'EF     DF  04ED  1105         PUSHAL  L^DS$GT_VDS_Version      ; ... version                           [24]
          00000000'EF     DD  04F3  1106         PUSHL   L^DS$GL_Flags            ; ... DS flags                          [24]
          0000FE00'EF     DD  04F9  1107         PUSHL   L^DSA$GL_Flags           ; ... DSA flags                         [24]
          00000237'EF     DF  04FF  1108         PUSHAL  L^FmtErrSup1             ; ... format string                     [24]
                7E    01  9A  0505  1109         MOVZBL  #DS$K_PrintF, -(SP)      ; ... use PRINTF                        [24]
                7E    04  98  0508  1110         CVTBL   #DS$K_Type_ErrSup, -(SP); Print Supervisor error                [24]
     00000000'GF    06   FB  050B  1111         CALLS   #6,G^DSX$PRINT           ;                                       [24]
                              0512  1112
                52    5E  D0  0512  1113         MOVL    SP,R2                    ; Copy current stack pointer
                53    08  D0  0515  1114         MOVL    #8,R3                    ; Print several lines                   [21]
                              0518  1115
            52    20 A2  DE  0518  1116 16$:    MOVAL   32(R2),R2                ; Point past
                51    52  D0  051C  1117         MOVL    R2,R1                    ; Copy
                              051F  1118
                7E    71  7D  051F  1119         MOVQ    -(R1),-(SP)              ; Push quad
                7E    71  7D  0522  1120         MOVQ    -(R1),-(SP)
                7E    71  7D  0525  1121         MOVQ    -(R1),-(SP)
```

```
                      7E   71   7D  0528  1122          MOVQ    -(R1),-(SP)
                           51   DD  052B  1123          PUSHL   R1                      ; Address of stack
                00000302'EF     9F  052D  1124          PUSHAB  L^FMTERRSUP3            ; Edit text                     [08]
                      7E   01   9A  0533  1125          movzbl  #ds$k_printf,-(sp)      ; Use PRINTF                    [18]
                      7E   09   98  0536  1126          cvtbl   #ds$k_type_error_body,-(sp)     ;                       [18]
             00000000'GF   0C   FB  0539  1127          CALLS   #12, G^DSX$PRINT        ; print it                      [18]
                   D5 53   F5  0540  1128          SOBGTR  R3,16$                  ; Count and continue
                           0543  1129
                      04 AC   DD  0543  1130          pushl   4(ap)                   ; Set up for PRINT error number [10]
                0000005C'EF     9F  0546  1131          pushab  L^ersuper               ; .. 'Supervisor detected error' [10]
                000001C9'EF     9F  054C  1132          pushab  L^fmtenderror           ; .. use end of error format    [10]
                           01   DD  0552  1133          pushl   #ds$k_printf            ; .. use PRINTF                 [10]
                           0A   DD  0554  1134          pushl   #ds$k_type_error_end    ; .. print end of error         [10]
             00000000'GF   05   FB  0556  1135          calls   #5, G^dsx$print         ; .. print it                   [10]
             0000FE00'EF   01   D3  055D  1136          BITL    #DSA$M_HALT, -          ;                               [09]
                           0564  1137                   L^DSA$GL_FLAGS          ; Halt on detection or isolation?
                           2A   13  0564  1138          BEQL    DSERRSUPX               ; Branch if not
                           0566  1139
          00000000'EF   10 BD   DE  0566  1140          MOVAL   @16(FP),L^DS$AA_BPTADDR ; SAVE RETURN 'PC'              [08]
          00000000'EF   10 BD   90  056E  1141          MOVB    @16(FP),L^DS$AB_BPTINST ; SAVE THE OPCODE LOCATED THERE [08]
                  10 BD   03   90  0576  1142          MOVB    #BPTCODE,@16(FP)        ; CAUSE A BREAK TO THE "CLI"
                           057A  1143          $PRINT  #ds$k_type_err_halt, -   ; Print error halt message     [10]
                           057A  1144                    #ds$k_printf, -         ; .. use PRINTF                 [10]
                           057A  1145                    L^FMTERRHLT, -          ; .. error halt format          [10]
                           057A  1146                    16(FP)                  ; .. display halt PC            [08]
                      10 AD   DD  057A                  PUSHL   16(FP)
                0000010A'EF     9F  057D                  PUSHAB  L^FMTERRHLT
                      7E   01   9A  0583                  movzbl  #ds$k_printf, -(sp)
                      7E   0D   98  0586                  cvtbl   #ds$k_type_err_halt, -(sp)
             00000000'GF   04   FB  0589                  CALLS   #$$N+2, G^DSX$PRINT
                           0590  1147                            ;
                           0590  1148 DSERRSUPX:
             0000FE40'EF   05   D0  0590  1149                  MOVL    #APM$_PRGERR,L^DSA$GL_MSGTYP    ; Indicate error type in mailbox
                  00000000'EF   D6  0597  1150                  Incl    L^Ds$GL_ErrSup_Count    ; Count this error      [15]
                           059D  1151                  Br_If_Not_APT   10$             ; If not running from APT, branch [19]
             0000FE00'EF   1F   E1  059D                  BBC     #DSA$V_Apt, -                   ; If bit not set, branch
                           06         05A4                    L^DSA$GL_Flags, 10$             ;
                  00000000'EF   16  05A5  1152                  JSB     L^APT_MSG               ; Else, transfer control to APT [08]
                           05AB  1153
                           05AB  1154 10$:
                           05AB  1155
                           04    05AB  1156                  RET
```

```
05AC  1158          .SBTTL  DSR$Completion Routine
05AC  1159 ;++
05AC  1160 ; FUNCTIONAL DESCRIPTION:
05AC  1161 ;
05AC  1162 ;       This routine can be called unconditionally to print
05AC  1163 ;       the text for a given error code.  It returns immediately
05AC  1164 ;       if R0 does not indicate an error
05AC  1165 ;
05AC  1166 ; CALLING SEQUENCE:
05AC  1167 ;
05AC  1168 ;       BSBB    DSR$COMPLETION
05AC  1169 ;
05AC  1170 ; INPUT PARAMETERS:     NONE
05AC  1171 ;
05AC  1172 ; IMPLICIT INPUTS:
05AC  1173 ;
05AC  1174 ;       R0      Contains the Completion code to be translated and typed
05AC  1175 ;
05AC  1176 ; OUTPUT PARAMETERS:    NONE
05AC  1177 ;
05AC  1178 ; IMPLICIT OUTPUTS:     NONE
05AC  1179 ;
05AC  1180 ; COMPLETION CODES:     NONE
05AC  1181 ;
05AC  1182 ; SIDE EFFECTS:         NONE... ALL REGISTERS ARE LEFT INTACT
05AC  1183 ;
05AC  1184 ;--
```

```
                        05AC  1186 DSR$COMPLETION::
                 01 50   E9  05AC  1187          BLBC    R0,10$                      ; Continue only if error
                         05  05AF  1188          RSB                                 ; Return since not in error
                             05B0  1189 10$:
                      1F   BB  05B0  1190          PUSHR   #^M<R0,R1,R2,R3,R4>         ; Save working registers
                   54   5E   D0  05B2  1191          MOVL    SP,R4                       ; Copy current stack pointer
                             05B5  1192
                      50   DD  05B5  1193          PUSHL   R0                          ; Last arg is actual return
                   52   50   D0  05B7  1194          MOVL    R0,R2                       ; Save uncut code
                             05BA  1195
                   50   07   CA  05BA  1196          BICL    #7,R0                       ; Clear level bits
               53  00000357'EF   9E  05BD  1197          MOVAB   L^ERROR_CODES,R3            ; Point to table of error codes     [08]
                         61   10  05C4  1198          BSBB    80$                         ; Locate error text
            51  00000413'EF41   3E  05C6  1199          MOVAW   L^ERROR_TEXT[R1],R1         ; Get offset to error text          [08]
                      7E   61   3C  05CE  1200          MOVZWL  (R1),-(SP)                  ; Push offset to error text
                      6E   51   C0  05D1  1201          ADDL    R1,(SP)                     ; And make it absolute
                             05D4  1202
               51   52   02   01   EF  05D4  1203          EXTZV   #1,#2,R2,R1                 ; Get severity
                00000974'EF41   9F  05D9  1204          PUSHAB  L^SEVERITY[R1]              ; Push address of severity          [08]
                         01   DD  05E0  1205          PUSHL   #1                          ; String length=1
                             05E2  1206
               50   52   0C   10   EF  05E2  1207          EXTZV   #16,#12,R2,R0               ; Get facility code
                53  0000032C'EF   9E  05E7  1208          MOVAB   L^UTILITY_CODES,R3          ; Point to table of utilities       [08]
                         2A   10  05EE  1209          BSBB    50$                         ; Locate name of utility
            51  0000033C'EF41   3E  05F0  1210          MOVAW   L^UTILITY_TEXT[R1],R1       ; Get offset to error text          [08]
                      7E   61   3C  05F8  1211          MOVZWL  (R1),-(SP)                  ; Push offset to error text
                      6E   51   C0  05FB  1212          ADDL    R1,(SP)                     ; And make it absolute
                             05FE  1213
                00000313'EF   9F  05FE  1214          PUSHAB  L^T_ERROR_MASK              ; Push address for edit string      [08]
                   7E   01   9A  0604  1215          movzbl  #dss$k_printf, -(sp)        ; Use PRINTF                        [10]
                   7E   03   9A  0607  1216          movzbl  #dss$k_type_general_error, -(sp) ; And general error code       [10]
                   54   5E   C2  060A  1217          SUBL    SP,R4                       ; Bytes used by args
                   54   04   C6  060D  1218          DIVL2   #4,R4                       ; Longwords
            00000000'GF   54   FB  0610  1219          CALLS   R4, G^DSX$PRINT             ; Type the error text               [10]
                             0617  1220                                                                                  ;G:5
                      1F   BA  0617  1221          POPR    #^M<R0,R1,R2,R3,R4>         ; Restore
                         05  0619  1222          RSB                                 ; Return
                             061A  1223 50$:
                   51   63   D0  061A  1224          MOVL    (R3),R1                     ; Get length of table
                 6341   50   D1  061D  1225 60$:     CMPL    R0, (R3)[R1]                ; Is this it?                       [12]
                         03   13  0621  1226          BEQL    70$                         ; Copy text address and return
                      F7 51   F5  0623  1227          SOBGTR  R1,60$                      ; Count and continue
                         05  0626  1228 70$:     RSB                                 ; Return
                             0627  1229
                             0627  1230
                             0627  1231 80$:
                   51   63   D0  0627  1232          MOVL    (R3),R1                     ; Get length of table
                7E 6341   07   CB  062A  1233 90$:     bicl3   #7, (r3)[r1], -(sp)         ; Clear out low bits                [12]
                   8E   50   D1  062F  1234          CMPL    R0, (sp)+                   ; Is this it?                       [12]
                         03   13  0632  1235          BEQL    100$                        ; Copy text address and return
                      F3 51   F5  0634  1236          SOBGTR  R1,90$                      ; Count and continue
                         05  0637  1237 100$:    RSB                                 ; Return
                             0638  1238          .END
```

```
$$ARGS                          = 0000000A     D
$$N                             = 00000002     D
$$T1                            = 00000001     D
$$T2                            = 00000005     D
ANULL                             00000074 R   D   03
APC$_ABORT                      = 00000006     D
APC$_CONTINUE                   = 00000009     D
APC$_DUMP                       = 00000003     D
APC$_NOP                        = 00000000     D
APC$_START                      = 00000005     D
APC$_ZERO                       = 00000004     D
APM$_ABTDON                     = 00000006     D
APM$_DEVERR                     = 00000002     D
APM$_DONE                       = 0000000A     D
APM$_EXCEPT                     = 0000000B     D
APM$_HARDERR                    = 00000003     D
APM$_MORE                       = 00000008     D
APM$_NOMES                      = 00000000     D
APM$_PREPERR                    = 0000000C     D
APM$_PRGERR                     = 00000005     D
APM$_SOFTERR                    = 00000004     D
APM$_SPOOL                      = 00000009     D
APM$_SYSERR                     = 00000001     D
APT_MSG                           ********   X   04
BIT...                          = 00000004     D
BPTCODE                         = 00000003     D
CF$L_AP                           00000008     D
CF$L_FP                           0000000C     D
CF$L_ONCOND                       00000000     D
CF$L_PC                           00000010     D
CF$L_REG                          00000014     D
CF$W_MASK                         00000006     D
CF$W_PSW                          00000004     D
DEVFER                            00000019 R   D   03
DEVREV$T                          ********   X   04
DS$AA_BPTADDR                     ********   X   04
DS$AB_BPTINST                     ********   X   04
DS$GA_CHKLPPC                     ********   X   04
DS$GA_USER_ERROR_TEXT             ********   X   04
DS$GB_TYPECODE                    ********   X   04
DS$GK_SID                         ********   X   04
DS$GL_DEVERR_COUNT                ********   X   04
DS$GL_ERRCNT                      ********   X   04
DS$GL_ERRSUP_COUNT                ********   X   04
DS$GL_FLAGS                       ********   X   04
DS$GL_HARDERR_COUNT               ********   X   04
DS$GL_PREPERR_COUNT               ********   X   04
DS$GL_SOFTERR_COUNT               ********   X   04
DS$GL_SYSERR_COUNT                ********   X   04
DS$GPHARD                         ********   X   04
DS$GQ_TESTID                      00000008 RG  D   02
DS$GT_VDS_VERSION                 ********   X   04
DS$GW_TTOUT                       ********   X   04
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
DS$K_TYPE_ERRSOFT               = 00000007     D
DS$K_TYPE_ERRSUP                = 00000004     D
DS$K_TYPE_ERRSYS                = 00000005     D
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
```

```
DS$M_EXETST                     =  00040000     D       DS$_ICERR                       =  006600C0     D
DS$M_HLTFLG                     =  00000008     D       DS$_IHWE                        =  00660060     D
DS$M_LODFLG                     =  00000002     D       DS$_ILLCHAR                     =  00660018     D
DS$M_MEMMGT                     =  00008000     D       DS$_ILLPAGCNT                   =  00660078     D
DS$M_NI                         =  04000000     D       DS$_ILLUNIT                     =  00660100     D
DS$M_OUTPUT                     =  00800000     D       DS$_INIT_FAIL                   =  00660140     D
DS$M_RUBFLG                     =  00000020     D       DS$_INSFMEM                     =  00660050     D
DS$M_SCRIPT                     =  00200000     D       DS$_INVCPU                      =  00660130     D
DS$M_SETIMR                     =  02000000     D       DS$_IPL2HI                      =  006600B8     D
DS$M_STRFLG                     =  00000004     D       DS$_IVADDR                      =  00660040     D
DS$M_SUBT                       =  00004000     D       DS$_IVVECT                      =  00660038     D
DS$M_SYSFLG                     =  00000400     D       DS$_KRNLSTK                     =  00660090     D
DS$M_TIMED                      =  02000000     D       DS$_LOGIC                       =  00660070     D
DS$M_TIMED_OUT                  =  08000000     D       DS$_MCHK                        =  00660088     D
DS$M_TIMRON                     =  00020000     D       DS$_MEM_ALLOC_ERR               =  00660138     D
DS$V_ABRTFLG                    =  00000006     D       DS$_MMOFF                       =  00660058     D
DS$V_BADTIME                    =  00000014     D       DS$_NEEDUNIT                    =  006600F8     D
DS$V_BATCH                      =  00000016     D       DS$_NODE                        =  00660128     D
DS$V_BRKCLR                     =  0000000C     D       DS$_NOPCS                       =  00660110     D
DS$V_BRKPT                      =  0000000B     D       DS$_NORMAL                      =  00660001     D
DS$V_CHARFLG                    =  00000008     D       DS$_NOSUPPORT                   =  006600B1     D
DS$V_CMDFLG                     =  00000007     D       DS$_NOTALLOWMP                  =  00660148     D
DS$V_CTRLC                      =  00000000     D       DS$_NOTDON                      =  00660030     D
DS$V_CTRLO                      =  00000010     D       DS$_NOTIMP                      =  006600B0     D
DS$V_DEVFLG                     =  00000009     D       DS$_NULLSTR                     =  00660010     D
DS$V_DISABLCC                   =  00000018     D       DS$_OVERFLOW                    =  00660008     D
DS$V_DONFLG                     =  0000000D     D       DS$_POWER                       =  00660098     D
DS$V_ERRFLG                     =  00000004     D       DS$_PROGERR                     =  00660020     D
DS$V_EXCEPT                     =  00000013     D       DS$_SEVERE                      =  00660004     D
DS$V_EXETST                     =  00000012     D       DS$_TRANSL                      =  006600A0     D
DS$V_HLTFLG                     =  00000003     D       DS$_TRUNCATE                    =  00660028     D
DS$V_LODFLG                     =  00000001     D       DS$_UNEXPINT                    =  006600D8     D
DS$V_MEMMGT                     =  0000000F     D       DS$_UNSUPPDEV                   =  00660158     D
DS$V_NI                         =  0000001A     D       DS$_VASFULL                     =  00660048     D
DS$V_OUTPUT                     =  00000017     D       DS$_WARNING                     =  00660000     D
DS$V_RUBFLG                     =  00000005     D       DSA$AL_APTMAIL                     0000FE00     D
DS$V_SCRIPT                     =  00000015     D       DSA$AT_APTTXT                      0000FA00     D
DS$V_SETIMR                     =  00000019     D       DSA$GL_APTCOM                      0000FE04     D
DS$V_STRFLG                     =  00000002     D       DSA$GL_DEVLEN                      0000FE58     D
DS$V_SUBT                       =  0000000E     D       DSA$GL_ERRNO                       0000FE44     D
DS$V_SYSFLG                     =  0000000A     D       DSA$GL_EVENT                       0000FE48     D
DS$V_TIMED                      =  00000019     D       DSA$GL_FLAGS                       0000FE00     D
DS$V_TIMED_OUT                  =  0000001B     D       DSA$GL_MSGTYP                      0000FE40     D
DS$V_TIMRON                     =  00000011     D       DSA$GL_PASSES                      0000FE08     D
DS$_AP_NORMAL_BREAK             =  00660150     D       DSA$GL_PASSNO                      0000FE54     D
DS$_ARITH                       =  006600D0     D       DSA$GL_SECTNO                      0000FE10     D
DS$_ASBE                        =  00660118     D       DSA$GL_SID                         0000FE14     D
DS$_BADLINK                     =  006600F0     D       DSA$GL_SUBTNO                      0000FE4C     D
DS$_BADTYPE                     =  006600E8     D       DSA$GL_TESTNO                      0000FE50     D
DS$_BIIC                        =  00660120     D       DSA$GL_UNITS                       0000FE0C     D
DS$_CHME                        =  006600A8     D       DSA$GQ_MSGPTR                      0000FE68     D
DS$_CHMK                        =  006600E0     D       DSA$GT_DEVNAM                      0000FE5C     D
DS$_DEVNAME                     =  00660108     D       DSA$M_HALT                      =  00000001     D
DS$_ERROR                       =  00660002     D       DSA$V_APT                       =  0000001F     D
DS$_FHWE                        =  00660068     D       DSA$V_BELL                      =  00000003     D
DS$_FRAGBUF                     =  00660080     D       DSA$V_HALT                      =  00000000     D
DS$_ICBUSY                      =  006600C8     D       DSA$V_IE1                       =  00000004     D
```

```
DSERRSUPX                       00000590 R   D  04
DSQA$K_DISPAT_AFTER_INIT      = 00000014     D
DSQA$K_DISPAT_BEFORE_INIT     = 00000013     D
DSQA$K_DISPAT_CALLTEST        = 00000015     D
DSQA$K_DISPAT_DONE_TESTS      = 00000018     D
DSQA$K_DISPAT_DSX$ENDPASS_BGN = 00000001     D
DSQA$K_DISPAT_DSX$ENDPASS_END = 00000002     D
DSQA$K_DISPAT_DSX$ENDPASS_MID = 00000000     D
DSQA$K_DISPAT_DS_CLEANUP_BGN  = 00000003     D
DSQA$K_DISPAT_DS_CLEANUP_END  = 00000004     D
DSQA$K_DUMMY_I                = 00000007     D
DSQA$K_ERROR_ERROR_BGN        = 00000006     D
DSQA$K_ERROR_ERROR_END        = 00000005     D
DSQA$K_KERNEL_INIT_CONTEXT    = 00000008     D
DSQA$K_LOOP_DSX$BGNSUB        = 00000009     D
DSQA$K_LOOP_DSX$CKLOOP        = 0000000B     D
DSQA$K_LOOP_DSX$ENDSUB        = 0000000A     D
DSQA$K_PARAM_DSX$GETADDRESS   = 0000000E     D
DSQA$K_PARAM_DSX$GETDATA      = 0000000C     D
DSQA$K_PARAM_DSX$GETLOGICAL   = 0000000F     D
DSQA$K_PARAM_DSX$GETSTRING    = 00000010     D
DSQA$K_PARAM_DSX$GETVIELD     = 0000000D     D
DSQA$K_QA_DSX$BRANCH          = 00000011     D
DSQA$K_START_BEFORE_HEADER    = 00000017     D
DSQA$K_START_VRRESTART        = 00000012     D
DSQA$K_START_VRSTART          = 00000016     D
DSR$CHECK_AUTOTEST_OFF_SET      ********   X   04
DSR$CHECK_MENUTEST_OFF_SET      ********   X   04
DSR$COMPLETION                  000005AC RG  D  04
DSX$ERRDEV                      00000075 RG  D  04
DSX$ERRHARD                     00000027 RG  D  04
DSX$ERRPREP                     0000004E RG  D  04
DSX$ERRSOFT                     00000000 RG  D  04
DSX$ERRSYS                      000000A3 RG  D  04
DSX$PRINT                       ********     X  04
DS_ERRSUP                       0000047F RG  D  04
DS_TESTID                       00000413 RG  D  04
ERR$_MSGADR                   = 0000000C     D
ERR$_NARGS                    = 0000000A     D
ERR$_NUM                      = 00000004     D
ERR$_P1                       = 00000014     D
ERR$_P2                       = 00000018     D
ERR$_P3                       = 0000001C     D
ERR$_P4                       = 00000020     D
ERR$_P5                       = 00000024     D
ERR$_P6                       = 00000028     D
ERR$_POINTER                  = 00000010     D
ERR$_UNIT                     = 00000008     D
ERROR_CODES                     00000357 R   D  03
ERROR_TEXT                      00000413 R   D  03
ERSUPER                         000005C R   D  03
FIRMREV                         000000D0 R   D  03
FMTENDERROR                     000001C9 R   D  03
FMTERRHDR                       0000012C R   D  03
FMTERRHLT                       0000010A R   D  03
FMTERRSUP                       000001F8 R   D  03
FMTERRSUP1                      00000237 R   D  03
FMTERRSUP2                      00000271 R   D  03
FMTERRSUP3                      00000302 R   D  03
FMTMODULE                       000001B1 R   D  03
FMTNO_MOD                       000001BC R   D  03
FMTREVERROR                     00000193 R   D  03
HARDER                          00000001 R   D  03
HARDREV                         000000C0 R   D  03
HP$A_DEPENDENT                  00000032     D
HP$A_DEVICE                     00000018     D
HP$A_DVA                        0000001C     D
HP$A_LINK                       00000020     D
HP$B_DRIVE                      0000000B     D
HP$B_FLAGS                      0000000A     D
HP$Q_DEVICE                     00000000     D
HP$T_DEVICE                     0000000C     D
HP$T_TYPE                       00000026     D
HP$W_SIZE                       00000008     D
HP$W_VECTOR                     00000024     D
IO$_WRITEVBLK                   ********   X   04
L$A_CCP                         00000240     D
L$A_DEVP                        0000021C     D
L$A_DREG                        00000224     D
L$A_DTP                         00000218     D
L$A_ICP                         0000023C     D
L$A_LASTADR                     00000214     D
L$A_NAME                        00000208     D
L$A_REPP                        00000244     D
L$A_SECNAM                      00000250     D
L$A_STATAB                      00000248     D
L$A_TSTCNT                      00000254     D
L$L_ENVIRON                     00000204     D
L$L_ERRTYP                      0000024C     D
L$L_HEADLENGTH                  00000200     D
L$L_REV                         0000020C     D
L$L_UNIT                        00000220     D
L$L_UNUSED                      00000228     D
L$L_UPDATE                      00000210     D
LINEFEED                        00000190 R   D  03
L_OFFSET                        00000010 R   D  02
MODULE                          0000001C R   D  02
MSGBELL                         00000000 R   D  03
PATCHREV                        000000E0 R   D  03
PREPER                          0000002D R   D  03
PRINT                           00000018 R   D  02
QA$MAIN                         ********     X  04
QCLEANSEC                       000000A9 R   D  03
QFMTTESTID                      000000ED R   D  03
QINITSEC                        0000008B R   D  03
QUNKDEV                         00000075 R   D  03
Q_BUFQWD                        00000000 R   D  02
REVERROR                        00000014 R   D  02
RMS$_ACC                      = 0001C002     D
RMS$_CCR                      = 00018494     D
RMS$_DEV                      = 000184C4     D
RMS$_DME                      = 000184D4     D
RMS$_DNF                      = 0001C04A     D
RMS$_EOF                      = 0001827A     D
```

```
RMS$_FAB                        =  0001850C     D
RMS$_FNF                        =  00018292     D
RMS$_FNM                        =  0001852C     D
RMS$_IFI                        =  00018564     D
RMS$_IOP                        =  00018574     D
RMS$_IRC                        =  0001857C     D
RMS$_ISI                        =  00018584     D
RMS$_PRV                        =  0001829A     D
RMS$_RAB                        =  0001863C     D
RMS$_RER                        =  0001C0F4     D
RMS$_RFA                        =  0001865C     D
RMS$_RTB                        =  000181A8     D
RRPTERROR                          000000D1 R   D   04
RRPTERRORX                         00000409 R   D   04
SEVERITY                           00000974 R   D   03
SIZ...                          =  00000001     D
SOFTER                             0000000D R   D   03
SS$_ACCVIO                         ********     X   03
SS$_CTRLERR                        ********     X   03
SS$_NOSUCHDEV                      ********     X   03
SS$_NOSUCHFILE                     ********     X   03
SYS$FAO                            ********     X   04
SYS$QIOW                           ********    GX   04
SYSFER                             00000048 R   D   03
TESTID_LEN                      =  00000020     D
T_ERROR_MASK                       00000313 R   D   03
T_TESTID                           00000020 R   D   02
UTILITY_CODES                      0000032C R   D   03
UTILITY_TEXT                       0000033C R   D   03
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000040 (64.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000978 (2424.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000638 (1592.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=0000000A	228 (1)	228 (1)
$$N	=00000002	1146 (1)	1000 (1) 1098 (1) 1146 (1) 872 (1)
			#-882 (1) 908 (1) 916 (1) #-920 (1)
			929 (1) 940 (1) 950 (1) 964 (1)
			988 (1)
$$T1	=00000001	808 (1)	228 (1) 808 (1)
$$T2	=00000005	1029 (1)	1029 (1)
ANULL	00000074-R	278 (1)	1090 (1) 835 (1)
APC$_ABORT	=00000006	222 (1)	
APC$_CONTINUE	=00000009	222 (1)	
APC$_DUMP	=00000003	222 (1)	
APC$_NOP	=00000000	222 (1)	
APC$_START	=00000005	222 (1)	
APC$_ZERO	=00000004	222 (1)	
APM$_ABTDON	=00000006	222 (1)	
APM$_DEVERR	=00000002	222 (1)	#-694 (1)
APM$_DONE	=0000000A	222 (1)	
APM$_EXCEPT	=0000000B	222 (1)	
APM$_HARDERR	=00000003	222 (1)	#-587 (1)
APM$_MORE	=00000008	222 (1)	
APM$_NOMES	=00000000	222 (1)	
APM$_PREPERR	=0000000C	222 (1)	#-639 (1)
APM$_PRGERR	=00000005	222 (1)	#-1149 (1)
APM$_SOFTERR	=00000004	222 (1)	#-536 (1)
APM$_SPOOL	=00000009	222 (1)	
APM$_SYSERR	=00000001	222 (1)	#-749 (1)
APT_MSG	00000000-XR		1152 (1) 539 (1) 590 (1) 642 (1)
			697 (1) 752 (1)
BIT...	=00000004	232 (1)	223 (1) 224 (1) 226 (1) 232 (1)
BPTCODE	=00000003	219 (1)	#-1142 (1) #-984 (1) #-996 (1)
CF$L_PC	00000010		#-1000 (1) 994 (1) #-995 (1) #-996 (1)
DEVFER	00000019-R	267 (1)	689 (1)
DEVREV$T	00000000-XR		889 (1)
DS$AA_BPTADDR	00000000-XR		#-1140 (1) #-982 (1) #-994 (1)
DS$AB_BPTINST	00000000-XR		#-1141 (1) #-983 (1) #-995 (1)
DS$GA_CHKLPPC	00000000-XR		#-802 (1)
DS$GA_USER_ERROR_TEXT	00000000-XR		#-855 (1)
DS$GB_TYPECODE	00000000-XR		#-878 (1)
DS$GK_SID	00000000-XR		#-971 (1)
DS$GL_DEVERR_COUNT	00000000-XR		#-688 (1)
DS$GL_ERRCNT	00000000-XR		#-800 (1)
DS$GL_ERRSUP_COUNT	00000000-XR		#-1150 (1)
DS$GL_FLAGS	00000000-XR		#-1106 (1) 691 (1) 746 (1) 801 (1)
DS$GL_HARDERR_COUNT	00000000-XR		#-584 (1)
DS$GL_PREPERR_COUNT	00000000-XR		#-636 (1)
DS$GL_SOFTERR_COUNT	00000000-XR		#-533 (1)
DS$GL_SYSERR_COUNT	00000000-XR		#-743 (1)
DS$GPHARD	00000000-XR		810 (1)
DS$GQ_TESTID	00000008-R	243 (1)	#-1019 (1) #-1023 (1) #-1025 (1) #-1026 (1)
			1029 (1) #-872 (1)

```
DS$GT_VDS_VERSION                00000000-XR                    1105    (1)
DS$GW_TTOUT                      00000000-XR                  #-808     (1)
DS$K_ERROR                      =00000002      226     (1)
DS$K_NORMAL                     =00000001      226     (1)
DS$K_PRINTB                     =00000002      232     (1)
DS$K_PRINTF                     =00000001      232     (1)    #-1000    (1)    #-1098    (1)    #-1101    (1)    #-1109    (1)
                                                              #-1125    (1)    #-1133    (1)    #-1146    (1)    #-1215    (1)
                                                              #-872     (1)    #-882     (1)    #-908     (1)    #-916     (1)
                                                              #-920     (1)    #-929     (1)    #-940     (1)    #-950     (1)
                                                              #-964     (1)    #-988     (1)
DS$K_PRINTI                     =00000000      232     (1)
DS$K_PRINTX                     =00000003      232     (1)
DS$K_SEVERE                     =00000004      226     (1)
DS$K_SUBSYS                     =00000066      226     (1)    226       (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014      232     (1)
DS$K_TYPE_ABORT_TEST            =00000013      232     (1)
DS$K_TYPE_COMMAND_ERR           =00000015      232     (1)
DS$K_TYPE_COMMAND_OUT           =00000016      232     (1)    #-882     (1)    #-908     (1)    #-916     (1)    #-920     (1)
                                                              #-929     (1)    #-940     (1)    #-950     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A      232     (1)
DS$K_TYPE_DS_PROMPT             =00000001      232     (1)
DS$K_TYPE_DS_START              =0000001D      232     (1)
DS$K_TYPE_ERRDEV                =00000008      232     (1)    268       (1)
DS$K_TYPE_ERRHARD               =00000006      232     (1)    262       (1)
DS$K_TYPE_ERROR_BODY            =00000009      232     (1)    #-1102    (1)    #-1126    (1)    #-877     (1)
DS$K_TYPE_ERROR_END             =0000000A      232     (1)    #-1134    (1)    #-964     (1)
DS$K_TYPE_ERRPREP               =0000001B      232     (1)    271       (1)
DS$K_TYPE_ERRSOFT               =00000007      232     (1)    265       (1)
DS$K_TYPE_ERRSUP                =00000004      232     (1)    #-1098    (1)    #-1110    (1)
DS$K_TYPE_ERRSYS                =00000005      232     (1)    274       (1)
DS$K_TYPE_ERR_HALT              =0000000D      232     (1)    #-1000    (1)    #-1146    (1)    #-988     (1)
DS$K_TYPE_EXCEPTION             =0000000C      232     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      232     (1)
DS$K_TYPE_FIRST_PASS            =00000011      232     (1)
DS$K_TYPE_GENERAL               =00000000      232     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      232     (1)    #-1216    (1)
DS$K_TYPE_NO_TESTS              =00000012      232     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      232     (1)
DS$K_TYPE_PROGRAM_END           =00000010      232     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      232     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      232     (1)
DS$K_TYPE_QIO_INVADP            =00000024      232     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      232     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      232     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      232     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      232     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      232     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      232     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      232     (1)
DS$K_TYPE_START_ERR             =00000018      232     (1)
DS$K_TYPE_START_LIST            =00000025      232     (1)
DS$K_TYPE_SUMMARY               =0000000E      232     (1)
DS$K_TYPE_USER_PROMPT           =00000002      232     (1)
DS$K_WARNING                    =00000000      226     (1)
DS$M_ABRTFLG                    =00000040      223     (1)
DS$M_BADTIME                    =00100000      223     (1)
DS$M_BATCH                      =00400000      223     (1)
```

```
DS$M_BRKCLR                     =00001000     223    (1)
DS$M_BRKPT                      =00000800     223    (1)
DS$M_CHARFLG                    =00000100     223    (1)
DS$M_CMDFLG                     =00000080     223    (1)
DS$M_CTRLC                      =00000001     223    (1)
DS$M_CTRLO                      =00010000     223    (1)
DS$M_DEVFLG                     =00000200     223    (1)
DS$M_DISABLCC                   =01000000     223    (1)
DS$M_DONFLG                     =00002000     223    (1)
DS$M_ERRFLG                     =00000010     223    (1)
DS$M_EXCEPT                     =00080000     223    (1)
DS$M_EXETST                     =00040000     223    (1)
DS$M_HLTFLG                     =00000008     223    (1)
DS$M_LODFLG                     =00000002     223    (1)
DS$M_MEMMGT                     =00008000     223    (1)
DS$M_NI                         =04000000     223    (1)
DS$M_OUTPUT                     =00800000     223    (1)
DS$M_RUBFLG                     =00000020     223    (1)
DS$M_SCRIPT                     =00200000     223    (1)
DS$M_SETIMR                     =02000000     223    (1)
DS$M_STRFLG                     =00000004     223    (1)
DS$M_SUBT                       =00004000     223    (1)
DS$M_SYSFLG                     =00000400     223    (1)
DS$M_TIMED                      =02000000     223    (1)
DS$M_TIMED_OUT                  =08000000     223    (1)
DS$M_TIMRON                     =00020000     223    (1)
DS$V_ABRTFLG                    =00000006     223    (1)
DS$V_BADTIME                    =00000014     223    (1)
DS$V_BATCH                      =00000016     223    (1)
DS$V_BRKCLR                     =0000000C     223    (1)
DS$V_BRKPT                      =0000000B     223    (1)
DS$V_CHARFLG                    =00000008     223    (1)
DS$V_CMDFLG                     =00000007     223    (1)
DS$V_CTRLC                      =00000000     223    (1)
DS$V_CTRLO                      =00000010     223    (1)
DS$V_DEVFLG                     =00000009     223    (1)     #-690     (1)
DS$V_DISABLCC                   =00000018     223    (1)
DS$V_DONFLG                     =0000000D     223    (1)
DS$V_ERRFLG                     =00000004     223    (1)     #-801     (1)
DS$V_EXCEPT                     =00000013     223    (1)
DS$V_EXETST                     =00000012     223    (1)
DS$V_HLTFLG                     =00000003     223    (1)
DS$V_LODFLG                     =00000001     223    (1)
DS$V_MEMMGT                     =0000000F     223    (1)
DS$V_NI                         =0000001A     223    (1)
DS$V_OUTPUT                     =00000017     223    (1)
DS$V_RUBFLG                     =00000005     223    (1)
DS$V_SCRIPT                     =00000015     223    (1)
DS$V_SETIMR                     =00000019     223    (1)
DS$V_STRFLG                     =00000002     223    (1)
DS$V_SUBT                       =0000000E     223    (1)
DS$V_SYSFLG                     =0000000A     223    (1)     #-745     (1)
DS$V_TIMED                      =00000019     223    (1)
DS$V_TIMED_OUT                  =0000001B     223    (1)
DS$V_TIMRON                     =00000011     223    (1)
DS$_AP_NORMAL_BREAK             =00660150     226    (1)
DS$_ARITH                       =006600D0     226    (1)       387     (1)
```

```
DS$_ASBE                        =00660118     226    (1)
DS$_BADLINK                     =006600F0     226    (1)
DS$_BADTYPE                     =006600E8     226    (1)
DS$_BIIC                        =00660120     226    (1)
DS$_CHME                        =006600A8     226    (1)
DS$_CHMK                        =006600E0     226    (1)
DS$_DEVNAME                     =00660108     226    (1)
DS$_ERROR                       =00660002     226    (1)
DS$_FHWE                        =00660068     226    (1)     376    (1)
DS$_FRAGBUF                     =00660080     226    (1)     379    (1)
DS$_ICBUSY                      =006600C8     226    (1)
DS$_ICERR                       =006600C0     226    (1)     386    (1)
DS$_IHWE                        =00660060     226    (1)     375    (1)
DS$_ILLCHAR                     =00660018     226    (1)     368    (1)
DS$_ILLPAGCNT                   =00660078     226    (1)     378    (1)
DS$_ILLUNIT                     =00660100     226    (1)
DS$_INIT_FAIL                   =00660140     226    (1)
DS$_INSFMEM                     =00660050     226    (1)     374    (1)
DS$_INVCPU                      =00660130     226    (1)
DS$_IPL2HI                      =006600B8     226    (1)     385    (1)
DS$_IVADDR                      =00660040     226    (1)     372    (1)
DS$_IVVECT                      =00660038     226    (1)     371    (1)
DS$_KRNLSTK                     =00660090     226    (1)     381    (1)
DS$_LOGIC                       =00660070     226    (1)     377    (1)
DS$_MCHK                        =00660088     226    (1)     380    (1)
DS$_MEM_ALLOC_ERR               =00660138     226    (1)
DS$_MMOFF                       =00660058     226    (1)
DS$_NEEDUNIT                    =006600F8     226    (1)
DS$_NODE                        =00660128     226    (1)
DS$_NOPCS                       =00660110     226    (1)
DS$_NORMAL                      =00660001     226    (1)
DS$_NOSUPPORT                   =006600B1     226    (1)
DS$_NOTALLOWMP                  =00660148     226    (1)
DS$_NOTDON                      =00660030     226    (1)
DS$_NOTIMP                      =006600B0     226    (1)      226    (1)      384    (1)
DS$_NULLSTR                     =00660010     226    (1)      367    (1)
DS$_OVERFLOW                    =00660008     226    (1)      366    (1)
DS$_POWER                       =00660098     226    (1)      382    (1)
DS$_PROGERR                     =00660020     226    (1)      369    (1)
DS$_SEVERE                      =00660004     226    (1)
DS$_TRANSL                      =006600A0     226    (1)      383    (1)
DS$_TRUNCATE                    =00660028     226    (1)      370    (1)
DS$_UNEXPINT                    =006600D8     226    (1)      388    (1)
DS$_UNSUPPDEV                   =00660158     226    (1)      389    (1)
DS$_VASFULL                     =00660048     226    (1)      373    (1)
DS$_WARNING                     =00660000     226    (1)      226    (1)
DSA$GL_DEVLEN                    0000FE58                    #-1082   (1)    #-820    (1)
DSA$GL_ERRNO                     0000FE44                    #-1079   (1)    #-804    (1)    #-872    (1)    #-964    (1)
DSA$GL_FLAGS                     0000FE00                    #-1107   (1)    #-1137   (1)     1151    (1)     538    (1)
                                                              589    (1)      641    (1)      696    (1)     751    (1)
                                                              805    (1)      830    (1)      883    (1)     959    (1)
                                                              966    (1)      969    (1)
DSA$GL_MSGTYP                    0000FE40                    #-1149   (1)    #-537    (1)    #-588    (1)    #-640    (1)
                                                             #-695    (1)    #-750    (1)
DSA$GL_PASSNO                    0000FE54                    #-872    (1)
DSA$GL_SUBTNO                    0000FE4C                    #-1017   (1)    #-1021   (1)    #-1029   (1)
DSA$GL_TESTNO                    0000FE50                    #-1014   (1)    #-1029   (1)
```

```
DSA$GQ_MSGPTR                    0000FE68                        #-803    (1)
DSA$GT_DEVNAM                    0000FE5C                          1083    (1)       824    (1)
DSA$M_HALT                      =00000001                        #-1136    (1)
DSA$V_APT                       =0000001F                        #-1151    (1)     #-538    (1)     #-589    (1)     #-641    (1)
                                                                 #-696     (1)     #-751    (1)
DSA$V_BELL                      =00000003                        #-805     (1)
DSA$V_HALT                      =00000000                        #-966     (1)     #-969    (1)
DSA$V_IE1                       =00000004                        #-830     (1)     #-883    (1)     #-959    (1)
DSERRSUPX                        00000590-R   1148    (1)       #-1138    (1)
DSQA$K_DISPAT_AFTER_INIT        =00000014     224     (1)
DSQA$K_DISPAT_BEFORE_INIT       =00000013     224     (1)
DSQA$K_DISPAT_CALLTEST          =00000015     224     (1)
DSQA$K_DISPAT_DONE_TESTS        =00000018     224     (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001     224     (1)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002     224     (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000     224     (1)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003     224     (1)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004     224     (1)
DSQA$K_DUMMY_1                  =00000007     224     (1)
DSQA$K_ERROR_ERROR_BGN          =00000006     224     (1)
DSQA$K_ERROR_ERROR_END          =00000005     224     (1)     #-1003    (1)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008     224     (1)
DSQA$K_LOOP_DSX$BGNSUB          =00000009     224     (1)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B     224     (1)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A     224     (1)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E     224     (1)
DSQA$K_PARAM_DSX$GETDATA        =0000000C     224     (1)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F     224     (1)
DSQA$K_PARAM_DSX$GETSTRING      =00000010     224     (1)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D     224     (1)
DSQA$K_QA_DSX$BRANCH            =00000011     224     (1)
DSQA$K_START_BEFORE_HEADER      =00000017     224     (1)
DSQA$K_START_VRRESTART          =00000012     224     (1)
DSQA$K_START_VRSTART            =00000016     224     (1)
DSR$CHECK_AUTOTEST_OFF_SET       00000000-XR                     849    (1)
DSR$CHECK_MENUTEST_OFF_SET       00000000-XR                     846    (1)
DSR$COMPLETION                   000005AC-R   1186    (1)
DSX$ERRDEV                       00000075-R    686    (1)
DSX$ERRHARD                      00000027-R    582    (1)
DSX$ERRPREP                      0000004E-R    634    (1)
DSX$ERRSOFT                      00000000-R    531    (1)
DSX$ERRSYS                       000000A3-R    741    (1)
DSX$PRINT                        00000000-XR                    1000    (1)      1098    (1)      1103    (1)      1111    (1)
                                                                1127    (1)      1135    (1)      1146    (1)      1219    (1)
                                                                872     (1)      882     (1)      908     (1)      916     (1)
                                                                920     (1)      929     (1)      940     (1)      950     (1)
                                                                964     (1)      988     (1)
DS_ERRSUP                        0000047F-R   1073    (1)
DS_TESTID                        00000413-R   1013    (1)     #-832    (1)
ERR$_MSGADR                     =0000000C     228     (1)       833    (1)
ERR$_NARGS                      =0000000A     228     (1)
ERR$_NUM                        =00000004     228     (1)     #-804    (1)
ERR$_P1                         =00000014     228     (1)
ERR$_P2                         =00000018     228     (1)
ERR$_P3                         =0000001C     228     (1)
ERR$_P4                         =00000020     228     (1)
ERR$_P5                         =00000024     228     (1)
```

```
ERR$_P6                         =00000028      228     (1)
ERR$_POINTER                    =00000010      228     (1)    #-874    (1)      879    (1)
ERR$_UNIT                       =00000008      228     (1)    #-810    (1)    #-888    (1)
ERROR_CODES                      00000357-R    342     (1)      1197   (1)
ERROR_TEXT                       00000413-R    391     (1)      1199   (1)
ERSUPER                          0000005C-R    276     (1)      1131   (1)
FIRMREV                          000000D0-R    288     (1)      940    (1)
FMTENDERROR                      000001C9-R    309     (1)      1132   (1)      964    (1)
FMTERRHDR                        0000012C-R    297     (1)      872    (1)
FMTERRHLT                        0000010A-R    295     (1)      1000   (1)      1146   (1)      988    (1)
FMTERRSUP                        000001F8-R    311     (1)      1098   (1)
FMTERRSUP1                       00000237-R    314     (1)      1108   (1)
FMTERRSUP2                       00000271-R    317     (1)      1100   (1)
FMTERRSUP3                       00000302-R    322     (1)      1124   (1)
FMTMODULE                        000001B1-R    305     (1)      916    (1)
FMTNO_MOD                        000001BC-R    307     (1)      920    (1)
FMTREVERROR                      00000193-R    303     (1)      908    (1)
HARDER                           00000001-R    261     (1)      585    (1)
HARDREV                          000000C0-R    286     (1)      929    (1)
HP$Q_DEVICE                      00000000                       #-813    (1)    #-814    (1)
IO$_WRITEVBLK                    00000000-XR                    #-808    (1)
L$A_NAME                         00000208                       #-872    (1)
L$L_REV                          0000020C                       #-872    (1)
L$L_UPDATE                       00000210                       #-872    (1)
LINEFEED                         00000190-R    301     (1)      882    (1)
L_OFFSET                         00000010-R    245     (1)    #-888    (1)    #-890    (1)    #-892    (1)    #-953    (1)
                                                              #-954    (1)
MODULE                           0000001C-R    251     (1)    #-910    (1)    #-911    (1)    #-916    (1)
MSGBELL                          00000000-R    259     (1)      808    (1)
PATCHREV                         000000E0-R    290     (1)      950    (1)
PREPER                           0000002D-R    270     (1)      637    (1)
PRINT                            00000018-R    249     (1)    #-799    (1)    #-903    (1)    #-909    (1)
QA$MAIN                          00000000-XR                    1003    (1)
QCLEANSEC                        000000A9-R    284     (1)    #-1023   (1)
QFMTTESTID                       000000ED-R    292     (1)      1029   (1)
QINITSEC                         0000008B-R    282     (1)    #-1019   (1)
QUNKDEV                          00000075-R    280     (1)    #-818    (1)
Q_BUFQWD                         00000000-R    241     (1)    #-813    (1)    #-815    (1)    #-818    (1)    #-820    (1)
                                                              #-822    (1)    #-872    (1)    #-908    (1)
REVERROR                         00000014-R    247     (1)
RMS$_ACC                        =0001C002                       352    (1)
RMS$_CCR                        =00018494                       353    (1)
RMS$_DEV                        =000184C4                       350    (1)
RMS$_DME                        =000184D4                       354    (1)
RMS$_DNF                        =0001C04A                       351    (1)
RMS$_EOF                        =0001827A                       355    (1)
RMS$_FAB                        =0001850C                       356    (1)
RMS$_FNF                        =00018292                       349    (1)
RMS$_FNM                        =0001852C                       365    (1)
RMS$_IFI                        =00018564                       357    (1)
RMS$_IOP                        =00018574                       358    (1)
RMS$_IRC                        =0001857C                       364    (1)
RMS$_ISI                        =00018584                       359    (1)
RMS$_PRV                        =0001829A                       348    (1)
RMS$_RAB                        =0001863C                       360    (1)
RMS$_RER                        =0001C0F4                       361    (1)
RMS$_RFA                        =0001865C                       363    (1)
```

```
RMS$_RTB                =000181A8                  362    (1)
RRPTERROR                000000D1-R   798    (1)   #-535    (1)   #-586    (1)   #-638    (1)   #-693    (1)
                                                  #-748    (1)
RRPTERRORX               00000409-R   1002   (1)   #-967    (1)   #-990    (1)
SEVERITY                 00000974-R   487    (1)    1204    (1)
SIZ...                  =00000001     223    (1)    223     (1)
SOFTER                   0000000D-R   264    (1)    534     (1)
SS$_ACCVIO               00000000-XR                344     (1)
SS$_CTRLERR              00000000-XR                347     (1)
SS$_NOSUCHDEV            00000000-XR                345     (1)
SS$_NOSUCHFILE           00000000-XR                346     (1)
SYS$FAO                  00000000-XR                1029    (1)
SYS$QIOW                 00000000-XR                808     (1)
SYSFER                   00000048-R   273    (1)    744     (1)
TESTID_LEN              =00000020     220    (1)   #-1025    (1)     254    (1)
T_ERROR_MASK             00000313-R   325    (1)    1214    (1)
T_TESTID                 00000020-R   253    (1)    1026    (1)
UTILITY_CODES            0000032C-R   327    (1)    1208    (1)
UTILITY_TEXT             0000033C-R   333    (1)    1210    (1)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO            SIZE        DEFINITION      REFERENCES...
-----            ----        ----------      -------------
$D1_PRINT_S      1           872    (1)      1000   (1)      1098   (1)      1146   (1)      872    (1)      882    (1)
                                            908    (1)      916    (1)      920    (1)      929    (1)      940    (1)
                                            950    (1)      964    (1)      988    (1)
$DEF             1           232    (1)
$DEFINI          1           225    (1)      225    (1)      227    (1)      229    (1)      230    (1)      231    (1)
$DS_CFDEF        1           225    (1)      225    (1)
$DS_DSADEF       5           227    (1)      227    (1)
$DS_DSDEF        3           226    (1)      226    (1)
$DS_ERRDEF       1           228    (1)      228    (1)
$DS_GPHARD_S     1           810    (1)      810    (1)
$DS_HDRDEF       2           229    (1)      229    (1)
$DS_HPODEF       2           230    (1)      230    (1)
$DS_TYPEDEF      4           232    (1)      232    (1)
$EQU             1           232    (1)      224    (1)      226    (1)      232    (1)
$EQULS1          1           232    (1)      224    (1)      226    (1)      232    (1)
$EQULST          1           224    (1)      224    (1)      226    (1)      232    (1)
$FAO_S           2           1027   (1)      1027   (1)
$GBLINI          2           223    (1)      223    (1)      224    (1)      226    (1)      232    (1)
$OFFDEF          1           228    (1)      228    (1)
$PRINT           2           860    (1)      1092   (1)      1143   (1)      860    (1)      880    (1)      905    (1)
                                            913    (1)      918    (1)      926    (1)      937    (1)      947    (1)
                                            960    (1)      985    (1)      997    (1)
$PUSHADR         1           808    (1)      1029   (1)      808    (1)
$PUSHTWO         1           808    (1)      808    (1)
$QIOPUSH         1           808    (1)      808    (1)
$QIOW_S          1           806    (1)      806    (1)
$RMSDEF          13          231    (1)      231    (1)
$VIELD           1           223    (1)      223    (1)
$VIELD1          1           232    (1)      223    (1)
APTDEF           1           222    (1)      222    (1)
BR_IF_HALT       1           966    (1)      966    (1)
BR_IF_IE1        1           830    (1)      830    (1)      883    (1)      959    (1)
BR_IF_MP         1           971    (1)      971    (1)
BR_IF_NOT_APT    1           538    (1)      1151   (1)      538    (1)      589    (1)      641    (1)      696    (1)
                                            751    (1)
BR_IF_NOT_BELL   1           805    (1)      805    (1)
DSFDEF           3           223    (1)      223    (1)
DSQA             2           224    (1)      224    (1)
QA_MAIN          1           1003   (1)      1003   (1)
SET_ERRFLG       1           801    (1)      801    (1)
SET_HALT         1           969    (1)      969    (1)
```

```
                                    +-------------------------+
                                    ! Opcode Cross Reference !
                                    +-------------------------+

OPCODE          VALUE      REFERENCES...
------          -----      -------------
ADDL            00C0       1201   (1)    1212   (1)
ADDL2           00C0       890    (1)    891    (1)    921    (1)    931    (1)    942    (1)    952    (1)
ADDL3           00C1       814    (1)
BBC             00E1       1151   (1)    538    (1)    589    (1)    641    (1)    696    (1)    751    (1)    805    (1)
BBS             00E0       830    (1)    883    (1)    959    (1)    966    (1)
BBSS            00E2       690    (1)    745    (1)    801    (1)    969    (1)
BEQL            0013       1138   (1)    1226   (1)    1235   (1)    912    (1)    923    (1)    933    (1)    944    (1)
                           955    (1)    971    (1)
BICL            00CA       1196   (1)
BICL3           00CB       1233   (1)
BITL            00D3       1136   (1)
BLBC            00E9       1187   (1)    812    (1)    851    (1)
BLBS            00E8       848    (1)
BNEQ            0012       1015   (1)    1018   (1)    1022   (1)    1089   (1)    834    (1)    875    (1)    894    (1)
                           896    (1)    898    (1)    900    (1)    904    (1)
BRB             0011       1020   (1)    1024   (1)    816    (1)    856    (1)    885    (1)    917    (1)
BRW             0031       876    (1)    886    (1)    901    (1)    956    (1)    967    (1)    972    (1)    990    (1)
BSBB            0010       1198   (1)    1209   (1)    693    (1)    748    (1)
BSBW            0030       535    (1)    586    (1)    638    (1)    832    (1)
CALLG           00FA       879    (1)
CALLS           00FB       1000   (1)    1003   (1)    1029   (1)    1098   (1)    1103   (1)    1111   (1)    1127   (1)
                           1135   (1)    1146   (1)    1219   (1)    808    (1)    810    (1)    872    (1)    882    (1)
                           908    (1)    916    (1)    920    (1)    929    (1)    940    (1)    950    (1)    964    (1)
                           988    (1)
CLRL            00D4       799    (1)    802    (1)
CLRQ            007C       803    (1)    808    (1)
CMPL            00D1       1017   (1)    1021   (1)    1225   (1)    1234   (1)    971    (1)
CVTBL           0098       1000   (1)    1098   (1)    1110   (1)    1126   (1)    1146   (1)    872    (1)    882    (1)
                           908    (1)    916    (1)    920    (1)    929    (1)    940    (1)    950    (1)    964    (1)
                           988    (1)
DIVL2           00C6       1218   (1)
EXTZV           00EF       1203   (1)    1207   (1)
INCL            00D6       1150   (1)    533    (1)    584    (1)    636    (1)    688    (1)    743    (1)    800    (1)
                           838    (1)
JSB             0016       1152   (1)    539    (1)    590    (1)    642    (1)    697    (1)    752    (1)    846    (1)
                           849    (1)
MOVAB           009E       1083   (1)    1197   (1)    1208   (1)    824    (1)
MOVAL           00DE       1026   (1)    1088   (1)    1090   (1)    1116   (1)    1140   (1)    833    (1)    835    (1)
                           889    (1)    994    (1)
MOVAW           003E       1199   (1)    1210   (1)
MOVB            0090       1085   (1)    1141   (1)    1142   (1)    826    (1)    877    (1)    983    (1)    984    (1)
                           995    (1)    996    (1)
MOVL            00D0       1079   (1)    1080   (1)    1082   (1)    1113   (1)    1114   (1)    1117   (1)    1149   (1)
                           1191   (1)    1194   (1)    1224   (1)    1232   (1)    536    (1)    587    (1)    639    (1)
                           694    (1)    749    (1)    811    (1)    854    (1)    892    (1)    909    (1)    910    (1)
                           980    (1)    981    (1)    982    (1)
MOVPSL          00DC       1076   (1)
MOVQ            007D       1019   (1)    1023   (1)    1119   (1)    1120   (1)    1121   (1)    1122   (1)    818    (1)
                           822    (1)
MOVZBL          009A       1000   (1)    1081   (1)    1098   (1)    1101   (1)    1109   (1)    1125   (1)    1146   (1)
```

		1215	(1)	1216	(1)	837	(1)	872	(1)	882	(1)	908	(1)	916	(1)
		920	(1)	929	(1)	940	(1)	950	(1)	964	(1)	988	(1)		
MOVZWL	003C	1025	(1)	1200	(1)	1211	(1)	804	(1)	808	(1)	820	(1)	823	(1)
MULL3	00C5	888	(1)												
POPR	00BA	1221	(1)	828	(1)	853	(1)	858	(1)	957	(1)				
PUSHAB	009F	1000	(1)	1098	(1)	1100	(1)	1124	(1)	1131	(1)	1132	(1)	1146	(1)
		1204	(1)	1214	(1)	872	(1)	882	(1)	908	(1)	916	(1)	920	(1)
		929	(1)	940	(1)	950	(1)	964	(1)	988	(1)				
PUSHAL	00DF	1105	(1)	1108	(1)	534	(1)	585	(1)	637	(1)	689	(1)	744	(1)
		808	(1)	810	(1)										
PUSHAQ	007F	1029	(1)												
PUSHAW	003F	1029	(1)												
PUSHL	00DD	1000	(1)	1003	(1)	1029	(1)	1077	(1)	1098	(1)	1102	(1)	1106	(1)
		1107	(1)	1123	(1)	1130	(1)	1133	(1)	1134	(1)	1146	(1)	1193	(1)
		1205	(1)	808	(1)	810	(1)	872	(1)	908	(1)	916	(1)	929	(1)
		940	(1)	950	(1)	964	(1)	988	(1)						
PUSHR	00BB	1075	(1)	1190	(1)	821	(1)	845	(1)	884	(1)				
RET	0004	1156	(1)	541	(1)	592	(1)	644	(1)	699	(1)	754	(1)		
RSB	0005	1004	(1)	1030	(1)	1188	(1)	1222	(1)	1228	(1)	1237	(1)		
SOBGTR	00F5	1086	(1)	1128	(1)	1227	(1)	1236	(1)	827	(1)				
SUBL	00C2	1217	(1)												
SUBL2	00C2	953	(1)												
SUBL3	00C3	813	(1)												
TSTL	00D5	1014	(1)	874	(1)	893	(1)	895	(1)	897	(1)	899	(1)	903	(1)
		911	(1)	922	(1)	932	(1)	943	(1)	954	(1)				

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	263	(1)	266	(1)	269	(1)	272	(1)	275	(1)	277	(1)	279	(1)	287	(1)
	289	(1)	291	(1)	296	(1)	298	(1)	302	(1)	304	(1)	306	(1)	308	(1)
	310	(1)	312	(1)	315	(1)	318	(1)	323	(1)	326	(1)	338	(1)	339	(1)
	340	(1)	439	(1)	440	(1)	441	(1)	442	(1)	443	(1)	444	(1)	445	(1)
	446	(1)	447	(1)	448	(1)	449	(1)	450	(1)	451	(1)	452	(1)	453	(1)
	454	(1)	455	(1)	456	(1)	457	(1)	458	(1)	459	(1)	460	(1)	461	(1)
	462	(1)	463	(1)	464	(1)	465	(1)	466	(1)	467	(1)	468	(1)	469	(1)
	470	(1)	471	(1)	472	(1)	473	(1)	474	(1)	475	(1)	476	(1)	477	(1)
	478	(1)	479	(1)	480	(1)	481	(1)	482	(1)	483	(1)	484	(1)	485	(1)
.ASCID	281	(1)	283	(1)	285	(1)	293	(1)								
.ASCII	260	(1)	488	(1)												
.BLKB	254	(1)														
.BLKL	246	(1)	248	(1)	250	(1)	252	(1)								
.BLKQ	242	(1)	244	(1)												
.BYTE	262	(1)	265	(1)	268	(1)	271	(1)	274	(1)						
.END	1238	(1)														
.ENDC	1000	(1)	1003	(1)	1029	(1)	1098	(1)	1146	(1)	223	(1)	224	(1)	226	(1)
	232	(1)	808	(1)	872	(1)	882	(1)	908	(1)	916	(1)	920	(1)	929	(1)
	940	(1)	950	(1)	964	(1)	988	(1)								
.ENDM	1003	(1)	1027	(1)	1029	(1)	222	(1)	223	(1)	224	(1)	225	(1)	226	(1)
	227	(1)	228	(1)	229	(1)	230	(1)	231	(1)	232	(1)	538	(1)	801	(1)
	805	(1)	806	(1)	808	(1)	810	(1)	830	(1)	860	(1)	872	(1)	966	(1)
	969	(1)	971	(1)												
.ENDR	1000	(1)	1098	(1)	1146	(1)	223	(1)	224	(1)	226	(1)	232	(1)	872	(1)
	882	(1)	908	(1)	916	(1)	920	(1)	929	(1)	940	(1)	950	(1)	964	(1)
	988	(1)														
.ENTRY	1073	(1)	531	(1)	582	(1)	634	(1)	686	(1)	741	(1)				
.GLOBL	808	(1)														
.IDENT	9	(1)														
.IF	1000	(1)	1003	(1)	1029	(1)	1098	(1)	1146	(1)	223	(1)	224	(1)	226	(1)
	232	(1)	808	(1)	872	(1)	882	(1)	908	(1)	916	(1)	920	(1)	929	(1)
	940	(1)	950	(1)	964	(1)	988	(1)								
.IFF	1029	(1)	223	(1)	224	(1)	226	(1)	232	(1)	808	(1)				
.IF_FALSE	1003	(1)														
.IIF	223	(1)	224	(1)	226	(1)	232	(1)								
.IRP	1000	(1)	1029	(1)	1098	(1)	1146	(1)	223	(1)	224	(1)	226	(1)	228	(1)
	232	(1)	872	(1)	882	(1)	908	(1)	916	(1)	920	(1)	929	(1)	940	(1)
	950	(1)	964	(1)	988	(1)										
.LIBRARY	211	(1)	212	(1)	213	(1)										
.LIST	225	(1)	227	(1)	228	(1)	229	(1)								
.LONG	328	(1)	329	(1)	330	(1)	331	(1)	343	(1)	344	(1)	345	(1)	346	(1)
	347	(1)	348	(1)	349	(1)	350	(1)	351	(1)	352	(1)	353	(1)	354	(1)
	355	(1)	356	(1)	357	(1)	358	(1)	359	(1)	360	(1)	361	(1)	362	(1)
	363	(1)	364	(1)	365	(1)	366	(1)	367	(1)	368	(1)	369	(1)	370	(1)
	371	(1)	372	(1)	373	(1)	374	(1)	375	(1)	376	(1)	377	(1)	378	(1)
	379	(1)	380	(1)	381	(1)	382	(1)	383	(1)	384	(1)	385	(1)	386	(1)
	387	(1)	388	(1)	389	(1)										
.MACRO	223	(1)	224	(1)	226	(1)	232	(1)								
.MDELETE	226	(1)														
.NLIST	225	(1)	227	(1)	228	(1)	229	(1)								

```
.NOCROSS        225  (1)    227  (1)    229  (1)    230  (1)    231  (1)
.NOSHOW          10  (1)
.PAGE          1005  (1)   1031  (1)   1072  (1)   1157  (1)   1185  (1)    206  (1)    233  (1)    255  (1)
                294  (1)     31  (1)    324  (1)    489  (1)    530  (1)    581  (1)    593  (1)    633  (1)
                645  (1)    685  (1)    700  (1)    740  (1)    755  (1)    797  (1)    829  (1)
.PSECT          239  (1)    257  (1)    491  (1)
.RESTORE        225  (1)    227  (1)    229  (1)    230  (1)    231  (1)
.SAVE           225  (1)    227  (1)    229  (1)    230  (1)    231  (1)
.SBTTL         1006  (1)   1032  (1)   1158  (1)    207  (1)    490  (1)    542  (1)    594  (1)    646  (1)
                701  (1)    756  (1)
.SUBTITLE       234  (1)    256  (1)
.TITLE            8  (1)
.WORD           334  (1)    335  (1)    336  (1)    337  (1)    392  (1)    393  (1)    394  (1)    395  (1)
                396  (1)    397  (1)    398  (1)    399  (1)    400  (1)    401  (1)    402  (1)    403  (1)
                404  (1)    405  (1)    406  (1)    407  (1)    408  (1)    409  (1)    410  (1)    411  (1)
                412  (1)    413  (1)    414  (1)    415  (1)    416  (1)    417  (1)    418  (1)    419  (1)
                420  (1)    421  (1)    422  (1)    423  (1)    424  (1)    425  (1)    426  (1)    427  (1)
                428  (1)    429  (1)    430  (1)    431  (1)    432  (1)    433  (1)    434  (1)    435  (1)
                436  (1)    437  (1)    438  (1)
```

```
                                  +---------------------------+
                                  ! Register Cross Reference !
                                  +---------------------------+

REGISTER  NO. REFERENCES     REFERENCES...
--------  --------------     -------------
AP                13         #-1079   (1)   #-1080   (1)     1088   (1)   #-1098   (1)   #-1130   (1)   #-804    (1)
                             #-810    (1)     833    (1)   #-874    (1)     879    (1)   #-888    (1)
FP                14         #-1000   (1)   #-1077   (1)     1140   (1)   #-1141   (1)   #-1142   (1)   #-1146   (1)
                             #-837    (1)   #-838    (1)   #-872    (1)   #-964    (1)   #-980    (1)     994    (1)
                             #-995    (1)   #-996    (1)
R0                29         #-1075   (1)   #-1080   (1)   #-1081   (1)   #-1085   (1)   #-1187   (1)   #-1190   (1)
                             #-1193   (1)   #-1194   (1)   #-1196   (1)   #-1207   (1)   #-1221   (1)   #-1225   (1)
                             #-1234   (1)   #-812    (1)   #-821    (1)   #-822    (1)   #-823    (1)   #-827    (1)
                             #-828    (1)   #-833    (1)   #-835    (1)   #-845    (1)   #-848    (1)   #-851    (1)
                             #-853    (1)   #-854    (1)   #-858    (1)   #-872    (1)
R1                36         #-1075   (1)   #-1081   (1)   #-1082   (1)   #-1086   (1)   #-1117   (1)   #-1119   (1)
                             #-1120   (1)   #-1121   (1)   #-1122   (1)   #-1123   (1)   #-1190   (1)     1199   (1)
                             #-1200   (1)   #-1201   (1)   #-1203   (1)     1204   (1)     1210   (1)   #-1211   (1)
                             #-1212   (1)   #-1221   (1)   #-1224   (1)   #-1225   (1)   #-1227   (1)   #-1232   (1)
                             #-1233   (1)   #-1236   (1)   #-811    (1)   #-813    (1)   #-814    (1)   #-821    (1)
                             #-826    (1)   #-828    (1)   #-837    (1)   #-872    (1)
R10                9           1073   (1)   #-1075   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)   #-980    (1)   #-981    (1)
R11               12           1073   (1)   #-1075   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)   #-981    (1)   #-982    (1)   #-983    (1)   #-984    (1)   #-988    (1)
R2                45           1073   (1)   #-1075   (1)   #-1083   (1)   #-1085   (1)   #-1088   (1)   #-1090   (1)
                             #-1098   (1)   #-1113   (1)     1116   (1)   #-1117   (1)   #-1190   (1)   #-1194   (1)
                               1203   (1)     1207   (1)   #-1221   (1)     531    (1)     582    (1)     634    (1)
                               686   (1)     741    (1)   #-821    (1)   #-824    (1)   #-826    (1)   #-828    (1)
                             #-884    (1)   #-889    (1)   #-890    (1)   #-891    (1)   #-893    (1)   #-895    (1)
                             #-897    (1)   #-899    (1)   #-910    (1)   #-921    (1)   #-922    (1)   #-929    (1)
                             #-931    (1)   #-932    (1)   #-940    (1)   #-942    (1)   #-943    (1)   #-950    (1)
                             #-952    (1)   #-957    (1)
R3                17           1073   (1)   #-1075   (1)   #-1114   (1)   #-1128   (1)   #-1190   (1)   #-1197   (1)
                             #-1208   (1)   #-1221   (1)   #-1224   (1)   #-1225   (1)   #-1232   (1)   #-1233   (1)
                               531   (1)     582    (1)     634    (1)     686    (1)     741    (1)
R4                13           1073   (1)   #-1075   (1)   #-1190   (1)   #-1191   (1)   #-1217   (1)   #-1218   (1)
                             #-1219   (1)   #-1221   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)
R5                 7           1073   (1)   #-1075   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)
R6                 7           1073   (1)   #-1075   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)
R7                 7           1073   (1)   #-1075   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)
R8                 7           1073   (1)   #-1075   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)
R9                 7           1073   (1)   #-1075   (1)     531    (1)     582    (1)     634    (1)     686    (1)
                               741   (1)
SP                54         #-1000   (1)   #-1076   (1)   #-1098   (1)   #-1101   (1)   #-1109   (1)   #-1110   (1)
                             #-1113   (1)   #-1119   (1)   #-1120   (1)   #-1121   (1)   #-1122   (1)   #-1125   (1)
                             #-1126   (1)   #-1146   (1)   #-1191   (1)   #-1200   (1)   #-1201   (1)   #-1211   (1)
                             #-1212   (1)   #-1215   (1)   #-1216   (1)   #-1217   (1)   #-1233   (1)   #-1234   (1)
                             #-808    (1)     810    (1)   #-811    (1)   #-872    (1)   #-882    (1)   #-908    (1)
                             #-916    (1)   #-920    (1)   #-929    (1)   #-940    (1)   #-950    (1)   #-964    (1)
```

#-988   (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.02	00:00:00.21
Command processing	880	00:00:00.28	00:00:00.64
Pass 1	992	00:00:03.29	00:00:05.49
Symbol table sort	0	00:00:00.20	00:00:00.19
Pass 2	154	00:00:01.05	00:00:02.17
Symbol table output	2	00:00:00.05	00:00:00.16
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	9	00:00:00.44	00:00:00.63
Assembler run totals	2066	00:00:05.34	00:00:09.50

The working set limit was 2900 pages.
190058 bytes (372 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 664 non-local and 105 local symbols.
1238 source lines were read in Pass 1, producing 81 object records in Pass 2.
97 pages of virtual memory were used to define 38 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	9
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	12
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	33

813 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ERROR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element EVENT.B32
;    0002  0      ! *1     6-FEB-1986 15:18:20 DS ""
;    0003  0      ! DEC/CMS REPLACEMENT HISTORY, Element EVENT.B32
;    0004  0      %title '*** EVENT flag system services'
;    0005  0      module event (ident = '06-01',
;    0006  0                    addressing_mode (external = long_relative),
;    0007  0                    optlevel = 3,
;    0008  0                    optimize) =
;    0009  1      begin
;    0010  1
;    0011  1      !
;    0012  1      ! Copyright (c) 1977, 1981, 1983
;    0013  1      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0014  1      !
;    0015  1      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0016  1      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0017  1      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0018  1      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0019  1      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0020  1      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0021  1      ! REMAIN IN DEC.
;    0022  1      !
;    0023  1      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0024  1      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0025  1      ! CORPORATION.
;    0026  1      !
;    0027  1      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0028  1      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0029  1
;    0030  1      !++
;    0031  1      ! FACILITY:     VAX Diagnostic Supervisor
;    0032  1      !
;    0033  1      ! ABSTRACT:
;    0034  1      !
;    0035  1      ! ENVIRONMENT:
;    0036  1      !
;    0037  1      ! AUTHOR:       Dave Butenhof, 11-Sep-1981 [DS version 6.5]
;    0038  1      !
;    0039  1      ! MODIFIED BY:
;    0040  1      !
;    0041  1      !       00      - dave butenhof, 11-Sep-1981.  Converted module from Macro-32
;    0042  1      !                 to Bliss-32.  Primary purpose was to add SS$_WASSET and
;    0043  1      !                 SS$_WASCLR status returns to READEF, SETEF and CLREF.  Also,
;    0044  1      !                 routines were poorly structured and did not have entry masks.
;    0045  1      !
;    0046  1      !       01      Bob Bergazzi    May 17, 1983    Version 6.11
;    0047  1      !               Changed the order of searching libraries.
;    0048  1      !--
```

```
;    0050  1     %sbttl 'Declarations'
;    0051  1
;    0052  1     !
;    0053  1     ! Include Files:
;    0054  1     !
;    0055  1     library '$diag';                !                                   [01]
;    0056  1
;    0057  1     library '$ds';                  ! DS-specific library               [01]
;    0058  1
;    0059  1     library 'sys$library:lib';      ! Common VMS library                [01]
;    0060  1
;    0061  1     !
;    0062  1     ! Macros:
;    0063  1     !
;    0064  1
;    0065  1     builtin
;    0066  1         testbitss,
;    0067  1         testbitsc;
;    0068  1
;    0069  1     !
;    0070  1     ! Equated Symbols:
;    0071  1     !
;    0072  1
;    0073  1     !
;    0074  1     ! Own Storage:
;    0075  1     !
;    0076  1
;    0077  1     !
;    0078  1     ! Externals:
;    0079  1     !
;    0080  1
;    0081  1     external
;    0082  1         ds$ax_softpcb,
;    0083  1         ds$gq_efl : bitvector [64]; ! The event flags
;    0084  1
;    0085  1     external routine
;    0086  1         kb_check : jsb_none;        ! Check for console input
;    0087  1
;    0088  1     !
;    0089  1     ! Psect definition
;    0090  1     !
;    0091  1
;    0092  1     psect code = code (share);
```

```
;   0094  1     %sbttl 'check for legal event flag number'
;   0095  1
;   0096  1     routine check_efn : jsb_none =
;   0097  2         begin
;   0098  2
;   0099  2         builtin
;   0100  2             ap;
;   0101  2
;   0102  2         bind
;   0103  2             efn = .ap + 4;
;   0104  2
;   0105  2         if .efn gtr 127 then return ss$_illefc;
;   0106  2
;   0107  2         if .efn gtr 63 then return ss$_unasefc;
;   0108  2
;   0109  2         ss$_normal
;   0110  1         end;


                                                                .TITLE  EVENT *** EVENT flag system services
                                                                .IDENT  \06-01\

                                                                .EXTRN  DS$AX_SOFTPCB, DS$GQ_EFL
                                                                .EXTRN  KB_CHECK

                                                                .PSECT  CODE,NOWRT,  SHR,2

                0000007F   8F       04   AC  D1 00000 CHECK_EFN:
                                                                CMPL    4(AP), #127                     ; 0105
                                         05  15 00008           BLEQ    1$                              ;
                           50       EC   8F  9A 0000A           MOVZBL  #236, R0                        ;
                                             05 0000E           RSB                                     ;
                           3F       04   AC  D1 0000F 1$:       CMPL    4(AP), #63                      ; 0107
                                         06  15 00013           BLEQ    2$                              ;
                           50     0234   8F  3C 00015           MOVZWL  #564, R0                        ;
                                             05 0001A           RSB                                     ;
                           50           01  D0 0001B 2$:       MOVL    #1, R0                          ; 0110
                                             05 0001E           RSB                                     ;

; Routine Size:  31 bytes,    Routine Base:  CODE + 0000
```

```
;   0112  1     %sbttl 'set event flag routine'
;   0113  1
;   0114  1     !++
;   0115  1     ! Functional Description:
;   0116  1     !
;   0117  1     !       Sets a single event flag specified by the caller
;   0118  1     !
;   0119  1     ! Calling Sequence:
;   0120  1     !
;   0121  1     !       status = $setef (efn)
;   0122  1     !
;   0123  1     ! Input Parameters:
;   0124  1     !
;   0125  1     !       efn =>  event flag number
;   0126  1     !
;   0127  1     ! Implicit Inputs:
;   0128  1     !
;   0129  1     !       None
;   0130  1     !
;   0131  1     ! Output Parameters:
;   0132  1     !
;   0133  1     !       None
;   0134  1     !
;   0135  1     ! Implicit Outputs:
;   0136  1     !
;   0137  1     !       None
;   0138  1     !
;   0139  1     ! Completion Codes:
;   0140  1     !
;   0141  1     !       SS$_WASCLR
;   0142  1     !       SS$_WASSET
;   0143  1     !       SS$_UNASEFC                     ; 128 > EFN > 63
;   0144  1     !       SS$_ILLEFC                      ; EFN > 127
;   0145  1     !
;   0146  1     ! Side Effects:
;   0147  1     !
;   0148  1     !       None
;   0149  1     !
;   0150  1     !--
```

```
;    0152  1     global routine exe$setef (efn) =
;    0153  2         begin
;    0154  2
;    0155  2         (local s; if not (s = check_efn ()) then return .s);
;    0156  2
;    0157  2         testbitss (ds$gq_efl [.efn])^3 + 1
;    0158  1         end;


                                          0FFC 00000          .ENTRY  EXE$SETEF, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 0152
                                                                      R10,R11                                          ;
                                      DD  10 00002            BSBB    CHECK_EFN                                    ; 0155
                              12      50  E9 00004            BLBC    S, 2$                                        ;
                                      50  D4 00007            CLRL    R0                                           ; 0157
                  02 00000000G  EF  04  AC  E3 00009            BBCS    EFN, DS$GQ_EFL, 1$                           ;
                                      50  D6 00012            INCL    R0                                           ;
                              50      08  C4 00014 1$:         MULL2   #8, R0                                       ;
                                      50  D6 00017            INCL    R0                                           ;
                                          04 00019 2$:        RET                                                  ; 0158

; Routine Size:  26 bytes,    Routine Base:  CODE + 001F
```

```
;    0160  1      %sbttl 'Clear event flag routine'
;    0161  1      !++
;    0162  1      ! Functional Description:
;    0163  1      !
;    0164  1      !       Clears a single event flag specified by the caller
;    0165  1      !
;    0166  1      ! Calling Sequence:
;    0167  1      !
;    0168  1      !       status = $clref (efn)
;    0169  1      !
;    0170  1      ! Input Parameters:
;    0171  1      !
;    0172  1      !       efn =>  event flag number
;    0173  1      !
;    0174  1      ! Implicit Inputs:
;    0175  1      !
;    0176  1      !       None
;    0177  1      !
;    0178  1      ! Output Parameters:
;    0179  1      !
;    0180  1      !       None
;    0181  1      !
;    0182  1      ! Implicit Outputs:
;    0183  1      !
;    0184  1      !       None
;    0185  1      !
;    0186  1      ! Completion Codes:
;    0187  1      !
;    0188  1      !       SS$_WASSET
;    0189  1      !       SS$_WASCLR
;    0190  1      !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0191  1      !       SS$_ILLEFC                      ; EFN > 127
;    0192  1      !
;    0193  1      ! Side Effects:
;    0194  1      !
;    0195  1      !       None
;    0196  1      !
;    0197  1      !--
;    0198  1
;    0199  1      global routine exe$clref (efn) =
;    0200  2          begin
;    0201  2
;    0202  2          (local s; if not (s = check_efn ()) then return .s);
;    0203  2
;    0204  2          testbitsc (ds$gq_efl [.efn])^3 + 1
;    0205  1          end;


                                        0FFC 00000          .ENTRY  EXE$CLREF, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 0199
                                                                    R10,R11                                      ;
                                        C3  10 00002        BSBB    CHECK_EFN                                    ; 0202
                                    12  50  E9 00004        BLBC    S, 2$                                        ;
                                        50  D4 00007        CLRL    R0                                           ; 0204
                     02 00000000G   EF      04   AC  E5 00009        BBCC    EFN, DS$GQ_EFL, 1$                   ;
```

```
                                    50  D6 00012          INCL    R0                                  ;
                          50        08  C4 00014 1$:      MULL2   #8, R0                              ;
                                    50  D6 00017          INCL    R0                                  ;
                                        04 00019 2$:      RET                                         ; 0205

; Routine Size:  26 bytes,    Routine Base:  CODE + 0039
```

```
;   0207  1     %sbttl 'post event flag'
;   0208  1     !++
;   0209  1     ! Functional Description:
;   0210  1     !       Post event flag will set an event flag specified by number
;   0211  1     !       for a process specified by process ID (PID) and cause any
;   0212  1     !       necessary rescheduling if this event satisfies a wait
;   0213  1     !       condition for the process.
;   0214  1     !
;   0215  1     !       In the supervisor, this routine merely sets the flag and returns the
;   0216  1     !       PCB address as a side effect.
;   0217  1     !
;   0218  1     ! Calling Sequence:
;   0219  1     !
;   0220  1     !       JSB     SCH$POSTEF
;   0221  1     !
;   0222  1     ! Input Parameters:
;   0223  1     !       r1 - process identification (pid)
;   0224  1     !       r2 - priority increment class number
;   0225  1     !       r3 - event flag number
;   0226  1     !
;   0227  1     ! Implicit Inputs:
;   0228  1     !       None
;   0229  1     !
;   0230  1     ! Output Parameters:
;   0231  1     !       r0 - completion status code
;   0232  1     !       r4 - pcb address of process specified by pid
;   0233  1     !
;   0234  1     ! Implicit Outputs:
;   0235  1     !       event flag bit selected by pid and event flag number
;   0236  1     !       is set.
;   0237  1     !
;   0238  1     ! Completion Codes:
;   0239  1     !       SS$_NORMAL  -  normal successful completion
;   0240  1     !       SS$_NONEXPR -  non-existent process
;   0241  1     !
;   0242  1     ! Side Effects:
;   0243  1     !
;   0244  1     !       None
;   0245  1     !--
;   0246  1
;   0247  1     global routine sch$postef (pid, pri_inc, efn; pcb) : jsb_postef =
;   0248  2         begin
;   0249  2         pcb = ds$ax_softpcb;        ! Set PCB value for return
;   0250  2         ds$gq_efl [.efn] = 1;       ! Set flag
;   0251  2         ss$_normal
;   0252  1         end;



                        54 00000000G  EF  9E 00000 SCH$POSTEF::
                                                   MOVAB   DS$AX_SOFTPCB, PCB                  ; 0249
          00 00000000G  EF            53  E2 00007   BBSS    EFN, DS$GQ_EFL, 1$                  ; 0250
                        50            01  D0 0000F 1$:  MOVL    #1, R0                              ; 0252
                                          05 00012   RSB                                         ;
```

; Routine Size: 19 bytes, Routine Base: CODE + 0053

```
%sbttl 'read event flags routine'

!++
! Functional Description:
!
!       Returns a specified event flag cluster to a longword
!       in the caller's area
!
! Calling Sequence:
!
!       status = $readef (efn, state)
!
! Input Parameters:
!
!       efn =>   any event flag number within desired cluster
!       state => longword address for returned cluster
!
! Implicit Inputs:
!
!       None
!
! Output Parameters:
!
!       state => longword pointed to receives all flags in cluseter
!
! Implicit Outputs:
!
!       None
!
! Completion Codes:
!
!       SS$_UNASEFC                     ; 128 > EFN > 63
!       SS$_ILLEFC                      ; EFN > 127
!       SS$_WASSET                      ; Success, specified flag was set
!       SS$_WASCLR                      ; Success, specified flag was clear
!
! Side Effects:
!
!       None
!
!--
```

```
;   0296  1     global routine exe$readef (efn, state) =
;   0297  2         begin
;   0298  2
;   0299  2         (local s; if not (s = check_efn ()) then return .s);
;   0300  2
;   0301  3         (map ds$gq_efl : vector [2];        ! Access flags by longword
;   0302  3         .state = .ds$gq_efl [.efn/32]       ! Read correct longword
;   0303  2         );
;   0304  2
;   0305  2         .ds$gq_efl [.efn]^3 + 1     ! Return SS$_WASSET or SS_$WASCLR
;   0306  1         end;



                                                  0FFC 00000          .ENTRY  EXE$READEF, Save R2,R3,R4,R5,R6,R7,R8,R9,-   ; 0296
                                                                              R10,R11                                      ;
                                  52 00000000G  EF   9E 00002          MOVAB   DS$GQ_EFL, R2                                ;
                                                8F   10 00009          BSBB    CHECK_EFN                                    ; 0299
                                  15            50   E9 0000B          BLBC    S, 1$                                        ;
                    50        04  AC            20   C7 0000E          DIVL3   #32, EFN, R0                                 ; 0302
                              08  BC          6240   D0 00013          MOVL    DS$GQ_EFL[R0], @STATE                        ;
          50        62            01        04  AC   EF 00018          EXTZV   EFN, #1, DS$GQ_EFL, R0                       ; 0305
                                  50            08   C4 0001E          MULL2   #8, R0                                       ;
                                                50   D6 00021          INCL    R0                                           ;
                                                     04 00023 1$:      RET                                                  ; 0306

; Routine Size:  36 bytes,    Routine Base:  CODE + 0066
```

```
;    0308  1      %sbttl 'wait for single event flag routine'
;    0309  1
;    0310  1      !++
;    0311  1      ! Functional Description:
;    0312  1      !
;    0313  1      !       Suspends program execution until a single specified
;    0314  1      !       event flag sets
;    0315  1      !
;    0316  1      ! Calling Sequence:
;    0317  1      !
;    0318  1      !       state = $waitfr (efn)
;    0319  1      !
;    0320  1      ! Input Parameters:
;    0321  1      !
;    0322  1      !       efn => event flag number
;    0323  1      !
;    0324  1      ! Implicit Inputs:
;    0325  1      !
;    0326  1      !       None
;    0327  1      !
;    0328  1      ! Output Parameters:
;    0329  1      !
;    0330  1      !       None
;    0331  1      !
;    0332  1      ! Implicit Outputs:
;    0333  1      !
;    0334  1      !       None
;    0335  1      !
;    0336  1      ! Completion Codes:
;    0337  1      !
;    0338  1      !       SS$_NORMAL
;    0339  1      !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0340  1      !       SS$_ILLEFC                      ; EFN > 127
;    0341  1      !
;    0342  1      ! Side Effects:
;    0343  1      !
;    0344  1      !       None
;    0345  1      !
;    0346  1      !--
;    0347  1
;    0348  1      global routine exe$waitfr (efn) =
;    0349  2          begin
;    0350  2
;    0351  2          (local s; if not (s = check_efn ()) then return .s);
;    0352  2
;    0353  2          while not .ds$gq_efl [.efn] do kb_check ();
;    0354  2          ss$_normal
;    0355  1          end;


                                    0FFC 00000          .ENTRY  EXE$WAITFR, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 0348
                                                                R10,R11                                     ;
                                FF71  30 00002          BSBW    CHECK_EFN                                   ; 0351
                          14     50  E9 00005          BLBC    S, 3$                                       ;
```

```
                    08 00000000G  EF       04   AC  E0 00008 1$:     BBS     EFN, DS$GQ_EFL, 2$                    ; 0353
                                  00000000G  EF  16 00011             JSB     KB_CHECK                              ;
                                             EF  11 00017             BRB     1$                                    ;
                                  50         01  D0 00019 2$:        MOVL    #1, R0                                ; 0355
                                             04  0001C 3$:        RET                                               ;

; Routine Size:  29 bytes,    Routine Base:  CODE + 008A
```

```
;    0357  1      %sbttl 'wait for logical or of event flags routine'
;    0358  1
;    0359  1      !++
;    0360  1      ! Functional Description:
;    0361  1      !
;    0362  1      !       Suspends program execution until any one of a set of
;    0363  1      !       specified event flags sets
;    0364  1      !
;    0365  1      ! Calling Sequence:
;    0366  1      !
;    0367  1      !       status = $wflor (efn, mask)
;    0368  1      !
;    0369  1      ! Input Parameters:
;    0370  1      !
;    0371  1      !       efn =>  any event flag number within desired cluster
;    0372  1      !       mask => mask of desired flags within the specified cluster
;    0373  1      !
;    0374  1      ! Implicit Inputs:
;    0375  1      !
;    0376  1      !       None
;    0377  1      !
;    0378  1      ! Output Parameters:
;    0379  1      !
;    0380  1      !       None
;    0381  1      !
;    0382  1      ! Implicit Outputs:
;    0383  1      !
;    0384  1      !       None
;    0385  1      !
;    0386  1      ! Completion Codes:
;    0387  1      !
;    0388  1      !       SS$_NORMAL
;    0389  1      !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0390  1      !       SS$_ILLEFC                      ; EFN > 127
;    0391  1      !
;    0392  1      ! Side Effects:
;    0393  1      !
;    0394  1      !       None
;    0395  1      !
;    0396  1      !--
;    0397  1
;    0398  1      global routine exe$wflor (efn, mask) =
;    0399  2          begin
;    0400  2
;    0401  2          while 1 do
;    0402  3              begin
;    0403  3
;    0404  3              local
;    0405  3                  cluster;
;    0406  3
;    0407  3              kb_check ();
;    0408  3
;    0409  3              (local s; if not (s = exe$readef (.efn, cluster)) then return .s);
;    0410  3
;    0411  3              if (.cluster and .mask) neq 0 then exitloop
;    0412  2              end;
;    0413  2
```

```
;   0414  2          ss$_normal
;   0415  1          end;



                                                  0FFC 00000          .ENTRY  EXE$WFLOR, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 0398
                                                                              R10,R11                                      ;
                                  5E                04  C2 00002          SUBL2   #4, SP                                       ;
                                     00000000G      EF  16 00005 1$:      JSB     KB_CHECK                                     ; 0407
                                                    5E  DD 0000B          PUSHL   SP                                           ; 0409
                                            04      AC  DD 0000D          PUSHL   EFN                                          ;
                              AB  AF                02  FB 00010          CALLS   #2, EXE$READEF                               ;
                                  09                50  E9 00014          BLBC    S, 2$                                        ;
                              08  AC                6E  D3 00017          BITL    CLUSTER, MASK                                ; 0411
                                                    E8  13 0001B          BEQL    1$                                           ;
                                  50                01  D0 0001D          MOVL    #1, R0                                       ; 0415
                                                        04 00020 2$:      RET                                                  ;

; Routine Size:  33 bytes,    Routine Base:  CODE + 00A7
```

```
;    0417  1      %sbttl 'wait for logical and of event flags routine'
;    0418  1
;    0419  1      !++
;    0420  1      ! Functional Description:
;    0421  1      !
;    0422  1      !       Suspends program execution until all specified
;    0423  1      !       event flags are set
;    0424  1      !
;    0425  1      ! Calling Sequence:
;    0426  1      !
;    0427  1      !       status = $wfland (efn, mask)
;    0428  1      !
;    0429  1      ! Input Parameters:
;    0430  1      !
;    0431  1      !       efn =>  any event flag number within desired cluster
;    0432  1      !       mask => mask of desired flags within the specified cluster
;    0433  1      !
;    0434  1      ! Implicit Inputs:
;    0435  1      !
;    0436  1      !       None
;    0437  1      !
;    0438  1      ! Output Parameters:
;    0439  1      !
;    0440  1      !       None
;    0441  1      !
;    0442  1      ! Implicit Outputs:
;    0443  1      !
;    0444  1      !       None
;    0445  1      !
;    0446  1      ! Completion Codes:
;    0447  1      !
;    0448  1      !       SS$_NORMAL
;    0449  1      !       SS$_UNASEFC                     ; 128 > EFN > 63
;    0450  1      !       SS$_ILLEFC                      ; EFN > 127
;    0451  1      !
;    0452  1      ! Side Effects:
;    0453  1      !
;    0454  1      !       None
;    0455  1      !
;    0456  1      !--
;    0457  1
;    0458  1      global routine exe$wfland (efn, mask) =
;    0459  2          begin
;    0460  2
;    0461  2          while 1 do
;    0462  3              begin
;    0463  3
;    0464  3              local
;    0465  3                  cluster;
;    0466  3
;    0467  3              kb_check ();
;    0468  3
;    0469  3              (local s; if not (s = exe$readef (.efn, cluster)) then return .s);
;    0470  3
;    0471  3              if (.cluster and .mask) eql .mask then exitloop
;    0472  3
;    0473  2              end;
```

```
;   0474  2
;   0475  2         ss$_normal
;   0476  1         end;



                                                  0FFC 00000          .ENTRY  EXE$WFLAND, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 0458
                                                                              R10,R11                                     ;
                                  5E                04  C2 00002          SUBL2   #4, SP                                      ;
                                     00000000G      EF  16 00005 1$:      JSB     KB_CHECK                                    ; 0467
                                                    5E  DD 0000B          PUSHL   SP                                          ; 0469
                                           04       AC  DD 0000D          PUSHL   EFN                                         ;
                            8A    AF                02  FB 00010          CALLS   #2, EXE$READEF                              ;
                                  11                50  E9 00014          BLBC    S, 2$                                       ;
                                  50        08      AC  D2 00017          MCOML   MASK, R0                                    ; 0471
                     50           6E                50  CB 0001B          BICL3   R0, CLUSTER, R0                             ;
                            08    AC                50  D1 0001F          CMPL    R0, MASK                                    ;
                                                    E0  12 00023          BNEQ    1$                                          ;
                                  50                01  D0 00025          MOVL    #1, R0                                      ; 0476
                                                        04 00028 2$:      RET                                                 ;

; Routine Size:  41 bytes,    Routine Base:  CODE + 00C8


;   0477  1
;   0478  1     end
;   0479  0     eludom



;                                   PSECT SUMMARY
;
;       Name                    Bytes                       Attributes
;
;  CODE                            241  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...
AP	Builtin	100	103.
CHECK_EFN	Routine	96	155c 202c 299c 351c
CLUSTER	Local	405	409a 411.
	Local	465	469a 471.
DS$AX_SOFTPCB	External	82	249a
DS$GQ_EFL	External	83	157. 157= 204. 204= 250= 301m 302. 305. 353.
EFN	Bind	103	105. 107.
	RoutForm	152	157.
	RoutForm	199	204.
	RoutForm	247	250.
	RoutForm	296	302. 305.
	RoutForm	348	353.
	RoutForm	398	409.
	RoutForm	458	469.
EXE$CLREF	GlobRout	199	
EXE$READEF	GlobRout	296	409c 469c
EXE$SETEF	GlobRout	152	
EXE$WAITFR	GlobRout	348	
EXE$WFLAND	GlobRout	458	
EXE$WFLOR	GlobRout	398	
JSB_NONE	Linkage	Lib02 86	96
JSB_POSTEF	Linkage	Lib02 247	
KB_CHECK	ExtRout	86	353c 407c 467c
MASK	RoutForm	398	411.
	RoutForm	458	471.
PCB	RoutForm	247	249=
PID	RoutForm	247	
PRI_INC	RoutForm	247	
S	Local	155	155= 155.
	Local	202	202= 202.
	Local	299	299= 299.
	Local	351	351= 351.
	Local	409	409= 409.
	Local	469	469= 469.
SCH$POSTEF	GlobRout	247	
SS$_ILLEFC	Literal	Lib03 105	
SS$_NORMAL	Literal	Lib03 109	251 354 414 475
SS$_UNASEFC	Literal	Lib03 107	
STATE	RoutForm	296	302a=
TESTBITSC	Builtin	67	204
TESTBITSS	Builtin	66	157

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]EVENT.B32;1
5	Module EVENT	
55	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
57	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
59	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
479	Eludom EVENT	

```
        KEY TO REFERENCE TYPE FLAGS

            .   Fetch
            =   Store
            c   Routine call
            a   Address use
            @   Indirect use
            e   External, external routine, or external literal declaration
            f   Forward or forward routine declaration
            h   Condition handler enabling
            m   Map declaration
            u   Undeclare declaration


;                           Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                          940         0         0          96         00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                          675         2         0          47         00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2          20450         3         0        1101         00:00.8




;                                  COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:EVENT.B32

; Size:          241 code + 0 data bytes
; Run Time:         00:02.2
; Elapsed Time:     00:04.0
; Lines/CPU Min:    13367
; Lexemes/CPU-Min: 20232
; Memory Used:  65 pages
; Compilation Complete
```

Table of contents

(1) 176 Libraries, External & Equated Symbols, Macros
(1) 232 Work Psect Declarations
(1) 243 Data Psect Declarations
(1) 374 COND_HANDLR ROUTINE - HANDLE UNWANTED EXCEPTIONS
(1) 504 DSX$PrintSig Routine - Format Signal Array
(1) 917 FIND_USERPC Find USER PC on stack
(1) 970 TST$MChk Routine - Machine Check Handler ; [14]
(1) 1026 HARDWARE EXCEPTION ENTRY POINTS
(1) 1406 SEARCH FOR CONDITION HANDLER
(1) 1532 DSX$SETPRIEXV - ESTABLISH PRIMARY EXCEPTION VECTOR [19]

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element EXCEPT.MAR
0000     2 ; *2     5-MAY-1986 11:34:56 SALEM "DONE"
0000     3 ; *1     6-FEB-1986 15:18:23 DS ""
0000     4 ; DEC/CMS REPLACEMENT HISTORY, Element EXCEPT.MAR
0000     5         .TITLE  EXCEPT  *** EXCEPT Machine Exception handling
0000     6         .IDENT  /07-34/
0000     7         .NoShow Conditionals            ;                       [18]
0000     8         .DSABL  GBL
0000     9
0000    10 ;++
0000    11 ; Copyright (c) 1977, 1982, 1983, 1984, 1985
0000    12 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    13 ;
0000    14 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    15 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    16 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    17 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    18 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    19 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    20 ; REMAIN IN DEC.
0000    21 ;
0000    22 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    23 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    24 ; CORPORATION.
0000    25 ;
0000    26 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    27 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    28 ;--
```

```
0000     30 ;++
0000     31 ; FACILITY:
0000     32 ;
0000     33 ; ABSTRACT:
0000     34 ;
0000     35 ; ENVIRONMENT:
0000     36 ;
0000     37 ; AUTHOR:        ROGER RIGGS     15-AUG-78       VERSION 01.
0000     38 ;
0000     39 ; MODIFICATIONS:
0000     40 ;                Roger Riggs     07-FEB-79       VERSION 01. (EVSAA 5.00)
0000     41 ;       01       Added conversion of T-bit trap in compatability mode
0000     42 ;                to compatability mode trap with code=7.
0000     43 ;       02       Fixed lower limit of user stack in SEARCH routine
0000     44 ;       03       Removed ISTK/KSTK Switching stuff.  Interrupts occur on
0000     45 ;                the correct stack as indicated by the SCB
0000     46 ;       04       Fixed up position of carriage prior to calling CLI.
0000     47 ;                Make sure its on a new line at the left margin
0000     48 ;
0000     49 ;                Roger Riggs     11-Sept-1979
0000     50 ;       05       Added TST$MCHK Machine check entry point for
0000     51 ;                testing the existence of a memory location
0000     52 ;
0000     53 ;                Roger Riggs     29-OCT-1979
0000     54 ;       06       More cleanup and added DS$PRINTSIG, which can
0000     55 ;                be used by the program to print the signal array of an
0000     56 ;                exception
0000     57 ;
0000     58 ;                Roger Riggs, 14-Feb-1980
0000     59 ;       07       Added QIO to cancel Control O before printing exception info
0000     60 ;
0000     61 ;                Roger Riggs, 21-Feb-1980
0000     62 ;       08       Added duplicate defs for entry points to satisify internal debugger
0000     63 ;                Also added code to revert to old handler for arithmetic traps
0000     64 ;                and faults.
0000     65 ;
0000     66 ;                Roger Riggs, 16-Apr-1980
0000     67 ;       09       Corrected bug that reports the incorrect value in R3 during
0000     68 ;                exception call to CLI
0000     69 ;
0000     70 ;                Roger Riggs, 21-Jun-1980, Version 5.5
0000     71 ;       10       Changed DSX$PRINTSIG to use PRINTB for First line
0000     72 ;                and PRINTX for info lines.  Added new entry point
0000     73 ;                DSX$PRINTSIGI that uses PRINTI for all output.
0000     74 ;
0000     75 ;                Dave Butenhof, 11-sep-1980, Version 6.0
0000     76 ;       11       Fixed link-time truncation errors in module (change
0000     77 ;                word-relative to long-relative)
0000     78 ;
0000     79 ;                Roger Riggs, 17-Sep-1980
0000     80 ;       12       Added Type and VA to error typeout for translation not
0000     81 ;
0000     82 ;                Dave Butenhof, 23-apr-1981, version 6.4
0000     83 ;       13       Correct spelling in some messages, change to mixed case.
0000     84 ;
0000     85 ;       14       - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     86 ;                Changed SEP Psect to CODE. Fixed truncation errors.
```

```
0000  ;                 Also added .LIBRARY statements for $DS and $DIAG.
0000  ;
0000  ;        15       - Dave Butenhof, 16-Nov-1981, Version 6.5
0000  ;                 Split the FIND_USERPC routine into two--one will actually
0000  ;                 find the last user PC and return the address (FIND_USER_PC),
0000  ;                 while the other (FIND_USERPC) will call the first, then
0000  ;                 format and type the response.
0000  ;
0000  ;        16       - Dave Butenhof, 17-Dec-1981, Version 6.6
0000  ;                 Correct a bug in TST$MCHK which seems to have been here all
0000  ;                 along--it clears off machine check logout incorrectly.
0000  ;
0000  ;        17       - Dave Butenhof, 09-Mar-1982, version 6.6
0000  ;                 Fix PRINTSIG to know about SS$_MCHECK code...new for V3a.
0000  ;
0000  ;        18       - Jack Stansbury, 23-Mar-1982, Version 6.8
0000  ;                 Added typecoding to all the messages. Also made a few minor
0000  ;                 enhancements (e.g., taking out the .LIST statements, changing
0000  ;                 the text in the .SBTTL's, etc.).
0000  ;
0000  ;        19       - Richard Brown, 27-May-82, Version 6.8
0000  ;                 Addition of routine DSX$SETPRIEXV, which is used to establish
0000  ;                 a primary exception vector.
0000  ;
0000  ;        20       Jack Stansbury, 8-Mar-1983, Version 6.11
0000  ;                 Took out the Find_User_PC routine and put it into the new
0000  ;                 CallFrame.B32 module.
0000  ;
0000  ;        21       Bob Bergazzi    5-May-1983      Version 6.11
0000  ;                 Added array single bit error interrupt (vector 064) reporting for
0000  ;                 VENUS processor.
0000  ;
0000  ;        22       Bob Bergazzi    Feb. 3, 1984    Version 6.14
0000  ;                 Changed FMT_EXCEPT and FMT_UNEXPINT to say ..."SCB vector"
0000  ;                 instead of ..."vector". This makes it clearer to some people.
0000  ;
0000  ;        23       Domenic Andella 8-Mar-1984      Version 6.14
0000  ;                 Remove ASBE handler from DSX$PRINTSIG routine to
0000  ;                 MCHK8600 module, routine ASBE_WORK.
0000  ;
0000  ;        24       Domenic Andella 1-May-1984      Version 7.0
0000  ;                 Include FAULTCLEAR selection macro, and push/pop
0000  ;                 of selection code to each call to DSR$FAULT_CLEAR in
0000  ;                 routines EXE_MCHK, and TST$MCHK. This allows cpu
0000  ;                 and routine dependent action to occur.
0000  ;
0000  ;        25       M. Baggett      31-1-1985       Version 8.1
0000  ;                 Added call to do pre-processing to signal/mechanism arrays
0000  ;                 for AURORA trap/interrupt/fault/exception.
0000  ;
0000  ;        26       M. Baggett      7-MAR-1985      Version 8.1
0000  ;                 Fixed truncation error.
0000  ;
0000  ;        27       M. Baggett      12-Mar-1985     Version 8.1
0000  ;                 Added label for Aurora exceptions.
0000  ;
0000  ;        28       Domenic Andella 4-Apr-1985      Version 8.2
```

```
0000    144 ;               Mods to COND_HANDL to call
0000    145 ;               MP$_STOP_EVERYTHING if mp system.
0000    146 ;               *** have commented out this code until able
0000    147 ;               *** to debug.
0000    148 ;
0000    149 ;       29      Bob Bergazzi    18-Apr-1985     Version 8.2
0000    150 ;               Print out node # and Bus Error register for
0000    151 ;               SCORPIO systems on unexpected interrupts.
0000    152 ;
0000    153 ;       30      Bob Bergazzi    18-Apr-1985     Version 8.2
0000    154 ;               CHange CLRL DS$GB_TYPECODE to CLRB DS$GB_TYPECODE.
0000    155 ;               No symptom connected with this bug.
0000    156 ;
0000    157 ;       31      Bob Bergazzi    22-Apr-1985     Version 8.2
0000    158 ;               Fix edit 29 to check for vectors 100-1FC.
0000    159 ;
0000    160 ;       32      Ted Salem       23-April-1985   Version 8.2
0000    161 ;               Interlocked COND_HANDLER.
0000    162 ;
0000    163 ;       33      Bob Bergazzi    25-Apr-1985     Version 8.2
0000    164 ;               Clear BER for edit 29.
0000    165 ;
0000    166 ;       34      Domenic Andella 22-May-1985     Version 8.2
0000    167 ;               Add mods to COND_HANDLR to support multi-
0000    168 ;               processing. Note, mod 28 is changed as a
0000    169 ;               result of this mod.
0000    170 ;                                                               ;G:2
0000    171 ;       35      Ted Salem       4-May-1986      Version 10.0    ;G:2
0000    172 ;               Changed to check to MicroVax SID instead of     ;G:2
0000    173 ;               just Aurora                                     ;G:2
0000    174 ;--
```

```
                     0000   176          .SBTTL  Libraries, External & Equated Symbols, Macros
                     0000   177 ;
                     0000   178 ;       INCLUDE FILES:
                     0000   179 ;
                     0000   180
                     0000   181          .Library        /Sys$Library:Lib/        ;                                       [15]
                     0000   182          .Library        /$DS/                   ;                                       [14]
                     0000   183          .Library        /$Diag/                 ;                                       [14]
                     0000   184
                     0000   185 ;
                     0000   186 ; EXTERNAL SYMBOLS
                     0000   187 ;
                     0000   188
                     0000   189          .WEAK   MP$R_ACTIVE_ROUTINES,MP$R_ROUTINE_COUNT_TABLE                   ;[32]
                     0000   190          .WEAK   MP$R_ROUTINE_ID_TABLE,MP$R_GET_PROCESSOR_ID                     ;[32]
                     0000   191          .WEAK   VAX$MODIFY_EXCEPTION, FMT_BER, BI_BER_BIT                       ;[29]
                     0000   192          .WEAK   MP$GL_AP_SIGNAL_ARRAY,MP$_COND_HANDLR                           ;[34]
                     0000   193          .WEAK   MP$GR_SAVED_GPRS, MP$GL_SAVED_PSL                               ;[34]
                     0000   194
                     0000   195          .EXTRN  DSX$PRINTI,     DS_ERRSUP,      DS$ABORT,       BPTMAX
                     0000   196          .EXTRN  DS$AA_BPTADDR,  DS$GA_BREAKVEC, DS$GA_TBITVEC,  DEBUG$GB_FLAGS
                     0000   197          .EXTRN  CHR$MCHK,       DS$CLI,         COND_DEBUG,     SYS$UNWIND
                     0000   198          .EXTRN  MODELST,        APSLMNE,        DS$CVTREG,      USTKPTR
                     0000   199          .EXTRN  PCB_BASE,       DS$GL_FLAGS,    STACK_BASE,     DSX$PRINTB
                     0000   200          .EXTRN  SYS$CALL_HANDL, DSR$FAULT_CLEAR,SCRIPT$STOP,    DSX$PRINTX
                     0000   201          .EXTRN  ISTKPTR,        ESTKPTR,        SSTKPTR,        IO$M_CANCTRLO
                     0000   202          .EXTRN  IO$_WRITEVBLK,  DS$GW_TTOUT,    DS$GB_TypeCode                  ;[18
                     0000   203          .EXTRN  SCB_IMAGE,      DSR$Find_User_PC,VAX$MODIFY_EXCEPTION, EXE$GB_CPUTYP
                     0000   204          .EXTRN DS$SERVICE_DISPATCHER
                     0000   205          .EXTRN  MP$GR_BOOTED_PROCESSORS, DS$GB_PRIMARY_CPU_IDENTIFIER ;[34]
                     0000   206          .EXTRN  MP$_RESUME_ATTACHED_PROCESSORS, MP$_GET_PROCESSOR_ID  ;[34]
                     0000   207
                     0000   208 ;       EQUATED SYMBOLS:
                     0000   209 ;
                     0000   210
                     0000   211          $DS_TypeDef                             ; Define typecodes                      [18]
                     0000   212          $DS_CVTREG_DEF
                     0000   213          $DS_DSDEF
                     0000   214          $PRDEF
                     0000               .SAVE   LOCAL_BLOCK
                     0000               .RESTORE
                     0000   215          $PSLDEF
                     0000               .SAVE   LOCAL_BLOCK
                     0000               .RESTORE
                     0000   216          $DS_SCBDEF
                     0000   217          $SSDEF
                     0000               .SAVE   LOCAL_BLOCK
                     0000               .PSECT  $ABS$,ABS
                00000000    0200               .=0
                     0000               .RESTORE
                     0000   218          CLIDEF
                     0000   219          CMKDEF
                     0000   220          DSFDEF
                     0000   221          APTDEF
                     0000   222          $MP_ILDEFS                              ; interlock literal defs.
                     0000   223          $DS_DSADEF
                     0000   224          $CHFDEF
```

```
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
                00000000  FE70          .=0
                    0000                .RESTORE
                    0000    225         $SSDEF
                    0000    226         $BIICDEF                                        ; BIIC DEFINITIONS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
                00000000  FE70          .=0
                    0000                .RESTORE
                    0000    227         FLTCLR_SEL
                00000001    0000    228 IS      = 1
                00000100    0000    229 BUFSIZ  = 256                                   ;
                    0000    230
```

```
                        0000   232          .SBTTL  Work Psect Declarations
                    00000000   233          .PSECT  Work, NoExe, NoShr, Wrt, Long                    ;              [14]
                        0000   234
00000004                0000   235 NEW_SP:          .BLKL   1
                        0004   236
0000000C                0004   237 NEW_AP_FP:       .BLKQ   1
                        000C   238
00000014                000C   239 NEW_PC_PSL:      .BLKQ   1
                        0014   240
00000000                0014   241 ID_TEMP:         .LONG   0                                        ; Storage for cpu id
```

```
                                0018   243          .SBTTL  Data Psect Declarations
                            00000000   244          .PSECT  Data, NoExe, Shr, NoWrt, Long                    ;                [14]
                                0000   245
                                0000   246          MODNAM  EXCEPT
                  54 50 45 43 58 45 00' 0000      $MODULE:        .ASCIC  \EXCEPT\
                                06  0000
                                0007   247
                                0007   248 CTL$AQ_EXCVEC:   ; PRIMARY,SECONDARY EXCEPTION VECTORS FOR EXCEPTION HANDLERS
                  00000000'00000000' 0007   249          .ADDRESS        0,COND_DEBUG    ; KERNEL EXCEPTION HANDLER
                  00000000'00000000' 000F   250          .ADDRESS        0,COND_DEBUG    ; EXEC   EXCEPTION HANDLER
                  00000000'00000000' 0017   251          .ADDRESS        0,COND_DEBUG    ; SUPER  EXCEPTION HANDLER
                  00000000'00000000' 001F   252          .ADDRESS        0,COND_DEBUG    ; USER   EXCEPTION HANDLER
                                0027   253
                                0027   254 CTL$AL_STACK:    ; ADDRESS OF TOP OF EACH MODE STACK INDEX BY LONGWORD MODE
00000000'00000000'00000000'00000000' 0027   255          .ADDRESS        ISTKPTR,ESTKPTR,SSTKPTR,USTKPTR
```

```
                     0037   257 ;+
                     0037   258 ; These are the messages output by the exception printing routines. These do    [18]
                     0037   259 ; not need to contain typecodes since the typecode will be set explicitly by    [18]
                     0037   260 ; the routines, and the appropriate print routine will be called (rather than   [18]
                     0037   261 ; calling Print).                                                               [18]
                     0037   262 ;-
                     0037   263
                     0037   264 Msg_UMchk:      $PrintI_L       Fmt_UMchk               ;                      [17]
00000001             0037                      .LONG   $$N
00000383'            003B                      .LONG   Fmt_UMchk
                     003F   265 MSG_KRNLSTK:    $PRINTI_L       FMT_EXCEPT, T_KRNLSTK, T_ABORT, <<^X08>>
00000004             003F                      .LONG   $$N
000003BD'            0043                      .LONG   FMT_EXCEPT
00000581'            0047                      .LONG   T_KRNLSTK
000006D0'            004B                      .LONG   T_ABORT
00000008             004F                      .LONG   ^X08
                     0053   266 MSG_POWER:     $PRINTI_L       FMT_EXCEPT, T_POWER, T_INTRPT, <<^X0C>>
00000004             0053                      .LONG   $$N
000003BD'            0057                      .LONG   FMT_EXCEPT
00000598'            005B                      .LONG   T_POWER
000006D6'            005F                      .LONG   T_INTRPT
0000000C             0063                      .LONG   ^X0C
                     0067   267 MSG_OPCDEC:     $PRINTI_L       FMT_EXCEPT, T_OPCDEC, T_FAULT, <<^X10>>
00000004             0067                      .LONG   $$N
000003BD'            006B                      .LONG   FMT_EXCEPT
000005A3'            006F                      .LONG   T_OPCDEC
000006CA'            0073                      .LONG   T_FAULT
00000010             0077                      .LONG   ^X10
                     007B   268 MSG_OPCCUS:     $PRINTI_L       FMT_EXCEPT, T_OPCCUS, T_FAULT, <<^X14>>
00000004             007B                      .LONG   $$N
000003BD'            007F                      .LONG   FMT_EXCEPT
000005C3'            0083                      .LONG   T_OPCCUS
000006CA'            0087                      .LONG   T_FAULT
00000014             008B                      .LONG   ^X14
                     008F   269 MSG_ROPRAND:    $PRINTI_L       FMT_EXCEPT, T_ROPRAND, T_FLT_ABO, <<^X18>>
00000004             008F                      .LONG   $$N
000003BD'            0093                      .LONG   FMT_EXCEPT
000005E1'            0097                      .LONG   T_ROPRAND
000006E0'            009B                      .LONG   T_FLT_ABO
00000018             009F                      .LONG   ^X18
                     00A3   270 MSG_RADRMOD:    $PRINTI_L       FMT_EXCEPT, T_RADRMOD, T_FAULT, <<^X1C>>
00000004             00A3                      .LONG   $$N
000003BD'            00A7                      .LONG   FMT_EXCEPT
000005F2'            00AB                      .LONG   T_RADRMOD
000006CA'            00AF                      .LONG   T_FAULT
0000001C             00B3                      .LONG   ^X1C
                     00B7   271 MSG_ACCVIO:     $PRINTI_L       FMT_EXCEPT, T_ACCVIO, T_FAULT, <<^X20>>
00000004             00B7                      .LONG   $$N
000003BD'            00BB                      .LONG   FMT_EXCEPT
0000060B'            00BF                      .LONG   T_ACCVIO
000006CA'            00C3                      .LONG   T_FAULT
00000020             00C7                      .LONG   ^X20
                     00CB   272 MSG_TRANSL:     $PRINTI_L       FMT_EXCEPT, T_TRANSL, T_FAULT, <<^X24>>
00000004             00CB                      .LONG   $$N
000003BD'            00CF                      .LONG   FMT_EXCEPT
00000624'            00D3                      .LONG   T_TRANSL
000006CA'            00D7                      .LONG   T_FAULT
```

```
                    00000024  00DB                    .LONG   ^X24
                              00DF  273 MSG_TBIT:       $PRINTI_L       FMT_EXCEPT, T_TBIT, T_TRAP, <<^X28>>
                    00000004  00DF                    .LONG   $$N
                    000003BD' 00E3                    .LONG   FMT_EXCEPT
                    0000063A' 00E7                    .LONG   T_TBIT
                    000006C5' 00EB                    .LONG   T_TRAP
                    00000028  00EF                    .LONG   ^X28
                              00F3  274 MSG_BREAK:      $PRINTI_L       FMT_EXCEPT, T_BREAK, T_FAULT, <<^X2C>>
                    00000004  00F3                    .LONG   $$N
                    000003BD' 00F7                    .LONG   FMT_EXCEPT
                    00000640' 00FB                    .LONG   T_BREAK
                    000006CA' 00FF                    .LONG   T_FAULT
                    0000002C  0103                    .LONG   ^X2C
                              0107  275 MSG_COMPAT:     $PRINTI_L       FMT_EXCEPT, T_COMPAT, T_FLT_ABO, <<^X30>>
                    00000004  0107                    .LONG   $$N
                    000003BD' 010B                    .LONG   FMT_EXCEPT
                    000006EC' 010F                    .LONG   T_COMPAT
                    000006E0' 0113                    .LONG   T_FLT_ABO
                    00000030  0117                    .LONG   ^X30
                              011B  276 MSG_CHMK:       $PRINTI_L       FMT_EXCEPT, T_CHMK, T_TRAP, <<^X40>>
                    00000004  011B                    .LONG   $$N
                    000003BD' 011F                    .LONG   FMT_EXCEPT
                    0000064B' 0123                    .LONG   T_CHMK
                    000006C5' 0127                    .LONG   T_TRAP
                    00000040  012B                    .LONG   ^X40
                              012F  277 MSG_CHME:       $PRINTI_L       FMT_EXCEPT, T_CHME, T_TRAP, <<^X44>>
                    00000004  012F                    .LONG   $$N
                    000003BD' 0133                    .LONG   FMT_EXCEPT
                    0000065E' 0137                    .LONG   T_CHME
                    000006C5' 013B                    .LONG   T_TRAP
                    00000044  013F                    .LONG   ^X44
                              0143  278 MSG_CHMS:       $PRINTI_L       FMT_EXCEPT, T_CHMS, T_TRAP, <<^X48>>
                    00000004  0143                    .LONG   $$N
                    000003BD' 0147                    .LONG   FMT_EXCEPT
                    00000674' 014B                    .LONG   T_CHMS
                    000006C5' 014F                    .LONG   T_TRAP
                    00000048  0153                    .LONG   ^X48
                              0157  279 MSG_CHMU:       $PRINTI_L       FMT_EXCEPT, T_CHMU, T_TRAP, <<^X4C>>
                    00000004  0157                    .LONG   $$N
                    000003BD' 015B                    .LONG   FMT_EXCEPT
                    0000068B' 015F                    .LONG   T_CHMU
                    000006C5' 0163                    .LONG   T_TRAP
                    0000004C  0167                    .LONG   ^X4C
                              016B  280
```

```
                                0000018B' 016B   282 AAT_ARITH::      .ADDRESS        0$
                                0000019B' 016F   283                  .ADDRESS        1$
                                000001AC' 0173   284                  .ADDRESS        2$
                                000001C3' 0177   285                  .ADDRESS        3$
                                000001D5' 017B   286                  .ADDRESS        4$
                                000001ED' 017F   287                  .ADDRESS        5$
                                00000200' 0183   288                  .ADDRESS        6$
                                00000211' 0187   289                  .ADDRESS        7$
                                          018B   290
6E 65 70 20 50 41 52 54 20 6F 4E 00' 018B   291 0$:      .ASCIC  .No TRAP pending.              ;                    [13]
                         67 6E 69 64  0197
                                   0F  018B
65 76 6F 20 72 65 67 65 74 6E 49 00' 019B   292 1$:      .ASCIC  .Integer overflow.             ;                    [13]
                      77 6F 6C 66 72  01A7
                                   10  019B
76 69 64 20 72 65 67 65 74 6E 49 00' 01AC   293 2$:      .ASCIC  .Integer divide by zero.       ;                    [13]
   6F 72 65 7A 20 79 62 20 65 64 69  01B8
                                   16  01AC
76 6F 20 67 6E 69 74 61 6F 6C 46 00' 01C3   294 3$:      .ASCIC  .Floating overflow.            ;                    [13]
                   77 6F 6C 66 72 65  01CF
                                   11  01C3
69 64 20 67 6E 69 74 61 6F 6C 46 00' 01D5   295 4$:      .ASCIC  .Floating divide by zero.      ;                    [13]
6F 72 65 7A 20 79 62 20 65 64 69 76  01E1
                                   17  01D5
6E 75 20 67 6E 69 74 61 6F 6C 46 00' 01ED   296 5$:      .ASCIC  .Floating underflow.           ;                    [13]
                77 6F 6C 66 72 65 64  01F9
                                   12  01ED
65 76 6F 20 6C 61 6D 69 63 65 44 00' 0200   297 6$:      .ASCIC  .Decimal overflow.             ;                    [13]
                      77 6F 6C 66 72  020C
                                   10  0200
76 69 64 20 6C 61 6D 69 63 65 44 00' 0211   298 7$:      .ASCIC  .Decimal divide by zero.       ;                    [13]
   6F 72 65 7A 20 79 62 20 65 64 69  021D
                                   16  0211
```

```
                    0228   300 MSG_INTOVF:     $PRINTI_L       FMT_EXCEPT, T_INTOVF, T_TRAP, <<^X34>>
00000004  0228                                 .LONG   $$N
000003BD' 022C                                 .LONG   FMT_EXCEPT
0000071F' 0230                                 .LONG   T_INTOVF
000006C5' 0234                                 .LONG   T_TRAP
00000034  0238                                 .LONG   ^X34
                    023C   301 MSG_INTDIV:     $PRINTI_L       FMT_EXCEPT, T_INTDIV, T_TRAP, <<^X34>>
00000004  023C                                 .LONG   $$N
000003BD' 0240                                 .LONG   FMT_EXCEPT
00000730' 0244                                 .LONG   T_INTDIV
000006C5' 0248                                 .LONG   T_TRAP
00000034  024C                                 .LONG   ^X34
                    0250   302 MSG_FLTOVF:     $PRINTI_L       FMT_EXCEPT, T_FLTOVF, T_TRAP, <<^X34>>
00000004  0250                                 .LONG   $$N
000003BD' 0254                                 .LONG   FMT_EXCEPT
00000747' 0258                                 .LONG   T_FLTOVF
000006C5' 025C                                 .LONG   T_TRAP
00000034  0260                                 .LONG   ^X34
                    0264   303 MSG_FLTDIV:     $PRINTI_L       FMT_EXCEPT, T_FLTDIV, T_TRAP, <<^X34>>
00000004  0264                                 .LONG   $$N
000003BD' 0268                                 .LONG   FMT_EXCEPT
00000759' 026C                                 .LONG   T_FLTDIV
000006C5' 0270                                 .LONG   T_TRAP
00000034  0274                                 .LONG   ^X34
                    0278   304 MSG_FLTUND:     $PRINTI_L       FMT_EXCEPT, T_FLTUND, T_TRAP, <<^X34>>
00000004  0278                                 .LONG   $$N
000003BD' 027C                                 .LONG   FMT_EXCEPT
00000779' 0280                                 .LONG   T_FLTUND
000006C5' 0284                                 .LONG   T_TRAP
00000034  0288                                 .LONG   ^X34
                    028C   305 MSG_DECOVF:     $PRINTI_L       FMT_EXCEPT, T_DECOVF, T_TRAP, <<^X34>>
00000004  028C                                 .LONG   $$N
000003BD' 0290                                 .LONG   FMT_EXCEPT
0000078C' 0294                                 .LONG   T_DECOVF
000006C5' 0298                                 .LONG   T_TRAP
00000034  029C                                 .LONG   ^X34
                    02A0   306 MSG_SUBRNG:     $PRINTI_L       FMT_EXCEPT, T_SUBRNG, T_TRAP, <<^X34>>
00000004  02A0                                 .LONG   $$N
000003BD' 02A4                                 .LONG   FMT_EXCEPT
0000079D' 02A8                                 .LONG   T_SUBRNG
000006C5' 02AC                                 .LONG   T_TRAP
00000034  02B0                                 .LONG   ^X34
                    02B4   307 MSG_FLTOVF_F:   $PRINTI_L       FMT_EXCEPT, T_FLTOVF, T_FAULT, <<^X34>>
00000004  02B4                                 .LONG   $$N
000003BD' 02B8                                 .LONG   FMT_EXCEPT
00000747' 02BC                                 .LONG   T_FLTOVF
000006CA' 02C0                                 .LONG   T_FAULT
00000034  02C4                                 .LONG   ^X34
                    02C8   308 MSG_FLTDIV_F:   $PRINTI_L       FMT_EXCEPT, T_FLTDIV, T_FAULT, <<^X34>>
00000004  02C8                                 .LONG   $$N
000003BD' 02CC                                 .LONG   FMT_EXCEPT
00000759' 02D0                                 .LONG   T_FLTDIV
000006CA' 02D4                                 .LONG   T_FAULT
00000034  02D8                                 .LONG   ^X34
                    02DC   309 MSG_FLTUND_F:   $PRINTI_L       FMT_EXCEPT, T_FLTUND, T_FAULT, <<^X34>>
00000004  02DC                                 .LONG   $$N
000003BD' 02E0                                 .LONG   FMT_EXCEPT
```

```
                    00000779' 02E4          .LONG   T_FLTUND
                    000006CA' 02E8          .LONG   T_FAULT
                    00000034  02EC          .LONG   ^X34
```

```
                        00000310' 02F0   311 AAT_COMPAT:     .ADDRESS        0$              ; RESERVED OPCODE FAULT
                        00000326' 02F4   312                 .ADDRESS        1$              ; BPT FAULT
                        00000330' 02F8   313                 .ADDRESS        2$              ; IOT FAULT
                        0000033A' 02FC   314                 .ADDRESS        3$              ; EMT FAULT
                        00000344' 0300   315                 .ADDRESS        4$              ; TRAP FAULT
                        0000034F' 0304   316                 .ADDRESS        5$              ; ILLEGAL INSTRUCTION FAULT
                        00000369' 0308   317                 .ADDRESS        6$              ; ODD ADDRESS ABORT
                        0000037B' 030C   318                 .ADDRESS        7$              ; UNUSED
70 6F 20 64 65 76 72 65 73 65 52 00' 0310   319 0$:          .ASCIC  .Reserved opcode fault. ;                       [13]
      74 6C 75 61 66 20 65 64 6F 63  031C
                                 15  0310
      74 6C 75 61 66 20 54 50 42 00' 0326   320 1$:          .ASCIC  .BPT fault.             ;                       [13]
                                 09  0326
      74 6C 75 61 66 20 54 4F 49 00' 0330   321 2$:          .ASCIC  .IOT fault.             ;                       [13]
                                 09  0330
      74 6C 75 61 66 20 54 4D 45 00' 033A   322 3$:          .ASCIC  .EMT fault.             ;                       [13]
                                 09  033A
   74 6C 75 61 66 20 50 41 52 54 00' 0344   323 4$:          .ASCIC  .TRAP fault.            ;                       [13]
                                 0A  0344
73 6E 69 20 6C 61 67 65 6C 6C 49 00' 034F   324 5$:          .ASCIC  .Illegal instruction fault.     ;               [13]
75 61 66 20 6E 6F 69 74 63 75 72 74  035B
                              74 6C  0367
                                 19  034F
73 73 65 72 64 64 61 20 64 64 4F 00' 0369   325 6$:          .ASCIC  .Odd address abort.     ;                       [13]
                  74 72 6F 62 61 20  0375
                                 11  0369
            6E 77 6F 6E 6B 6E 55 00' 037B   326 7$:          .ASCIC  .Unknown.               ;                       [13]
                                 07  037B
```

```
61 6D 20 53 4D 56 20 3F 3F 2F 21 00' 0383   328 Fmt_UMchk:       .AsciC  '!/?? VMS machine check (hardware' -    ;              [17]
20 6B 63 65 68 63 20 65 6E 69 68 63  038F
6F 6C 20 65 72 61 77 64 72 61 68 28  039B
61 76 61 20 74 6F 6E 20 74 75 6F 67  03A7
      2F 21 3A 29 65 6C 62 61 6C 69  03B3
                                 39  0383
                                     03BD   329                             ' logout not available):!/'             ;              [17]
41 21 20 43 41 21 20 3F 3F 2F 21 00' 03BD   330 FMT_EXCEPT:      .ASCIC  '!/?? !AC !AC through SCB vector: !XB(X)!/'    ;       [22]
43 53 20 68 67 75 6F 72 68 74 20 43  03C9
58 21 20 3A 72 6F 74 63 65 76 20 42  03D5
               2F 21 29 58 28 42 42  03E1
                                 29  03BD
65 70 78 65 6E 55 20 3F 3F 2F 21 00' 03E7   331 FMT_UNEXPINT:    .ASCIC  '!/?? Unexpected trap or interrupt thru SCB vector !XW!/' ;[
72 6F 20 70 61 72 74 20 64 65 74 63  03F3
74 20 74 70 75 72 72 65 74 6E 69 20  03FF
74 63 65 76 20 42 43 53 20 75 72 68  040B
            2F 21 57 58 21 20 72 6F  0417
                                 37  03E7
65 70 78 65 6E 55 20 3F 3F 2F 21 00' 041F   332 FMT_BI_UNEXPINT: .ASCIC '!/?? Unexpected trap or interrupt thru SCB vector !XW from
72 6F 20 70 61 72 74 20 64 65 74 63  042B
74 20 74 70 75 72 72 65 74 6E 69 20  0437
74 63 65 76 20 42 43 53 20 75 72 68  0443
20 6D 6F 72 66 20 57 58 21 20 72 6F  044F
      2F 21 42 58 21 20 65 64 6F 6E  045B
                                 45  041F
72 6F 72 72 65 20 74 61 20 43 50 00' 0465   333 FMT_ERRPC:       .ASCIC  'PC at error:!_!_!XL(X)!/'              ;              [13]
21 29 58 28 4C 58 21 5F 21 5F 21 3A  0471
                                 2F  047D
                                 18  0465
6E 72 75 74 65 72 20 72 65 73 55 00' 047E   334 FMT_USRPC:       .ASCIC  'User return PC:!_!_!XL(X)!/'           ;              [13]
28 4C 58 21 5F 21 5F 21 3A 43 50 20  048A
                     2F 21 29 58 20  0496
                                 1B  047E
6E 72 75 74 65 72 20 72 65 73 55 00' 049A   335 Fmt_User_PC_NF: .Ascic  'User return PC: none found!!!/'         ;              [20]
6F 66 20 65 6E 6F 6E 20 3A 43 50 20  04A6
               2F 21 21 21 64 6E 75  04B2
                                 1E  049A
6F 72 72 65 20 74 61 20 4C 53 50 00' 04B9   336 FMT_ERRPSL:      .ASCIC  'PSL at error:!_!_!XL(X)!_; !AC!/'      ;              [13]
29 58 28 4C 58 21 5F 21 5F 21 3A 72  04C5
         2F 21 43 41 21 20 3B 5F 21  04D1
                                 20  04B9
64 64 61 20 6C 61 75 74 72 69 56 00' 04DA   337 FMT_VIRT:        .ASCIC  'Virtual address:!_!XL(X)!/Fault type:!_!_!XL(X)!/' ;   [13]
58 28 4C 58 21 5F 21 3A 73 73 65 72  04E6
70 79 74 20 74 6C 75 61 46 2F 21 29  04F2
29 58 28 4C 58 21 5F 21 5F 21 3A 65  04FE
                              2F 21  050A
                                 31  04DA
63 20 43 41 21 20 43 41 21 5F 21 00' 050C   338 FMT_EXCEP_CODE: .ASCIC  '!_!AC !AC code:!_!AC!/'               ;              [13]
   2F 21 43 41 21 5F 21 3A 65 64 6F  0518
                                 16  050C
21 5F 21 5F 21 5F 21 3A 67 72 41 00' 0523   339 FMT_CHMX_ARG:    .ASCIC  'Arg:!_!_!_!XL(X)!/'                    ;              [13]
               2F 21 29 58 28 4C 58  052F
                                 12  0523
20 6E 77 6F 6E 6B 6E 55 20 3F 3F 00' 0536   340 FMT_UNK_ARG:     .ASCIC  '?? Unknown exception/condition:!/'      ;              [13]
6F 63 2F 6E 6F 69 74 70 65 63 78 65  0542
      2F 21 3A 6E 6F 69 74 69 64 6E  054E
                                 21  0536
```

```
21 5F 21 5F 21 3A 74 6E 75 6F 43 00' 0558   341 FMT_UNK_CNT:    .ASCIC  'Count:!_!_!_!XL(X)!/'                    ;              [13]
         2F 21 29 58 28 4C 58 21 5F   0564
                                 14   0558
5F 21 5F 21 5F 21 3A 4C 55 21 50 00' 056D   342 FMT_PXXX:       .ASCIC  'P!UL:!_!_!_!XL(X)!/'
            2F 21 29 58 28 4C 58 21   0579
                                 13   056D
```

```
63 61 74 73 20 4C 45 4E 52 45 4B 00' 0581    344 T_KRNLSTK:      .ASCIC  'KERNEL stack not valid'        ;  [13]
   64 69 6C 61 76 20 74 6F 6E 20 6B  058D
                                     16  0581
   6C 69 61 66 20 72 65 77 6F 50 00' 0598    345 T_POWER:        .ASCIC  'Power fail'             ;  [13]
                                     0A  0598
72 70 2F 64 65 76 72 65 73 65 52 00' 05A3    346 T_OPCDEC:       .ASCIC  'Reserved/privileged instruction'  ;  [13]
73 6E 69 20 64 65 67 65 6C 69 76 69  05AF
            6E 6F 69 74 63 75 72 74  05BB
                                     1F  05A3
65 72 20 72 65 6D 6F 74 73 75 43 00' 05C3    347 T_OPCCUS:       .ASCIC  'Customer reserved instruction' ;  [13]
72 74 73 6E 69 20 64 65 76 72 65 73  05CF
                  6E 6F 69 74 63 75  05DB
                                     1D  05C3
70 6F 20 64 65 76 72 65 73 65 52 00' 05E1    348 T_ROPRAND:      .ASCIC  'Reserved operand'       ;  [13]
                     64 6E 61 72 65  05ED
                                     10  05E1
64 61 20 64 65 76 72 65 73 65 52 00' 05F2    349 T_RADRMOD:      .ASCIC  'Reserved addressing mode'       ;  [13]
64 6F 6D 20 67 6E 69 73 73 65 72 64  05FE
                                     65  060A
                                     18  05F2
74 6E 6F 63 20 73 73 65 63 63 41 00' 060B    350 T_ACCVIO:       .ASCIC  'Access control violation'       ;  [13]
6F 69 74 61 6C 6F 69 76 20 6C 6F 72  0617
                                     6E  0623
                                     18  060B
6E 6F 69 74 61 6C 73 6E 61 72 54 00' 0624    351 T_TRANSL:       .ASCIC  'Translation not valid' ;  [13]
      64 69 6C 61 76 20 74 6F 6E 20  0630
                                     15  0624
                  65 63 61 72 54 00' 063A    352 T_TBIT:         .ASCIC  'Trace'                  ;  [13]
                                     05  063A
   74 6E 69 6F 70 6B 61 65 72 42 00' 0640    353 T_BREAK:        .ASCIC  'Breakpoint'             ;  [13]
                                     0A  0640
65 64 6F 6D 20 65 67 6E 61 68 43 00' 064B    354 T_CHMK:         .ASCIC  'Change mode KERNEL'     ;  [13]
               4C 45 4E 52 45 4B 20  0657
                                     12  064B
65 64 6F 6D 20 65 67 6E 61 68 43 00' 065E    355 T_CHME:         .ASCIC  'Change mode EXECUTIVE' ;  [13]
      45 56 49 54 55 43 45 58 45 20  066A
                                     15  065E
65 64 6F 6D 20 65 67 6E 61 68 43 00' 0674    356 T_CHMS:         .ASCIC  'Change mode SUPERVISOR'         ;  [13]
   52 4F 53 49 56 52 45 50 55 53 20  0680
                                     16  0674
65 64 6F 6D 20 65 67 6E 61 68 43 00' 068B    357 T_CHMU:         .ASCIC  'Change mode USER'       ;  [13]
                  52 45 53 55 20     0697
                                     10  068B
20 64 65 74 63 65 70 78 65 6E 55 00' 069C    358 T_XCHFAILURE:   .ASCIC  'Unexpected status from condition handler'       ;  [13]
20 6D 6F 72 66 20 73 75 74 61 74 73  06A8
61 68 20 6E 6F 69 74 69 64 6E 6F 63  06B4
               72 65 6C 64 6E        06C0
                                     28  069C
                  70 61 72 54 00' 06C5    359 T_TRAP:         .ASCIC  'Trap'                   ;  [13]
                                     04  06C5
               74 6C 75 61 46 00' 06CA    360 T_FAULT:        .ASCIC  'Fault'                  ;  [13]
                                     05  06CA
               74 72 6F 62 41 00' 06D0    361 T_ABORT:        .ASCIC  'Abort'                  ;  [13]
                                     05  06D0
      74 70 75 72 72 65 74 6E 49 00' 06D6    362 T_INTRPT:       .ASCIC  'Interrupt'              ;  [13]
                                     09  06D6
74 72 6F 62 61 2F 74 6C 75 61 46 00' 06E0    363 T_FLT_ABO:      .ASCIC  'Fault/abort'            ;  [13]
```

```
                                  0B  06E0
69 6C 69 62 61 74 61 70 6D 6F 43 00' 06EC   364 T_COMPAT:       .ASCIC  'Compatability mode'    ;                    [13]
                65 64 6F 6D 20 79 74  06F8
                                  12  06EC
74 73 20 73 75 6F 69 76 65 72 50 00' 06FF   365 T_STKVIO:       .ASCIC  'Previous stack access violation'       ;    [13]
76 20 73 73 65 63 63 61 20 6B 63 61  070B
            6E 6F 69 74 61 6C 6F 69  0717
                                  1F  06FF
65 76 6F 20 72 65 67 65 74 6E 49 00' 071F   366 T_INTOVF:       .ASCIC  'Integer overflow'      ;                    [13]
                      77 6F 6C 66 72  072B
                                  10  071F
76 69 64 20 72 65 67 65 74 6E 49 00' 0730   367 T_INTDIV:       .ASCIC  'Integer divide by zero'        ;            [13]
   6F 72 65 7A 20 79 62 20 65 64 69  073C
                                  16  0730
76 6F 20 67 6E 69 74 61 6F 6C 46 00' 0747   368 T_FLTOVF:       .ASCIC  'Floating overflow'     ;                    [13]
                   77 6F 6C 66 72 65  0753
                                  11  0747
65 64 2F 67 6E 69 74 61 6F 6C 46 00' 0759   369 T_FLTDIV:       .ASCIC  'Floating/decimal divide by zero'       ;    [13]
65 64 69 76 69 64 20 6C 61 6D 69 63  0765
            6F 72 65 7A 20 79 62 20  0771
                                  1F  0759
6E 75 20 67 6E 69 74 61 6F 6C 46 00' 0779   370 T_FLTUND:       .ASCIC  'Floating underflow'    ;                    [13]
                77 6F 6C 66 72 65 64  0785
                                  12  0779
65 76 6F 20 6C 61 6D 69 63 65 44 00' 078C   371 T_DECOVF:       .ASCIC  'Decimal overflow'      ;                    [13]
                      77 6F 6C 66 72  0798
                                  10  078C
72 20 74 70 69 72 63 73 62 75 53 00' 079D   372 T_SUBRNG:       .ASCIC  'Subscript range'       ;                    [13]
                         65 67 6E 61  07A9
                                  0F  079D
```

```
                     07AD   374          .SBTTL  COND_HANDLR ROUTINE - HANDLE UNWANTED EXCEPTIONS
                 00000000   375          .PSECT  CODE, EXE, SHR, NOWRT, LONG
                     0000   376 ;++
                     0000   377 ; FUNCTIONAL DESCRIPTION:
                     0000   378 ;
                     0000   379 ;       This routine handles all exceptions not wanted by other handlers.
                     0000   380 ;       It could be considered a last chance exception handler and is setup
                     0000   381 ;       as such in user mode.
                     0000   382 ;
                     0000   383 ; CALLING SEQUENCE:
                     0000   384 ;
                     0000   385 ;       PROCEEDURE CALL
                     0000   386 ;
                     0000   387 ; INPUT PARAMETERS:
                     0000   388 ;
                     0000   389 ;       STANDARD MECHANISM AND SIGNAL ARRAY ARGS
                     0000   390 ;
                     0000   391 ; IMPLICIT INPUTS:
                     0000   392 ;
                     0000   393 ;       NONE
                     0000   394 ;
                     0000   395 ; OUTPUT PARAMETERS:
                     0000   396 ;
                     0000   397 ;       NONE
                     0000   398 ;
                     0000   399 ; IMPLICIT OUTPUTS:
                     0000   400 ;
                     0000   401 ;       NONE
                     0000   402 ;
                     0000   403 ; COMPLETION CODES:
                     0000   404 ;
                     0000   405 ;       SS$_CONTINUE -or-
                     0000   406 ;       SS$_RESIGNAL
                     0000   407 ;
                     0000   408 ; SIDE EFFECTS:
                     0000   409 ;
                     0000   410 ;       NONE
                     0000   411 ;--
```

```
                        000C  0000  .ENTRY  COND_HANDLR,^M<R2,R3>          ; Entry point to handle exceptions
                              0002          $MP_SET_INTERLOCK  #MP$V_IL_CNHN ; Interlock routine                [32]
                   06   DD    0002                  PUSHL   #MP$V_IL_CNHN
                   08   DD    0004                  PUSHL   #$SRVCODE_SET_INTERLOCK
      00000000'9F  02   FB    0006                  CALLS   #2, @#DS$SERVICE_DISPATCHER
                              000D
          00000000'EF  D5     000D          TSTL    MP$GR_BOOTED_PROCESSORS ; Are we an mp system?              [34]
                   0E   13    0013          BEQL    3$                      ; Not mp, therefore branch          [34]
   00000000'EF   04 BC  DE    0015          MOVAL   @4(AP),MP$GL_AP_SIGNAL_ARRAY    ; Store address of Ap's signal array
          00000000'EF  16     001D          JSB     MP$_COND_HANDLR                 ; Go to mp handler      [34]
                              0023                                                  ; Note Ap will not RSB [34]
                              0023
                              0023  3$:     $QIO_S  FUNC=#IO$_WRITEVBLK!IO$M_CANCTRLO, -
                              0023                  CHAN=DS$GW_TTOUT        ; Cancel Control-O
                   7E   7C    0023                  CLRQ    -(SP)
                   7E   7C    0025                  CLRQ    -(SP)
                   00   DD    0027                  PUSHL   #0
                   00   DD    0029                  PUSHL   #0
                   7E   7C    002B                  CLRQ    -(SP)
                   00   DD    002D                  PUSHL   #0
         7E   0000'8F  3C     002F                  MOVZWL  #IO$_WRITEVBLK!IO$M_CANCTRLO,-(SP)
      7E   00000000'EF  3C    0034                  MOVZWL  DS$GW_TTOUT,-(SP)
                   00   DD    003B                  PUSHL   #0
      00000000'GF  0C   FB    003D                  CALLS   #12,G^SYS$QIO
   0000015A'EF   04 BC  FA    0044          CALLG   @4(AP),DSX$PRINTSIGI    ; Format signal array
                03 50   E8    004C          BLBS    R0,COND_HANDLR_CLI      ; Branch if known condition
                              004F
                 00D4   31    004F          BRW     COND_HANDLR_EXIT        ; Jump to clear interlock and exit  [32]
                              0052
                              0052  COND_HANDLR_CLI:
                              0052          Br_If_User      5$              ; Branch if in user mode            [18]
      0000FE00'EF  1C   E0    0052          BBS     #DSA$V_User, -                  ; If bit set, branch        [28]
                   07         0059                    L^DSA$GL_Flags, 5$              ;                           [28]
      0000FE40'EF  0B   D0    005A          MOVL    #APM$_EXCEPT, -         ; Tell APT about the error
                              0061                    DSA$GL_MSGTYP
                              0061
      00000000'EF  0C   90    0061  5$:     MovB    #DS$K_Type_Exception, - ; Set typecode                      [18]
                              0068                    L^DS$GB_TypeCode        ;                                   [18]
          04A0'CF  6C   FA    0068          CALLG   (AP),W^FIND_USERPC      ; Locate and print PC<10000
          00000000'EF  94     006D          ClrB    L^DS$GB_TypeCode        ; Clear typecode                    [30]
                              0073
                              0073  ;+
                              0073  ; INFORMATION ABOUT THE UNEXPECTED CONDITION HAS BEEN PRINTED.
                              0073  ; CALL CLI TO ALLOW THE OPERATOR TO EXAMINE THE SITUATION.
                              0073  ; DETERMINE THE STACK POINTER AT THE TIME OF THE INTERRUPT
                              0073  ;-
                              0073
            50   04 BC  DE    0073          MOVAL   @4(AP),R0               ; ADDRESS OF SIGNAL ARRAY
               51   60  D0    0077          MOVL    (R0),R1                 ; GET COUNT OF ARGS
         50   FC A041  DE     007A          MOVAL   -4(R0)[R1],R0           ; POINT AT PC/PSL PAIR
            53   08 A0  DE    007F          MOVAL   8(R0),R3                ; Stack pointer before exception
         22 04 A0   1A  E0    0083          BBS     #PSL$V_IS,4(R0), 10$    ; IF IS WAS SET THAT WAS CORRECT
                              0088          Br_If_User      10$             ; Branch if in user mode            [18]
      0000FE00'EF  1C   E0    0088          BBS     #DSA$V_User, -                  ; If bit set, branch        [28]
                   1A         008F                    L^DSA$GL_Flags, 10$             ;                           [28]
                              0090
   51   04 A0   02   18  EF  0090          EXTZV   #PSL$V_CURMOD, #PSL$S_CURMOD, -
```

```
                                0096                    4(R0),R1                ; GET PREVIOUS MODE
                     52   DC    0096          MOVPSL  R2                      ; Get current PSL
          52   02    18   ED    0098          CMPZV   #PSL$V_CURMOD,#PSL$S_CURMOD, -
                     51         009C                    R2,R1                   ; Same mode as previous?
                     0B   13    009D          BEQL    10$                     ; Use stack pointer in R3 if so
     53   00000000'EF41   D0    009F          MOVL    PCB_BASE[R1],R3         ; Get SP from PCB
               53    51   DB    00A7          MFPR    R1,R3                   ; Get implementation dependent SP
                                00AA
               7E    60   7D    00AA   10$:   MOVQ    (R0),-(SP)              ; STACK PC/PSL
                     08   BB    00AD          PUSHR   #^MR3                   ; Push exception SP
            7E   08 AD   7D    00AF          MOVQ    8(FP),-(SP)             ; PUSH AP/FP
               0FF0 8F   BB    00B3          PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11>; SAVE R4-R11
            7E   14 AD   7D    00B7          MOVQ    20(FP),-(SP)            ; Push R2 and R3
            50   08 BC   DE    00BB          MOVAL   @8(AP),R0               ; POINT TO MECHANISM ARGS
            7E   0C A0   7D    00BF          MOVQ    12(R0),-(SP)            ; SAVE R0-R1
                                00C3
                                00C3          Set_CmdFlg                      ; Set command mode                  [18]
     00000000'EF    07   E2    00C3          BBSS    #DS$V_CmdFlg, -         ; Branch if CMDFLG bit is set to    [29]
                     00         00CA                    L^DS$GL_Flags, 30000$ ; ...                               [29]
                                00CB   30000$:                                ; ... here. Set bit regardless.     [29]
                                00CB          Set_Except                      ; Set Exception bit                 [18]
     00000000'EF    13   E2    00CB          BBSS    #DS$V_Except, -         ; Branch if EXCEPT bit is set to    [29]
                     00         00D2                    L^DS$GL_Flags, 30001$ ; ...                               [29]
                                00D3   30001$:                                ; ... here. Set bit regardless.     [29]
                                00D3
     0000FE40'EF    01   D0    00D3          MOVL    #APM$_SYSERR, -
                                00DA                    L^DSA$GL_MSGTYP         ; Inform APT of error
          00000000'EF    16    00DA          JSB     SCRIPT$STOP             ; GET OUT OF SCRIPT MODE
                                00E0
     00000000'EF    6E   FA    00E0   15$:   CALLG   (SP),L^DS$CLI           ; PAUSE FOR OPERATOR                [14]
                                00E7
          00000000'EF    D5    00E7          TSTL    MP$GR_BOOTED_PROCESSORS ; Are we an mp system?              [34]
                     07   13    00ED          BEQL    17$                     ; Not mp, therefore branch          [34]
     00000000'EF    00   FB    00EF          CALLS   #0,MP$_RESUME_ATTACHED_PROCESSORS ; Call mp code            [34]
                                00F6
            50   04 AC   7D    00F6   17$:   MOVQ    4(AP),R0                ; POINT TO SIGNAL/MECHANISM ARRAY   [34]
            0C A1   8E   7D    00FA          MOVQ    (SP)+,12(R1)            ; PUT MODIFIED R0/R1 BACK IN MECHANISM
               0FFC 8F   BA    00FE          POPR    #^XFFC                  ; RESTORE R2-R11
                                0102
     00000004'EF    8E   7D    0102          MOVQ    (SP)+,L^NEW_AP_FP       ; AP - FP                           [14]
     00000000'EF    8E   D0    0109          MOVL    (SP)+,L^NEW_SP          ; PUT SP, PC, PSL                   [14]
     0000000C'EF    8E   7D    0110          MOVQ    (SP)+,L^NEW_PC_PSL      ; IN A SAFE PLACE                   [14]
               51    60   D0    0117          MOVL    (R0),R1                 ; SIGNAL ARRAY SIZE
                  6041   DC    011A          MOVPSL  (R0)[R1]                ; REPLACE ORIGINAL PSL WITH CURRENT
      FC A041   32'AF   DE    011D          MOVAL   B^EXCPT,-4(R0)[R1]      ; LOCAL PC REPLACES ORIGINAL
               50    01   D0    0123          MOVL    #SS$_CONTINUE,R0        ; INDICATE SUCCESS
                                0126
                                0126   COND_HANDLR_EXIT:
                                0126          $MP_CLEAR_INTERLOCK  #MP$V_IL_CNHN ; Clear Interlock                [32]
                     06   DD    0126                  PUSHL   #MP$V_IL_CNHN
                     09   DD    0128                  PUSHL   #$SRVCODE_CLEAR_INTERLOCK
     00000000'9F    02   FB    012A                  CALLS   #2, @#DS$SERVICE_DISPATCHER
                                0131
                     04         0131          RET                             ; RETURN TO DISPATCHER
                                0132
     5C   00000004'EF    7D    0132   EXCPT: MOVQ    L^NEW_AP_FP,AP          ; SET AP AND FP                     [14]
     5E   00000000'EF    D0    0139          MOVL    L^NEW_SP,SP             ; SWITCH "SP'S"                     [14]
```

```
                7E   0000000C'EF    7D  0140   501          MOVQ    L^NEW_PC_PSL,-(SP)      ; SETUP FOR REI  PC/PSL PAIR          [14]
                                     02  0147   502          REI                             ; RETURN TO PROGRAM CAUSING EXCEPTION
```

```
                     0148    504          .SBTTL  DSX$PrintSig Routine - Format Signal Array
                     0148    505 ;++
                     0148    506 ; FUNCTIONAL DESCRIPTION:
                     0148    507 ;
                     0148    508 ;       This routine can be called to format the signal array
                     0148    509 ;       created by the condition/exception handler.
                     0148    510 ;
                     0148    511 ; CALLING SEQUENCE:
                     0148    512 ;
                     0148    513 ;       DSX$PRINTSIG(signal_array)      Print using DS$PRINTB/DS$PRINTX
                     0148    514 ;       DSX$PRINTSIGI(signal_array)     Print using DS$PRINTI
                     0148    515 ;
                     0148    516 ; INPUT PARAMETERS:
                     0148    517 ;
                     0148    518 ;       AP -> Signal array, Specific contents dependent on the condition
                     0148    519 ;
                     0148    520 ; IMPLICIT INPUTS:
                     0148    521 ;
                     0148    522 ;       NONE
                     0148    523 ;
                     0148    524 ; OUTPUT PARAMETERS:
                     0148    525 ;
                     0148    526 ;       NONE
                     0148    527 ;
                     0148    528 ; IMPLICIT OUTPUTS:
                     0148    529 ;
                     0148    530 ;       NONE
                     0148    531 ;
                     0148    532 ; SIDE EFFECTS:
                     0148    533 ;
                     0148    534 ;       An error message is typed
                     0148    535 ;
                     0148    536 ; REGISTER USAGE:
                     0148    537 ;
                     0148    538 ;       R5      Print routine for message header
                     0148    539 ;       R6      Print routine for information text
                     0148    540 ;
                     0148    541 ; COMPLETION CODES:
                     0148    542 ;
                     0148    543 ;       SS$_NORMAL      If condition recoginzed
                     0148    544 ;       SS$_RESIGNAL    If condition unknown
                     0148    545 ;--
                     0148    546
                     0148    547          $OFFSET 0,NEGATIVE, < -
                     0148    548                  <BUFFER,BUFSIZ> -      ; Length of buffer for conversion
                     0148    549                  <LOCAL,0>>             ; Length of local storage
                     0148                 .SAVE   LOCAL_BLOCK
                     0148                 .PSECT $ABS$ ABS
00000000  FE70                            .=0
FFFFFF00  0000                            .BLKB   -BUFSIZ
          FF00           BUFFER:
          FF00           LOCAL:
00000148                  .RESTORE
```

```
                     006C  0148    551 .ENTRY  DSX$PRINTSIG,^M<R2,R3,R5,R6>    ; Save registers
55   00000000'9F       9E  014A    552         MOVAB   a#DSX$PRINTB,R5         ; Use PRINTB for first message
56   00000000'9F       9E  0151    553         MOVAB   a#DSX$PRINTX,R6         ; Use PRINTX for information message
                       0C  11  0158    554         BRB     COMMON                  ; Join common code
                           015A    555
                     006C  015A    556 .ENTRY  DSX$PRINTSIGI,^M<R2,R3,R5,R6>; Save registers
55   00000000'9F       9E  015C    557         MOVAB   a#DSX$PRINTI,R5         ; Use PRINTI for first message
          56   65       9E  0163    558         MOVAB   (R5),R6                 ; Use PRINTI for information message
                           0166    559
                           0166    560 COMMON:
00000000'EF         0B    90  0166    561         MovB    #DS$K_Type_Exception_Head, - ; Set typecode                 [18]
                           016D    562           L^DS$GB_TypeCode      ;                                             [18]
                           016D    563
   5E    FF00 CD    9E  016D    564         MOVAB   LOCAL(FP),SP            ; Allocate space for local storage
 50    04 AC    08    C7  0172    565         DIVL3   #8,4(AP),R0             ; Get index from condition code
                           0177    566
     00000433'EF       9F  0177    567         PUSHAB  L^PRINT_PC_PSL          ; Fake JSB to handler:                  [18]
                           017D    568                                         ; ... set return address to print PC/PSL[14]
                           017D    569         CASE    R0,LIMIT=#SS$_BREAKa-3,TYPE=L,DISPLIST=<-
                           017D    570           X_BREAK, -                    ; SS$_BREAK
                           017D    571           X_CHMS, -                     ; SS$_CHMS
                           017D    572           X_CHMU, -                     ; SS$_CHMU
                           017D    573           X_COMPAT, -                   ; SS$_COMPAT
                           017D    574           X_OPCCUS, -                   ; SS$_OPCCUS
                           017D    575           X_OPCDEC, -                   ; SS$_OPCDEC
                           017D    576           RESIGNAL, -                   ; SS$_PAGRDERR, not handled
                           017D    577           X_RADRMOD, -                  ; SS$_RADRMOD
                           017D    578           X_ROPRAND, -                  ; SS$_ROPRAND
                           017D    579           RESIGNAL, -                   ; SS$_FAIL, not handled
                           017D    580           X_TBIT, -                     ; SS$_TBIT
                           017D    581           RESIGNAL, -                   ; SS$_DEBUG, not handled
                           017D    582           RESIGNAL, -                   ; SS$_ARTRES, not handled
                           017D    583           X_INTOVF, -                   ; SS$_INTOVF
                           017D    584           X_INTDIV, -                   ; SS$_INTDIV
                           017D    585           X_FLTOVF, -                   ; SS$_FLTOVF
                           017D    586           X_FLTDIV, -                   ; SS$_FLTDIV
                           017D    587           X_FLTUND, -                   ; SS$_FLTUND
                           017D    588           X_DECOVF, -                   ; SS$_DECOVF
                           017D    589           X_SUBRNG, -                   ; SS$_SUBRNG
                           017D    590           X_FLTOVF_F, -                 ; SS$_?????? Floating overflow fault
                           017D    591           X_FLTDIV_F, -                 ; SS$_?????? Floating divide fault
                           017D    592           X_FLTUND_F -                  ; SS$_?????? Floating underflow fault
                           017D    593           >
16'  00000082 8F    50    CF  017D                CASEL   R0,#SS$_BREAKa-3,S^#<<30003$-30002$>/2>-1
                           0185            30002$:
                     01C8' 0185                    .SIGNED_WORD    X_BREAK-30002$
                     01D0' 0187                    .SIGNED_WORD    X_CHMS-30002$
                     01D8' 0189                    .SIGNED_WORD    X_CHMU-30002$
                     01E0' 018B                    .SIGNED_WORD    X_COMPAT-30002$
                     020A' 018D                    .SIGNED_WORD    X_OPCCUS-30002$
                     0212' 018F                    .SIGNED_WORD    X_OPCDEC-30002$
                     030F' 0191                    .SIGNED_WORD    RESIGNAL-30002$
                     021A' 0193                    .SIGNED_WORD    X_RADRMOD-30002$
                     0222' 0195                    .SIGNED_WORD    X_ROPRAND-30002$
                     030F' 0197                    .SIGNED_WORD    RESIGNAL-30002$
                     022A' 0199                    .SIGNED_WORD    X_TBIT-30002$
                     030F' 019B                    .SIGNED_WORD    RESIGNAL-30002$
```

```
                    030F' 019D                .SIGNED_WORD    RESIGNAL-30002$
                    0232' 019F                .SIGNED_WORD    X_INTOVF-30002$
                    023A' 01A1                .SIGNED_WORD    X_INTDIV-30002$
                    0242' 01A3                .SIGNED_WORD    X_FLTOVF-30002$
                    024A' 01A5                .SIGNED_WORD    X_FLTDIV-30002$
                    0252' 01A7                .SIGNED_WORD    X_FLTUND-30002$
                    025A' 01A9                .SIGNED_WORD    X_DECOVF-30002$
                    0262' 01AB                .SIGNED_WORD    X_SUBRNG-30002$
                    026A' 01AD                .SIGNED_WORD    X_FLTOVF_F-30002$
                    0272' 01AF                .SIGNED_WORD    X_FLTDIV_F-30002$
                    027A' 01B1                .SIGNED_WORD    X_FLTUND_F-30002$
                          01B3        30003$:
                          01B3   594
          04 AC    0C   B1  01B3   595         CMPW    #SS$_ACCVIO,4(AP)       ; Access violation?
                    0A   12  01B7   596         BNEQ    200$                    ; Branch if not
     65   000000B7'EF   FA  01B9   597         CALLG   L^MSG_ACCVIO,(R5)       ; Print header                          [14]
                  0244   31  01C0   598         BRW     PRINT_TYPE_VA           ; Format access type and address
                          01C3   599
  04 AC   000002BC 8F   D1  01C3   600 200$:   CmpL    #SS$_MCHECK, 4(Ap)      ; User mode machine check?              [17]
                    08   12  01CB   601         BNeq    205$                    ; Branch if not                         [17]
     65   00000037'EF   FA  01CD   602         CallG   L^Msg_UMChk, (R5)       ; Print header                          [17]
                        05  01D4   603         Rsb                             ; Print PC/PSL                          [17]
                          01D5   604
  04 AC   00660088 8F   D1  01D5   605 205$:   CMPL    #DS$_MCHK,4(AP)         ; Standalone Machine check?             [17]
                    0E   12  01DD   606         BNEQ    210$                    ; Branch if not
              0060 8F   BB  01DF   607         PUSHR   #^M<R5,R6>              ; Use Separate print routines for hdr/txt
                    5C   DD  01E3   608         PUSHL   AP                      ; Address of signal array
     00000000'EF   03   FB  01E5   609         CALLS   #3,CHR$MCHK             ; Format Machine check information
                        05  01EC   610         RSB                             ; All done
                          01ED   611
  04 AC   00660090 8F   D1  01ED   612 210$:   CMPL    #DS$_KRNLSTK,4(AP)      ; Kernel stack not valid?
                    08   12  01F5   613         BNEQ    220$                    ; Branch if not
     65   0000003F'EF   FA  01F7   614         CALLG   L^MSG_KRNLSTK,(R5)      ; Print header                          [14]
                        05  01FE   615         RSB                             ; All done
                          01FF   616
  04 AC   006600A0 8F   D1  01FF   617 220$:   CMPL    #DS$_TRANSL,4(AP)       ; Translation not valid?
                    0A   12  0207   618         BNEQ    240$                    ; Branch if not                         [23]
     65   000000CB'EF   FA  0209   619         CALLG   L^MSG_TRANSL,(R5)       ; Print header                          [14]
                  01F4   31  0210   620         BRW     PRINT_TYPE_VA           ; Format access type and address
                          0213   621
  04 AC   006600A8 8F   D1  0213   622 240$:   CMPL    #DS$_CHME,4(AP)         ; Change mode to EXEC?
                    09   12  021B   623         BNEQ    250$                    ; Branch if not
     65   0000012F'EF   FA  021D   624         CALLG   L^MSG_CHME,(R5)         ; Print header                          [14]
                    11   11  0224   625         BRB     255$                    ; Branch to print argument
                          0226   626
  04 AC   006600E0 8F   D1  0226   627 250$:   CMPL    #DS$_CHMK,4(AP)         ; Change mode to KERNEL?
                    0A   12  022E   628         BNEQ    260$                    ; Branch if not
     65   0000011B'EF   FA  0230   629         CALLG   L^MSG_CHMK,(R5)         ; Print header                          [14]
                  01E4   31  0237   630 255$:   BRW     PRINT_CHMX_ARG          ; Print argument
                          023A   631
  04 AC   006600D8 8F   D1  023A   632 260$:   CMPL    #DS$_UNEXPINT,4(AP)     ; Unexpected trap or interrupt?
                    03   13  0242   633         BEQL    263$                    ; Yes,
                  009D   31  0244   634         BRW     270$                    ; Branch if not
                          0247   635 263$:   BR_IF_NOT_USER  264$             ;                                       [31]
     0000FE00'EF   1C   E1  0247               BBC     #DSA$V_User, -                  ; If bit not set, branch
                    03      024E                        L^DSA$GL_Flags, 264$          ;
                  0085   31  024F   636         BRW     265$                    ; User mode, branch                     [31]
```

```
                05   0000FE17'EF   91   0252    637 264$:   CMPB    DSA$GL_SID+3,#PR$_SID_TYP8SS ; Print node # and BER if ZZZ      [29]
                                   7C   12   0259    638         BNEQ    265$                         ; Branch if not ZZZ             [29]
                 00000100 8F    08 AC   D1   025B    639         CMPL    8(AP),#^X100                 ; Only want adapter vectors     [31]
                                   72   19   0263    640         BLSS    265$                         ; Branch if vector < 100        [31]
                 000001FC 8F    08 AC   D1   0265    641         CMPL    8(AP),#^X1FC                 ;                               [31]
                                   68   14   026D    642         BGTR    265$                         ; Branch if vector > 1FC        [31]
              52   08 AC    04   02   EF   026F    643         EXTZV   #2,#4,8(AP),R2               ; Extract node # from vector    [29]
                                   52   DD   0275    644         PUSHL   R2                           ; Node #                        [29]
                         08 AC   DD   0277    645         PUSHL   8(AP)                        ; Vector offset                 [29]
                    0000041F'EF   9F   027A    646         PUSHAB  L^FMT_BI_UNEXPINT            ; Edit string                   [29]
                          65   03   FB   0280    647         CALLS   #3,(R5)                      ; Print it                      [29]
                00000000'EF   0B   90   0283    648         MOVB    #DS$K_TYPE_EXCEPTION_HEAD, - ; Set typecode for header      [29]
                                   028A    649                 L^DS$GB_TYPECODE             ;                               [29]
                    52   52   0D   78   028A    650         ASHL    #13,R2,R2                    ; Build BI address              [29]
                52   60000000 8F   C8   028E    651         BISL2   #^X60000000,R2               ; Make it virtual and in IO space [29]
                     53   08 A2   D0   0295    652         MOVL    BIIC$L_BER(R2),R3            ; Fetch Bus Error Register      [29]
                     08 A2   53   D0   0299    653         MOVL    R3,BIIC$L_BER(R2)            ; Clear Bus Error Register      [31]
                                   029D    654         $DS_CVTREG_S    #31,R3, -                ; MSB and value to be converted [29]
                                   029D    655                 BI_BER_BIT,BUFFER(FP), - ; control string and buffer address [29]
                                   029D    656                 #BUFSIZ                      ; buffer size                   [29]
                                   00   DD   029D                  PUSHL   #0
                                   00   DD   029F                  PUSHL   #0
                                   00   DD   02A1                  PUSHL   #0
                                   00   DD   02A3                  PUSHL   #0
                                   00   DD   02A5                  PUSHL   #0
                                   00   DD   02A7                  PUSHL   #0
                         00000100 8F   DD   02A9                  PUSHL   #BUFSIZ
                             FF00 CD   9F   02AF                  PUSHAB  BUFFER(FP)
                         00000000'EF   9F   02B3                  PUSHAB  BI_BER_BIT
                                   53   DD   02B9                  PUSHL   R3
                                   1F   DD   02BB                  PUSHL   #31
              00000000'9F   0B   FB   02BD                  CALLS   #11, a#DS$CVTREG
                   50   FF00 CD   9E   02C4    657         MOVAB   BUFFER(FP),R0               ; Get address of buffer         [29]
                                   50   DD   02C9    658         PUSHL   R0                           ; Address of mnemonic string    [29]
                                   53   DD   02CB    659         PUSHL   R3                           ; Contents of BER               [29]
                         00000000'EF   9F   02CD    660         PUSHAB  FMT_BER                      ; Address of format string      [29]
                             66   03   FB   02D3    661         CALLS   #3,(R6)                      ; Output message of lesser importance [29]
                                   05   02D6    662         RSB                                  ; All done                      [29]
                                   02D7    663
                            08 AC   DD   02D7    664 265$:   PUSHL   8(AP)                        ; Vector offset                 [29]
                         000003E7'EF   9F   02DA    665         PUSHAB  L^FMT_UNEXPINT               ; Edit string                   [14]
                             65   02   FB   02E0    666         CALLS   #2,(R5)                      ; Print it
                                   05   02E3    667         RSB                                  ; All done
                                   02E4    668
           04 AC    006600D0 8F   D1   02E4    669 270$:   CMPL    #DS$_ARITH,4(AP)             ; Arithmetic fault or trap?
                                   1F   12   02EC    670         BNEQ    280$                         ; Branch if this not it
                             0433'CF   9F   02EE    671         PUSHAB  W^PRINT_PC_PSL               ; Set return address to print PC/PSL
                                   02F2    672         CASE    8(AP),TYPE=W,LIMIT=#1,DISPLIST=<-        ; Case on Trap/Fault type
                                   02F2    673                   X_INTOVF, -                ; SS$_INTOVF
                                   02F2    674                   X_INTDIV, -                ; SS$_INTDIV
                                   02F2    675                   X_FLTOVF, -                ; SS$_FLTOVF
                                   02F2    676                   X_FLTDIV, -                ; SS$_FLTDIV
                                   02F2    677                   X_FLTUND, -                ; SS$_FLTUND
                                   02F2    678                   X_DECOVF, -                ; SS$_DECOVF
                                   02F2    679                   X_SUBRNG, -                ; SS$_SUBRNG
                                   02F2    680                   X_FLTOVF_F, -              ; SS$_?????? Floating overflow fault
                                   02F2    681                   X_FLTDIV_F, -              ; SS$_?????? Floating divide fault
```

```
                        02F2   682                  X_FLTUND_F -          ; SS$_?????? Floating underflow fault
                        02F2   683                  >
09'  01    08 AC    AF  02F2                CASEW   8(AP),#1,S^#<<30005$-30004$>/2>-1
                        02F7            30004$:
                00C0'   02F7                    .SIGNED_WORD    X_INTOVF-30004$
                00C8'   02F9                    .SIGNED_WORD    X_INTDIV-30004$
                00D0'   02FB                    .SIGNED_WORD    X_FLTOVF-30004$
                00D8'   02FD                    .SIGNED_WORD    X_FLTDIV-30004$
                00E0'   02FF                    .SIGNED_WORD    X_FLTUND-30004$
                00E8'   0301                    .SIGNED_WORD    X_DECOVF-30004$
                00F0'   0303                    .SIGNED_WORD    X_SUBRNG-30004$
                00F8'   0305                    .SIGNED_WORD    X_FLTOVF_F-30004$
                0100'   0307                    .SIGNED_WORD    X_FLTDIV_F-30004$
                0108'   0309                    .SIGNED_WORD    X_FLTUND_F-30004$
                        030B            30005$:
              00   11  030B   684               BRB     PRINT_UNK_EXC         ; Bad value of trap code, fake it.
                        030D   685
                        030D   686 ;+
                        030D   687 ; Unknown exception code, Print the arglist and PC/PSL
                        030D   688 ;-
                        030D   689
                        030D   690 280$:
                        030D   691 PRINT_UNK_EXC:
00000000'EF   0B   90   030D   692               MovB    #DS$K_Type_Exception_Head, - ; Set typecode for header      [18]
                        0314   693                 L^DS$GB_TypeCode      ;           [18]
                        0314   694
    00000536'EF   9F   0314   695               PUSHAB  L^FMT_UNK_ARG         ; Edit string for count               [14]
        65   02   FB   031A   696               CALLS   #2,(R5)               ; Print header
                        031D   697
00000000'EF   0C   90   031D   698               MovB    #DS$K_Type_Exception, - ; Set typecode for text             [18]
                        0324   699                 L^DS$GB_TypeCode      ;           [18]
              6C   DD   0324   700               PUSHL   (AP)                  ; Count of args
    00000558'EF   9F   0326   701               PUSHAB  L^FMT_UNK_CNT         ; format for count of args            [14]
        66   02   FB   032C   702               CALLS   #2,(R6)               ; Print count of args
                        032F   703
53    6C   02   C3   032F   704               SUBL3   #2,(AP),R3            ; Count less PC/PSL
              15   15   0333   705               BLEQ    20$                   ; None, Print PC/PSL
              52   01   D0   0335   706               MOVL    #1,R2                 ; Start with first arg
                        0338   707
            6C42   DD   0338   708 10$:     PUSHL   (AP)[R2]              ; Value of arg
              52   DD   033B   709               PUSHL   R2                    ; arg number
    0000056D'EF   9F   033D   710               PUSHAB  L^FMT_PXXX            ; Address of edit string              [14]
        66   03   FB   0343   711               CALLS   #3,(R6)               ; Print arg
 EE   52   53   F2   0346   712               AOBLSS  R3,R2,10$             ; Print all args
                        034A   713
            00E6   31   034A   714 20$:     BRW     PRINT_PC_PSL          ; Now format and print PC/PSL
```

```
                     034D   716 ;+
                     034D   717 ; SS$_BREAK             Breakpoint exception
                     034D   718 ;-
                     034D   719
                     034D   720 X_BREAK:
      65 000000F3'EF FA   034D   721         CALLG   L^MSG_BREAK,(R5)        ; Print header                          [14]
                  05   0354   722         RSB                             ; All done
                     0355   723
                     0355   724 ;+
                     0355   725 ; SS$_CHMS              Change mode to SUPERVISOR
                     0355   726 ;-
                     0355   727
                     0355   728 X_CHMS:
      65 00000143'EF FA   0355   729         CALLG   L^MSG_CHMS,(R5)         ; Print header                          [14]
                  05   035C   730         RSB                             ; All done
                     035D   731
                     035D   732 ;+
                     035D   733 ; SS$_CHMU              Change mode to USER
                     035D   734 ;-
                     035D   735
                     035D   736 X_CHMU:
      65 00000157'EF FA   035D   737         CALLG   L^MSG_CHMU,(R5)         ; Print header                          [14]
                  05   0364   738         RSB                             ; All done
                     0365   739
                     0365   740 ;+
                     0365   741 ; SS$_COMPAT            Compatability mode trap
                     0365   742 ;-
                     0365   743
                     0365   744 X_COMPAT:
   65 00000107'EF FA   0365   745         CALLG   L^MSG_COMPAT,(R5)       ; Print Compatability mode fault header [14]
50  08 AC    03   00 EF   036C   746         EXTZV   #0,#3,8(AP),R0          ; Get fault type
      000002F0'EF40 DD   0372   747         PUSHL   L^AAT_COMPAT[R0]        ; Address of insert                     [14]
         000006E0'EF 9F   0379   748         PUSHAB  L^T_FLT_ABO             ; Address of FAULT/ABORT                [14]
         000006EC'EF 9F   037F   749         PUSHAB  L^T_COMPAT              ; Address of Compatability mode ...     [14]
         0000050C'EF 9F   0385   750         PUSHAB  L^FMT_EXCEP_CODE        ; Format for exception code             [14]
             66  04 FB   038B   751         CALLS   #4,(R6)                 ; Print type code
                  05   038E   752         RSB                             ; All done
                     038F   753
                     038F   754 ;+
                     038F   755 ; SS$_OPCCUS              Opcode reserved to customer
                     038F   756 ;-
                     038F   757
                     038F   758 X_OPCCUS:
      65 0000007B'EF FA   038F   759         CALLG   L^MSG_OPCCUS,(R5)       ; Print header                          [14]
                  05   0396   760         RSB                             ; All done
                     0397   761
                     0397   762 ;+
                     0397   763 ; SS$_OPCDEC              Opcode reserved to DIGITAL
                     0397   764 ;-
                     0397   765
                     0397   766 X_OPCDEC:
      65 00000067'EF FA   0397   767         CALLG   L^MSG_OPCDEC,(R5)       ; Print header                          [14]
                  05   039E   768         RSB                             ; All done
                     039F   769
                     039F   770 ;+
                     039F   771 ; SS$_RADRMOD             Reserved addressing mode
                     039F   772 ;-
```

```
                     039F   773
                     039F   774 X_RADRMOD:
65   000000A3'EF  FA 039F   775          CALLG   L^MSG_RADRMOD,(R5)      ; Print header                [14]
                  05 03A6   776          RSB                             ; All done
                     03A7   777
                     03A7   778 ;+
                     03A7   779 ; SS$_ROPRAND           Reserved operand
                     03A7   780 ;-
                     03A7   781
                     03A7   782 X_ROPRAND:
65   0000008F'EF  FA 03A7   783          CALLG   L^MSG_ROPRAND,(R5)      ; Print header                [14]
                  05 03AE   784          RSB                             ; All done
                     03AF   785
                     03AF   786 ;+
                     03AF   787 ; SS$_TBIT              T-bit exception
                     03AF   788 ;-
                     03AF   789
                     03AF   790 X_TBIT:
65   000000DF'EF  FA 03AF   791          CALLG   L^MSG_TBIT,(R5)         ; Print header                [14]
                  05 03B6   792          RSB                             ; All done
                     03B7   793
                     03B7   794 ;+
                     03B7   795 ; SS$_INTOVF            Integer overflow
                     03B7   796 ;-
                     03B7   797
                     03B7   798 X_INTOVF:
65   00000228'EF  FA 03B7   799          CALLG   L^MSG_INTOVF,(R5)       ; Print header                [14]
                  05 03BE   800          RSB                             ; All done
                     03BF   801 ;+
                     03BF   802 ; SS$_INTDIV            Integer division by zero
                     03BF   803 ;-
                     03BF   804
                     03BF   805 X_INTDIV:
65   0000023C'EF  FA 03BF   806          CALLG   L^MSG_INTDIV,(R5)       ; Print header                [14]
                  05 03C6   807          RSB                             ; All done
                     03C7   808
                     03C7   809 ;+
                     03C7   810 ; SS$_FLTOVF            Floating overflow
                     03C7   811 ;-
                     03C7   812
                     03C7   813 X_FLTOVF:
65   00000250'EF  FA 03C7   814          CALLG   L^MSG_FLTOVF,(R5)       ; Print header                [14]
                  05 03CE   815          RSB                             ; All done
                     03CF   816
                     03CF   817 ;+
                     03CF   818 ; SS$_FLTDIV            Floating division by zero
                     03CF   819 ;-
                     03CF   820
                     03CF   821 X_FLTDIV:
65   00000264'EF  FA 03CF   822 170$:    CALLG   L^MSG_FLTDIV,(R5)       ; Print header                [14]
                  05 03D6   823          RSB                             ; All done
                     03D7   824
                     03D7   825 ;+
                     03D7   826 ; SS$_FLTUND            Floating underflow trap
                     03D7   827 ;-
                     03D7   828
                     03D7   829 X_FLTUND:
```

```
                65   00000278'EF   FA  03D7   830          CALLG   L^MSG_FLTUND,(R5)       ; Print header                  [14]
                                   05  03DE   831          RSB                             ; All done
                                       03DF   832
                                       03DF   833 ;+
                                       03DF   834 ; SS$_DECOVF             Decimal overflow trap
                                       03DF   835 ;-
                                       03DF   836
                                       03DF   837 X_DECOVF:
                65   0000028C'EF   FA  03DF   838          CALLG   L^MSG_DECOVF,(R5)       ; Print header                  [14]
                                   05  03E6   839          RSB                             ; All done
                                       03E7   840
                                       03E7   841 ;+
                                       03E7   842 ; SS$_SUBRNG             Subscript range trap
                                       03E7   843 ;-
                                       03E7   844
                                       03E7   845 X_SUBRNG:
                65   000002A0'EF   FA  03E7   846          CALLG   L^MSG_SUBRNG,(R5)       ; Print header                  [14]
                                   05  03EE   847          RSB                             ; All done
                                       03EF   848
                                       03EF   849 ;+
                                       03EF   850 ; SS$_??????             Floating overflow fault
                                       03EF   851 ;-
                                       03EF   852
                                       03EF   853 X_FLTOVF_F:
                65   000002B4'EF   FA  03EF   854          CALLG   L^MSG_FLTOVF_F,(R5)     ; Print header                  [14]
                                   05  03F6   855          RSB                             ; All done
                                       03F7   856
                                       03F7   857 ;+
                                       03F7   858 ; SS$_??????             Floating division by zero fault
                                       03F7   859 ;-
                                       03F7   860
                                       03F7   861 X_FLTDIV_F:
                65   000002C8'EF   FA  03F7   862          CALLG   L^MSG_FLTDIV_F,(R5)     ; Print header                  [14]
                                       03FE   863
                                   05  03FE   864          RSB                             ; All done
                                       03FF   865
                                       03FF   866 ;+
                                       03FF   867 ; SS$_??????             Floating underflow fault
                                       03FF   868 ;-
                                       03FF   869
                                       03FF   870 X_FLTUND_F:
                65   000002DC'EF   FA  03FF   871          CALLG   L^MSG_FLTUND_F,(R5)     ; Print header                  [14]
                                   05  0406   872          RSB                             ; All done
                                       0407   873
                                       0407   874 Print_Type_VA:                           ;                               [18]
00000000'EF       0C   90  0407   875          MovB    #DS$K_Type_Exception, -  ; Set typecode                          [18]
                                       040E   876            L^DS$GB_TypeCode       ;                               [18]
                                       040E   877
                    08 AC   DD  040E   878          PushL   8(AP)                   ; Reference type                [18]
                    0C AC   DD  0411   879          PUSHL   12(AP)                  ; Virtual address
            000004DA'EF     9F  0414   880          PUSHAB  L^FMT_VIRT              ; Format string                 [14]
                    66   03   FB  041A   881          CALLS   #3,(R6)                 ; Print message of lesser importance
                                   041D   882          RSB                             ; Now format and print PC/PSL
                                       041E   883
                                       041E   884 PRINT_CHMX_ARG:
00000000'EF       0C   90  041E   885          MovB    #DS$K_Type_Exception, -  ; Set typecode                          [18]
                                       0425   886            L^DS$GB_TypeCode       ;                               [18]
```

```
                                0425
                08 AC    DD     0425          PUSHL   8(AP)                      ; Output CHMx ARG
        00000523'EF      9F     0428          PUSHAB  L^FMT_CHMX_ARG             ; Format CHMx ARG                    [14]
            66   02      FB     042E          CALLS   #2,(R6)                    ; Output message of lesser importance
                 00      11     0431          BRB     PRINT_PC_PSL               ; Now format and print PC/PSL
                                0433
                                0433  PRINT_PC_PSL:
  00000000'EF    0C      90     0433          MovB    #DS$K_Type_Exception, -    ; Set typecode                       [18]
                                043A                    L^DS$GB_TypeCode         ;                                    [18]
                                043A
            52   6C      D0     043A          MOVL    (AP),R2                    ; Get count of args
            FC AC42      DD     043D          PUSHL   -4(AP)[R2]                 ; PC to print
        00000465'EF      9F     0441          PUSHAB  L^FMT_ERRPC                ; Address of format string           [14]
            66   02      FB     0447          CALLS   #2,(R6)                    ; Output message of lesser importance
                                044A          $DS_CVTREG_S    #31,(AP)[R2], -    ; MSB and value to be converted
                                044A                  APSLMNE, BUFFER(FP), -     ; Control string and buffer address
                                044A                  #BUFSIZ, MODELST,MODELST; Buffer size and mode list twice
                 00      DD     044A                  PUSHL   #0
                 00      DD     044C                  PUSHL   #0
                 00      DD     044E                  PUSHL   #0
                 00      DD     0450                  PUSHL   #0
        00000000'EF      DF     0452                  PUSHAL  MODELST
        00000000'EF      DF     0458                  PUSHAL  MODELST
        00000100 8F      DD     045E                  PUSHL   #BUFSIZ
            FF00 CD      9F     0464                  PUSHAB  BUFFER(FP)
        00000000'EF      9F     0468                  PUSHAB  APSLMNE
              6C42       DD     046E                  PUSHL   (AP)[R2]
                 1F      DD     0471                  PUSHL   #31
  00000000'9F    0B      FB     0473                  CALLS   #11, a#DS$CVTREG
      50   FF00 CD      9E     047A          MOVAB   BUFFER(FP),R0              ; Get address of buffer
                 50      DD     047F          PUSHL   R0                         ; Address of mneumonic string
              6C42       DD     0481          PUSHL   (AP)[R2]                   ; Push PSL
        000004B9'EF      9F     0484          PUSHAB  L^FMT_ERRPSL               ; Address of format string           [14]
            66   03      FB     048A          CALLS   #3,(R6)                    ; output message of lesser importance
        00000000'EF      94     048D          ClrB    L^DS$GB_TypeCode           ; Clear typecode                     [30]
                         04     0493          RET                                ; Return all done
                                0494
                                0494  RESIGNAL:
      50   0918 8F      3C     0494          MOVZWL  #SS$_RESIGNAL,R0           ; Unknown signal
        00000000'EF      94     0499          ClrB    L^DS$GB_TypeCode           ; Clear typecode                     [30]
                         04     049F          RET                                ; Return
```

```
04A0    917          .SBTTL  FIND_USERPC     Find USER PC on stack
04A0    918 ;++
04A0    919 ; FUNCTIONAL DESCRIPTION:
04A0    920 ;
04A0    921 ;       This routine is used to find and print a user program PC
04A0    922 ;       in the event of an error, exception, abort or fault.
04A0    923 ;
04A0    924 ; CALLING SEQUENCE:
04A0    925 ;
04A0    926 ;       CALLG   (AP),FIND_USERPC
04A0    927 ;
04A0    928 ; INPUT PARAMETERS:
04A0    929 ;
04A0    930 ;       R0 = ADDRESS OF FAILURE
04A0    931 ;
04A0    932 ; IMPLICIT INPUTS:
04A0    933 ;
04A0    934 ;       NONE
04A0    935 ;
04A0    936 ; OUTPUT PARAMETERS:
04A0    937 ;
04A0    938 ;       NONE
04A0    939 ;
04A0    940 ; IMPLICIT OUTPUTS:
04A0    941 ;
04A0    942 ;       NONE
04A0    943 ;
04A0    944 ; COMPLETION CODES:
04A0    945 ;
04A0    946 ;       NONE
04A0    947 ;
04A0    948 ; SIDE EFFECTS:
04A0    949 ;
04A0    950 ;       NONE
04A0    951 ;--
```

```
                    0000  04A0   953 .ENTRY FIND_USERPC,^M<>
                          04A2   954
          50   04 BC     DE  04A2   955         MOVAL   a4(AP),R0               ; GET ADDRESS OF SIGNAL ARRAY
             51   60     9A  04A6   956         MOVZBL  (R0),R1                 ; GET COUNT OF SIGNAL ARGS
       50    FC A041     D0  04A9   957         MOVL    -4(R0)[R1],R0           ; Get PC of failure
00010000 8F    50     D1  04AE   958         cmpl    r0, #^X10000            ; Less than 10000?                  [15]
               0C     1F  04B5   959         blssu   10$                     ; If so, don't search               [15]
00000000'EF    00     FB  04B7   960         CallS   #0, L^DSR$Find_User_PC  ; Search for it                     [20]
          50   00     D1  04BE   961         CmpL    #0, R0                  ; Return a zero?                    [20]
               10     13  04C1   962         Beql    20$                     ; Yes, so no user PC was found      [20]
                          04C3   963
                          04C3   964 10$:     $PRINTI_S  L^FMT_USRPC, R0      ; Print the real User PC            [20]
               50     DD  04C3                       PUSHL   R0
     0000047E'EF      9F  04C5                       PUSHAB  L^FMT_USRPC
00000000'EF    02     FB  04CB                       CALLS   #$$N, DSX$PRINTI
                      04  04D2   965         RET                             ; Return to caller
                          04D3   966
                          04D3   967 20$:     $PrintI_S  L^Fmt_User_PC_NF     ; Print "none found"                [20]
     0000049A'EF      9F  04D3                       PUSHAB  L^Fmt_User_PC_NF
00000000'EF    01     FB  04D9                       CALLS   #$$N, DSX$PRINTI
                      04  04E0   968         RET                             ; Return to caller                  [20]
```

```
                        04E1   970          .SBTTL  TST$MChk Routine - Machine Check Handler        ;              [14]
                   000004E1    971          .PSECT  Code, Exe, Shr, NoWrt, Long                     ;              [14]
                        04E1   972 ;++
                        04E1   973 ; FUNCTIONAL DESCRIPTION:
                        04E1   974 ;
                        04E1   975 ;       This routine is entered from SCB offset 4.
                        04E1   976 ;       It removes the machine check info from the stack and adds 2
                        04E1   977 ;       to the PC, sets the "C" bit in the PSL and returns
                        04E1   978 ;
                        04E1   979 ; CALLING SEQUENCE:
                        04E1   980 ;
                        04E1   981 ;       Vectored from SCB offset 4
                        04E1   982 ;       FOR EXAMPLE:
                        04E1   983 ;               PUSHL   4(scb)          ; Save old machine check
                        04E1   984 ;               MOVAL   TST$MCHK,4(SCB) ; Set new machine check handler
                        04E1   985 ;               TSTW    (R0)            ; Instruction expected to get mchk
                        04E1   986 ;               POPL    4(SCB)          ; Restore old machine check handler
                        04E1   987 ;               BCS     ...             ; Branch if machine check occured
                        04E1   988 ;
                        04E1   989 ; INPUT PARAMETERS:
                        04E1   990 ;
                        04E1   991 ;       NONE
                        04E1   992 ;
                        04E1   993 ; IMPLICIT INPUTS:
                        04E1   994 ;
                        04E1   995 ;       (SP)    number of bytes of machine check info
                        04E1   996 ;       (SP)+SP PC and PSL of fault inst which MUST be 2 bytes long
                        04E1   997 ;
                        04E1   998 ; OUTPUT PARAMETERS:
                        04E1   999 ;
                        04E1  1000 ;       NONE
                        04E1  1001 ;
                        04E1  1002 ; IMPLICIT OUTPUTS:
                        04E1  1003 ;
                        04E1  1004 ;       C-bit is set and interrupt PC updated by 2
                        04E1  1005 ;
                        04E1  1006 ; SIDE EFFECTS:
                        04E1  1007 ;
                        04E1  1008 ;       NONE
                        04E1  1009 ;
                        04E1  1010 ; COMPLETION CODES:
                        04E1  1011 ;
                        04E1  1012 ;       N/A
                        04E1  1013 ;--
```

```
                                04E1  1015          .ALIGN  LONG
                                04E4  1016 TST$MCHK::
                          5E    DD  04E4  1017          PUSHL   SP                      ; Store stack frame address
                          01    DD  04E6  1018          PUSHL   #DS$_GK_TST$MCHK                ; Pass fault clear selection code
                00000000'EF     16  04E8  1019          JSB     L^DSR$FAULT_CLEAR       ; Clear fault status
                          5E    08  C0  04EE  1020          ADDL2   #8,SP                   ; Reset the stack                       ;[24
                          5E    8E  C0  04F1  1021          ADDL    (SP)+, SP               ; Remove specific machine check info    [16]
                          6E    02  C0  04F4  1022          ADDL    #2,(SP)                 ; Update PC
                       04 AE    01  C8  04F7  1023          BISL    #1,4(SP)                ; Set C-bit in condition codes
                                02  04FB  1024          REI                             ; Return after faulting instruction
```

```
                     04FC  1026          .SBTTL  HARDWARE EXCEPTION ENTRY POINTS
                000004FC  1027          .PSECT  Code, Exe, Shr, NoWrt, Long                     ;                [14]
                     04FC  1028 ;++
                     04FC  1029 ; FUNCTIONAL DESCRIPTION:
                     04FC  1030 ;
                     04FC  1031 ;       THESE ROUTINES EMULATE THE STARLET EXCEPTION HANDLER MECHANISM.
                     04FC  1032 ;       SEE THE STARLET WORKING DESIGN DOCUMENT FOR FURTHER DETAILS.
                     04FC  1033 ;
                     04FC  1034 ; CALLING SEQUENCE:
                     04FC  1035 ;
                     04FC  1036 ;       EXCEPTION OR INTERRUPT.
                     04FC  1037 ;
                     04FC  1038 ; INPUT PARAMETERS:
                     04FC  1039 ;
                     04FC  1040 ;       EXCEPTION FRAME ON THE STACK.
                     04FC  1041 ;
                     04FC  1042 ; IMPLICIT INPUTS:
                     04FC  1043 ;
                     04FC  1044 ;       NONE
                     04FC  1045 ;
                     04FC  1046 ; OUTPUT PARAMETERS:
                     04FC  1047 ;
                     04FC  1048 ;       NONE
                     04FC  1049 ;
                     04FC  1050 ; IMPLICIT OUTPUTS:
                     04FC  1051 ;
                     04FC  1052 ;       NONE
                     04FC  1053 ;
                     04FC  1054 ; COMPLETION CODES:
                     04FC  1055 ;
                     04FC  1056 ;       NONE
                     04FC  1057 ;
                     04FC  1058 ; SIDE EFFECTS:
                     04FC  1059 ;
                     04FC  1060 ;       NONE
                     04FC  1061 ;
                     04FC  1062 ;--
```

```
                                04FC  1064 ;+
                                04FC  1065 ; ENTER HERE FOR MACHINE CHECK ABORTS, FAULTS AND TRAPS
                                04FC  1066 ;-
                                04FC  1067
                                04FC  1068          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                                04FC  1069 EXE_MCHK::                                   ; HERE FROM MCHK VECTOR
          00660088 8F    DD     04FC  1070          PUSHL   #DS$_MCHK                   ; INDICATE MACHINE CHECK CONDITION
      7E    04 AE    04    C7   0502  1071          DIVL3   #4,4(SP),-(SP)              ; CONVERT BYTES TO LONGWORD COUNT
             6E    04    C0   0507  1072          ADDL2   #4,(SP)                     ; INCLUDE PSL/PC/COND NAME & BYTE COUNT
                   5E    DD   050A  1073          PUSHL   SP                          ; Store stack frame address                 ;[24
                   00    DD   050C  1074          PUSHL   #DS$_GK_EXE_MCHK            ; PUSH THE FAULT CLEAR SELECTION CODE       ;[24
          00000000'EF    16   050E  1075          JSB     L^DSR$FAULT_CLEAR           ; CLEAR ERROR FIRST PASS AND RE-INIT
             5E    08    C0   0514  1076          ADDL2   #8,SP                       ; Reset the stack                           ;[24
                 014D    31   0517  1077          BRW     EXE_EXCEPTION               ; PROCEED
                              051A  1078
                              051A  1079 ;+
                              051A  1080 ; ENTER HERE FOR KERNEL STACK NOT VALID ABORT
                              051A  1081 ;-
                              051A  1082
                              051A  1083          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                              051C  1084 EXE_KRNLSTK::                                ; HERE FROM KRNLSTK VECTOR
          00660090 8F    DD     051C  1085          PUSHL   #DS$_KRNLSTK                ; INDICATE KERNEL STACK CONDITION
                 013E    31   0522  1086          BRW     EXE_3ARG                    ; SETUP 3 ARGS, AND PROCESS
                              0525  1087
                              0525  1088 ;+
                              0525  1089 ; ENTER HERE FOR POWER INTERRUPT
                              0525  1090 ;-
                              0525  1091
                              0525  1092          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                              0528  1093 EXE_POWER::                                  ; HERE FROM POWER FAIL VECTOR
          00660098 8F    DD     0528  1094          PUSHL   #DS$_POWER                  ; INDICATE POWER FAIL CONDITION
                   00    00   052E  1095          HALT                                ; NOT GRACEFUL, BUT EFFECTIVE
                 0131    31   052F  1096          BRW     EXE_3ARG                    ; SETUP 3 ARGS, AND PROCESS
                              0532  1097
                              0532  1098 ;+
                              0532  1099 ; ENTER HERE FOR RESERVED OR PRIVILEGED INSTRUCTION FAULT
                              0532  1100 ;-
                              0532  1101
                              0532  1102          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                              0534  1103 EXE_OPCDEC::                                 ; HERE FROM OPCDEC VECTOR
      7E    043C 8F    3C     0534  1104          MOVZWL  #SS$_OPCDEC, -(SP)          ; INDICATE RESERVED OPCODE CONDITION
                 0127    31   0539  1105          BRW     EXE_3ARG                    ; SETUP 3 ARGS
                              053C  1106
                              053C  1107 ;+
                              053C  1108 ; ENTER HERE FOR CUSTOMER RESERVED INSTRUCTION FAULT
                              053C  1109 ;-
                              053C  1110
                              053C  1111          .ALIGN  LONG                        ; REQUIRED FOR EXCEPTION VECTOR
                              053C  1112 EXE_OPCCUS::                                 ; HERE FROM OPCCUS VECTOR
      7E    0434 8F    3C     053C  1113          MOVZWL  #SS$_OPCCUS, -(SP)          ; INDICATE CUSTOMER OPCODE CONDITION
                 011F    31   0541  1114          BRW     EXE_3ARG                    ; SETUP 3 ARGS
```

```
                        0544  1116 ;+
                        0544  1117 ; ENTER HERE FOR RESERVED OPERAND FAULT OR ABORT
                        0544  1118 ;-
                        0544  1119
                        0544  1120          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                        0544  1121 EXE_ROPRAND::                            ; HERE FROM ROPRAND VECTOR
                        0544  1122 EXE$ROPRAND::                            ; Used by XDELTA only
      7E    0454 8F   3C  0544  1123          MOVZWL  #SS$_ROPRAND, -(SP)   ; INDICATE RESERVED OPERAND CONDITION
          0117   31     0549  1124          BRW     EXE_3ARG                ; SETUP 3 ARGS
                        054C  1125
                        054C  1126 ;+
                        054C  1127 ; ENTER HERE FOR RESERVED ADDRESSING MODE FAULT
                        054C  1128 ;-
                        054C  1129
                        054C  1130          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                        054C  1131 EXE_RADRMOD::                            ; HERE FROM RADRMOD VECTOR
      7E    044C 8F   3C  054C  1132          MOVZWL  #SS$_RADRMOD, -(SP)   ; INDICATE ADDRESSING MODE CONDITION
          010F   31     0551  1133          BRW     EXE_3ARG                ; SETUP 3 ARGS
                        0554  1134
                        0554  1135 ;+
                        0554  1136 ; ENTER HERE FOR ACCESS CONTROL VIOLATION FAULT
                        0554  1137 ;-
                        0554  1138
                        0554  1139          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                        0554  1140 EXE_ACCVIO::                             ; HERE FROM ACCVIO VECTOR
                        0554  1141 EXE$ACVIOLAT::                           ; Used by XDELTA only
      7E      0C   3C  0554  1142          MOVZWL  #SS$_ACCVIO, -(SP)    ; INDICATE ACCESS VIOLATION CONDITION
            05   DD     0557  1143          PUSHL   #5                      ; INDICATE SIGNAL ARRAY SIZE
          010B   31     0559  1144          BRW     EXE_EXCEPTION           ; PROCEED
                        055C  1145
                        055C  1146 ;+
                        055C  1147 ; ENTER HERE FOR TRANSLATION NOT VALID FAULT
                        055C  1148 ;-
                        055C  1149
                        055C  1150          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                        055C  1151 EXE_TRANSL::                             ; HERE FROM TRANSL VECTOR
                        055C  1152 MMG$PAGEFAULT::                          ; Used by XDELTA only
  006600A0 8F   DD      055C  1153          PUSHL   #DS$_TRANSL             ; INDICATE TRANSLATION CONDITION
            05   DD     0562  1154          PUSHL   #5                      ; INDICATE SIGNAL ARRAY SIZE
          0100   31     0564  1155          BRW     EXE_EXCEPTION           ; PROCEED
                        0567  1156
                        0567  1157 ;+
                        0567  1158 ; ENTER HERE FOR COMPATIBILITY FAULT OR ABORT
                        0567  1159 ;-
                        0567  1160
                        0567  1161          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                        0568  1162 EXE_COMPAT::                             ; HERE FROM COMPAT VECTOR
      7E    042C 8F   3C  0568  1163          MOVZWL  #SS$_COMPAT, -(SP)    ; INDICATE COMPATIBILITY CONDITION
          00EF   31     056D  1164          BRW     EXE_4ARG                ; SETUP 4 ARGS
```

```
                                0570  1166 ;+
                                0570  1167 ; ENTER HERE FOR ARITHMETIC TRAP
                                0570  1168 ;-
                                0570  1169
                                0570  1170         .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                                0570  1171 EXE_ARITH::                            ; HERE FROM ARITH VECTOR
                006600D0 8F     DD    0570  1172         PUSHL   #DS$_ARITH          ; Indicate Arithmetic fault
                      00E6      31    0576  1173         BRW     EXE_4ARG            ; Form 4 argument list
                                0579  1174
                                0579  1175 ;+
                                0579  1176 ; ENTER HERE FOR "ALL" BREAKPOINT FAULTS (I.E., DIAGNOSTIC PROGRAM
                                0579  1177 ; WILL ALWAYS GET SECOND CHANCE AT THIS VECTOR)
                                0579  1178 ;-
                                0579  1179
                                0579  1180         .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                                057C  1181 EXE_BREAK::                           ; HERE FROM BREAKPOINT VECTOR
                                057C  1182 EXE$BREAK::                           ; Used by XDELTA only
                                057C  1183
                00000000'EF     D5    057C  1184         TSTL    MP$GR_BOOTED_PROCESSORS     ; Are we an mp system?   [34]
                          62    13    0582  1185         BEQL    5$                          ; Not mp, therefore branch [34]
                       5E   04  C2    0584  1186         SUBL2   #4,SP                       ; Reserve storage
                       5E       DD    0587  1187         PUSHL   SP                          ; Storage for CPU identifier [34]
                00000000'EF     01    FB    0589  1188         CALLS   #1, MP$_GET_PROCESSOR_ID    ; Get current CPU identifier
            6E  00000000'EF     91    0590  1189         CMPB    DS$GB_PRIMARY_CPU_IDENTIFIER,(SP); Are we the primary? [34]
                          4A    13    0597  1190         BEQL    4$                          ; Branch if Pp           [34]
                       5E   04  C0    0599  1191         ADDL2   #4,SP                       ; Reset the stack        [34]
                                059C  1192 ;++
                                059C  1193 ; The following code is used to save the GPR context of an Ap in       [34]
                                059C  1194 ; the event the user wished to do EXAMINES and DEPOSITS after setting
                                059C  1195 ; a Breakpoint in the Ap code.
                                059C  1196 ;--
                                059C  1197
                    00000000'EF DC    059C  1198         MOVPSL  MP$GL_SAVED_PSL             ; Store the Ap PSL       [34]
                00000000'EF     50    7D    05A2  1199         MOVQ    R0,MP$GR_SAVED_GPRS         ; Store Ap GPR's         [34]
                00000008'EF     52    7D    05A9  1200         MOVQ    R2,MP$GR_SAVED_GPRS+8       ; ...
                00000010'EF     54    7D    05B0  1201         MOVQ    R4,MP$GR_SAVED_GPRS+16      ; ...
                00000018'EF     56    7D    05B7  1202         MOVQ    R6,MP$GR_SAVED_GPRS+24      ; ...
                00000020'EF     58    7D    05BE  1203         MOVQ    R8,MP$GR_SAVED_GPRS+32      ; ...
                00000028'EF     5A    7D    05C5  1204         MOVQ    R10,MP$GR_SAVED_GPRS+40     ; ...
                00000030'EF     5C    7D    05CC  1205         MOVQ    AP,MP$GR_SAVED_GPRS+48      ; ...
                00000038'EF     5E    D0    05D3  1206         MOVL    SP,MP$GR_SAVED_GPRS+56      ; ...
                0000003C'EF     6E    D0    05DA  1207         MOVL    (SP),MP$GR_SAVED_GPRS+60    ; ...                    [34]
                                03    11    05E1  1208         BRB     5$                          ; Continue               [34]
                                      05E3  1209
                          5E    04    C0    05E3  1210 4$:     ADDL2   #4,SP                       ; Reset stack        [34]
            7E  00000000'EF     03    CB    05E6  1211 5$:     BICL3   #3,DS$GA_BREAKVEC,-(SP); USE SOFT VECTOR
                                18    13    05EE  1212         BEQL    30$                 ; BRANCH IF NO SOFT VECTOR
                                      05F0  1213
                                01    BB    05F0  1214         PUSHR   #^M<R0>             ; NEED A LITTLE WORKING ROOM
                          50    00'   D0    05F2  1215         MOVL    S^#BPTMAX,R0        ; INIT AN INDEX
            00000000'EF40       08  AE    D1    05F5  1216 10$:    CMPL    8(SP),DS$AA_BPTADDR[R0] ; LOOK AT A SUPERVISOR'S KNOWN BPT'S
                                06    13    05FE  1217         BEQL    20$                 ; IT MATCHES, SO GO CALL HANDLER
                             F2 50    F4    0600  1218         SOBGEQ  R0,10$              ; LOOP THRU ENTIRE LIST (INCL. [0])
                                01    BA    0603  1219         POPR    #^M<R0>             ; RESTORE R0
                                05    05    0605  1220         RSB                         ; ENTER USER I.S.R
                                      0606  1221
                                01    BA    0606  1222 20$:    POPR    #^M<R0>             ; RESTORE REG
```

```
                6E   0414 8F   3C  0608  1223 30$:    MOVZWL  #SS$_BREAK, (SP)        ; INDICATE BREAKPOINT CONDITION
                          54   11  060D  1224          BRB     EXE_3ARG                ; SETUP 3 ARGS AND DISPATCH
                               060F  1225
                               060F  1226
                               060F  1227 ;+
                               060F  1228 ; ENTER HERE FOR "ALL" TBIT TRAPS (I.E., DIAGNOSTIC PROGRAM WILL ALWAYS
                               060F  1229 ; GET SECOND CHANCE AT THIS VECTOR)
                               060F  1230 ;-
                               060F  1231
                               060F  1232          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                               0610  1233 EXE_TBIT::                               ; HERE FROM TBIT VECTOR
                               0610  1234 EXE$TBIT::                               ; Used by XDELTA only
      7E   00000000'EF   03   CB  0610  1235          BICL3   #3,DS$GA_TBITVEC,-(SP)  ; Set secondary vector
                          08   13  0618  1236          BEQL    10$                     ; Branch if not set vec'ed
             01 00000000'EF   E8  061A  1237          BLBS    DEBUG$GB_FLAGS,10$      ; Branch if supervisor expects this
                          05   05  0621  1238          RSB                             ; Dispatch to user ISR
                               0622  1239
                   6E   07   D0  0622  1240 10$:    MOVL    #7,(SP)                 ; Assume compatability mode T-bit
                02 08 AE   1F   E0  0625  1241          BBS     #PSL$V_CM,8(SP), -
                               062A  1242                  15$                     ; BRANCH TO COMPATABILITY MODE HANDLER
                          03   11  062A  1243          BRB     16$                     ;
                        FF39   31  062C  1244 15$:    BRW     EXE_COMPAT              ;
                6E   0464 8F   3C  062F  1245 16$:    MOVZWL  #SS$_TBIT, (SP)         ; INDICATE T-BIT CONDITION
                          2D   11  0634  1246          BRB     EXE_3ARG                ; SETUP 3 ARGS, AND DISPATCH
```

```
                     0636  1248 ;+
                     0636  1249 ; HERE FOR UNEXPECTED INTERRUPT FROM SCB
                     0636  1250 ;-
                     0636  1251 EXE_UNEXPINT::
006600D8 8F    DD    0636  1252          PUSHL   #DS$_UNEXPINT           ; CODE FOR EXCEPTION ARGLIST
        0020   31    063C  1253          BRW     EXE_4ARG                ; USE COMMON CODE FOR REST
                     063F  1254
                     063F  1255 ;+
                     063F  1256 ; ENTER HERE FOR CHMK TRAP
                     063F  1257 ;-
                     063F  1258
                     063F  1259          .ALIGN  LONG
                     0640  1260 EXE_CHMK::
006600E0 8F    DD    0640  1261          PUSHL   #DS$_CHMK               ; Indicate extraneous CHMK TRAP
          17   11    0646  1262          BRB     EXE_4ARG                ; And build rest of 4 ARG list
                     0648  1263 ;+
                     0648  1264 ; ENTER HERE FOR CHME TRAP
                     0648  1265 ;-
                     0648  1266
                     0648  1267          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                     0648  1268 EXE_CHME::                               ; HERE FROM CHME VECTOR
006600A8 8F    DD    0648  1269          PUSHL   #DS$_CHME               ; INDICATE CHANGE MODE CONDITION
          0F   11    064E  1270          BRB     EXE_4ARG                ; SETUP 4 ARGS, NO SECONDARY VECTOR
                     0650  1271
                     0650  1272 ;+
                     0650  1273 ; ENTER HERE FOR CHMS TRAP
                     0650  1274 ;-
                     0650  1275
                     0650  1276          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                     0650  1277 EXE_CHMS::                               ; HERE FROM CHMS VECTOR
7E    041C 8F    3C    0650  1278          MOVZWL  #SS$_CMODSUPR, -(SP)    ; INDICATE CHANGE MODE CONDITION
          08   11    0655  1279          BRB     EXE_4ARG                ; SETUP 4 ARGS, NO SECONDARY VECTOR
                     0657  1280
                     0657  1281 ;+
                     0657  1282 ; ENTER HERE FOR CHMU TRAP
                     0657  1283 ;-
                     0657  1284
                     0657  1285          .ALIGN  LONG                    ; REQUIRED FOR EXCEPTION VECTOR
                     0658  1286 EXE_CHMU::                               ; HERE FROM CHMU VECTOR
7E    0424 8F    3C    0658  1287          MOVZWL  #SS$_CMODUSER, -(SP)    ; INDICATE CHANGE MODE CONDITION
          00   11    065D  1288          BRB     EXE_4ARG                ; SETUP 4 ARGS, NO SECONDARY VECTOR
                     065F  1289
                     065F  1290 EXE_4ARG:
          04   DD    065F  1291          PUSHL   #4                      ; 4 ARGS
          04   11    0661  1292          BRB     EXE_EXCEPTION           ; JOIN COMMON CODE
                     0663  1293
                     0663  1294 EXE_3ARG:
          03   DD    0663  1295          PUSHL   #3                      ; 3 ARGS
          00   11    0665  1296          BRB     EXE_EXCEPTION
```

```
                                0667  1298 ;+
                                0667  1299 ; COMMON CODE FOR ALL EXCEPTIONS.
                                0667  1300 ; STACK CONTAINS SIGNAL ARRAY, BUILD MECHANISM ARRARY
                                0667  1301 ; THEN SWITCH TO PREVIOUS MODE STACK UNLESS 'IS' IS SET OR PREVMODE WAS KERNEL
                                0667  1302 ;-
                                0667  1303
                                0667  1304 EXE$REFLECT::                          ;                                   [27]
                                0667  1305 EXE_EXCEPTION:
                     03   BB    0667  1306         PUSHR   #^M<R0,R1>             ; SAVE VOLATILE REGISTERS
                  7E 03   CE    0669  1307         MNEGL   #3,-(SP)               ; INITIAL FRAME DEPTH   [34]
                     5D   DD    066C  1308         PUSHL   FP                     ; INITIAL HANDLER ESTABLISHER FRAME
                     04   DD    066E  1309         PUSHL   #4                     ; MECHANISM ARRAY SIZE
                                0670  1310
            00000000'EF   D5    0670  1311         TSTL    MP$GR_BOOTED_PROCESSORS     ; Are we an mp system?   [34]
                     1E   13    0676  1312         BEQL    35$                         ; Not mp, therefore branch [34]
                  5E 04   C2    0678  1313         SUBL2   #4,SP                       ; Reserve storage        [34]
                     5E   DD    067B  1314         PUSHL   SP                          ; Storage for CPU identifier [34]
         00000000'EF   01   FB    067D  1315         CALLS   #1, MP$_GET_PROCESSOR_ID    ; Get current CPU identifier
      6E 00000000'EF   91   0684  1316         CMPB    DS$GB_PRIMARY_CPU_IDENTIFIER,(SP); Are we the primary? [34]
                     06   13    068B  1317         BEQL    30$                         ; Branch if Pp           [34]
                  5E 04   C0    068D  1318         ADDL2   #4,SP                       ; Reset the stack        [34]
                   0077   31    0690  1319         BRW     NORMAL                  ; Skip stack switch code   [34]
                                0693  1320
                  5E 04   C0    0693  1321 30$:    ADDL2   #4,SP                  ; Reset the stack       [34]
                     51   DC    0696  1322 35$:    MOVPSL  R1                     ; GET CURRENT PSL
                  6E 51 1A   E0    0698  1323         BBS     #PSL$V_IS,R1,NORMAL    ; INTERRUPT ON INTERRUPT STACK, PROCESS
               50 51 02   16   EF    069C  1324         EXTZV   #PSL$V_PRVMOD, -
                                06A1  1325                 #PSL$S_PRVMOD,R1,R0    ; GET PREVIOUS MODE
                     67   13    06A1  1326         BEQL    NORMAL                 ; BRANCH IF WAS IN KERNEL MODE, PROCESS
                                06A3  1327
                     0C   BB    06A3  1328         PUSHR   #^M<R2,R3>             ; SAVE WORKING REGISTERS
                  52 50   D0    06A5  1329         MOVL    R0,R2                  ; SAVE PREVIOUS MODE STACK POINTER
            53 1C AE 06   C1    06A8  1330         ADDL3   #6,28(SP),R3           ; CALCULATE NUMBER OF LONGWORDS TO MOVE
                                06AD  1331
         51 00000000'EF42   D0    06AD  1332         MOVL    PCB_BASE[R2],R1        ; GET SAVED SP FROM 'PCB'
                  51 52   DB    06B5  1333         MFPR    R2,R1                  ; GET HARDWARE SAVED PREV MODE SP
               50 53 02   78    06B8  1334         ASHL    #2,R3,R0               ; CALCULATE NUMBER OF BYTES TO MOVE
                  51 50   C2    06BC  1335         SUBL2   R0,R1                  ; SET UP A NEW STACK POINTER
               61 50 52   0D    06BF  1336         PROBEW  R2,R0,(R1)             ; CAN IT BE DONE ?
                     1C   12    06C3  1337         BNEQ    50$                    ; "NO PROBLEM", PROCEED
                                06C5  1338                                        ;
                                06C5  1339         ERRSUP_S  ,T_STKVIO            ; CANNOT COPY STACK !
                00000000'EF   DF    06C5                   PUSHAL  $MODULE
                000006FF'EF   DF    06CB                   PUSHAL  T_STKVIO
                     01   DD    06D1                   PUSHL   #$ER
            00000000'EF   03   FB    06D3                   CALLS   #3, DS_ERRSUP
                                06DA  1340         $DS_ABORT                      ; SO GO AHEAD & DO THE UGLY DEED
            00000000'9F   6C   FA    06DA                   CALLG   (AP), @#DS$ABORT
                                06E1  1341                                        ;
      00000000'EF40   51   D0    06E1  1342 50$:    MOVL    R1,PCB_BASE[R0]        ; STORE NEW SP INTO 'PCB'
                  52 51   DA    06E9  1343         MTPR    R1,R2                  ; STORE NEW HARDWARE SP
               50 08 AE   9E    06EC  1344         MOVAB   8(SP),R0               ; ADDRESS OF ARGS TO COPY
                  81 80   D0    06F0  1345 60$:    MOVL    (R0)+,(R1)+            ; COPY CURR STACK TO NEW ONE
                  FA 53   F5    06F3  1346         SOBGTR  R3,60$                 ; UNTIL DONE
                                06F6  1347
                                06F6  1348 ;+
                                06F6  1349 ; NOW MAKE THE SWITCH TO THE NEW STACK
```

```
                                06F6  1350 ;-
                                06F6  1351
      70   C8000010 8F    CA    06F6  1352          BICL    #PSL$M_CM!PSL$M_TBIT! -
                                06FD  1353                  PSL$M_FPD!PSL$M_TP,-(R0); CLEAR CM,T,TP,FPD
      70   0000070A'EF    9E    06FD  1354          MOVAB   L^NORMAL,-(R0)           ; PUSH ADDRESS TO RESUME AT IN PREV MODE[14]
                     0C    BA    0704  1355          POPR    #^M<R2,R3>               ; RESTORE WORKING REGISTERS
                5E   50    D0    0706  1356          MOVL    R0,SP                    ; POINT TO PC/PSL PAIR
                           02    0709  1357          REI                              ; RETURN IN PREVIOUS MODE AT 'NORMAL'
```

```
                                070A  ;+
                                070A  ; HERE TO BUILD THE CONDITION ARGLIST AND CALL THE HANDLER
                                070A  ;-
                                070A
                                070A  NORMAL:
                   6E   DF      070A          PUSHAL  (SP)                    ; POINT TO MECHANISM ARRAY
                18 AE   DF      070C          PUSHAL  24(SP)                  ; POINT TO SIGNAL ARRAY
                   02   DD      070F          PUSHL   #2                      ; INDICATE ARGLIST LENGTH
      00000000'EF  08   D1      0711          CMPL    #PR$_SID_TYPUV2, EXE$GB_CPUTYPE ; TEST CPU FOR MICROVAX [25][35] ;G:
                   06   12      0718          BNEQ    10$                     ; SKIP IF NOT MICROVAX          [25][35] ;G:
           00000000'EF  16      071A          JSB     VAX$MODIFY_EXCEPTION    ; GO DO PRE-PROCESSING                  [26]
                                0720  10$:
                                0720
                                0720  ;
                                0720  ; SEARCH FOR CONDITION HANDLER
                                0720  ;
                                0720
      0000076B'EF  6E   FA      0720  120$:   CALLG   (SP),L^SEARCH           ;SEARCH FOR CONDITION HANDLER           [14]
                0B 50   E9      0727          BLBC    R0,130$                 ;IF LBC TRY LAST CHANCE
                                072A
                                072A  ;
                                072A  ; CALL CONDITION HANDLER
                                072A  ;
           00000000'9F  16      072A          JSB     @#SYS$CALL_HANDL        ;CALL HANDLER VIA SYSTEM VECTOR
                ED 50   E9      0730          BLBC    R0,120$                 ;IF LBS TRY AGAIN
                   26   11      0733          BRB     140$                    ;CONTINUE
                                0735
                                0735  ;+
                                0735  ; HERE WHEN "RESIGNALLED"  (I.E., CONDITION HANDLER DIDN'T WANT IT)
                                0735  ;-
                                0735
      FFFFF8C4 EF  6E   FA      0735  130$:   CALLG   (SP), L^COND_HANDLR     ; Call last chance condition handler    [14]
                1C 50   E8      073C          BLBS    R0,140$                 ; HANDLED, CONTINUE                     [34]
                                073F
                                073F          ERRSUP_S  ,T_XCHFAILURE         ; UNRECOVERABLE SITUATION !
           00000000'EF  DF      073F                  PUSHAL  $MODULE
           0000069C'EF  DF      0745                  PUSHAL  T_XCHFAILURE
                   02   DD      074B                  PUSHL   #$ER
      00000000'EF  03   FB      074D                  CALLS   #3, DS_ERRSUP
                                0754          $DS_ABORT                       ; NO OTHER WAY AT THIS POINT
      00000000'9F  6C   FA      0754                  CALLG   (AP), @#DS$ABORT
                                075B
                                075B  ;+
                                075B  ; HERE TO CLEAN THE STACK AND RETURN
                                075B  ;-
                                075B
       6E   20 AE   07   C1     075B  140$:   ADDL3   #7,32(SP),(SP)          ; CALCULATE LONGWORD OFFSET TO SAVED PC
               6E   04   C4     0760          MULL2   #4,(SP)                 ; CALCULATE NUMBER OF BYTES TO REMOVE
            50   18 AE   7D     0763          MOVQ    24(SP),R0               ; RESTORE R0,R1
               5E   6E   C0     0767          ADDL2   (SP),SP                 ; REMOVE ARGUMENT LIST FROM STACK
                         02     076A          REI
```

```
                     076B  1406          .SBTTL  SEARCH FOR CONDITION HANDLER
                     076B  1407 ;
                     076B  1408 ; SEARCH - SEARCH FOR CONDITION HANDLER
                     076B  1409 ;
                     076B  1410 ; THIS IS A SPECIAL INTERNAL ROUTINE THAT IS CALLED IN THE INITIAL SEARCH
                     076B  1411 ; FOR A CONDITION HANDLER AND ON RESIGNAL FROM A PREVIOUSLY SIGNALLED
                     076B  1412 ; CONDITION.
                     076B  1413 ;
                     076B  1414 ; INPUTS:
                     076B  1415 ;
                     076B  1416 ;       00(AP) = NUMBER OF CONDITION ARGUMENTS.
                     076B  1417 ;       04(AP) = ADDRESS OF SIGNAL ARGUMENT LIST.
                     076B  1418 ;       08(AP) = ADDRESS OF MECHANISM ARGUMENT LIST.
                     076B  1419 ;       12(AP) = NUMBER OF MECHANISM ARGUMENTS.
                     076B  1420 ;       16(AP) = FP OF HANDLER ESTABLISHER FRAME.
                     076B  1421 ;       20(AP) = FRAME DEPTH.
                     076B  1422 ;       24(AP) = SAVED R0.
                     076B  1423 ;       28(AP) = SAVED R1.
                     076B  1424 ;       32(AP) = NUMBER OF SIGNAL ARGUMENTS.
                     076B  1425 ;       36(AP) = EXCEPTION NAME (INTEGER VALUE).
                     076B  1426 ;       40(AP) = FIRST EXCEPTION PARAMETER (IF ANY).
                     076B  1427 ;       44(AP) = SECOND EXCEPTION PARAMETER (IF ANY).
                     076B  1428 ;              .
                     076B  1429 ;              .
                     076B  1430 ;              .
                     076B  1431 ;       36+N*4(AP) = N'TH EXCEPTION PARAMETER (IF ANY).
                     076B  1432 ;       36+N*4+4(AP) = EXCEPTION PC.
                     076B  1433 ;       36+N*4+8(AP) = EXCEPTION PSL.
                     076B  1434 ;
                     076B  1435 ; OUTPUTS:
                     076B  1436 ;
                     076B  1437 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO LOCATE CONDITION HANDLER.
                     076B  1438 ;
                     076B  1439 ;              R0 = SS$_ACCVIO - STACK CANNOT BE READ FROM CURRENT MODE.
                     076B  1440 ;
                     076B  1441 ;              R0 = SS$_NOHANDLER - NO CONDITION HANDLER COULD BE FOUND.
                     076B  1442 ;
                     076B  1443 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                     076B  1444 ;
                     076B  1445 ;              R1 = ADDRESS OF CONDITION HANDLER.
                     076B  1446 ;
                     076B  1447
00000000             076B  1448          HANDLER = 0                     ; CONDITION HANDLER ADDRESS
00000004             076B  1449          SAVPSW  = 4                     ; SAVED PSW FROM CALL
00000006             076B  1450          SAVMSK  = 6                     ; REGISTER SAVE MASK
00000008             076B  1451          SAVAP   = 8                     ; SAVED AP REGISTER IMAGE
0000000C             076B  1452          SAVFP   = 12                    ; SAVED FP REGISTER IMAGE
00000010             076B  1453          SAVPC   = 16                    ; SAVED PC REGISTER IMAGE
00000014             076B  1454          SAVRG   = 20                    ; OTHER SAVED REGISTER IMAGES
```

```
                        076B  1456 SEARCH:                                ;SEARCH FOR CONDITION HANDLER
                0000    076B  1457          .WORD   0                      ;ENTRY MASK
    6D  00000811'EF     076D  1458          MOVAL   L^CHANDL,(FP)          ;SET ADDRESS OF CONDITION HANDLER       [14]
                        0774  1459
          50  10 AC     0774  1460 10$:     MOVL    16(AP),R0              ;GET PREVIOUS FRAME ADDRESS
                        0778  1461
                51      0778  1462 20$:     MOVPSL  R1                     ;READ CURRENT PSL
    51    51  02  18    077A  1463          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1 ;EXTRACT CURRENT MODE
                14 AC   077F  1464          INCL    20(AP)                 ;INCREMENT FRAME DEPTH
                28      0782  1465          BEQL    50$                    ;IF EQL FIRST STACK FRAME
                18      0784  1466          BGTR    40$                    ;IF GTR OTHER STACK FRAME
    50  00000007'9F41   0786  1467          MOVAQ   @#CTL$AQ_EXCVEC[R1],R0 ;GET ADDRESS OF EXCEPTION VECTOR QUADWORD
        14 AC   FE 8F   078E  1468          CMPB    #-2,20(AP)             ;EXAMINE PRIMARY VECTOR?
                02      0793  1469          BEQL    30$                    ;IF EQL YES
                80      0795  1470          TSTL    (R0)+                  ;ADJUST TO SECONDARY VECTOR
                        0797  1471
              51  60    0797  1472 30$:     MOVL    (R0),R1                ;GET ADDRESS OF CONDITION HANDLER
                37      079A  1473          BNEQ    60$                    ;IF NEQ CONDITION HANDLER FOUND
                D6      079C  1474          BRB     10$                    ;
                        079E  1475
           69 16 AC     079E  1476 40$:     BLBS    22(AP),100$            ;IF LBS SEARCH COUNT OVERFLOW
          50   0C A0     07A2  1477          MOVL    SAVFP(R0),R0           ;GET ADDRESS OF PREVIOUS FRAME
                63      07A6  1478          BEQL    100$                   ;IF EQL NONE
          10 AC   50      07A8  1479          MOVL    R0,16(AP)              ;SAVE ADDRESS OF ESTABLISHER FRAME
                        07AC  1480
00000027'EF41   50      07AC  1481 50$:     CMPL    R0,CTL$AL_STACK[R1]    ;FRAME POINTER WITHIN STACK RANGE?
                55      07B4  1482          BGTRU   100$                   ;IF GTRU NO
        00000000'EF     07B6  1483          PUSHAB  STACK_BASE             ;SET LOWER LIMIT OF USER STACK
              51  03    07BC  1484          CMPL    #PSL$C_USER,R1         ;CURRENT MODE USER?
                08      07BF  1485          BEQL    55$                    ;IF EQL YES
    6E  00000023'EF41   07C1  1486          MOVL    CTL$AL_STACK-4[R1],(SP) ;SET CURRENT MODE STACK LIMIT
                        07C9  1487
              8E  50    07C9  1488 55$:     CMPL    R0,(SP)+               ;FRAME POINTER WITHIN STACK RANGE?
                3D      07CC  1489          BLSSU   100$                   ;IF LSSU NO
              51  60    07CE  1490          MOVL    (R0),R1                ;GET ADDRESS OF CONDITION HANDLER
                04      07D1  1491          BEQL    70$                    ;IF EQL NONE
                        07D3  1492
              50  01    07D3  1493 60$:     BISL    #1,R0                  ;INDICATE SUCCESSFUL COMPLETION
                04      07D6  1494          RET                            ;
                        07D7  1495
 10 A0  00000004'8F     07D7  1496 70$:     CMPL    #SYS$CALL_HANDL+4,SAVPC(R0) ;CALL FROM CONDITION DISPATCHER?
                93      07DF  1497          BNEQ    10$                    ;IF NEQ NO
51  06 A0  0C  00      07E1  1498          EXTZV   #0,#12,SAVMSK(R0),R1   ;GET REGISTER SAVE MASK
7E  06 A0  02  0E      07E7  1499          EXTZV   #14,#2,SAVMSK(R0),-(SP) ;GET STACK ALIGNMENT BIAS
              50  14    07ED  1500          ADDL    #SAVRG,R0              ;ADD OFFSET TO REGISTER SAVE AREA
              50  8E    07F0  1501          ADDL    (SP)+,R0               ;ADD STACK ALIGNMENT BIAS
                        07F3  1502
             03 51      07F3  1503 80$:     BLBC    R1,90$                 ;IF LBC CORRESPONDING REGISTER NOT SAVED
              50  04    07F6  1504          ADDL    #4,R0                  ;ADJUST FOR SAVED REGISTER
                        07F9  1505
    51  51  FF 8F   78  07F9  1506 90$:     ASHL    #-1,R1,R1              ;ANY MORE REGISTERS SAVED?
                F3      07FE  1507          BNEQ    80$                    ;IF NEQ YES
          50  0C A0    0800  1508          MOVL    CHF$L_MCHARGLST+4(R0),R0 ;GET ADDRESS OF MECHANISM ARGUMENTS
          50  04 A0    0804  1509          MOVL    CHF$L_MCH_FRAME(R0),R0 ;GET ADDRESS OF ESTABLISHER FRAME
                FF6D    0808  1510          BRW     20$                    ;
                        080B  1511
        50  08F8 8F    080B  1512 100$:    MOVZWL  #SS$_NOHANDLER,R0      ;SET NO HANDLER FOUND
```

```
                     04  0810  1513          RET                                 ;
                         0811  1514
                         0811  1515 ;
                         0811  1516 ; CONDITION HANDLER FOR SEARCH ROUTINE EXCEPTIONS
                         0811  1517 ;
                         0811  1518
                         0811  1519 CHANDL:                                       ;SEARCH ROUTINE CONDITION HANDLER
                   0000  0811  1520          .WORD   0                           ;ENTRY MASK
          50   04 AC    D0  0813  1521          MOVL    CHF$L_SIGARGLST(AP),R0      ;GET ADDRESS OF SIGNAL ARGUMENTS
    04 A0   0920 8F    B1  0817  1522          CMPW    #SS$_UNWIND, -              ;UNWINDING?
                           081D  1523                    CHF$L_SIG_NAME(R0)        ;
                12    13  081D  1524          BEQL    10$                         ;IF EQL YES
          51   08 AC    D0  081F  1525          MOVL    CHF$L_MCHARGLST(AP),R1      ;GET ADDRESS OF MECHANISM ARGUMENTS
       0C A1   04 A0    D0  0823  1526          MOVL    CHF$L_SIG_NAME(R0),CHF$L_MCH_SAVR0(R1) ;SET FINAL STATUS
                7E    7C  0828  1527          CLRQ    -(SP)                       ;CLEAR DEPTH AND NEW PC ARGUMENTS
00000000'EF    02    FB  082A  1528          CALLS   #2,SYS$UNWIND               ;UNWIND TO ESTABLISHER'S CALLER
                         0831  1529
                     04  0831  1530 10$:     RET                                 ;
```

```
                                 0832  1532 .SBTTL DSX$SETPRIEXV - ESTABLISH PRIMARY EXCEPTION VECTOR                [19]
                                 0832  1533
                                 0832  1534 ;FUNCTIONAL DESCRIPTION                                                  [19]
                                 0832  1535 ;       This routine services the DS$SETPRIEXV macro call, used by a    [19]
                                 0832  1536 ;       program running under the DS when the progam wishes to establish[19]
                                 0832  1537 ;       a primary exception vector to field exception conditions before [19]
                                 0832  1538 ;       the supervisor's condition handler takes over.                  [19]
                                 0832  1539 ;                                                                       [19]
                                 0832  1540 ;INPUTS                                                               [19]
                                 0832  1541 ;       4(AP) = Address of the condition handler                        [19]
                                 0832  1542 ;OUTPUTS                                                              [19]
                                 0832  1543 ;       None                                                            [19]
                                 0832  1544 ;                                                                       [19]
                                 0832  1545
                          0004   0832  1546 .ENTRY  DSX$SETPRIEXV, ^M<R2>          ;                              [19]
                                 0834  1547
                      1C    E1   0834  1548         BBC     #DSA$V_USER,-          ;IF USER MODE                  [19]
                 12 0000FE00'EF    0836  1549                 DSA$GL_FLAGS,100$      ;THEN                          [19]
                                 083C  1550         $SETEXV_S       #0,a4(AP)      ;   CALL SYSTEM SERVICE        [19]
                      00    DD   083C                PUSHL   #0
                      00    DD   083E                PUSHL   #0
                   04 BC    DF   0840                PUSHAL  a4(AP)
                      00    DD   0843                PUSHL   #0
            00000000'GF     04   FB   0845                CALLS   #4,G^SYS$SETEXV
                      20    11   084C  1551         BRB     200$                   ;ELSE                          [19]
        00000007'EF   04 AC   D0   084E  1552 100$:  MOVL    4(AP),CTL$AQ_EXCVEC    ;   SET KERNEL PRIMARY VECTOR  [19]
        0000000F'EF   04 AC   D0   0856  1553        MOVL    4(AP),CTL$AQ_EXCVEC+8  ;   SET EXEC   PRIMARY VECTOR  [19]
        00000017'EF   04 AC   D0   085E  1554        MOVL    4(AP),CTL$AQ_EXCVEC+16 ;   SET SUPER  PRIMARY VECTOR  [19]
        0000001F'EF   04 AC   D0   0866  1555        MOVL    4(AP),CTL$AQ_EXCVEC+24 ;   SET USER   PRIMARY VECTOR  [19]
                      04    086E  1556 200$:  RET                                 ;RETURN                        [19]
                            086F  1557
                            086F  1558         .END
```

```
$$ARGS                          =  0000000B     D
$$N                             =  00000001     D
$$T1                            =  00000000     D
$ER                             =  00000003     D
$MODULE                            00000000 R   D   03
$SRVCODE_BOOTATTACHED           =  00000000     D
$SRVCODE_CLEAR_INTERLOCK        =  00000009     D
$SRVCODE_GETSYSBUF              =  00000004     D
$SRVCODE_HALTATTACHED           =  00000002     D
$SRVCODE_INTERLOCK_TEST         =  00000007     D
$SRVCODE_PPSCB                  =  0000000A     D
$SRVCODE_PRINTREV               =  00000006     D
$SRVCODE_RELSYSBUF              =  00000005     D
$SRVCODE_SET_INTERLOCK          =  00000008     D
$SRVCODE_SHOWIDLE               =  00000003     D
$SRVCODE_STARTATTACHED          =  00000001     D
AAT_ARITH                          0000016B RG  D   03
AAT_COMPAT                         000002F0 R   D   03
APC$_ABORT                      =  00000006     D
APC$_CONTINUE                   =  00000009     D
APC$_DUMP                       =  00000003     D
APC$_NOP                        =  00000000     D
APC$_START                      =  00000005     D
APC$_ZERO                       =  00000004     D
APM$_ABTDON                     =  00000006     D
APM$_DEVERR                     =  00000002     D
APM$_DONE                       =  0000000A     D
APM$_EXCEPT                     =  0000000B     D
APM$_HARDERR                    =  00000003     D
APM$_MORE                       =  00000008     D
APM$_NOMES                      =  00000000     D
APM$_PREPERR                    =  0000000C     D
APM$_PRGERR                     =  00000005     D
APM$_SOFTERR                    =  00000004     D
APM$_SPOOL                      =  00000009     D
APM$_SYSERR                     =  00000001     D
APSLMNE                            ********   X   00
BIIC$L_BER                      =  00000008     D
BIT...                          =  00000004     D
BI_BER_BIT                         ********W G      00
BPTMAX                             ********   X    00
BUFFER                             FFFFFF00     D
BUFSIZ                          =  00000100     D
CHANDL                             00000811 R   D   04
CHF$L_MCHARGLST                 =  00000008     D
CHF$L_MCH_FRAME                 =  00000004     D
CHF$L_MCH_SAVR0                 =  0000000C     D
CHF$L_SIGARGLST                 =  00000004     D
CHF$L_SIG_NAME                  =  00000004     D
CHR$MCHK                           ********   X   00
CLI$K_BUFSIZ                    =  00000100     D
CLI$K_SIZE                         0000044C     D
CLI$L_ADDRESS                      00000018     D
CLI$L_COMMAND                      00000004     D
CLI$L_DATA                         0000001C     D
CLI$L_FLAGS                        00000000     D
CLI$L_FLAGS2                       00000444     D
CLI$L_LAST                         00000024     D
CLI$L_NEXT                         00000030     D
CLI$L_PASS                         0000002C     D
CLI$L_SUBT                         00000028     D
CLI$L_TEMP_BASE                    00000448     D
CLI$L_TEST                         00000020     D
CLI$M_START_OR_RUN              =  00000008     D
CLI$Q_BUFQWD                       00000034     D
CLI$Q_FILE                         00000008     D
CLI$Q_SECTION                      00000010     D
CLI$Q_TIME                         0000043C     D
CLI$T_BUFFER                       0000003C     D
CLI$V_ADAPTER                   =  00000018     D
CLI$V_ADR                       =  0000000B     D
CLI$V_ASCII                     =  00000013     D
CLI$V_BASE_QUAL                 =  00000002     D
CLI$V_BREAK                     =  0000000A     D
CLI$V_BRIEF                     =  0000001B     D
CLI$V_BYTE                      =  0000000D     D
CLI$V_CLEAR                     =  00000002     D
CLI$V_DEC                       =  00000010     D
CLI$V_DEFAULT                   =  0000000C     D
CLI$V_DEPOSIT                   =  00000019     D
CLI$V_EVENT                     =  00000008     D
CLI$V_EXAM                      =  00000005     D
CLI$V_FLAGS                     =  00000009     D
CLI$V_HEX                       =  00000012     D
CLI$V_KERNEL                    =  00000017     D
CLI$V_LOAD                      =  00000006     D
CLI$V_LONG                      =  0000000F     D
CLI$V_NEXT                      =  00000001     D
CLI$V_NOALLOC                   =  00000000     D
CLI$V_NOTNUF                    =  00000001     D
CLI$V_OCT                       =  00000011     D
CLI$V_PREG                      =  0000001A     D
CLI$V_QA                        =  00000007     D
CLI$V_QACKLOOPLOOPS             =  0000001C     D
CLI$V_QAERRORPRINTS             =  0000001B     D
CLI$V_QAMULTIPLEPASS            =  0000001F     D
CLI$V_QASUBTESTLOOPS            =  0000001E     D
CLI$V_QATESTLOOPS               =  0000001D     D
CLI$V_REG                       =  00000014     D
CLI$V_REQUIRED                  =  00000000     D
CLI$V_RUN                       =  00000015     D
CLI$V_SET                       =  00000003     D
CLI$V_SHOW                      =  00000004     D
CLI$V_START_OR_RUN              =  00000003     D
CLI$V_VALSEC                    =  00000016     D
CLI$V_WORD                      =  0000000E     D
CMK$_APBOOT                     =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
```

```
CMK$_REFRESH                    = 00000005     D
CMK$_SCHDWK                     = 00000006     D
CMK$_SETIMR                     = 0000000C     D
CMK$_SETIPL                     = 00000009     D
CMK$_SETPRT                     = 0000000D     D
CMK$_SGIPR                      = 0000000A     D
CMK$_TCONSOLE                   = 00000002     D
COMMON                            00000166 R   D  04
COND_DEBUG                        ********     X  00
COND_HANDLR                       00000000 RG  D  04
COND_HANDLR_CLI                   00000052 R   D  04
COND_HANDLR_EXIT                  00000126 R   D  04
CTL$AL_STACK                      00000027 R   D  03
CTL$AQ_EXCVEC                     00000007 R   D  03
CVTREG$_DATA                    = 00000008     D
CVTREG$_MAXLEN                  = 00000014     D
CVTREG$_MNEADR                  = 0000000C     D
CVTREG$_MSB                     = 00000004     D
CVTREG$_NARGS                   = 0000000B     D
CVTREG$_STRBUF                  = 00000010     D
CVTREG$_V1                      = 00000018     D
CVTREG$_V2                      = 0000001C     D
CVTREG$_V3                      = 00000020     D
CVTREG$_V4                      = 00000024     D
CVTREG$_V5                      = 00000028     D
CVTREG$_V6                      = 0000002C     D
DEBUG$GB_FLAGS                    ********     X  00
DIR...                          = FFFFFFFF     D
DS$AA_BPTADDR                     ********     X  00
DS$ABORT                          ********     X  00
DS$CLI                            ********     X  00
DS$CVTREG                         ********     X  00
DS$GA_BREAKVEC                    ********     X  00
DS$GA_TBITVEC                     ********     X  00
DS$GB_PRIMARY_CPU_IDENTIFIER      ********     X  00
DS$GB_TYPECODE                    ********     X  00
DS$GL_FLAGS                       ********     X  00
DS$GW_TTOUT                       ********     X  00
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
DS$K_TYPE_ERRSOFT               = 00000007     D
DS$K_TYPE_ERRSUP                = 00000004     D
DS$K_TYPE_ERRSYS                = 00000005     D
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_NI                         = 04000000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMED                      = 02000000     D
DS$M_TIMED_OUT                  = 08000000     D
DS$M_TIMRON                     = 00020000     D
```

```
DS$SERVICE_DISPATCHER           ********  X   00
DS$V_ABRTFLG                  = 00000006    D
DS$V_BADTIME                  = 00000014    D
DS$V_BATCH                    = 00000016    D
DS$V_BRKCLR                   = 0000000C    D
DS$V_BRKPT                    = 0000000B    D
DS$V_CHARFLG                  = 00000008    D
DS$V_CMDFLG                   = 00000007    D
DS$V_CTRLC                    = 00000000    D
DS$V_CTRLO                    = 00000010    D
DS$V_DEVFLG                   = 00000009    D
DS$V_DISABLCC                 = 00000018    D
DS$V_DONFLG                   = 0000000D    D
DS$V_ERRFLG                   = 00000004    D
DS$V_EXCEPT                   = 00000013    D
DS$V_EXETST                   = 00000012    D
DS$V_HLTFLG                   = 00000003    D
DS$V_LODFLG                   = 00000001    D
DS$V_MEMMGT                   = 0000000F    D
DS$V_NI                       = 0000001A    D
DS$V_OUTPUT                   = 00000017    D
DS$V_RUBFLG                   = 00000005    D
DS$V_SCRIPT                   = 00000015    D
DS$V_SETIMR                   = 00000019    D
DS$V_STRFLG                   = 00000002    D
DS$V_SUBT                     = 0000000E    D
DS$V_SYSFLG                   = 0000000A    D
DS$V_TIMED                    = 00000019    D
DS$V_TIMED_OUT                = 0000001B    D
DS$V_TIMRON                   = 00000011    D
DS$_AP_NORMAL_BREAK           = 00660150    D
DS$_ARITH                     = 006600D0    D
DS$_ASBE                      = 00660118    D
DS$_BADLINK                   = 006600F0    D
DS$_BADTYPE                   = 006600E8    D
DS$_BIIC                      = 00660120    D
DS$_CHME                      = 006600A8    D
DS$_CHMK                      = 006600E0    D
DS$_DEVNAME                   = 00660108    D
DS$_ERROR                     = 00660002    D
DS$_FHWE                      = 00660068    D
DS$_FRAGBUF                   = 00660080    D
DS$_GK_EXE_MCHK               = 00000000    D
DS$_GK_INITSCB                = 00000002    D
DS$_GK_INT_WORK               = 00000003    D
DS$_GK_TST$MCHK               = 00000001    D
DS$_ICBUSY                    = 006600C8    D
DS$_ICERR                     = 006600C0    D
DS$_IHWE                      = 00660060    D
DS$_ILLCHAR                   = 00660018    D
DS$_ILLPAGCNT                 = 00660078    D
DS$_ILLUNIT                   = 00660100    D
DS$_INIT_FAIL                 = 00660140    D
DS$_INSFMEM                   = 00660050    D
DS$_INVCPU                    = 00660130    D
DS$_IPL2HI                    = 006600B8    D
DS$_IVADDR                    = 00660040    D
DS$_IVVECT                    = 00660038    D
DS$_KRNLSTK                   = 00660090    D
DS$_LOGIC                     = 00660070    D
DS$_MCHK                      = 00660088    D
DS$_MEM_ALLOC_ERR             = 00660138    D
DS$_MMOFF                     = 00660058    D
DS$_NEEDUNIT                  = 006600F8    D
DS$_NODE                      = 00660128    D
DS$_NOPCS                     = 00660110    D
DS$_NORMAL                    = 00660001    D
DS$_NOSUPPORT                 = 006600B1    D
DS$_NOTALLOWMP                = 00660148    D
DS$_NOTDON                    = 00660030    D
DS$_NOTIMP                    = 006600B0    D
DS$_NULLSTR                   = 00660010    D
DS$_OVERFLOW                  = 00660008    D
DS$_POWER                     = 00660098    D
DS$_PROGERR                   = 00660020    D
DS$_SEVERE                    = 00660004    D
DS$_TRANSL                    = 006600A0    D
DS$_TRUNCATE                  = 00660028    D
DS$_UNEXPINT                  = 006600D8    D
DS$_UNSUPPDEV                 = 00660158    D
DS$_VASFULL                   = 00660048    D
DS$_WARNING                   = 00660000    D
DSA$AL_APTMAIL                  0000FE00    D
DSA$AT_APTTXT                   0000FA00    D
DSA$GL_APTCOM                   0000FE04    D
DSA$GL_DEVLEN                   0000FE58    D
DSA$GL_ERRNO                    0000FE44    D
DSA$GL_EVENT                    0000FE48    D
DSA$GL_FLAGS                    0000FE00    D
DSA$GL_MSGTYP                   0000FE40    D
DSA$GL_PASSES                   0000FE08    D
DSA$GL_PASSNO                   0000FE54    D
DSA$GL_SECTNO                   0000FE10    D
DSA$GL_SID                      0000FE14    D
DSA$GL_SUBTNO                   0000FE4C    D
DSA$GL_TESTNO                   0000FE50    D
DSA$GL_UNITS                    0000FE0C    D
DSA$GQ_MSGPTR                   0000FE68    D
DSA$GT_DEVNAM                   0000FE5C    D
DSA$V_USER                    = 0000001C    D
DSR$FAULT_CLEAR                 ********  X   00
DSR$FIND_USER_PC                ********  X   00
DSX$PRINTB                      ********  X   00
DSX$PRINTI                      ********  X   00
DSX$PRINTSIG                    00000148 RG  D  04
DSX$PRINTSIGI                   0000015A RG  D  04
DSX$PRINTX                      ********    X   00
DSX$SETPRIEXV                   00000832 RG  D  04
DS_ERRSUP                       ********    X   00
ESTKPTR                         ********    X   00
EXCPT                           00000132 R   D  04
EXE$ACVIOLAT                    00000554 RG  D  04
EXE$BREAK                       0000057C RG  D  04
EXE$GB_CPUTYPE                  ********    X   00
```

```
EXE$REFLECT                     00000667 RG D  04
EXE$ROPRAND                     00000544 RG D  04
EXE$TBIT                        00000610 RG D  04
EXE_3ARG                        00000663 R  D  04
EXE_4ARG                        0000065F R  D  04
EXE_ACCVIO                      00000554 RG D  04
EXE_ARITH                       00000570 RG D  04
EXE_BREAK                       0000057C RG D  04
EXE_CHME                        00000648 RG D  04
EXE_CHMK                        00000640 RG D  04
EXE_CHMS                        00000650 RG D  04
EXE_CHMU                        00000658 RG D  04
EXE_COMPAT                      00000568 RG D  04
EXE_EXCEPTION                   00000667 R  D  04
EXE_KRNLSTK                     0000051C RG D  04
EXE_MCHK                        000004FC RG D  04
EXE_OPCCUS                      0000053C RG D  04
EXE_OPCDEC                      00000534 RG D  04
EXE_POWER                       00000528 RG D  04
EXE_RADRMOD                     0000054C RG D  04
EXE_ROPRAND                     00000544 RG D  04
EXE_TBIT                        00000610 RG D  04
EXE_TRANSL                      0000055C RG D  04
EXE_UNEXPINT                    00000636 RG D  04
FIND_USERPC                     000004A0 RG D  04
FMT_BER                         ********W G    00
FMT_BI_UNEXPINT                 0000041F R  D  03
FMT_CHMX_ARG                    00000523 R  D  03
FMT_ERRPC                       00000465 R  D  03
FMT_ERRPSL                      000004B9 R  D  03
FMT_EXCEPT                      000003BD R  D  03
FMT_EXCEP_CODE                  0000050C R  D  03
FMT_PXXX                        0000056D R  D  03
FMT_UMCHK                       00000383 R  D  03
FMT_UNEXPINT                    000003E7 R  D  03
FMT_UNK_ARG                     00000536 R  D  03
FMT_UNK_CNT                     00000558 R  D  03
FMT_USER_PC_NF                  0000049A R  D  03
FMT_USRPC                       0000047E R  D  03
FMT_VIRT                        000004DA R  D  03
HANDLER                       = 00000000    D
ID_TEMP                         00000014 R  D  02
IO$M_CANCTRLO                   ********    X  00
IO$_WRITEVBLK                   ********    X  00
IS                            = 00000001    D
ISTKPTR                         ********    X  00
LOCAL                           FFFFFF00    D
MMG$PAGEFAULT                   0000055C RG D  04
MODELST                         ********    X  00
MP$GL_AP_SIGNAL_ARRAY           ********W G    00
MP$GL_SAVED_PSL                 ********W G    00
MP$GR_BOOTED_PROCESSORS         ********    X  00
MP$GR_SAVED_GPRS                ********W G    00
MP$V_IL_ALLN                  = 00000004    D
MP$V_IL_CNDB                  = 00000007    D
MP$V_IL_CNHN                  = 00000006    D
MP$V_IL_DELN                  = 00000005    D
MP$V_IL_DSBK                  = 00000008    D
MP$V_IL_ERSP                  = 00000003    D
MP$V_IL_GTLN                  = 00000009    D
MP$V_IL_NUMT                  = 00000000    D
MP$V_IL_PRNT                  = 00000001    D
MP$V_IL_RRPT                  = 00000002    D
MP$_COND_HANDLR                 ********W G    00
MP$_GET_PROCESSOR_ID            ********    X  00
MP$_RESUME_ATTACHED_PROCESSORS  ********    X  00
MSG_ACCVIO                      000000B7 R  D  03
MSG_BREAK                       000000F3 R  D  03
MSG_CHME                        0000012F R  D  03
MSG_CHMK                        0000011B R  D  03
MSG_CHMS                        00000143 R  D  03
MSG_CHMU                        00000157 R  D  03
MSG_COMPAT                      00000107 R  D  03
MSG_DECOVF                      0000028C R  D  03
MSG_FLTDIV                      00000264 R  D  03
MSG_FLTDIV_F                    000002C8 R  D  03
MSG_FLTOVF                      00000250 R  D  03
MSG_FLTOVF_F                    000002B4 R  D  03
MSG_FLTUND                      00000278 R  D  03
MSG_FLTUND_F                    000002DC R  D  03
MSG_INTDIV                      0000023C R  D  03
MSG_INTOVF                      00000228 R  D  03
MSG_KRNLSTK                     0000003F R  D  03
MSG_OPCCUS                      0000007B R  D  03
MSG_OPCDEC                      00000067 R  D  03
MSG_POWER                       00000053 R  D  03
MSG_RADRMOD                     000000A3 R  D  03
MSG_ROPRAND                     0000008F R  D  03
MSG_SUBRNG                      000002A0 R  D  03
MSG_TBIT                        000000DF R  D  03
MSG_TRANSL                      000000CB R  D  03
MSG_UMCHK                       00000037 R  D  03
NEW_AP_FP                       00000004 R  D  02
NEW_PC_PSL                      0000000C R  D  02
NEW_SP                          00000000 R  D  02
NORMAL                          0000070A R  D  04
PCB_BASE                        ********    X  00
PR$_SID_TYP8SS                = 00000005    D
PR$_SID_TYPUV2                = 00000008    D
PRINT_CHMX_ARG                  0000041E R  D  04
PRINT_PC_PSL                    00000433 R  D  04
PRINT_TYPE_VA                   00000407 R  D  04
PRINT_UNK_EXC                   0000030D R  D  04
PSL$C_USER                    = 00000003    D
PSL$M_CM                      = 80000000    D
PSL$M_FPD                     = 08000000    D
PSL$M_TBIT                    = 00000010    D
PSL$M_TP                      = 40000000    D
PSL$S_CURMOD                  = 00000002    D
PSL$S_PRVMOD                  = 00000002    D
PSL$V_CM                      = 0000001F    D
PSL$V_CURMOD                  = 00000018    D
PSL$V_IS                      = 0000001A    D
PSL$V_PRVMOD                  = 00000016    D
```

```
RESIGNAL                        00000494 R   D  04
SAVABS...                     = FFFFFF00     D
SAVAP                         = 00000008     D
SAVFP                         = 0000000C     D
SAVMSK                        = 00000006     D
SAVPC                         = 00000010     D
SAVPSW                        = 00000004     D
SAVRG                         = 00000014     D
SCB$L_ACCESS                    00000020     D
SCB$L_ARITH                     00000034     D
SCB$L_BREAK                     0000002C     D
SCB$L_CHME                      00000044     D
SCB$L_CHMK                      00000040     D
SCB$L_CHMS                      00000048     D
SCB$L_CHMU                      0000004C     D
SCB$L_COMPAT                    00000030     D
SCB$L_KNLSTK                    00000008     D
SCB$L_MACHCHK                   00000004     D
SCB$L_OPCCUS                    00000014     D
SCB$L_OPCDEC                    00000010     D
SCB$L_POWER                     0000000C     D
SCB$L_RADRMOD                   0000001C     D
SCB$L_ROPRAND                   00000018     D
SCB$L_RXDB                      000000F8     D
SCB$L_SFTLVL1                   00000084     D
SCB$L_SFTLVL10                  000000A8     D
SCB$L_SFTLVL11                  000000AC     D
SCB$L_SFTLVL12                  000000B0     D
SCB$L_SFTLVL13                  000000B4     D
SCB$L_SFTLVL14                  000000B8     D
SCB$L_SFTLVL15                  000000BC     D
SCB$L_SFTLVL2                   00000088     D
SCB$L_SFTLVL3                   0000008C     D
SCB$L_SFTLVL4                   00000090     D
SCB$L_SFTLVL5                   00000094     D
SCB$L_SFTLVL6                   00000098     D
SCB$L_SFTLVL7                   0000009C     D
SCB$L_SFTLVL8                   000000A0     D
SCB$L_SFTLVL9                   000000A4     D
SCB$L_TBIT                      00000028     D
SCB$L_TIMER                     000000C0     D
SCB$L_TRANSL                    00000024     D
SCB$L_TXDB                      000000FC     D
SCB$L_VM                        00000038     D
SCB$L_ZERO                      00000000     D
SCB_IMAGE                       ********   X   00
SCRIPT$STOP                     ********   X   00
SEARCH                          0000076B R   D  04
SIZ...                        = 00000001     D
SS$_ACCVIO                    = 0000000C     D
SS$_BREAK                     = 00000414     D
SS$_CMODSUPR                  = 0000041C     D
SS$_CMODUSER                  = 00000424     D
SS$_COMPAT                    = 0000042C     D
SS$_CONTINUE                  = 00000001     D
SS$_MCHECK                    = 000002BC     D
SS$_NOHANDLER                 = 000008F8     D
SS$_OPCCUS                    = 00000434     D
SS$_OPCDEC                    = 0000043C     D
SS$_RADRMOD                   = 0000044C     D
SS$_RESIGNAL                  = 00000918     D
SS$_ROPRAND                   = 00000454     D
SS$_TBIT                      = 00000464     D
SS$_UNWIND                    = 00000920     D
SSTKPTR                         ********   X   00
STACK_BASE                      ********   X   00
SYS$CALL_HANDL                  ********   X   00
SYS$QIO                         ********   G   04
SYS$SETEXV                      ********   G   04
SYS$UNWIND                      ********   X   00
TST$MCHK                        000004E4 RG  D  04
T_ABORT                         000006D0 R   D  03
T_ACCVIO                        0000060B R   D  03
T_BREAK                         00000640 R   D  03
T_CHME                          0000065E R   D  03
T_CHMK                          0000064B R   D  03
T_CHMS                          00000674 R   D  03
T_CHMU                          0000068B R   D  03
T_COMPAT                        000006EC R   D  03
T_DECOVF                        0000078C R   D  03
T_FAULT                         000006CA R   D  03
T_FLTDIV                        00000759 R   D  03
T_FLTOVF                        00000747 R   D  03
T_FLTUND                        00000779 R   D  03
T_FLT_ABO                       000006E0 R   D  03
T_INTDIV                        00000730 R   D  03
T_INTOVF                        0000071F R   D  03
T_INTRPT                        000006D6 R   D  03
T_KRNLSTK                       00000581 R   D  03
T_OPCCUS                        000005C3 R   D  03
T_OPCDEC                        000005A3 R   D  03
T_POWER                         00000598 R   D  03
T_RADRMOD                       000005F2 R   D  03
T_ROPRAND                       000005E1 R   D  03
T_STKVIO                        000006FF R   D  03
T_SUBRNG                        0000079D R   D  03
T_TBIT                          0000063A R   D  03
T_TRANSL                        00000624 R   D  03
T_TRAP                          000006C5 R   D  03
T_XCHFAILURE                    0000069C R   D  03
USTKPTR                         ********     X  00
VAX$MODIFY_EXCEPTION            ********W GX     00
X_BREAK                         0000034D R   D  04
X_CHMS                          00000355 R   D  04
X_CHMU                          0000035D R   D  04
X_COMPAT                        00000365 R   D  04
X_DECOVF                        000003DF R   D  04
X_FLTDIV                        000003CF R   D  04
X_FLTDIV_F                      000003F7 R   D  04
X_FLTOVF                        000003C7 R   D  04
X_FLTOVF_F                      000003EF R   D  04
X_FLTUND                        000003D7 R   D  04
X_FLTUND_F                      000003FF R   D  04
X_INTDIV                        000003BF R   D  04
```

```
X_INTOVF                    000003B7 R  D  04
X_OPCCUS                    0000038F R  D  04
X_OPCDEC                    00000397 R  D  04
X_RADRMOD                   0000039F R  D  04
X_ROPRAND                   000003A7 R  D  04
X_SUBRNG                    000003E7 R  D  04
X_TBIT                      000003AF R  D  04

                                  +----------------+
                                  ! Psect synopsis !
                                  +----------------+

PSECT name                  Allocation          PSECT No.  Attributes
----------                  ----------          ---------  ----------
.  ABS  .                   00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                       FFFFFF00  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
WORK                        00000018  (   24.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
DATA                        000007AD  ( 1965.)  03 (  3.)  NOPIC   USR   CON   REL   LCL   SHR NOEXE   RD  NOWRT NOVEC LONG
CODE                        0000086F  ( 2159.)  04 (  4.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
```

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=0000000B	212 (1)	212 (1)
$$N	=00000001	967 (1)	264 (1) 265 (1) 266 (1) 267 (1)
			268 (1) 269 (1) 270 (1) 271 (1)
			272 (1) 273 (1) 274 (1) 275 (1)
			276 (1) 277 (1) 278 (1) 279 (1)
			300 (1) 301 (1) 302 (1) 303 (1)
			304 (1) 305 (1) 306 (1) 307 (1)
			308 (1) 309 (1) 964 (1) #-967 (1)
$$T1	=00000000	1550 (1)	1550 (1) 212 (1) 423 (1)
$ER	=00000003	1393 (1)	#-1339 (1) #-1393 (1)
$MODULE	00000000-R	246 (1)	1339 (1) 1393 (1)
$SRVCODE_BOOTATTACHED	=00000000	495 (1)	
$SRVCODE_CLEAR_INTERLOCK	=00000009	495 (1)	#-495 (1)
$SRVCODE_GETSYSBUF	=00000004	495 (1)	
$SRVCODE_HALTATTACHED	=00000002	495 (1)	
$SRVCODE_INTERLOCK_TEST	=00000007	495 (1)	
$SRVCODE_PPSCB	=0000000A	495 (1)	
$SRVCODE_PRINTREV	=00000006	495 (1)	
$SRVCODE_RELSYSBUF	=00000005	495 (1)	
$SRVCODE_SET_INTERLOCK	=00000008	495 (1)	#-414 (1)
$SRVCODE_SHOWIDLE	=00000003	495 (1)	
$SRVCODE_STARTATTACHED	=00000001	495 (1)	
AAT_ARITH	0000016B-R	282 (1)	
AAT_COMPAT	000002F0-R	311 (1)	#-747 (1)
APC$_ABORT	=00000006	221 (1)	
APC$_CONTINUE	=00000009	221 (1)	
APC$_DUMP	=00000003	221 (1)	
APC$_NOP	=00000000	221 (1)	
APC$_START	=00000005	221 (1)	
APC$_ZERO	=00000004	221 (1)	
APM$_ABTDON	=00000006	221 (1)	
APM$_DEVERR	=00000002	221 (1)	
APM$_DONE	=0000000A	221 (1)	
APM$_EXCEPT	=0000000B	221 (1)	#-431 (1)
APM$_HARDERR	=00000003	221 (1)	
APM$_MORE	=00000008	221 (1)	
APM$_NOMES	=00000000	221 (1)	
APM$_PREPERR	=0000000C	221 (1)	
APM$_PRGERR	=00000005	221 (1)	
APM$_SOFTERR	=00000004	221 (1)	
APM$_SPOOL	=00000009	221 (1)	
APM$_SYSERR	=00000001	221 (1)	#-472 (1)
APSLMNE	00000000-XR		198 (1) 903 (1)
BIIC$L_BER	=00000008		#-652 (1) #-653 (1)
BIT...	=00000004	227 (1)	211 (1) 213 (1) 219 (1) 220 (1)
			227 (1)
BI_BER_BIT	00000000-XR		191 (1) 656 (1)
BPTMAX	00000000-XR		#-1215 (1) 195 (1)
BUFFER	FFFFFF00	549 (1)	656 (1) 657 (1) 903 (1) 904 (1)
BUFSIZ	=00000100	229 (1)	549 (1) #-656 (1) #-903 (1)

```
CHANDL                          00000811-R   1519   (1)      1458   (1)
CHF$L_MCHARGLST                =00000008                  #-1508   (1)    #-1525   (1)
CHF$L_MCH_FRAME                =00000004                  #-1509   (1)
CHF$L_MCH_SAVR0                =0000000C                  #-1526   (1)
CHF$L_SIGARGLST                =00000004                  #-1521   (1)
CHF$L_SIG_NAME                 =00000004                  #-1523   (1)    #-1526   (1)
CHR$MCHK                        00000000-XR                 197   (1)      609   (1)
CLI$K_BUFSIZ                   =00000100     218   (1)      218   (1)
CLI$K_SIZE                      0000044C     218   (1)
CLI$L_ADDRESS                   00000018     218   (1)
CLI$L_COMMAND                   00000004     218   (1)
CLI$L_DATA                      0000001C     218   (1)
CLI$L_FLAGS                     00000000     218   (1)
CLI$L_FLAGS2                    00000444     218   (1)
CLI$L_LAST                      00000024     218   (1)
CLI$L_NEXT                      00000030     218   (1)
CLI$L_PASS                      0000002C     218   (1)
CLI$L_SUBT                      00000028     218   (1)
CLI$L_TEMP_BASE                 00000448     218   (1)
CLI$L_TEST                      00000020     218   (1)
CLI$M_START_OR_RUN             =00000008     218   (1)
CLI$Q_BUFQWD                    00000034     218   (1)
CLI$Q_FILE                      00000008     218   (1)
CLI$Q_SECTION                   00000010     218   (1)
CLI$Q_TIME                      0000043C     218   (1)
CLI$T_BUFFER                    0000003C     218   (1)
CLI$V_ADAPTER                  =00000018     218   (1)
CLI$V_ADR                      =0000000B     218   (1)
CLI$V_ASCII                    =00000013     218   (1)
CLI$V_BASE_QUAL                =00000002     218   (1)
CLI$V_BREAK                    =0000000A     218   (1)
CLI$V_BRIEF                    =0000001B     218   (1)
CLI$V_BYTE                     =0000000D     218   (1)
CLI$V_CLEAR                    =00000002     218   (1)
CLI$V_DEC                      =00000010     218   (1)
CLI$V_DEFAULT                  =0000000C     218   (1)
CLI$V_DEPOSIT                  =00000019     218   (1)
CLI$V_EVENT                    =00000008     218   (1)
CLI$V_EXAM                     =00000005     218   (1)
CLI$V_FLAGS                    =00000009     218   (1)
CLI$V_HEX                      =00000012     218   (1)
CLI$V_KERNEL                   =00000017     218   (1)
CLI$V_LOAD                     =00000006     218   (1)
CLI$V_LONG                     =0000000F     218   (1)
CLI$V_NEXT                     =00000001     218   (1)
CLI$V_NOALLOC                  =00000000     218   (1)
CLI$V_NOTNUF                   =00000001     218   (1)
CLI$V_OCT                      =00000011     218   (1)
CLI$V_PREG                     =0000001A     218   (1)
CLI$V_QA                       =00000007     218   (1)
CLI$V_QACKLOOPLOOPS            =0000001C     218   (1)
CLI$V_QAERRORPRINTS            =0000001B     218   (1)
CLI$V_QAMULTIPLEPASS           =0000001F     218   (1)
CLI$V_QASUBTESTLOOPS           =0000001E     218   (1)
CLI$V_QATESTLOOPS              =0000001D     218   (1)
CLI$V_REG                      =00000014     218   (1)
CLI$V_REQUIRED                 =00000000     218   (1)
```

```
CLI$V_RUN                       =00000015     218    (1)
CLI$V_SET                       =00000003     218    (1)
CLI$V_SHOW                      =00000004     218    (1)
CLI$V_START_OR_RUN              =00000003     218    (1)       218    (1)
CLI$V_VALSEC                    =00000016     218    (1)
CLI$V_WORD                      =0000000E     218    (1)
CMK$_APBOOT                     =00000004     219    (1)
CMK$_ASTEXIT                    =00000000     219    (1)
CMK$_CANTIM                     =0000000B     219    (1)
CMK$_HIBER                      =00000007     219    (1)
CMK$_INITSCB                    =00000008     219    (1)
CMK$_LAST                       =0000000E     219    (1)
CMK$_MMENABLE                   =00000003     219    (1)
CMK$_RCONSOLE                   =00000001     219    (1)
CMK$_REFRESH                    =00000005     219    (1)
CMK$_SCHDWK                     =00000006     219    (1)
CMK$_SETIMR                     =0000000C     219    (1)
CMK$_SETIPL                     =00000009     219    (1)
CMK$_SETPRT                     =0000000D     219    (1)
CMK$_SGIPR                      =0000000A     219    (1)
CMK$_TCONSOLE                   =00000002     219    (1)
COMMON                           00000166-R   560    (1)    #-554    (1)
COND_DEBUG                       00000000-XR                  197    (1)       249    (1)       250    (1)       251    (1)
                                                              252    (1)
COND_HANDLR                      00000000-R   413    (1)      1390    (1)
COND_HANDLR_CLI                  00000052-R   429    (1)    #-425    (1)
COND_HANDLR_EXIT                 00000126-R   494    (1)    #-427    (1)
CTL$AL_STACK                     00000027-R   254    (1)    #-1481   (1)    #-1486   (1)
CTL$AQ_EXCVEC                    00000007-R   248    (1)      1467   (1)    #-1552   (1)    #-1553   (1)    #-1554   (1)
                                                            #-1555   (1)
CVTREG$_DATA                    =00000008     212    (1)
CVTREG$_MAXLEN                  =00000014     212    (1)
CVTREG$_MNEADR                  =0000000C     212    (1)
CVTREG$_MSB                     =00000004     212    (1)
CVTREG$_NARGS                   =0000000B     212    (1)
CVTREG$_STRBUF                  =00000010     212    (1)
CVTREG$_V1                      =00000018     212    (1)
CVTREG$_V2                      =0000001C     212    (1)
CVTREG$_V3                      =00000020     212    (1)
CVTREG$_V4                      =00000024     212    (1)
CVTREG$_V5                      =00000028     212    (1)
CVTREG$_V6                      =0000002C     212    (1)
DEBUG$GB_FLAGS                   00000000-XR                #-1237   (1)       196    (1)
DIR...                          =FFFFFFFF     549    (1)      549    (1)
DS$AA_BPTADDR                    00000000-XR                #-1216   (1)       196    (1)
DS$ABORT                         00000000-XR                1340   (1)      1394   (1)       195    (1)
DS$CLI                           00000000-XR                 197   (1)       476   (1)
DS$CVTREG                        00000000-XR                 198   (1)       656   (1)       903    (1)
DS$GA_BREAKVEC                   00000000-XR                #-1211   (1)       196    (1)
DS$GA_TBITVEC                    00000000-XR                #-1235   (1)       196    (1)
DS$GB_PRIMARY_CPU_IDENTIFIER     00000000-XR                #-1189   (1)    #-1316   (1)       205    (1)
DS$GB_TYPECODE                   00000000-XR                 202   (1)      #-435   (1)    #-437    (1)    #-562    (1)
                                                           #-649   (1)      #-693   (1)    #-699    (1)    #-876    (1)
                                                           #-886   (1)      #-895   (1)    #-909    (1)    #-914    (1)
DS$GL_FLAGS                      00000000-XR                 199   (1)       469   (1)       470    (1)
DS$GW_TTOUT                      00000000-XR                 202   (1)      #-423   (1)
DS$K_ERROR                      =00000002     213    (1)
```

```
DS$K_NORMAL                    =00000001    213    (1)
DS$K_PRINTB                    =00000002    211    (1)
DS$K_PRINTF                    =00000001    211    (1)
DS$K_PRINTI                    =00000000    211    (1)
DS$K_PRINTX                    =00000003    211    (1)
DS$K_SEVERE                    =00000004    213    (1)
DS$K_SUBSYS                    =00000066    213    (1)      213    (1)
DS$K_TYPE_ABORT_PROGRAM        =00000014    211    (1)
DS$K_TYPE_ABORT_TEST           =00000013    211    (1)
DS$K_TYPE_COMMAND_ERR          =00000015    211    (1)
DS$K_TYPE_COMMAND_OUT          =00000016    211    (1)
DS$K_TYPE_CRD_AUTOTEST         =0000001A    211    (1)
DS$K_TYPE_DS_PROMPT            =00000001    211    (1)
DS$K_TYPE_DS_START             =0000001D    211    (1)
DS$K_TYPE_ERRDEV               =00000008    211    (1)
DS$K_TYPE_ERRHARD              =00000006    211    (1)
DS$K_TYPE_ERROR_BODY           =00000009    211    (1)
DS$K_TYPE_ERROR_END            =0000000A    211    (1)
DS$K_TYPE_ERRPREP              =0000001B    211    (1)
DS$K_TYPE_ERRSOFT              =00000007    211    (1)
DS$K_TYPE_ERRSUP               =00000004    211    (1)
DS$K_TYPE_ERRSYS               =00000005    211    (1)
DS$K_TYPE_ERR_HALT             =0000000D    211    (1)
DS$K_TYPE_EXCEPTION            =0000000C    211    (1)    #-434    (1)    #-698    (1)    #-875    (1)    #-885    (1)
                                                           #-894    (1)
DS$K_TYPE_EXCEPTION_HEAD       =0000000B    211    (1)    #-561    (1)    #-648    (1)    #-692    (1)
DS$K_TYPE_FIRST_PASS           =00000011    211    (1)
DS$K_TYPE_GENERAL              =00000000    211    (1)
DS$K_TYPE_GENERAL_ERROR        =00000003    211    (1)
DS$K_TYPE_NO_TESTS             =00000012    211    (1)
DS$K_TYPE_PARAM_ERROR          =0000001C    211    (1)
DS$K_TYPE_PROGRAM_END          =00000010    211    (1)
DS$K_TYPE_PROGRAM_INFO         =00000017    211    (1)
DS$K_TYPE_PROGRAM_START        =0000000F    211    (1)
DS$K_TYPE_QIO_INVADP           =00000024    211    (1)
DS$K_TYPE_QIO_NODRIVER         =00000022    211    (1)
DS$K_TYPE_QIO_WRONGVER         =00000023    211    (1)
DS$K_TYPE_SCRIPT_ECHO          =00000021    211    (1)
DS$K_TYPE_SCRIPT_PNF           =0000001E    211    (1)
DS$K_TYPE_SCRIPT_PROMPT        =00000020    211    (1)
DS$K_TYPE_SCRIPT_SKIP          =0000001F    211    (1)
DS$K_TYPE_SEQUENCE_ERROR       =00000019    211    (1)
DS$K_TYPE_START_ERR            =00000018    211    (1)
DS$K_TYPE_START_LIST           =00000025    211    (1)
DS$K_TYPE_SUMMARY              =0000000E    211    (1)
DS$K_TYPE_USER_PROMPT          =00000002    211    (1)
DS$K_WARNING                   =00000000    213    (1)
DS$M_ABRTFLG                   =00000040    220    (1)
DS$M_BADTIME                   =00100000    220    (1)
DS$M_BATCH                     =00400000    220    (1)
DS$M_BRKCLR                    =00001000    220    (1)
DS$M_BRKPT                     =00000800    220    (1)
DS$M_CHARFLG                   =00000100    220    (1)
DS$M_CMDFLG                    =00000080    220    (1)
DS$M_CTRLC                     =00000001    220    (1)
DS$M_CTRLO                     =00010000    220    (1)
DS$M_DEVFLG                    =00000200    220    (1)
```

```
DS$M_DISABLCC                =01000000     220    (1)
DS$M_DONFLG                  =00002000     220    (1)
DS$M_ERRFLG                  =00000010     220    (1)
DS$M_EXCEPT                  =00080000     220    (1)
DS$M_EXETST                  =00040000     220    (1)
DS$M_HLTFLG                  =00000008     220    (1)
DS$M_LODFLG                  =00000002     220    (1)
DS$M_MEMMGT                  =00008000     220    (1)
DS$M_NI                      =04000000     220    (1)
DS$M_OUTPUT                  =00800000     220    (1)
DS$M_RUBFLG                  =00000020     220    (1)
DS$M_SCRIPT                  =00200000     220    (1)
DS$M_SETIMR                  =02000000     220    (1)
DS$M_STRFLG                  =00000004     220    (1)
DS$M_SUBT                    =00004000     220    (1)
DS$M_SYSFLG                  =00000400     220    (1)
DS$M_TIMED                   =02000000     220    (1)
DS$M_TIMED_OUT               =08000000     220    (1)
DS$M_TIMRON                  =00020000     220    (1)
DS$SERVICE_DISPATCHER         00000000-XR                  204    (1)      414    (1)      495    (1)
DS$V_ABRTFLG                 =00000006     220    (1)
DS$V_BADTIME                 =00000014     220    (1)
DS$V_BATCH                   =00000016     220    (1)
DS$V_BRKCLR                  =0000000C     220    (1)
DS$V_BRKPT                   =0000000B     220    (1)
DS$V_CHARFLG                 =00000008     220    (1)
DS$V_CMDFLG                  =00000007     220    (1)    #-469    (1)
DS$V_CTRLC                   =00000000     220    (1)
DS$V_CTRLO                   =00000010     220    (1)
DS$V_DEVFLG                  =00000009     220    (1)
DS$V_DISABLCC                =00000018     220    (1)
DS$V_DONFLG                  =0000000D     220    (1)
DS$V_ERRFLG                  =00000004     220    (1)
DS$V_EXCEPT                  =00000013     220    (1)    #-470    (1)
DS$V_EXETST                  =00000012     220    (1)
DS$V_HLTFLG                  =00000003     220    (1)
DS$V_LODFLG                  =00000001     220    (1)
DS$V_MEMMGT                  =0000000F     220    (1)
DS$V_NI                      =0000001A     220    (1)
DS$V_OUTPUT                  =00000017     220    (1)
DS$V_RUBFLG                  =00000005     220    (1)
DS$V_SCRIPT                  =00000015     220    (1)
DS$V_SETIMR                  =00000019     220    (1)
DS$V_STRFLG                  =00000002     220    (1)
DS$V_SUBT                    =0000000E     220    (1)
DS$V_SYSFLG                  =0000000A     220    (1)
DS$V_TIMED                   =00000019     220    (1)
DS$V_TIMED_OUT               =0000001B     220    (1)
DS$V_TIMRON                  =00000011     220    (1)
DS$_AP_NORMAL_BREAK          =00660150     213    (1)
DS$_ARITH                    =006600D0     213    (1)    #-1172   (1)    #-669    (1)
DS$_ASBE                     =00660118     213    (1)
DS$_BADLINK                  =006600F0     213    (1)
DS$_BADTYPE                  =006600E8     213    (1)
DS$_BIIC                     =00660120     213    (1)
DS$_CHME                     =006600A8     213    (1)    #-1269   (1)    #-622    (1)
DS$_CHMK                     =006600E0     213    (1)    #-1261   (1)    #-627    (1)
```

```
DS$_DEVNAME                     =00660108      213     (1)
DS$_ERROR                       =00660002      213     (1)
DS$_FHWE                        =00660068      213     (1)
DS$_FRAGBUF                     =00660080      213     (1)
DS$_GK_EXE_MCHK                 =00000000      227     (1)     #-1074    (1)
DS$_GK_INITSCB                  =00000002      227     (1)
DS$_GK_INT_WORK                 =00000003      227     (1)
DS$_GK_TST$MCHK                 =00000001      227     (1)     #-1018    (1)
DS$_ICBUSY                      =006600C8      213     (1)
DS$_ICERR                       =006600C0      213     (1)
DS$_IHWE                        =00660060      213     (1)
DS$_ILLCHAR                     =00660018      213     (1)
DS$_ILLPAGCNT                   =00660078      213     (1)
DS$_ILLUNIT                     =00660100      213     (1)
DS$_INIT_FAIL                   =00660140      213     (1)
DS$_INSFMEM                     =00660050      213     (1)
DS$_INVCPU                      =00660130      213     (1)
DS$_IPL2HI                      =006600B8      213     (1)
DS$_IVADDR                      =00660040      213     (1)
DS$_IVVECT                      =00660038      213     (1)
DS$_KRNLSTK                     =00660090      213     (1)     #-1085    (1)     #-612     (1)
DS$_LOGIC                       =00660070      213     (1)
DS$_MCHK                        =00660088      213     (1)     #-1070    (1)     #-605     (1)
DS$_MEM_ALLOC_ERR               =00660138      213     (1)
DS$_MMOFF                       =00660058      213     (1)
DS$_NEEDUNIT                    =006600F8      213     (1)
DS$_NODE                        =00660128      213     (1)
DS$_NOPCS                       =00660110      213     (1)
DS$_NORMAL                      =00660001      213     (1)
DS$_NOSUPPORT                   =006600B1      213     (1)
DS$_NOTALLOWMP                  =00660148      213     (1)
DS$_NOTDON                      =00660030      213     (1)
DS$_NOTIMP                      =006600B0      213     (1)       213     (1)
DS$_NULLSTR                     =00660010      213     (1)
DS$_OVERFLOW                    =00660008      213     (1)
DS$_POWER                       =00660098      213     (1)     #-1094    (1)
DS$_PROGERR                     =00660020      213     (1)
DS$_SEVERE                      =00660004      213     (1)
DS$_TRANSL                      =006600A0      213     (1)     #-1153    (1)     #-617     (1)
DS$_TRUNCATE                    =00660028      213     (1)
DS$_UNEXPINT                    =006600D8      213     (1)     #-1252    (1)     #-632     (1)
DS$_UNSUPPDEV                   =00660158      213     (1)
DS$_VASFULL                     =00660048      213     (1)
DS$_WARNING                     =00660000      213     (1)       213     (1)
DSA$GL_FLAGS                     0000FE00                        1549    (1)       430     (1)       450     (1)       635     (1)
DSA$GL_MSGTYP                    0000FE40                      #-432     (1)     #-473     (1)
DSA$GL_SID                       0000FE14                      #-637     (1)
DSA$V_USER                      =0000001C                      #-1548    (1)     #-430     (1)     #-450     (1)     #-635     (1)
DSR$FAULT_CLEAR                  00000000-XR                     1019    (1)       1075    (1)       200     (1)
DSR$FIND_USER_PC                 00000000-XR                     203     (1)       960     (1)
DSX$PRINTB                       00000000-XR                     199     (1)       552     (1)
DSX$PRINTI                       00000000-XR                     195     (1)       557     (1)       964     (1)       967     (1)
DSX$PRINTSIG                     00000148-R      551     (1)
DSX$PRINTSIGI                    0000015A-R      556     (1)     424     (1)
DSX$PRINTX                       00000000-XR                     200     (1)       553     (1)
DSX$SETPRIEXV                    00000832-R     1546     (1)
DS_ERRSUP                        00000000-XR                     1339    (1)       1393    (1)       195     (1)
```

```
ESTKPTR                         00000000-XR                      201    (1)      255    (1)
EXCPT                           00000132-R    499    (1)         491    (1)
EXE$ACVIOLAT                    00000554-R   1141    (1)
EXE$BREAK                       0000057C-R   1182    (1)
EXE$GB_CPUTYPE                  00000000-XR                    #-1367   (1)      203    (1)
EXE$REFLECT                     00000667-R   1304    (1)
EXE$ROPRAND                     00000544-R   1122    (1)
EXE$TBIT                        00000610-R   1234    (1)
EXE_3ARG                        00000663-R   1294    (1)    #-1086   (1)    #-1096   (1)    #-1105   (1)    #-1114   (1)
                                                           #-1124   (1)    #-1133   (1)    #-1224   (1)    #-1246   (1)
EXE_4ARG                        0000065F-R   1290    (1)    #-1164   (1)    #-1173   (1)    #-1253   (1)    #-1262   (1)
                                                           #-1270   (1)    #-1279   (1)    #-1288   (1)
EXE_ACCVIO                      00000554-R   1140    (1)
EXE_ARITH                       00000570-R   1171    (1)
EXE_BREAK                       0000057C-R   1181    (1)
EXE_CHME                        00000648-R   1268    (1)
EXE_CHMK                        00000640-R   1260    (1)
EXE_CHMS                        00000650-R   1277    (1)
EXE_CHMU                        00000658-R   1286    (1)
EXE_COMPAT                      00000568-R   1162    (1)    #-1244   (1)
EXE_EXCEPTION                   00000667-R   1305    (1)    #-1077   (1)    #-1144   (1)    #-1155   (1)    #-1292   (1)
                                                           #-1296   (1)
EXE_KRNLSTK                     0000051C-R   1084    (1)
EXE_MCHK                        000004FC-R   1069    (1)
EXE_OPCCUS                      0000053C-R   1112    (1)
EXE_OPCDEC                      00000534-R   1103    (1)
EXE_POWER                       00000528-R   1093    (1)
EXE_RADRMOD                     0000054C-R   1131    (1)
EXE_ROPRAND                     00000544-R   1121    (1)
EXE_TBIT                        00000610-R   1233    (1)
EXE_TRANSL                      0000055C-R   1151    (1)
EXE_UNEXPINT                    00000636-R   1251    (1)
FIND_USERPC                     000004A0-R    953    (1)      436    (1)
FMT_BER                         00000000-XR                      191    (1)      660    (1)
FMT_BI_UNEXPINT                 0000041F-R    332    (1)      646    (1)
FMT_CHMX_ARG                    00000523-R    339    (1)      889    (1)
FMT_ERRPC                       00000465-R    333    (1)      899    (1)
FMT_ERRPSL                      000004B9-R    336    (1)      907    (1)
FMT_EXCEPT                      000003BD-R    330    (1)      265    (1)      266    (1)      267    (1)      268    (1)
                                                              269    (1)      270    (1)      271    (1)      272    (1)
                                                              273    (1)      274    (1)      275    (1)      276    (1)
                                                              277    (1)      278    (1)      279    (1)      300    (1)
                                                              301    (1)      302    (1)      303    (1)      304    (1)
                                                              305    (1)      306    (1)      307    (1)      308    (1)
                                                              309    (1)
FMT_EXCEP_CODE                  0000050C-R    338    (1)      750    (1)
FMT_PXXX                        0000056D-R    342    (1)      710    (1)
FMT_UMCHK                       00000383-R    328    (1)      264    (1)
FMT_UNEXPINT                    000003E7-R    331    (1)      665    (1)
FMT_UNK_ARG                     00000536-R    340    (1)      695    (1)
FMT_UNK_CNT                     00000558-R    341    (1)      701    (1)
FMT_USER_PC_NF                  0000049A-R    335    (1)      967    (1)
FMT_USRPC                       0000047E-R    334    (1)      964    (1)
FMT_VIRT                        000004DA-R    337    (1)      880    (1)
HANDLER                        =00000000      1448    (1)
ID_TEMP                         00000014-R    241    (1)
IO$M_CANCTRLO                   00000000-XR                      201    (1)    #-423    (1)
```

```
IO$_WRITEVBLK                  00000000-XR                  202    (1)    #-423    (1)
IS                            =00000001     228    (1)
ISTKPTR                        00000000-XR                  201    (1)      255    (1)
LOCAL                          FFFFFF00     549    (1)     564    (1)
MMG$PAGEFAULT                  0000055C-R   1152   (1)
MODELST                        00000000-XR                  198    (1)      903    (1)
MP$GL_AP_SIGNAL_ARRAY          00000000-XR                  192    (1)    #-418    (1)
MP$GL_SAVED_PSL                00000000-XR                #-1198   (1)      193    (1)
MP$GR_BOOTED_PROCESSORS        00000000-XR                #-1184   (1)    #-1311   (1)      205    (1)    #-416    (1)
                                                          #-478    (1)
MP$GR_SAVED_GPRS               00000000-XR                #-1199   (1)    #-1200   (1)    #-1201   (1)    #-1202   (1)
                                                          #-1203   (1)    #-1204   (1)    #-1205   (1)    #-1206   (1)
                                                          #-1207   (1)      193    (1)
MP$R_ACTIVE_ROUTINES           00000000-XR                  189    (1)
MP$R_GET_PROCESSOR_ID          00000000-XR                  190    (1)
MP$R_ROUTINE_COUNT_TABLE       00000000-XR                  189    (1)
MP$R_ROUTINE_ID_TABLE          00000000-XR                  190    (1)
MP$V_IL_ALLN                  =00000004     222    (1)
MP$V_IL_CNDB                  =00000007     222    (1)
MP$V_IL_CNHN                  =00000006     222    (1)    #-414    (1)    #-495    (1)
MP$V_IL_DELN                  =00000005     222    (1)
MP$V_IL_DSBK                  =00000008     222    (1)
MP$V_IL_ERSP                  =00000003     222    (1)
MP$V_IL_GTLN                  =00000009     222    (1)
MP$V_IL_NUMT                  =00000000     222    (1)
MP$V_IL_PRNT                  =00000001     222    (1)
MP$V_IL_RRPT                  =00000002     222    (1)
MP$_COND_HANDLR                00000000-XR                  192    (1)      419    (1)
MP$_GET_PROCESSOR_ID           00000000-XR                  1188   (1)      1315   (1)      206    (1)
MP$_RESUME_ATTACHED_PROCESSORS 00000000-XR                  206    (1)      480    (1)
MSG_ACCVIO                     000000B7-R   271    (1)     597    (1)
MSG_BREAK                      000000F3-R   274    (1)     721    (1)
MSG_CHME                       0000012F-R   277    (1)     624    (1)
MSG_CHMK                       0000011B-R   276    (1)     629    (1)
MSG_CHMS                       00000143-R   278    (1)     729    (1)
MSG_CHMU                       00000157-R   279    (1)     737    (1)
MSG_COMPAT                     00000107-R   275    (1)     745    (1)
MSG_DECOVF                     0000028C-R   305    (1)     838    (1)
MSG_FLTDIV                     00000264-R   303    (1)     822    (1)
MSG_FLTDIV_F                   000002C8-R   308    (1)     862    (1)
MSG_FLTOVF                     00000250-R   302    (1)     814    (1)
MSG_FLTOVF_F                   000002B4-R   307    (1)     854    (1)
MSG_FLTUND                     00000278-R   304    (1)     830    (1)
MSG_FLTUND_F                   000002DC-R   309    (1)     871    (1)
MSG_INTDIV                     0000023C-R   301    (1)     806    (1)
MSG_INTOVF                     00000228-R   300    (1)     799    (1)
MSG_KRNLSTK                    0000003F-R   265    (1)     614    (1)
MSG_OPCCUS                     0000007B-R   268    (1)     759    (1)
MSG_OPCDEC                     00000067-R   267    (1)     767    (1)
MSG_POWER                      00000053-R   266    (1)
MSG_RADRMOD                    000000A3-R   270    (1)     775    (1)
MSG_ROPRAND                    0000008F-R   269    (1)     783    (1)
MSG_SUBRNG                     000002A0-R   306    (1)     846    (1)
MSG_TBIT                       000000DF-R   273    (1)     791    (1)
MSG_TRANSL                     000000CB-R   272    (1)     619    (1)
MSG_UMCHK                      00000037-R   264    (1)     602    (1)
NEW_AP_FP                      00000004-R   237    (1)    #-486    (1)    #-499    (1)
```

```
NEW_PC_PSL                  0000000C-R   239    (1)    #-488    (1)    #-501    (1)
NEW_SP                      00000000-R   235    (1)    #-487    (1)    #-500    (1)
NORMAL                      0000070A-R   1363   (1)    #-1319   (1)    #-1323   (1)    #-1326   (1)      1354   (1)
PCB_BASE                    00000000-XR                #-1332   (1)    #-1342   (1)      199    (1)    #-458    (1)
PR$_SID_TYP8SS             =00000005                   #-637    (1)
PR$_SID_TYPUV2             =00000008                   #-1367   (1)
PRINT_CHMX_ARG              0000041E-R   884    (1)    #-630    (1)
PRINT_PC_PSL                00000433-R   893    (1)      567    (1)      671    (1)    #-714    (1)    #-891    (1)
PRINT_TYPE_VA               00000407-R   874    (1)    #-598    (1)    #-620    (1)
PRINT_UNK_EXC               0000030D-R   691    (1)    #-684    (1)
PSL$C_USER                 =00000003                   #-1484   (1)
PSL$M_CM                   =80000000                   #-1352   (1)
PSL$M_FPD                  =08000000                   #-1353   (1)
PSL$M_TBIT                 =00000010                   #-1352   (1)
PSL$M_TP                   =40000000                   #-1353   (1)
PSL$S_CURMOD               =00000002                   #-1463   (1)    #-452    (1)    #-455    (1)
PSL$S_PRVMOD               =00000002                   #-1325   (1)
PSL$V_CM                   =0000001F                   #-1241   (1)
PSL$V_CURMOD               =00000018                   #-1463   (1)    #-452    (1)    #-455    (1)
PSL$V_IS                   =0000001A     219    (1)    #-1323   (1)    #-449    (1)
PSL$V_PRVMOD               =00000016                   #-1324   (1)
RESIGNAL                    00000494-R   912    (1)      593    (1)
SAVABS...                  =FFFFFF00     549    (1)
SAVAP                      =00000008     1451   (1)
SAVFP                      =0000000C     1452   (1)    #-1477   (1)
SAVMSK                     =00000006     1450   (1)      1498   (1)      1499   (1)
SAVPC                      =00000010     1453   (1)    #-1496   (1)
SAVPSW                     =00000004     1449   (1)
SAVRG                      =00000014     1454   (1)    #-1500   (1)
SCB_IMAGE                   00000000-XR                  203    (1)
SCRIPT$STOP                 00000000-XR                  200    (1)      474    (1)
SEARCH                      0000076B-R   1456   (1)      1376   (1)
SIZ...                     =00000001     220    (1)      220    (1)
SS$_ACCVIO                 =0000000C                   #-1142   (1)    #-595    (1)
SS$_BREAK                  =00000414                   #-1223   (1)    #-593    (1)
SS$_CMODSUPR               =0000041C                   #-1278   (1)
SS$_CMODUSER               =00000424                   #-1287   (1)
SS$_COMPAT                 =0000042C                   #-1163   (1)
SS$_CONTINUE               =00000001                   #-492    (1)
SS$_MCHECK                 =000002BC                   #-600    (1)
SS$_NOHANDLER              =000008F8                   #-1512   (1)
SS$_OPCCUS                 =00000434                   #-1113   (1)
SS$_OPCDEC                 =0000043C                   #-1104   (1)
SS$_RADRMOD                =0000044C                   #-1132   (1)
SS$_RESIGNAL               =00000918                   #-913    (1)
SS$_ROPRAND                =00000454                   #-1123   (1)
SS$_TBIT                   =00000464                   #-1245   (1)
SS$_UNWIND                 =00000920                   #-1522   (1)
SSTKPTR                     00000000-XR                  201    (1)      255    (1)
STACK_BASE                  00000000-XR                  1483   (1)      199    (1)
SYS$CALL_HANDL              00000000-XR                  1382   (1)    #-1496   (1)      200    (1)
SYS$QIO                     00000000-XR                  423    (1)
SYS$SETEXV                  00000000-XR                  1550   (1)
SYS$UNWIND                  00000000-XR                  1528   (1)      197    (1)
TST$MCHK                    000004E4-R   1016   (1)
T_ABORT                     000006D0-R   361    (1)      265    (1)
T_ACCVIO                    0000060B-R   350    (1)      271    (1)
```

```
T_BREAK                         00000640-R   353   (1)      274   (1)
T_CHME                          0000065E-R   355   (1)      277   (1)
T_CHMK                          0000064B-R   354   (1)      276   (1)
T_CHMS                          00000674-R   356   (1)      278   (1)
T_CHMU                          0000068B-R   357   (1)      279   (1)
T_COMPAT                        000006EC-R   364   (1)      275   (1)      749   (1)
T_DECOVF                        0000078C-R   371   (1)      305   (1)
T_FAULT                         000006CA-R   360   (1)      267   (1)      268   (1)      270   (1)      271   (1)
                                                            272   (1)      274   (1)      307   (1)      308   (1)
                                                            309   (1)
T_FLTDIV                        00000759-R   369   (1)      303   (1)      308   (1)
T_FLTOVF                        00000747-R   368   (1)      302   (1)      307   (1)
T_FLTUND                        00000779-R   370   (1)      304   (1)      309   (1)
T_FLT_ABO                       000006E0-R   363   (1)      269   (1)      275   (1)      748   (1)
T_INTDIV                        00000730-R   367   (1)      301   (1)
T_INTOVF                        0000071F-R   366   (1)      300   (1)
T_INTRPT                        000006D6-R   362   (1)      266   (1)
T_KRNLSTK                       00000581-R   344   (1)      265   (1)
T_OPCCUS                        000005C3-R   347   (1)      268   (1)
T_OPCDEC                        000005A3-R   346   (1)      267   (1)
T_POWER                         00000598-R   345   (1)      266   (1)
T_RADRMOD                       000005F2-R   349   (1)      270   (1)
T_ROPRAND                       000005E1-R   348   (1)      269   (1)
T_STKVIO                        000006FF-R   365   (1)      1339  (1)
T_SUBRNG                        0000079D-R   372   (1)      306   (1)
T_TBIT                          0000063A-R   352   (1)      273   (1)
T_TRANSL                        00000624-R   351   (1)      272   (1)
T_TRAP                          000006C5-R   359   (1)      273   (1)      276   (1)      277   (1)      278   (1)
                                                            279   (1)      300   (1)      301   (1)      302   (1)
                                                            303   (1)      304   (1)      305   (1)      306   (1)
T_XCHFAILURE                    0000069C-R   358   (1)      1393  (1)
USTKPTR                         00000000-XR                 198   (1)      255   (1)
VAX$MODIFY_EXCEPTION            00000000-XR                 1369  (1)      191   (1)      203   (1)
X_BREAK                         0000034D-R   720   (1)      593   (1)
X_CHMS                          00000355-R   728   (1)      593   (1)
X_CHMU                          0000035D-R   736   (1)      593   (1)
X_COMPAT                        00000365-R   744   (1)      593   (1)
X_DECOVF                        000003DF-R   837   (1)      593   (1)      683   (1)
X_FLTDIV                        000003CF-R   821   (1)      593   (1)      683   (1)
X_FLTDIV_F                      000003F7-R   861   (1)      593   (1)      683   (1)
X_FLTOVF                        000003C7-R   813   (1)      593   (1)      683   (1)
X_FLTOVF_F                      000003EF-R   853   (1)      593   (1)      683   (1)
X_FLTUND                        000003D7-R   829   (1)      593   (1)      683   (1)
X_FLTUND_F                      000003FF-R   870   (1)      593   (1)      683   (1)
X_INTDIV                        000003BF-R   805   (1)      593   (1)      683   (1)
X_INTOVF                        000003B7-R   798   (1)      593   (1)      683   (1)
X_OPCCUS                        0000038F-R   758   (1)      593   (1)
X_OPCDEC                        00000397-R   766   (1)      593   (1)
X_RADRMOD                       0000039F-R   774   (1)      593   (1)
X_ROPRAND                       000003A7-R   782   (1)      593   (1)
X_SUBRNG                        000003E7-R   845   (1)      593   (1)      683   (1)
X_TBIT                          000003AF-R   790   (1)      593   (1)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO                       SIZE        DEFINITION      REFERENCES...
-----                       ----        ----------      -------------
$BIICDEF                    8           226    (1)      226    (1)
$CHFDEF                     1           224    (1)      224    (1)
$D1_ABPROGRAM               1           1340   (1)      1340   (1)      1394   (1)
$D1_PRINT_L                 2           264    (1)      264    (1)      265    (1)      266    (1)      267    (1)
                                                        268    (1)      269    (1)      270    (1)      271    (1)
                                                        272    (1)      273    (1)      274    (1)      275    (1)
                                                        276    (1)      277    (1)      278    (1)      279    (1)
                                                        300    (1)      301    (1)      302    (1)      303    (1)
                                                        304    (1)      305    (1)      306    (1)      307    (1)
                                                        308    (1)      309    (1)
$D1_PRINT_S                 1           964    (1)      964    (1)      967    (1)
$DEF                        1           227    (1)
$DEFINI                     1           214    (1)      214    (1)      215    (1)      216    (1)      217    (1)
                                                        223    (1)      224    (1)      226    (1)
$DS_ABORT                   1           1340   (1)      1340   (1)      1394   (1)
$DS_CVTREG_DEF              1           212    (1)      212    (1)
$DS_CVTREG_S                1           654    (1)      654    (1)      901    (1)
$DS_DSADEF                  5           223    (1)      223    (1)
$DS_DSDEF                   3           213    (1)      213    (1)
$DS_SCBDEF                  3           216    (1)      216    (1)
$DS_SRVCODE_DEF             2           495    (1)      414    (1)      495    (1)
$DS_TYPEDEF                 4           211    (1)      211    (1)
$EQU                        1           227    (1)      211    (1)      213    (1)      219    (1)      227    (1)
$EQULS1                     1           227    (1)      211    (1)      213    (1)      219    (1)      227    (1)
$EQULST                     1           211    (1)      211    (1)      213    (1)      219    (1)      227    (1)
$GBLINI                     2           211    (1)      211    (1)      213    (1)      219    (1)      220    (1)
                                                        227    (1)
$MP_CLEAR_INTERLOCK         1           495    (1)      495    (1)
$MP_ILDEFS                  1           222    (1)      222    (1)
$MP_SET_INTERLOCK           1           414    (1)      414    (1)
$OFFDEF                     1           212    (1)      212    (1)
$OFFSET                     2           547    (1)      547    (1)
$OFFST1                     1           549    (1)      549    (1)
$PRDEF                      5           214    (1)      214    (1)
$PRINTI_L                   1           264    (1)      264    (1)      265    (1)      266    (1)      267    (1)
                                                        268    (1)      269    (1)      270    (1)      271    (1)
                                                        272    (1)      273    (1)      274    (1)      275    (1)
                                                        276    (1)      277    (1)      278    (1)      279    (1)
                                                        300    (1)      301    (1)      302    (1)      303    (1)
                                                        304    (1)      305    (1)      306    (1)      307    (1)
                                                        308    (1)      309    (1)
$PRINTI_S                   1           964    (1)      964    (1)      967    (1)
$PSLDEF                     2           215    (1)      215    (1)
$PUSHADR                    1           423    (1)      1339   (1)      1393   (1)      1550   (1)      423    (1)
                                                        656    (1)      903    (1)
$PUSHTWO                    1           423    (1)      423    (1)
$QIOPUSH                    1           423    (1)      1550   (1)      423    (1)
$QIO_S                      1           422    (1)      422    (1)
$SETEXV_S                   1           1550   (1)      1550   (1)
$SSDEF                      1           217    (1)      217    (1)      225    (1)
```

$VIELD	1	220	(1)	220	(1)			
$VIELD1	1	227	(1)	220	(1)			
APTDEF	1	221	(1)	221	(1)			
BR_IF_NOT_USER	1	635	(1)	635	(1)			
BR_IF_USER	1	430	(1)	430	(1)	450	(1)	
CASE	1	569	(1)	569	(1)	672	(1)	
CLIDEF	4	218	(1)	218	(1)			
CMKDEF	1	219	(1)	219	(1)			
DSFDEF	3	220	(1)	220	(1)			
ERRSUP_S	1	1339	(1)	1339	(1)	1393	(1)	
FLTCLR_SEL	1	227	(1)	227	(1)			
MODNAM	1	246	(1)	246	(1)			
SET_CMDFLG	1	469	(1)	469	(1)			
SET_EXCEPT	1	470	(1)	470	(1)			

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	1021	(1)	1022	(1)	1500	(1)	1501	(1)	1504	(1)				
ADDL2	00C0	1020	(1)	1072	(1)	1076	(1)	1191	(1)	1210	(1)	1318	(1)	1321	(1)
		1403	(1)												
ADDL3	00C1	1330	(1)	1400	(1)										
AOBLSS	00F2	712	(1)												
ASHL	0078	1334	(1)	1506	(1)	650	(1)								
BBC	00E1	1548	(1)	635	(1)										
BBS	00E0	1241	(1)	1323	(1)	430	(1)	449	(1)	450	(1)				
BBSS	00E2	469	(1)	470	(1)										
BEQL	0013	1185	(1)	1190	(1)	1212	(1)	1217	(1)	1236	(1)	1312	(1)	1317	(1)
		1326	(1)	1465	(1)	1469	(1)	1478	(1)	1485	(1)	1491	(1)	1524	(1)
		417	(1)	457	(1)	479	(1)	633	(1)	962	(1)				
BGTR	0014	1466	(1)	642	(1)										
BGTRU	001A	1482	(1)												
BICL	00CA	1352	(1)												
BICL3	00CB	1211	(1)	1235	(1)										
BISL	00C8	1023	(1)	1493	(1)										
BISL2	00C8	651	(1)												
BLBC	00E9	1377	(1)	1383	(1)	1503	(1)								
BLBS	00E8	1237	(1)	1391	(1)	1476	(1)	425	(1)						
BLEQ	0015	705	(1)												
BLSS	0019	640	(1)												
BLSSU	001F	1489	(1)	959	(1)										
BNEQ	0012	1337	(1)	1368	(1)	1473	(1)	1497	(1)	1507	(1)	596	(1)	601	(1)
		606	(1)	613	(1)	618	(1)	623	(1)	628	(1)	638	(1)	670	(1)
BRB	0011	1208	(1)	1224	(1)	1243	(1)	1246	(1)	1262	(1)	1270	(1)	1279	(1)
		1288	(1)	1292	(1)	1296	(1)	1384	(1)	1474	(1)	1551	(1)	554	(1)
		625	(1)	684	(1)	891	(1)								
BRW	0031	1077	(1)	1086	(1)	1096	(1)	1105	(1)	1114	(1)	1124	(1)	1133	(1)
		1144	(1)	1155	(1)	1164	(1)	1173	(1)	1244	(1)	1253	(1)	1319	(1)
		1510	(1)	427	(1)	598	(1)	620	(1)	630	(1)	634	(1)	636	(1)
		714	(1)												
CALLG	00FA	1340	(1)	1376	(1)	1390	(1)	1394	(1)	424	(1)	436	(1)	476	(1)
		597	(1)	602	(1)	614	(1)	619	(1)	624	(1)	629	(1)	721	(1)
		729	(1)	737	(1)	745	(1)	759	(1)	767	(1)	775	(1)	783	(1)
		791	(1)	799	(1)	806	(1)	814	(1)	822	(1)	830	(1)	838	(1)
		846	(1)	854	(1)	862	(1)	871	(1)						
CALLS	00FB	1188	(1)	1315	(1)	1339	(1)	1393	(1)	1528	(1)	1550	(1)	414	(1)
		423	(1)	480	(1)	495	(1)	609	(1)	647	(1)	656	(1)	661	(1)
		666	(1)	696	(1)	702	(1)	711	(1)	751	(1)	881	(1)	890	(1)
		900	(1)	903	(1)	908	(1)	960	(1)	964	(1)	967	(1)		
CASEL	00CF	593	(1)												
CASEW	00AF	683	(1)												
CLRB	0094	437	(1)	909	(1)	914	(1)								
CLRQ	007C	1527	(1)	423	(1)										
CMPB	0091	1189	(1)	1316	(1)	1468	(1)	637	(1)						
CMPL	00D1	1216	(1)	1367	(1)	1481	(1)	1484	(1)	1488	(1)	1496	(1)	600	(1)
		605	(1)	612	(1)	617	(1)	622	(1)	627	(1)	632	(1)	639	(1)
		641	(1)	669	(1)	958	(1)	961	(1)						
CMPW	00B1	1522	(1)	595	(1)										

CMPZV	00ED	455	(1)												
DIVL3	00C7	1071	(1)	565	(1)										
EXTZV	00EF	1324	(1)	1463	(1)	1498	(1)	1499	(1)	452	(1)	643	(1)	746	(1)
HALT	0000	1095	(1)												
INCL	00D6	1464	(1)												
JSB	0016	1019	(1)	1075	(1)	1369	(1)	1382	(1)	419	(1)	474	(1)		
MFPR	00DB	1333	(1)	459	(1)										
MNEGL	00CE	1307	(1)												
MOVAB	009E	1344	(1)	1354	(1)	552	(1)	553	(1)	557	(1)	558	(1)	564	(1)
		657	(1)	904	(1)										
MOVAL	00DE	1458	(1)	418	(1)	445	(1)	447	(1)	448	(1)	466	(1)	491	(1)
		955	(1)												
MOVAQ	007E	1467	(1)												
MOVB	0090	434	(1)	561	(1)	648	(1)	692	(1)	698	(1)	875	(1)	885	(1)
		894	(1)												
MOVL	00D0	1206	(1)	1207	(1)	1215	(1)	1240	(1)	1329	(1)	1332	(1)	1342	(1)
		1345	(1)	1356	(1)	1460	(1)	1472	(1)	1477	(1)	1479	(1)	1486	(1)
		1490	(1)	1508	(1)	1509	(1)	1521	(1)	1525	(1)	1526	(1)	1552	(1)
		1553	(1)	1554	(1)	1555	(1)	431	(1)	446	(1)	458	(1)	472	(1)
		487	(1)	489	(1)	492	(1)	500	(1)	652	(1)	653	(1)	706	(1)
		897	(1)	957	(1)										
MOVPSL	00DC	1198	(1)	1322	(1)	1462	(1)	454	(1)	490	(1)				
MOVQ	007D	1199	(1)	1200	(1)	1201	(1)	1202	(1)	1203	(1)	1204	(1)	1205	(1)
		1402	(1)	461	(1)	463	(1)	465	(1)	467	(1)	482	(1)	483	(1)
		486	(1)	488	(1)	499	(1)	501	(1)						
MOVZBL	009A	956	(1)												
MOVZWL	003C	1104	(1)	1113	(1)	1123	(1)	1132	(1)	1142	(1)	1163	(1)	1223	(1)
		1245	(1)	1278	(1)	1287	(1)	1512	(1)	423	(1)	913	(1)		
MTPR	00DA	1343	(1)												
MULL2	00C4	1401	(1)												
POPR	00BA	1219	(1)	1222	(1)	1355	(1)	484	(1)						
PROBEW	000D	1336	(1)												
PUSHAB	009F	1483	(1)	567	(1)	646	(1)	656	(1)	660	(1)	665	(1)	671	(1)
		695	(1)	701	(1)	710	(1)	748	(1)	749	(1)	750	(1)	880	(1)
		889	(1)	899	(1)	903	(1)	907	(1)	964	(1)	967	(1)		
PUSHAL	00DF	1339	(1)	1364	(1)	1365	(1)	1393	(1)	1550	(1)	903	(1)		
PUSHL	00DD	1017	(1)	1018	(1)	1070	(1)	1073	(1)	1074	(1)	1085	(1)	1094	(1)
		1143	(1)	1153	(1)	1154	(1)	1172	(1)	1187	(1)	1252	(1)	1261	(1)
		1269	(1)	1291	(1)	1295	(1)	1308	(1)	1309	(1)	1314	(1)	1339	(1)
		1366	(1)	1393	(1)	1550	(1)	414	(1)	423	(1)	495	(1)	608	(1)
		644	(1)	645	(1)	656	(1)	658	(1)	659	(1)	664	(1)	700	(1)
		708	(1)	709	(1)	747	(1)	878	(1)	879	(1)	888	(1)	898	(1)
		903	(1)	905	(1)	906	(1)	964	(1)						
PUSHR	00BB	1214	(1)	1306	(1)	1328	(1)	462	(1)	464	(1)	607	(1)		
REI	0002	1024	(1)	1357	(1)	1404	(1)	502	(1)						
RET	0004	1494	(1)	1513	(1)	1530	(1)	1556	(1)	497	(1)	910	(1)	915	(1)
		965	(1)	968	(1)										
RSB	0005	1220	(1)	1238	(1)	603	(1)	610	(1)	615	(1)	662	(1)	667	(1)
		722	(1)	730	(1)	738	(1)	752	(1)	760	(1)	768	(1)	776	(1)
		784	(1)	792	(1)	800	(1)	807	(1)	815	(1)	823	(1)	831	(1)
		839	(1)	847	(1)	855	(1)	864	(1)	872	(1)	882	(1)		
SOBGEQ	00F4	1218	(1)												
SOBGTR	00F5	1346	(1)												
SUBL2	00C2	1186	(1)	1313	(1)	1335	(1)								
SUBL3	00C3	704	(1)												
TSTL	00D5	1184	(1)	1311	(1)	1470	(1)	416	(1)	478	(1)				

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	249	(1)	250	(1)	251	(1)	252	(1)	255	(1)	282	(1)	283	(1)	284	(1)
	285	(1)	286	(1)	287	(1)	288	(1)	289	(1)	311	(1)	312	(1)	313	(1)
	314	(1)	315	(1)	316	(1)	317	(1)	318	(1)						
.ALIGN	1015	(1)	1068	(1)	1083	(1)	1092	(1)	1102	(1)	1111	(1)	1120	(1)	1130	(1)
	1139	(1)	1150	(1)	1161	(1)	1170	(1)	1180	(1)	1232	(1)	1259	(1)	1267	(1)
	1276	(1)	1285	(1)												
.ASCIC	246	(1)	291	(1)	292	(1)	293	(1)	294	(1)	295	(1)	296	(1)	297	(1)
	298	(1)	319	(1)	320	(1)	321	(1)	322	(1)	323	(1)	324	(1)	325	(1)
	326	(1)	328	(1)	330	(1)	331	(1)	332	(1)	333	(1)	334	(1)	335	(1)
	336	(1)	337	(1)	338	(1)	339	(1)	340	(1)	341	(1)	342	(1)	344	(1)
	345	(1)	346	(1)	347	(1)	348	(1)	349	(1)	350	(1)	351	(1)	352	(1)
	353	(1)	354	(1)	355	(1)	356	(1)	357	(1)	358	(1)	359	(1)	360	(1)
	361	(1)	362	(1)	363	(1)	364	(1)	365	(1)	366	(1)	367	(1)	368	(1)
	369	(1)	370	(1)	371	(1)	372	(1)								
.BLKB	549	(1)														
.BLKL	218	(1)	235	(1)												
.BLKQ	237	(1)	239	(1)												
.DSABL	8	(1)														
.END	1558	(1)														
.ENDC	1339	(1)	1393	(1)	1550	(1)	211	(1)	213	(1)	219	(1)	220	(1)	227	(1)
	264	(1)	265	(1)	266	(1)	267	(1)	268	(1)	269	(1)	270	(1)	271	(1)
	272	(1)	273	(1)	274	(1)	275	(1)	276	(1)	277	(1)	278	(1)	279	(1)
	300	(1)	301	(1)	302	(1)	303	(1)	304	(1)	305	(1)	306	(1)	307	(1)
	308	(1)	309	(1)	423	(1)	549	(1)	656	(1)	903	(1)	964	(1)	967	(1)
.ENDM	1339	(1)	1340	(1)	1550	(1)	211	(1)	212	(1)	213	(1)	214	(1)	215	(1)
	216	(1)	217	(1)	218	(1)	219	(1)	220	(1)	221	(1)	222	(1)	223	(1)
	224	(1)	226	(1)	227	(1)	246	(1)	264	(1)	414	(1)	422	(1)	423	(1)
	430	(1)	469	(1)	470	(1)	495	(1)	547	(1)	549	(1)	569	(1)	635	(1)
	654	(1)	964	(1)												
.ENDR	211	(1)	213	(1)	219	(1)	220	(1)	227	(1)	264	(1)	265	(1)	266	(1)
	267	(1)	268	(1)	269	(1)	270	(1)	271	(1)	272	(1)	273	(1)	274	(1)
	275	(1)	276	(1)	277	(1)	278	(1)	279	(1)	300	(1)	301	(1)	302	(1)
	303	(1)	304	(1)	305	(1)	306	(1)	307	(1)	308	(1)	309	(1)	549	(1)
	593	(1)	683	(1)	964	(1)	967	(1)								
.ENTRY	1546	(1)	413	(1)	551	(1)	556	(1)	953	(1)						
.EXTRN	195	(1)	196	(1)	197	(1)	198	(1)	199	(1)	200	(1)	201	(1)	202	(1)
	203	(1)	204	(1)	205	(1)	206	(1)								
.GLOBL	1550	(1)	423	(1)												
.IDENT	6	(1)														
.IF	1339	(1)	1393	(1)	1550	(1)	211	(1)	213	(1)	219	(1)	220	(1)	227	(1)
	264	(1)	265	(1)	266	(1)	267	(1)	268	(1)	269	(1)	270	(1)	271	(1)
	272	(1)	273	(1)	274	(1)	275	(1)	276	(1)	277	(1)	278	(1)	279	(1)
	300	(1)	301	(1)	302	(1)	303	(1)	304	(1)	305	(1)	306	(1)	307	(1)
	308	(1)	309	(1)	423	(1)	549	(1)	656	(1)	903	(1)	964	(1)	967	(1)
.IFF	1339	(1)	1393	(1)	1550	(1)	211	(1)	213	(1)	219	(1)	220	(1)	227	(1)
	423	(1)	549	(1)	656	(1)	903	(1)								
.IIF	1339	(1)	1393	(1)	211	(1)	213	(1)	219	(1)	220	(1)	227	(1)		
.IRP	211	(1)	212	(1)	213	(1)	219	(1)	220	(1)	227	(1)	264	(1)	265	(1)
	266	(1)	267	(1)	268	(1)	269	(1)	270	(1)	271	(1)	272	(1)	273	(1)
	274	(1)	275	(1)	276	(1)	277	(1)	278	(1)	279	(1)	300	(1)	301	(1)

```
                302  (1)    303  (1)    304  (1)    305  (1)    306  (1)    307  (1)    308  (1)    309  (1)
                549  (1)    593  (1)    683  (1)    964  (1)    967  (1)
.LIBRARY        181  (1)    182  (1)    183  (1)
.LIST           212  (1)    216  (1)    218  (1)    223  (1)    549  (1)
.LONG           241  (1)    264  (1)    265  (1)    266  (1)    267  (1)    268  (1)    269  (1)    270  (1)
                271  (1)    272  (1)    273  (1)    274  (1)    275  (1)    276  (1)    277  (1)    278  (1)
                279  (1)    300  (1)    301  (1)    302  (1)    303  (1)    304  (1)    305  (1)    306  (1)
                307  (1)    308  (1)    309  (1)
.MACRO          211  (1)    213  (1)    219  (1)    220  (1)    227  (1)
.MDELETE        212  (1)    213  (1)    219  (1)    414  (1)    495  (1)
.NLIST          212  (1)    216  (1)    218  (1)    223  (1)    549  (1)
.NOCROSS        214  (1)    215  (1)    216  (1)    217  (1)    223  (1)    224  (1)    226  (1)
.NOSHOW           7  (1)
.PAGE          1014  (1)   1025  (1)   1063  (1)   1115  (1)   1165  (1)   1247  (1)   1297  (1)   1358  (1)
               1405  (1)   1455  (1)   1531  (1)    175  (1)    231  (1)    242  (1)    256  (1)    281  (1)
                 29  (1)    299  (1)    310  (1)    327  (1)    343  (1)    373  (1)    412  (1)    503  (1)
                550  (1)    715  (1)    916  (1)    952  (1)    969  (1)
.PSECT         1027  (1)    218  (1)    233  (1)    244  (1)    375  (1)    549  (1)    971  (1)
.RESTORE        214  (1)    215  (1)    216  (1)    217  (1)    218  (1)    223  (1)    224  (1)    226  (1)
                549  (1)
.SAVE           214  (1)    215  (1)    216  (1)    217  (1)    218  (1)    223  (1)    224  (1)    226  (1)
                549  (1)
.SBTTL         1026  (1)   1406  (1)   1532  (1)    176  (1)    232  (1)    243  (1)    374  (1)    504  (1)
                917  (1)    970  (1)
.SIGNED_WORD    593  (1)    683  (1)
.TITLE            5  (1)
.WEAK           189  (1)    190  (1)    191  (1)    192  (1)    193  (1)
.WORD          1457  (1)   1520  (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	51	#-1205 (1), 1340 (1), 1394 (1), #-1460 (1), #-1464 (1), #-1468 (1)
		#-1476 (1), #-1479 (1), #-1521 (1), #-1525 (1), 1550 (1), #-1552 (1)
		#-1553 (1), #-1554 (1), #-1555 (1), 418 (1), 424 (1), 436 (1)
		445 (1), 466 (1), #-482 (1), #-499 (1), #-565 (1), #-595 (1)
		#-600 (1), #-605 (1), #-608 (1), #-612 (1), #-617 (1), #-622 (1)
		#-627 (1), #-632 (1), #-639 (1), #-641 (1), 643 (1), #-645 (1)
		#-664 (1), #-669 (1), #-683 (1), #-700 (1), #-704 (1), #-708 (1)
		746 (1), #-878 (1), #-879 (1), #-888 (1), #-897 (1), #-898 (1)
		#-903 (1), #-906 (1), 955 (1)
FP	9	#-1308 (1), #-1458 (1), #-463 (1), #-465 (1), 564 (1), 656 (1)
		657 (1), 903 (1), 904 (1)
R0	80	#-1199 (1), #-1214 (1), #-1215 (1), #-1216 (1), #-1218 (1), #-1219 (1)
		#-1222 (1), #-1306 (1), #-1325 (1), #-1329 (1), #-1334 (1), #-1335 (1)
		#-1336 (1), #-1342 (1), #-1344 (1), #-1345 (1), #-1353 (1), #-1354 (1)
		#-1356 (1), #-1377 (1), #-1383 (1), #-1391 (1), #-1402 (1), #-1460 (1)
		#-1467 (1), #-1470 (1), #-1472 (1), #-1477 (1), #-1479 (1), #-1481 (1)
		#-1488 (1), #-1490 (1), #-1493 (1), #-1496 (1), 1498 (1), 1499 (1)
		#-1500 (1), #-1501 (1), #-1504 (1), #-1508 (1), #-1509 (1), #-1512 (1)
		#-1521 (1), #-1523 (1), #-1526 (1), #-425 (1), #-445 (1), #-446 (1)
		447 (1), 448 (1), 449 (1), 453 (1), #-461 (1), #-466 (1)
		#-467 (1), #-482 (1), #-489 (1), #-490 (1), #-491 (1), #-492 (1)
		#-565 (1), #-593 (1), #-657 (1), #-658 (1), #-746 (1), #-747 (1)
		#-904 (1), #-905 (1), #-913 (1), #-955 (1), #-956 (1), #-957 (1)
		#-958 (1), #-961 (1), #-964 (1)
R1	38	#-1306 (1), #-1322 (1), 1323 (1), 1325 (1), #-1332 (1), #-1333 (1)
		#-1335 (1), 1336 (1), #-1342 (1), #-1343 (1), #-1345 (1), #-1462 (1)
		1463 (1), 1467 (1), #-1472 (1), #-1481 (1), #-1484 (1), #-1486 (1)
		#-1490 (1), #-1498 (1), #-1503 (1), #-1506 (1), #-1525 (1), #-1526 (1)
		#-446 (1), 447 (1), #-453 (1), #-456 (1), #-458 (1), #-459 (1)
		#-483 (1), #-489 (1), #-490 (1), #-491 (1), #-956 (1), #-957 (1)
R10	2	#-1204 (1), #-464 (1)
R11	1	#-464 (1)
R2	29	#-1200 (1), #-1328 (1), #-1329 (1), #-1332 (1), #-1333 (1), #-1336 (1)
		#-1343 (1), #-1355 (1), 1546 (1), 413 (1), #-454 (1), 456 (1)
		551 (1), 556 (1), #-643 (1), #-644 (1), #-650 (1), #-651 (1)
		#-652 (1), #-653 (1), #-706 (1), #-708 (1), #-709 (1), #-712 (1)
		#-897 (1), #-898 (1), #-903 (1), #-906 (1)
R3	18	#-1328 (1), #-1330 (1), #-1334 (1), #-1346 (1), #-1355 (1), 413 (1)
		#-448 (1), #-458 (1), #-459 (1), #-462 (1), 551 (1), 556 (1)
		#-652 (1), #-653 (1), #-656 (1), #-659 (1), #-704 (1), #-712 (1)
R4	2	#-1201 (1), #-464 (1)
R5	35	#-464 (1), 551 (1), #-552 (1), 556 (1), #-557 (1), 558 (1)
		597 (1), 602 (1), #-607 (1), 614 (1), 619 (1), 624 (1)
		629 (1), 647 (1), 666 (1), 696 (1), 721 (1), 729 (1)
		737 (1), 745 (1), 759 (1), 767 (1), 775 (1), 783 (1)
		791 (1), 799 (1), 806 (1), 814 (1), 822 (1), 830 (1)
		838 (1), 846 (1), 854 (1), 862 (1), 871 (1)
R6	15	#-1202 (1), #-464 (1), 551 (1), #-553 (1), 556 (1), #-558 (1)
		#-607 (1), 661 (1), 702 (1), 711 (1), 751 (1), 881 (1)
		890 (1), 900 (1), 908 (1)

```
R7                1       #-464    (1)
R8                2       #-1203   (1)    #-464    (1)
R9                1       #-464    (1)
SP               74       #-1017   (1)    #-1020   (1)    #-1021   (1)    #-1022   (1)    #-1023   (1)    #-1071   (1)
                          #-1072   (1)    #-1073   (1)    #-1076   (1)    #-1104   (1)    #-1113   (1)    #-1123   (1)
                          #-1132   (1)    #-1142   (1)    #-1163   (1)    #-1186   (1)    #-1187   (1)    #-1189   (1)
                          #-1191   (1)    #-1206   (1)    #-1207   (1)    #-1210   (1)    #-1211   (1)    #-1216   (1)
                          #-1223   (1)    #-1235   (1)    #-1240   (1)      1241   (1)    #-1245   (1)    #-1278   (1)
                          #-1287   (1)    #-1307   (1)    #-1313   (1)    #-1314   (1)    #-1316   (1)    #-1318   (1)
                          #-1321   (1)    #-1330   (1)      1344   (1)    #-1356   (1)      1364   (1)      1365   (1)
                            1376   (1)      1390   (1)    #-1400   (1)    #-1401   (1)    #-1402   (1)    #-1403   (1)
                          #-1486   (1)    #-1488   (1)    #-1499   (1)    #-1501   (1)    #-1527   (1)    #-423    (1)
                          #-461    (1)    #-463    (1)    #-465    (1)    #-467    (1)      476    (1)    #-483    (1)
                          #-486    (1)    #-487    (1)    #-488    (1)    #-500    (1)    #-501    (1)    #-564    (1)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.19
Command processing	872	00:00:00.21	00:00:00.65
Pass 1	1183	00:00:05.70	00:00:08.75
Symbol table sort	0	00:00:00.39	00:00:00.39
Pass 2	12	00:00:01.51	00:00:02.89
Symbol table output	0	00:00:00.09	00:00:00.29
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	7	00:00:00.67	00:00:01.05
Assembler run totals	2096	00:00:08.59	00:00:14.21

```
The working set limit was 3900 pages.
314051 bytes (614 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1322 non-local and 75 local symbols.
1558 source lines were read in Pass 1, producing 113 object records in Pass 2.
127 pages of virtual memory were used to define 47 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	13
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	16
SYS$COMMON:[SYSLIB]LIB.MLB;2	1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	17
TOTALS (all libraries)	47

```
1486 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:EXCEPT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 88 Libraries, Equated Symbols
(1) 197 FAO - Main program
(1) 410 GETCHAR - Routine to get next char from input string
(1) 458 GETCOUNT - Routine to get repeat-count or field-width
(1) 528 CVTASC - Insert ASCII string
(1) 666 CVTNUM - Convert numeric parameter to ASCII
(1) 877 QUICKSERVE - Small service routines
(1) 1015 PERCENT - Time directives and plural 'S'
(1) 1114 HANDLER - Condition handler

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element FAO.MAR
0000     2 ; *1     6-FEB-1986 15:18:28 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element FAO.MAR
0000     4         .TITLE  FAO     *** - FORMATTED ASCII OUTPUT
0000     5         .IDENT  '03-02'
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1978, 1980, 1982 BY                                        *
0000    11 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    12 ;*  ALL RIGHTS RESERVED.                                                     *
0000    13 ;*                                                                           *
0000    14 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    15 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    16 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    17 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    18 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    19 ;*  TRANSFERRED.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    22 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    23 ;*  CORPORATION.                                                             *
0000    24 ;*                                                                           *
0000    25 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    26 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    27 ;*                                                                           *
0000    28 ;*                                                                           *
0000    29 ;*****************************************************************************
0000    30
0000    31 ;++
0000    32 ; FACILITY: SYSTEM SERVICE
0000    33 ;
0000    34 ; ABSTRACT:
0000    35 ;
0000    36 ;       This module provides general formatting services.  It converts
0000    37 ;       binary values to octal, hexadecimal, and decimal ASCII
0000    38 ;       representations, and also inserts ASCII strings and converts
0000    39 ;       date and time to ASCII.
0000    40 ;
0000    41 ; ENVIRONMENT:
0000    42 ;
0000    43 ;       VAX Diagnostic Supervisor.
0000    44 ;
0000    45 ; AUTHOR:
0000    46 ;
0000    47 ;       Henry M. Levy
0000    48 ;
0000    49 ; CREATION DATE:
0000    50 ;
0000    51 ;       29-JAN-1977
0000    52 ;
0000    53 ; MODIFIED BY:
0000    54 ;
0000    55 ;       03-02   Jack Stansbury, Version 6.11?, January 6, 1983
0000    56 ;               Change the IvSSRq error return code to BadParam. This is
0000    57 ;               to make it consistent with the Design Guide.
```

```
0000     58 ;
0000     59 ;       03-01   Jack Stansbury, Version 6.9, July 25, 1982
0000     60 ;               Changed the real SYSFAO module to work in the Diagnostic
0000     61 ;               Supervisor environment. This includes changing the PSECT names,
0000     62 ;               changing a return error code, and changing the Handler routine
0000     63 ;               code.
0000     64 ;
0000     65 ;       V03-005 MSH0001         Maryann S. Hinden       20-NOV-1981
0000     66 ;               Use longword displacement to reference EXE$SIGTORET.
0000     67 ;
0000     68 ;       V03-004 DWT0001         David W. Thiel          06-Nov-1981
0000     69 ;               Fixed condition handler.  Check argument to $ASCTIM to
0000     70 ;               prevent exception in $ASCTIM.
0000     71 ;
0000     72 ;       V03-003 PCA0001         Paul C. Anagnostopoulos         22-Jul-1981
0000     73 ;               Fixed a bug wherein !AF did not replace unprintable
0000     74 ;               characters if it encountered result string overflow.
0000     75 ;               Now it replaces those characters that it does copy.
0000     76 ;
0000     77 ;       V03-002 TCM0001         Trudy C. Matthews       10-Mar-1981
0000     78 ;               Change CALLS with word displacement to CALLS with longword
0000     79 ;               displacement.
0000     80 ;
0000     81 ;       V03-001 TMH0001         Tim Halvorsen   24-Feb-1981
0000     82 ;               Add condition handler to catch access violations
0000     83 ;               and the like, so that services like $PUTMSG do
0000     84 ;               not cause an access violation in programs like DCL
0000     85 ;               simply because not enough arguments were supplied.
0000     86 ;--
```

```
                                0000     88          .SBTTL  Libraries, Equated Symbols
                                0000     89
                                0000     90 ;
                                0000     91 ; MACROS:
                                0000     92 ;
                                0000     93
                                0000     94          $SSDEF                          ; define system status codes
                                0000                 .SAVE   LOCAL_BLOCK
                                0000                 .RESTORE
                                0000     95 ;        $CHFDEF                         ; Condition handling facility          [01]
                                0000     96 ;        $SFDEF                          ; Call frame definitions               [01]
                                0000     97
                                0000     98 ;
                                0000     99 ; EQUATED SYMBOLS:
                                0000    100 ;
                                0000    101
                    00000000    0000    102          ARGCOUNT = 0                    ; offset to argument count
                    00000004    0000    103          INDSC    = 4                    ; offset to input string descriptor
                    00000008    0000    104          OUTLEN   = 8                    ; offset to output length
                    0000000C    0000    105          OUTDSC   = 12                   ; offset to output buffer descriptor
                    00000010    0000    106          FIRSTARG = 16                   ; offset to first conversion param
                                0000    107
                    FFFFFFF0    0000    108          INLEN    = -16                  ; local offset to input length remaining
                    FFFFFFF4    0000    109          INPTR    = -12                  ; local offset to input string pointer
                    FFFFFFF8    0000    110          LASTVAL  = -8                   ; local offset to last value converted
                    FFFFFFFC    0000    111          FIELDEND = -4                   ; local offset to end of defined field
                                0000    112
                    0000000D    0000    113          CR       = 13                   ; carriage return
                    0000000A    0000    114          LF       = 10                   ; line feed
                    00000021    0000    115          EXCL     = 33                   ; exclamation ('!')
                    00000009    0000    116          TAB      = 9                    ; horizontal tab
                    0000000C    0000    117          FF       = 12                   ; form feed
```

```
                                0000   119
                                0000   120 ;
                                0000   121 ; OWN STORAGE:
                                0000   122 ;
                                0000   123
                           00000000    124          .PSect  Data    NoExe, Shr, NoWrt, Long ;                       [01]
                                0000   125
                                0000   126 ASC_NAMES:
42 41 39 38 37 36 35 34 33 32 31 30     127          .ASCII  /0123456789ABCDEF/      ; ASCII digits
                  46 45 44 43   000C
                                0010   128
                                0010   129 ;
                                0010   130 ;       The following table contains the first character for all
                                0010   131 ;       FAO conversion directives.  The first part of the table
                                0010   132 ;       contains the first character for two-character directives,
                                0010   133 ;       while the second half of the table contains the one-character
                                0010   134 ;       directives.
                                0010   135 ;
                                0010   136 ;       NOTE -- The ordering of this table must be preserved.  The index
                                0010   137 ;               of the directives found in this table is used to dispatch
                                0010   138 ;               via a CASE statement in the main program (FAO).
                                0010   139 ;               Routine CVTNUM also uses the index to dispatch and to
                                0010   140 ;               compute the proper radix for the conversion.
                                0010   141 ;
                                0010   142
                                0010   143 CNTRL_TABLE:
                                0010   144 TWO_CHAR_CNTRLS:
                             4F 0010   145          .ASCII  /O/                     ; octal conversions
                             58 0011   146          .ASCII  /X/                     ; hex conversions
                             55 0012   147          .ASCII  /U/                     ; unsigned decimal
                             53 0013   148          .ASCII  /S/                     ; signed decimal
                             5A 0014   149          .ASCII  /Z/                     ; unsigned decimal zero filled
                             41 0015   150          .ASCII  /A/                     ; ascii insertion directives
                             25 0016   151          .ASCII  /%/                     ; time conversion, or plural indication
                             2A 0017   152          .ASCII  /*/                     ; character repeater
                                0018   153
                                0018   154 ONE_CHAR_CNTRLS:
                             2B 0018   155          .ASCII  /+/                     ; skip argument
                             2D 0019   156          .ASCII  /-/                     ; backup argument
                             3C 001A   157          .ASCII  /</                     ; begin field definition
                             3E 001B   158          .ASCII  />/                     ; end of field definition
                                001C   159
                                001C   160 REPLACE_CHRS:                           ; these are one or two char replacements
                             2F 001C   161          .ASCII  ./.                     ; newline
                             5F 001D   162          .ASCII  /_/                     ; tab
                             5E 001E   163          .ASCII  /^/                     ; form feed
                             21 001F   164          .ASCII  /!/                     ; insert exclamation
                                0020   165
                       00000010 0020   166 CNTRL_LENGTH = .-CNTRL_TABLE           ; length of table
                                0020   167
                       00000008 0020   168 ONECHAR_INDEX = CNTRL_LENGTH - <ONE_CHAR_CNTRLS - CNTRL_TABLE>
                                0020   169
                       0000000C 0020   170 REPL_OFFSET = REPLACE_CHRS - CNTRL_TABLE ; offset of replacement chars
                                0020   171
                                0020   172 STRING_TYPES:
                   46 44 53 43  0020   173          .ASCII  /CSDF/                  ; ascii string types
                                0024   174 DATA_TYPES:
```

```
                     4C 57 42  0024   175          .ASCII  /BWL/                   ; byte, word , or long
                               0027   176 PERCENT_STR:
                     54 44 53  0027   177          .ASCII  /SDT/                   ; subtypes for ! directive
                               002A   178 FIELDS:
                     20 10 08  002A   179          .BYTE   8,16,32                 ; field size for B,W,and L
                               002D   180 REPLACEMENT:
                  21 0C 09 0A  002D   181          .BYTE   LF,TAB,FF,EXCL          ; simple replacement table
                               0031   182
                               0031   183 ;
                               0031   184 ; The following array contains the number of Octal and Hex digits in
                               0031   185 ; byte , word, and longword fields.  The byte digits are first, the
                               0031   186 ; hex digits starting at the 4'th entry so that the array may be
                               0031   187 ; context indexed.
                               0031   188 ;
                               0031   189
                               0031   190 OCT_HEX_DIGITS:
                  00 0B 06 03  0031   191          .BYTE   3,6,11,0
                     08 04 02  0035   192          .BYTE   2,4,8
                               0038   193
                               0038   194 RADIX:
               0A 0A 0A 10 08  0038   195          .BYTE   8,16,10,10,10           ; radix for numeric conversisons
```

```
                     003D   197          .SBTTL  FAO - Main program
                 00000000   198          .PSect  Code    Exe, Shr, NoWrt, Long   ;                                    [01]
                     0000   199 ;++
                     0000   200 ; FUNCTIONAL DESCRIPTION:
                     0000   201 ;
                     0000   202 ;       This routine is the entry point for the FAO and FAOL system
                     0000   203 ;       services.   The caller's control string is scanned for control
                     0000   204 ;       characters ('!').  All other information is simply passed to
                     0000   205 ;       the output buffer.  If a control directive is found, it is parsed
                     0000   206 ;       and an action routine is dispatched.
                     0000   207 ;
                     0000   208 ; CALLING SEQUENCE:
                     0000   209 ;
                     0000   210 ;       CALLS or CALLG  to SYS$FAO or SYS$FAOL
                     0000   211 ;
                     0000   212 ; INPUT PARAMETERS:
                     0000   213 ;
                     0000   214 ;       INDSC   - The address of a string descriptor for the input
                     0000   215 ;                 control string.
                     0000   216 ;       OUTLEN  - The address of a word to receive the length of
                     0000   217 ;                 the output string
                     0000   218 ;       OUTDSC  - The address of a string descriptor for the output
                     0000   219 ;                 buffer.
                     0000   220 ;       FIRSTARG - For FAOL , this is the address of a list of longword
                     0000   221 ;                  parameters.  For FAO , this is the first of a
                     0000   222 ;                  variable number of parameters which
                     0000   223 ;                  may have been passed on the call argument list.
                     0000   224 ;
                     0000   225 ; IMPLICIT INPUTS:
                     0000   226 ;
                     0000   227 ;       none
                     0000   228 ;
                     0000   229 ; OUTPUT PARAMETERS:
                     0000   230 ;
                     0000   231 ;       OUTLEN  - Word pointed to will receive length of output buffer.
                     0000   232 ;
                     0000   233 ; IMPLICIT OUTPUTS:
                     0000   234 ;
                     0000   235 ;       none
                     0000   236 ;
                     0000   237 ; COMPLETION CODES:
                     0000   238 ;
                     0000   239 ;       SS$_NORMAL      - success code, normal return
                     0000   240 ;       SS$_BUFFEROVF   - output buffer overflow, attempt to write past end
                     0000   241 ;                         of output buffer
                     0000   242 ;       SS$_BadParam    - Invalid directive specified or unable to read        [02]
                     0000   243 ;                         argument list or address arguments                  [02]
                     0000   244 ;
                     0000   245 ; SIDE EFFECTS:
                     0000   246 ;
                     0000   247 ;       none
                     0000   248 ;
                     0000   249 ;--
```

```
                                0000
                                0000  ;+
                                0000  ; Global register usage:
                                0000  ;
                                0000  ;       R7,R8 - scratch registers
                                0000  ;       R9  - number of characters remaining in output buffer
                                0000  ;       R10 - current position in output buffer
                                0000  ;       R11 - pointer to next conversion parameter
                                0000  ;
                                0000  ; Locals
                                0000  ;
                                0000  ;       INLEN(FP) - (word) length of input control string
                                0000  ;       INPTR(FP) - address of position in input control string
                                0000  ;-
                                0000
                                0000  ;
                                0000  ; Entry point for call with multiple arguments on stack
                                0000  ;
                                0000
                                0000  EXE$FAO::
                                0000
                          0FFC  0000          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>     ; save all registers
                                0002  ;       MOVAB   W^HANDLER,(FP)          ; Establish condition handler          [01]
                5B   10 AC   DE  0002          MOVAL   FIRSTARG(AP),R11        ; get address of first argument
                        06   11  0006          BRB     FAO                     ; go to main routine
                                0008
                                0008  ;
                                0008  ; Entry point for FAOL call.
                                0008  ;
                                0008
                                0008  EXE$FAOL::
                                0008
                          0FFC  0008          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                000A  ;       MOVAB   W^HANDLER,(FP)          ; Establish condition handler          [01]
                5B   10 AC   D0  000A          MOVL    FIRSTARG(AP),R11        ; address of first argument
```

```
                                000E
                                000E  FAO:
                      7E   7C   000E          CLRQ    -(SP)                      ; save space for LASTVAL and FIELDEND
              7E   04 BC   7D   0010          MOVQ    @INDSC(AP),-(SP)           ; save locals on stack
              59   0C BC   7D   0014          MOVQ    @OUTDSC(AP),R9             ; load output descriptor into R9,R10
                 59   59   3C   0018          MOVZWL  R9,R9                      ; ensure word length
                                001B
                                001B  ;
                                001B  ; Look for a control character in the input string.  Copy text
                                001B  ; up to the control, if any , to the output buffer.
                                001B  ;
                                001B
                                001B  MAIN_SCAN:
                      7E   D4   001B          CLRL    -(SP)                      ; indicate control not found
        F4 BD   F0 AD   21   3A   001D          LOCC    #EXCL,INLEN(FP),@INPTR(FP) ; search for control char
                      02   13   0023          BEQL    10$                        ; branch if not found
                      6E   D6   0025          INCL    (SP)                       ; set indicator to show char. found
                                0027  10$:
           56   F0 AD   50   A3   0027          SUBW3   R0,INLEN(FP),R6            ; calculate bytes to move
                F0 AD   50   D0   002C          MOVL    R0,INLEN(FP)               ; update input length remaining
                   59   56   A2   0030          SUBW    R6,R9                      ; update and test output length
                        75   19   0033          BLSS    OVERFLOW                   ; not enough room, error exit
           6A   F4 BD   56   28   0035          MOVC3   R6,@INPTR(FP),(R10)        ; move text part of input string
                   76 8E   E9   003A          BLBC    (SP)+,DONE                 ; leave if no controls left
                F4 AD   51   D0   003D          MOVL    R1,INPTR(FP)               ; update input address pointer
                   5A   53   D0   0041          MOVL    R3,R10                     ; update output address pointer
                        7C   10   0044          BSBB    GETCHAR                    ; skip control char
```

```
                                0046  ;
                                0046  ; Parse the directive which has been found in the input string.  Set
                                0046  ; up: R0 = remaining count in CNTRL_TABLE
                                0046  ;     R4 = second char if two-char directive
                                0046  ;     R5 = repeat count
                                0046  ;     R6 = field width
                                0046  ;
                                0046
                                0046  PARSE_DIRECTIVE:
                                0046
                 55   01   D0   0046          MOVL    #1,R5                   ; default repeat count is 1
                      52   D4   0049          CLRL    R2                      ; paren indicator ( not found yet )
                    0081   30   004B          BSBW    GETCOUNT                ; pull off count, if any
                      72   10   004E          BSBB    GETCHAR                 ; get next char from input string
                 53   28   91   0050          CMPB    #^A/(/,R3               ; was next char a paren?
                      0D   12   0053          BNEQ    20$                     ; branch if not
                      52   D6   0055          INCL    R2                      ; set paren found indicator
                      56   D5   0057          TSTL    R6                      ; was there a repeat count?
                      03   19   0059          BLSS    10$                     ; no..use default
                 55   56   D0   005B          MOVL    R6,R5                   ; else get repeat count
                                005E  10$:
                      6F   10   005E          BSBB    GETCOUNT                ; look for field width
                      60   10   0060          BSBB    GETCHAR                 ; get next char
                                0062  20$:
     00000010'EF  10   53   3A   0062          LOCC    R3,#CNTRL_LENGTH,CNTRL_TABLE ; check character in table
                      39   13   006A          BEQL    ILLEGAL                 ; illegal directive exit
                 08   50   D1   006C          CMPL    R0,#ONECHAR_INDEX       ; is this a one char directive?
                      05   15   006F          BLEQ    30$                     ; yes, don't need any more
                      4F   10   0071          BSBB    GETCHAR                 ; get second control char
                 54   53   D0   0073          MOVL    R3,R4                   ; move to R4 for return
                                0076  30$:
                   02 52   E9   0076          BLBC    R2,40$                  ; skip if no paren found
                      47   10   0079          BSBB    GETCHAR                 ; else skip paren char
                                007B  40$:
                                007B
            53   10   50   C3   007B          SUBL3   R0,#CNTRL_LENGTH,R3     ; compute offset for case table
                                007F
                                007F  ;
                                007F  ; The following  does a BSBB to the case dispatch
                                007F  ; table.  The service routines do an RSB and return into CASE_LOOP.
                                007F  ;
                                007F
                      02   11   007F          BRB     CASE_LOOP               ; start processing loop
                                0081  CASE_BSB:
                      05   10   0081          BSBB    FAO_CASE                ; dispatch next directive
                                0083  CASE_LOOP:
                   FB 55   F4   0083          SOBGEQ  R5,CASE_BSB             ; repeat as specified
                      93   11   0086          BRB     MAIN_SCAN               ; else continue string processing
```

```
                     0088   364 ;
                     0088   365 ; Here is the main dispatch table for dispatching FAO service
                     0088   366 ; routines.  The case is entered via BSBB from CASE_BSB. The routines
                     0088   367 ; RSB to CASE_LOOP.  Since the 5 numeric conversion directives all
                     0088   368 ; dispatch to the same routine, the case has a base of 5 and the
                     0088   369 ; numeric directives fall through to the statement following the CASE.
                     0088   370 ;
                     0088   371 ; Registers R0, R1, and R2 may be scratched by service routines.
                     0088   372 ;
                     0088   373
                     0088   374 FAO_CASE:
                     0088   375         CASE    R3,<-                           ; dispatch to service routine
                     0088   376                 CVTASC,-                        ; ascii string insertion
                     0088   377                 PERCENT,-                       ; insert ascii time or plural 'S'
                     0088   378                 REPEATIT,-                      ; repeat character 'n' times
                     0088   379                 INCR_ARGPTR,-                   ; skip next parameter
                     0088   380                 DECR_ARGPTR,-                   ; backup to previous parameter
                     0088   381                 STARTFIELD,-                    ; define fixed length field
                     0088   382                 ENDFIELD,-                      ; terminate fixed length field
                     0088   383                 NEWLINE,-                       ; insert CR/LF
                     0088   384                 INSERT_CHAR,-                   ; insert TAB
                     0088   385                 INSERT_CHAR,-                   ; insert form feed
                     0088   386                 INSERT_CHAR,-                   ; insert '!'
                     0088   387             >,B,#5                              ; offset start by 5
 0A'  05    53   8F  0088                  CASEB   R3,#5,S^#<<30001$-30000$>/2>-1
                     008C       30000$:
            0078'    008C                       .SIGNED_WORD    CVTASC-30000$
            022F'    008E                       .SIGNED_WORD    PERCENT-30000$
            01F4'    0090                       .SIGNED_WORD    REPEATIT-30000$
            01D7'    0092                       .SIGNED_WORD    INCR_ARGPTR-30000$
            01DA'    0094                       .SIGNED_WORD    DECR_ARGPTR-30000$
            020B'    0096                       .SIGNED_WORD    STARTFIELD-30000$
            021D'    0098                       .SIGNED_WORD    ENDFIELD-30000$
            01DD'    009A                       .SIGNED_WORD    NEWLINE-30000$
            01E5'    009C                       .SIGNED_WORD    INSERT_CHAR-30000$
            01E5'    009E                       .SIGNED_WORD    INSERT_CHAR-30000$
            01E5'    00A0                       .SIGNED_WORD    INSERT_CHAR-30000$
                     00A2       30001$:
                     00A2   388
      00D3   31      00A2   389                 BRW     CVTNUM                  ; dispatch to numeric conversion
                     00A5   390
                     00A5   391
                     00A5   392 ILLEGAL:
   50   14   3C      00A5   393                 MOVZWL  #SS$_BadParam,R0        ; Error return code                     [02]
        0C   11      00A8   394                 BRB     FAO_EXIT
                     00AA   395
                     00AA   396 OVERFLOW:
50  0601  8F   3C    00AA   397                 MOVZWL  #SS$_BUFFEROVF,R0       ; error return code
        59   D4      00AF   398                 CLRL    R9                      ; ensure correct return length
        03   11      00B1   399                 BRB     FAO_EXIT
                     00B3   400
                     00B3   401 DONE:
   50   01   3C      00B3   402                 MOVZWL  #SS$_NORMAL,R0          ; no errors
                     00B6   403
                     00B6   404 FAO_EXIT:
   08   AC   D5      00B6   405                 TSTL    OUTLEN(AP)              ; was a return length required?
        06   13      00B9   406                 BEQL    10$                     ; branch if not
```

```
                08 BC    0C BC    59    A3  00BB    407          SUBW3   R9,aOUTDSC(AP),aOUTLEN(AP) ; compute and return output buffer length
                                    04  00C1    408 10$:     RET
```

```
                     00C2    410          .SBTTL  GETCHAR - Routine to get next char from input string
                     00C2    411
                     00C2    412 ;++
                     00C2    413 ;
                     00C2    414 ; FUNCTIONAL DESCRIPTION:
                     00C2    415 ;
                     00C2    416 ;       This routine gets the next character from the input control
                     00C2    417 ;       string, updating the length and address pointers.  If the length
                     00C2    418 ;       goes negative, an error exit is called.
                     00C2    419 ;
                     00C2    420 ; CALLING SEQUENCE:
                     00C2    421 ;
                     00C2    422 ;       JSB (R8)
                     00C2    423 ;
                     00C2    424 ; INPUT PARAMETERS:
                     00C2    425 ;
                     00C2    426 ;       none
                     00C2    427 ;
                     00C2    428 ; IMPLICIT INPUTS:
                     00C2    429 ;
                     00C2    430 ;       INLEN(FP) -  lower word has remaining length of input string
                     00C2    431 ;       INPTR(FP) -  is pointer to current string position
                     00C2    432 ;
                     00C2    433 ; OUTPUTS:
                     00C2    434 ;
                     00C2    435 ;       R3 - next character in input string
                     00C2    436 ;
                     00C2    437 ; IMPLICIT OUTPUTS:
                     00C2    438 ;
                     00C2    439 ;       none
                     00C2    440 ;
                     00C2    441 ; COMPLETION CODES:
                     00C2    442 ;
                     00C2    443 ;       none
                     00C2    444 ;
                     00C2    445 ; SIDE EFFECTS:
                     00C2    446 ;
                     00C2    447 ;       input pointers on stack are updated
                     00C2    448 ;       error may cause jump to ILLEGAL
                     00C2    449 ;--
```

```
                        00C2   451 GETCHAR:
          F0 AD   B7    00C2   452         DECW    INLEN(FP)                ; decr input length remaining
             DE   19    00C5   453         BLSS    ILLEGAL                  ; error if no more left
     53   F4 BD   9A    00C7   454         MOVZBL  @INPTR(FP),R3            ; get next character
          F4 AD   D6    00CB   455         INCL    INPTR(FP)                ; update pointer
                  05    00CE   456         RSB                              ; return
```

```
                        00CF    458          .SBTTL  GETCOUNT - Routine to get repeat-count or field-width
                        00CF    459
                        00CF    460 ;++
                        00CF    461 ;
                        00CF    462 ; FUNCTIONAL DESCRIPTION:
                        00CF    463 ;
                        00CF    464 ;       This subroutine to PARSE_DIRECTIVE scans for a repeat-count or
                        00CF    465 ;       field-width in the directive in the input stream.  If a numeric
                        00CF    466 ;       count is found, it is converted to binary.  If a '#' character
                        00CF    467 ;       is found, the count is taken from the next parameter
                        00CF    468 ;       in the parameter list.
                        00CF    469 ;
                        00CF    470 ; CALLING SEQUENCE:
                        00CF    471 ;
                        00CF    472 ;       JSB or BSB
                        00CF    473 ;
                        00CF    474 ; INPUTS:
                        00CF    475 ;
                        00CF    476 ;       R11     - parameter pointer
                        00CF    477 ;
                        00CF    478 ; IMPLICIT INPUTS:
                        00CF    479 ;
                        00CF    480 ;       none
                        00CF    481 ;
                        00CF    482 ; OUTPUTS:
                        00CF    483 ;
                        00CF    484 ;       R6      - value of count, if # or number found, else -1
                        00CF    485 ;
                        00CF    486 ; IMPLICIT OUTPUTS:
                        00CF    487 ;
                        00CF    488 ;       R11 may be modified if a parameter is taken from the stack
                        00CF    489 ;
                        00CF    490 ; COMPLETION CODES:
                        00CF    491 ;
                        00CF    492 ;       none
                        00CF    493 ;
                        00CF    494 ; SIDE EFFECTS:
                        00CF    495 ;
                        00CF    496 ;       R1, R3, and R4 are destroyed
                        00CF    497 ;--
```

```
                                00CF
                                00CF  GETCOUNT:
          56    01    CE      00CF          MNEGL   #1,R6                      ; not found indicator
       F4 BD    23    91      00D2          CMPB    #^A/#/,@INPTR(FP)          ; is this a param. count?
                26    13      00D6          BEQL    40$                        ; yes .. pull next param
                53    7C      00D8          CLRQ    R3                         ; zero buffer for digit (R3)
                              00DA                                             ; ... and accumulator for sum (R4)
       51   F4 AD    D0      00DA          MOVL    INPTR(FP),R1               ; remember where we were
                              00DE  10$:
   53  F4 BD    30    83      00DE          SUBB3   #^A/0/,@INPTR(FP),R3       ; subtract ascii 0 from char
                0F    19      00E3          BLSS    20$                        ; branch if not numeric
          53    09    91      00E5          CMPB    #^A/9/-^A/0/,R3            ; still numeric?
                0A    19      00E8          BLSS    20$                        ; no, branch
          54    0A    C4      00EA          MULL2   #10,R4                     ; shift for next digit
          54    53    C0      00ED          ADDL    R3,R4                      ; add in next digit
                D0    10      00F0          BSBB    GETCHAR                    ; skip digit we took
                EA    11      00F2          BRB     10$                        ; continue while numeric
                              00F4  20$:
       F4 AD    51    D1      00F4          CMPL    R1,INPTR(FP)               ; did we get any chars?
                03    13      00F8          BEQL    30$                        ; no, leave
          56    54    D0      00FA          MOVL    R4,R6                      ; yes, return value
                              00FD  30$:
                      05      00FD          RSB                                ; return
                              00FE
                              00FE  40$:
          56    8B    D0      00FE          MOVL    (R11)+,R6                  ; get value from next parameter
                BF    10      0101          BSBB    GETCHAR                    ; skip '#'
                      05      0103          RSB                                ; return
```

```
                     528          .SBTTL  CVTASC  - Insert ASCII string
                     529          .LIST MEB
                     530
                     531 ;++
                     532 ;
                     533 ; FUNCTIONAL DESCRIPTION:
                     534 ;
                     535 ;       Service routine to handle ASCII string insertions.
                     536 ;       Strings are specified by several different methods.  For
                     537 ;       filled strings (AF) , non-printing characters are output
                     538 ;       as dots ('.').
                     539 ;
                     540 ; CALLING SEQUENCE:
                     541 ;
                     542 ;       JSB or BSB
                     543 ;
                     544 ; INPUTS:
                     545 ;
                     546 ;       R3      - index of first control char in CNTRL_TABLE
                     547 ;       R4      - second control character
                     548 ;       R6      - output field width
                     549 ;       R9      - output buffer length remaining
                     550 ;       R10     - output buffer pointer
                     551 ;       R11     - parameter pointer
                     552 ;
                     553 ; IMPLICIT INPUTS:
                     554 ;
                     555 ;       none
                     556 ;
                     557 ; OUTPUTS:
                     558 ;
                     559 ;       none
                     560 ;
                     561 ; IMPLICIT OUTPUTS:
                     562 ;
                     563 ;       R9 and R10 are update to point to current position in output buffer
                     564 ;       R11 is updated as parameters are taken from the stack
                     565 ;
                     566 ; ROUTINE VALUE:
                     567 ;
                     568 ;       none
                     569 ;
                     570 ; SIDE EFFECTS:
                     571 ;
                     572 ;       R7 and R8 are destroyed
                     573 ;--
```

```
                                0104
                                0104  CVTASC:
                0078 8F   BB    0104          PUSHR   #^M<R3,R4,R5,R6>          ; save registers
                     57   D4    0108          CLRL    R7                        ; set filled indicator to not filled
  00000020'EF   04   54   3A    010A          LOCC    R4,#4,STRING_TYPES        ; search for string subtype
                     61   13    0112          BEQL    110$                      ; error if not found
                                0114
                                0114  ;
                                0114  ;       R0 = 1 - filled , 2 - 2 arg desc. , 3 - str. desc. , 4 - cstring
                                0114  ;
                                0114          CASE    R0,<10$,20$,30$>,B,#2     ; case on descriptor type, base = 2
          02'   02   50   8F    0114          CASEB   R0,#2,S^#<<30003$-30002$>/2>-1
                                0118      30002$:
                        0008'   0118          .SIGNED_WORD    10$-30002$
                        000D'   011A          .SIGNED_WORD    20$-30002$
                        0015'   011C          .SIGNED_WORD    30$-30002$
                                011E      30003$:
                                011E
                                011E  ;
                                011E  ; Case falls through here for filled ascii strings. Two argument
                                011E  ; descriptor is used.
                                011E  ;
                                011E
                     57   D6    011E          INCL    R7                        ; set filled indicator for filled ascii
                                0120  10$:
                51   8B   7D    0120          MOVQ    (R11)+,R1                 ; get length and address
                     0E   11    0123          BRB     40$                       ; continue
                                0125
                                0125  ;
                                0125  ; Standard system string descriptor
                                0125  ;
                                0125
                                0125  20$:
                51   9B   7D    0125          MOVQ    @(R11)+,R1                ; move descriptor to R1,R2
                51   51   3C    0128          MOVZWL  R1,R1                     ; make sure length is word
                     06   11    012B          BRB     40$                       ; continue
                                012D
                                012D  ;
                                012D  ; Ascii counted string, first byte contains length
                                012D  ;
                                012D
                                012D  30$:
                52   8B   D0    012D          MOVL    (R11)+,R2                 ; address of counted string
                51   82   9A    0130          MOVZBL  (R2)+,R1                  ; get length and skip byte count
                                0133
                                0133  40$:
                                0133
                                0133  ;
                                0133  ; Here, R1 has string length, R2 has string address. Check length against
                                0133  ; specified field width to decide how much string to move.
                                0133  ;
                                0133
                58   56   D0    0133          MOVL    R6,R8                     ; was a width specified?
                     03   18    0136          BGEQ    50$                       ; branch if so
                58   51   D0    0138          MOVL    R1,R8                     ; if not, use string length instead
                                013B  50$:
                                013B
```

```
                                013B   ;
                                013B   ; The string is moved to the output buffer with blank fill at the
                                013B   ; end.  The output pointers are then updated by the field width, so
                                013B   ; that the string will be truncated if it was longer than the field
                                013B   ; width.  If the string is filled, a second pass is made to change
                                013B   ; non-printing characters to dots.
                                013B   ;
                                013B
                  56   59   D0  013B          MOVL    R9,R6                   ; copy remaining char count
                                013E                                          ; NOTE we have to use R6 here.
                  59   58   C2  013E          SUBL    R8,R9                   ; update length remaining
                       03   19  0141          BLSS    55$                     ; Overflow, use remaining length.
                  56   58   D0  0143          MOVL    R8,R6                   ; else move only required length
   6A   56   20   62   51   2C  0146   55$:   MOVC5   R1,(R2),#^A/ /,R6,(R10) ; move string, fill at end
                                014C
                  52   5A   D0  014C          MOVL    R10,R2                  ; save output address
                  5A   56   C0  014F          ADDL    R6,R10                  ; update output pointer
                    14 57   E9  0152          BLBC    R7,90$                  ; all done if not filled ASCII
                                0155   60$:                                   ; R7 will now become loop counter.
                  20   62   91  0155          CMPB    (R2),#^O40              ; printing character?
                       06   19  0158          BLSS    70$                     ; no, fill with dot
              7E 8F   62   91  015A          CMPB    (R2),#^O176             ; still printing?
                       03   15  015E          BLEQ    80$                     ; yes, skip this one
                                0160   70$:
                  62   2E   90  0160          MOVB    #^A/./,(R2)             ; insert dot in place of char
                                0163   80$:
                       52   D6  0163          INCL    R2                      ; point to next character
              EC 57    56   F3  0165          AOBLEQ  R6,R7,60$               ; continue until done
                                0169
                                0169   90$:
                       59   D5  0169          TSTL    R9                      ; Did we get result overflow above?
                       05   19  016B          BLSS    100$                    ; Yes, branch to tell user.
                 0078 8F   BA  016D          POPR    #^M<R3,R4,R5,R6>
                           05  0171          RSB                             ; return
                                0172
                                0172   100$:
                    FF35   31  0172          BRW     OVERFLOW
                                0175   110$:
                    FF2D   31  0175          BRW     ILLEGAL
```

```
0178    666          .SBTTL  CVTNUM - Convert numeric parameter to ASCII
0178    667
0178    668 ;++
0178    669 ;
0178    670 ; FUNCTIONAL DESCRIPTION:
0178    671 ;
0178    672 ;       This routine handles the various HEX, OCTAL, and DECIMAL
0178    673 ;       conversions. The proper field is extracted from the
0178    674 ;       parameter (byte, word , or long) and the needed output
0178    675 ;       width is determined.  This is compared with the user
0178    676 ;       specified field width to determine if padding of filling
0178    677 ;       is needed.  The entire field with fill is built on the
0178    678 ;       stack and then moved so that the result will be correct
0178    679 ;       on buffer overflow.
0178    680 ;
0178    681 ; CALLING SEQUENCE:
0178    682 ;
0178    683 ;       JSB or BSB
0178    684 ;
0178    685 ; INPUTS:
0178    686 ;
0178    687 ;       R3      - index of directive in CNTRL_TABLE.
0178    688 ;                       0 = Octal
0178    689 ;                       1 = heX
0178    690 ;                       2 = Unsigned decimal
0178    691 ;                       3 = Signed decimal
0178    692 ;                       4 = Zero filled unsigned decimal
0178    693 ;       R4      - second char of directive (B,W, or L)
0178    694 ;       R6      - field width, or -1 if none
0178    695 ;       R9      - output length remaining
0178    696 ;       R10     - output position pointer
0178    697 ;       R11     - next parameter pointer
0178    698 ;
0178    699 ; IMPLICIT INPUTS:
0178    700 ;
0178    701 ;       none
0178    702 ;
0178    703 ; OUTPUTS:
0178    704 ;
0178    705 ;       none
0178    706 ;
0178    707 ; IMPLICIT OUTPUTS:
0178    708 ;
0178    709 ;       none
0178    710 ;
0178    711 ; ROUTINE VALUE:
0178    712 ;
0178    713 ;       none
0178    714 ;
0178    715 ; SIDE EFFECTS:
0178    716 ;
0178    717 ;       none
0178    718 ;--
```

```
                                0178   720 ;
                                0178   721 ; The registers will be set up as follows
                                0178   722 ;
                                0178   723 ;       R0  - max digits to be output
                                0178   724 ;       R1  - 0 -> byte, 1 -> word , 2 -> long
                                0178   725 ;       R2  - value to be converted
                                0178   726 ;       R4  - conversion radix
                                0178   727 ;       R5  - sign indicator, 1 -> sign to be output, 0 otherwise
                                0178   728 ;       R7  - fill character, (blank, zero for !Z, or * on width too small
                                0178   729 ;               for decimal conversions)
                                0178   730 ;       R8  - total width of field to be output
                                0178   731 ;
                                0178   732
                                0178   733 CVTNUM:
                          38    BB    0178   734         PUSHR   #^M<R3,R4,R5>
                                017A   735
        00000024'EF     03    54    3A    017A   736         LOCC    R4,#3,DATA_TYPES        ; determine data type
                              03    12    0182   737         BNEQ    10$                     ; continue if legal directive
                            FF1E    31    0184   738         BRW     ILLEGAL                 ; else take error condition
                                0187   739 10$:
                51    03      50    C3    0187   740         SUBL3   R0,#3,R1                ; convert to index
                          52    8B    D0    018B   741         MOVL    (R11)+,R2               ; get next longword parameter
  52    52    0000002A'EF41     00    EF    018E   742         EXTZV   #0,FIELDS[R1],R2,R2     ; select proper field
                          55    D4    0198   743         CLRL    R5                      ; note unsigned
                       57   20    90    019A   744         MOVB    #^A/ /,R7               ; default fill char is blank
             54    00000038'EF43    9A    019D   745         MOVZBL  RADIX[R3],R4            ; get conversion radix
                                01A5   746
                                01A5   747 ;
                                01A5   748 ; Case on the type of conversion.  Note that base is set
                                01A5   749 ; so that octal and hex conversions fall through case table.
                                01A5   750 ;
                                01A5   751
                                01A5   752         CASE    R3,<40$,30$,20$>,.#2    ; base index of 2
          02'   02    53    AF    01A5           CASEW   R3,#2,S^#<<30005$-30004$>/2>-1
                                01A9       30004$:
                     003C'      01A9           .SIGNED_WORD    40$-30004$
                     0029'      01AB           .SIGNED_WORD    30$-30004$
                     0024'      01AD           .SIGNED_WORD    20$-30004$
                                01AF       30005$:
                                01AF   753
                                01AF   754 ;
                                01AF   755 ; Octal and Hex fall through here
                                01AF   756 ;
                                01AF   757
             50    6143    DE    01AF   758         MOVAL   (R1)[R3],R0             ; compute index in OCT_HEX_DIGITS
  50    00000031'EF40    9A    01B3   759         MOVZBL  OCT_HEX_DIGITS[R0],R0   ; get number of digits to output
                58    56    D0    01BB   760         MOVL    R6,R8                   ; user specified width?
                      03    18    01BE   761         BGEQ    15$                     ; yes, use it as width
                58    50    D0    01C0   762         MOVL    R0,R8                   ; else take needed space
                                01C3   763 15$:
                50    58    D1    01C3   764         CMPL    R8,R0                   ; width lss default digits?
                      47    18    01C6   765         BGEQ    60$                     ; no, fill to user specified width
                50    58    D0    01C8   766         MOVL    R8,R0                   ; else output only specified width
                      42    11    01CB   767         BRB     60$
                                01CD   768
                                01CD   769 ;
                                01CD   770 ; Unsigned decimal with zero fill
```

```
                                01CD    771 ;
                                01CD    772
                                01CD    773 20$:
                     57   30    90  01CD    774         MOVB    #^A/0/,R7               ; insert new fill char
                          13    11  01D0    775         BRB     40$                     ; continue with normal dec. code
                                01D2    776
                                01D2    777 ;
                                01D2    778 ; Signed decimal conversion
                                01D2    779 ;
                                01D2    780
                                01D2    781 30$:
 52   52   0000002A'EF41    00    EE  01D2    782         EXTV    #0,FIELDS[R1],R2,R2     ; sign extend the field
                   05 52    1F    E1  01DC    783         BBC     #31,R2,40$              ; not negative, continue
                            55    D6  01E0    784         INCL    R5                      ; else note that value negative
                     52     52    CE  01E2    785         MNEGL   R2,R2                   ; and make it positive
                                01E5    786
                                01E5    787 40$:                                        ; common decimal processing
                                01E5    788
                                01E5    789 ;
                                01E5    790 ; Determine the number of digits needed to print number in ASCII
                                01E5    791 ; decimal representation.
                                01E5    792 ;
                                01E5    793
                     50     01    D0  01E5    794         MOVL    #1,R0                   ; init digit counter
                     53     54    D0  01E8    795         MOVL    R4,R3                   ; copy first power of 10
                                01EB    796 44$:
                     53     52    D1  01EB    797         CMPL    R2,R3                   ; does it fit?
                            07    1F  01EE    798         BLSSU   48$                     ; yes, R0 has count if so
                     53     54    C4  01F0    799         MULL    R4,R3                   ; else compute next power of ten
                  F4 50     54    F2  01F3    800         AOBLSS  R4,R0,44$               ; continue (10 digits is largest possible)
                                01F7    801 48$:
               53   55     50    C1  01F7    802         ADDL3   R0,R5,R3                ; add in sign, if one exists
                    58     56    D0  01FB    803         MOVL    R6,R8                   ; did user specify width?
                            05    18  01FE    804         BGEQ    50$                     ; yes, use it for field width
                    58     53    D0  0200    805         MOVL    R3,R8                   ; else use amount needed
                            0A    11  0203    806         BRB     60$                     ; continue
                                0205    807 50$:
                    58     53    D1  0205    808         CMPL    R3,R8                   ; is there space within specified width?
                            05    15  0208    809         BLEQ    60$                     ; yes, go on
                    57     2A    90  020A    810         MOVB    #^A/*/,R7               ; no room, fill with stars
                            50    D4  020D    811         CLRL    R0                      ; output no digits
                                020F    812
                                020F    813 60$:
                  F8 AD     52    D0  020F    814         MOVL    R2,LASTVAL(FP)          ; remember value to be converted
                                0213    815
                                0213    816 ;
                                0213    817 ; Insert the ASCII representation for the value in R2 into the
                                0213    818 ; output buffer.
                                0213    819 ;
                                0213    820
                                0213    821 CVT_BIN_TO_ASC:
                                0213    822
                    0840 8F     BB  0213    823         PUSHR   #^M<R6,R11>             ; save work registers
                      04 A8     9F  0217    824         PUSHAB  4(R8)                   ; compute stack space needed for buffer
                    6E   03     CA  021A    825         BICL    #3,(SP)                 ; round stack to longword
                    5B   5E     D0  021D    826         MOVL    SP,R11                  ; save stack pointer
                    5E   6B     C2  0220    827         SUBL    (R11),SP                ; leave buffer space on stack
```

```
                                0223
                     53   D4    0223          CLRL    R3                      ; clear upper half of quad quotient
                51   01   CE    0225          MNEGL   #1,R1                   ; init digit counter for loop
                     0D   11    0228          BRB     15$                     ; start loop
                                022A   10$:
     56   52   52   54   7B    022A          EDIV    R4,R2,R2,R6             ; R2 <- quotient, R6 <- remainder
   7B   00000000'EF46   90    022F          MOVB    ASC_NAMES[R6],-(R11)    ; output ascii digit
                                0237   15$:
           EF 51   50   F2    0237          AOBLSS  R0,R1,10$               ; one more digit, done yet?
                05 55   E9    023B          BLBC    R5,20$                  ; branch if no sign to output
                7B   2D   90    023E          MOVB    #^A/-/,-(R11)           ; output sign
                     51   D6    0241          INCL    R1                      ;
                                0243   20$:
                                0243
                                0243   ;
                                0243   ; If field (R8) is not full, then fill remainder with the fill character
                                0243   ;
                                0243
                     03   11    0243          BRB     40$                     ; start the loop
                                0245   30$:
                7B   57   90    0245          MOVB    R7,-(R11)               ; insert fill character
                                0248   40$:
           F9 51   58   F3    0248          AOBLEQ  R8,R1,30$               ; fill until full
                                024C
                                024C   ;
                                024C   ; Now copy stack back to buffer, checking for overflow
                                024C   ;
                                024C
                     08   11    024C          BRB     70$                     ; start loop
                                024E   50$:
                02 59   F4    024E          SOBGEQ  R9,60$                  ; update length, check for overflow
                     21   11    0251          BRB     INSERT_OVF              ; handle overflow
                8A   8B   90    0253   60$:   MOVB    (R11)+,(R10)+           ; move char to output buffer
                F5 58   F4    0256   70$:   SOBGEQ  R8,50$                  ; move entire string
                                0259
                                0259   ;
                                0259   ; Now clean up mess on stack
                                0259   ;
                                0259
                5E   5B   D0    0259          MOVL    R11,SP                  ; restore stack
                0841 8F   BA    025C          POPR    #^M<R0,R6,R11>          ; remove top of stack and restore regs
                                0260
                                0260   ;
                                0260   ; Restore registers and return from service routine.
                                0260   ;
                                0260
                     38   BA    0260          POPR    #^M<R3,R4,R5>
                          05    0262          RSB
```

```
0263    877          .SBTTL  QUICKSERVE - Small service routines
0263    878
0263    879 ;++
0263    880 ;
0263    881 ; FUNCTIONAL DESCRIPTION:
0263    882 ;
0263    883 ;       Following are a collection of short service routines for
0263    884 ;       FAO directives.
0263    885 ;
0263    886 ; CALLING SEQUENCE:
0263    887 ;
0263    888 ;       JSB or BSB
0263    889 ;
0263    890 ; INPUTS:
0263    891 ;
0263    892 ;       R3  - index in CNTRL_TABLE of the directive
0263    893 ;       R4  - second character of two-char directive, if any
0263    894 ;       R6  - user specified field width, if any (ignored for singal char
0263    895 ;             and argument directives)
0263    896 ;       R9  - output length remaining
0263    897 ;       R10 - output position pointer
0263    898 ;
0263    899 ; IMPLICIT INPUTS:
0263    900 ;
0263    901 ;       none
0263    902 ;
0263    903 ; OUTPUTS:
0263    904 ;
0263    905 ;       none
0263    906 ;
0263    907 ; IMPLICIT OUTPUTS:
0263    908 ;
0263    909 ;       R9 and R10 are modified
0263    910 ;
0263    911 ; COMPLETION CODES:
0263    912 ;
0263    913 ;       none
0263    914 ;
0263    915 ; SIDE EFFECTS:
0263    916 ;
0263    917 ;       none
0263    918 ;--
```

```
                        0263   920
                        0263   921 INCR_ARGPTR:
                        0263   922 ;
                        0263   923 ; Directive to skip next parameter in parameter list
                        0263   924 ;
                        0263   925
                  8B    D5  0263   926         TSTL    (R11)+                  ; skip next parameter
                        05  0265   927         RSB                             ; exit
                        0266   928
                        0266   929 DECR_ARGPTR:
                        0266   930 ;
                        0266   931 ; Directive to back up and reuse last parameter in parameter list
                        0266   932 ;
                        0266   933
                  7B    D5  0266   934         TSTL    -(R11)                  ; back up argument pointer
                        05  0268   935         RSB                             ; exit
                        0269   936
                        0269   937 NEWLINE:
                        0269   938 ;
                        0269   939 ; Insert carriage return, line feed into output buffer
                        0269   940 ;
                        0269   941
               02 59    F4  0269   942         SOBGEQ  R9,10$                  ; room for CR?, branch if so
                  06    11  026C   943         BRB     INSERT_OVF              ; no room in output buffer
                        026E   944 10$:
            8A    0D    90  026E   945         MOVB    #CR,(R10)+              ; insert CR in output buffer
                        0271   946                                             ; continue for LF insertion
                        0271   947
                        0271   948 INSERT_CHAR:
                        0271   949 ;
                        0271   950 ; Make simple one character insertion in the output buffer.
                        0271   951 ;
                        0271   952
                 03 59  F4  0271   953         SOBGEQ  R9,INSERT_IT            ; check length, branch if ok
                        0274   954
                        0274   955 INSERT_OVF:
                  FE33  31  0274   956         BRW     OVERFLOW                ; error , no room in output buffer
                        0277   957
                        0277   958 INSERT_IT:
                        0277   959 ;
                        0277   960 ; Insert the character by computing the index into the replacement table
                        0277   961 ;
                        0277   962
      8A  00000021'EF43 90  0277   963         MOVB    REPLACEMENT-REPL_OFFSET[R3],(R10)+ ; insert the char
                        05  027F   964         RSB
                        0280   965
                        0280   966 ;
                        0280   967 ; Directive to repeat a particular character 'n' times, where 'n' is
                        0280   968 ; specified by the field width in the directive.
                        0280   969 ;
                        0280   970
                        0280   971 REPEATIT:
                     38 BB  0280   972         PUSHR   #^M<R3,R4,R5>           ; save regs for MOVC5 clobber
                     56 D5  0282   973         TSTL    R6                      ; check if width was specified
                     15 19  0284   974         BLSS    ILLFIELD                ; illegal if none specified
                  59 56 C2  0286   975         SUBL    R6,R9                   ; compute remaining output length
                     E9 19  0289   976         BLSS    INSERT_OVF              ; not enough room, error
```

```
                6A   56   54   6E   00   2C   028B    977          MOVC5   #0,(SP),R4,R6,(R10)     ; fill with specified character
                               5A   56   C0   0291    978          ADDL    R6,R10                  ; update output pointer
                                    38   BA   0294    979          POPR    #^M<R3,R4,R5>           ; restore regs
                                         05   0296    980          RSB
                                              0297    981
                                              0297    982 ;
                                              0297    984 ; The field width is specified with the define field directive.  At the
                                              0297    985 ; end field directive, any of the field remaining is blank filled, else
                                              0297    986 ; the field is truncated to the specified length.
                                              0297    987 ;
                                              0297    988
                                              0297    989 STARTFIELD:
                                         56   D5   0297    990          TSTL    R6                      ; did user specify field (must be specified)
                                         03   18   0299    991          BGEQ    STARTOK                 ; yes, continue
                                              029B    992
                                              029B    993 ILLFIELD:
                                    FE07  31   029B    994          BRW     ILLEGAL                 ; illegal directive
                                              029E    995
                                              029E    996 STARTOK:
                     FC AD   5A   56   C1   029E    997          ADDL3   R6,R10,FIELDEND(FP)      ; compute and save ending address
                                    59   56   D1   02A3    998          CMPL    R6,R9                   ; was that much space remaining?
                                         CC   14   02A6    999          BGTR    INSERT_OVF              ; no, take error here
                                              05   02A8   1000          RSB                             ; return
                                              02A9   1001
                                              02A9   1002 ;
                                              02A9   1003 ; Set up registers so that if fill is needed, a phony call is made
                                              02A9   1004 ; to REPEATIT with the length in R6 and the 'blank' character in R4
                                              02A9   1005 ;
                                              02A9   1006
                                              02A9   1007 ENDFIELD:
                                         54   20   9A   02A9   1008          MOVZBL  #^A/ /,R4               ; generate blank fill character
                          56   FC AD   5A   C3   02AC   1009          SUBL3   R10,FIELDEND(FP),R6     ; compute remaining field length
                                         CD   14   02B1   1010          BGTR    REPEATIT                ; if any left, go fill with blanks
                               5A   FC AD   D0   02B3   1011          MOVL    FIELDEND(FP),R10        ; else truncate by setting back pointer
                                    59   56   C2   02B7   1012          SUBL    R6,R9                   ; subtract negative difference from counter
                                              05   02BA   1013          RSB                             ; return
```

```
                     1015          .SBTTL  PERCENT - Time directives and plural 'S'
                     1016
                     1017 ;++
                     1018 ;
                     1019 ; FUNCTIONAL DESCRIPTION:
                     1020 ;
                     1021 ;       These directives are for date and time conversion, and for
                     1022 ;       conditionally inserting a plural 'S' into messages.
                     1023 ;       The time directives insert an ASCII time string into the output buffer.
                     1024 ;       The user may supply a quadword binary time to be converted,
                     1025 ;       or have the current date or time inserted.
                     1026 ;
                     1027 ; CALLING SEQUENCE:
                     1028 ;
                     1029 ;       JSB/BSB
                     1030 ;
                     1031 ; INPUTS:
                     1032 ;
                     1033 ;       R4  - second character of directive.  D -> convert
                     1034 ;               date and time, T -> convert time only
                     1035 ;               S -> plural indicator
                     1036 ;       R9  - remaining length of output buffer
                     1037 ;       R10 - current output buffer position
                     1038 ;       R11 - next parameter address
                     1039 ;
                     1040 ; IMPLICIT INPUTS:
                     1041 ;
                     1042 ;       none
                     1043 ;
                     1044 ; OUTPUTS:
                     1045 ;
                     1046 ;       none
                     1047 ;
                     1048 ; IMPLICIT OUTPUTS:
                     1049 ;
                     1050 ;       none
                     1051 ;
                     1052 ; ROUTINE VALUE:
                     1053 ;
                     1054 ;       none
                     1055 ;
                     1056 ; SIDE EFFECTS:
                     1057 ;
                     1058 ;       none
                     1059 ;--
```

```
                                  02BB  1061 PERCENT:
   00000027'EF   03   54   3A     02BB  1062          LOCC    R4,#3,PERCENT_STR        ; find directive type
                      D6   13     02C3  1063          BEQL    ILLFIELD                 ; illegal directive if not found
                      57   D4     02C5  1064          CLRL    R7                       ; assume date and time
                                  02C7  1065          CASE    R0,<10$,30$>,B,#2        ; branch on directive type
            01'   02   50   8F     02C7                CASEB   R0,#2,S^#<<30007$-30006$>/2>-1
                                  02CB        30006$:
                         0006'    02CB                .SIGNED_WORD    10$-30006$
                         003F'    02CD                .SIGNED_WORD    30$-30006$
                                  02CF        30007$:
                                  02CF  1066
                                  02CF  1067 ;
                                  02CF  1068 ; Time only directive falls through here
                                  02CF  1069 ;
                                  02CF  1070
                      57   D6     02CF  1071          INCL    R7                       ; indicate time only
                                  02D1  1072 10$:                                      ; time and date enters here
                      38   BB     02D1  1073          PUSHR   #^M<R3,R4,R5>            ; save registers
   6A   59   20   6A   00   2C     02D3  1074          MOVC5   #0,(R10),#^A/ /,R9,(R10) ; blank fill rest of output buffer
                  58   7E   DE     02D9  1075          MOVAL   -(SP),R8                 ; space for return length
                  7E   59   7D     02DC  1076          MOVQ    R9,-(SP)                 ; form descriptor for output buffer
                  52   6E   DE     02DF  1077          MOVAL   (SP),R2                  ; get address of buffer descriptor
                  51   8B   D0     02E2  1078          MOVL    (R11)+,R1                ; get binary time address
                                  02E5  1079 ;        BEQL    12$                      ; branch if no address                [01]
                                  02E5  1080 ;        CMPL    (R1),4(R1)               ; let potential access violation      [01]
                                  02E5  1081                                           ; ...happen in this frame rather than
                                  02E5  1082                                           ; ...within $ASCTIM to help condition
                                  02E5  1083                                           ; ...handler
                                  02E5  1084 12$:     $ASCTIM_S (R8),(R2),(R1),R7      ; convert time to ascii
                      57   DD     02E5                        PUSHL   R7
                      61   7F     02E7                        PUSHAQ  (R1)
                      62   7F     02E9                        PUSHAQ  (R2)
                      68   3F     02EB                        PUSHAW  (R8)
      00000000'GF     04   FB     02ED                        CALLS   #4,G^SYS$ASCTIM
                  52   56   D0     02F4  1085          MOVL    R6,R2                    ; did user specify width?
                      03   18     02F7  1086          BGEQ    20$                      ; yes, use it
                  52   68   3C     02F9  1087          MOVZWL  (R8),R2                  ; else use returned length
                                  02FC  1088 20$:
                  59   52   C2     02FC  1089          SUBL    R2,R9                    ; update output length
                      12   19     02FF  1090          BLSS    40$                      ; error, not enough room
                  5A   52   C0     0301  1091          ADDL    R2,R10                   ; update output buffer
                  5E   0C   C0     0304  1092          ADDL    #12,SP                   ; pop locals from stack
                      38   BA     0307  1093          POPR    #^M<R3,R4,R5>            ; restore registers
                           05     0309  1094          RSB                              ;
                                  030A  1095 30$:
                                  030A  1096
                                  030A  1097 ;
                                  030A  1098 ; Check if the last value converted was equal to one.  If so, then do
                                  030A  1099 ; nothing, else output an 'S' into the output buffer.
                                  030A  1100 ;
                                  030A  1101
            F8 AD   01   D1       030A  1102          CMPL    #1,LASTVAL(FP)           ; was last value a one
                      13   13     030E  1103          BEQL    60$                      ; yes, simply return
                 03 59   F4       0310  1104          SOBGEQ  R9,50$                   ; check if room in buffer
                  FD94   31       0313  1105 40$:     BRW     OVERFLOW                 ; no room , error
                                  0316  1106
            8A   53 8F   90       0316  1107 50$:     MOVB    #^A/S/,(R10)+            ; plural, insert 'S'
```

```
                04 FE AA    05    E1  031A  1108          BBC     #5,-2(R10),60$          ; continue if previous character was
                                      031F  1109                                          ; ...upper case
                   FF AA    20    88  031F  1110          BISB    #^X20,-1(R10)           ; else convert upper 'S' to lower 's'
                                      0323  1111
                            05    0323  1112 60$:       RSB                             ; return
```

```
0324  1114          .SBTTL  HANDLER - Condition handler
0324  1115
0324  1116 ;++
0324  1117 ;
0324  1118 ; FUNCTIONAL DESCRIPTION:
0324  1119 ;
0324  1120 ;       This condition handler is used to catch any errors which
0324  1121 ;       ocurred while processing the arguments, such as access
0324  1122 ;       violation.  This is because we don't want exceptions
0324  1123 ;       occurring within the system service.
0324  1124 ;       Care must be taken in this handler to deal with a second access
0324  1125 ;       violation while storing the return value for $FAO.
0324  1126 ;
0324  1127 ; INPUTS:
0324  1128 ;
0324  1129 ;       CHF$L_SIGARGLST(AP) = Address of signal vector
0324  1130 ;       CHF$L_MCHARGLST(AP) = Address of mechanism vector
0324  1131 ;
0324  1132 ; OUTPUTS:
0324  1133 ;
0324  1134 ;       The final R0 is set to the status code and the service
0324  1135 ;       is exited via $UNWIND.
0324  1136 ;--
```

```
                     0324  1138 ;HANDLER:
                     0324  1139 ;       .WORD   0                                                               [01]
                     0324  1140 ;                                                                               [01]
                     0324  1141 ;       MOVAB   L^EXE$SIGTORET,(FP)     ;Simple handler for errors here         [01]
                     0324  1142 ;                                                                               [01]
                     0324  1143 ;       ASSUME  CHF$L_MCHARGLST,EQ,CHF$L_SIGARGLST+4                            [01]
                     0324  1144 ;       MOVQ    CHF$L_SIGARGLST(AP),R0  ; Get address of signal argument list   [01]
                     0324  1145 ;       CMPL    #SS$_UNWIND,CHF$L_SIG_NAME(R0) ;Unwinding?                      [01]
                     0324  1146 ;       BEQL    90$                     ;Exit if yes                            [01]
                     0324  1147 ;       TSTL    CHF$L_MCH_DEPTH(R1)     ;Exception within FAO?                  [01]
                     0324  1148 ;       BNEQ    80$                     ;Resignal if no                         [01]
                     0324  1149 ;       MOVL    CHF$L_SIG_NAME(R0),CHF$L_MCH_SAVR0(R1) ;Set final return status [01]
                     0324  1150 ;       CLRQ    -(SP)                   ;Clear depth and new PC arguments       [01]
                     0324  1151 ;       CALLS   #2,G^SYS$UNWIND         ;Unwind to establisher's caller         [01]
                     0324  1152 ;                                       ;***** The next instruction my ACCVIO [01]
                     0324  1153 ;       MOVL    SF$L_SAVE_AP(FP),R0     ;Get address of FAO's argument list     [01]
                     0324  1154 ;       MOVL    OUTLEN(R0),R0           ;Output length requested?               [01]
                     0324  1155 ;       BEQL    10$                     ;Branch if not                          [01]
                     0324  1156 ;       CLRW    (R0)                    ;Indicate nothing returned in buffer    [01]
                     0324  1157 ;10$:                                    ;***** End of potential ACCVIO        [01]
                     0324  1158 ;80$:   MOVZWL  #SS$_RESIGNAL,R0         ;Resignal (ignore after UNWIND)         [01]
                     0324  1159 ;90$:   RET                              ;                                       [01]
                     0324  1160
                     0324  1161         .END
```

```
ARGCOUNT                = 00000000     D
ASC_NAMES                 00000000 R   D   02
CASE_BSB                  00000081 R   D   03
CASE_LOOP                 00000083 R   D   03
CNTRL_LENGTH            = 00000010     D
CNTRL_TABLE               00000010 R   D   02
CR                      = 0000000D     D
CVTASC                    00000104 R   D   03
CVTNUM                    00000178 R   D   03
CVT_BIN_TO_ASC            00000213 R   D   03
DATA_TYPES                00000024 R   D   02
DECR_ARGPTR               00000266 R   D   03
DONE                      000000B3 R   D   03
ENDFIELD                  000002A9 R   D   03
EXCL                    = 00000021     D
EXE$FAO                   00000000 RG  D   03
EXE$FAOL                  00000008 RG  D   03
FAO                       0000000E R   D   03
FAO_CASE                  00000088 R   D   03
FAO_EXIT                  000000B6 R   D   03
FF                      = 0000000C     D
FIELDEND                = FFFFFFFC     D
FIELDS                    0000002A R   D   02
FIRSTARG                = 00000010     D
GETCHAR                   000000C2 R   D   03
GETCOUNT                  000000CF R   D   03
ILLEGAL                   000000A5 R   D   03
ILLFIELD                  0000029B R   D   03
INCR_ARGPTR               00000263 R   D   03
INDSC                   = 00000004     D
INLEN                   = FFFFFFF0     D
INPTR                   = FFFFFFF4     D
INSERT_CHAR               00000271 R   D   03
INSERT_IT                 00000277 R   D   03
INSERT_OVF                00000274 R   D   03
LASTVAL                 = FFFFFFF8     D
LF                      = 0000000A     D
MAIN_SCAN                 0000001B R   D   03
NEWLINE                   00000269 R   D   03
OCT_HEX_DIGITS            00000031 R   D   02
ONECHAR_INDEX           = 00000008     D
ONE_CHAR_CNTRLS           00000018 R   D   02
OUTDSC                  = 0000000C     D
OUTLEN                  = 00000008     D
OVERFLOW                  000000AA R   D   03
PARSE_DIRECTIVE           00000046 R   D   03
PERCENT                   000002BB R   D   03
PERCENT_STR               00000027 R   D   02
RADIX                     00000038 R   D   02
REPEATIT                  00000280 R   D   03
REPLACEMENT               0000002D R   D   02
REPLACE_CHRS              0000001C R   D   02
REPL_OFFSET             = 0000000C     D
SS$_BADPARAM            = 00000014     D
SS$_BUFFEROVF           = 00000601     D
SS$_NORMAL              = 00000001     D
STARTFIELD                00000297 R   D   03
STARTOK                   0000029E R   D   03
STRING_TYPES              00000020 R   D   02
SYS$ASCTIM                ********  GX   03
TAB                     = 00000009     D
TWO_CHAR_CNTRLS           00000010 R   D   02
```

```
                                    +-----------------+
                                    ! Psect synopsis !
                                    +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
DATA                            0000003D  (   61.)  02 (  2.)  NOPIC   USR   CON   REL   LCL   SHR NOEXE   RD  NOWRT NOVEC LONG
CODE                            00000324  (  804.)  03 (  3.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ARGCOUNT	=00000000	102 (1)	
ASC_NAMES	00000000-R	126 (1)	#-834 (1)
CASE_BSB	00000081-R	358 (1)	#-361 (1)
CASE_LOOP	00000083-R	360 (1)	#-357 (1)
CNTRL_LENGTH	=00000010	166 (1)	168 (1) #-339 (1) #-350 (1)
CNTRL_TABLE	00000010-R	143 (1)	166 (1) 168 (1) 170 (1) 339 (1)
CR	=0000000D	113 (1)	#-945 (1)
CVTASC	00000104-R	576 (1)	387 (1)
CVTNUM	00000178-R	733 (1)	#-389 (1)
CVT_BIN_TO_ASC	00000213-R	821 (1)	
DATA_TYPES	00000024-R	174 (1)	736 (1)
DECR_ARGPTR	00000266-R	929 (1)	387 (1)
DONE	000000B3-R	401 (1)	#-310 (1)
ENDFIELD	000002A9-R	1007 (1)	387 (1)
EXCL	=00000021	115 (1)	181 (1) #-301 (1)
EXE$FAO	00000000-R	270 (1)	
EXE$FAOL	00000008-R	281 (1)	
FAO	0000000E-R	288 (1)	#-275 (1)
FAO_CASE	00000088-R	374 (1)	#-359 (1)
FAO_EXIT	000000B6-R	404 (1)	#-394 (1) #-399 (1)
FF	=0000000C	117 (1)	181 (1)
FIELDEND	=FFFFFFFC	111 (1)	#-1009 (1) #-1011 (1) #-997 (1)
FIELDS	0000002A-R	178 (1)	#-742 (1) #-782 (1)
FIRSTARG	=00000010	106 (1)	274 (1) #-285 (1)
GETCHAR	000000C2-R	451 (1)	#-313 (1) #-328 (1) #-337 (1) #-343 (1)
			#-347 (1) #-514 (1) #-525 (1)
GETCOUNT	000000CF-R	500 (1)	#-327 (1) #-336 (1)
ILLEGAL	000000A5-R	392 (1)	#-340 (1) #-453 (1) #-664 (1) #-738 (1)
			#-994 (1)
ILLFIELD	0000029B-R	993 (1)	#-1063 (1) #-974 (1)
INCR_ARGPTR	00000263-R	921 (1)	387 (1)
INDSC	=00000004	103 (1)	#-290 (1)
INLEN	=FFFFFFF0	108 (1)	#-301 (1) #-305 (1) #-306 (1) #-452 (1)
INPTR	=FFFFFFF4	109 (1)	301 (1) 309 (1) #-311 (1) #-454 (1)
			#-455 (1) #-502 (1) #-506 (1) #-508 (1)
			#-517 (1)
INSERT_CHAR	00000271-R	948 (1)	387 (1)
INSERT_IT	00000277-R	958 (1)	#-953 (1)
INSERT_OVF	00000274-R	955 (1)	#-859 (1) #-943 (1) #-976 (1) #-999 (1)
LASTVAL	=FFFFFFF8	110 (1)	#-1102 (1) #-814 (1)
LF	=0000000A	114 (1)	181 (1)
MAIN_SCAN	0000001B-R	299 (1)	#-362 (1)
NEWLINE	00000269-R	937 (1)	387 (1)
OCT_HEX_DIGITS	00000031-R	190 (1)	#-759 (1)
ONECHAR_INDEX	=00000008	168 (1)	#-341 (1)
ONE_CHAR_CNTRLS	00000018-R	154 (1)	168 (1)
OUTDSC	=0000000C	105 (1)	#-291 (1) #-407 (1)
OUTLEN	=00000008	104 (1)	#-405 (1) #-407 (1)
OVERFLOW	000000AA-R	396 (1)	#-1105 (1) #-308 (1) #-662 (1) #-956 (1)
PARSE_DIRECTIVE	00000046-R	323 (1)	

```
PERCENT                          000002BB-R   1061    (1)       387    (1)
PERCENT_STR                      00000027-R   176     (1)       1062   (1)
RADIX                            00000038-R   194     (1)     #-745    (1)
REPEATIT                         00000280-R   971     (1)     #-1010   (1)       387    (1)
REPLACEMENT                      0000002D-R   180     (1)     #-963    (1)
REPLACE_CHRS                     0000001C-R   160     (1)       170    (1)
REPL_OFFSET                     =0000000C     170     (1)     #-963    (1)
SS$_BADPARAM                    =00000014                     #-393    (1)
SS$_BUFFEROVF                   =00000601                     #-397    (1)
SS$_NORMAL                      =00000001                     #-402    (1)
STARTFIELD                       00000297-R   989     (1)       387    (1)
STARTOK                          0000029E-R   996     (1)     #-991    (1)
STRING_TYPES                     00000020-R   172     (1)       579    (1)
SYS$ASCTIM                       00000000-XR                    1084   (1)
TAB                             =00000009     116     (1)       181    (1)
TWO_CHAR_CNTRLS                  00000010-R   144     (1)
```

```
                                        +-------------------------+
                                        ! Macros Cross Reference !
                                        +-------------------------+

MACRO            SIZE          DEFINITION       REFERENCES...
-----            ----          ----------       -------------
$ASCTIM_S        1             1084    (1)      1084    (1)
$DEFINI          1             94      (1)      94      (1)
$PUSHADR         1             1084    (1)      1084    (1)
$SSDEF           21            94      (1)      94      (1)
CASE             1             375     (1)      1065    (1)      375     (1)      585     (1)      752     (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	1091 (1) 1092 (1) 513 (1) 642 (1) 978 (1)
ADDL3	00C1	802 (1) 997 (1)
AOBLEQ	00F3	653 (1) 850 (1)
AOBLSS	00F2	800 (1) 836 (1)
BBC	00E1	1108 (1) 783 (1)
BEQL	0013	1063 (1) 1103 (1) 302 (1) 340 (1) 406 (1) 503 (1) 518 (1)
		580 (1)
BGEQ	0018	1086 (1) 622 (1) 761 (1) 765 (1) 804 (1) 991 (1)
BGTR	0014	1010 (1) 999 (1)
BICL	00CA	825 (1)
BISB	0088	1110 (1)
BLBC	00E9	310 (1) 346 (1) 643 (1) 837 (1)
BLEQ	0015	342 (1) 648 (1) 809 (1)
BLSS	0019	1090 (1) 308 (1) 333 (1) 453 (1) 509 (1) 511 (1) 637 (1)
		646 (1) 657 (1) 974 (1) 976 (1)
BLSSU	001F	798 (1)
BNEQ	0012	330 (1) 737 (1)
BRB	0011	275 (1) 357 (1) 362 (1) 394 (1) 399 (1) 515 (1) 595 (1)
		604 (1) 767 (1) 775 (1) 806 (1) 831 (1) 846 (1) 856 (1)
		859 (1) 943 (1)
BRW	0031	1105 (1) 389 (1) 662 (1) 664 (1) 738 (1) 956 (1) 994 (1)
BSBB	0010	313 (1) 328 (1) 336 (1) 337 (1) 343 (1) 347 (1) 359 (1)
		514 (1) 525 (1)
BSBW	0030	327 (1)
CALLS	00FB	1084 (1)
CASEB	008F	1065 (1) 387 (1) 585 (1)
CASEW	00AF	752 (1)
CLRL	00D4	1064 (1) 300 (1) 326 (1) 398 (1) 578 (1) 743 (1) 811 (1)
		829 (1)
CLRQ	007C	289 (1) 504 (1)
CMPB	0091	329 (1) 502 (1) 510 (1) 645 (1) 647 (1)
CMPL	00D1	1102 (1) 341 (1) 517 (1) 764 (1) 797 (1) 808 (1) 998 (1)
DECW	00B7	452 (1)
EDIV	007B	833 (1)
EXTV	00EE	782 (1)
EXTZV	00EF	742 (1)
INCL	00D6	1071 (1) 303 (1) 331 (1) 455 (1) 592 (1) 652 (1) 784 (1)
		839 (1)
LOCC	003A	1062 (1) 301 (1) 339 (1) 579 (1) 736 (1)
MNEGL	00CE	501 (1) 785 (1) 830 (1)
MOVAL	00DE	1075 (1) 1077 (1) 274 (1) 758 (1)
MOVB	0090	1107 (1) 650 (1) 744 (1) 774 (1) 810 (1) 834 (1) 838 (1)
		848 (1) 860 (1) 945 (1) 963 (1)
MOVC3	0028	309 (1)
MOVC5	002C	1074 (1) 639 (1) 977 (1)
MOVL	00D0	1011 (1) 1078 (1) 1085 (1) 285 (1) 306 (1) 311 (1) 312 (1)
		325 (1) 334 (1) 344 (1) 506 (1) 519 (1) 524 (1) 611 (1)
		621 (1) 623 (1) 634 (1) 638 (1) 641 (1) 741 (1) 760 (1)
		762 (1) 766 (1) 794 (1) 795 (1) 803 (1) 805 (1) 814 (1)
		826 (1) 867 (1)

```
MOVQ            007D    1076  (1)   290  (1)   291  (1)   594  (1)   602  (1)
MOVZBL          009A    1008  (1)   454  (1)   612  (1)   745  (1)   759  (1)
MOVZWL          003C    1087  (1)   292  (1)   393  (1)   397  (1)   402  (1)   603  (1)
MULL            00C4     799  (1)
MULL2           00C4     512  (1)
POPR            00BA    1093  (1)   658  (1)   868  (1)   874  (1)   979  (1)
PUSHAB          009F     824  (1)
PUSHAQ          007F    1084  (1)
PUSHAW          003F    1084  (1)
PUSHL           00DD    1084  (1)
PUSHR           00BB    1073  (1)   577  (1)   734  (1)   823  (1)   972  (1)
RET             0004     408  (1)
RSB             0005    1000  (1)  1013  (1)  1094  (1)  1112  (1)   456  (1)   521  (1)   526  (1)
                         659  (1)   875  (1)   927  (1)   935  (1)   964  (1)   980  (1)
SOBGEQ          00F4    1104  (1)   361  (1)   858  (1)   861  (1)   942  (1)   953  (1)
SUBB3           0083     508  (1)
SUBL            00C2    1012  (1)  1089  (1)   636  (1)   827  (1)   975  (1)
SUBL3           00C3    1009  (1)   350  (1)   740  (1)
SUBW            00A2     307  (1)
SUBW3           00A3     305  (1)   407  (1)
TSTL            00D5     332  (1)   405  (1)   656  (1)   926  (1)   934  (1)   973  (1)   990  (1)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ASCII          127    (1)    145    (1)    146    (1)    147    (1)    148    (1)    149    (1)    150    (1)    151    (1)
                152    (1)    155    (1)    156    (1)    157    (1)    158    (1)    161    (1)    162    (1)    163    (1)
                164    (1)    173    (1)    175    (1)    177    (1)
.BYTE           179    (1)    181    (1)    191    (1)    192    (1)    195    (1)
.END            1161   (1)
.ENDC           1084   (1)
.ENDM           1084   (1)    375    (1)    94     (1)
.ENDR           1065   (1)    387    (1)    585    (1)    752    (1)
.GLOBL          1084   (1)
.IDENT          5      (1)
.IF             1084   (1)
.IFF            1084   (1)
.IRP            1065   (1)    387    (1)    585    (1)    752    (1)
.LIST           529    (1)
.NOCROSS        94     (1)
.PAGE           1014   (1)    1060   (1)    1113   (1)    1137   (1)    118    (1)    196    (1)    250    (1)    286    (1)
                314    (1)    363    (1)    409    (1)    450    (1)    457    (1)    498    (1)    527    (1)    574    (1)
                665    (1)    719    (1)    87     (1)    876    (1)    919    (1)
.PSECT          124    (1)    198    (1)
.RESTORE        94     (1)
.SAVE           94     (1)
.SBTTL          1015   (1)    1114   (1)    197    (1)    410    (1)    458    (1)    528    (1)    666    (1)    877    (1)
                88     (1)
.SIGNED_WORD    1065   (1)    387    (1)    585    (1)    752    (1)
.TITLE          4      (1)
.WORD           272    (1)    283    (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	7	274	(1)	#-285	(1)	#-290	(1)	#-291	(1)	#-405	(1)	#-407	(1)
FP	18	#-1009	(1)	#-1011	(1)	#-1102	(1)	#-301	(1)	#-305	(1)	#-306	(1)
		309	(1)	#-311	(1)	#-452	(1)	#-454	(1)	#-455	(1)	#-502	(1)
		#-506	(1)	#-508	(1)	#-517	(1)	#-814	(1)	#-997	(1)		
R0	22	#-1065	(1)	#-305	(1)	#-306	(1)	#-341	(1)	#-350	(1)	#-393	(1)
		#-397	(1)	#-402	(1)	#-585	(1)	#-740	(1)	#-758	(1)	#-759	(1)
		#-762	(1)	#-764	(1)	#-766	(1)	#-794	(1)	#-800	(1)	#-802	(1)
		#-811	(1)	#-836	(1)	#-868	(1)						
R1	20	#-1078	(1)	1084	(1)	#-311	(1)	#-506	(1)	#-517	(1)	#-594	(1)
		#-602	(1)	#-603	(1)	#-612	(1)	#-623	(1)	#-639	(1)	#-740	(1)
		#-742	(1)	758	(1)	#-782	(1)	#-830	(1)	#-836	(1)	#-839	(1)
		#-850	(1)										
R10	21	#-1009	(1)	#-1011	(1)	1074	(1)	#-1091	(1)	#-1107	(1)	1108	(1)
		#-1110	(1)	272	(1)	283	(1)	309	(1)	#-312	(1)	639	(1)
		#-641	(1)	#-642	(1)	#-860	(1)	#-945	(1)	#-963	(1)	977	(1)
		#-978	(1)	#-997	(1)								
R11	21	#-1078	(1)	272	(1)	#-274	(1)	283	(1)	#-285	(1)	#-524	(1)
		#-594	(1)	#-602	(1)	#-611	(1)	#-741	(1)	#-823	(1)	#-826	(1)
		#-827	(1)	#-834	(1)	#-838	(1)	#-848	(1)	#-860	(1)	#-867	(1)
		#-868	(1)	#-926	(1)	#-934	(1)						
R2	31	#-1077	(1)	1084	(1)	#-1085	(1)	#-1087	(1)	#-1089	(1)	#-1091	(1)
		272	(1)	283	(1)	#-326	(1)	#-331	(1)	#-346	(1)	#-611	(1)
		#-612	(1)	639	(1)	#-641	(1)	#-645	(1)	#-647	(1)	#-650	(1)
		#-652	(1)	#-741	(1)	742	(1)	782	(1)	783	(1)	#-785	(1)
		#-797	(1)	#-814	(1)	#-833	(1)						
R3	32	#-1073	(1)	#-1093	(1)	272	(1)	283	(1)	#-312	(1)	#-329	(1)
		#-339	(1)	#-344	(1)	#-350	(1)	#-387	(1)	#-454	(1)	#-504	(1)
		#-508	(1)	#-510	(1)	#-513	(1)	#-577	(1)	#-658	(1)	#-734	(1)
		#-745	(1)	#-752	(1)	758	(1)	#-795	(1)	#-797	(1)	#-799	(1)
		#-802	(1)	#-805	(1)	#-808	(1)	#-829	(1)	#-874	(1)	#-963	(1)
		#-972	(1)	#-979	(1)								
R4	24	#-1008	(1)	#-1062	(1)	#-1073	(1)	#-1093	(1)	272	(1)	283	(1)
		#-344	(1)	#-512	(1)	#-513	(1)	#-519	(1)	#-577	(1)	#-579	(1)
		#-658	(1)	#-734	(1)	#-736	(1)	#-745	(1)	#-795	(1)	#-799	(1)
		#-800	(1)	#-833	(1)	#-874	(1)	#-972	(1)	#-977	(1)	#-979	(1)
R5	17	#-1073	(1)	#-1093	(1)	272	(1)	283	(1)	#-325	(1)	#-334	(1)
		#-361	(1)	#-577	(1)	#-658	(1)	#-734	(1)	#-743	(1)	#-784	(1)
		#-802	(1)	#-837	(1)	#-874	(1)	#-972	(1)	#-979	(1)		
R6	34	#-1009	(1)	#-1012	(1)	#-1085	(1)	272	(1)	283	(1)	#-305	(1)
		#-307	(1)	#-309	(1)	#-332	(1)	#-334	(1)	#-501	(1)	#-519	(1)
		#-524	(1)	#-577	(1)	#-621	(1)	#-634	(1)	#-638	(1)	#-639	(1)
		#-642	(1)	#-653	(1)	#-658	(1)	#-760	(1)	#-803	(1)	#-823	(1)
		#-833	(1)	#-834	(1)	#-868	(1)	#-973	(1)	#-975	(1)	#-977	(1)
		#-978	(1)	#-990	(1)	#-997	(1)	#-998	(1)				
R7	13	#-1064	(1)	#-1071	(1)	#-1084	(1)	272	(1)	283	(1)	#-578	(1)
		#-592	(1)	#-643	(1)	#-653	(1)	#-744	(1)	#-774	(1)	#-810	(1)
		#-848	(1)										
R8	19	#-1075	(1)	1084	(1)	#-1087	(1)	272	(1)	283	(1)	#-621	(1)
		#-623	(1)	#-636	(1)	#-638	(1)	#-760	(1)	#-762	(1)	#-764	(1)
		#-766	(1)	#-803	(1)	#-805	(1)	#-808	(1)	824	(1)	#-850	(1)

```
                    #-861   (1)
R9          21      #-1012  (1)   #-1074  (1)   #-1076  (1)   #-1089  (1)   #-1104  (1)     272   (1)
                      283   (1)   #-291   (1)   #-292   (1)   #-307   (1)   #-398   (1)   #-407   (1)
                    #-634   (1)   #-636   (1)   #-656   (1)   #-858   (1)   #-942   (1)   #-953   (1)
                    #-975   (1)   #-998   (1)
SP          14        1075  (1)   #-1076  (1)     1077  (1)   #-1092  (1)   #-289   (1)   #-290   (1)
                    #-300   (1)   #-303   (1)   #-310   (1)   #-825   (1)   #-826   (1)   #-827   (1)
                    #-867   (1)     977   (1)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.03	00:00:00.29
Command processing	891	00:00:00.22	00:00:00.67
Pass 1	509	00:00:01.35	00:00:02.18
Symbol table sort	0	00:00:00.15	00:00:00.15
Pass 2	52	00:00:00.67	00:00:01.30
Symbol table output	2	00:00:00.01	00:00:00.01
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	7	00:00:00.14	00:00:00.20
Assembler run totals	1490	00:00:02.58	00:00:04.81

```
The working set limit was 1900 pages.
58153 bytes (114 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 469 non-local and 55 local symbols.
1161 source lines were read in Pass 1, producing 42 object records in Pass 2.
12 pages of virtual memory were used to define 10 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	7

```
492 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:FAO.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DIA
```

Table of contents

(1) 90 Libraries, Macros, Equated Symbols
(1) 121 Data Psect Declarations
(1) 143 VRSetFlg Routine - Set control flags
(1) 200 VRClrFlg Routine - Clear control flags
(1) 248 VRSetEF Routine - Set event flags
(1) 298 VRClrEF Routine - Clear event flags
(1) 348 VRShowFlg Routine - Show control flags
(1) 418 VRShowEF Routine - Show event flags

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element FLAGS.MAR
0000     2 ; *1     6-FEB-1986 15:18:35 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element FLAGS.MAR
0000     4         .TITLE  FLAGS   *** FLAGS Set/Clear/Show flags
0000     5         .IDENT  /07-14/
0000     6         .NoShow Conditionals    ;                                                    [11]
0000     7
0000     8 ;++
0000     9 ; Copyright (c) 1977, 1982, 1985
0000    10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    11 ;
0000    12 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    13 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    14 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    15 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    16 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    17 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    18 ; REMAIN IN DEC.
0000    19 ;
0000    20 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    21 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    22 ; CORPORATION.
0000    23 ;
0000    24 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    26 ;--
```

```
                        0000     28 ;++
                        0000     29 ; FACILITY:
                        0000     30 ;
                        0000     31 ;       VAX DIAGNOSTIC SUPERVISOR
                        0000     32 ;
                        0000     33 ; ABSTRACT:
                        0000     34 ;
                        0000     35 ;       Set/clear/show control and event flags.
                        0000     36 ;
                        0000     37 ; ENVIRONMENT:
                        0000     38 ;
                        0000     39 ; AUTHOR:
                        0000     40 ;
                        0000     41 ;       TOM SOUTTER     17-NOV-77       VERSION 01
                        0000     42 ;
                        0000     43 ; MODIFIED BY:
                        0000     44 ;       01      TOM SOUTTER     21-NOV-77       VERSION 02
                        0000     45 ;               DSPR #14 - DEFAULT FLAGS SETTING.
                        0000     46 ;
                        0000     47 ;       02      KEN CHAPMAN     05-DEC-77       VERSION 03 (ESSAA-3.05)
                        0000     48 ;               DSPR #45 - REMOVED HALTI.
                        0000     49 ;
                        0000     50 ;       03      KEN CHAPMAN     09-JAN-78       VERSION 04 (ESSAA-3.06)
                        0000     51 ;               DSPR #14 - DEFAULT FLAG SETTING (AND CLEARING).
                        0000     52 ;       04      SET AND CLEAR FLAGS ALL SHOULDN'T MODIFY APT FLAGS.
                        0000     53 ;
                        0000     54 ;       05      KEN CHAPMAN     19-JAN-78       VERSION 05 (ESSAA-3.07)
                        0000     55 ;               FIXED SET AND CLEAR FLAGS ALL BY DEFINING DSA$M_ALLFLAGS.
                        0000     56 ;
                        0000     57 ;       06      Dave Butenhof   15-feb-80
                        0000     58 ;               Include SEARCH flag
                        0000     59 ;
                        0000     60 ;       07      Dave Butenhof   30-apr-80
                        0000     61 ;               Change DSA$GL_FLAGS references from word relative to long
                        0000     62 ;               relative
                        0000     63 ;
                        0000     64 ;       08      - Jack Stansbury, 22-Oct-1981, Version 6.-
                        0000     65 ;               Fixed truncation errors by changing remaining W^'s to L^.
                        0000     66 ;               Also added .LIBRARY statements for $DS and $DIAG.
                        0000     67 ;
                        0000     68 ;       09      - Jack Stansbury, 4-Nov-1981, Version 6.-
                        0000     69 ;               Took out all references to the following:
                        0000     70 ;               LOCK, SPOOL, COMPAT, ENSPEC, HALTD, HALTI, SUBMIT, LOG, and
                        0000     71 ;               CONSOLE. These flags are no longer used anywhere. I changed
                        0000     72 ;               the HALTD and HALTI flags to just plain HALT.
                        0000     73 ;
                        0000     74 ;       10      - Jack Stansbury, 12-Nov-1981, Version 6.-
                        0000     75 ;               Added the VERIFY and BINARY flags.
                        0000     76 ;
                        0000     77 ;       11      - Jack Stansbury, 25-Mar-1982, Version 6.?
                        0000     78 ;               Added typecoding to the print statements.
                        0000     79 ;
                        0000     80 ;       [12]    Dave Butenhof   07-May-1982     Version 6.8
                        0000     81 ;               Fix truncation error
                        0000     82 ;
                        0000     83 ;       [13]    Marion Baggett  16-June-1982    Version 6.8
                        0000     84 ;               Show HALT  flag if cleared.
```

```
                        0000     85 ;
                        0000     86 ;       [14]    Amy Iorio,      19-July-1985    Version 8.3
                        0000     87 ;               Show ASKPROMPT flag, cleared and set.
                        0000     88 ;--
```

```
                        0000     90          .SBTTL  Libraries, Macros, Equated Symbols
                        0000     91 ;
                        0000     92 ; INCLUDE FILES:
                        0000     93 ;
                        0000     94          .Library        /Sys$Library:Lib/       ;                       [11]
                        0000     95          .Library        /$DS/                   ;                       [08]
                        0000     96          .Library        /$DIAG/                 ;                       [08]
                        0000     97
                        0000     98 ;
                        0000     99 ; MACROS:
                        0000    100 ;
                        0000    101
                        0000    102 ;
                        0000    103 ; EQUATED SYMBOLS:
                        0000    104 ;
                        0000    105
                        0000    106          $DS_DSADEF                      ; CONTROL FLAG DEFINITIONS
                        0000    107          CLIDEF                          ; CLI COMMAND BLOCK OFFSETS
                        0000    108          ENVDEF                          ; ENVIRONMENT BIT DEFINITIONS
```

```
                 0000   109        $DS_TypeDef                    ; Define TypeCodes                    [11]
                 0000   110
        00000002 0000   111        $ENV            = 1@V_ENV      ; FORCE SS CALLS TO SS VECTORS
                 0000   112
        00000013 0000   113        LEFTMOSTFLAG    = DSA$V_ASK_PROMPT; MSB for CVTREG function          [13]
                 0000   114
                 0000   115        DSA$M_ALLFLAGS  = DSA$M_HALT  ! DSA$M_LOOP    ! DSA$M_BELL  -   ;    [10]
                 0000   116                        ! DSA$M_IE1   ! DSA$M_IE2     ! DSA$M_IE3   -   ;    [09]
                 0000   117                        ! DSA$M_QUICK ! DSA$M_VERIFY  ! DSA$M_TRACE -   ;    [10]
                 0000   118                        ! DSA$M_OPER  ! DSA$M_PROMPT  ! DSA$M_IES   -   ;    [09]
        000037FD 0000   119        ;               ! DSA$M_ASK_PROMPT                              ;    [13]
```

```
                                        0000   121          .Subtitle       Data Psect Declarations
                                        0000   122 ;
                                        0000   123 ; OWN STORAGE:
                                        0000   124 ;
                                        0000   125
                                    00000000   126          .Psect  Data, Shr, NoExe, NoWrt, Byte
                                        0000   127
                                        0000   128 FMTSHOWFLGS:
66 20 6C 6F 72 74 6E 6F 43 2F 21 00'    0000   129          .ASCIC  .!/Control flags  set : !AC.
20 3A 20 74 65 73 20 20 73 67 61 6C     000C
                              43 41 21  0018
                                    1A  0000
                                        001B   130
                                        001B   131 FMTSHOWFLGC:
66 20 6C 6F 72 74 6E 6F 43 2F 21 00'    001B   132          .ASCIC  .!/Control flags clear: !AC!/.
20 3A 72 61 65 6C 63 20 73 67 61 6C     0027
                        2F 21 43 41 21  0033
                                    1C  001B
                                        0038   133
                                        0038   134 FMTSHOWEF:
61 6C 66 20 74 6E 65 76 45 2F 21 00'    0038   135          .ASCIC  .!/Event flags set: !AC!/.
21 43 41 21 20 3A 74 65 73 20 73 67     0044
                                    2F  0050
                                    18  0038
                                        0051   136
                                        0051   137 AFLAGMNE:
2C 74 70 6D 6F 72 50 5F 6B 73 41 00'    0051   138          .ASCIC  .Ask_Prompt,,,,,Search,Prompt,Oper,,Trace,Verify,Quick,.-       ;
50 2C 68 63 72 61 65 53 2C 2C 2C 2C     005D
2C 2C 72 65 70 4F 2C 74 70 6D 6F 72     0069
79 66 69 72 65 56 2C 65 63 61 72 54     0075
                        2C 6B 63 69 75  0081
2C 32 45 49 2C 33 45 49 2C 53 45 49     0088   139                  .IES,IE3,IE2,IE1,Bell,Loop,Binary,Halt.                         ;
6F 6F 4C 2C 6C 6C 65 42 2C 31 45 49     0094
6C 61 48 2C 79 72 61 6E 69 42 2C 70     00A0
                                    74  00AC
                                    5B  0051
                                        00AD   140 AEFNMNE:
30 32 2C 31 32 2C 32 32 2C 33 32 00'    00AD   141          .ASCIC  .23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1.
36 31 2C 37 31 2C 38 31 2C 39 31 2C     00B9
32 31 2C 33 31 2C 34 31 2C 35 31 2C     00C5
37 2C 38 2C 39 2C 30 31 2C 31 31 2C     00D1
31 2C 32 2C 33 2C 34 2C 35 2C 36 2C     00DD
                                    3B  00AD
```

```
                     00E9   143          .SBTTL  VRSetFlg Routine - Set control flags
                 00000000   144          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0000   145 ;++
                     0000   146 ; FUNCTIONAL DESCRIPTION:
                     0000   147 ;
                     0000   148 ;       SETS SPECIFIED OPERATOR CONTROL FLAGS
                     0000   149 ;
                     0000   150 ; CALLING SEQUENCE:
                     0000   151 ;
                     0000   152 ;       BSBW    VRSETFLG
                     0000   153 ;
                     0000   154 ; INPUT PARAMETERS:
                     0000   155 ;
                     0000   156 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
                     0000   157 ;
                     0000   158 ; IMPLICIT INPUTS:
                     0000   159 ;
                     0000   160 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
                     0000   161 ;
                     0000   162 ; OUTPUT PARAMETERS:
                     0000   163 ;
                     0000   164 ;       NONE
                     0000   165 ;
                     0000   166 ; IMPLICIT OUTPUTS:
                     0000   167 ;
                     0000   168 ;       NONE
                     0000   169 ;
                     0000   170 ; COMPLETION CODES:
                     0000   171 ;
                     0000   172 ;       NONE
                     0000   173 ;
                     0000   174 ; SIDE EFFECTS:
                     0000   175 ;
                     0000   176 ;       NONE
                     0000   177 ;
                     0000   178 ;--
```

```
                                  0000  VRSETFLG::                                  ;
              0B 62    0C    E1   0000          BBC     #CLI$V_DEFAULT, -          ; SET TO DEFAULT ?
                                  0004                    CLI$L_FLAGS(R2), 10$
   0000FE00'EF    0000'8F    B0   0004          MOVW    #DSA$M_DFLTFLGS, -         ; Yes, plug in default
                                  000D                    L^DSA$GL_FLAGS
                       15    11   000D          BRB     VRSETFLG_X
                                  000F
                    1C A2    D5   000F  10$:    TSTL    CLI$L_DATA(R2)             ; CHECK FOR ALL.
                       08    18   0012          BGEQ    20$                        ; SKIP IF NOT.
                                  0014  ;       MOVZWL  #DSA$M_ALLFLAGS, -         ; GET "ALL" FLAGS.
                                  0014  ;                 CLI$L_DATA(R2)
      1C A2    000037FD 8F    D0   0014          MOVL    #DSA$M_ALLFLAGS, -         ; GET "ALL" FLAGS.
                                  001C                    CLI$L_DATA(R2)
                                  001C
   0000FE00'EF    1C A2    C8   001C  20$:    BISL2   CLI$L_DATA(R2), -          ; No, set the specified flags
                                  0024                    L^DSA$GL_FLAGS
                                  0024
                                  0024  VRSETFLG_X:
                          05    0024          RSB                                ; RETURN
```

```
0025    200          .SBTTL  VRClrFlg Routine - Clear control flags
0025    201 ;++
0025    202 ; FUNCTIONAL DESCRIPTION:
0025    203 ;
0025    204 ;       CLEARS SPECIFIED OPERATOR CONTROL FLAGS
0025    205 ;
0025    206 ; CALLING SEQUENCE:
0025    207 ;
0025    208 ;       BSBW    VRCLRFLG
0025    209 ;
0025    210 ; INPUT PARAMETERS:
0025    211 ;
0025    212 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
0025    213 ;
0025    214 ; IMPLICIT INPUTS:
0025    215 ;
0025    216 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
0025    217 ;
0025    218 ; OUTPUT PARAMETERS:
0025    219 ;
0025    220 ;       NONE
0025    221 ;
0025    222 ; IMPLICIT OUTPUTS:
0025    223 ;
0025    224 ;       NONE
0025    225 ;
0025    226 ; COMPLETION CODES:
0025    227 ;
0025    228 ;       NONE
0025    229 ;
0025    230 ; SIDE EFFECTS:
0025    231 ;
0025    232 ;       NONE
0025    233 ;
0025    234 ;--
```

```
                                0025   236 VRCLRFLG::                                  ;
                    1C A2   D5  0025   237         TSTL    CLI$L_DATA(R2)             ; CHECK FOR "ALL".
                       08   18  0028   238         BGEQ    10$                        ; BR IF NOT.
                                002A   239 ;       MOVZWL  #DSA$M_ALLFLAGS, -         ; GET "ALL" FLAGS.
                                002A   240 ;                 CLI$L_DATA(R2)
        1C A2   000037FD 8F  D0  002A   241         MOVL    #DSA$M_ALLFLAGS, -         ; GET "ALL" FLAGS.
                                0032   242                   CLI$L_DATA(R2)
                                0032   243
        0000FE00'EF   1C A2  CA  0032   244 10$:    BICL2   CLI$L_DATA(R2), -          ; Clear specified flags
                                003A   245                   L^DSA$GL_FLAGS
                             05  003A   246         RSB                                ; RETURN
```

```
003B   248          .SBTTL  VRSetEF Routine - Set event flags
003B   249 ;++
003B   250 ; FUNCTIONAL DESCRIPTION:
003B   251 ;
003B   252 ;       SETS OPERATOR SPECIFIED EVENT FLAGS (1-23 ONLY)
003B   253 ;
003B   254 ; CALLING SEQUENCE:
003B   255 ;
003B   256 ;       BSBW    VRSETEF
003B   257 ;
003B   258 ; INPUT PARAMETERS:
003B   259 ;
003B   260 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
003B   261 ;
003B   262 ; IMPLICIT INPUTS:
003B   263 ;
003B   264 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
003B   265 ;
003B   266 ; OUTPUT PARAMETERS:
003B   267 ;
003B   268 ;       NONE
003B   269 ;
003B   270 ; IMPLICIT OUTPUTS:
003B   271 ;
003B   272 ;       NONE
003B   273 ;
003B   274 ; COMPLETION CODES:
003B   275 ;
003B   276 ;       NONE
003B   277 ;
003B   278 ; SIDE EFFECTS:
003B   279 ;
003B   280 ;       NONE
003B   281 ;
003B   282 ;--
```

```
                                 003B   VRSETEF::                                  ;
                      08    BB   003B           PUSHR   #^M<R3>                    ; SAVE SOME WORKING REGISTERS
      1C A2   FF000001 8F    CA   003D           BICL    #^XFF000001, -             ; CLEAR ILLEGAL FLAGS
                                 0045                     CLI$L_DATA(R2)
                                 0045
                53    17    D0   0045   10$:    MOVL    #23,R3                     ; UPPER FLAG NUMBER LIMIT
                                 0048
          09 1C A2    53    E1   0048   20$:    BBC     R3, CLI$L_DATA(R2), 30$    ; EFN SPECIFIED ?
                                 004D           $SETEF_S  R3                       ; YES, THEN GO SET IT
                      53    DD   004D                   PUSHL   R3
      00000000'GF     01    FB   004F                   CALLS   #1,G^SYS$SETEF
                                 0056
                EF    53    F5   0056   30$:    SOBGTR  R3,20$                     ; NEXT TILL DONE (DON'T DO 0)
                      08    BA   0059           POPR    #^M<R3>                    ; RESTORE REGISTERS
                            05   005B           RSB                                ; RETURN
```

```
                        005C    298          .SBTTL  VRClrEF Routine - Clear event flags
                        005C    299 ;++
                        005C    300 ; FUNCTIONAL DESCRIPTION:
                        005C    301 ;
                        005C    302 ;       CLEARS OPERATOR SPECIFIED EVENT FLAGS (1-23 ONLY)
                        005C    303 ;
                        005C    304 ; CALLING SEQUENCE:
                        005C    305 ;
                        005C    306 ;       BSBW    VRCLREF
                        005C    307 ;
                        005C    308 ; INPUT PARAMETERS:
                        005C    309 ;
                        005C    310 ;       CLI$L_DATA = MASK OF SPECIFIED FLAGS
                        005C    311 ;
                        005C    312 ; IMPLICIT INPUTS:
                        005C    313 ;
                        005C    314 ;       R2      = BASE ADDRESS FOR CLI DATA BLOCK (DS$GL_CLIBASE).
                        005C    315 ;
                        005C    316 ; OUTPUT PARAMETERS:
                        005C    317 ;
                        005C    318 ;       NONE
                        005C    319 ;
                        005C    320 ; IMPLICIT OUTPUTS:
                        005C    321 ;
                        005C    322 ;       NONE
                        005C    323 ;
                        005C    324 ; COMPLETION CODES:
                        005C    325 ;
                        005C    326 ;       NONE
                        005C    327 ;
                        005C    328 ; SIDE EFFECTS:
                        005C    329 ;
                        005C    330 ;       NONE
                        005C    331 ;
                        005C    332 ;--
```

```
                                  005C   334 VRCLREF::                                  ;
                        08   BB   005C   335         PUSHR   #^M<R3>                    ; SAVE SOME WORKING REGISTERS
      1C A2   FF000001  8F   CA   005E   336         BICL    #^XFF000001, -             ; CHECK FOR ILLEGAL FLAGS
                                  0066   337                   CLI$L_DATA(R2)
                                  0066   338
                  53    17   D0   0066   339 10$:    MOVL    #23,R3                     ; UPPER FLAG NUMBER LIMIT
                                  0069   340
            09 1C A2    53   E1   0069   341 20$:    BBC     R3, CLI$L_DATA(R2), 30$ ; EFN SPECIFIED ?
                                  006E   342         $CLREF_S  R3                       ; YES, THEN GO CLEAR IT
                        53   DD   006E                       PUSHL   R3
         00000000'GF    01   FB   0070                       CALLS   #1,G^SYS$CLREF
                                  0077   343
                    EF  53   F5   0077   344 30$:    SOBGTR  R3,20$                     ; NEXT TILL DONE (DON'T DO 0)
                        08   BA   007A   345         POPR    #^M<R3>                    ; RESTORE REGISTERS
                             05   007C   346         RSB                                ; RETURN
```

```
                        007D    348          .SBTTL  VRShowFlg Routine - Show control flags
                        007D    349 ;++
                        007D    350 ; FUNCTIONAL DESCRIPTION:
                        007D    351 ;
                        007D    352 ;       DISPLAYS OPERATOR CONTROL FLAGS WHICH ARE SET FOLLOWED BY
                        007D    353 ;       THOSE WHICH ARE CLEAR (I.E., DISPLAYS ALL CONTROL FLAGS)
                        007D    354 ;
                        007D    355 ; CALLING SEQUENCE:
                        007D    356 ;
                        007D    357 ;       BSBW    VRSHOWFLG
                        007D    358 ;
                        007D    359 ; INPUT PARAMETERS:
                        007D    360 ;
                        007D    361 ;       NONE
                        007D    362 ;
                        007D    363 ; IMPLICIT INPUTS:
                        007D    364 ;
                        007D    365 ;       DSA$GL_FLAGS = FLAGS WHICH ARE TO BE DISPLAYED
                        007D    366 ;
                        007D    367 ; OUTPUT PARAMETERS:
                        007D    368 ;
                        007D    369 ;       NONE
                        007D    370 ;
                        007D    371 ; IMPLICIT OUTPUTS:
                        007D    372 ;
                        007D    373 ;       NONE
                        007D    374 ;
                        007D    375 ; COMPLETION CODES:
                        007D    376 ;
                        007D    377 ;       NONE
                        007D    378 ;
                        007D    379 ; SIDE EFFECTS:
                        007D    380 ;
                        007D    381 ;       NONE
                        007D    382 ;
                        007D    383 ;--
```

```
                                  007D   385 VRSHOWFLG::                                  ;
                        04    BB  007D   386         PUSHR   #^M<R2>                      ; SAVE A WORKING REGISTER
                                  007F   387         $DS_CVTREG_S                       - ; BUILD MNEMONIC STRING
                                  007F   388             MSB=#LEFTMOSTFLAG,             - ; FIRST BIT TO BE INTERPRETED
                                  007F   389             DATA=L^DSA$GL_FLAGS,           - ; Register location
                                  007F   390             MNEADR=L^AFLAGMNE,             - ; CONTROL STRING ADDRESS                [08]
                                  007F   391             STRBUF=L^DS$GT_STRBUF,         - ; MNEMONIC STRING BUFFER                [08]
                                  007F   392             MAXLEN=#DS$K_STRBUF              ; SIZE OF ABOVE BUFFER
                        00    DD  007F                     PUSHL   #0
                        00    DD  0081                     PUSHL   #0
                        00    DD  0083                     PUSHL   #0
                        00    DD  0085                     PUSHL   #0
                        00    DD  0087                     PUSHL   #0
                        00    DD  0089                     PUSHL   #0
              00000000'8F    DD  008B                     PUSHL   #DS$K_STRBUF
              00000000'EF    9F  0091                     PUSHAB  L^DS$GT_STRBUF
              00000051'EF    9F  0097                     PUSHAB  L^AFLAGMNE
              0000FE00'EF    DD  009D                     PUSHL   L^DSA$GL_FLAGS
                        13    DD  00A3                     PUSHL   #LEFTMOSTFLAG
        00000000'9F     0B    FB  00A5                     CALLS   #11, @#DS$CVTREG
                                  00AC   393
                                  00AC   394         $Print -                           ; Display flags which are set             [11]
                                  00AC   395                 #DS$K_Type_Command_Out, -; ... typecode number                    [11]
                                  00AC   396                 #DS$K_PrintF, -          ; ... type of print                       [11]
                                  00AC   397                 L^FMTSHOWFLGS, -         ; ... the format string                   [11]
                                  00AC   398                 #DS$GT_STRBUF            ; .. the converted register output        [11]
              00000000'8F    DD  00AC                     PUSHL   #DS$GT_STRBUF
              00000000'EF    9F  00B2                     PUSHAB  L^FMTSHOWFLGS
                     7E    01  9A  00B8                     movzbl  #DS$K_PrintF, -(sp)
                     7E    16  98  00BB                     cvtbl   #DS$K_Type_Command_Out, -(sp)
        00000000'GF     04    FB  00BE                     CALLS   #$$N+2, G^DSX$PRINT
                                  00C5   399
        52    0000FE00'EF    D2  00C5   400         MCOML   L^DSA$GL_FLAGS, R2         ; Get cleared flags
                                  00CC   401
                                  00CC   402         $DS_CVTREG_S                       - ; BUILD MNEMONIC STRING
                                  00CC   403             MSB=#LEFTMOSTFLAG,             - ; FIRST BIT TO BE INTERPRETED
                                  00CC   404             DATA=R2,                       - ; REGISTER LOCATION
                                  00CC   405             MNEADR=L^AFLAGMNE,             - ; CONTROL STRING ADDRESS                [08]
                                  00CC   406             STRBUF=L^DS$GT_STRBUF,         - ; MNEMONIC STRING BUFFER                [08]
                                  00CC   407             MAXLEN=#DS$K_STRBUF              ; SIZE OF ABOVE BUFFER
                        00    DD  00CC                     PUSHL   #0
                        00    DD  00CE                     PUSHL   #0
                        00    DD  00D0                     PUSHL   #0
                        00    DD  00D2                     PUSHL   #0
                        00    DD  00D4                     PUSHL   #0
                        00    DD  00D6                     PUSHL   #0
              00000000'8F    DD  00D8                     PUSHL   #DS$K_STRBUF
              00000000'EF    9F  00DE                     PUSHAB  L^DS$GT_STRBUF
              00000051'EF    9F  00E4                     PUSHAB  L^AFLAGMNE
                        52    DD  00EA                     PUSHL   R2
                        13    DD  00EC                     PUSHL   #LEFTMOSTFLAG
        00000000'9F     0B    FB  00EE                     CALLS   #11, @#DS$CVTREG
                                  00F5   408
                                  00F5   409         $Print -                           ; Display flags which are clear           [11]
                                  00F5   410                 #DS$K_Type_Command_Out, -; ... typecode number                    [11]
                                  00F5   411                 #DS$K_PrintF, -          ; ... type of print                       [11]
                                  00F5   412                 L^FMTSHOWFLGC, -         ; ... the format string                   [11]
```

```
                                00F5   413                  #DS$GT_STRBUF          ; .. the converted register output      [11]
        00000000'8F      DD     00F5                        PUSHL   #DS$GT_STRBUF
        0000001B'EF      9F     00FB                        PUSHAB  L^FMTSHOWFLGC
            7E    01     9A     0101                        movzbl  #DS$K_PrintF, -(sp)
            7E    16     98     0104                        cvtbl   #DS$K_Type_Command_Out, -(sp)
    00000000'GF    04     FB     0107                        CALLS   #$$N+2, G^DSX$PRINT
                                010E   414
                   04     BA     010E   415          POPR    #^M<R2>                ; RESTORE REGISTERS
                          05     0110   416          RSB                            ; RETURN
```

```
0111    418          .SBTTL  VRShowEF Routine - Show event flags
0111    419 ;++
0111    420 ; FUNCTIONAL DESCRIPTION:
0111    421 ;
0111    422 ;       DISPLAYS ALL OPERATOR ACCESSABLE EVENT FLAGS WHICH ARE
0111    423 ;       CURRENTLY SET
0111    424 ;
0111    425 ; CALLING SEQUENCE:
0111    426 ;
0111    427 ;       BSBW    VRSHOWEF
0111    428 ;
0111    429 ; INPUT PARAMETERS:
0111    430 ;
0111    431 ;       NONE
0111    432 ;
0111    433 ; IMPLICIT INPUTS:
0111    434 ;
0111    435 ;       NONE
0111    436 ;
0111    437 ; OUTPUT PARAMETERS:
0111    438 ;
0111    439 ;       NONE
0111    440 ;
0111    441 ; IMPLICIT OUTPUTS:
0111    442 ;
0111    443 ;       DS$GL_EFC0 = CLUSTER 0 EVENT FLAGS
0111    444 ;
0111    445 ; COMPLETION CODES:
0111    446 ;
0111    447 ;       NONE
0111    448 ;
0111    449 ; SIDE EFFECTS:
0111    450 ;
0111    451 ;       NONE
0111    452 ;
0111    453 ;--
```

```
                                  0111   455 VRSHOWEF::                                  ;
                        04   BB   0111   456         PUSHR   #^M<R2>                    ; SAVE A WORKING REGISTER
                                  0113   457         $READEF_S  #0,L^DS$GL_EFC0         ; READ EVENT FLAG CLUSTER 0          [08]
              00000000'EF   DF   0113                       PUSHAL  L^DS$GL_EFC0
                        00   DD   0119                       PUSHL   #0
         00000000'GF    02   FB   011B                       CALLS   #2,G^SYS$READEF
   52   00000000'EF   FF 8F   9C   0122   458         ROTL    #-1,L^DS$GL_EFC0,R2        ; POSITION FOR CVTREG                [12]
                                  012B   459
                                  012B   460         $DS_CVTREG_S                     - ; BUILD MNEMONIC STRING
                                  012B   461             MSB=#22,                     - ; FIRST BIT TO BE INTERPRETED
                                  012B   462             DATA=R2,                     - ; REGISTER LOCATION
                                  012B   463             MNEADR=L^AEFNMNE,            - ; CONTROL STRING ADDRESS           [08]
                                  012B   464             STRBUF=L^DS$GT_STRBUF,       - ; MNEMONIC STRING BUFFER           [08]
                                  012B   465             MAXLEN=#DS$K_STRBUF            ; SIZE OF ABOVE BUFFER
                        00   DD   012B                       PUSHL   #0
                        00   DD   012D                       PUSHL   #0
                        00   DD   012F                       PUSHL   #0
                        00   DD   0131                       PUSHL   #0
                        00   DD   0133                       PUSHL   #0
                        00   DD   0135                       PUSHL   #0
              00000000'8F   DD   0137                       PUSHL   #DS$K_STRBUF
              00000000'EF   9F   013D                       PUSHAB  L^DS$GT_STRBUF
              000000AD'EF   9F   0143                       PUSHAB  L^AEFNMNE
                        52   DD   0149                       PUSHL   R2
                        16   DD   014B                       PUSHL   #22
         00000000'9F    0B   FB   014D                       CALLS   #11, a#DS$CVTREG
                                  0154   466
                                  0154   467         $Print -                           ; Display flags which are set        [11]
                                  0154   468                 #DS$K_Type_Command_Out, -; ... typecode number                [11]
                                  0154   469                 #DS$K_PrintF, -           ; ... type of print                  [11]
                                  0154   470                 L^FMTSHOWEF, -            ; ... the format string              [11]
                                  0154   471                 #DS$GT_STRBUF             ; .. the converted register output   [11]
              00000000'8F   DD   0154                       PUSHL   #DS$GT_STRBUF
              00000038'EF   9F   015A                       PUSHAB  L^FMTSHOWEF
                   7E   01   9A   0160                       movzbl  #DS$K_PrintF, -(sp)
                   7E   16   98   0163                       cvtbl   #DS$K_Type_Command_Out, -(sp)
         00000000'GF    04   FB   0166                       CALLS   #$$N+2, G^DSX$PRINT
                                  016D   472
                        04   BA   016D   473         POPR    #^M<R2>                    ; RESTORE REGISTERS
                             05   016F   474         RSB                                ; RETURN
                                  0170   475         .END
```

```
$$N                             = 00000002    D
$ENV                            = 00000002    D
AEFNMNE                           000000AD R  D  02
AFLAGMNE                          00000051 R  D  02
BIT...                          = 00000004    D
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        0000044C    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_FLAGS2                      00000444    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEMP_BASE                   00000448    D
CLI$L_TEST                        00000020    D
CLI$M_START_OR_RUN              = 00000008    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BASE_QUAL                 = 00000002    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NEXT                      = 00000001    D
CLI$V_NOALLOC                   = 00000000    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_START_OR_RUN              = 00000003    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
DS$CVTREG                         ********    X  03
DS$GL_EFCO                        ********    X  03
DS$GT_STRBUF                      ********    X  03
DS$K_PRINTB                     = 00000002    D
DS$K_PRINTF                     = 00000001    D
DS$K_PRINTI                     = 00000000    D
DS$K_PRINTX                     = 00000003    D
DS$K_STRBUF                       ********    X  03
DS$K_TYPE_ABORT_PROGRAM         = 00000014    D
DS$K_TYPE_ABORT_TEST            = 00000013    D
DS$K_TYPE_COMMAND_ERR           = 00000015    D
DS$K_TYPE_COMMAND_OUT           = 00000016    D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A    D
DS$K_TYPE_DS_PROMPT             = 00000001    D
DS$K_TYPE_DS_START              = 0000001D    D
DS$K_TYPE_ERRDEV                = 00000008    D
DS$K_TYPE_ERRHARD               = 00000006    D
DS$K_TYPE_ERROR_BODY            = 00000009    D
DS$K_TYPE_ERROR_END             = 0000000A    D
DS$K_TYPE_ERRPREP               = 0000001B    D
DS$K_TYPE_ERRSOFT               = 00000007    D
DS$K_TYPE_ERRSUP                = 00000004    D
DS$K_TYPE_ERRSYS                = 00000005    D
DS$K_TYPE_ERR_HALT              = 0000000D    D
DS$K_TYPE_EXCEPTION             = 0000000C    D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D
DS$K_TYPE_FIRST_PASS            = 00000011    D
DS$K_TYPE_GENERAL               = 00000000    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D
DS$K_TYPE_NO_TESTS              = 00000012    D
DS$K_TYPE_PARAM_ERROR           = 0000001C    D
DS$K_TYPE_PROGRAM_END           = 00000010    D
DS$K_TYPE_PROGRAM_INFO          = 00000017    D
DS$K_TYPE_PROGRAM_START         = 0000000F    D
DS$K_TYPE_QIO_INVADP            = 00000024    D
DS$K_TYPE_QIO_NODRIVER          = 00000022    D
DS$K_TYPE_QIO_WRONGVER          = 00000023    D
DS$K_TYPE_SCRIPT_ECHO           = 00000021    D
DS$K_TYPE_SCRIPT_PNF            = 0000001E    D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020    D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F    D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D
DS$K_TYPE_START_ERR             = 00000018    D
DS$K_TYPE_START_LIST            = 00000025    D
DS$K_TYPE_SUMMARY               = 0000000E    D
DS$K_TYPE_USER_PROMPT           = 00000002    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
```

```
DSA$GL_MSGTYP                  0000FE40    D
DSA$GL_PASSES                  0000FE08    D
DSA$GL_PASSNO                  0000FE54    D
DSA$GL_SECTNO                  0000FE10    D
DSA$GL_SID                     0000FE14    D
DSA$GL_SUBTNO                  0000FE4C    D
DSA$GL_TESTNO                  0000FE50    D
DSA$GL_UNITS                   0000FE0C    D
DSA$GQ_MSGPTR                  0000FE68    D
DSA$GT_DEVNAM                  0000FE5C    D
DSA$M_ALLFLAGS               = 000037FD    D
DSA$M_BELL                   = 00000008    D
DSA$M_DFLTFLGS                 ********   X   03
DSA$M_HALT                   = 00000001    D
DSA$M_IE1                    = 00000010    D
DSA$M_IE2                    = 00000020    D
DSA$M_IE3                    = 00000040    D
DSA$M_IES                    = 00000080    D
DSA$M_LOOP                   = 00000004    D
DSA$M_OPER                   = 00001000    D
DSA$M_PROMPT                 = 00002000    D
DSA$M_QUICK                  = 00000100    D
DSA$M_TRACE                  = 00000400    D
DSA$M_VERIFY                 = 00000200    D
DSA$V_ASK_PROMPT             = 00000013    D
DSX$PRINT                      ********   X   03
FMTSHOWEF                      00000038 R  D  02
FMTSHOWFLGC                    0000001B R  D  02
FMTSHOWFLGS                    00000000 R  D  02
LEFTMOSTFLAG                 = 00000013    D
SIZ...                       = 00000001    D
SYS$CLREF                      ********  GX   03
SYS$READEF                     ********  GX   03
SYS$SETEF                      ********  GX   03
VRCLREF                        0000005C RG D  03
VRCLRFLG                       00000025 RG D  03
VRSETEF                        0000003B RG D  03
VRSETFLG                       00000000 RG D  03
VRSETFLG_X                     00000024 R  D  03
VRSHOWEF                       00000111 RG D  03
VRSHOWFLG                      0000007D RG D  03
V_ENV                        = 00000001    D
V_LVL                        = 00000000    D

                                        +----------------+
                                        ! Psect synopsis !
                                        +----------------+

PSECT name                     Allocation          PSECT No.  Attributes
----------                     ----------          ---------  ----------
.  ABS  .                      00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                          0000FE70  (65136.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
DATA                           000000E9  (  233.)  02 (  2.)  NOPIC   USR   CON   REL   LCL   SHR NOEXE   RD  NOWRT NOVEC BYTE
CODE                           00000170  (  368.)  03 (  3.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000002	471 (1)	398 (1) 413 (1) 471 (1)
$ENV	=00000002	111 (1)	
AEFNMNE	000000AD-R	140 (1)	465 (1)
AFLAGMNE	00000051-R	137 (1)	392 (1) 407 (1)
BIT...	=00000004	109 (1)	109 (1)
CLI$K_BUFSIZ	=00000100	107 (1)	107 (1)
CLI$K_SIZE	0000044C	107 (1)	
CLI$L_ADDRESS	00000018	107 (1)	
CLI$L_COMMAND	00000004	107 (1)	
CLI$L_DATA	0000001C	107 (1)	#-187 (1) #-192 (1) #-194 (1) #-237 (1)
			#-242 (1) #-244 (1) #-287 (1) 291 (1)
			#-337 (1) 341 (1)
CLI$L_FLAGS	00000000	107 (1)	182 (1)
CLI$L_FLAGS2	00000444	107 (1)	
CLI$L_LAST	00000024	107 (1)	
CLI$L_NEXT	00000030	107 (1)	
CLI$L_PASS	0000002C	107 (1)	
CLI$L_SUBT	00000028	107 (1)	
CLI$L_TEMP_BASE	00000448	107 (1)	
CLI$L_TEST	00000020	107 (1)	
CLI$M_START_OR_RUN	=00000008	107 (1)	
CLI$Q_BUFQWD	00000034	107 (1)	
CLI$Q_FILE	00000008	107 (1)	
CLI$Q_SECTION	00000010	107 (1)	
CLI$Q_TIME	0000043C	107 (1)	
CLI$T_BUFFER	0000003C	107 (1)	
CLI$V_ADAPTER	=00000018	107 (1)	
CLI$V_ADR	=0000000B	107 (1)	
CLI$V_ASCII	=00000013	107 (1)	
CLI$V_BASE_QUAL	=00000002	107 (1)	
CLI$V_BREAK	=0000000A	107 (1)	
CLI$V_BRIEF	=0000001B	107 (1)	
CLI$V_BYTE	=0000000D	107 (1)	
CLI$V_CLEAR	=00000002	107 (1)	
CLI$V_DEC	=00000010	107 (1)	
CLI$V_DEFAULT	=0000000C	107 (1)	#-181 (1)
CLI$V_DEPOSIT	=00000019	107 (1)	
CLI$V_EVENT	=00000008	107 (1)	
CLI$V_EXAM	=00000005	107 (1)	
CLI$V_FLAGS	=00000009	107 (1)	
CLI$V_HEX	=00000012	107 (1)	
CLI$V_KERNEL	=00000017	107 (1)	
CLI$V_LOAD	=00000006	107 (1)	
CLI$V_LONG	=0000000F	107 (1)	
CLI$V_NEXT	=00000001	107 (1)	
CLI$V_NOALLOC	=00000000	107 (1)	
CLI$V_NOTNUF	=00000001	107 (1)	
CLI$V_OCT	=00000011	107 (1)	
CLI$V_PREG	=0000001A	107 (1)	
CLI$V_QA	=00000007	107 (1)	

```
CLI$V_QACKLOOPLOOPS             =0000001C     107     (1)
CLI$V_QAERRORPRINTS             =0000001B     107     (1)
CLI$V_QAMULTIPLEPASS            =0000001F     107     (1)
CLI$V_QASUBTESTLOOPS            =0000001E     107     (1)
CLI$V_QATESTLOOPS               =0000001D     107     (1)
CLI$V_REG                       =00000014     107     (1)
CLI$V_REQUIRED                  =00000000     107     (1)
CLI$V_RUN                       =00000015     107     (1)
CLI$V_SET                       =00000003     107     (1)
CLI$V_SHOW                      =00000004     107     (1)
CLI$V_START_OR_RUN              =00000003     107     (1)     107     (1)
CLI$V_VALSEC                    =00000016     107     (1)
CLI$V_WORD                      =0000000E     107     (1)
DS$CVTREG                        00000000-XR                  392     (1)      407     (1)      465     (1)
DS$GL_EFCO                       00000000-XR                  457     (1)    #-458     (1)
DS$GT_STRBUF                     00000000-XR                  392     (1)    #-398     (1)      407     (1)    #-413     (1)
                                                              465     (1)    #-471     (1)
DS$K_PRINTB                     =00000002     109     (1)
DS$K_PRINTF                     =00000001     109     (1)    #-398     (1)    #-413     (1)    #-471     (1)
DS$K_PRINTI                     =00000000     109     (1)
DS$K_PRINTX                     =00000003     109     (1)
DS$K_STRBUF                      00000000-XR                #-392     (1)    #-407     (1)    #-465     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     109     (1)
DS$K_TYPE_ABORT_TEST            =00000013     109     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     109     (1)
DS$K_TYPE_COMMAND_OUT           =00000016     109     (1)    #-398     (1)    #-413     (1)    #-471     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     109     (1)
DS$K_TYPE_DS_PROMPT             =00000001     109     (1)
DS$K_TYPE_DS_START              =0000001D     109     (1)
DS$K_TYPE_ERRDEV                =00000008     109     (1)
DS$K_TYPE_ERRHARD               =00000006     109     (1)
DS$K_TYPE_ERROR_BODY            =00000009     109     (1)
DS$K_TYPE_ERROR_END             =0000000A     109     (1)
DS$K_TYPE_ERRPREP               =0000001B     109     (1)
DS$K_TYPE_ERRSOFT               =00000007     109     (1)
DS$K_TYPE_ERRSUP                =00000004     109     (1)
DS$K_TYPE_ERRSYS                =00000005     109     (1)
DS$K_TYPE_ERR_HALT              =0000000D     109     (1)
DS$K_TYPE_EXCEPTION             =0000000C     109     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     109     (1)
DS$K_TYPE_FIRST_PASS            =00000011     109     (1)
DS$K_TYPE_GENERAL               =00000000     109     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     109     (1)
DS$K_TYPE_NO_TESTS              =00000012     109     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     109     (1)
DS$K_TYPE_PROGRAM_END           =00000010     109     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     109     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     109     (1)
DS$K_TYPE_QIO_INVADP            =00000024     109     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     109     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     109     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     109     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     109     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     109     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     109     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     109     (1)
DS$K_TYPE_START_ERR             =00000018     109     (1)
```

```
DS$K_TYPE_START_LIST            =00000025      109    (1)
DS$K_TYPE_SUMMARY               =0000000E      109    (1)
DS$K_TYPE_USER_PROMPT           =00000002      109    (1)
DSA$GL_FLAGS                     0000FE00                    #-184    (1)    #-195    (1)    #-245    (1)    #-392    (1)
                                                             #-400    (1)
DSA$M_ALLFLAGS                  =000037FD      119    (1)   #-191    (1)    #-241    (1)
DSA$M_BELL                      =00000008                      115    (1)
DSA$M_DFLTFLGS                   00000000-XR                  #-183    (1)
DSA$M_HALT                      =00000001                      115    (1)
DSA$M_IE1                       =00000010                      116    (1)
DSA$M_IE2                       =00000020                      116    (1)
DSA$M_IE3                       =00000040                      116    (1)
DSA$M_IES                       =00000080                      118    (1)
DSA$M_LOOP                      =00000004                      115    (1)
DSA$M_OPER                      =00001000                      118    (1)
DSA$M_PROMPT                    =00002000                      118    (1)
DSA$M_QUICK                     =00000100                      117    (1)
DSA$M_TRACE                     =00000400                      117    (1)
DSA$M_VERIFY                    =00000200                      117    (1)
DSA$V_ASK_PROMPT                =00000013                      113    (1)
DSX$PRINT                        00000000-XR                   398    (1)      413    (1)      471    (1)
FMTSHOWEF                        00000038-R     134    (1)     471    (1)
FMTSHOWFLGC                      0000001B-R     131    (1)     413    (1)
FMTSHOWFLGS                      00000000-R     128    (1)     398    (1)
LEFTMOSTFLAG                    =00000013       113    (1)   #-392    (1)    #-407    (1)
SYS$CLREF                        00000000-XR                   342    (1)
SYS$READEF                       00000000-XR                   457    (1)
SYS$SETEF                        00000000-XR                   292    (1)
VRCLREF                          0000005C-R     334    (1)
VRCLRFLG                         00000025-R     236    (1)
VRSETEF                          0000003B-R     284    (1)
VRSETFLG                         00000000-R     180    (1)
VRSETFLG_X                       00000024-R     197    (1)   #-185    (1)
VRSHOWEF                         00000111-R     455    (1)
VRSHOWFLG                        0000007D-R     385    (1)
V_ENV                           =00000001       108    (1)     111    (1)
V_LVL                           =00000000       108    (1)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO            SIZE        DEFINITION      REFERENCES...
-----            ----        ----------      -------------
$CLREF_S         1           342    (1)      342    (1)
$D1_PRINT_S      1           398    (1)      398    (1)      413    (1)      471    (1)
$DEF             1           109    (1)
$DEFINI          1           106    (1)      106    (1)
$DS_CVTREG_S     1           388    (1)      388    (1)      403    (1)      461    (1)
$DS_DSADEF       5           106    (1)      106    (1)
$DS_TYPEDEF      4           109    (1)      109    (1)
$EQU             1           109    (1)      109    (1)
$EQULS1          1           109    (1)      109    (1)
$EQULST          1           109    (1)      109    (1)
$GBLINI          2                           109    (1)
$PRINT           2           395    (1)      395    (1)      410    (1)      468    (1)
$PUSHADR         1           392    (1)      392    (1)      407    (1)      457    (1)      465    (1)
$READEF_S        1           457    (1)      457    (1)
$SETEF_S         1           292    (1)      292    (1)
$VIELD1          1           109    (1)
CLIDEF           4           107    (1)      107    (1)
ENVDEF           1           108    (1)      108    (1)
```

```
                                    +-------------------------+
                                    ! Opcode Cross Reference !
                                    +-------------------------+

OPCODE          VALUE      REFERENCES...
------          -----      -------------
BBC             00E1       181     (1)     291     (1)     341     (1)
BGEQ            0018       188     (1)     238     (1)
BICL            00CA       286     (1)     336     (1)
BICL2           00CA       244     (1)
BISL2           00C8       194     (1)
BRB             0011       185     (1)
CALLS           00FB       292     (1)     342     (1)     392     (1)     398     (1)     407     (1)     413     (1)     457     (1)
                           465     (1)     471     (1)
CVTBL           0098       398     (1)     413     (1)     471     (1)
MCOML           00D2       400     (1)
MOVL            00D0       191     (1)     241     (1)     289     (1)     339     (1)
MOVW            00B0       183     (1)
MOVZBL          009A       398     (1)     413     (1)     471     (1)
POPR            00BA       295     (1)     345     (1)     415     (1)     473     (1)
PUSHAB          009F       392     (1)     398     (1)     407     (1)     413     (1)     465     (1)     471     (1)
PUSHAL          00DF       457     (1)
PUSHL           00DD       292     (1)     342     (1)     392     (1)     398     (1)     407     (1)     413     (1)     457     (1)
                           465     (1)     471     (1)
PUSHR           00BB       285     (1)     335     (1)     386     (1)     456     (1)
ROTL            009C       458     (1)
RSB             0005       198     (1)     246     (1)     296     (1)     346     (1)     416     (1)     474     (1)
SOBGTR          00F5       294     (1)     344     (1)
TSTL            00D5       187     (1)     237     (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	129	(1)	132	(1)	135	(1)	138	(1)	141	(1)						
.BLKL	107	(1)														
.END	475	(1)														
.ENDC	109	(1)	392	(1)	398	(1)	407	(1)	413	(1)	457	(1)	465	(1)	471	(1)
.ENDM	106	(1)	107	(1)	108	(1)	109	(1)	292	(1)	342	(1)	388	(1)	392	(1)
	395	(1)	398	(1)	457	(1)										
.ENDR	109	(1)	398	(1)	413	(1)	471	(1)								
.GLOBL	292	(1)	342	(1)	457	(1)										
.IDENT	5	(1)														
.IF	109	(1)	392	(1)	398	(1)	407	(1)	413	(1)	457	(1)	465	(1)	471	(1)
.IFF	109	(1)	392	(1)	407	(1)	457	(1)	465	(1)						
.IIF	109	(1)														
.IRP	109	(1)	398	(1)	413	(1)	471	(1)								
.LIBRARY	94	(1)	95	(1)	96	(1)										
.LIST	106	(1)	107	(1)												
.MACRO	109	(1)														
.NLIST	106	(1)	107	(1)												
.NOCROSS	106	(1)														
.NOSHOW	6	(1)														
.PAGE	120	(1)	142	(1)	179	(1)	199	(1)	235	(1)	247	(1)	27	(1)	283	(1)
	297	(1)	333	(1)	347	(1)	384	(1)	417	(1)	454	(1)	89	(1)		
.PSECT	107	(1)	126	(1)	144	(1)										
.RESTORE	106	(1)	107	(1)												
.SAVE	106	(1)	107	(1)												
.SBTTL	143	(1)	200	(1)	248	(1)	298	(1)	348	(1)	418	(1)	90	(1)		
.SUBTITLE	121	(1)														
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R2	19	182	(1)	#-187	(1)	#-192	(1)	#-194	(1)	#-237	(1)	#-242	(1)
		#-244	(1)	#-287	(1)	291	(1)	#-337	(1)	341	(1)	#-386	(1)
		#-400	(1)	#-407	(1)	#-415	(1)	#-456	(1)	#-458	(1)	#-465	(1)
		#-473	(1)										
R3	12	#-285	(1)	#-289	(1)	#-291	(1)	#-292	(1)	#-294	(1)	#-295	(1)
		#-335	(1)	#-339	(1)	#-341	(1)	#-342	(1)	#-344	(1)	#-345	(1)
SP	6	#-398	(1)	#-413	(1)	#-471	(1)						

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.03	00:00:00.32
Command processing	883	00:00:00.26	00:00:00.68
Pass 1	571	00:00:01.21	00:00:02.80
Symbol table sort	0	00:00:00.05	00:00:00.04
Pass 2	24	00:00:00.34	00:00:00.70
Symbol table output	2	00:00:00.03	00:00:00.06
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	5	00:00:00.14	00:00:00.23
Assembler run totals	1512	00:00:02.06	00:00:04.83

The working set limit was 2400 pages.
59865 bytes (117 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 201 non-local and 9 local symbols.
475 source lines were read in Pass 1, producing 27 object records in Pass 2.
40 pages of virtual memory were used to define 21 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	3
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	16

318 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:FLAGS.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

Table of contents

(1) 86 DECLARATIONS
(2) 211 FIL$OPENFILE - RETURN FILE HEADER AND STATISTICS BLOCK
(3) 483 FIL$CACHE_INIT - INIT FILEREAD CACHE
(4) 571 FIL$CACHE_TRUNC - TRUNCATE FILEREAD CACHE
(5) 621 ASSIGN_DEV - ASSIGN A DEVICE AND RETURN A CHANNEL
(6) 665 STORE3DIGITS - STORE 3 ASCII DIGITS
(7) 700 FORMDIRSTRING - GET A DIRECTORY STRING
(8) 767 MOUNT - MOUNT THE VOLUME, INIT FOR FILE LOOKUP
(9) 883 FINDFILID - FIND FILE ID FOR SPECIFIED FILE
(10) 1150 FIL$FINDFILID - STRUCTURE LEVEL 2
(11) 1271 FIL$FINDFILID - STRUCTURE LEVEL 1
(12) 1354 READ_DIR_LBN - READ NEXT DIRECTORY LBN
(13) 1397 RDCHKFILHDR - READ AND CHECK FILE HEADER
(14) 1547 READVBN, WRITEVBN - READ/WRITE VIRTUAL BLOCK
(15) 1643 INIRTRVPTRSCAN - INITIALIZE RETRIEVAL POINTER SCAN
(16) 1676 GETRTRVPTR - CONVERT NEXT RETRIEVAL POINTER
(17) 1773 STATBLK - GET FILE STATISTICS BLOCK
(18) 1890 FIL$CHKFILHDR - CHECK FILE HEADER VALIDITY
(19) 1962 CHECKSUM - VALIDATE A CHECKSUM

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element FILEREAD.MAR
0000     2 ; *6    29-SEP-1987 09:45:36 TRABUCCHI "FIX TRUNCATION ERROR IN CALL TO FIL$RDWRTLBN
0000     3 ; *5    16-MAR-1987 15:24:08 GEARIN "FIXED TRUNC ERROR"
0000     4 ; *4    23-JAN-1987 15:48:47 GEARIN "ADD 1987 COPYRIGHT DATE"
0000     5 ; *3    23-JAN-1987 09:46:52 GEARIN "DISPLACEMENT BUG FIXED"
0000     6 ; *2    22-JAN-1987 14:30:04 GEARIN "FIXED TRUNCATION ERROR"
0000     7 ; *1     6-FEB-1986 15:18:31 DS ""
0000     8 ; DEC/CMS REPLACEMENT HISTORY, Element FILEREAD.MAR
0000     9         .TITLE  FILEREAD - FILES11 LEVEL 1 & 2 FILE READING ROUTINES
0000    10         .IDENT  '10-03'                                                 ;G:3
0000    11
0000    12 ;
0000    13 ;*****************************************************************************
0000    14 ;*                                                                           *
0000    15 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983, 1987                               *  ;G:4
0000    16 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    17 ;*  ALL RIGHTS RESERVED.                                                     *
0000    18 ;*                                                                           *
0000    19 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    20 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    21 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    22 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    23 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    24 ;*  TRANSFERRED.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    27 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    28 ;*  CORPORATION.                                                             *
0000    29 ;*                                                                           *
0000    30 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    31 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    32 ;*                                                                           *
0000    33 ;*                                                                           *
0000    34 ;*****************************************************************************
0000    35 ;
0000    36 ;++
0000    37 ; FACILITY:     USER CALLABLE PROCEDURES
0000    38 ;
0000    39 ; ABSTRACT:
0000    40 ;
0000    41 ;       THIS SET OF ROUTINES PROVIDES THE CAPABILITY OF "OPENING" AND
0000    42 ; READING FILES BY FILE NAME FROM A FILES11 STRUCTURE LEVEL 1 OR 2 VOLUME.
0000    43 ; THERE IS NO MULTI-VOLUME SUPPORT, AND MULTI-HEADER SUPPORT IS LIMITED
0000    44 ; TO RETURNING THE CORRECT FILE SIZE IN THE STATBLK.
0000    45 ;
0000    46 ; ENVIRONMENT:  USER MODE
0000    47 ;
0000    48 ; AUTHOR: PETER H. LIPMAN       , CREATION DATE: 14-DEC-76
0000    49 ;
0000    50 ; MODIFIED BY:
0000    51 ;       02      Bob Bergazzi    Jun 29,1983     Version 6.12
0000    52 ;               Adapted for supervisor 6.12.
0000    53 ;
0000    54 ;       03      David Gearin    Jan 22,1987     Version 10.02           ;G:2
0000    55 ;               Change displacement in call to FIL$RDWRTLBN from        ;G:2
0000    56 ;               word displacement to longword displacement.             ;G:2
0000    57 ;                                                                       ;G:2
```

```
                                0000     58 ;       V03-001 PHL0103         Peter H. Lipman         20-Jul-1982
                                0000     59 ;               Correct error in FIL$FINDFILID for structure level 1
                                0000     60 ;               when directory is cached.
                                0000     61 ;
                                0000     62 ;       V02-007 PHL0019         Peter H. Lipman         31-Oct-1981
                                0000     63 ;               Fix bug in FIL$RDCHKFILHDR having to do with overflowing
                                0000     64 ;               the callers retrieval pointer buffer for multi-header files.
                                0000     65 ;               Fix documentation for FIL$STATBLK showing that the returned
                                0000     66 ;               retrieval pointer length is the number of bytes that would
                                0000     67 ;               have been stored if the buffer had been large enough.
                                0000     68 ;
                                0000     69 ;       V02-006 PHL0009         Peter H. Lipman         06-May-1981
                                0000     70 ;               TOPSYS directory was not being used properly.
                                0000     71 ;
                                0000     72 ;       V02-005 PHL0007         Peter H. Lipman         14-Mar-1981
                                0000     73 ;               Add cacheing of directory lookups, directory header info,
                                0000     74 ;               directory data blocks, and index file header.
                                0000     75 ;               Change interface to FIL$RDWRTLBN to allow the reading
                                0000     76 ;               of multiple blocks.
                                0000     77 ;               Implement multiple level directory lookups and a new
                                0000     78 ;               top level directory in which all the system directories
                                0000     79 ;               are found.
                                0000     80 ;       V02-004 PHL0006         Peter H. Lipman         13-Dec-1980
                                0000     81 ;               Add capability to return retrieval pointer information
                                0000     82 ;               and thus allow all non-contiguous files to be handled
                                0000     83 ;               at boot time.
                                0000     84 ;--
```

```
                        0000     86          .SBTTL  DECLARATIONS
                        0000     87 ;
                        0000     88 ; INCLUDE FILES:
                        0000     89 ;
                        0000     90          .nocross
                        0000     91          $DIRDEF                         ;DIRECTORY ENTRY OFFSET DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     92          $FATDEF                         ;RECORD ATTRIBUTE AREA DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     93          $FH1DEF                         ;FILE HEADER DEFINITIONS, LEVEL 1
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     94          $FH2DEF                         ;FILE HEADER DEFINITIONS, LEVEL 2
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     95          $FM1DEF                         ;       MAP AREA, LEVEL 1
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     96          $FM2DEF                         ;       MAP AREA, LEVEL 2
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     97          $FIDDEF                         ;FILE ID OFFSET DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     98          $HM1DEF                         ;HOME BLOCK DEFINITIONS, LEVEL 1
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000     99          $HM2DEF                         ;HOME BLOCK DEFINITIONS, LEVEL 2
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    100          $IODEF                          ;I/O DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    101          $PSLDEF                         ;PROCESSOR STATUS LONG WORD DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    102          $SSDEF                          ;SYSTEM SERVICE DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    103          $VADEF                          ;VIRTUAL ADDRESS DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    104 ;
                        0000    105 ; MACROS:
                        0000    106 ;
                        0000    107          .MACRO  READVBN CHAN,VBN,BUFADR,HDRADR
                        0000    108          .LIST   MEB
                        0000    109                  PUSHAL  HDRADR
                        0000    110                  PUSHAL  BUFADR
                        0000    111                  PUSHL   VBN
                        0000    112                  PUSHL   CHAN
                        0000    113                  CALLS   #4,W^FIL$READVBN
                        0000    114          .NLIST  MEB
                        0000    115          .ENDM   READVBN
                        0000    116
```

```
                     0000   117          .MACRO  READLBN CHAN,VBN,BUFADR
                     0000   118          .LIST   MEB
                     0000   119                  ROTL    #9,#1,-(SP)
                     0000   120                  MOVZWL  #IO$_READLBLK,-(SP)
                     0000   121                  PUSHAL  BUFADR
                     0000   122                  PUSHL   VBN
                     0000   123                  PUSHL   CHAN                                    ;G:6
                     0000   124                  CALLS   #5,L^FIL$RDWRTLBN                       ;G:5
                     0000   125          .NLIST  MEB
                     0000   126          .ENDM   READLBN
                     0000   127          .cross
                     0000   128 ;
                     0000   129 ; EQUATED SYMBOLS:
                     0000   130 ;
                     0000   131          ASSUME  FH1$B_MPOFFSET EQ FH2$B_MPOFFSET
                     0000   132          ASSUME  FH1$W_STRUCLEV EQ FH2$W_STRUCLEV
                     0000   133          ASSUME  FH1$W_CHECKSUM EQ FH2$W_CHECKSUM
                     0000   134          ASSUME  HM1$W_STRUCLEV EQ HM2$W_STRUCLEV
                     0000   135          ASSUME  HM1$W_CHECKSUM1 EQ HM2$W_CHECKSUM1
                     0000   136          ASSUME  HM1$W_CHECKSUM2 EQ HM2$W_CHECKSUM2
                     0000   137          ASSUME  FH1$W_CHECKSUM EQ HM1$W_CHECKSUM2
                     0000   138          ASSUME  FH1$V_CONTIG EQ FH2$V_CONTIG
                     0000   139
000001FE             0000   140          FH1$W_VBNOFFSET = FH1$W_CHECKSUM ;SAVE INDEX FILE VBN OFFSET
                     0000   141                                           ;IN THIS PLACE IN INDEX FILE HEADER
                     0000   142
                     0000   143          ASSUME  FH1$C_LEVEL1a-8 EQ 1
00000008             0000   144          FH1$V_LEVEL1 = 8
                     0000   145
                     0000   146          ASSUME  FH2$C_LEVEL2a-8 EQ 2
00000009             0000   147          FH2$V_LEVEL2 = 9
0000000A             0000   148          FH2$V_BIGFILNUM = 10            ;IF SET USE HIGH 8 BITS OF FILE ID RVN
                     0000   149                                          ;FIELD AS FILE NUMBER EXTENSION
                     0000   150                                          ;BIT IS PLACED IN FH2$W_STRUCLEV
                     0000   151                                          ;BY THE FIL$MOUNT CODE
                     0000   152
00000001             0000   153          FIL$C_CACHE_ID = 1              ;VERSION OF THE FILEREAD CACHE
                     0000   154 ;
                     0000   155 ; OFFSETS INTO HEADER PORTION OF THE FILEREAD CACHE
                     0000   156 ;
                     0000   157          $OFFSET 0,POSITIVE,<-
                     0000   158          <FIL$W_CACHE_ID,2>,-            ;CACHE IDENT, AND WRITE INTERLOCK BIT
                     0000   159          <,2>,-                          ;SPARE
                     0000   160          FIL$L_DIROFF,-                  ;OFFSET IN BYTES TO DIRECTORY CACHE
                     0000   161          FIL$L_DIRNXT,-                  ;NEXT OFFSET TO ALLOCATE DIR CACHE ENTRY
                     0000   162          <FIL$L_DIRMAX,0>,-              ;MAX OFFSET FOR DIR CACHE
                     0000   163          FIL$L_LBNOFF,-                  ;OFFSET IN BYTES TO BEGIN OF LBN CACHE
                     0000   164          FIL$L_LBNNXT,-                  ;NEXT OFFSET TO ALLOCATE LBN CACHE
                     0000   165          FIL$L_LBNMAX,-                  ;MAX OFFSET FOR LBN CACHE
                     0000   166          <FIL$A_IXFHDR,512>,-            ;INDEX FILE HEADER
                     0000   167          <FIL$C_SIZE,0>-                 ;START OF DIRECTORY CACHE
                     0000   168          >
                     0000                .SAVE   LOCAL_BLOCK
                     0000        FIL$W_CACHE_ID:
00000002             0000                .BLKB   2
00000004             0002                .BLKB   2
                     0004        FIL$L_DIROFF:
```

```
00000008  0004                .BLKB   4
          0008      FIL$L_DIRNXT:
0000000C  0008                .BLKB   4
          000C      FIL$L_DIRMAX:
          000C      FIL$L_LBNOFF:
00000010  000C                .BLKB   4
          0010      FIL$L_LBNNXT:
00000014  0010                .BLKB   4
          0014      FIL$L_LBNMAX:
00000018  0014                .BLKB   4
          0018      FIL$A_IXFHDR:
00000218  0018                .BLKB   512
          0218      FIL$C_SIZE:
          0000                .RESTORE
          0000   169 ;
          0000   170 ; OFFSETS INTO DIRECTORY CACHE ENTRIES
          0000   171 ;
          0000   172          $OFFSET 0,POSITIVE,<-
          0000   173          <FIL$A_DIR_FID,6>,-              ;DIRECTORY ID
          0000   174          <FIL$T_DIR_NAM,10>,-             ;COUNTED NAME OF DIRECTORY - NO ".DIR"
          0000   175          <FIL$Q_DIR_HDR,0>,-              ;DIRECTORY HEADER INFORMATION
          0000   176          <FIL$W_DIR_BKCNT,2>,-            ;SIZE IN BLOCKS OF DIRECTORY FILE
          0000   177          <FIL$B_DIR_LVL,1>,-              ;STRUCTURE LEVEL OF DIRECTORY
          0000   178          <,1>,-                           ;SPARE BYTE
          0000   179          FIL$L_DIR_LBN,-                  ;STARTING LBN OF DIRECTORY
          0000   180          FIL$L_DIR_BFOFF,-                ;OFFSET TO DIR LBN BUFFER
          0000   181          <FIL$W_DIR_BFCNT,2>,-            ;SIZE IN BLOCKS OF DIR LBN BUFFER
          0000   182          <FIL$A_DIR_OFID,6>,-             ;OUTPUT FILE ID FROM LOOKUP
          0000   183          <FIL$C_DIR_SIZE,0>-              ;SIZE OF DIRECTORY CACHE ENTRY
          0000   184          >
          0000                .SAVE   LOCAL_BLOCK
          0000                .PSECT $ABS$ ABS
00000000  0218                .=0
          0000      FIL$A_DIR_FID:
00000006  0000                .BLKB   6
          0006      FIL$T_DIR_NAM:
00000010  0006                .BLKB   10
          0010      FIL$Q_DIR_HDR:
          0010      FIL$W_DIR_BKCNT:
00000012  0010                .BLKB   2
          0012      FIL$B_DIR_LVL:
00000013  0012                .BLKB   1
00000014  0013                .BLKB   1
          0014      FIL$L_DIR_LBN:
00000018  0014                .BLKB   4
          0018      FIL$L_DIR_BFOFF:
0000001C  0018                .BLKB   4
          001C      FIL$W_DIR_BFCNT:
0000001E  001C                .BLKB   2
          001E      FIL$A_DIR_OFID:
00000024  001E                .BLKB   6
          0024      FIL$C_DIR_SIZE:
          0000                .RESTORE
          0000   185 ;
          0000   186 ; MAKE THESE GLOBAL SO THAT A CACHE SIZE CAN BE PROPERLY CALCULATED
          0000   187 ; THE CALCULATION IS:
          0000   188 ;
```

```
                     0000   189 ;       FIL$C_SIZE + (DIRCNT * FIIL$C_DIR_SIZE> + <LBNCNT * 512>
                     0000   190 ;
                     0000   191         .GLOBAL FIL$C_SIZE,FIL$C_DIR_SIZE
                     0000   192 ;
                     0000   193 ; DEFINE THE FOLLOWING WEAK REFERENCES, THEY NEED NOT BE PRESENT
                     0000   194 ;
                     0000   195         .WEAK   FIL$GQ_CACHE            ;DESCRIPTOR FOR FILEREAD CACHE
                     0000   196         .WEAK   FIL$GT_DDDEV            ;ASCIC DEFAULT DEVICE NAME STRING
                     0000   197         .WEAK   FIL$GT_TOPSYS           ;ASCIC TOP LEVEL SYSTEM DIRECTORY
                     0000   198 ;
                     0000   199 ; OWN STORAGE:
                     0000   200 ;
                     0000   201
                00000000    202         .psect  code, nowrt, exe, rd, shr, long ;                       [02]
                     0000   203
                     0000   204 FIL_GQ_CACHE:
           00000000'  0000   205         .ADDRESS FIL$GQ_CACHE
                     0004   206 FIL_GT_DDDEV:
           00000000'  0004   207         .ADDRESS FIL$GT_DDDEV
                     0008   208 FIL_GT_TOPSYS:
           00000000'  0008   209         .ADDRESS FIL$GT_TOPSYS
```

```
                        211          .SBTTL  FIL$OPENFILE - RETURN FILE HEADER AND STATISTICS BLOCK
                        212 ;++
                        213 ; FUNCTIONAL DESCRIPTION:
                        214 ;
                        215 ;       THE OPENFILE ROUTINE ACCEPTS A FULL FILE NAME IN THE FORMAT
                        216 ;       DEV:[DIR]FILE.TYP;VERSION.
                        217 ; IT ASSIGNS AND RETURNS A CHANNEL, READS THE FILE HEADER, RETURNS THE
                        218 ; STATISTICS BLOCK, AND OPTIONALLY RETURNS THE RETRIEVAL POINTERS IN
                        219 ; A NORMALIZED (LONG WORD COUNT, LONG WORD LBN) FORMAT.
                        220 ; THE DIRECTORY MAY BE IN ANY OF THE STANDARD FORMATS:
                        221 ;       [10,40], [010040], [ABCDEFGHI], OR WITH < AND > REPLACING [ AND ].
                        222 ; VERSION MAY BE ZERO IN WHICH CASE THE HIGHEST VERSION IS FOUND
                        223 ;
                        224 ; CALLING SEQUENCE:
                        225 ;
                        226 ;       CALLG   ARGLIST,FIL$OPENFILE
                        227 ;
                        228 ; INPUT PARAMETERS:
                        229 ;
                        230 ;       CHANADR(AP)     =       ADDRESS OF RPB                                  [02]
                        231 ;       FILNAM(AP)              ADDRESS OF 2 LONG WORD FILE NAME STRING DESCRIPTOR
                        232 ;                               1 - SIZE OF STRING
                        233 ;                               2 - ADDRESS OF STRING
                        234 ;                                       DB1:[10,40]FILTST.EXE
                        235 ;       IXFHDR(AP)              ADDRESS OF 512 BYTE BUFFER TO BE USED FOR
                        236 ;                                       THE INDEX FILE HEADER
                        237 ;       FILHDR(AP)              ADDRESS OF 512 BYTE BUFFER TO RETURN FILE HEADER
                        238 ;       STATBLK(AP)             ADDRESS OF 2 LONG WORD BLOCK IN WHICH THE
                        239 ;                               FOLLOWING WILL BE RETURNED
                        240 ;                               1 - LOGICAL BLOCK NUMBER OF FIRST BLOCK OF
                        241 ;                                       FILE OR 0 IF FILE IS NOT CONTIGUOUS
                        242 ;                               2 - SIZE OF FILE IN BLOCKS
                        243 ;       RTRVPTRLEN(AP)  =       ADDRESS TO RETURN THE NUMBER OF
                        244 ;                               BYTES OF RETRIEVAL POINTERS STORED
                        245 ;                               ***** OPTIONAL PARAMETER *****
                        246 ;       RTRVPTRBUF(AP)  =       ADDRESS OF RETRIEVAL POINTER
                        247 ;                               BUFFER DESCRIPTOR.  THIS PARAMETER
                        248 ;                               IS PRESENT IF AND ONLY IF
                        249 ;                               RTRVPTRLEN IS PRESENT.
                        250 ;                               THE RETRIEVAL POINTERS ARE RETURNED IN
                        251 ;                               THE FORM 32 BIT BLOCK COUNT, 32 BIT LBN
                        252 ;                               A ZERO BUFFER DESCRIPTOR ADDRESS OR A
                        253 ;                               ZERO BUFFER ADDRESS MEANS DON'T
                        254 ;                               RETURN RETRIEVAL POINTER INFO
                        255 ;
                        256 ; IMPLICIT INPUTS:
                        257 ;
                        258 ;       NONE
                        259 ;
                        260 ; OUTPUT PARAMETERS:
                        261 ;
                        262 ;       R0 = SYSTEM STATUS CODE
                        263 ;
                        264 ; IMPLICIT OUTPUTS:
                        265 ;
                        266 ;       NONE
                        267 ;
```

```
                     000C   268 ; COMPLETION CODES:
                     000C   269 ;
                     000C   270 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                     000C   271 ;       SS$_NOSUCHFILE          FAILED TO FIND DIRECTORY OR FILE
                     000C   272 ;       SS$_BADFILENAME         SYNTAX ERROR IN DIRECTORY OR FILE NAME STRING
                     000C   273 ;
                     000C   274 ; THE FOLLOWING COMPLETION CODES INDICATE FILE STRUCTURE PROBLEMS
                     000C   275 ;
                     000C   276 ;       SS$_BADCHKSUM           CHECKSUM ERROR IN HOME BLOCK, INDEX FILE HEADER
                     000C   277 ;                               DIRECTORY FILE HEADER OR FILE HEADER
                     000C   278 ;       SS$_BADFILEHDR          FILE HEADER CONSISTENCY CHECK FAILED FOR
                     000C   279 ;                               INDEX FILE, DIRECTORY FILE, OR DESIRED FILE
                     000C   280 ;       SS$_FILESTRUCT          HOME BLOCK INDICATES THAT THIS VOLUME
                     000C   281 ;                               CONTAINS A NON-SUPPORTED FILE STRUCTURE
                     000C   282 ;                               OR POSSIBLY THE HOMEBLOCK IS GARBAGE
                     000C   283 ;
                     000C   284 ; SIDE EFFECTS:
                     000C   285 ;
                     000C   286 ;       NONE
                     000C   287 ;
                     000C   288 ; EQUATED SYMBOLS
                     000C   289 ;
                     000C   290 ;    OFFSETS FROM AP
                     000C   291 ;
00000000             000C   292          ARGCNT          = 0
00000004             000C   293          CHANADR         = 4
00000008             000C   294          FILNAM          = 8
0000000C             000C   295          IXFHDR          = 12
00000010             000C   296          FILHDR          = 16
00000014             000C   297          STATBLK         = 20
00000018             000C   298          RTRVPTRLEN      = 24            ; OPTIONAL
0000001C             000C   299          RTRVPTRBUF      = 28            ; PRESENT IF AND ONLY IF RTRVPTRLEN IS
                     000C   300 ;
                     000C   301 ;    OFFSETS FROM FP
                     000C   302 ;
                     000C   303          $OFFSET 0,NEGATIVE,<-
                     000C   304          <FID,6>,-                       ;3 WORD FILE IDENTIFIER
                     000C   305          <DIRNAM,46>,-                   ;DIRECTORY NAME AREA
                     000C   306          <NAMBLK,40>,-                   ;5 WORD NAME BLOCK AREA
                     000C   307          <NAMDSC,12>,-                   ;NAME DESCRIPTOR AREA
                     000C   308          <SCRATCHSIZE,0>-                ;SIZE OF SCRATCH AREA
                     000C   309          >
                     000C                .SAVE   LOCAL_BLOCK
                     000C                .PSECT $ABS$ ABS
00000000             0218                .=0
FFFFFFFA             0000                .BLKB   -6
                     FFFA        FID:
FFFFFFCC             FFFA                .BLKB   -46
                     FFCC        DIRNAM:
FFFFFFA4             FFCC                .BLKB   -40
                     FFA4        NAMBLK:
FFFFFF98             FFA4                .BLKB   -12
                     FF98        NAMDSC:
                     FF98        SCRATCHSIZE:
                 0000000C                .RESTORE
                     000C   310 ;
                     000C   311 ; THE FILE DESCRIPTION ON THE STACK LOOKS AS FOLLOWS:
```

```
                                000C   312 ;
                                000C   313 ;       +-----------------------------+
                                000C   314 ;       |  DIRECTORY NAME COUNT       |   :NAMDSC
                                000C   315 ;       |-----------------------------|
                                000C   316 ;       | ADDRESS OF DIRECTORY NAME   |
                                000C   317 ;       +-----------------------------+
                                000C   318 ;       |    ADDRESS OF NAMBLK        |
                                000C   319 ;       +-----------------------------+
                                000C   320 ;       |                             |   :NAMBLK
                                000C   321 ;       +-----------------------------+
                                000C   322 ;       |                             |
                                000C   323 ;       +-----------------------------+
                                000C   324 ;       |                             |
                                000C   325 ;       |                             |
                                000C   326 ;       +-----------------------------+
                                000C   327 ;       |                             |   :DIRNAM
                                000C   328 ;       |      ASCII STRING OF        |
                                000C   329 ;       |       DIRECTORY FILE        |
                                000C   330 ;       |      NAME.TYPE;VERSION      |
                                000C   331 ;       |                             |
                                000C   332 ;       +-----------------------------+
                                000C   333 ;       |          FILE ID            |   :FILID
                                000C   334 ;       |          NUMBER             |
                                000C   335 ;       +-----------------------------+
                                000C   336 ;       | RELATIVE VOLUME NUMBER      |
                                000C   337 ;       +-----------------------------+
                                000C   338 ;
                                000C   339 ;--
                                000C   340
                                000C   341          .ENABL  LSB
                                000C   342
                                000C   343 FIL$OPENFILE::
                                000C   344 ;        .WORD   ^M<R2,R3,R4,R5,R6,R7,R11>                                [02]
                                000C   345 ;        BSBW    ASSIGN_DEV                ;ASSIGN THE DEVICE OONCE IN CALLER'S MODE
                                000C   346 ;        BLBS    R0,OPENFILE_2             ;BRANCH IF SUCCESSFUL                [02]
                                000C   347 ;        RET                                                              [02]
                                000C   348 OPENFILE_1:
                          08FC  000C   349          .WORD   ^M<R2,R3,R4,R5,R6,R7,R11>
                                000E   350 OPENFILE_2:
5E   00000068 8F    C2          000E   351          SUBL    #-SCRATCHSIZE,SP          ;RESERVE SCRATCH STORAGE
   A0 AD    A4 AD    DE          0015   352          MOVAL   NAMBLK(FP),NAMDSC+8(FP) ;SET ADDRESS OF NAME BLOCK
                                001A   353 ;
                                001A   354 ; IF CACHE DESCRIPTOR EXISTS AND IS IN SYSTEM SPACE, THEN WE
                                001A   355 ; HAD BETTER BE IN KERNEL MODE TO USE THE CACHE.
                                001A   356 ;
   5B    FFE2 CF    D0          001A   357          MOVL    W^FIL_GQ_CACHE,R11        ;IS CACHE IN SYSTEM SPACE?
                02    14        001F   358          BGTR    10$                       ;BRANCH IF DESCRIPTOR PRESENT
                                0021   359                                            ;AND NOT IN SYSTEM SPACE
                10    13        0021   360          BEQL    20$                       ;BRANCH IF NO DESCRIPTOR PRESENT
                                0023   361 ;        MOVPSL  R0                        ;GET PSL                         [02]
                                0023   362 ;        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R0,R0 ;FETCH CURRENT MODE    [02]
                                0023   363 ;        BEQL    10$                       ;BRANCH IF ALREADY IN KERNEL MODE [02]
                                0023   364 ;        $CMKRNL_S B^OPENFILE_1,(AP)       ;CALL THIS PROCEDURE IN KERNEL MODE [02]
                                0023   365 ;        CMPL    R0,#SS$_NOPRIV            ;ASSUME NOPRIV MEANS WE COULDN'T [02]
                                0023   366 ;                                          ;GET IN TO KERNEL MODE           [02]
                                0023   367 ;        BEQL    15$                                                      [02]
                                0023   368 ;        RET                                                              [02]
```

```
                5B   00000004'EF   D0   0023   369 10$:    MOVL    FIL$GQ_CACHE+4,R11      ;IS THE CACHE ENABLED?
                               07   13   002A   370         BEQL    20$                     ;BRANCH IF NOT
                           01  6B   B1   002C   371         CMPW    FIL$W_CACHE_ID(R11),#FIL$C_CACHE_ID ;CORRECT VERSION OF CACHE?
                               02   13   002F   372         BEQL    20$                     ;BRANCH IF YES
                               5B   D4   0031   373 15$:    CLRL    R11                     ;DISABLE THE CACHE
                57   00000000'EF   DE   0033   374 20$:    MOVAL   FIL$GT_DDSTRING,R7      ;ADDRESS OF COUNTED STRING
                           56   87   9A   003A   375         MOVZBL  (R7)+,R6                ;GET BYTE COUNT
                               57   D6   003D   376         INCL    R7                      ;STEP OVER BRACKET
                           56  02   C2   003F   377         SUBL    #2,R6                   ;DON'T COUNT THE BRACKETS
                                    0042   378 ;
                                    0042   379 ; GET FILE NAME STRING, AND STRIP DEVICE OFF IF PRESENT
                                    0042   380 ;
                     50    08 AC   D0   0042   381         MOVL    FILNAM(AP),R0           ;ADDRESS OF FILE NAME DESCRIPTOR
                               27   13   0046   382         BEQL    32$                     ;BRANCH IF NO NAME SPECIFIED
                           52  60   7D   0048   383         MOVQ    (R0),R2                 ;R2 = SIZE, R3 = ADDRESS
                63         52  3A   3A   004B   384         LOCC    #^A/:/,R2,(R3)          ;DEVICE NAME PRESENT?
                               07   13   004F   385         BEQL    25$                     ;BRANCH IF NOT
                     53    01 A1   9E   0051   386         MOVAB   1(R1),R3                ;ADDRESS BEYOND ":"
                           52  70   9E   0055   387         MOVAB   -(R0),R2                ;REMAINING SIZE
                                    0058   388 ;
                                    0058   389 ; SEE IF DIRECTORY SPECIFIED IN THE FILE NAME STRING
                                    0058   390 ;
                     63    5B 8F   91   0058   391 25$:    CMPB    #^A/[/,(R3)             ;DIRECTORY DELIMITER?
                               05   13   005C   392         BEQL    30$                     ;BRANCH IF YES
                           63  3C   91   005E   393         CMPB    #^A/</,(R3)             ;ALTERNATE CHARACTER
                               1D   12   0061   394         BNEQ    40$                     ;BRANCH IF NO DIRECTORY SPECIFIED
                50         83  02   81   0063   395 30$:    ADDB3   #2,(R3)+,R0             ;SCAN FOR MATCHING BRACKET ] OR >
                               52   D7   0067   396         DECL    R2                      ;ADJUST SIZE AND ADR OF STRING
                63         52  50   3A   0069   397         LOCC    R0,R2,(R3)              ;SCAN FOR CLOSE BRACKET
                               03   12   006D   398         BNEQ    35$                     ;BRANCH IF FOUND IT
                         03EB   31   006F   399 32$:    BRW     BADFILNAM               ;BAD FILE NAME IF NO CLOSE BRACKET
                     57    53   D0   0072   400 35$:    MOVL    R3,R7                   ;ADDRESS OF DIRECTORY NAME
                56   51    53   C3   0075   401         SUBL3   R3,R1,R6                ;SIZE OF DIRECTORY NAME
                     52    70   9E   0079   402         MOVAB   -(R0),R2                ;SIZE REMAINING SKIP CLOSE BRACKET
                  53    01 A1   DE   007C   403         MOVAL   1(R1),R3                ;ADR OF REMAINING STRING BEYOND CLOSE BRACKE
                                    0080   404 ;
                                    0080   405 ; SET UP COMMON ARGUMENT LIST FOR MOUNT, FINDFILID, RDCHKFILHDR
                                    0080   406 ;
                                    0080   407 ;       +-----------------------------+
                                    0080   408 ;       |       ARGUMENT COUNT        |   : AP
                                    0080   409 ;       +-----------------------------+
                                    0080   410 ;       |       CHANNEL NUMBER        |
                                    0080   411 ;       +-----------------------------+
                                    0080   412 ;       |      NAME DESCRIPTOR        |
                                    0080   413 ;       +-----------------------------+
                                    0080   414 ;       | INDEX FILE HEADER BUF ADR   |
                                    0080   415 ;       +-----------------------------+
                                    0080   416 ;       |  FILE HEADER BUFFER ADDR    |
                                    0080   417 ;       +-----------------------------+
                                    0080   418 ;       | ADDR OF STATISTICS BLOCK    |
                                    0080   419 ;       +-----------------------------+
                                    0080   420 ;       | ADDRESS OF FILE ID BLOCK    |
                                    0080   421 ;       +-----------------------------+
                                    0080   422 ;       |  ADDR OF RTRV PTR LENGTH    |
                                    0080   423 ;       +-----------------------------+
                                    0080   424 ;       | ADDR OF RTRV PTR BUF DSCR   |
                                    0080   425 ;       +-----------------------------+
```

```
                                0080    426 ;
                          7E    7C 0080    427 40$:    CLRQ    -(SP)                           ;ASSUME NO RETRIEVAL POINTERS REQUESTED
                       07 6C    D1 0082    428         CMPL    ARGCNT(AP),#RTRVPTRBUF/4        ;RETRIEVAL POINTER PARAMETERS PRESENT?
                          04    19 0085    429         BLSS    45$                             ;BRANCH IF NOT
                    6E 18 AC    7D 0087    430         MOVQ    RTRVPTRLEN(AP),(SP)             ;PUT RTRV PTR PARAMS IN LIST
                       FA AD    DF 008B    431 45$:    PUSHAL  FID(FP)                         ;ADDRESS OF FILE ID
                    7E 10 AC    7D 008E    432         MOVQ    FILHDR(AP),-(SP)                ;PUSH STATBLK ADR, FILHDR ADR
                       0C AC    DD 0092    433         PUSHL   IXFHDR(AP)                      ;INDEX FILE HEADER ADDRESS
                       98 AD    DF 0095    434         PUSHAL  NAMDSC(FP)                      ;ADR OF 3 LONG WORD NAME DESCRIPTOR
                       04 AC    DD 0098    435         PUSHL   CHANADR(AP)                     ;CHANNEL TO USE, LONG WORD FOR BOOTING  [02]
                          06    DD 009B    436         PUSHL   #6                              ;PARAMETER COUNT
                          5B    D5 009D    437         TSTL    R11                             ;CACHE ENABLED?
                          07    13 009F    438         BEQL    50$
                0C AE 18 AB     DE 00A1    439         MOVAL   FIL$A_IXFHDR(R11),IXFHDR(SP) ;USE CACHED INDEX FILE HEADER
                          08    11 00A6    440         BRB     60$                             ;AND SKIP THE MOUNT
                 025D'CF  6E    FA 00A8    441 50$:    CALLG   (SP),W^FIL$MOUNT                ;"MOUNT THE VOLUME" (READ HOME
                                   00AD    442                                                 ;BLOCK, INDEX FILE HEADER, GET
                                   00AD    443                                                 ;STRUCTURE LEVEL OF VOLUME)
                       61 50    E9 00AD    444         BLBC    R0,100$                         ;BRANCH IF ERROR
                                   00B0    445 ;
                                   00B0    446 ; SET UP FOR THE DIRECTORY LOOK UP
                                   00B0    447 ;
                    FA AD    04    B0 00B0    448 60$:    MOVW    #FID$C_MFD,FID(FP)              ;MFD FILE NUMBER
                    FC AD    04    D0 00B4    449         MOVL    #FID$C_MFD,FID+2(FP)            ;MFD FILE SEQUENCE NO., RVN = 0
                       FF4C  CF    D5 00B8    450         TSTL    W^FIL_GT_TOPSYS                 ;TOP LEVEL SYSTEM DIRECTORY PRESENT?
                             1A    13 00BC    451         BEQL    70$                             ;BRANCH IF NOT
                51   00000000'EF    DE 00BE    452         MOVAL   FIL$GT_TOPSYS,R1                ;GET ADDRESS OF TOP LEVEL DIR STRING
                          50    81    9A 00C5    453         MOVZBL  (R1)+,R0                        ;GET SIZE TO R0, ADR TO R1
                             0E    13 00C8    454         BEQL    70$                             ;BRANCH IF NONE SPECIFIED
                          7E    56    7D 00CA    455         MOVQ    R6,-(SP)                        ;SAVE DIRECTORY STRING DESCRIPTOR
                          56    50    7D 00CD    456         MOVQ    R0,R6                           ;TREAT TOPSYS LIKE DIR STRING
                                0116    30 00D0    457         BSBW    FORMDIRSTRING                   ;FORM THE DIRECTORY NAME
                          56    8E    7D 00D3    458         MOVQ    (SP)+,R6                        ;RESTORE READ DIRECTORY DESCRIPTOR
                             03    11 00D6    459         BRB     75$
                                0116    30 00D8    460 70$:    BSBW    FORMDIRSTRING                   ;GET NEXT DIRECTORY TO LOOKUP
                       98 AD    50    7D 00DB    461 75$:    MOVQ    R0,NAMDSC(FP)                   ;STORE DESCRIPTOR OF ITS NAME
                             6E    DF 00DF    462 80$:    PUSHAL  (SP)                            ;REAL ADDRESS OF ARGUMENT LIST
                             5B    DD 00E1    463         PUSHL   R11                             ;CACHE ADDRESS IF ANY
                 02F1'CF     02    FB 00E3    464         CALLS   #2,W^FIL$FINDFILID              ;FIND THE FILE ID
                          26 50    E9 00E8    465         BLBC    R0,100$                         ;BRANCH IF ERROR
                             3F    BB 00EB    466         PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; SAVE THE REGISTER
          A4 AD    00 8F    28    28 00ED    467         MOVC3   #<DIRNAM-NAMBLK>,#0,NAMBLK(FP) ;REINIT NAME BLOCK
                             3F    BA 00F3    468         POPR    #^M<R0,R1,R2,R3,R4,R5>          ; RESTORE REGISTERS
                             56    D5 00F5    469         TSTL    R6                              ;ANY MORE DIRECTORY NAMES?
                             DF    14 00F7    470         BGTR    70$                             ;BRANCH IF YES, LOOKUP THE NEXT
                       98 AD    52    7D 00F9    471         MOVQ    R2,NAMDSC(FP)                   ;DESCRIPTOR FOR FILE TO LOOKUP
                             52    D4 00FD    472         CLRL    R2                              ;STOP THE LOOKUP LOOP
                          98 AD    D5 00FF    473         TSTL    NAMDSC(FP)                      ;ALREADY DONE?
                             DB    14 0102    474         BGTR    80$                             ;BRANCH IF NO, DO THE LAST ONE
                          07    6C    D1 0104    475 85$:    CMPL    ARGCNT(AP),#RTRVPTRBUF/4        ;RETRIEVAL POINTERS DESIRED?
                             03    19 0107    476         BLSS    90$                             ;BRANCH IF NOT
                          6E    02    C0 0109    477         ADDL    #2,(SP)                         ;ADDITIONAL ARGUMENTS ARE PRESENT
                 05FA'CF     6E    FA 010C    478 90$:    CALLG   (SP),W^FIL$RDCHKFILHDR          ;READ AND CHECK FILE HEADER
                             04    11 0111    479 100$:   RET
                                   0112    480
                                   0112    481         .DSABL  LSB
```

```
                     0112   483          .SBTTL  FIL$CACHE_INIT - INIT FILEREAD CACHE
                     0112   484 ;**
                     0112   485 ; FUNCTIONAL DESCRIPTION:
                     0112   486 ;
                     0112   487 ;       CACHE_INIT PERFORMS THE INITIALIZATION FOR THE FILEREAD CACHE
                     0112   488 ;
                     0112   489 ; CALLING SEQUENCE:
                     0112   490 ;
                     0112   491 ;       CALLG   ARGLIST,FIL$CACHE_INIT
                     0112   492 ;
                     0112   493 ; INPUT PARAMETERS:
                     0112   494 ;
                     0112   495 ;       CHANADR(AP)             ADDRESS TO RETURN LONG WORD CHANNEL
                     0112   496 ;       FILNAM(AP)              ADDRESS OF DEVICE NAME STRING DESCRIPTOR
                     0112   497 ;                               THE DEVICE NAME MUST CONTAIN THE ":"
                     0112   498 ;                               IF THE ADDRESS IS 0, THE STRING IS NULL,
                     0112   499 ;                               OR THE NAME DOES NOT CONTAIN A ":", THE
                     0112   500 ;                               DEFAULT DEVICE NAME IS USED
                     0112   501 ;       CACHE_SIZE(AP)          SIZE IN BYTES OF FILEREAD CACHE
                     0112   502 ;       CACHE_ADR(AP)           ADDRESS OF FILEREAD CACHE
                     0112   503 ;       DIR_CACHE_CNT(AP)       NUMBER OF DIRECTORY CACHE ENTRIES
                     0112   504 ;       LBN_CACHE_CNT(AP)       NUMBER OF LBN CACHE ENTRIES
                     0112   505 ;
                     0112   506 ; IMPLICIT INPUTS:
                     0112   507 ;
                     0112   508 ;       NONE
                     0112   509 ;
                     0112   510 ; OUTPUT PARAMETERS:
                     0112   511 ;
                     0112   512 ;       R0 = ALWAYS SUCCESSFUL STATUS CODE
                     0112   513 ;
                     0112   514 ; IMPLICIT OUTPUTS:
                     0112   515 ;
                     0112   516 ;       FIL$GQ_CACHE QUAD WORD FILLED IN WITH SIZE AND ADDRESS OF CACHE
                     0112   517 ;
                     0112   518 ; COMPLETION CODES:
                     0112   519 ;
                     0112   520 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                     0112   521 ;
                     0112   522 ; SIDE EFFECTS:
                     0112   523 ;
                     0112   524 ;       NONE
                     0112   525 ;
                     0112   526 ;
                     0112   527 ; EQUATED SYMBOLS, OFFSETS FROM AP
                     0112   528 ;
00000004             0112   529          CHANADR         =       4
00000008             0112   530          FILNAM          =       8
0000000C             0112   531          CACHE_SIZE      =       12
00000010             0112   532          CACHE_ADR       =       16
00000014             0112   533          DIR_CACHE_CNT   =       20
00000018             0112   534          LBN_CACHE_CNT   =       24
                     0112   535 ;
                     0112   536 ;--
                     0112   537 FIL$CACHE_INIT::
                 0C3C 0112   538          .WORD   ^M<R2,R3,R4,R5,R10,R11>
                     0114   539
```

```
                                  0114   540          ASSUME  CACHE_SIZE+4 EQ CACHE_ADR
                5A    0C AC     7D  0114   541          MOVQ    CACHE_SIZE(AP),R10      ;R10=SIZE, R11=ADR
          50    5A    00000218 8F  C3  0118   542          SUBL3   #FIL$C_SIZE,R10,R0      ;BYTES LEFT FOR DIR AND LBN CACHES
                            60   19  0120   543          BLSS    100$                    ;BRANCH IF NOT ENOUGH CACHE SPACE
                      6B    01   B0  0122   544          MOVW    #FIL$C_CACHE_ID,FIL$W_CACHE_ID(R11) ;SET CACHE_ID
                                  0125   545                                          ;ALLOWS MOVING CACHES BETWEEN FILEREAD'S
            04 AB     00000218 8F  D0  0125   546          MOVL    #FIL$C_SIZE,FIL$L_DIROFF(R11) ;BEGINNING OF DIR CACHE
            08 AB     00000218 8F  D0  012D   547          MOVL    #FIL$C_SIZE,FIL$L_DIRNXT(R11) ;NEXT AVAILABLE SLOT IN DIR CACHE
               51     14 AC    24  C5  0135   548          MULL3   #FIL$C_DIR_SIZE,DIR_CACHE_CNT(AP),R1 ;BYTE COUNT FOR DIR CACHE
                      50    51   C2  013A   549          SUBL    R1,R0                   ;BYTE COUNT LEFT FOR LBN CACHE
                            43   19  013D   550          BLSS    100$                    ;BRANCH IF NOT ENOUGH SPACE
       0C AB    51    00000218 8F  C1  013F   551          ADDL3   #FIL$C_SIZE,R1,FIL$L_DIRMAX(R11) ;END OF DIR CACHE
                                  0148   552
                                  0148   553          ASSUME  FIL$L_DIRMAX EQ FIL$L_LBNOFF
              10 AB     0C AB    D0  0148   554          MOVL    FIL$L_LBNOFF(R11),FIL$L_LBNNXT(R11) ;NEXT LBN ENTRY TO ALLOCATE
               51     18 AC    09  78  014D   555          ASHL    #9,LBN_CACHE_CNT(AP),R1 ;BYTE COUNT IN LBN CACHE
                      50    51   D1  0152   556          CMPL    R1,R0                   ;ENOUGH ROOM FOR WHOLE LBN CACHE
                            08   15  0155   557          BLEQ    20$                     ;BRANCH IF YES
          51    50    000001FF 8F  CB  0157   558          BICL3   #^X1FF,R0,R1            ;USE WHAT IS LEFT TRUNCATED
             14 AB     10 AB    51  C1  015F   559 20$:     ADDL3   R1,FIL$L_LBNNXT(R11),FIL$L_LBNMAX(R11) ;END OF LBN CACHE
                          003D  30  0165   560          BSBW    ASSIGN_DEV              ;ASSIGN THE DEVICE
                        17 50   E9  0168   561          BLBC    R0,100$                 ;BRANCH IF ERROR
                        18 AB   DF  016B   562          PUSHAL  FIL$A_IXFHDR(R11)       ;ADDRESS TO READ INDEX FILE HEADER
                           7E   D4  016E   563          CLRL    -(SP)                   ;UNUSED PARAMETER
                        04 BC   DD  0170   564          PUSHL   @CHANADR(AP)            ;CHANNEL JUST ASSIGNED
                   025D'CF    03   FB  0173   565          CALLS   #3,W^FIL$MOUNT          ;MOUNT THE VOLUME, RETURN INDEX FILE HDR
                      07 50   E9  0178   566          BLBC    R0,100$                 ;BRANCH IF ERROR
          00000000'EF    5A   7D  017B   567          MOVQ    R10,FIL$GQ_CACHE        ;SAVE DESCRIPTOR OF CACHE
                   50    01   D0  0182   568 100$:     MOVL    S^#SS$_NORMAL,R0
                              04  0185   569          RET
```

```
                                0186   571          .SBTTL  FIL$CACHE_TRUNC - TRUNCATE FILEREAD CACHE
                                0186   572 ;++
                                0186   573 ; FUNCTIONAL DESCRIPTION:
                                0186   574 ;
                                0186   575 ;       CACHE_TRUNC TRUNCATES THE FILEREAD CACHE AND MAKES IT IMPOSSIBLE
                                0186   576 ; TO ADD MORE DIRECTORY CACHE OR DIRECTORY LBN ENTRIES TO IT.  IN EFFECT
                                0186   577 ; THIS ROUTINE TURNS THE CACHE INTO A READ-ONLY DATA BASE.
                                0186   578 ;
                                0186   579 ; CALLING SEQUENCE:
                                0186   580 ;
                                0186   581 ;       CALLG   ARGLIST,FIL$CACHE_TRUNC
                                0186   582 ;
                                0186   583 ; INPUT PARAMETERS:
                                0186   584 ;
                                0186   585 ;       NONE
                                0186   586 ;
                                0186   587 ; IMPLICIT INPUTS:
                                0186   588 ;
                                0186   589 ;       FIL$GQ_CACHE            DESCRIPTOR FOR THE CACHE
                                0186   590 ;
                                0186   591 ; OUTPUT PARAMETERS:
                                0186   592 ;
                                0186   593 ;       R0 = ALWAYS SUCCESSFUL STATUS CODE
                                0186   594 ;
                                0186   595 ; IMPLICIT OUTPUTS:
                                0186   596 ;
                                0186   597 ;       FIL$GQ_CACHE FILLED IN WITH ALTERED SIZE OF CACHE
                                0186   598 ;
                                0186   599 ; COMPLETION CODES:
                                0186   600 ;
                                0186   601 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                                0186   602 ;
                                0186   603 ; SIDE EFFECTS:
                                0186   604 ;
                                0186   605 ;       NONE
                                0186   606 ;
                                0186   607 ; EQUATED SYMBOLS
                                0186   608 ;
                                0186   609 ;
                                0186   610 ;--
                                0186   611
                                0186   612 FIL$CACHE_TRUNC::
                        0000    0186   613          .WORD   0
    50   00000004'EF    D0      0188   614          MOVL    FIL$GQ_CACHE+4,R0       ;ADDRESS OF THE CACHE
       0C A0    08 A0  D0      018F   615          MOVL    FIL$L_DIRNXT(R0),FIL$L_DIRMAX(R0) ;NO NEW DIRECTORY CACHE ENTRIES
       14 A0    10 A0  D0      0194   616          MOVL    FIL$L_LBNNXT(R0),FIL$L_LBNMAX(R0) ;NO MORE LBN BUFFERS
00000000'EF     10 A0  D0      0199   617          MOVL    FIL$L_LBNNXT(R0),FIL$GQ_CACHE   ;SET NEW SIZE OF CACHE
            50    01   D0      01A1   618          MOVL    S^#SS$_NORMAL,R0
                       04      01A4   619          RET
```

```
                                01A5   621          .SBTTL  ASSIGN_DEV - ASSIGN A DEVICE AND RETURN A CHANNEL
                                01A5   622 ;++
                                01A5   623 ; FUNCTIONAL DESCRIPTION:
                                01A5   624 ;
                                01A5   625 ;       ASSIGN THE GIVEN DEVICE AND RETURN A CHANNEL NUMBER
                                01A5   626 ;       FOR BOOTING THIS VALUE IS A LONG WORD
                                01A5   627 ;
                                01A5   628 ; INPUTS:
                                01A5   629 ;
                                01A5   630 ;       FILNAM(AP)      = ADDRESS OF STRING DESCRIPTOR
                                01A5   631 ;                         IF DEVICE NAME IS PRESENT IT MUST HAVE ":"
                                01A5   632 ;       CHANADR(AP)     = ADDRESS TO RETURN LONG WORD OF CHANNEL NUMBER
                                01A5   633 ;
                                01A5   634 ; OUTPUTS:
                                01A5   635 ;
                                01A5   636 ;       R0,R1 ALTERED
                                01A5   637 ;       R2 = SIZE OF DEVICE NAME STRING (DEFAULT IF NOT IN NAME STRING)
                                01A5   638 ;            MAYBE 0 IF DEFAULT DEVICE STRING WAS NOT PRESENT
                                01A5   639 ;            THIS IS THE CASE WHEN BOOTSTRAPPING.
                                01A5   640 ;       R3 = ADDRESS OF DEVICE NAME STRING
                                01A5   641 ;            0 IF DEFAULT DEVICE STRING WAS NOT PRESENT
                                01A5   642 ;
                                01A5   643 ;--
                                01A5   644
                                01A5   645 ASSIGN_DEV:
                    50    D4    01A5   646          CLRL    R0                      ;ASSUME NULL DEFAULT DEVICE STRING
                FE59 CF   D5    01A7   647          TSTL    W^FIL_GT_DDDEV          ;ADDRESS OF DEFAULT DEVICE COUNTED STRING
                    0A    13    01AB   648          BEQL    10$                     ;BRANCH IF NO DEFAULT DEVICE STRING
            51  00000000'EF    DE    01AD   649          MOVAL   FIL$GT_DDDEV,R1         ;GET THE ADDRESS
                    50    81    9A    01B4   650          MOVZBL  (R1)+,R0                ;SIZE OF DEFAULT DEVICE STRING
                    03    BB    01B7   651 10$:     PUSHR   #^M<R0,R1>              ;PUSH DEFAULT DEVICE DESCRIPTOR
                    52    7C    01B9   652          CLRQ    R2                      ;ASSUME NULL STRING DESCRIPTOR
                 50   08 AC    D0    01BB   653          MOVL    FILNAM(AP),R0           ;ADDRESS OF STRING DESCRIPTOR
                    11    13    01BF   654          BEQL    20$                     ;BRANCH IF NO NAME GIVEN
                    52    60    7D    01C1   655          MOVQ    (R0),R2                 ;R2 = SIZE, R3 = ADR OF FILE NAME STRING
                63  52    3A    3A    01C4   656          LOCC    #^A/:/,R2,(R3)          ;ANY DEVICE SPECIFIED?
                    08    13    01C8   657          BEQL    20$                     ;BRANCH IF NONE
                04 AE   53    D0    01CA   658          MOVL    R3,4(SP)                ;ADDRESS OF DEVICE NAME
                6E  51    53    C3    01CE   659          SUBL3   R3,R1,(SP)              ;SIZE OF DEVICE NAME STRING
                    50    5E    D0    01D2   660 20$:     MOVL    SP,R0                   ;ADDRESS OF DEV NAME DESCRIPTOR
                                01D5   661 ;        $ASSIGN_S (R0),aCHANADR(AP)     ;ASSIGN THE CHANNEL
                    0C    BA    01D5   662          POPR    #^M<R2,R3>              ;GET DEVICE NAME SIZE AND ADDRESS
                          05    01D7   663          RSB
```

```
                        01D8   665          .SBTTL  STORE3DIGITS - STORE 3 ASCII DIGITS
                        01D8   666 ;++
                        01D8   667 ; FUNCTIONAL DESCRIPTION:
                        01D8   668 ;
                        01D8   669 ;        STORE 3 DIGITS OF DIRECTORY STRING
                        01D8   670 ;
                        01D8   671 ; CALLING SEQUENCE:
                        01D8   672 ;
                        01D8   673 ;        BSBB    STORE3DIGITS
                        01D8   674 ;
                        01D8   675 ; INPUT:
                        01D8   676 ;
                        01D8   677 ;        R0 = NO. OF DIGITS TO PUT IN STRING
                        01D8   678 ;        R1 = ADDRESS + 1 OF RIGHT MOST DIGIT
                        01D8   679 ;        R2 = ADDRESS AT WHICH TO STORE 3 DIGITS
                        01D8   680 ;
                        01D8   681 ; OUTPUTS:
                        01D8   682 ;
                        01D8   683 ;        NONE
                        01D8   684 ;
                        01D8   685 ;--
                        01D8   686
                        01D8   687 STORE3DIGITS:
              03   50   D1  01D8   688          CMPL    R0,#3                   ;3 DIGITS OR LESS SPECIFIED?
                   03   15  01DB   689          BLEQ    5$                      ;YES. BRANCH.
                 027D   31  01DD   690          BRW     BADFILNAM               ;NO. DIRECTORY STRING BAD. EXIT
                        01E0   691                                             ;WITH ERROR.
            82 3030 8F   B0  01E0   692 5$:      MOVW    #^A/00/,(R2)+           ;BACKGROUND WITH ASCII 0
                 82   30   90  01E5   693          MOVB    #^A/0/,(R2)+            ;
                      03   11  01E8   694          BRB     20$                     ;START LOOP AT BOTTOM
                 72   71   90  01EA   695 10$:     MOVB    -(R1),-(R2)             ;STORE BYTES LAST TO FIRST
                        01ED   696                                             ;LEAVING LEADING ASCII 0'S
                   FA 50   F4  01ED   697 20$:     SOBGEQ  R0,10$                  ;LOOP ZERO OR MORE TIMES
                      05   01F0   698          RSB
```

```
                                01F1    700          .SBTTL  FORMDIRSTRING - GET A DIRECTORY STRING
                                01F1    701 ;++
                                01F1    702 ; FUNCTIONAL DESCRIPTION:
                                01F1    703 ;
                                01F1    704 ;       PULL THE FIRST DIRECTORY NAME OFF THE FRONT OF THE INPUT
                                01F1    705 ;       DIRECTORY STRING AND FORM THE FULL FILE NAME OF THE DIRECTORY
                                01F1    706 ;       TO LOOK UP.
                                01F1    707 ;
                                01F1    708 ; CALLING SEQUENCE:
                                01F1    709 ;
                                01F1    710 ;       BSBW    FORMDIRSTRING
                                01F1    711 ;
                                01F1    712 ; INPUTS:
                                01F1    713 ;
                                01F1    714 ;       R6 = SIZE OF DIRECTORY STRING
                                01F1    715 ;       R7 = ADDRESS OF DIRECTORY STRING
                                01F1    716 ;               THE STRING CONTAINS NO BRACKETS,
                                01F1    717 ;               IT MAY BE OF THE FORM "DIR1.DIR2.DIR3...DIRN"
                                01F1    718 ;               THE FIRST AND ONLY ITEM MAY BE IN THE FORM GROUP,MEMBER
                                01F1    719 ;       DIRNAM(FP) = ADDRESS OF AREA TO BUILD THE NAME
                                01F1    720 ;
                                01F1    721 ; OUTPUTS:
                                01F1    722 ;
                                01F1    723 ;       R0 = SIZE OF DIRECTORY STRING
                                01F1    724 ;       R1 = ADDRESS OF DIRECTORY STRING
                                01F1    725 ;       R2,R3 PRESERVED
                                01F1    726 ;       R6,R7 UPDATED TO POINT AT THE REST OF THE STRING
                                01F1    727 ;--
                                01F1    728
                                01F1    729 FORMDIRSTRING:
     67    56    2E    3A      01F1    730          LOCC    #^A/./,R6,(R7)          ;FIND NEXT DIRECTORY STRING
     56    50    01    C3      01F5    731          SUBL3   #1,R0,R6                ;SIZE OF REST, SKIP THE "."
                                01F9    732                                          ;-1 IF EMPTY
     50    51    57    C3      01F9    733          SUBL3   R7,R1,R0                ;BYTE COUNT OF DIRECTORY NAME
           51    57    D0      01FD    734          MOVL    R7,R1                   ;ADDRESS OF DIRECTORY NAME
     57    01  A140    9E      0200    735          MOVAB   1(R1)[R0],R7            ;ADDRESS OF NEXT BYTE BEYOND "."
                 07    BB      0205    736          PUSHR   #^M<R0,R1,R2>           ;SAVE STRING DESCRIPTORS AND R2
           27    50    D1      0207    737          CMPL    R0,#39                  ;LENGTH OF DIRECTORY STRING OKAY?
                 03    15      020A    738          BLEQ    5$                      ;YES. BRANCH.
              024E     31      020C    739          BRW     BADFILNAM               ;NO. EXIT WITH ERROR.
     61    50    2C    3A      020F    740 5$:      LOCC    #^A/./,R0,(R1)          ;GOOD STRING; ANY "."?
                 39    13      0213    741          BEQL    20$                     ;BRANCH IF NOT, RETURN THE DESCRIPTOR AS IS
                 50    DD      0215    742          PUSHL   R0                      ;SAVE REMAINING BYTE COUNT
     50    04  AE    50    C3      0217    743          SUBL3   R0,4(SP),R0         ;BYTE COUNT TO LEFT OF "."
     52      CC  AD    DE      021C    744          MOVAL   DIRNAM(FP),R2           ;ADDRESS TO STORE FIRST 3 CHARS
                 B6    10      0220    745          BSBB    STORE3DIGITS            ;STORE THEM
     50    8E    01    C3      0222    746          SUBL3   #1,(SP)+,R0             ;COUNT OF CHARS TO RIGHT OF "."
     51    8E    8E    C1      0226    747          ADDL3   (SP)+,(SP)+,R1          ;ADR OF BYTE TO RIGHT OF LAST CHAR
     52      CF  AD    DE      022A    748          MOVAL   DIRNAM+3(FP),R2         ;ADR TO STORE LAST 3 CHARS OF DIR NAME
                 A8    10      022E    749          BSBB    STORE3DIGITS            ;STORE THEM
                 04    BA      0230    750          POPR    #^M<R2>                 ;RESTORE SAVED R2
           50    06    D0      0232    751          MOVL    #6,R0                   ;6 BYTES STRING SIZE
     51    CC  AD40    9E      0235    752 10$:     MOVAB   DIRNAM(FP)[R0],R1       ;POINT TO END OF STRING
81    5249442E  8F    D0      023A    753          MOVL    #^A/.DIR/,(R1)+         ;PUT TYPE IN STRING
     61   313B  8F    B0      0241    754          MOVW    #^A/;1/,(R1)            ;AND VERSION AS WELL
      51    CC  AD    9E      0246    755          MOVAB   DIRNAM(FP),R1           ;ADDRESS OF STRING
           50    06    C0      024A    756          ADDL    #6,R0                   ;SIZE INCLUDES ".DIR;1"
```

```
                          05  024D   757          RSB
                     38   BB  024E   758 20$:     PUSHR   #^M<R3,R4,R5>           ;SAVE THESE FROM MOVC3
                              0250   759 ;
                              0250   760 ; 12(SP) = SIZE OF STRING, 16(SP) = ADDRESS
                              0250   761 ;
   CC AD   10 BE   0C AE   28  0250   762          MOVC3   12(SP),@16(SP),DIRNAM(FP) ;MOVE NAME TO SCRATCH AREA
                     38   BA  0257   763          POPR    #^M<R3,R4,R5>           ;RESTORE REGISTERS
                     07   BA  0259   764          POPR    #^M<R0,R1,R2>
                     D8   11  025B   765          BRB     10$
```

```
                                025D   767          .SBTTL  MOUNT - MOUNT THE VOLUME, INIT FOR FILE LOOKUP
                                025D   768 ;++
                                025D   769 ; FUNCTIONAL DESCRIPTION:
                                025D   770 ;
                                025D   771 ;       MOUNT PERFORMS THE NECESSARY INITIALIZATION FOR FILE LOOKUP.
                                025D   772 ; IT READS THE HOME BLOCK, AND THEN RETURNS THE INDEX FILE HEADER TO THE
                                025D   773 ; SPECIFIED BUFFER.  THE INDEX FILE HEADER IS ALTERED BY RECORDING THE
                                025D   774 ; VIRTUAL BLOCK OFFSET REQUIRED TO TRANSLATE 'FILE NUMBER' TO INDEX FILE VBN
                                025D   775 ;
                                025D   776 ; CALLING SEQUENCE:
                                025D   777 ;
                                025D   778 ;       CALLG   ARGLIST,FIL$MOUNT
                                025D   779 ;
                                025D   780 ; INPUT PARAMETERS:
                                025D   781 ;
                                025D   782 ;       CHAN(AP)                CHANNEL ON WHICH DEVICE IS ASSIGNED
                                025D   783 ;       UNUSED                  2ND PARAMETER NOT USED
                                025D   784 ;       IXFHDR(AP)              ADDRESS TO RETURN INDEX FILE HEADER
                                025D   785 ;
                                025D   786 ; IMPLICIT INPUTS:
                                025D   787 ;
                                025D   788 ;       NONE
                                025D   789 ;
                                025D   790 ; OUTPUT PARAMETERS:
                                025D   791 ;
                                025D   792 ;       R0 = SYSTEM STATUS CODE
                                025D   793 ;
                                025D   794 ; IMPLICIT OUTPUTS:
                                025D   795 ;
                                025D   796 ;       NONE
                                025D   797 ;
                                025D   798 ; COMPLETION CODES:
                                025D   799 ;
                                025D   800 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                                025D   801 ;       SS$_FILESTRUCT          FILE STRUCTURE LEVEL NOT SUPPORTED
                                025D   802 ;       SS$_BADCHKSUM           CHECKSUM ERROR ON HOME BLOCK OR INDEX FILE HEADER
                                025D   803 ;       SS$_BADFILEHDR          INDEX FILE HEADER IS BAD
                                025D   804 ;
                                025D   805 ; SIDE EFFECTS:                                                          ;G:6
                                025D   806 ;
                                025D   807 ;       NONE                                                            ;G:6
                                025D   808 ;
                                025D   809 ;
                                025D   810 ; EQUATED SYMBOLS, OFFSETS FROM AP
                                025D   811 ;
                    00000004    025D   812          CHAN            =       4
                    0000000C    025D   813          IXFHDR          =       12
                                025D   814 ;
                                025D   815 ;--
                                025D   816
                                025D   817 FIL$MOUNT::
                        001C    025D   818          .WORD   ^M<R2,R3,R4>
              53    0C AC   D0  025F   819          MOVL    IXFHDR(AP),R3           ;ADDRESS OF BUFFER
          7E  01      09   9C  0263   820          ROTL    #9,#1,-(SP)             ;NUMBER OF BYTES TO READ
              7E      21   3C  0267   821          MOVZWL  #IO$_READLBLK,-(SP)     ;READ LOGICAL BLOCK FUNCTION
                      53   DD  026A   822          PUSHL   R3                      ;BUFFER ADDRESS
                      01   DD  026C   823          PUSHL   #1                      ;LOGICAL BLOCK NUMBER 1 IS HOME BLK
```

```
                04 AC    DD  026E    824          PUSHL   CHAN(AP)                        ;CHANNEL
                   05    DD  0271    825          PUSHL   #5                              ;NO. OF ARGUMENTS
00000000'EF        6E    FA  0273    826          CALLG   (SP),L^FIL$RDWRTLBN            ;READ THE HOME BLOCK                    ;G:6
              73 50       E9  027A    827          BLBC    R0,30$                          ;BRANCH IF ERROR
           51    53       D0  027D    828          MOVL    R3,R1                           ;ADDRESS OF HOME BLOCK
           50    1D       3C  0280    829          MOVZWL  #HM1$W_CHECKSUM1a-1,R0          ;NO. OF WORDS IN FIRST CHECKSUM
              05BF        30  0283    830          BSBW    FIL$CHECKSUM1                   ;CHECK THE FIRST CHECKSUM
           51    53       D0  0286    831          MOVL    R3,R1                           ;ADR OF HOME BLOCK AGAIN
              05B4        30  0289    832          BSBW    FIL$CHECKSUM                    ;CHECK THE MAIN CHECKSUM
                          028C    833          CASE    HM1$W_STRUCLEV+1(R3),<-
                          028C    834                  15$,-                           ;STRUCTURE LEVEL 1
                          028C    835                  10$-                            ;STRUCTURE LEVEL 2
                          028C    836                  >,TYPE=B,LIMIT=#1
01'  01    0D A3    8F   028C                CASEB   HM1$W_STRUCLEV+1(R3),#1,S^#<<30001$-30000$>/2>-1
                          0291         30000$:
                001D'     0291                 .SIGNED_WORD    15$-30000$
                000A'     0293                 .SIGNED_WORD    10$-30000$
                          0295         30001$:
   50    08C0 8F    3C   0295    837 5$:      MOVZWL  #SS$_FILESTRUCT,R0              ;UNSUPPORTED FILE STRUCTURE LEVEL
                    04    029A    838          RET
                          029B    839 ;
                          029B    840 ; STRUCTURE LEVEL 2
                          029B    841 ;
      51    18 A3    D0   029B    842 10$:     MOVL    HM2$L_IBMAPLBN(R3),R1           ;INDEX BIT MAP STARTING LBN
      50    20 A3    3C   029F    843          MOVZWL  HM2$W_IBMAPSIZE(R3),R0          ;INDEX BIT MAP SIZE IN BLOCKS
      54    1C A3    D0   02A3    844          MOVL    HM2$L_MAXFILES(R3),R4           ;MAXIMUM FILES ON VOLUME
                          02A7    845                                                  ;ONLY INTERESETED IN HIGH 16 BITS       ;G:6
54    0E A3    04    A5   02A7    846          MULW3   #4,HM2$W_CLUSTER(R3),R4         ;4*CLUSTER TO LOW WORD OF R4
                    0B    11   02AC    847          BRB     20$
                          02AE    848 ;
                          02AE    849 ; STRUCTURE LEVEL 1
                          02AE    850 ;
51    02 A3    10    9C   02AE    851 15$:     ROTL    #16,HM1$L_IBMAPLBN(R3),R1       ;LOGICAL BLK NO. OF 1ST INDEX BIT MAP BLK
      50    63    3C   02B3    852          MOVZWL  HM1$W_IBMAPSIZE(R3),R0          ;NO. OF BLOCKS OF INDEX BIT MAP
      54    02    D0   02B6    853          MOVL    #2,R4                           ;NUMBER OF VBN'S BEFORE INDEX FILE BIT MAP
      54    50    A0   02B9    854 20$:     ADDW    R0,R4                           ;LOW WORD IS VBNOFFSET
                          02BC    855                                                  ;FROM FILE ID TO INDEX FILE VBN
                          02BC    856 ;
                          02BC    857 ; READ INDEX FILE HEADER
                          02BC    858 ;        R0 - NUMBER OF BLOCKS IN INDEX FILE
                          02BC    859 ;        R1 - STARTING LBN OF INDEX FILE
                          02BC    860 ;
 08 AE    51    50    C1   02BC    861          ADDL3   R0,R1,8(SP)                     ;DESIRED LBN TO ARG LIST
00000000'EF    8E    FB   02C1    862          CALLS   (SP)+,L^FIL$RDWRTLBN            ;READ INDEX FILE HEADER                 ;G:6
                          02C8    863                                                  ;STRIP OFF THE ARGUMENT LIST
         25 50     E9   02C8    864          BLBC    R0,30$                          ;BRANCH IF ERROR
      51    53    D0   02CB    865          MOVL    R3,R1                           ;ADDRESS OF HEADER
            7E    D4   02CE    866          CLRL    -(SP)                           ;FORM FILE ID ON STACK
   00010001 8F    DD   02D0    867          PUSHL   #^X10001                        ;FOR THE INDEX FILE HEADER
      50    5E    D0   02D6    868          MOVL    SP,R0                           ;ADDRESS OF FILE ID
         0526     30   02D9    869          BSBW    FIL$CHKFILHDR                   ;CHECK THE FILE HEADER (SEE IF
                          02DC    870                                                  ;FILE IDS MATCH)
   01FE C3    54    B0   02DC    871          MOVW    R4,FH1$W_VBNOFFSET(R3)          ;STORE VBN OFFSET
                          02E1    872 ;
                          02E1    873 ; IF MAXFILES WAS GREATER THAN ^XFFFF THEN HIGH 16 BITS OF R4 WILL BE
                          02E1    874 ; NON-ZERO.  IN THIS CASE, RECORD THE BIGFILNUM BIT IN THE STRUCLEV WORD
                          02E1    875 ;
```

```
                54   54    F0 8F    78  02E1   876          ASHL    #-16,R4,R4              ;SEE IF HIGH 16 BITS = 0
                                05    13  02E6   877          BEQL    25$
                  00 06 A3      0A    E2  02E8   878          BBSS    #FH2$V_BIGFILNUM,FH2$W_STRUCLEV(R3),25$ ;MUST USE HIGH 8 BITS
                                                   02ED   879                                  ;OF RVN FIELD AS FILE NUMBER EXTENSION
                        50    01    3C  02ED   880 25$:    MOVZWL  #SS$_NORMAL,R0          ;SUCCESSFUL COMPLETION
                                      04  02F0   881 30$:    RET
```

```
                        02F1    883          .SBTTL  FINDFILID - FIND FILE ID FOR SPECIFIED FILE
                        02F1    884 ;++
                        02F1    885 ; FUNCTIONAL DESCRIPTION:
                        02F1    886 ;
                        02F1    887 ;       FINDFILID SCANS A SPECIFIED DIRECTORY FOR A FILE AND
                        02F1    888 ; RETURNS ITS FILE ID IF FOUND.  STRUCTURE LEVEL 1 AND 2 DIRECTORIES
                        02F1    889 ; ARE SUPPORTED, 0 VERSION NUMBER MEANS FIND MOST RECENT VERSION,
                        02F1    890 ; -1 VERSION (FIND OLDEST) IS NOT SUPPORTED.  VERSION FOUND IS RETURNED IN
                        02F1    891 ; THE NAME BLOCK ADDRESSED BY THE FILE DESCRIPTOR (LEVEL 1 ONLY).
                        02F1    892 ; NOTE THAT NON-CONTIGUOUS DIRECTORIES ARE NOT SUPPORTED.
                        02F1    893 ;
                        02F1    894 ; CALLING SEQUENCE:
                        02F1    895 ;
                        02F1    896 ;       CALLG   ARGLIST,FIL$FINDFILID
                        02F1    897 ;
                        02F1    898 ; INPUT PARAMETERS:
                        02F1    899 ;
                        02F1    900 ;       CHAN(AP)                ;CHANNEL ON WHICH DEVICE IS ASSIGNED
                        02F1    901 ;       FILDSC(AP)      =       ADDRESS OF 3 LONG WORD FILE DESCRIPTOR
                        02F1    902 ;                               1 - SIZE OF ASCII STRING,  A 0 VALUE MEANS
                        02F1    903 ;                                       USE THE CONTENTS OF THE NAMBLK BELOW
                        02F1    904 ;                               2 - ADDRESS OF ASCII STRING
                        02F1    905 ;                               3 - ADDRESS OF NAME BLOCK - USED ONLY FOR LEVEL 1
                        02F1    906 ;                                       MAY CONTAIN DEFAULTS, BUT MUST BE
                        02F1    907 ;                                       AT LEAST INITIALIZED TO ZERO
                        02F1    908 ;                                       IT WILL BE WRITTEN.
                        02F1    909 ;       IXFHDR(AP)              ADR OF INDEX FILE HDR AS RETURNED FROM FIL$MOUNT
                        02F1    910 ;       DIRBUF(AP)              ADR OF 512 BYTE BUFFER TO USE FOR DIRECTORY SCAN
                        02F1    911 ;       STATBLK(AP)             ADDRESS OF 2 LONG WORD AREA USED FOR A
                        02F1    912 ;                               SCRATCH STATISTICS BLOCK
                        02F1    913 ;       FILID(AP)               ADR OF 3 WORD AREA USED BOTH AS THE ID OF
                        02F1    914 ;                               THE DIRECTORY TO SCAN AND AS THE PLACE TO
                        02F1    915 ;                               RETURN THE ID OF THE FILE FOUND
                        02F1    916 ;
                        02F1    917 ; IMPLICIT INPUTS:
                        02F1    918 ;
                        02F1    919 ;       NONE
                        02F1    920 ;
                        02F1    921 ; OUTPUT PARAMETERS:
                        02F1    922 ;
                        02F1    923 ;       R0 = SYSTEM STATUS CODE
                        02F1    924 ;
                        02F1    925 ; IMPLICIT OUTPUTS:
                        02F1    926 ;
                        02F1    927 ;       NONE
                        02F1    928 ;
                        02F1    929 ; COMPLETION CODES:
                        02F1    930 ;
                        02F1    931 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                        02F1    932 ;       SS$_NOSUCHFILE          FILE NOT FOUND
                        02F1    933 ;       SS$_BADFILENAME         SYNTAX ERROR IN FILE NAME
                        02F1    934 ;       SS$_BADCHKSUM           CHECKSUM ERROR ON DIRECTORY FILE HEADER
                        02F1    935 ;       SS$_BADFILEHDR          DIRECTORY FILE HEADER WAS BAD
                        02F1    936 ;
                        02F1    937 ; SIDE EFFECTS:
                        02F1    938 ;
                        02F1    939 ;       NONE
```

```
                                02F1    940 ;
                                02F1    941 ;
                                02F1    942 ; EQUATED SYMBOLS, OFFSETS FROM AP
                                02F1    943 ;
                    00000004    02F1    944         CHAN            =       4
                    00000008    02F1    945         FILDSC          =       8
                    0000000C    02F1    946         IXFHDR          =       12
                    00000010    02F1    947         DIRBUF          =       16
                    00000014    02F1    948         STATBLK         =       20
                    00000018    02F1    949         FILID           =       24
                                02F1    950 ;
                                02F1    951 ; OFFSETS FROM FP
                                02F1    952 ;
                                02F1    953         $OFFSET 0,NEGATIVE,<-
                                02F1    954         DIR_BFCNT,-                     ;BUFFER COUNT REMAINING IN LBN CACHE
                                02F1    955         DIR_BUF,-                       ;NEXT BUFFER ADDRESS IN DIR LBN CACHE
                                02F1    956         ENTRY_ADR,-                     ;FOUND CACHE ENTRY ADR
                                02F1    957         <ENTRY,FIL$C_DIR_SIZE>,-        ;CACHE ENTRY FOR SEARCH/CREATE
                                02F1    958         <SCRATCH_SIZE,0>-               ;SIZE OF SCRATCH AREA
                                02F1    959         >
                                02F1                .SAVE   LOCAL_BLOCK
                                02F1                .PSECT $ABS$ ABS
                    00000000    FFFA                .=0
                    FFFFFFFC    0000                .BLKB   -4
                                FFFC        DIR_BFCNT:
                    FFFFFFF8    FFFC                .BLKB   -4
                                FFF8        DIR_BUF:
                    FFFFFFF4    FFF8                .BLKB   -4
                                FFF4        ENTRY_ADR:
                    FFFFFFD0    FFF4                .BLKB   -FIL$C_DIR_SIZE
                                FFD0        ENTRY:
                                FFD0        SCRATCH_SIZE:
                                000002F1            .RESTORE
                                    02F1    960 ;
                                    02F1    961 ;--
                                    02F1    962
                                    02F1    963 FIL$FINDFILID::
                                0FFC    02F1    964         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                          5E    30    C2    02F3    965         SUBL    #-SCRATCH_SIZE,SP       ;ALLOCATE SCRATCH SPACE
6E    30    00    6E    00    2C    02F6    966         MOVC5   #0,(SP),#0,#-SCRATCH_SIZE,(SP) ;ZERO THE SCRATCH STORAGE
                          5B    D4    02FC    967         CLRL    R11                     ;ASSUME NO CACHE
                    02    6C    D1    02FE    968         CMPL    (AP),#2                 ;IF ONLY 2 ARGUMENTS
                          08    12    0301    969         BNEQ    5$
                 5B   04    AC    D0    0303    970         MOVL    4(AP),R11               ;THE FIRST IS THE CACHE ADDRESS
                 5C   08    AC    D0    0307    971         MOVL    8(AP),AP                ;THE SECOND IS THE REAL ARGUMENT LIST
                          5B    D5    030B    972 5$:     TSTL    R11                     ;CACHE ENABLED?
                          66    13    030D    973         BEQL    65$                     ;BRANCH IF NOT
                                    030F    974 ;
                                    030F    975 ; WE DO HAVE A CACHE TO LOOK IN AND MAKE ENTRIES IN, SET UP THE
                                    030F    976 ; SCRATCH REGION FOR A LOOKUP
                                    030F    977 ;
                                    030F    978         ASSUME  ENTRY EQ SCRATCH_SIZE   ;SCRATCH CACHE ENTRY MUST BE ON TOP
                                    030F    979         ASSUME  FIL$A_DIR_FID EQ 0      ;DIR ID IS AT FRONT OF CACHE ENTRY
          6E    18 BC    06    28    030F    980         MOVC3   #6,@FILID(AP),(SP)      ;STORE DIRECTORY ID
                50   08 BC    7D    0314    981         MOVQ    @FILDSC(AP),R0          ;GET LOOKUP NAME DESCRIPTOR
                     06   50    D1    0318    982         CMPL    R0,#6                   ;MUST BE MORE THAN ".DIR;1"
                          25    15    031B    983         BLEQ    10$                     ;BRANCH IF NOT A DIRECTORY NAME
```

```
                0F   50   D1  031D   984          CMPL    R0,#15                          ;AT MOST 9 CHAR WITH ".DIR;1"
                     20   14  0320   985          BGTR    10$                             ;BRANCH IF NOT A DIRECTORY NAME
             52  51   50   C1  0322   986          ADDL3   R0,R1,R2                        ;POINT OFF END OF NAME STRING
             72  313B 8F   B1  0326   987          CMPW    #^A/;1/,-(R2)                   ;LAST 2 BYTES ";1" ?
                     15   12  032B   988          BNEQ    10$                             ;BRANCH IF NOT A DIRECTORY NAME
         72  5249442E 8F   D1  032D   989          CMPL    #^A/.DIR/,-(R2)                 ;PRECEDED BY ".DIR" ?
                     0C   12  0334   990          BNEQ    10$                             ;BRANCH IF NOT A DIRECTORY NAME
                50   06   C2  0336   991          SUBL    #6,R0                           ;JUST KEEP THE NAME PART
              D6 AD   50   90  0339   992          MOVB    R0,FIL$T_DIR_NAM+ENTRY(FP)      ;PUT SIZE AND NAME STRING
           D7 AD   61   50   28  033D   993          MOVC3   R0,(R1),FIL$T_DIR_NAM+1+ENTRY(FP) ;IN ENTRY TO LOOKUP
           58   5B   04 AB   C1  0342   994 10$:     ADDL3   FIL$L_DIROFF(R11),R11,R8        ;ADDRESS OF DIRECTORY CACHE
           59   5B   08 AB   C1  0347   995          ADDL3   FIL$L_DIRNXT(R11),R11,R9         ;ADDRESS OF LAST+1 BYTE
                     1E   11  034C   996          BRB     60$                             ;LOOP 0 OR MORE TIMES
                              034E   997
                              034E   998          ASSUME  FIL$T_DIR_NAM EQ FIL$A_DIR_FID+6
           68   D0 AD   10   29  034E   999 20$:     CMPC3   #6+10,FIL$A_DIR_FID+ENTRY(FP), - ;DOES FID AND NAME MATCH?
                              0353  1000                  FIL$A_DIR_FID(R8)
                     06   12  0353  1001          BNEQ    30$                             ;BRANCH IF DIDN'T MATCH ALL OF IT
                F4 AD   58   D0  0355  1002          MOVL    R8,ENTRY_ADR(FP)                ;RECORD THAT A MATCH WAS FOUND
                     1E   11  0359  1003          BRB     70$
                              035B  1004 ;
                              035B  1005 ; FAILED TO MATCH THE ENTIRE ENTRY, DID WE MATCH THE DIR ID FIELD?
                              035B  1006 ;
                   0A   50   D1  035B  1007 30$:     CMPL    R0,#10                          ;IF 10 OR LESS CHAR'S LEFT
                        09   14  035E  1008          BGTR    50$                             ;THEN MATCHED THE DIR ID
                F4 AD   58   D0  0360  1009          MOVL    R8,ENTRY_ADR(FP)                ;SAVE THIS PARTIAL MATCH
                     D6 AD   95  0364  1010          TSTB    FIL$T_DIR_NAM+ENTRY(FP)         ;IF NOT SEARCHING FOR A DIR NAME
                        10   13  0367  1011          BEQL    70$                             ;THEN THIS ENTRY WILL DO FINE
                   58   24   C0  0369  1012 50$:     ADDL    #FIL$C_DIR_SIZE,R8              ;ADDRESS OF NEXT CACHE ENTRY
                   59   58   D1  036C  1013 60$:     CMPL    R8,R9                           ;DONE SCANNING DIR CACHE?
                        DD   1F  036F  1014          BLSSU   20$                             ;BRANCH IF NOT, CHECK NEXT ENTRY
                              0371  1015 ;
                              0371  1016 ; ENTRY_ADR(FP) = ADDRESS OF CACHE HIT ENTRY
                              0371  1017 ;                = 0 IF NO MATCH FOUND
                              0371  1018 ; IF WE DROP THROUGH TO HERE AND WE GOT A CACHE HIT, THEN IT WAS
                              0371  1019 ; NOT EXACTLY WHAT WE WERE LOOKING FOR.  BUT IT DID MATCH THE DIRECTORY.
                              0371  1020 ;
                58   F4 AD   D0  0371  1021          MOVL    ENTRY_ADR(FP),R8                ;WAS THERE A CACHE HIT?
                        45   13  0375  1022 65$:     BEQL    READ_DIR_HEADER                 ;BRANCH IF NO
                        0F   11  0377  1023          BRB     80$                             ;YES, FOR DIRECTORY LBN AND SIZE
                              0379  1024 ;
                              0379  1025 ; FOUND WHAT WE WERE LOOKING FOR - MAY ONLY NEED DIRECTORY LBN AND SIZE
                              0379  1026 ;
                   1E A8    D5  0379  1027 70$:     TSTL    FIL$A_DIR_OFID(R8)              ;DID WE GET A FILE ID?
                        0A   13  037C  1028          BEQL    80$                             ;BRANCH IF NOT
          18 BC    1E A8    06  037E  1029          MOVC3   #6,FIL$A_DIR_OFID(R8),@FILID(AP) ;RETURN THE FILE ID
                50   01   3C  0384  1030          MOVZWL  S^#SS$_NORMAL,R0                ;SET SUCCESS STATUS
                        04   0387  1031 72$:     RET                                     ;AND RETURN
                              0388  1032 ;
                              0388  1033 ; CACHE HIT ONLY FOUND THE DIRECTORY LBN AND SIZE, SAVING THE
                              0388  1034 ; READ OF THE DIRECTORY FILE HEADER.
                              0388  1035 ;
             E0 AD   10 A8    7D  0388  1036 80$:     MOVQ    FIL$Q_DIR_HDR(R8),FIL$Q_DIR_HDR+ENTRY(FP) ;SAVE DIRHDR
                              038D  1037                                                  ;INFO FOR MAKING A NEW ENTRY
                              038D  1038                                                  ;WITH THE DIRECTORY FID IN IT
             E8 AD   18 A8    D0  038D  1039          MOVL    FIL$L_DIR_BFOFF(R8),FIL$L_DIR_BFOFF+ENTRY(FP)
             EC AD   1C A8    B0  0392  1040          MOVW    FIL$W_DIR_BFCNT(R8),FIL$W_DIR_BFCNT+ENTRY(FP)
```

```
                        0397  1041                                    ;SAVE DIR LBN CACHE INFO TOO
                        0397  1042 ;
                        0397  1043 ; AT THIS POINT ENTRY(FP) CONTAINS DIRECTORY LBN CACHE INFORMATION
                        0397  1044 ; IF ONE HAD ALREADY EXISTED OR IF WE JUST CREATED IT.
                        0397  1045 ; SET UP THE WORKING LOCATIONS FOR THE DIRECTORY LBN CACHE
                        0397  1046 ;
     56   E4 AD   01   C3  0397  1047 90$:    SUBL3   #1,FIL$L_DIR_LBN+ENTRY(FP),R6 ;R6=STARTING LBN - 1
          57   E0 AD   3C  039C  1048          MOVZWL  FIL$W_DIR_BKCNT+ENTRY(FP),R7 ;R7=SIZE OF DIRECTORY IN BLOCKS
        FC AD   EC AD   3C  03A0  1049          MOVZWL  FIL$W_DIR_BFCNT+ENTRY(FP),DIR_BFCNT(FP) ;BUFFER COUNT IN LBN CACHE
     F8 AD   5B   E8 AD   C1  03A5  1050          ADDL3   FIL$L_DIR_BFOFF+ENTRY(FP),R11,DIR_BUF(FP) ;STARTING ADR IN CACHE
             57   56   C0  03AB  1051          ADDL    R6,R7                   ;LAST LBN OF FILE INCLUSIVE
          03 E2 AD   E9  03AE  1052          BLBC    FIL$B_DIR_LVL+ENTRY(FP),100$ ;BRANCH IF STRUCTURE LEVEL 1
               00BA  31  03B2  1053          BRW     FIND_LEVEL2_1
               0192  31  03B5  1054 100$:    BRW     FIND_LEVEL1_1
                     03B8  1055 BADDIR2:
               009C  31  03B8  1056          BRW     BADDIR
                     03BB  1057 BADRET1:
                     04  03BB  1058          RET
                     03BC  1059 ;
                     03BC  1060 ; CACHE WAS NOT ENABLED OR THERE WAS NOT A HIT FOR THIS DIRECTORY
                     03BC  1061 ;
                     03BC  1062 READ_DIR_HEADER:
 00000SFA'EF   6C    FA  03BC  1063          CALLG   (AP),FIL$RDCHKFILHDR    ;READ AND CHECK DIRECTORY FILE HEADER
          F5 50   E9  03C3  1064          BLBC    R0,BADRET1              ;BRANCH IF ERROR
     55   10 AC   D0  03C6  1065          MOVL    DIRBUF(AP),R5           ;ADDRESS OF BUFFER TO READ DIRECTORY BLOCKS
56   14 BC   01   C3  03CA  1066          SUBL3   #1,@STATBLK(AP),R6      ;GET START LBN - 1
                     03CF  1067 ;
                     03CF  1068 ; IF THIS RESULT IS NEGATIVE, THEN THE DIRECTORY WAS NOT CONTIGUOUS.
                     03CF  1069 ; THIS CODE SUPPORTS ONLY CONTIGUOUS DIRECTORIES, ANOTHER BUFFER WOULD
                     03CF  1070 ; BE REQUIRED TO HOLD THE DIRECTORY HEADER IN ORDER TO SUPPORT NON-CONTIGUOS
                     03CF  1071 ; DIRECTORIES.  SUCH DIRECTORIES ARE ONLY CREATED BY FILES-11 WHEN
                     03CF  1072 ; A DIRECTORY MUST BE EXTENDED AND THERE IS NOT ENOUGH CONTIGUOUS SPACE
                     03CF  1073 ; ANYWHERE ON THE VOLUME TO MAKE A NEW DIRECTORY OF THE CORRECT SIZE.
                     03CF  1074 ;
               E7    19  03CF  1075          BLSS    BADDIR2                 ;BRANCH IF NOT CONTIGUOUS
                     03D1  1076 ;
                     03D1  1077 ; SEE IF THIS LOOKS LIKE A DIRECTORY FILE
                     03D1  1078 ;
     54   0E A5   DE  03D1  1079          MOVAL   FH1$W_RECATTR(R5),R4    ;ADDRESS OF LEVEL 1 RECORD ATTRIBUTES
5A   07 A5   01   83  03D5  1080          SUBB3   #1,FH1$W_STRUCLEV+1(R5),R10 ;0 IF LEVEL 1, 1 IF LEVEL 2
               04   13  03DA  1081          BEQL    10$                     ;BRANCH IF LEVEL 1
     54   14 A5   DE  03DC  1082          MOVAL   FH2$W_RECATTR(R5),R4    ;ADDRESS OF LEVEL 2 RECORD ATTRIBUTES
57   08 A4   10   9C  03E0  1083 10$:    ROTL    #16,FAT$L_EFBLK(R4),R7  ;VBN OF DIRECTORY EOF
          0C A4   B5  03E5  1084          TSTW    FAT$W_FFBYTE(R4)        ;IF ZERO, EFBLK IS LAST+1 VBN
               02   12  03E8  1085          BNEQ    20$
               57   D7  03EA  1086          DECL    R7                      ;CORRECT TO GET LAST VBN
E4 AD   56   01   C1  03EC  1087 20$:    ADDL3   #1,R6,FIL$L_DIR_LBN+ENTRY(FP) ;SAVE START LBN,
     E0 AD   57   B0  03F1  1088          MOVW    R7,FIL$W_DIR_BKCNT+ENTRY(FP) ;DIRECTORY SIZE,
     E2 AD   5A   90  03F5  1089          MOVB    R10,FIL$B_DIR_LVL+ENTRY(FP) ;AND STRUCTURE LEVEL
                     03F9  1090 ;
                     03F9  1091 ; SEE IF WE CAN SET UP A CACHE OF THE DIRECTORY BLOCKS FOR THIS DIRECTORY
                     03F9  1092 ;
               5B   D5  03F9  1093          TSTL    R11                     ;ANY CACHEING ENABLED?
               51   13  03FB  1094          BEQL    80$                     ;BRANCH IF NOT
52   14 AB   10 AB   C3  03FD  1095          SUBL3   FIL$L_LBNNXT(R11),FIL$L_LBNMAX(R11),R2 ;NO. OF BYTES
                     0403  1096                                          ;AVAILABLE TO ALLOCATE
52   52   F7 8F   78  0403  1097          ASHL    #-9,R2,R2               ;NO. OF PAGES AVAILABLE
```

```
                44   13   0408  1098          BEQL    80$                             ;BRANCH IF NO SPACE AT ALL
             57 52   D1   040A  1099          CMPL    R2,R7                           ;ENOUGH ROOM FOR WHOLE DIR
                03   15   040D  1100          BLEQ    40$                             ;BRANCH IF NOT, USE WHAT IS LEFT
             52 57   D0   040F  1101          MOVL    R7,R2                           ;YES, USE THE RIGHT SIZE
          53   10 AB   D0   0412  1102 40$:    MOVL    FIL$L_LBNNXT(R11),R3            ;OFFSET TO DIR LBN CACHE
                       0416  1103 ;
                       0416  1104 ; READ THE DISK BLOCKS INTO THE LBN CACHE.
                       0416  1105 ;
       7E   52   09    78   0416  1106          ASHL    #9,R2,-(SP)                     ;BYTE COUNT TO TRANSFER
            7E   21    3C   041A  1107          MOVZWL  #IO$_READLBLK,-(SP)             ;READ LOGICAL BLOCK FUNCTION
              6B43     9F   041D  1108          PUSHAB  (R11)[R3]                       ;BUFFER ADDRESS
             E4 AD     DD   0420  1109          PUSHL   FIL$L_DIR_LBN+ENTRY(FP)         ;STARTING LBN
             04 AC     DD   0423  1110          PUSHL   CHAN(AP)                        ;CHANNEL
00000000'EF    05     FB   0426  1111          CALLS   #5,FIL$RDWRTLBN                 ;FILL THE DIR LBN CACHE
          1E 50     E9   042D  1112          BLBC    R0,80$                          ;BRANCH IF ERROR
                       0430  1113 ;
                       0430  1114 ; NOTE THAT DIRECTORY BLOCKS ARE IN MEMORY
                       0430  1115 ;
          FC AD   52   D0   0430  1116          MOVL    R2,DIR_BFCNT(FP)                ;COUNT OF BLOCKS READ IN
       F8 AD    6B43   9E   0434  1117          MOVAB   (R11)[R3],DIR_BUF(FP)           ;ADDRESS OF FIRST BLOCK READ IN
             D6 AD     95   0439  1118          TSTB    FIL$T_DIR_NAM+ENTRY(FP)         ;ARE WE LOOKING UP ANOTHER DIRECTORY?
                10     12   043C  1119          BNEQ    80$                             ;BRANCH IF YES, DON'T ALLOCATE
                       043E  1120                                                       ;A PERMANENT CACHE ENTRY FOR
                       043E  1121                                                       ;AN INTERMEDIATE LEVEL DIRECTORY
          EC AD   52   B0   043E  1122          MOVW    R2,FIL$W_DIR_BFCNT+ENTRY(FP)    ;SET UP FOR PERMANENT ENTRY
          E8 AD   53   D0   0442  1123          MOVL    R3,FIL$L_DIR_BFOFF+ENTRY(FP)    ;SAVE THE SIZE AND OFFSET OF CACHE
       51 52   09   78   0446  1124          ASHL    #9,R2,R1                        ;NO. OF BYTES IN CACHE
          10 AB   51   C0   044A  1125          ADDL    R1,FIL$L_LBNNXT(R11)            ;ALLOCATE THE CACHE
                       044E  1126 ;
                       044E  1127 ; IF IT WAS POSSIBLE TO READ THE DIRECTORY INTO THE LBN CACHE AREA,
                       044E  1128 ; THE SCAN WILL FIND THESE BLOCKS AS THEY ARE NEEDED.
                       044E  1129 ;
                       044E  1130 80$:
                       044E  1131 ;
                       044E  1132 ; R6 = STARTING LBN - 1 FOR THE DIRECTORY
                       044E  1133 ; R7 = COUNT OF BLOCKS OF DIRECTORY TO BE SCANNED
                       044E  1134 ; R10[0:7] = 0 IF STRUCTURE LEVEL 1
                       044E  1135 ;          = 1 IF STRUCTURE LEVEL 2
                       044E  1136 ;
          57   56   C0   044E  1137          ADDL    R6,R7                           ;LAST LBN OF FILE (INCLUSIVE)
            0F 5A   E8   0451  1138          BLBS    R10,FIND_LEVEL2                 ;BRANCH IF STRUCTURE LEVEL 2
              00E8   31   0454  1139          BRW     FIND_LEVEL1
                       0457  1140
                       0457  1141 ;
                       0457  1142 ; ERROR RETURNS.
                       0457  1143 ;
50   0828 8F   3C   0457  1144 BADDIR: MOVZWL  #SS$_BADIRECTORY,R0
                04   045C  1145 BADRET: RET
                       045D  1146 BADFILNAM:
50   0818 8F   3C   045D  1147          MOVZWL  #SS$_BADFILENAME,R0                     ;RETURN ERROR CODE.
                04   0462  1148          RET
```

```
                                0463  1150          .SBTTL  FIL$FINDFILID - STRUCTURE LEVEL 2
                                0463  1151 ;
                                0463  1152 ; STRUCTURE LEVEL 2
                                0463  1153 ;
                                0463  1154 FIND_LEVEL2:
                EF 34 A5   0D   E1  0463  1155          BBC     #FH2$V_DIRECTORY,FH2$L_FILECHAR(R5),BADDIR ;DIRECTORY BIT MUST BE SE
                                0468  1156
                                0468  1157          ASSUME  FAT$B_RATTRIB EQ FAT$B_RTYPE+1
                     0802 8F    B1  0468  1158          CMPW    #<FAT$M_NOSPAN @ 8 + FAT$C_VARIABLE>,- ;VARIABLE LENGTH
                          64        046C  1159                  FAT$B_RTYPE(R4)         ;RECORDS NOT CROSSING BLOCK BOUNDARIES
                          E8   12   046D  1160          BNEQ    BADDIR                  ;BRANCH IF BAD RECORD ATTRIBUTES
                                046F  1161 ;
                                046F  1162 ; ***** NOTE THAT EACH BLOCK MUST END IN A RECORD SIZE OF -1
                                046F  1163 ; ***** A RECORD IS NOT ALLOWED TO EXACTLY FILL THE BLOCK
                                046F  1164 ; ***** PDP-11 FILE CONTROL SERVICES WILL READ THIS FILE CORRECTLY, BUT
                                046F  1165 ; ***** WILL NOT WRITE IT PROPERLY.  LIKEWISE FOR RMS-11 AND RMS-32
                                046F  1166 ;
                                046F  1167 FIND_LEVEL2_1:
                   58   08 BC   7D  046F  1168          MOVQ    @FILDSC(AP),R8          ;R8 = SIZE, R9 = ADDRESS OF FILE NAME STRING
                          5A   D4   0473  1169          CLRL    R10                     ;ASSUME DEFAULT VERSION
                        53   58   7D  0475  1170          MOVQ    R8,R3                   ;COPY FILE NAME DESCRIPTOR
                64      53   2E   3A  0478  1171          LOCC    #^A/./,R3,(R4)          ;FIND FILE TYPE DELIMITER IF PRESENT
                          07   13   047C  1172          BEQL    40$                     ;BRANCH IF NOT PRESENT
                        53   70   9E  047E  1173          MOVAB   -(R0),R3                ;SIZE OF REMAINING STRING
                     54   01 A1   9E  0481  1174          MOVAB   1(R1),R4                ;ADDRESS OF STRING BEYOND DELIMITER
                64      53   3B   3A  0485  1175 40$:     LOCC    #^A/;/,R3,(R4)          ;SEE IF VERSION DELIMITER PRESENT
                          06   12   0489  1176          BNEQ    60$                     ;BRANCH IF IT IS
                64      53   2E   3A  048B  1177          LOCC    #^A/./,R3,(R4)          ;TRY ALTERNATE VERSION DELIMITER
                          34   13   048F  1178          BEQL    120$                    ;BRANCH IF NO VERSION STRING PRESENT
                                0491  1179 ;
                                0491  1180 ; R0 = BYTE COUNT OF VERSION STRING PLUS DELIMITER
                                0491  1181 ; R1 = ADDRESS OF VERSION DELIMITER
                                0491  1182 ;
                        58   50   C2  0491  1183 60$:     SUBL    R0,R8                   ;REDUCE FILE NAME STRING SIZE
                                0494  1184                                              ;ELIMINATING VERSION STRING AND DELIMITER
                          7E   DF   0494  1185          PUSHAL  -(SP)                   ;RESERVE LONG WORD FOR VERSION NUMBER
                                0496  1186                                              ;AND PUSH ITS ADDRESS
                       01 A1   9F  0496  1187          PUSHAB  1(R1)                   ;ADDRESS OF VERSION STRING
                          70   9F   0499  1188          PUSHAB  -(R0)                   ;SIZE OF VERSION STRING
          00000000'EF     03   FB  049B  1189          CALLS   #3,LIB$CVT_DTB          ;CONVERT DECIMAL VERSION STRING TO BINARY
                     B8   50   E9  04A2  1190          BLBC    R0,BADFILNAM            ;BRANCH IF SYNTAX ERROR IN VERSION STRING
                        5A   8E   D0  04A5  1191          MOVL    (SP)+,R10               ;FETCH EXPLICIT VERSION NUMBER
                                04A8  1192 ;
                                04A8  1193 ; R6  = ADDRESS OF LAST LBN READ FROM DIRECTORY FILE (FIRST - 1)
                                04A8  1194 ; R7  = ADDRESS OF LAST LBN (INCLUSIVE) TO BE READ FROM DIRECTORY FILE
                                04A8  1195 ; R8  = SIZE OF NAME STRING TO SCAN FOR
                                04A8  1196 ; R9  = ADDRESS OF NAME STRING TO SCAN FOR
                                04A8  1197 ; R10 = FILE VERSION NUMBER IF EXPLICIT, OR 0 IF DEFAULT TO LATEST VERSION
                                04A8  1198 ;
                          1B   11   04A8  1199          BRB     120$                    ;BEGIN LOOP AT BOTTOM
                                04AA  1200 ;
                                04AA  1201 ; R5 = ADDRESS OF NEXT RECORD
                                04AA  1202 ;
                     54   05 A5   9A  04AA  1203 100$:    MOVZBL  DIR$B_NAMECOUNT(R5),R4  ;GET SIZE OF "NAME.TYP" STRING
06 A5      54   00      69   58   2D  04AE  1204          CMPC5   R8,(R9),#0,R4,DIR$T_NAME(R5) ;SEE IF STRINGS MATCH
                          1D   13   04B5  1205          BEQL    200$                    ;BRANCH IF THEY DO
                          10   19   04B7  1206          BLSS    140$                    ;BRANCH IF BEYOND WHERE NAME WOULD GO
```

```
                50   65   3C   04B9  1207          MOVZWL  DIR$W_SIZE(R5),R0       ;USING THE SIZE OF THIS RECORD
             55 02 A540   9E   04BC  1208          MOVAB   2(R5)[R0],R5            ;FORM ADDRESS OF NEXT RECORD
                   65   B5   04C1  1209 110$:      TSTW    DIR$W_SIZE(R5)          ;END OF BLOCK? (MARKED WITH -1)
                   E5   14   04C3  1210          BGTR    100$                    ;BRANCH IF NOT
             06 56   57   F3   04C5  1211 120$:      AOBLEQ  R7,R6,160$
             50 0910 8F   3C   04C9  1212 140$:      MOVZWL  #SS$_NOSUCHFILE,R0      ;CANNOT FIND FILE
                   04   04CE  1213 150$:      RET
                 00F8   30   04CF  1214 160$:      BSBW    READ_DIR_LBN            ;READ THE NEXT DIRECTORY LBN
                   ED   11   04D2  1215          BRB     110$
                        04D4  1216 ;
                        04D4  1217 ; FOUND A MATCH OF FILE NAME AND TYPE
                        04D4  1218 ;
                   54   D6   04D4  1219 200$:      INCL    R4                      ;ROUND UP NAME COUNT
                54   01   CA   04D6  1220          BICL    #1,R4                   ;TO EVEN NUMBER OF BYTES
             53 06 A544   9E   04D9  1221          MOVAB   DIR$T_NAME(R5)[R4],R3   ;ADDRESS OF FIRST VERSION ENTRY
                50   65   3C   04DE  1222          MOVZWL  DIR$W_SIZE(R5),R0       ;SIZE OF THIS RECORD
             55 02 A540   9E   04E1  1223          MOVAB   2(R5)[R0],R5            ;FORM ADDRESS OF BEGINNING OF NEX RECORD
                        04E6  1224                                                  ;WHICH IS ALSO THE END OF THE VERSIONS
                   5A   D5   04E6  1225          TSTL    R10                     ;LATEST VERSION DESIRED?
                   11   13   04E8  1226          BEQL    240$                    ;BRANCH IF YES, R3 IS ADDRESS OF
                        04EA  1227                                                  ;DESIRED VERSION AND FILE ID
                63   5A   B1   04EA  1228 230$:      CMPW    R10,DIR$W_VERSION(R3)   ;IS THIS THE RIGHT VERSION?
                   0C   13   04ED  1229          BEQL    240$                    ;BRANCH IF YES
                   D8   1A   04EF  1230          BGTRU   140$                    ;BRANCH IF PAST WHERE IT WOULD BE
                53   08   C0   04F1  1231          ADDL    #DIR$C_VERSION,R3       ;NEXT VERSION ENTRY
                55   53   D1   04F4  1232          CMPL    R3,R5                   ;END OF RECORD?
                   F1   1F   04F7  1233          BLSSU   230$                    ;BRANCH IF NOT, CHECK NEXT VERSION
                   C6   11   04F9  1234          BRB     110$                    ;VERSION NOT IN THIS VERSION CHAIN
                        04FB  1235                                                  ;SEE IF IT IS IN THE NEXT RECORD
                        04FB  1236 ;
                        04FB  1237 ; FOUND THE FILE ID, RETURN IT TO CALLER
                        04FB  1238 ;
             56   02 A3   7D   04FB  1239 240$:      MOVQ    DIR$W_FID(R3),R6        ;GET THE FILE ID
                57   57   3C   04FF  1240          MOVZWL  R7,R7
          18 BC 02 A3   06   28   0502  1241          MOVC3   #6,DIR$W_FID(R3),@FILID(AP) ;AND RETURN IT TO THE CALLER
                        0508  1242 ;
                        0508  1243 ; SEE IF WE SHOULD MAKE A CACHE ENTRY FOR THIS LOOKUP
                        0508  1244 ; R6,R7 = FID
                        0508  1245 ;
                        0508  1246 EXIT_FILID_FND:
                   5B   D5   0508  1247          TSTL    R11                     ;IS THE CACHE ENABLED?
                   2C   13   050A  1248          BEQL    100$                    ;BRANCH IF NOT, JUST RETURN THE FID
                 D6 AD   95   050C  1249          TSTB    FIL$T_DIR_NAM+ENTRY(FP) ;WAS LOOKUP FOR A DIRECTORY?
                   07   12   050F  1250          BNEQ    20$                     ;BRANCH IF YES, MAKE A CACHE ENTRY
                 F4 AD   D5   0511  1251          TSTL    ENTRY_ADR(FP)           ;CACHE HIT FOR THIS DIR HDR?
                   22   12   0514  1252          BNEQ    100$                    ;BRANCH IF YES
                   08   11   0516  1253          BRB     30$                     ;MAKE THE CACHE ENTRY FOR THIS DIR HDR
             EE AD   56   D0   0518  1254 20$:      MOVL    R6,FIL$A_DIR_OFID+ENTRY(FP) ;STORE THE FID FOUND
             F2 AD   57   B0   051C  1255          MOVW    R7,FIL$A_DIR_OFID+4+ENTRY(FP)
             58   08 AB   D0   0520  1256 30$:      MOVL    FIL$L_DIRNXT(R11),R8    ;GET OFFSET TO FREE SPACE
          50 58   24   C1   0524  1257          ADDL3   #FIL$C_DIR_SIZE,R8,R0   ;FORM OFFSET TO END OF NEW ENTRY
             0C AB   50   D1   0528  1258          CMPL    R0,FIL$L_DIRMAX(R11)    ;ENOUGH SPACE FOR NEW ENTRY?
                   0A   14   052C  1259          BGTR    90$                     ;BRANCH IF NOT
             08 AB   50   D0   052E  1260          MOVL    R0,FIL$L_DIRNXT(R11)    ;YES, ALLOCATE THE NEW ENTRY
          6B48 D0 AD   24   28   0532  1261          MOVC3   #FIL$C_DIR_SIZE,ENTRY(FP),(R11)[R8] ;AND WRITE IT
                        0538  1262 90$:
                50   01   3C   0538  1263 100$:      MOVZWL  #SS$_NORMAL,R0          ;INDICATE SUCCESSFUL COMPLETION
```

```
                     04  053B  1264          RET
                         053C  1265 ;
                         053C  1266 ; BAD DIRECTORY FILE
                         053C  1267 ;
                         053C  1268 BADDIR1:
                FF18     31  053C  1269          BRW     BADDIR
```

```
                                053F  1271          .SBTTL  FIL$FINDFILID - STRUCTURE LEVEL 1
                                053F  1272 ;
                                053F  1273 ; STRUCTURE LEVEL 1 FIND
                                053F  1274 ; R4 = ADDRESS OF RECORD ATTRIBUTES
                                053F  1275 ;
                                053F  1276 FIND_LEVEL1:
                   64   01   91 053F  1277          CMPB    #FAT$C_FIXED,FAT$B_RTYPE(R4) ;FIXED LENGTH RECORD?
                        F8   12 0542  1278          BNEQ    BADDIR1                      ;BRANCH IF NOT
                02 A4   10   B1 0544  1279          CMPW    #16,FAT$W_RSIZE(R4)          ;16 BYTES EACH
                        F2   12 0548  1280          BNEQ    BADDIR1                      ;BRANCH IF NOT
                                054A  1281 ;
                                054A  1282 ; GET FILE NAME BLOCK FOR STRUCTURE LEVEL 1 DIRECTORY SCAN
                                054A  1283 ;
                                054A  1284 FIND_LEVEL1_1:
                53   08 AC   D0 054A  1285          MOVL    FILDSC(AP),R3                ;ADDRESS OF 3 LONG WORD FILE DESCRIPTOR
                        63   D5 054E  1286          TSTL    (R3)                         ;SIZE OF ASCII STRING
                        0D   13 0550  1287          BEQL    25$                          ;BRANCH IF NAME BLOCK ALL SET UP
                     08 A3   DD 0552  1288          PUSHL   8(R3)                        ;ADDRESS OF NAME BLOCK
                        63   DF 0555  1289          PUSHAL  (R3)                         ;ADDRESS OF STRING DESCRIPTOR
           0000'CF      02   FB 0557  1290          CALLS   #2,W^FIL$CVTFILNAM           ;CONVERT ASCII STRING TO NAME BLOCK (RAD50)
                 5D 50  E9 055C  1291          BLBC    R0,90$                       ;BRANCH IF ERROR
                53   08 A3   D0 055F  1292 25$:     MOVL    8(R3),R3                     ;ADDRESS OF NAME BLOCK
                6E   08 A3   3C 0563  1293          MOVZWL  8(R3),(SP)                   ;SAVE FILE VERSION OVER FILE SIZE
                                0567  1294                                               ;THAT WAS LEFT FROM THE STATBLK CALL
                50   18 AC   D0 0567  1295          MOVL    FILID(AP),R0                 ;ADDRESS OF RETURN FILE ID
                        80   D4 056B  1296          CLRL    (R0)+                        ;INIT TO ZERO
                        60   B4 056D  1297          CLRW    (R0)                         ;ALL 3 WORDS.
                        3D   11 056F  1298          BRB     80$                          ;START DIRECTORY SCAN AT BOTTOM OF LOOP
                                0571  1299 ;
                                0571  1300 ; THE FOLLOWING REGISTER CONVENTION ARE USED IN THE DIRECTORY SCAN
                                0571  1301 ;       R2 = CURRENT DIRECTORY ENTRY BEING SCANNED
                                0571  1302 ;       R3 = NAMEBLOCK TO COMPARE AGAINST
                                0571  1303 ;       R4 = ADDRESS OF LAST BYTE OF DIRECTORY BLOCK
                                0571  1304 ;       R5 = ADDRESS OF DIRECTORY BUFFER
                                0571  1305 ;       R6 = LOGICAL BLOCK NUMBER OF CURRENT DIRECTORY BLOCK
                                0571  1306 ;       R7 = LAST LOGICAL BLOCK NUMBER (INCLUSIVE) TO SCAN
                                0571  1307 ;       0(SP) = VERSION OF FILE FROM NAME BLOCK
                                0571  1308 ;
                     0056   30 0571  1309 30$:     BSBW    READ_DIR_LBN                 ;READ THE NEXT DIRECTORY LBN
           54   01FF C5   DE 0574  1310          MOVAL   511(R5),R4                   ;ADDRESS OF LAST BYTE OF BUFFER
                52   55   D0 0579  1311          MOVL    R5,R2                        ;SET DIRECTORY RECORD POINTER
                        62   B5 057C  1312 40$:     TSTW    (R2)                         ;IS THIS DIRECTORY RECORD EMPTY
                        28   13 057E  1313          BEQL    70$                          ;BRANCH IF YES
                  51   06 A2   DE 0580  1314          MOVAL   6(R2),R1                     ;STEP OVER FILE ID TO NAME PORTION
                50   53   D0 0584  1315          MOVL    R3,R0                        ;MAKE A SCRATCH COPY OF NAME BLOCK ADR
                81   80   D1 0587  1316          CMPL    (R0)+,(R1)+                  ;CHECK 1ST 2 WORDS OF RAD50 FILE NAME
                        1C   12 058A  1317          BNEQ    70$                          ;BRANCH IF NO MATCH
                81   80   D1 058C  1318          CMPL    (R0)+,(R1)+                  ;LAST WORD OF RAD50 FILE NAME
                                058F  1319                                               ;AND THE 1 WORD RAD50 FILE TYPE
                        17   12 058F  1320          BNEQ    70$                          ;BRANCH IF NO MATCH
                                0591  1321 ;
                                0591  1322 ; NOW CHECK THE VERSION, IF SPECIFIED VERSION WAS 0 WE'RE LOOKING FOR
                                0591  1323 ; THE LARGEST VERSION WHICH OTHERWISE MATCHES, KEEP THE CURRENT LARGEST
                                0591  1324 ; IN THE INPUT NAME BLOCK, THUS RETURNING THE VERSION NUMBER FOUND.
                                0591  1325 ;
                        6E   B5 0591  1326          TSTW    (SP)                         ;SEARCHING FOR HIGHEST VERSION?
                        07   13 0593  1327          BEQL    50$                          ;BRANCH IF YES
```

```
                61   60   B1  0595          CMPW    (R0),(R1)                ;NO, JUST COMPARE FOR EXACT MATCH
                     0E   12  0598          BNEQ    70$                      ;BRANCH IF NO MATCH
                     21   11  059A          BRB     100$                     ;FILE FOUND, EXIT LOOP
                61   60   B1  059C  50$:    CMPW    (R0),(R1)                ;HIGHER THAN PREVIOUS HIGHEST VERSION?
                     07   1E  059F          BGEQU   70$                      ;BRANCH IF NOT
                60   61   B0  05A1          MOVW    (R1),(R0)                ;RECORD HIGHEST VERSION FOUND
             18 BC   62   D0  05A4          MOVL    (R2),@FILID(AP)          ;AND RECORD ITS FILE ID
      FFCE 52   10   54   F1  05A8  70$:    ACBL    R4,#16,R2,40$            ;NEXT DIRECTORY RECORD IF ANY
             BF 56   57   F3  05AE  80$:    AOBLEQ  R7,R6,30$                ;NEXT DIRECTORY BLOCK IF ANY
                             05B2  ;
                             05B2  ; DIRECTORY COMPLETELY SCANNED, IF LOOKING FOR HIGHEST VERSION, THEN
                             05B2  ; POSSIBLY A FILE WAS FOUND, OTHERWISE NO MATCH.
                             05B2  ;
                18 BC   D5  05B2          TSTL    @FILID(AP)               ;IF ANY FILE ID WAS STORED
                             05B5                                           ;THEN A FILE WAS FOUND
                     0A   12  05B5          BNEQ    120$                     ;BRANCH IF FILE WAS FOUND
          50   0910 8F   3C  05B7          MOVZWL  #SS$_NOSUCHFILE,R0       ;NO SUCH FILE
                         04  05BC  90$:    RET
                             05BD  ;
                             05BD  ; EXACT MATCH, FILE FOUND
                             05BD  ;
             18 BC   62   D0  05BD  100$:   MOVL    (R2),@FILID(AP)          ;RETURN THE FILE ID
             56   18 BC   D0  05C1  120$:   MOVL    @FILID(AP),R6            ;FETCH FILE ID
                     57   D4  05C5          CLRL    R7                       ;INTO R6,R7
                   FF3E   31  05C7          BRW     EXIT_FILID_FND
```

```
                     05CA  1354          .SBTTL  READ_DIR_LBN - READ NEXT DIRECTORY LBN
                     05CA  1355 ;++
                     05CA  1356 ; FUNCTIONAL DESCRIPTION:
                     05CA  1357 ;
                     05CA  1358 ;       READ THE NEXT DIRECTORY LBN FROM THE DISK OR POINT AT
                     05CA  1359 ;       THE CACHED COPY IF ONE IS PRESENT
                     05CA  1360 ;
                     05CA  1361 ; CALLING SEQUENCE:
                     05CA  1362 ;
                     05CA  1363 ;       BSBW    READ_DIR_LBN
                     05CA  1364 ;
                     05CA  1365 ; INPUT:
                     05CA  1366 ;
                     05CA  1367 ;       R6 = DESIRED LBN
                     05CA  1368 ;       DIRBUF(AP) = BUFFER ADDRESS TO READ IT INTO
                     05CA  1369 ;       CHAN(AP) = CHANNEL FOR FIL$RDWRTLBN
                     05CA  1370 ;       DIR_BFCNT(FP) = COUNT OF BUFFERS REMAINING IN DIR LBN CACHE
                     05CA  1371 ;       DIR_BUF(FP) = ADDRESS OF NEXT BUFFER IN DIR LBN CACHE
                     05CA  1372 ;
                     05CA  1373 ; OUTPUTS:
                     05CA  1374 ;
                     05CA  1375 ;       R5 = ADDRESS OF DESIRED DIRECTORY LBN
                     05CA  1376 ;       RSB IF SUCCESSFUL
                     05CA  1377 ;       RET WITH STATUS IN R0 IF ERROR
                     05CA  1378 ;       R0, R1 DESTROYED, OTHERS PRESERVED
                     05CA  1379 ;
                     05CA  1380 ;--
                     05CA  1381
                     05CA  1382 READ_DIR_LBN:
          FC AD    D5  05CA  1383          TSTL    DIR_BFCNT(FP)           ;ANYTHING LEFT IN DIR LBN CACHE?
             0E    13  05CD  1384          BEQL    20$                     ;BRANCH IF NOT
          FC AD    D7  05CF  1385          DECL    DIR_BFCNT(FP)           ;COUNT ANOTHER BUFFER USED
       55 F8 AD    D0  05D2  1386          MOVL    DIR_BUF(FP),R5          ;LOAD ADDRESS OF BUFFER
F8 AD    0200 C5    DE  05D6  1387          MOVAL   512(R5),DIR_BUF(FP)     ;AND POINT TO NEXT BUFFER IF ANY
                     05  05DC  1388 10$:     RSB
                         05DD  1389 ;
                         05DD  1390 ; DIR LBN CACHE RAN OUT OF BLOCKS OR NEVER HAD ANY AT ALL
                         05DD  1391 ;
       55 10 AC    D0  05DD  1392 20$:     MOVL    DIRBUF(AP),R5           ;ADDRESS OF BUFFER TO READ INTO
                         05E1  1393          READLBN CHAN(AP),R6,(R5)        ;READ THE DESIRED LBN
    7E 01    09    9C  05E1                      ROTL    #9,#1,-(SP)
       7E    21    3C  05E5                      MOVZWL  #IO$_READLBLK,-(SP)
             65    DF  05E8                      PUSHAL  (R5)
             56    DD  05EA                      PUSHL   R6
          04 AC    DD  05EC                      PUSHL   CHAN(AP)
00000000'EF    05    FB  05EF                      CALLS   #5,L^FIL$RDWRTLBN                               ;G:5
          E3 50    E8  05F6  1394          BLBS    R0,10$                  ;BRANCH IF READ SUCCESSFULLY
                   04  05F9  1395          RET                             ;RETURN ERROR STATUS
```

```
                        1397          .SBTTL  RDCHKFILHDR - READ AND CHECK FILE HEADER
                        1398 ;++
                        1399 ; FUNCTIONAL DESCRIPTION:
                        1400 ;
                        1401 ;       RDCHKFILHDR READS AND VALIDATES A FILE HEADER GIVEN ITS FILE ID
                        1402 ; AND THE INDEX FILE HEADER AS RETURNED BY FIL$MOUNT.
                        1403 ;
                        1404 ; CALLING SEQUENCE:
                        1405 ;
                        1406 ;       CALLG   ARGLIST,FIL$RDCHKFILHDR
                        1407 ;
                        1408 ; INPUT PARAMETERS:
                        1409 ;
                        1410 ;       CHAN(AP)                CHANNEL ON WHICH DEVICE IS ASSIGNED
                        1411 ;       UNUSED                  UNUSED PARAMETER
                        1412 ;       IXFHDR(AP)              ADR OF INDEX FILE HEADER AS RETURNED BY FIL$MOUNT
                        1413 ;       FILHDR(AP)              ADDRESS OF 512 BYTE BUFFER FOR FILE HEADER
                        1414 ;       STATBLK(AP)             ADR OF 2 LONG WORD AREA TO RETURN STATISTICS BLOCK
                        1415 ;       FILID(AP)               ADDRESS OF 3 WORD FILE ID OF DESIRED FILE HEADER
                        1416 ;       RTRVPTRLEN(AP)  =       ADDRESS TO RETURN THE NUMBER OF
                        1417 ;                               BYTES OF RETRIEVAL POINTERS STORED
                        1418 ;                               ***** OPTIONAL PARAMETER *****
                        1419 ;       RTRVPTRBUF(AP)  =       ADDRESS OF RETRIEVAL POINTER
                        1420 ;                               BUFFER DESCRIPTOR.  THIS PARAMETER
                        1421 ;                               IS PRESENT IF AND ONLY IF
                        1422 ;                               RTRVPTRLEN IS PRESENT.
                        1423 ;                               THE RETRIEVAL POINTERS ARE RETURNED IN
                        1424 ;                               THE FORM 32 BIT BLOCK COUNT, 32 BIT LBN
                        1425 ;                               A ZERO BUFFER DESCRIPTOR ADDRESS OR A
                        1426 ;                               ZERO BUFFER ADDRESS MEANS DON'T
                        1427 ;                               RETURN RETRIEVAL POINTER INFO
                        1428 ;
                        1429 ; IMPLICIT INPUTS:
                        1430 ;
                        1431 ;       NONE
                        1432 ;
                        1433 ; OUTPUT PARAMETERS:
                        1434 ;
                        1435 ;       R0 = SYSTEM STATUS CODE
                        1436 ;
                        1437 ; IMPLICIT OUTPUTS:
                        1438 ;
                        1439 ;       NONE
                        1440 ;
                        1441 ; COMPLETION CODES:
                        1442 ;
                        1443 ;       SS$_NORMAL              SUCCESSFUL COMPLETION
                        1444 ;
                        1445 ; SIDE EFFECTS:
                        1446 ;
                        1447 ;       NONE
                        1448 ;
                        1449 ;
                        1450 ; EQUATED SYMBOLS
                        1451 ;
                        1452 ;    OFFSETS FROM AP
                        1453 ;
```

```
                     00000000  05FA  1454          ARGCNT          =       0
                     00000004  05FA  1455          CHAN            =       4
                     0000000C  05FA  1456          IXFHDR          =       12
                     00000010  05FA  1457          FILHDR          =       16
                     00000014  05FA  1458          STATBLK         =       20
                     00000018  05FA  1459          FILID           =       24
                     0000001C  05FA  1460          RTRVPTRLEN      =       28      ; OPTIONAL PARAMETER
                     00000020  05FA  1461          RTRVPTRBUF      =       32      ; PRESENT IF AND ONLY IF RTRVPTRLEN IS
                               05FA  1462 ;
                               05FA  1463 ;    OFFSETS FROM FP
                               05FA  1464 ;
                               05FA  1465          $OFFSET 0,NEGATIVE,<-
                               05FA  1466          <HDRCNT>,-                      ;COUNT OF FILE HEADERS READ
                               05FA  1467          <TMPRTRVLEN>,-                  ;TEMP RTRV PTR BYTE COUNT
                               05FA  1468          <TMPRTRVDSC,8>-                 ;TEMP RTRV PTR BUFFER DESCRIPTOR
                               05FA  1469          >
                               05FA                .SAVE   LOCAL_BLOCK
                               05FA                .PSECT $ABS$ ABS
                     00000000  FFFC                .=0
                     FFFFFFFC  0000                .BLKB   -4
                               FFFC          HDRCNT:
                     FFFFFFF8  FFFC                .BLKB   -4
                               FFF8          TMPRTRVLEN:
                     FFFFFFF0  FFF8                .BLKB   -8
                               FFF0          TMPRTRVDSC:
                           000005FA                .RESTORE
                               05FA  1470 ;
                               05FA  1471 ;--
                               05FA  1472
                               05FA  1473 FIL$RDCHKFILHDR::
                                   007C  05FA  1474          .WORD   ^M<R2,R3,R4,R5,R6>
                        7E  01   CE    05FC  1475          MNEGL   #1,-(SP)                ;COUNT OF HEADER BLOCKS READ
                            7E   D4    05FF  1476          CLRL    -(SP)                   ;INIT RTRV PTR BYTE COUNT
                            7E   7C    0601  1477          CLRQ    -(SP)                   ;ASSUME NO RTRV PTR BUFFER
                        08  6C   D1    0603  1478          CMPL    ARGCNT(AP),#RTRVPTRBUF/4 ;RTRV PTR PARAMS PRESENT?
                            0D   19    0606  1479          BLSS    2$                      ;BRANCH IF NOT
                     50   20 AC   D0    0608  1480          MOVL    RTRVPTRBUF(AP),R0       ;RTRV BUFFER DESCRIPTOR ADDRESS
                            07   13    060C  1481          BEQL    2$                      ;BRANCH IF NONE SPECIFIED
                     F0 AD   60   7D    060E  1482          MOVQ    (R0),TMPRTRVDSC(FP)     ;MAKE COPY OF BUF DESCRIPTOR
                        1C BC   D4    0612  1483          CLRL    @RTRVPTRLEN(AP)         ;INIT RETURNED BYTE COUNT
                               0615  1484
                               0615  1485          ASSUME  FILHDR EQ IXFHDR+4
                     52   0C AC   7D    0615  1486 2$:      MOVQ    IXFHDR(AP),R2           ;R2 = INDEX FILE HEADER ADDRESS
                               0619  1487                                              ;R3 = FILE HEADER ADDRESS
                               0619  1488          ASSUME  FILID EQ STATBLK+4
                     54   14 AC   7D    0619  1489          MOVQ    STATBLK(AP),R4          ;R4 = RETURN STATBLK ADDRESS
                               061D  1490                                              ;R5 = ADDRESS OF FILE ID
                        7E   65   7D    061D  1491          MOVQ    (R5),-(SP)              ;COPY FILE ID TO WRITABLE SCRATCH
                        55   5E   D0    0620  1492          MOVL    SP,R5                   ;AND REMEMBER ITS ADDRESS
                        56   7E   7E    0623  1493          MOVAQ   -(SP),R6                ;RESERVE SCRATCH STATISTICS BLOCK
                               0626  1494                                              ;AND SAVE ITS ADDRESS IN R6
                            64   7C    0626  1495          CLRQ    (R4)                    ;INIT THE RETURN STAT BLOCK
                            64   D7    0628  1496          DECL    (R4)                    ;-1 LBN MEANS NOT YET STORED
                        7E   65   3C    062A  1497 5$:      MOVZWL  (R5),-(SP)              ;FILE NUMBER TO LONG WORD
                  05 06 A2   0A   E1    062D  1498          BBC     #FH2$V_BIGFILNUM,FH2$W_STRUCLEV(R2),10$ ;BRANCH IF NO BIG FIL NUM
                  02 AE   05 A5   90    0632  1499          MOVB    FID$B_NMX(R5),2(SP)     ;HIGH 8 BITS OF RVN COMPLETE FILE NUMBER
                        50   8E   D0    0637  1500 10$:     MOVL    (SP)+,R0                ;IF FILE ID IS ZERO,
```

```
                03   12  063A  1501          BNEQ    12$
              0081   31  063C  1502          BRW     40$                             ;THEN READ LAST HEADER BLOCK
     51     01FE C2   3C  063F  1503 12$:     MOVZWL  FH1$W_VBNOFFSET(R2),R1          ;RECOVER VBN OFFSET FROM INDEX FILE HEADER
          50   51   C0  0644  1504          ADDL    R1,R0                           ;ADD VBN OFFSET TO FORM INDEX FILE VBN
            FC AD   D6  0647  1505          INCL    HDRCNT(FP)                      ;COUNT EACH HEADER READ
                        064A  1506          READVBN CHAN(AP),R0,(R3),(R2)           ;READ THE FILE HEADER
                62  DF  064A                        PUSHAL  (R2)
                63  DF  064C                        PUSHAL  (R3)
                50  DD  064E                        PUSHL   R0
              04 AC DD  0650                        PUSHL   CHAN(AP)
     06D9'CF    04  FB  0653                        CALLS   #4,W^FIL$READVBN
              76 50  E9  0658  1507          BLBC    R0,50$                          ;BRANCH IF ERROR
     71 FC AD   1F  E0  065B  1508          BBS     #31,HDRCNT(FP),50$              ;BRANCH IF JUST RE-READING MAIN HEADER
          50   55  D0  0660  1509          MOVL    R5,R0                           ;GET FILE ID ADDRESS
          51   53  D0  0663  1510          MOVL    R3,R1                           ;ADDRESS OF FILE HEADER
               0199  30  0666  1511          BSBW    FIL$CHKFILHDR                   ;CHECK THE FILE HEADER
        52   0C AC  D0  0669  1512          MOVL    IXFHDR(AP),R2                   ;INDEX FILE HEADER ADDRESS
             F0 AD  DF  066D  1513          PUSHAL  TMPRTRVDSC(FP)                  ;RTRV PTR BUF DESCRIPTOR
             F8 AD  DF  0670  1514          PUSHAL  TMPRTRVLEN(FP)                  ;ADDRESS TO RETURN BYTE COUNT
                56  DD  0673  1515          PUSHL   R6                              ;ADDRESS OF SCRATCH STAT BLOCK
                53  DD  0675  1516          PUSHL   R3                              ;ADDRESS OF FILE HEADER
     0792'CF    04  FB  0677  1517          CALLS   #4,W^FIL$STATBLK                ;READ STATISTICS BLOCK
        51   F8 AD  D0  067C  1518          MOVL    TMPRTRVLEN(FP),R1               ;ANY RTRV PTR INFO TO RETURN?
                16  13  0680  1519          BEQL    16$                             ;ZERO IF NONE REQUESTED
        1C BC   51  C0  0682  1520          ADDL    R1,@RTRVPTRLEN(AP)              ;ACCUMULATE RTRVPTR BYTE COUNT
        F0 AD   51  D1  0686  1521          CMPL    R1,TMPRTRVDSC(FP)               ;MORE SPACE NEEDED THAN WOULD FIT?
                04  15  068A  1522          BLEQ    14$                             ;BRANCH IF NOT
        51   F0 AD  D0  068C  1523          MOVL    TMPRTRVDSC(FP),R1               ;SAY WE USED IT ALL UP
        F4 AD   51  C0  0690  1524 14$:     ADDL    R1,TMPRTRVDSC+4(FP)             ;GET NEW STARTING ADDRESS
        F0 AD   51  C2  0694  1525          SUBL    R1,TMPRTRVDSC(FP)               ;AND CALC NEW SIZE REMAINING
           51   64  D2  0698  1526 16$:     MCOML   (R4),R1                         ;SEE IF START LBN HAS BEEN SET
                03  12  069B  1527          BNEQ    20$                             ;BRANCH IF IT HAS
           64   66  D0  069D  1528          MOVL    (R6),(R4)                       ;SET IT ONCE ONLY
     04 A4   04 A6  C0  06A0  1529 20$:     ADDL    4(R6),4(R4)                     ;ADD IN THE SIZE FROM THIS HEADER
        65   0E A3  7D  06A5  1530          MOVQ    FH2$W_EXT_FID(R3),(R5)          ;GET EXTENSION FILE ID IF ANY
     0F 06 A3   09  E0  06A9  1531          BBS     #FH2$V_LEVEL2,FH2$W_STRUCLEV(R3),30$ ;GOT IT IF LEVEL2
        50   01 A3  9A  06AE  1532          MOVZBL  FH1$B_MPOFFSET(R3),R0           ;GET WORD OFFSET TO LEVEL1 MAP AREA
         50   6340  3E  06B2  1533          MOVAW   (R3)[R0],R0                     ;ADDRESS OF MAP AREA
        65   02 A0  D0  06B6  1534          MOVL    FM1$W_EX_FILNUM(R0),(R5)        ;GET EXTENSION FILE NUMBER IF ANY
             04 A5  D4  06BA  1535          CLRL    4(R5)                           ;ZERO RVN
              FF6A  31  06BD  1536 30$:     BRW     5$                              ;READ THIS HEADER IF ANY
                        06C0  1537 ;
                        06C0  1538 ; LAST FILE HEADER READ, SEE IF MUST RE-READ THE ORIGINAL HEADER
                        06C0  1539 ;
             FC AD  D5  06C0  1540 40$:     TSTL    HDRCNT(FP)                      ;WAS -1, BUMPED ONCE PER READVBN
                09  15  06C3  1541          BLEQ    45$                             ;BRANCH IF STILL HAVE MASTER FILE HEADER
        65   18 BC  7D  06C5  1542          MOVQ    @FILID(AP),(R5)                 ;ORIGINAL FILE ID AGAIN
     EF FC AD   1F  E3  06C9  1543          BBCS    #31,HDRCNT(FP),30$              ;SET SIGN BIT AND GO READ ORIGINAL HEADER
        50   01  3C  06CE  1544 45$:     MOVZWL  #SS$_NORMAL,R0                  ;RETURN SUCCESS STATUS
                04  06D1  1545 50$:     RET
```

```
                     06D2  1547          .SBTTL  READVBN, WRITEVBN - READ/WRITE VIRTUAL BLOCK
                     06D2  1548 ;++
                     06D2  1549 ; FUNCTIONAL DESCRIPTION:
                     06D2  1550 ;
                     06D2  1551 ;       THESE ROUTINES READ OR WRITE A VIRTUAL BLOCK FROM A FILE.
                     06D2  1552 ; VOLUME IS SPECIFIED BY THE CHANNEL TO WHICH IT IS ASSIGNED, AND THE
                     06D2  1553 ; FILE IS SPECIFIED BY THE ADDRESS OF ITS FILE HEADER WHICH WAS PREVIOUSLY
                     06D2  1554 ; READ BY A CALL TO FIL$RDFILHDR.
                     06D2  1555 ;
                     06D2  1556 ; CALLING SEQUENCE:
                     06D2  1557 ;
                     06D2  1558 ;       CALLG   ARGLIST,FIL$READVBN
                     06D2  1559 ;       CALLG   ARGLIST,FIL$WRITEVBN
                     06D2  1560 ;
                     06D2  1561 ; INPUT PARAMETERS:
                     06D2  1562 ;
                     06D2  1563 ;       CHAN(AP)        =               ;CHANNEL TO WHICH VOLUME IS ASSIGNED
                     06D2  1564 ;       VBN(AP)         =               ;DESIRED VIRTUAL BLOCK NUMBER
                     06D2  1565 ;       BUFADR(AP)      =               ;ADDRESS OF BUFFER TO READ INTO
                     06D2  1566 ;       FILHDR(AP)      =               ;ADDRESS OF FILE HEADER
                     06D2  1567 ;
                     06D2  1568 ; IMPLICIT INPUTS:
                     06D2  1569 ;
                     06D2  1570 ;       NONE
                     06D2  1571 ;
                     06D2  1572 ; OUTPUT PARAMETERS:
                     06D2  1573 ;
                     06D2  1574 ;       R0 = SYSTEM STATUS CODE
                     06D2  1575 ;
                     06D2  1576 ; IMPLICIT OUTPUTS:
                     06D2  1577 ;
                     06D2  1578 ;       NONE
                     06D2  1579 ;
                     06D2  1580 ; COMPLETION CODES:
                     06D2  1581 ;
                     06D2  1582 ;       SS$_NORMAL                  SUCCESSFUL RETURN
                     06D2  1583 ;       SS$_ENDOFFILE               SPECIFIED VBN BEYOND END OF FILE
                     06D2  1584 ;
                     06D2  1585 ; SIDE EFFECTS:
                     06D2  1586 ;
                     06D2  1587 ;       NONE
                     06D2  1588 ;
                     06D2  1589 ;
                     06D2  1590 ; EQUATED SYMBOLS:
                     06D2  1591 ;
                     06D2  1592 ;    OFFSET FROM AP
                     06D2  1593 ;
00000004             06D2  1594          CHAN            =       4       ;CHANNEL TO WHICH VOLUME IS ASSIGNED
00000008             06D2  1595          VBN             =       8       ;VIRTUAL BLOCK NUMBER
0000000C             06D2  1596          BUFADR          =       12      ;BUFFER ADDRESS TO READ INTO
00000010             06D2  1597          FILHDR          =       16      ;ADDRESS OF FILE HEADER
                     06D2  1598 ;
                     06D2  1599 ;    OFFSETS FROM FP
                     06D2  1600 ;
FFFFFFFC             06D2  1601          IOFUNCTION      =       -4      ;SAVED I/O FUNCTION CODE
                     06D2  1602 ;
                     06D2  1603 ;--
```

```
                                06D2  1604
                                06D2  1605 FIL$WRITEVBN::
                           003C 06D2  1606          .WORD   ^M<R2,R3,R4,R5>
                        7E 20  3C 06D4  1607          MOVZWL  #IO$_WRITELBLK,-(SP)
                           05  11 06D7  1608          BRB     RDWRTVBN
                                06D9  1609
                                06D9  1610 FIL$READVBN::
                           003C 06D9  1611          .WORD   ^M<R2,R3,R4,R5>
                        7E 21  3C 06DB  1612          MOVZWL  #IO$_READLBLK,-(SP)
                                06DE  1613
                                06DE  1614 RDWRTVBN:
                     55 10 AC  D0 06DE  1615          MOVL    FILHDR(AP),R5           ;BASE ADR OF FILE HEADER
                     52 06 A5  3C 06E2  1616          MOVZWL  FH1$W_STRUCLEV(R5),R2   ;STRUCTURE LEVEL
                        33  10 06E6  1617          BSBB    INIRTRVPTRSCAN          ;SET UP TO SCAN RETRIEVAL POINTERS
                                06E8  1618 ;
                                06E8  1619 ; R4 = POINTER TO FIRST RETRIEVAL POINTER,
                                06E8  1620 ; R5 = POINTER TO FIRST BYTE BEYOND LAST RETRIEVAL POINTER
                                06E8  1621 ; LOOP THROUGH RETRIEVAL POINTERS TO FIND THE ONE WHICH CONTAINS THE DESIRED VBN
                                06E8  1622 ;
                  53 08 AC  01  C3 06E8  1623          SUBL3   #1,VBN(AP),R3           ;VBN BASE 0 TO LOOK FOR
                        4A  10 06ED  1624 20$:    BSBB    GETRTRVPTR              ;FETCH NEXT RETRIEVAL POINTER
                     50 53  D1 06EF  1625          CMPL    R3,R0                   ;IS VBN IN THIS RETRIEVAL POINTER
                        0E  19 06F2  1626          BLSS    40$                     ;BRANCH IF YES
                     53 50  C2 06F4  1627          SUBL    R0,R3                   ;PASS OVER THAT MANY VBN'S
                     55 54  D1 06F7  1628          CMPL    R4,R5                   ;ANY MORE RETRIEVAL POINTERS?
                        F1  1F 06FA  1629          BLSSU   20$                     ;BRANCH IF YES
                  50 0870 8F  3C 06FC  1630          MOVZWL  #SS$_ENDOFFILE,R0       ;RETURN END OF FILE INDICATION
                        04  0701  1631          RET
                                0702  1632 ;
                                0702  1633 ; VBN IS IN THIS RETRIEVAL POINTER, R1 = STARTING LBN
                                0702  1634 ;
                  7E 01  09  9C 0702  1635 40$:    ROTL    #9,#1,-(SP)             ;NUMBER OF BYTES TO READ/WRITE
                     FC AD  DD 0706  1636          PUSHL   IOFUNCTION(FP)          ;FUNCTION CODE
                     0C AC  DD 0709  1637          PUSHL   BUFADR(AP)              ;BUFFER TO TRANSFER TO/FROM
                  7E 51  53  C1 070C  1638          ADDL3   R3,R1,-(SP)             ;LBN
                     04 AC  DD 0710  1639          PUSHL   CHAN(AP)                ;CHANNEL
            00000000'EF  05  FB 0713  1640          CALLS   #5,L^FIL$RDWRTLBN       ;TRANSFER THE BLOCK ; [03]          ;G:3
                        04  071A  1641          RET
```

```
                     071B  1643          .SBTTL  INIRTRVPTRSCAN - INITIALIZE RETRIEVAL POINTER SCAN
                     071B  1644 ;++
                     071B  1645 ; FUNCTIONAL DESCRIPTION:
                     071B  1646 ;
                     071B  1647 ;       LOCATE START AND END OF RETRIEVAL POINTERS IN A FILE HEADER.
                     071B  1648 ;
                     071B  1649 ; CALLING SEQUENCE:
                     071B  1650 ;
                     071B  1651 ;       BSBW    INIRTRVPTRSCAN
                     071B  1652 ;
                     071B  1653 ; INPUT:
                     071B  1654 ;
                     071B  1655 ;       R2 = STRUCTURE LEVEL
                     071B  1656 ;       R5 = FILE HEADER ADDRESS
                     071B  1657 ;
                     071B  1658 ; OUTPUT:
                     071B  1659 ;
                     071B  1660 ;       R4 = ADDRESS OF 1ST RETRIEVAL POINTER
                     071B  1661 ;       R5 = ADDRESS OF FIRST BYTE BEYOND LAST RETREIVAL POINTER
                     071B  1662 ;
                     071B  1663 ;--
                     071B  1664
                     071B  1665 INIRTRVPTRSCAN:
     50   01 A5   9A 071B  1666          MOVZBL  FH1$B_MPOFFSET(R5),R0   ;WORD OFFSET TO MAP AREA
      54   6540   3E 071F  1667          MOVAW   (R5)[R0],R4             ;BASE ADR OF MAP AREA
   0C 52    09   E0 0723  1668          BBS     #FH2$V_LEVEL2,R2,20$    ;BRANCH IF LEVEL2
   55   08 A4   9A 0727  1669          MOVZBL  FM1$B_INUSE(R4),R5      ;WORDS OF RETRIEVAL POINTERS IN USE
      54   0A   C0 072B  1670          ADDL    #FM1$C_POINTERS,R4      ;BEGINNING OF RETRIEVAL POINTERS
      55   6445   3E 072E  1671 10$:    MOVAW   (R4)[R5],R5             ;ADR JUST BEYOND LAST VALID RTRV PTR
                  05 0732  1672          RSB
   55   3A A5   9A 0733  1673 20$:    MOVZBL  FH2$B_MAP_INUSE(R5),R5  ;NO. OF WORDS OF RTRV PTRS IN USE
                  F5 11 0737  1674          BRB     10$
```

```
                        0739  1676          .SBTTL  GETRTRVPTR - CONVERT NEXT RETRIEVAL POINTER
                        0739  1677 ;++
                        0739  1678 ; FUNCTIONAL DESCRIPTION:
                        0739  1679 ;
                        0739  1680 ;       CONVERT NEXT RETRIEVAL POINTER TO NUMBER OF BLOCKS COVERED BY
                        0739  1681 ;       POINTER AND STARTING LBN.
                        0739  1682 ;
                        0739  1683 ; CALLING SEQUENCE:
                        0739  1684 ;
                        0739  1685 ;       BSBW    GETRTRVPTR
                        0739  1686 ;
                        0739  1687 ; INPUTS:
                        0739  1688 ;
                        0739  1689 ;       R2 = STRUCTURE LEVEL WORD
                        0739  1690 ;       R4 = ADDRESS OF NEXT RETRIEVAL POINTER
                        0739  1691 ;
                        0739  1692 ; OUTPUTS:
                        0739  1693 ;
                        0739  1694 ;       R0 = NUMBER OF BLOCKS COVERED BY THE RETRIEVAL POINTER
                        0739  1695 ;       R1 = STARTING LOGICAL BLOCK NUMBER
                        0739  1696 ;       R2,R3 PRESERVED
                        0739  1697 ;
                        0739  1698 ;--
                        0739  1699
                        0739  1700 GETRTRVPTR:
             0F 52  09  E0  0739  1701          BBS     #FH2$V_LEVEL2,R2,20$    ;BRANCH IF STRUCTURE LEVEL 2
                        073D  1702 ;
                        073D  1703 ; STRUCTURE LEVEL 1 RETRIEVAL POINTERS - 4 BYTES EACH
                        073D  1704 ;       BYTE 0 = BITS 16 - 23 OF LOGICAL BLOCK NUMBER
                        073D  1705 ;       BYTE 1 = COUNT - 1 OF BLOCKS COVERED BY THIS POINTER
                        073D  1706 ;       BYTES 2-3 = BITS 0:15 OF LOGICAL BLOCK NUMBER
                        073D  1707 ;
          50  01 A4  9A 073D  1708          MOVZBL  1(R4),R0                ;COUNT - 1
          51  02 A4  3C 0741  1709          MOVZWL  2(R4),R1                ;LOW LBN BITS
51  08  10  84  F0 0745  1710          INSV    (R4)+,#16,#8,R1         ;PUT HIGH LBN BITS IN
              36  11 074A  1711          BRB     INCRSB                  ;INCREMENT COUNT AND EXIT
                        074C  1712 ;
                        074C  1713 ; STRUCTURE LEVEL 2 RETRIEVAL POINTERS
                        074C  1714 ;       BITS 14:15 = RETRIEVAL POINTER FORMAT
                        074C  1715 ;
50  64  02  0E  EF 074C  1716 20$:     EXTZV   #FM2$V_FORMAT,#FM2$S_FORMAT,(R4),R0 ;FORMAT TO R0
                        0751  1717          CASE    R0,<-
                        0751  1718                  PLACEMENT,-             ;PLACEMENT FORMAT
                        0751  1719                  FORMAT1,-               ;FORMAT 1
                        0751  1720                  FORMAT2-                ;FORMAT 2
                        0751  1721                  >
     02' 00  50  AF 0751                CASEW   R0,#0,S^#<<30003$-30002$>/2>-1
                        0755        30002$:
                0014' 0755                  .SIGNED_WORD    PLACEMENT-30002$
                001D' 0757                  .SIGNED_WORD    FORMAT1-30002$
                0030' 0759                  .SIGNED_WORD    FORMAT2-30002$
                        075B        30003$:
                        075B  1722 ;
                        075B  1723 ; FORMAT 3 = 8 BYTES
                        075B  1724 ;
                        075B  1725 ;       BITS 0:13 = BITS 16:29 OF COUNT - 1
                        075B  1726 ;       BITS 14:15 = FORMAT = 3
```

```
                                075B  1727 ;       BYTES 2-3 = BITS 0:15 OF COUNT - 1
                                075B  1728 ;       BYTES 4-7 = LOGICAL BLOCK NUMBER
                                075B  1729 ;
                                075B  1730 FORMAT3:
          50   84   10   9C     075B  1731         ROTL    #16,(R4)+,R0            ;FORM COUNT - 1
     50   02   1E   00   F0     075F  1732         INSV    #0,#30,#2,R0            ;ZERO HIGH 2 BITS
               51   84   D0     0764  1733         MOVL    (R4)+,R1                ;GET LBN
                    19   11     0767  1734         BRB     INCRSB                  ;INCREMENT COUNT AND EXIT
                                0769  1735 ;
                                0769  1736 ; PLACEMENT CONTROL - THIS IS NOT A RETRIEVAL POINTER, RATHER IT
                                0769  1737 ;       CONSISTS OF 2 BYTES OF PLACEMENT INFORMATION.  TREAT AS IF 0
                                0769  1738 ;       LENGTH RETRIEVAL POINTER.
                                0769  1739 ;       R0 = 0
                                0769  1740 ;
                                0769  1741 PLACEMENT:
               51   01   CE     0769  1742         MNEGL   #1,R1                   ;IMPOSSIBLE LBN
               54   02   C0     076C  1743         ADDL    #2,R4                   ;BUMP THE POINTER
                    50   D4     076F  1744         CLRL    R0                      ;CLEAR BLOCK COUNT
                         05     0771  1745         RSB
                                0772  1746 ;
                                0772  1747 ; FORMAT 1 = 4 BYTES
                                0772  1748 ;       BITS 0:7 = COUNT - 1
                                0772  1749 ;       BITS 8:13 = BITS 16:21 OF LOGICAL BLOCK NUMBER
                                0772  1750 ;       BYTES 2-3 = BITS 0:15  OF LOGICAL BLOCK NUMBER
                                0772  1751 ;
                                0772  1752 FORMAT1:
          50   84   D0     0772  1753         MOVL    (R4)+,R0                ;FETCH ENTIRE RETRIEVAL POINTER
51   50   06   08   EF     0775  1754         EXTZV   #FM2$V_HIGHLBN,#FM2$S_HIGHLBN,R0,R1 ;FETCH HIGH LBN BITS
     50   50   10   79     077A  1755         ASHQ    #16,R0,R0               ;FORM R1 = LBN
          50   FC A4   9A     077E  1756         MOVZBL  -4(R4),R0               ;REFETCH COUNT - 1
                               0782  1757 INCRSB:
               50   D6     0782  1758         INCL    R0                      ;FORM COUNT
                    05     0784  1759         RSB                             ;AND RETURN
                               0785  1760 ;
                               0785  1761 ; FORMAT 2 = 6 BYTES
                               0785  1762 ;
                               0785  1763 ;       BITS 0:13 = COUNT - 1
                               0785  1764 ;       BITS 14:15 = FORMAT = 2
                               0785  1765 ;       BYTES 2-5 = LBN
                               0785  1766 ;
                               0785  1767 FORMAT2:
          50   84   3C     0785  1768         MOVZWL  (R4)+,R0                ;FETCH COUNT - 1 AND FORMAT BITS
50   50   0E   00   EF     0788  1769         EXTZV   #FM2$V_COUNT2,#FM2$S_COUNT2,R0,R0 ;COUNT - 1
               51   84   D0     078D  1770         MOVL    (R4)+,R1                ;LBN
                    F0   11     0790  1771         BRB     INCRSB                  ;INCREMENT COUNT AND RETURN
```

```
0792  1773          .SBTTL  STATBLK - GET FILE STATISTICS BLOCK
0792  1774 ;**
0792  1775 ; FUNCTIONAL DESCRIPTION:
0792  1776 ;
0792  1777 ;       GIVEN A FILE HEADER, RETURN THE FILE STATISTICS BLOCK
0792  1778 ;       AND OPTIONALLY RETURN THE RETRIEVAL POINTERS
0792  1779 ;
0792  1780 ; CALLING SEQUENCE:
0792  1781 ;
0792  1782 ;       CALLG   ARGLIST,FIL$STATBLK
0792  1783 ;
0792  1784 ; INPUT PARAMETERS:
0792  1785 ;
0792  1786 ;       FILHDR(AP)      =               ;ADDRESS OF THE FILE HEADER
0792  1787 ;       STATBLK(AP)     =               ;ADDRESS TO RETURN STATISTICS BLOCK
0792  1788 ;       RTRVPTRLEN(AP)  =               ;ADDRESS TO RETURN THE NUMBER OF
0792  1789 ;                                       ;BYTES OF RETRIEVAL POINTERS
0792  1790 ;                                       ;FOUND IN THE FILE HEADER(S).
0792  1791 ;                                       ;***** OPTIONAL PARAMETER *****
0792  1792 ;       RTRVPTRBUF(AP)  =               ;ADDRESS OF RETRIEVAL POINTER
0792  1793 ;                                       ;BUFFER DESCRIPTOR.  THIS PARAMETER
0792  1794 ;                                       ;IS PRESENT IF AND ONLY IF
0792  1795 ;                                       ;RTRVPTRLEN IS PRESENT.
0792  1796 ;                                       ;ZERO DESCRIPTOR ADDRESS OR ZERO
0792  1797 ;                                       ;BUFFER ADDRESS MEANS DON'T
0792  1798 ;                                       ;RETURN RETRIEVAL POINTER INFO
0792  1799 ;
0792  1800 ; IMPLICIT INPUTS:
0792  1801 ;
0792  1802 ;       NONE
0792  1803 ;
0792  1804 ; OUTPUT PARAMETERS:
0792  1805 ;
0792  1806 ;       R0 = SYSTEM STATUS CODE
0792  1807 ;       STATBLK CONTAINS 2 LONGWORDS
0792  1808 ;               LBN OF 1ST BLOCK IF CONTIGUOUS OR ZERO IF NOT
0792  1809 ;               SIZE OF FILE IN BLOCKS
0792  1810 ;       RTRVPTRLEN RECEIVES THE NUMBER OF BYTES OF RETRIEVAL POINTER
0792  1811 ;               INFORMATION THAT WOULD HAVE BEEN STORED IN THE RETRIEVAL
0792  1812 ;               POINTER BUFFER GIVEN A LARGE ENOUGH BUFFER.
0792  1813 ;       THE RETRIEVAL POINTER BUFFER RECEIVES NORMALIZED RETRIEVAL
0792  1814 ;               POINTERS IN THE FORMAT 32 BIT COUNT, 32 BIT STARTING LBN
0792  1815 ;
0792  1816 ; IMPLICIT OUTPUTS:
0792  1817 ;
0792  1818 ;       NONE
0792  1819 ;
0792  1820 ; COMPLETION CODES:
0792  1821 ;
0792  1822 ;       SS$_NORMAL                      SUCCESSFUL COMPLETION
0792  1823 ;
0792  1824 ; SIDE EFFECTS:
0792  1825 ;
0792  1826 ;       NONE
0792  1827 ;
0792  1828 ;
0792  1829 ; EQUATED SYMBOLS:
```

```
                        0792  1830 ;
                        0792  1831 ;    OFFSETS FROM AP
                        0792  1832 ;
                00000000  0792  1833          ARGCNT          =       0       ;NUMBER OF ARGUMENTS
                00000004  0792  1834          FILHDR          =       4       ;ADDRESS OF FILE HEADER
                00000008  0792  1835          STATBLK         =       8       ;ADDRESS TO RETURN STATISTICS BLOCK
                0000000C  0792  1836          RTRVPTRLEN      =       12      ;ADDRESS TO RETURN COUNT OF BYTES
                        0792  1837                                          ;STORED IN THE RETRIEVAL POINTER BUFFER
                00000010  0792  1838          RTRVPTRBUF      =       16      ;ADDRESS OF RETRIEVAL POINTER
                        0792  1839                                          ;BUFFER DESCRIPTOR
                        0792  1840 ;
                        0792  1841 ;--
                        0792  1842
                        0792  1843 FIL$STATBLK::
                    00FC  0792  1844          .WORD   ^M<R2,R3,R4,R5,R6,R7>
                  56  7C  0794  1845          CLRQ    R6                      ;ASSUME NOT DOING RETRIEVAL POINTERS
              04  6C  D1  0796  1846          CMPL    ARGCNT(AP),#RTRVPTRBUF/4 ;RTRV PTR PARAMS PRESENT?
                  0F  19  0799  1847          BLSS    5$                      ;BRANCH IF NOT
           50  10 AC  D0  079B  1848          MOVL    RTRVPTRBUF(AP),R0       ;ADDRESS OF BUFFER DESCRIPTOR
                  09  13  079F  1849          BEQL    5$                      ;BRANCH IF NOT SPECIFIED
              56  60  7D  07A1  1850          MOVQ    (R0),R6                 ;R6 = MAX SIZE, R7 = BUFFER ADR
              56  07  CA  07A4  1851          BICL    #7,R6                   ;EVEN MULTIPLE OF 8 BYTES
                0C BC  D4  07A7  1852          CLRL    @RTRVPTRLEN(AP)         ;INIT RETURN BYTE COUNT
           55  04 AC  D0  07AA  1853 5$:      MOVL    FILHDR(AP),R5           ;ADDRESS OF FILE HEADER
           52  06 A5  3C  07AE  1854          MOVZWL  FH1$W_STRUCLEV(R5),R2   ;STRUCTURE LEVEL
           50  34 A5  9A  07B2  1855          MOVZBL  FH2$L_FILECHAR(R5),R0   ;FILE CHARACTERISTICS IF LEVEL 2
           04 52  09  E0  07B6  1856          BBS     #FH2$V_LEVEL2,R2,10$    ;BRANCH IF STRUCTURE LEVEL 2
           50  0C A5  9A  07BA  1857          MOVZBL  FH1$W_FILECHAR(R5),R0   ;GET LEVEL 1 FILE CHARACTERISTICS
      7E   50  01  07  EF  07BE  1858 10$:     EXTZV   #FH1$V_CONTIG,#1,R0,-(SP) ;CONTIGUOUS BIT TO TOP OF STACK
                FF55  30  07C3  1859          BSBW    INIRTRVPTRSCAN          ;INIT FOR SCAN OF RETRIEVAL POINTERS
                  53  D4  07C6  1860          CLRL    R3                      ;INIT REGISTER TO COUNT BLOCKS
                  29  11  07C8  1861          BRB     50$                     ;START AT BOTTOM OF LOOP IN CASE
                      07CA  1862                                              ;FILE HAS NO RETRIEVAL POINTERS
                FF6C  30  07CA  1863 20$:     BSBW    GETRTRVPTR              ;GET THE NEXT RETRIEVAL POINTER
                  53  D5  07CD  1864          TSTL    R3                      ;IS THIS FIRST RTRV PTR?
                  0A  12  07CF  1865          BNEQ    40$                     ;BRANCH IF ALREADY COUNTED SOME
                      07D1  1866 ;
                      07D1  1867 ; FIRST RETRIEVAL POINTER
                      07D1  1868 ;
                  50  D5  07D1  1869          TSTL    R0                      ;IGNORE EMPTY ONES
                  1E  13  07D3  1870          BEQL    50$
              03 6E  E9  07D5  1871          BLBC    (SP),40$                ;BRANCH IF FILE NOT CONTIGUOUS
                      07D8  1872                                              ;0(SP) = 0 IN THIS CASE
           6E  51  D0  07D8  1873          MOVL    R1,(SP)                 ;SET LBN OF 1ST NON-ZERO RTRV PTR
           53  50  C0  07DB  1874 40$:     ADDL    R0,R3                   ;ACCUMULATE COUNT OF BLOCKS
               56  D5  07DE  1875          TSTL    R6                      ;ANY MORE ROOM FOR RTRV PTRS?
               09  13  07E0  1876          BEQL    45$                     ;BRANCH IF NO MORE BUFFER SPACE
           87  50  D0  07E2  1877          MOVL    R0,(R7)+                ;STORE SIZE OF RETRIEVAL POINTER
           87  51  D0  07E5  1878          MOVL    R1,(R7)+                ;AND STORE LBN
           56  08  C2  07E8  1879          SUBL    #8,R6                   ;USED 8 MORE BYTES OF SPACE
               57  D5  07EB  1880 45$:     TSTL    R7                      ;DOES CALLER WANT RTRV PTR INFO?
               04  13  07ED  1881          BEQL    50$                     ;BRANCH IF NOT, DON'T COUNT POINTERS
        0C BC  08  C0  07EF  1882          ADDL    #8,@RTRVPTRLEN(AP)      ;UPDATE RTRV PTR BYTE COUNT
           55  54  D1  07F3  1883 50$:     CMPL    R4,R5                   ;ANY MORE RETRIEVAL POINTERS?
               D2  1F  07F6  1884          BLSSU   20$                     ;BRANCH IF YES
               04  BA  07F8  1885          POPR    #^M<R2>                 ;GET SAVED STARTING LBN
        08 BC  52  7D  07FA  1886          MOVQ    R2,@STATBLK(AP)         ;RETURN THE STATISTICS BLOCK
```

```
                     50   01   3C  07FE  1887          MOVZWL  #SS$_NORMAL,R0          ;SUCCESSFUL COMPLETION
                                04  0801  1888          RET
```

```
                     0802  1890          .SBTTL  FIL$CHKFILHDR - CHECK FILE HEADER VALIDITY
                     0802  1891 ;++
                     0802  1892 ; FUNCTIONAL DESCRIPTION:
                     0802  1893 ;
                     0802  1894 ;       CHECK THE VALIDITY OF A FILE HEADER
                     0802  1895 ;
                     0802  1896 ; CALLING SEQUENCE:
                     0802  1897 ;
                     0802  1898 ;       BSBW    FIL$CHKFILHDR
                     0802  1899 ;
                     0802  1900 ; INPUT PARAMETERS:
                     0802  1901 ;
                     0802  1902 ;       R0 = ADDRESS OF FILE ID
                     0802  1903 ;       R1 = ADDRESS OF FILE HEADER
                     0802  1904 ;
                     0802  1905 ; IMPLICIT INPUTS:
                     0802  1906 ;
                     0802  1907 ;       NONE
                     0802  1908 ;
                     0802  1909 ; OUTPUT PARAMETERS:
                     0802  1910 ;
                     0802  1911 ;       RSB TO CALLER IF FILE HEADER VALID
                     0802  1912 ;       RET IF NOT VALID WITH R0 = ERROR STATUS
                     0802  1913 ;
                     0802  1914 ; IMPLICIT OUTPUTS:
                     0802  1915 ;
                     0802  1916 ;       NONE
                     0802  1917 ;
                     0802  1918 ; COMPLETION CODES:
                     0802  1919 ;
                     0802  1920 ;       SS$_BADFILEHDR          FILE ID CODES DON'T MATCH
                     0802  1921 ;       SS$_NOSUCHFILE          FILE IS MARKED AS DELETED
                     0802  1922 ;
                     0802  1923 ; SIDE EFFECTS:
                     0802  1924 ;
                     0802  1925 ;       NONE
                     0802  1926 ;
                     0802  1927 ;--
                     0802  1928
                     0802  1929 FIL$CHKFILHDR:
                     0802  1930          CASE    FH1$W_STRUCLEV+1(R1),<-
                     0802  1931                  5$,-                    ;STRUCTURE LEVEL 1
                     0802  1932                  10$-                    ;STRUCTURE LEVEL 2
                     0802  1933                  >,TYPE=B,LIMIT=#1
01'  01   07 A1   8F 0802                  CASEB   FH1$W_STRUCLEV+1(R1),#1,S^#<<30005$-30004$>/2>-1
                     0807          30004$:
           0006'     0807                  .SIGNED_WORD    5$-30004$
           000F'     0809                  .SIGNED_WORD    10$-30004$
                     080B          30005$:
              27   11 080B  1934          BRB     30$                     ;BAD FILE HEADER
                     080D  1935 ;
                     080D  1936 ; STRUCTURE LEVEL 1
                     080D  1937 ;
        7E   01   CE 080D  1938 5$:     MNEGL   #1,-(SP)                ;NO RVN TO CHECK
     7E   02 A1   D0 0810  1939         MOVL    FH1$W_FID_NUM(R1),-(SP) ;PUSH FILE ID ON STACK
              08   11 0814  1940         BRB     15$
                     0816  1941 ;
```

```
                          0816  1942 ; STRUCTURE LEVEL 2
                          0816  1943 ;
      7E    0C A1   3C    0816  1944 10$:    MOVZWL  FH2$W_FID_RVN(R1),-(SP) ;PUSH RELATIVE VOLUME NUMBER
      7E    08 A1   D0    081A  1945          MOVL    FH2$W_FID_NUM(R1),-(SP) ;PUSH FILE ID ON STACK
            6E      B5    081E  1946 15$:    TSTW    (SP)                    ;FILE DELETED?
            18      13    0820  1947          BEQL    40$                     ;BRANCH IF YES
         8E 80      D1    0822  1948          CMPL    (R0)+,(SP)+             ;FILE NUM AND FILE SEQ NUM AGREE?
            0D      12    0825  1949          BNEQ    30$                     ;BRANCH IF NOT, BAD HEADER
            6E      D5    0827  1950          TSTL    (SP)                    ;CHECKING RVN?
            05      19    0829  1951          BLSS    20$                     ;BRANCH IF NOT
         6E 60      B1    082B  1952          CMPW    (R0),(SP)               ;RELATIVE VOLUME NUMBER AND
                          082E  1953                                          ;FILE NUMBER EXTENSION AGREE
            04      12    082E  1954          BNEQ    30$                     ;BRANCH IF NOT
            01      BA    0830  1955 20$:    POPR    #^M<R0>                 ;CLEAN OFF STACK
            0C      11    0832  1956          BRB     FIL$CHECKSUM            ;GO VERIFY THE CHECKSUM
   50   0810 8F    3C    0834  1957 30$:    MOVZWL  #SS$_BADFILEHDR,R0      ;THIS HEADER IS BAD
                    04    0839  1958          RET
   50   0910 8F    3C    083A  1959 40$:    MOVZWL  #SS$_NOSUCHFILE,R0      ;DELETED FILE
                    04    083F  1960          RET
```

```
                     0840  1962          .SBTTL  CHECKSUM - VALIDATE A CHECKSUM
                     0840  1963 ;++
                     0840  1964 ; FUNCTIONAL DESCRIPTION:
                     0840  1965 ;
                     0840  1966 ;       THIS ROUTINE CALCULATES AND CHECKS THE FILE11 CHECKSUM FOR
                     0840  1967 ; FILE HEADERS AND THE HOMEBLOCK.
                     0840  1968 ;
                     0840  1969 ; CALLING SEQUENCE:
                     0840  1970 ;
                     0840  1971 ;       BSBW    FIL$CHECKSUM            ;CHECK FILE HEADER CHECKSUM
                     0840  1972 ;       BSBW    FIL$CHECKSUM1           ;CHECK SPECIFIED NO. OF WORDS IN R0
                     0840  1973 ;
                     0840  1974 ; INPUT PARAMETERS:
                     0840  1975 ;
                     0840  1976 ;       R0 = NO. OF WORDS TO CHECK IF ENTERING AT CHECKSUM1
                     0840  1977 ;       R1 = ADDRESS OF BUFFER TO CHECK
                     0840  1978 ;
                     0840  1979 ; IMPLICIT INPUTS:
                     0840  1980 ;
                     0840  1981 ;       NONE
                     0840  1982 ;
                     0840  1983 ; OUTPUT PARAMETERS:
                     0840  1984 ;
                     0840  1985 ;       RSB TO CALLER IF CHECKSUM IS OK
                     0840  1986 ;       RET TO TOP LEVEL WITH ERROR CODE IN R0 IF CHECKSUM IS WRONG
                     0840  1987 ;
                     0840  1988 ; IMPLICIT OUTPUTS:
                     0840  1989 ;
                     0840  1990 ;       NONE
                     0840  1991 ;
                     0840  1992 ; COMPLETION CODES:
                     0840  1993 ;
                     0840  1994 ;       NONE
                     0840  1995 ;
                     0840  1996 ; SIDE EFFECTS:
                     0840  1997 ;
                     0840  1998 ;       NONE
                     0840  1999 ;
                     0840  2000 ;--
                     0840  2001
                     0840  2002 FIL$CHECKSUM:
50   00FF 8F   3C    0840  2003          MOVZWL  #FH1$W_CHECKSUM@-1,R0   ;NO. OF WORDS TO CHECK
                     0845  2004 FIL$CHECKSUM1:
     52   D4         0845  2005          CLRL    R2                      ;INIT THE SUM
                     0847  2006 10$:
52   81   A0         0847  2007          ADDW    (R1)+,R2                ;ACCUMULATE THE SUM
  FA 50   F5         084A  2008          SOBGTR  R0,10$                  ;ONCE FOR EACH WORD
61   52   B1         084D  2009          CMPW    R2,(R1)                 ;CHECKSUM OK?
     01   12         0850  2010          BNEQ    20$                     ;BRANCH IF NOT
          05         0852  2011          RSB
                     0853  2012 20$:
50   0808 8F   3C    0853  2013          MOVZWL  #SS$_BADCHKSUM,R0       ;ERROR STATUS IN R0
          04         0858  2014          RET
                     0859  2015
                     0859  2016
                     0859  2017          .END
```

```
ARGCNT                          = 00000000     D
ASSIGN_DEV                        000001A5 R   D   02
BADDIR                            00000457 R   D   02
BADDIR1                           0000053C R   D   02
BADDIR2                           000003B8 R   D   02
BADFILNAM                         0000045D R   D   02
BADRET                            0000045C R   D   02
BADRET1                           000003BB R   D   02
BUFADR                          = 0000000C     D
CACHE_ADR                       = 00000010     D
CACHE_SIZE                      = 0000000C     D
CHAN                            = 00000004     D
CHANADR                         = 00000004     D
DIR$B_NAMECOUNT                 = 00000005     D
DIR$C_VERSION                   = 00000008     D
DIR$T_NAME                      = 00000006     D
DIR$W_FID                       = 00000002     D
DIR$W_SIZE                      = 00000000     D
DIR$W_VERSION                   = 00000000     D
DIR...                          = FFFFFFFF     D
DIRBUF                          = 00000010     D
DIRNAM                            FFFFFFCC     D
DIR_BFCNT                         FFFFFFFC     D
DIR_BUF                           FFFFFFF8     D
DIR_CACHE_CNT                   = 00000014     D
ENTRY                             FFFFFFD0     D
ENTRY_ADR                         FFFFFFF4     D
EXIT_FILID_FND                    00000508 R   D   02
FAT$B_RATTRIB                   = 00000001     D
FAT$B_RTYPE                     = 00000000     D
FAT$C_FIXED                     = 00000001     D
FAT$C_VARIABLE                  = 00000002     D
FAT$L_EFBLK                     = 00000008     D
FAT$M_NOSPAN                    = 00000008     D
FAT$W_FFBYTE                    = 0000000C     D
FAT$W_RSIZE                     = 00000002     D
FH1$B_MPOFFSET                  = 00000001     D
FH1$C_LEVEL1                    = 00000101     D
FH1$V_CONTIG                    = 00000007     D
FH1$V_LEVEL1                    = 00000008     D
FH1$W_CHECKSUM                  = 000001FE     D
FH1$W_FID_NUM                   = 00000002     D
FH1$W_FILECHAR                  = 0000000C     D
FH1$W_RECATTR                   = 0000000E     D
FH1$W_STRUCLEV                  = 00000006     D
FH1$W_VBNOFFSET                 = 000001FE     D
FH2$B_MAP_INUSE                 = 0000003A     D
FH2$B_MPOFFSET                  = 00000001     D
FH2$C_LEVEL2                    = 00000200     D
FH2$L_FILECHAR                  = 00000034     D
FH2$V_BIGFILNUM                 = 0000000A     D
FH2$V_CONTIG                    = 00000007     D
FH2$V_DIRECTORY                 = 0000000D     D
FH2$V_LEVEL2                    = 00000009     D
FH2$W_CHECKSUM                  = 000001FE     D
FH2$W_EXT_FID                   = 0000000E     D
FH2$W_FID_NUM                   = 00000008     D
FH2$W_FID_RVN                   = 0000000C     D
FH2$W_RECATTR                   = 00000014     D
FH2$W_STRUCLEV                  = 00000006     D
FID                               FFFFFFFA     D
FID$B_NMX                       = 00000005     D
FID$C_MFD                       = 00000004     D
FIL$A_DIR_FID                     00000000     D
FIL$A_DIR_OFID                    0000001E     D
FIL$A_IXFHDR                      00000018     D
FIL$B_DIR_LVL                     00000012     D
FIL$CACHE_INIT                    00000112 RG  D   02
FIL$CACHE_TRUNC                   00000186 RG  D   02
FIL$CHECKSUM                      00000840 R   D   02
FIL$CHECKSUM1                     00000845 R   D   02
FIL$CHKFILHDR                     00000802 R   D   02
FIL$CVTFILNAM                     ********       X   02
FIL$C_CACHE_ID                  = 00000001       D
FIL$C_DIR_SIZE                    00000024   G   D
FIL$C_SIZE                        00000218   G   D
FIL$FINDFILID                     000002F1 RG  D   02
FIL$GQ_CACHE                      ********W  GX   00
FIL$GT_DDDEV                      ********W  GX   00
FIL$GT_DDSTRING                   ********     X   02
FIL$GT_TOPSYS                     ********W  GX   00
FIL$L_DIRMAX                      0000000C       D
FIL$L_DIRNXT                      00000008       D
FIL$L_DIROFF                      00000004       D
FIL$L_DIR_BFOFF                   00000018       D
FIL$L_DIR_LBN                     00000014       D
FIL$L_LBNMAX                      00000014       D
FIL$L_LBNNXT                      00000010       D
FIL$L_LBNOFF                      0000000C       D
FIL$MOUNT                         0000025D RG  D   02
FIL$OPENFILE                      0000000C RG  D   02
FIL$Q_DIR_HDR                     00000010       D
FIL$RDCHKFILHDR                   000005FA RG  D   02
FIL$RDWRTLBN                      ********       X   02
FIL$READVBN                       000006D9 RG  D   02
FIL$STATBLK                       00000792 RG  D   02
FIL$T_DIR_NAM                     00000006       D
FIL$WRITEVBN                      000006D2 RG  D   02
FIL$W_CACHE_ID                    00000000       D
FIL$W_DIR_BFCNT                   0000001C       D
FIL$W_DIR_BKCNT                   00000010       D
FILDSC                          = 00000008       D
FILHDR                          = 00000004       D
FILID                           = 00000018       D
FILNAM                          = 00000008       D
FIL_GQ_CACHE                      00000000 R   D   02
FIL_GT_DDDEV                      00000004 R   D   02
FIL_GT_TOPSYS                     00000008 R   D   02
FIND_LEVEL1                       0000053F R   D   02
FIND_LEVEL1_1                     0000054A R   D   02
FIND_LEVEL2                       00000463 R   D   02
FIND_LEVEL2_1                     0000046F R   D   02
FM1$B_INUSE                     = 00000008       D
FM1$C_POINTERS                  = 0000000A       D
```

```
FM1$W_EX_FILNUM          = 00000002    D          TMPRTRVLEN                 FFFFFFF8    D
FM2$S_COUNT2             = 0000000E    D          VBN                      = 00000008    D
FM2$S_FORMAT             = 00000002    D
FM2$S_HIGHLBN            = 00000006    D
FM2$V_COUNT2             = 00000000    D
FM2$V_FORMAT             = 0000000E    D
FM2$V_HIGHLBN            = 00000008    D
FORMAT1                    00000772 R  D  02
FORMAT2                    00000785 R  D  02
FORMAT3                    0000075B R  D  02
FORMDIRSTRING              000001F1 R  D  02
GETRTRVPTR                 00000739 R  D  02
HDRCNT                     FFFFFFFC    D
HM1$L_IBMAPLBN           = 00000002    D
HM1$W_CHECKSUM1          = 0000003A    D
HM1$W_CHECKSUM2          = 000001FE    D
HM1$W_IBMAPSIZE          = 00000000    D
HM1$W_STRUCLEV           = 0000000C    D
HM2$L_IBMAPLBN           = 00000018    D
HM2$L_MAXFILES           = 0000001C    D
HM2$W_CHECKSUM1          = 0000003A    D
HM2$W_CHECKSUM2          = 000001FE    D
HM2$W_CLUSTER            = 0000000E    D
HM2$W_IBMAPSIZE          = 00000020    D
HM2$W_STRUCLEV           = 0000000C    D
INCRSB                     00000782 R  D  02
INIRTRVPTRSCAN             0000071B R  D  02
IO$_READLBLK             = 00000021    D
IO$_WRITELBLK            = 00000020    D
IOFUNCTION               = FFFFFFFC    D
IXFHDR                   = 0000000C    D
LBN_CACHE_CNT            = 00000018    D
LIB$CVT_DTB                ********   X   02
NAMBLK                     FFFFFFA4    D
NAMDSC                     FFFFFF98    D
OPENFILE_1                 0000000C R  D  02
OPENFILE_2                 0000000E R  D  02
PLACEMENT                  00000769 R  D  02
RDWRTVBN                   000006DE R  D  02
READ_DIR_HEADER            000003BC R  D  02
READ_DIR_LBN               000005CA R  D  02
RTRVPTRBUF               = 00000010    D
RTRVPTRLEN               = 0000000C    D
SAVABS...                = FFFFFFF0    D
SCRATCHSIZE                FFFFFF98    D
SCRATCH_SIZE               FFFFFFD0    D
SS$_BADCHKSUM            = 00000808    D
SS$_BADFILEHDR           = 00000810    D
SS$_BADFILENAME          = 00000818    D
SS$_BADIRECTORY          = 00000828    D
SS$_ENDOFFILE            = 00000870    D
SS$_FILESTRUCT           = 000008C0    D
SS$_NORMAL               = 00000001    D
SS$_NOSUCHFILE           = 00000910    D
STATBLK                  = 00000008    D
STORE3DIGITS               000001D8 R  D  02
TMPRTRVDSC                 FFFFFFF0    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	00000859 (2137.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ARGCNT	=00000000	1833 (17)	#-1478 (13) #-1846 (17) #-428 (2) #-475 (2)
ASSIGN_DEV	000001A5-R	645 (5)	#-560 (3)
BADDIR	00000457-R	1144 (9)	#-1056 (9) #-1155 (10) #-1160 (10) #-1269 (10)
BADDIR1	0000053C-R	1268 (10)	#-1278 (11) #-1280 (11)
BADDIR2	000003B8-R	1055 (9)	#-1075 (9)
BADFILNAM	0000045D-R	1146 (9)	#-1190 (10) #-399 (2) #-690 (6) #-739 (7)
BADRET	0000045C-R	1145 (9)	
BADRET1	000003BB-R	1057 (9)	#-1064 (9)
BUFADR	=0000000C	1596 (14)	#-1637 (14)
CACHE_ADR	=00000010	532 (3)	540 (3)
CACHE_SIZE	=0000000C	531 (3)	540 (3) #-541 (3)
CHAN	=00000004	1594 (14)	#-1110 (9) #-1393 (12) #-1506 (13) #-1639 (14)
			#-824 (8)
CHANADR	=00000004	529 (3)	#-435 (2) #-564 (3)
DIR$B_NAMECOUNT	=00000005		#-1203 (10)
DIR$C_VERSION	=00000008		#-1231 (10)
DIR$T_NAME	=00000006		1204 (10) 1221 (10)
DIR$W_FID	=00000002		#-1239 (10) 1241 (10)
DIR$W_SIZE	=00000000		#-1207 (10) #-1209 (10) #-1222 (10)
DIR$W_VERSION	=00000000		#-1228 (10)
DIR...	=FFFFFFFF	1469 (13)	1469 (13) 168 (1) 184 (1) 309 (2)
			959 (9)
DIRBUF	=00000010	947 (9)	#-1065 (9) #-1392 (12)
DIRNAM	FFFFFFCC	309 (2)	#-467 (2) 744 (7) 748 (7) 752 (7)
			755 (7) 762 (7)
DIR_BFCNT	FFFFFFFC	959 (9)	#-1049 (9) #-1116 (9) #-1383 (12) #-1385 (12)
DIR_BUF	FFFFFFF8	959 (9)	#-1050 (9) #-1117 (9) #-1386 (12) #-1387 (12)
DIR_CACHE_CNT	=00000014	533 (3)	#-548 (3)
ENTRY	FFFFFFD0	959 (9)	#-1010 (9) #-1036 (9) #-1039 (9) #-1040 (9)
			#-1047 (9) #-1048 (9) #-1049 (9) #-1050 (9)
			#-1052 (9) #-1087 (9) #-1088 (9) #-1089 (9)
			#-1109 (9) #-1118 (9) #-1122 (9) #-1123 (9)
			#-1249 (10) #-1254 (10) #-1255 (10) 1261 (10)
			978 (9) #-992 (9) 993 (9) 999 (9)
ENTRY_ADR	FFFFFFF4	959 (9)	#-1002 (9) #-1009 (9) #-1021 (9) #-1251 (10)
EXIT_FILID_FND	00000508-R	1246 (10)	#-1352 (11)
FAT$B_RATTRIB	=00000001		1157 (10)
FAT$B_RTYPE	=00000000		1157 (10) #-1159 (10) #-1277 (11)
FAT$C_FIXED	=00000001		#-1277 (11)
FAT$C_VARIABLE	=00000002		#-1158 (10)
FAT$L_EFBLK	=00000008		#-1083 (9)
FAT$M_NOSPAN	=00000008		#-1158 (10)
FAT$W_FFBYTE	=0000000C		#-1084 (9)
FAT$W_RSIZE	=00000002		#-1279 (11)
FH1$B_MPOFFSET	=00000001		131 (1) #-1532 (13) #-1666 (15)
FH1$C_LEVEL1	=00000101		143 (1)
FH1$V_CONTIG	=00000007		138 (1) #-1858 (17)
FH1$V_LEVEL1	=00000008	144 (1)	
FH1$W_CHECKSUM	=000001FE		133 (1) 137 (1) 140 (1) #-2003 (19)
FH1$W_FID_NUM	=00000002		#-1939 (18)

```
FH1$W_FILECHAR                  =0000000C                       #-1857  (17)
FH1$W_RECATTR                   =0000000E                         1079  (9)
FH1$W_STRUCLEV                  =00000006                       #-1080  (9)     132     (1)     #-1616  (14)    #-1854  (17)
                                                                #-1933  (18)
FH1$W_VBNOFFSET                 =000001FE       140     (1)     #-1503  (13)    #-871   (8)
FH2$B_MAP_INUSE                 =0000003A                       #-1673  (15)
FH2$B_MPOFFSET                  =00000001                         131   (1)
FH2$C_LEVEL2                    =00000200                         146   (1)
FH2$L_FILECHAR                  =00000034                         1155  (10)    #-1855  (17)
FH2$V_BIGFILNUM                 =0000000A       148     (1)     #-1498  (13)    #-878   (8)
FH2$V_CONTIG                    =00000007                         138   (1)
FH2$V_DIRECTORY                 =0000000D                       #-1155  (10)
FH2$V_LEVEL2                    =00000009       147     (1)     #-1531  (13)    #-1668  (15)    #-1701  (16)    #-1856  (17)
FH2$W_CHECKSUM                  =000001FE                         133   (1)
FH2$W_EXT_FID                   =0000000E                       #-1530  (13)
FH2$W_FID_NUM                   =00000008                       #-1945  (18)
FH2$W_FID_RVN                   =0000000C                       #-1944  (18)
FH2$W_RECATTR                   =00000014                         1082  (9)
FH2$W_STRUCLEV                  =00000006                         132   (1)     1498    (13)    1531    (13)    878     (8)
FID                              FFFFFFFA       309     (2)       431   (2)     #-448   (2)     #-449   (2)
FID$B_NMX                       =00000005                       #-1499  (13)
FID$C_MFD                       =00000004                       #-448   (2)     #-449   (2)
FIL$A_DIR_FID                    00000000       184     (1)       1000  (9)     979     (9)     998     (9)     999     (9)
FIL$A_DIR_OFID                   0000001E       184     (1)     #-1027  (9)     1029    (9)     #-1254  (10)    #-1255  (10)
FIL$A_IXFHDR                     00000018       168     (1)       439   (2)     562     (3)
FIL$B_DIR_LVL                    00000012       184     (1)     #-1052  (9)     #-1089  (9)
FIL$CACHE_INIT                   00000112-R     537     (3)
FIL$CACHE_TRUNC                  00000186-R     612     (4)
FIL$CHECKSUM                     00000840-R     2002    (19)    #-1956  (18)    #-832   (8)
FIL$CHECKSUM1                    00000845-R     2004    (19)    #-830   (8)
FIL$CHKFILHDR                    00000802-R     1929    (18)    #-1511  (13)    #-869   (8)
FIL$CVTFILNAM                    00000000-XR                     1290  (11)
FIL$C_CACHE_ID                  =00000001       153     (1)     #-371   (2)     #-544   (3)
FIL$C_DIR_SIZE                   00000024       184     (1)     #-1012  (9)     #-1257  (10)    #-1261  (10)    191     (1)
                                                                #-548   (3)     959     (9)
FIL$C_SIZE                       00000218       168     (1)       191   (1)     #-542   (3)     #-546   (3)     #-547   (3)
                                                                #-551   (3)
FIL$FINDFILID                    000002F1-R     963     (9)       464   (2)
FIL$GQ_CACHE                     00000000-XR                       195   (1)     205     (1)     #-369   (2)     #-567   (3)
                                                                #-614   (4)     #-617   (4)
FIL$GT_DDDEV                     00000000-XR                       196   (1)     207     (1)     649     (5)
FIL$GT_DDSTRING                  00000000-XR                       374   (2)
FIL$GT_TOPSYS                    00000000-XR                       197   (1)     209     (1)     452     (2)
FIL$L_DIRMAX                     0000000C       168     (1)     #-1258  (10)    #-551   (3)     553     (3)     #-615   (4)
FIL$L_DIRNXT                     00000008       168     (1)     #-1256  (10)    #-1260  (10)    #-547   (3)     #-615   (4)
                                                                #-995   (9)
FIL$L_DIROFF                     00000004       168     (1)     #-546   (3)     #-994   (9)
FIL$L_DIR_BFOFF                  00000018       184     (1)     #-1039  (9)     #-1050  (9)     #-1123  (9)
FIL$L_DIR_LBN                    00000014       184     (1)     #-1047  (9)     #-1087  (9)     #-1109  (9)
FIL$L_LBNMAX                     00000014       168     (1)     #-1095  (9)     #-559   (3)     #-616   (4)
FIL$L_LBNNXT                     00000010       168     (1)     #-1095  (9)     #-1102  (9)     #-1125  (9)     #-554   (3)
                                                                #-559   (3)     #-616   (4)     #-617   (4)
FIL$L_LBNOFF                     0000000C       168     (1)       553   (3)     #-554   (3)
FIL$MOUNT                        0000025D-R     817     (8)       441   (2)     565     (3)
FIL$OPENFILE                     0000000C-R     343     (2)
FIL$Q_DIR_HDR                    00000010       184     (1)     #-1036  (9)
FIL$RDCHKFILHDR                  000005FA-R     1473    (13)      1063  (9)     478     (2)
```

```
FIL$RDWRTLBN                 00000000-XR                  1111   (9)      1393   (12)     1640   (14)     826    (8)
                                                          862   (8)
FIL$READVBN                  000006D9-R   1610   (14)     1506   (13)
FIL$STATBLK                  00000792-R   1843   (17)     1517   (13)
FIL$T_DIR_NAM                00000006     184    (1)    #-1010   (9)    #-1118   (9)    #-1249   (10)   #-992    (9)
                                                          993   (9)      998    (9)
FIL$WRITEVBN                 000006D2-R   1605   (14)
FIL$W_CACHE_ID               00000000     168    (1)    #-371    (2)    #-544    (3)
FIL$W_DIR_BFCNT              0000001C     184    (1)    #-1040   (9)    #-1049   (9)    #-1122   (9)
FIL$W_DIR_BKCNT              00000010     184    (1)    #-1048   (9)    #-1088   (9)
FILDSC                      =00000008     945    (9)    #-1168   (10)   #-1285   (11)   #-981    (9)
FILHDR                      =00000004     1834   (17)     1485   (13)   #-1615   (14)   #-1853   (17)   #-432    (2)
FILID                       =00000018     1459   (13)     1029   (9)      1241   (10)   #-1295   (11)   #-1334   (11)
                                                        #-1341   (11)   #-1349   (11)   #-1350   (11)     1488   (13)
                                                        #-1542   (13)     980    (9)
FILNAM                      =00000008     530    (3)    #-381    (2)    #-653    (5)
FIL_GQ_CACHE                 00000000-R   204    (1)    #-357    (2)
FIL_GT_DDDEV                 00000004-R   206    (1)    #-647    (5)
FIL_GT_TOPSYS                00000008-R   208    (1)    #-450    (2)
FIND_LEVEL1                  0000053F-R   1276   (11)   #-1139   (9)
FIND_LEVEL1_1                0000054A-R   1284   (11)   #-1054   (9)
FIND_LEVEL2                  00000463-R   1154   (10)   #-1138   (9)
FIND_LEVEL2_1                0000046F-R   1167   (10)   #-1053   (9)
FM1$B_INUSE                 =00000008                   #-1669   (15)
FM1$C_POINTERS              =0000000A                   #-1670   (15)
FM1$W_EX_FILNUM             =00000002                   #-1534   (13)
FM2$S_COUNT2                =0000000E                   #-1769   (16)
FM2$S_FORMAT                =00000002                   #-1716   (16)
FM2$S_HIGHLBN               =00000006                   #-1754   (16)
FM2$V_COUNT2                =00000000                   #-1769   (16)
FM2$V_FORMAT                =0000000E                   #-1716   (16)
FM2$V_HIGHLBN               =00000008                   #-1754   (16)
FORMAT1                      00000772-R   1752   (16)     1721   (16)
FORMAT2                      00000785-R   1767   (16)     1721   (16)
FORMAT3                      0000075B-R   1730   (16)
FORMDIRSTRING                000001F1-R   729    (7)    #-457    (2)    #-460    (2)
GETRTRVPTR                   00000739-R   1700   (16)   #-1624   (14)   #-1863   (17)
HDRCNT                       FFFFFFFC     1469   (13)   #-1505   (13)     1508   (13)   #-1540   (13)     1543   (13)
HM1$L_IBMAPLBN              =00000002                   #-851    (8)
HM1$W_CHECKSUM1             =0000003A                     135    (1)    #-829    (8)
HM1$W_CHECKSUM2             =000001FE                     136    (1)      137    (1)
HM1$W_IBMAPSIZE             =00000000                   #-852    (8)
HM1$W_STRUCLEV              =0000000C                     134    (1)    #-836    (8)
HM2$L_IBMAPLBN              =00000018                   #-842    (8)
HM2$L_MAXFILES              =0000001C                   #-844    (8)
HM2$W_CHECKSUM1             =0000003A                     135    (1)
HM2$W_CHECKSUM2             =000001FE                     136    (1)
HM2$W_CLUSTER               =0000000E                   #-846    (8)
HM2$W_IBMAPSIZE             =00000020                   #-843    (8)
HM2$W_STRUCLEV              =0000000C                     134    (1)
INCRSB                       00000782-R   1757   (16)   #-1711   (16)   #-1734   (16)   #-1771   (16)
INIRTRVPTRSCAN               0000071B-R   1665   (15)   #-1617   (14)   #-1859   (17)
IO$_READLBLK                =00000021                   #-1107   (9)    #-1393   (12)   #-1612   (14)   #-821    (8)
IO$_WRITELBLK               =00000020                   #-1607   (14)
IOFUNCTION                  =FFFFFFFC     1601   (14)   #-1636   (14)
IXFHDR                      =0000000C     1456   (13)     1485   (13)   #-1486   (13)   #-1512   (13)   #-433    (2)
                                                        #-439    (2)    #-819    (8)
```

```
LBN_CACHE_CNT                   =00000018     534    (3)     #-555    (3)
LIB$CVT_DTB                      00000000-XR                   1189    (10)
NAMBLK                           FFFFFFA4     309    (2)      352    (2)     #-467    (2)
NAMDSC                           FFFFFF98     309    (2)     #-352    (2)      434    (2)     #-461    (2)     #-471    (2)
                                                             #-473    (2)
OPENFILE_1                       0000000C-R   348    (2)
OPENFILE_2                       0000000E-R   350    (2)
PLACEMENT                        00000769-R   1741   (16)     1721    (16)
RDWRTVBN                         000006DE-R   1614   (14)    #-1608   (14)
READ_DIR_HEADER                  000003BC-R   1062   (9)     #-1022   (9)
READ_DIR_LBN                     000005CA-R   1382   (12)    #-1214   (10)    #-1309   (11)
RTRVPTRBUF                      =00000010     1838   (17)    #-1478   (13)    #-1480   (13)    #-1846   (17)    #-1848   (17)
                                                             #-428    (2)     #-475    (2)
RTRVPTRLEN                      =0000000C     1836   (17)    #-1483   (13)    #-1520   (13)    #-1852   (17)    #-1882   (17)
                                                             #-430    (2)
SAVABS...                       =FFFFFFF0     1469   (13)
SCRATCHSIZE                      FFFFFF98     309    (2)     #-351    (2)
SCRATCH_SIZE                     FFFFFFD0     959    (9)     #-965    (9)     #-966    (9)      978    (9)
SS$_BADCHKSUM                   =00000808                    #-2013   (19)
SS$_BADFILEHDR                  =00000810                    #-1957   (18)
SS$_BADFILENAME                 =00000818                    #-1147   (9)
SS$_BADIRECTORY                 =00000828                    #-1144   (9)
SS$_ENDOFFILE                   =00000870                    #-1630   (14)
SS$_FILESTRUCT                  =000008C0                    #-837    (8)
SS$_NORMAL                      =00000001                    #-1030   (9)     #-1263   (10)    #-1544   (13)    #-1887   (17)
                                                             #-568    (3)     #-618    (4)     #-880    (8)
SS$_NOSUCHFILE                  =00000910                    #-1212   (10)    #-1344   (11)    #-1959   (18)
STATBLK                         =00000008     1835   (17)    #-1066   (9)      1488    (13)    #-1489   (13)    #-1886   (17)
STORE3DIGITS                     000001D8-R   687    (6)     #-745    (7)     #-749    (7)
TMPRTRVDSC                       FFFFFFF0     1469   (13)    #-1482   (13)     1513    (13)    #-1521   (13)    #-1523   (13)
                                                             #-1524   (13)    #-1525   (13)
TMPRTRVLEN                       FFFFFFF8     1469   (13)     1514    (13)    #-1518   (13)
VBN                             =00000008     1595   (14)    #-1623   (14)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEFINI         1                           100     (1)     101     (1)     102     (1)     103     (1)     92      (1)
                                            93      (1)     94      (1)     95      (1)     96      (1)     97      (1)
                                            98      (1)     99      (1)
$FATDEF         3           92      (1)     92      (1)
$FH1DEF         2           93      (1)     93      (1)
$FH2DEF         6           94      (1)     94      (1)
$FIDDEF         1           97      (1)     97      (1)
$FM1DEF         2           95      (1)     95      (1)
$FM2DEF         2           96      (1)     96      (1)
$HM1DEF         2           98      (1)     98      (1)
$HM2DEF         4           99      (1)     99      (1)
$IODEF          15          100     (1)     100     (1)
$OFFSET         2           157     (1)     1465    (13)    157     (1)     172     (1)     303     (2)     953     (9)
$OFFST1         1           168     (1)     1469    (13)    168     (1)     184     (1)     309     (2)     959     (9)
$PSLDEF         2           101     (1)     101     (1)
$SSDEF          21          102     (1)     102     (1)
$VADEF          1           103     (1)     103     (1)
ASSUME          1           131     (1)     1157    (10)    131     (1)     132     (1)     133     (1)     134     (1)
                                            135     (1)     136     (1)     137     (1)     138     (1)     143     (1)
                                            146     (1)     1485    (13)    1488    (13)    540     (3)     553     (3)
                                            978     (9)     979     (9)     998     (9)
CASE            1           833     (8)     1717    (16)    1930    (18)    833     (8)
READLBN         1           117     (1)     1393    (12)
READVBN         1           107     (1)     1506    (13)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ACBL	00F1	1335 (11)
ADDB3	0081	395 (2)
ADDL	00C0	1012 (9) 1051 (9) 1125 (9) 1137 (9) 1231 (10) 1504 (13) 1520 (13) 1524 (13) 1529 (13) 1670 (15) 1743 (16) 1874 (17) 1882 (17) 477 (2) 756 (7)
ADDL3	00C1	1050 (9) 1087 (9) 1257 (10) 1638 (14) 551 (3) 559 (3) 747 (7) 861 (8) 986 (9) 994 (9) 995 (9)
ADDW	00A0	2007 (19) 854 (8)
AOBLEQ	00F3	1211 (10) 1336 (11)
ASHL	0078	1097 (9) 1106 (9) 1124 (9) 555 (3) 876 (8)
ASHQ	0079	1755 (16)
BBC	00E1	1155 (10) 1498 (13)
BBCS	00E3	1543 (13)
BBS	00E0	1508 (13) 1531 (13) 1668 (15) 1701 (16) 1856 (17)
BBSS	00E2	878 (8)
BEQL	0013	1011 (9) 1022 (9) 1028 (9) 1081 (9) 1094 (9) 1098 (9) 1172 (10) 1178 (10) 1205 (10) 1226 (10) 1229 (10) 1248 (10) 1287 (11) 1313 (11) 1327 (11) 1384 (12) 1481 (13) 1519 (13) 1849 (17) 1870 (17) 1876 (17) 1881 (17) 1947 (18) 360 (2) 370 (2) 372 (2) 382 (2) 385 (2) 392 (2) 438 (2) 451 (2) 454 (2) 648 (5) 654 (5) 657 (5) 741 (7) 877 (8) 973 (9)
BGEQU	001E	1332 (11)
BGTR	0014	1008 (9) 1210 (10) 1259 (10) 358 (2) 470 (2) 474 (2) 985 (9)
BGTRU	001A	1230 (10)
BICL	00CA	1220 (10) 1851 (17)
BICL3	00CB	558 (3)
BLBC	00E9	1052 (9) 1064 (9) 1112 (9) 1190 (10) 1291 (11) 1507 (13) 1871 (17) 444 (2) 465 (2) 561 (3) 566 (3) 827 (8) 864 (8)
BLBS	00E8	1138 (9) 1394 (12)
BLEQ	0015	1100 (9) 1522 (13) 1541 (13) 557 (3) 689 (6) 738 (7) 983 (9)
BLSS	0019	1075 (9) 1206 (10) 1479 (13) 1626 (14) 1847 (17) 1951 (18) 429 (2) 476 (2) 543 (3) 550 (3)
BLSSU	001F	1014 (9) 1233 (10) 1629 (14) 1884 (17)
BNEQ	0012	1001 (9) 1085 (9) 1119 (9) 1160 (10) 1176 (10) 1250 (10) 1252 (10) 1278 (11) 1280 (11) 1317 (11) 1320 (11) 1329 (11) 1343 (11) 1501 (13) 1527 (13) 1865 (17) 1949 (18) 1954 (18) 2010 (19) 394 (2) 398 (2) 969 (9) 988 (9) 990 (9)
BRB	0011	1003 (9) 1023 (9) 1199 (10) 1215 (10) 1234 (10) 1253 (10) 1298 (11) 1330 (11) 1608 (14) 1674 (15) 1711 (16) 1734 (16) 1771 (16) 1861 (17) 1934 (18) 1940 (18) 1956 (18) 440 (2) 459 (2) 694 (6) 765 (7) 847 (8) 996 (9)
BRW	0031	1053 (9) 1054 (9) 1056 (9) 1139 (9) 1269 (10) 1352 (11) 1502 (13) 1536 (13) 399 (2) 690 (6) 739 (7)
BSBB	0010	1617 (14) 1624 (14) 745 (7) 749 (7)
BSBW	0030	1214 (10) 1309 (11) 1511 (13) 1859 (17) 1863 (17) 457 (2) 460 (2) 560 (3) 830 (8) 832 (8) 869 (8)
CALLG	00FA	1063 (9) 441 (2) 478 (2) 826 (8)
CALLS	00FB	1111 (9) 1189 (10) 1290 (11) 1393 (12) 1506 (13) 1517 (13) 1640 (14) 464 (2) 565 (3) 862 (8)
CASEB	008F	1933 (18) 836 (8)

```
CASEW           00AF        1721  (16)
CLRL            00D4        1169  (10)    1296  (11)    1351  (11)    1476  (13)    1483  (13)    1535  (13)    1744  (16)
                            1852  (17)    1860  (17)    2005  (19)     373   (2)     472   (2)     563   (3)     646   (5)
                             866   (8)     967   (9)
CLRQ            007C        1477  (13)    1495  (13)    1845  (17)     427   (2)     652   (5)
CLRW            00B4        1297  (11)
CMPB            0091        1277  (11)     391   (2)     393   (2)
CMPC3           0029         999   (9)
CMPC5           002D        1204  (10)
CMPL            00D1        1007   (9)    1013   (9)    1099   (9)    1232  (10)    1258  (10)    1316  (11)    1318  (11)
                            1478  (13)    1521  (13)    1625  (14)    1628  (14)    1846  (17)    1883  (17)    1948  (18)
                             428   (2)     475   (2)     556   (3)     688   (6)     737   (7)     968   (9)     982   (9)
                             984   (9)     989   (9)
CMPW            00B1        1158  (10)    1228  (10)    1279  (11)    1328  (11)    1331  (11)    1952  (18)    2009  (19)
                             371   (2)     987   (9)
DECL            00D7        1086   (9)    1385  (12)    1496  (13)     396   (2)
EXTZV           00EF        1716  (16)    1754  (16)    1769  (16)    1858  (17)
INCL            00D6        1219  (10)    1505  (13)    1758  (16)     376   (2)
INSV            00F0        1710  (16)    1732  (16)
LOCC            003A        1171  (10)    1175  (10)    1177  (10)     384   (2)     397   (2)     656   (5)     730   (7)
                             740   (7)
MCOML           00D2        1526  (13)
MNEGL           00CE        1475  (13)    1742  (16)    1938  (18)
MOVAB           009E        1117   (9)    1173  (10)    1174  (10)    1208  (10)    1221  (10)    1223  (10)     386   (2)
                             387   (2)     402   (2)     735   (7)     752   (7)     755   (7)
MOVAL           00DE        1079   (9)    1082   (9)    1310  (11)    1314  (11)    1387  (12)     352   (2)     374   (2)
                             403   (2)     439   (2)     452   (2)     649   (5)     744   (7)     748   (7)
MOVAQ           007E        1493  (13)
MOVAW           003E        1533  (13)    1667  (15)    1671  (15)
MOVB            0090        1089   (9)    1499  (13)     693   (6)     695   (6)     992   (9)
MOVC3           0028        1029   (9)    1241  (10)    1261  (10)     467   (2)     762   (7)     980   (9)     993   (9)
MOVC5           002C         966   (9)
MOVL            00D0        1002   (9)    1009   (9)    1021   (9)    1039   (9)    1065   (9)    1101   (9)    1102   (9)
                            1116   (9)    1123   (9)    1191  (10)    1254  (10)    1256  (10)    1260  (10)    1285  (11)
                            1292  (11)    1295  (11)    1311  (11)    1315  (11)    1334  (11)    1349  (11)    1350  (11)
                            1386  (12)    1392  (12)    1480  (13)    1492  (13)    1500  (13)    1509  (13)    1510  (13)
                            1512  (13)    1518  (13)    1523  (13)    1528  (13)    1534  (13)    1615  (14)    1733  (16)
                            1753  (16)    1770  (16)    1848  (17)    1853  (17)    1873  (17)    1877  (17)    1878  (17)
                            1939  (18)    1945  (18)     357   (2)     369   (2)     381   (2)     400   (2)     449   (2)
                             546   (3)     547   (3)     554   (3)     568   (3)     614   (4)     615   (4)     616   (4)
                             617   (4)     618   (4)     653   (5)     658   (5)     660   (5)     734   (7)     751   (7)
                             753   (7)     819   (8)     828   (8)     831   (8)     842   (8)     844   (8)     853   (8)
                             865   (8)     868   (8)     970   (9)     971   (9)
MOVQ            007D        1036   (9)    1168  (10)    1170  (10)    1239  (10)    1482  (13)    1486  (13)    1489  (13)
                            1491  (13)    1530  (13)    1542  (13)    1850  (17)    1886  (17)     383   (2)     430   (2)
                             432   (2)     455   (2)     456   (2)     458   (2)     461   (2)     471   (2)     541   (3)
                             567   (3)     655   (5)     981   (9)
MOVW            00B0        1040   (9)    1088   (9)    1122   (9)    1255  (10)    1333  (11)     448   (2)     544   (3)
                             692   (6)     754   (7)     871   (8)
MOVZBL          009A        1203  (10)    1532  (13)    1666  (15)    1669  (15)    1673  (15)    1708  (16)    1756  (16)
                            1855  (17)    1857  (17)     375   (2)     453   (2)     650   (5)
MOVZWL          003C        1030   (9)    1048   (9)    1049   (9)    1107   (9)    1144   (9)    1147   (9)    1207  (10)
                            1212  (10)    1222  (10)    1240  (10)    1263  (10)    1293  (11)    1344  (11)    1393  (12)
                            1497  (13)    1503  (13)    1544  (13)    1607  (14)    1612  (14)    1616  (14)    1630  (14)
                            1709  (16)    1768  (16)    1854  (17)    1887  (17)    1944  (18)    1957  (18)    1959  (18)
                            2003  (19)    2013  (19)     821   (8)     829   (8)     837   (8)     843   (8)     852   (8)
                             880   (8)
```

MULL3	00C5	548	(3)												
MULW3	00A5	846	(8)												
POPR	00BA	1885	(17)	1955	(18)	468	(2)	662	(5)	750	(7)	763	(7)	764	(7)
PUSHAB	009F	1108	(9)	1187	(10)	1188	(10)								
PUSHAL	00DF	1185	(10)	1289	(11)	1393	(12)	1506	(13)	1513	(13)	1514	(13)	431	(2)
		434	(2)	462	(2)	562	(3)								
PUSHL	00DD	1109	(9)	1110	(9)	1288	(11)	1393	(12)	1506	(13)	1515	(13)	1516	(13)
		1636	(14)	1637	(14)	1639	(14)	433	(2)	435	(2)	436	(2)	463	(2)
		564	(3)	742	(7)	822	(8)	823	(8)	824	(8)	825	(8)	867	(8)
PUSHR	00BB	466	(2)	651	(5)	736	(7)	758	(7)						
RET	0004	1031	(9)	1058	(9)	1145	(9)	1148	(9)	1213	(10)	1264	(10)	1345	(11)
		1395	(12)	1545	(13)	1631	(14)	1641	(14)	1888	(17)	1958	(18)	1960	(18)
		2014	(19)	479	(2)	569	(3)	619	(4)	838	(8)	881	(8)		
ROTL	009C	1083	(9)	1393	(12)	1635	(14)	1731	(16)	820	(8)	851	(8)		
RSB	0005	1388	(12)	1672	(15)	1745	(16)	1759	(16)	2011	(19)	663	(5)	698	(6)
		757	(7)												
SOBGEQ	00F4	697	(6)												
SOBGTR	00F5	2008	(19)												
SUBB3	0083	1080	(9)												
SUBL	00C2	1183	(10)	1525	(13)	1627	(14)	1879	(17)	351	(2)	377	(2)	549	(3)
		965	(9)	991	(9)										
SUBL3	00C3	1047	(9)	1066	(9)	1095	(9)	1623	(14)	401	(2)	542	(3)	659	(5)
		731	(7)	733	(7)	743	(7)	746	(7)						
TSTB	0095	1010	(9)	1118	(9)	1249	(10)								
TSTL	00D5	1027	(9)	1093	(9)	1225	(10)	1247	(10)	1251	(10)	1286	(11)	1341	(11)
		1383	(12)	1540	(13)	1864	(17)	1869	(17)	1875	(17)	1880	(17)	1950	(18)
		437	(2)	450	(2)	469	(2)	473	(2)	647	(5)	972	(9)		
TSTW	00B5	1084	(9)	1209	(10)	1312	(11)	1326	(11)	1946	(18)				

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	205	(1)	207	(1)	209	(1)										
.BLKB	1469	(13)	168	(1)	184	(1)	309	(2)	959	(9)						
.CROSS	127	(1)														
.DSABL	481	(2)														
.ENABL	341	(2)														
.END	2017	(19)														
.ENDC	1157	(10)	131	(1)	132	(1)	133	(1)	134	(1)	135	(1)	136	(1)	137	(1)
	138	(1)	143	(1)	146	(1)	1469	(13)	1485	(13)	1488	(13)	168	(1)	184	(1)
	309	(2)	540	(3)	553	(3)	959	(9)	978	(9)	979	(9)	998	(9)		
.ENDM	100	(1)	101	(1)	102	(1)	103	(1)	115	(1)	126	(1)	131	(1)	157	(1)
	168	(1)	833	(8)	92	(1)	93	(1)	94	(1)	95	(1)	96	(1)	97	(1)
	98	(1)	99	(1)												
.ENDR	1469	(13)	168	(1)	1721	(16)	184	(1)	1933	(18)	309	(2)	836	(8)	959	(9)
.GLOBAL	191	(1)														
.IDENT	10	(1)														
.IF	1157	(10)	131	(1)	132	(1)	133	(1)	134	(1)	135	(1)	136	(1)	137	(1)
	138	(1)	143	(1)	146	(1)	1469	(13)	1485	(13)	1488	(13)	168	(1)	184	(1)
	309	(2)	540	(3)	553	(3)	959	(9)	978	(9)	979	(9)	998	(9)		
.IFF	1157	(10)	131	(1)	132	(1)	133	(1)	134	(1)	135	(1)	136	(1)	137	(1)
	138	(1)	143	(1)	146	(1)	1469	(13)	1485	(13)	1488	(13)	168	(1)	184	(1)
	309	(2)	540	(3)	553	(3)	959	(9)	978	(9)	979	(9)	998	(9)		
.IRP	1469	(13)	168	(1)	1721	(16)	184	(1)	1933	(18)	309	(2)	836	(8)	959	(9)
.LIST	1393	(12)	1469	(13)	1506	(13)	168	(1)	184	(1)	309	(2)	959	(9)		
.MACRO	107	(1)	117	(1)												
.NLIST	1393	(12)	1469	(13)	1506	(13)	168	(1)	184	(1)	309	(2)	959	(9)		
.NOCROSS	100	(1)	101	(1)	102	(1)	103	(1)	90	(1)	92	(1)	93	(1)	94	(1)
	95	(1)	96	(1)	97	(1)	98	(1)	99	(1)						
.PAGE	85	(1)														
.PSECT	1469	(13)	168	(1)	184	(1)	202	(1)	309	(2)	959	(9)				
.RESTORE	100	(1)	101	(1)	102	(1)	103	(1)	1469	(13)	168	(1)	184	(1)	309	(2)
	91	(1)	92	(1)	93	(1)	94	(1)	95	(1)	959	(9)	96	(1)	97	(1)
	98	(1)	99	(1)												
.SAVE	100	(1)	101	(1)	102	(1)	103	(1)	1469	(13)	168	(1)	184	(1)	309	(2)
	92	(1)	93	(1)	94	(1)	95	(1)	959	(9)	96	(1)	97	(1)	98	(1)
	99	(1)														
.SBTTL	1150	(10)	1271	(11)	1354	(12)	1397	(13)	1547	(14)	1643	(15)	1676	(16)	1773	(17)
	1890	(18)	1962	(19)	211	(2)	483	(3)	571	(4)	621	(5)	665	(6)	700	(7)
	767	(8)	86	(1)	883	(9)										
.SIGNED_WORD	1721	(16)	1933	(18)	836	(8)										
.TITLE	9	(1)														
.WEAK	195	(1)	196	(1)	197	(1)										
.WORD	1474	(13)	1606	(14)	1611	(14)	1844	(17)	349	(2)	538	(3)	613	(4)	818	(8)
	964	(9)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	54	1029	(9)	1063	(9)	#-1065	(9)	#-1066	(9)	#-1110	(9)	#-1168	(10)
		1241	(10)	#-1285	(11)	#-1295	(11)	#-1334	(11)	#-1341	(11)	#-1349	(11)
		#-1350	(11)	#-1392	(12)	#-1393	(12)	#-1478	(13)	#-1480	(13)	#-1483	(13)
		#-1486	(13)	#-1489	(13)	#-1506	(13)	#-1512	(13)	#-1520	(13)	#-1542	(13)
		#-1615	(14)	#-1623	(14)	#-1637	(14)	#-1639	(14)	#-1846	(17)	#-1848	(17)
		#-1852	(17)	#-1853	(17)	#-1882	(17)	#-1886	(17)	#-381	(2)	#-428	(2)
		#-430	(2)	#-432	(2)	#-433	(2)	#-435	(2)	#-475	(2)	#-541	(3)
		#-548	(3)	#-555	(3)	#-564	(3)	#-653	(5)	#-819	(8)	#-824	(8)
		#-968	(9)	#-970	(9)	#-971	(9)	980	(9)	#-981	(9)		
FP	63	#-1002	(9)	#-1009	(9)	#-1010	(9)	#-1021	(9)	#-1036	(9)	#-1039	(9)
		#-1040	(9)	#-1047	(9)	#-1048	(9)	#-1049	(9)	#-1050	(9)	#-1052	(9)
		#-1087	(9)	#-1088	(9)	#-1089	(9)	#-1109	(9)	#-1116	(9)	#-1117	(9)
		#-1118	(9)	#-1122	(9)	#-1123	(9)	#-1249	(10)	#-1251	(10)	#-1254	(10)
		#-1255	(10)	1261	(10)	#-1383	(12)	#-1385	(12)	#-1386	(12)	#-1387	(12)
		#-1482	(13)	#-1505	(13)	1508	(13)	1513	(13)	1514	(13)	#-1518	(13)
		#-1521	(13)	#-1523	(13)	#-1524	(13)	#-1525	(13)	#-1540	(13)	1543	(13)
		#-1636	(14)	352	(2)	431	(2)	434	(2)	#-448	(2)	#-449	(2)
		#-461	(2)	467	(2)	#-471	(2)	#-473	(2)	744	(7)	748	(7)
		752	(7)	755	(7)	762	(7)	#-992	(9)	993	(9)	999	(9)
R0	146	#-1007	(9)	#-1030	(9)	#-1064	(9)	#-1112	(9)	#-1144	(9)	#-1147	(9)
		1173	(10)	#-1183	(10)	1188	(10)	#-1190	(10)	#-1207	(10)	1208	(10)
		#-1212	(10)	#-1222	(10)	1223	(10)	#-1257	(10)	#-1258	(10)	#-1260	(10)
		#-1263	(10)	#-1291	(11)	#-1295	(11)	#-1296	(11)	#-1297	(11)	#-1315	(11)
		#-1316	(11)	#-1318	(11)	#-1328	(11)	#-1331	(11)	#-1333	(11)	#-1344	(11)
		#-1394	(12)	#-1480	(13)	#-1482	(13)	#-1500	(13)	#-1504	(13)	#-1506	(13)
		#-1507	(13)	#-1509	(13)	#-1532	(13)	1533	(13)	#-1534	(13)	#-1544	(13)
		#-1625	(14)	#-1627	(14)	#-1630	(14)	#-1666	(15)	1667	(15)	#-1708	(16)
		#-1716	(16)	#-1721	(16)	#-1731	(16)	1732	(16)	#-1744	(16)	#-1753	(16)
		1754	(16)	#-1755	(16)	#-1756	(16)	#-1758	(16)	#-1768	(16)	1769	(16)
		#-1848	(17)	#-1850	(17)	#-1855	(17)	#-1857	(17)	1858	(17)	#-1869	(17)
		#-1874	(17)	#-1877	(17)	#-1887	(17)	#-1948	(18)	#-1952	(18)	#-1955	(18)
		#-1957	(18)	#-1959	(18)	#-2003	(19)	#-2008	(19)	#-2013	(19)	#-381	(2)
		#-383	(2)	387	(2)	#-395	(2)	#-397	(2)	402	(2)	#-444	(2)
		#-453	(2)	#-456	(2)	#-461	(2)	#-465	(2)	#-466	(2)	#-468	(2)
		#-542	(3)	#-549	(3)	#-556	(3)	#-558	(3)	#-561	(3)	#-566	(3)
		#-568	(3)	#-614	(4)	#-615	(4)	#-616	(4)	#-617	(4)	#-618	(4)
		#-646	(5)	#-650	(5)	#-651	(5)	#-653	(5)	#-655	(5)	#-660	(5)
		#-688	(6)	#-697	(6)	#-731	(7)	#-733	(7)	735	(7)	#-736	(7)
		#-737	(7)	#-740	(7)	#-742	(7)	#-743	(7)	#-746	(7)	#-751	(7)
		752	(7)	#-756	(7)	#-764	(7)	#-827	(8)	#-829	(8)	#-837	(8)
		#-843	(8)	#-852	(8)	#-854	(8)	#-861	(8)	#-864	(8)	#-868	(8)
		#-880	(8)	#-981	(9)	#-982	(9)	#-984	(9)	#-986	(9)	#-991	(9)
		#-992	(9)	#-993	(9)								
R1	73	#-1124	(9)	#-1125	(9)	1174	(10)	1187	(10)	#-1314	(11)	#-1316	(11)
		#-1318	(11)	#-1328	(11)	#-1331	(11)	#-1333	(11)	#-1503	(13)	#-1504	(13)
		#-1510	(13)	#-1518	(13)	#-1520	(13)	#-1521	(13)	#-1523	(13)	#-1524	(13)
		#-1525	(13)	#-1526	(13)	#-1638	(14)	#-1709	(16)	1710	(16)	#-1733	(16)
		#-1742	(16)	#-1754	(16)	#-1770	(16)	#-1873	(17)	#-1878	(17)	#-1933	(18)
		#-1939	(18)	#-1944	(18)	#-1945	(18)	#-2007	(19)	#-2009	(19)	386	(2)
		#-401	(2)	403	(2)	#-452	(2)	#-453	(2)	#-466	(2)	#-468	(2)

```
                #-548   (3)     #-549   (3)     #-551   (3)     #-555   (3)     #-556   (3)     #-558   (3)
                #-559   (3)     #-649   (5)     #-650   (5)     #-651   (5)     #-659   (5)     #-695   (6)
                #-733   (7)     #-734   (7)       735   (7)     #-736   (7)       740   (7)     #-747   (7)
                #-752   (7)     #-753   (7)     #-754   (7)     #-755   (7)     #-764   (7)     #-828   (8)
                #-831   (8)     #-842   (8)     #-851   (8)     #-861   (8)     #-865   (8)     #-986   (9)
                  993   (9)
R10       12    #-1080  (9)     #-1089  (9)     #-1138  (9)     #-1169  (10)    #-1191  (10)    #-1225  (10)
                #-1228  (10)      538   (3)     #-541   (3)     #-542   (3)     #-567   (3)       964   (9)
R11       39    #-1050  (9)     #-1093  (9)     #-1095  (9)     #-1102  (9)       1108  (9)       1117  (9)
                #-1125  (9)     #-1247  (10)    #-1256  (10)    #-1258  (10)    #-1260  (10)      1261  (10)
                  349   (2)     #-357   (2)     #-369   (2)     #-371   (2)     #-373   (2)     #-437   (2)
                  439   (2)     #-463   (2)       538   (3)     #-544   (3)     #-546   (3)     #-547   (3)
                #-551   (3)     #-554   (3)     #-559   (3)       562   (3)       964   (9)     #-967   (9)
                #-970   (9)     #-972   (9)     #-994   (9)     #-995   (9)
R2        63    #-1095  (9)     #-1097  (9)     #-1099  (9)     #-1101  (9)     #-1106  (9)     #-1116  (9)
                #-1122  (9)     #-1124  (9)     #-1311  (11)    #-1312  (11)      1314  (11)    #-1334  (11)
                #-1335  (11)    #-1349  (11)      1474  (13)    #-1486  (13)      1498  (13)    #-1503  (13)
                  1506  (13)    #-1512  (13)      1606  (14)      1611  (14)    #-1616  (14)      1668  (15)
                  1701  (16)      1844  (17)    #-1854  (17)      1856  (17)    #-1885  (17)    #-1886  (17)
                #-2005  (19)    #-2007  (19)    #-2009  (19)       349  (2)     #-383   (2)     #-384   (2)
                #-387   (2)     #-396   (2)     #-397   (2)     #-402   (2)     #-466   (2)     #-468   (2)
                #-471   (2)     #-472   (2)       538   (3)     #-652   (5)     #-655   (5)     #-656   (5)
                #-662   (5)     #-692   (6)     #-693   (6)     #-695   (6)     #-736   (7)     #-744   (7)
                #-748   (7)     #-750   (7)     #-764   (7)       818   (8)       964   (9)     #-986   (9)
                #-987   (9)     #-989   (9)
R3        76    #-1102  (9)       1108  (9)       1117  (9)     #-1123  (9)     #-1170  (10)    #-1171  (10)
                #-1173  (10)    #-1175  (10)    #-1177  (10)    #-1221  (10)    #-1228  (10)    #-1231  (10)
                #-1232  (10)    #-1239  (10)      1241  (10)    #-1285  (11)    #-1286  (11)    #-1288  (11)
                  1289  (11)    #-1292  (11)    #-1293  (11)    #-1315  (11)      1474  (13)      1506  (13)
                #-1510  (13)    #-1516  (13)    #-1530  (13)      1531  (13)    #-1532  (13)      1533  (13)
                  1606  (14)      1611  (14)    #-1623  (14)    #-1625  (14)    #-1627  (14)    #-1638  (14)
                  1844  (17)    #-1860  (17)    #-1864  (17)    #-1874  (17)       349  (2)       384  (2)
                #-386   (2)     #-391   (2)     #-393   (2)     #-395   (2)       397  (2)     #-400   (2)
                #-401   (2)     #-403   (2)     #-466   (2)     #-468   (2)       538  (3)       656  (5)
                #-658   (5)     #-659   (5)     #-662   (5)     #-758   (7)     #-763   (7)       818  (8)
                #-819   (8)     #-822   (8)     #-828   (8)     #-831   (8)     #-836   (8)     #-842   (8)
                #-843   (8)     #-844   (8)     #-846   (8)     #-851   (8)     #-852   (8)     #-865   (8)
                #-871   (8)       878   (8)       964   (9)
R4        60    #-1079  (9)     #-1082  (9)     #-1083  (9)     #-1084  (9)     #-1159  (10)      1171  (10)
                #-1174  (10)      1175  (10)      1177  (10)    #-1203  (10)    #-1204  (10)    #-1219  (10)
                #-1220  (10)      1221  (10)    #-1277  (11)    #-1279  (11)    #-1310  (11)    #-1335  (11)
                  1474  (13)    #-1489  (13)    #-1495  (13)    #-1496  (13)    #-1526  (13)    #-1528  (13)
                #-1529  (13)      1606  (14)      1611  (14)    #-1628  (14)    #-1667  (15)    #-1669  (15)
                #-1670  (15)      1671  (15)    #-1708  (16)    #-1709  (16)    #-1710  (16)      1716  (16)
                #-1731  (16)    #-1733  (16)    #-1743  (16)    #-1753  (16)    #-1756  (16)    #-1768  (16)
                #-1770  (16)      1844  (17)    #-1883  (17)       349  (2)     #-466   (2)     #-468   (2)
                  538   (3)     #-758   (7)     #-763   (7)       818  (8)     #-844   (8)     #-846   (8)
                #-853   (8)     #-854   (8)     #-871   (8)     #-876   (8)       964   (9)
R5        57    #-1065  (9)       1079  (9)     #-1080  (9)       1082  (9)       1155  (10)    #-1203  (10)
                  1204  (10)    #-1207  (10)      1208  (10)    #-1209  (10)      1221  (10)    #-1222  (10)
                  1223  (10)    #-1232  (10)      1310  (11)    #-1311  (11)    #-1386  (12)      1387  (12)
                #-1392  (12)      1393  (12)      1474  (13)    #-1491  (13)    #-1492  (13)    #-1497  (13)
                #-1499  (13)    #-1509  (13)    #-1530  (13)    #-1534  (13)    #-1535  (13)    #-1542  (13)
                  1606  (14)      1611  (14)    #-1615  (14)    #-1616  (14)    #-1628  (14)    #-1666  (15)
                  1667  (15)    #-1669  (15)      1671  (15)    #-1673  (15)      1844  (17)    #-1853  (17)
                #-1854  (17)    #-1855  (17)    #-1857  (17)    #-1883  (17)       349  (2)     #-466   (2)
                #-468   (2)       538   (3)     #-758   (7)     #-763   (7)       964  (9)
```

```
R6                33      #-1047  (9)     #-1051  (9)     #-1066  (9)     #-1087  (9)     #-1137  (9)     #-1211  (10)
                          #-1239  (10)    #-1254  (10)    #-1336  (11)    #-1350  (11)    #-1393  (12)      1474  (13)
                          #-1493  (13)    #-1515  (13)    #-1528  (13)    #-1529  (13)      1844  (17)    #-1845  (17)
                          #-1850  (17)    #-1851  (17)    #-1875  (17)    #-1879  (17)       349  (2)     #-375   (2)
                          #-377   (2)     #-401   (2)     #-455   (2)     #-456   (2)     #-458   (2)     #-469   (2)
                          #-730   (7)     #-731   (7)       964   (9)
R7                28      #-1048  (9)     #-1051  (9)     #-1083  (9)     #-1086  (9)     #-1088  (9)     #-1099  (9)
                          #-1101  (9)     #-1137  (9)     #-1211  (10)    #-1240  (10)    #-1255  (10)    #-1336  (11)
                          #-1351  (11)      1844  (17)    #-1877  (17)    #-1878  (17)    #-1880  (17)       349  (2)
                          #-374   (2)     #-375   (2)     #-376   (2)     #-400   (2)       730  (7)     #-733   (7)
                          #-734   (7)     #-735   (7)       964   (9)
R8                20        1000  (9)     #-1002  (9)     #-1009  (9)     #-1012  (9)     #-1013  (9)     #-1021  (9)
                          #-1027  (9)       1029  (9)     #-1036  (9)     #-1039  (9)     #-1040  (9)     #-1168  (10)
                          #-1170  (10)    #-1183  (10)    #-1204  (10)    #-1256  (10)    #-1257  (10)      1261  (10)
                            964   (9)     #-994   (9)
R9                 4      #-1013  (9)       1204  (10)      964   (9)     #-995   (9)
SP                64      #-1106  (9)     #-1107  (9)       1185  (10)    #-1191  (10)    #-1293  (11)    #-1326  (11)
                          #-1393  (12)    #-1475  (13)    #-1476  (13)    #-1477  (13)    #-1491  (13)    #-1492  (13)
                            1493  (13)    #-1497  (13)    #-1499  (13)    #-1500  (13)    #-1607  (14)    #-1612  (14)
                          #-1635  (14)    #-1638  (14)    #-1858  (17)    #-1871  (17)    #-1873  (17)    #-1938  (18)
                          #-1939  (18)    #-1944  (18)    #-1945  (18)    #-1946  (18)    #-1948  (18)    #-1950  (18)
                          #-1952  (18)    #-351   (2)     #-427   (2)     #-430   (2)     #-432   (2)     #-439   (2)
                            441   (2)     #-455   (2)     #-458   (2)       462   (2)     #-477   (2)       478   (2)
                          #-563   (3)     #-658   (5)     #-659   (5)     #-660   (5)     #-743   (7)     #-746   (7)
                          #-747   (7)     #-762   (7)     #-820   (8)     #-821   (8)       826   (8)     #-861   (8)
                          #-862   (8)     #-866   (8)     #-868   (8)     #-965   (9)       966   (9)       980   (9)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.19
Command processing	885	00:00:00.26	00:00:01.18
Pass 1	910	00:00:03.76	00:00:05.53
Symbol table sort	0	00:00:00.38	00:00:00.38
Pass 2	71	00:00:01.32	00:00:02.62
Symbol table output	2	00:00:00.03	00:00:00.07
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	6	00:00:00.48	00:00:00.66
Assembler run totals	1900	00:00:06.26	00:00:10.64

The working set limit was 2900 pages.
176666 bytes (346 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1265 non-local and 111 local symbols.
2017 source lines were read in Pass 1, producing 85 object records in Pass 2.
76 pages of virtual memory were used to define 25 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           9
SYS$COMMON:[SYSLIB]STARLET.MLB;2                      10
TOTALS (all libraries)                                20

1325 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:FILEREAD.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$
```

Table of contents

(1) 64 CREATE I/O DRIVER FORK PROCESS
(2) 85 CREATE FORK PROCESS
(3) 113 SOFTWARE INTERRUPT FORK DISPATCHER

```
                     0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element FRKCTL.MAR
                     0000     2 ; *1     6-FEB-1986 15:18:39 DS ""
                     0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element FRKCTL.MAR
                     0000     4         .TITLE  FRKCTL  *** FRKCTL QIO fork control
                     0000     5         .IDENT  /06-03/
                     0000     6         .LIST   MEB
                     0000     7         .NLIST  CND
                     0000     8
                     0000     9 ;
                     0000    10 ;*****************************************************************************
                     0000    11 ;*                                                                           *
                     0000    12 ;*  COPYRIGHT (c) 1976, 1977, 1978, 1979, 1980                               *
                     0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                     0000    14 ;*                                                                           *
                     0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                     0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                     0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                     0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                     0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                     0000    20 ;*  TRANSFERRED.                                                             *
                     0000    21 ;*                                                                           *
                     0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                     0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                     0000    24 ;*  CORPORATION.                                                             *
                     0000    25 ;*                                                                           *
                     0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                     0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                     0000    28 ;*                                                                           *
                     0000    29 ;*****************************************************************************
                     0000    30 ;
                     0000    31 ; D. N. CUTLER 9-AUG-76
                     0000    32 ;
                     0000    33 ; MODIFIED BY R.HEINEN ON 13-OCT-76; ALLOW FORK TO IPL 6
                     0000    34 ; FORK CREATION AND DISPATCHING
                     0000    35 ;
                     0000    36 ; MODIFIED BY  T.L. SOUTTER  20-FEB-78
                     0000    37 ;
                     0000    38 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
                     0000    39 ;
                     0000    40 ;             Dave Butenhof   13-may-1980, Version 5.4
                     0000    41 ;       02    Create .UPD file to modify VMS V2.0 source to
                     0000    42 ;             Supervisor environment
                     0000    43 ;             Dave Butenhof, 23-feb-1981, version 6.3
                     0000    44 ;       03    Fix some truncation errors.
                     0000    45 ; MACRO LIBRARY CALLS
                     0000    46 ;
                     0000    47
                     0000    48
                     0000    52
                     0000    53         $FKBDEF                          ;DEFINE FKB OFFSETS
                     0000            .SAVE   LOCAL_BLOCK
                     0000            .IIF    NB,FKB$L_FQFL,  FKB$L_FQFL:
                     0000                            .BLKL   1
00000004             0004            .IIF    NB,FKB$L_FQBL,  FKB$L_FQBL:
00000008             0004                            .BLKL   1
                     0008            .IIF    NB,FKB$W_SIZE,  FKB$W_SIZE:
0000000A             0008                            .BLKW   1
```

```
                    000A                .IIF    NB,FKB$B_TYPE,  FKB$B_TYPE:
0000000B            000A                                .BLKB   1
                    000B                .IIF    NB,FKB$B_FIPL,  FKB$B_FIPL:
0000000C            000B                                .BLKB   1
                    000C                .IIF    NB,FKB$L_FPC,   FKB$L_FPC:
00000010            000C                                .BLKL   1
                    0010                .IIF    NB,FKB$L_FR3,   FKB$L_FR3:
00000014            0010                                .BLKL   1
                    0014                .IIF    NB,FKB$L_FR4,   FKB$L_FR4:
00000018            0014                                .BLKL   1
                    0018                .IIF    NB,FKB$C_LENGTH,        FKB$C_LENGTH:
                    0018                .IIF    NB,FKB$K_LENGTH,        FKB$K_LENGTH:
                    0000                .RESTORE
                    0000    54          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
00000000            0018                .=0
                    0000                .RESTORE
                    0000    55          $UCBDEF                         ;DEFINE UCB OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
00000000            0018                .=0
                    0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                    0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004            0000                                .BLKL   1
                    0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                    0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008            0004                                .BLKL   1
                    0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
0000000A            0008                                .BLKW   1
                    000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
0000000B            000A                                .BLKB   1
                    000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
0000000C            000B                                .BLKB   1
                    000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
0000000D            000C                                .BLKB   1
00000010            000D                                .BLKB   3
                    0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014            0010                                .BLKL   1
                    0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016            0014                                .BLKW   1
                    0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018            0016                                .BLKW   1
                    0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
0000001A            0018                                .BLKW   1
                    001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
0000001C            001A                                .BLKW   1
                    001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
00000020            001C                                .BLKL   1
                    0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
```

```
00000024  0020                                  .BLKL   1
          0024                  .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
00000028  0024                                  .BLKL   1
          0028                  .IIF    NB,UCB$L_PID,   UCB$L_PID:
0000002C  0028                                  .BLKL   1
          002C                  .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030  002C                                  .BLKL   1
          0030                  .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034  0030                                  .BLKL   1
          0034                  .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039  0038                                  .BLKB   1
          0039                  .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A  0039                                  .BLKB   1
          003A                  .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C  003A                                  .BLKW   1
          003C                  .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
          003C                  .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
          003C                  .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D  003C                                  .BLKB   1
          003D                  .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
          003D                  .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E  003D                                  .BLKB   1
          003E                  .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
          003E                  .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F  003E                                  .BLKB   1
          003F                  .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040  003F                                  .BLKB   1
          0040                  .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                                  .BLKW   1
          004A                  .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
          004A                  .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                                  .BLKB   1
          004B                  .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                                  .BLKB   1
          004C                  .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                  .BLKW   1
          0052                  .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
          0052                  .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                  .BLKB   1
          0053                  .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                  .BLKB   1
          0054                  .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
          005A                  .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
          005C                  .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C                  .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
```

```
00000060  005C                          .BLKL   1
          0060          .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                          .BLKL   1
          0064          .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
          0064          .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                          .BLKL   1
          0068          .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                          .BLKL   1
          006C          .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                          .BLKW   1
          006E          .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                          .BLKW   1
          0070          .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                          .BLKB   1
          0071          .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                          .BLKB   1
          0072          .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                          .BLKW   1
          0074          .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
          0074          .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074          . = 0
          0000          .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                          .BLKW   1
00000074  0002          . = 116
          0074          .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                          .BLKL   1
          0078          .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                          .BLKL   1
          0080          .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                          .BLKL   1
          0084          .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                          .BLKL   1
          008C          .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                          .BLKL   1
          0090          .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
          0090          .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090          . = 116
          0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                          .BLKB   1
          0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                          .BLKB   1
          0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                          .BLKB   1
          0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                          .BLKB   1
          0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                          .BLKL   1
0000007E  007C                          .BLKW   1
          007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                          .BLKW   1
          0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                          .BLKL   1
00000080  0084          . = 128
```

```
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                                  .BLKW   1
          008A                  .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                                  .BLKW   1
          008C                  .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C                  .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                  .BLKW   1
          008E                  .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                  .BLKW   1
          0090                  .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                  .BLKW   1
          0092                  .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                  .BLKW   1
          0094                  .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                  .BLKB   1
          0095                  .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                  .BLKB   1
          0096                  .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                  .BLKW   1
00000096  0098                  . = 150
00000098  0096                                  .BLKW   1
          0098                  .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                  .BLKL   1
          00A0                  .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                                  .BLKL   1
          00A4                  .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                                  .BLKW   1
          00A6                  .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                                  .BLKB   1
000000A8  00A7                                  .BLKB   1
00000074  00A8                  . = 116
00000078  0074                                  .BLKL   1
0000007C  0078                                  .BLKL   1
00000080  007C                                  .BLKL   1
00000084  0080                                  .BLKL   1
00000088  0084                                  .BLKL   1
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                  .BLKL   1
00000094  0090                                  .BLKL   1
00000098  0094                                  .BLKL   1
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                  .BLKB   1
          009D                  .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                  .BLKB   1
          009E                  .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                  .BLKB   1
000000A0  009F                                  .BLKB   1
          00A0                  .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                  .BLKB   1
          00A1                  .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
```

```
000000A2  00A1                          .BLKB   1
          00A2          .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                          .BLKB   1
000000A4  00A3                          .BLKB   1
          00A4          .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                          .BLKB   1
          00A5          .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                          .BLKW   1
000000A8  00A7                          .BLKB   1
          00A8          .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                          .BLKL   1
000000B0  00AC                          .BLKL   1
000000B4  00B0                          .BLKL   1
000000B8  00B4                          .BLKL   1
          00B8          .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                          .BLKL   1
          00BC          .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC          .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC          . = 116
          0074          .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                          .BLKL   1
          0078          .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                          .BLKW   1
00000080  007E                          .BLKW   1
          0000          .RESTORE
          0000     59
          0000     60
00000000  0000     61   .PSECT  SEP, SHR, EXE, WRT, LONG
          0000     62
```

```
                        0000     64          .SBTTL  CREATE I/O DRIVER FORK PROCESS
                        0000     65 ;+
                        0000     66 ; EXE$IOFORK - CREATE I/O DRIVER FORK PROCESS
                        0000     67 ;
                        0000     68 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO CREATE A FORK PROCESS.
                        0000     69 ;
                        0000     70 ; INPUTS:
                        0000     71 ;
                        0000     72 ;       00(SP) = RETURN ADDRESS OF CALLER.
                        0000     73 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                        0000     74 ;
                        0000     75 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                        0000     76 ;
                        0000     77 ; OUTPUTS:
                        0000     78 ;
                        0000     79 ;       ***TBS***
                        0000     80 ;-
                        0000     81
                        0000     82 EXE$IOFORK::                             ;CREATE I/O DRIVER FORK PROCESS
00 58 A5   00   E5      0000     83          BBCC    #UCB$V_TIM,UCB$W_STS(R5),EXE$FORK ;DISABLE TIMEOUT
```

```
                        0005     85          .SBTTL  CREATE FORK PROCESS
                        0005     86 ;+
                        0005     87 ; EXE$FORK - CREATE FORK PROCESS
                        0005     88 ;
                        0005     89 ; THIS ROUTINE IS CALLED TO CREATE A FORK PROCESS.
                        0005     90 ;
                        0005     91 ; INPUTS:
                        0005     92 ;
                        0005     93 ;       00(SP) = RETURN ADDRESS OF CALLER.
                        0005     94 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                        0005     95 ;
                        0005     96 ;       R5 = ADDRESS OF FORK BLOCK.
                        0005     97 ;
                        0005     98 ; OUTPUTS:
                        0005     99 ;
                        0005    100 ;       ***TBS***
                        0005    101 ;-
                        0005    102
                        0005    103 EXE$FORK::                                  ;CREATE FORK PROCESS
          10 A5   53    7D    0005    104         MOVQ    R3,FKB$L_FR3(R5)      ;SAVE REGISTERS R3 AND R4
             0C A5 8ED0    0009    105         POPL    FKB$L_FPC(R5)         ;SET FORK PROCESS PC
       54    0B A5   9A    000D    106         MOVZBL  FKB$B_FIPL(R5),R4     ;GET FORK IPL
   53  FFFFFFD0'EF44   7E    0011    107         MOVAQ   L^SWI$GL_FQFL-<6*8>[R4], R3     ; Get address of fork queue
       04 B3   65    0E    0019    108         INSQUE  (R5),@4(R3)           ;INSERT FORK BLOCK IN FORK QUEUE
               03    12    001D    109         BNEQ    10$                   ;IF NEQ NOT FIRST ENTRY IN QUEUE
                     001F    110         SOFTINT R4                    ;INITIATE SOFTWARE INTERRUPT
          14   54   DA    001F                MTPR    R4,S^#PR$_SIRR
               05    0022    111 10$:    RSB                           ;
```

```
                                0023   113          .SBTTL  SOFTWARE INTERRUPT FORK DISPATCHER
                                0023   114 ;+
                                0023   115 ; EXE$FORKDSPTH - SOFTWARE INTERRUPT FORK DISPATCHER
                                0023   116 ;
                                0023   117 ; THIS ROUTINE IS AUTOMATICALLY VECTORED TO WHEN THE SOFTWARE INTERRUPT
                                0023   118 ; PRIORITY ARBITRATION LOGIC IN THE CENTRAL PROCESSOR DETECTS A PENDING
                                0023   119 ; INTERRUPT AT LEVEL 6, 7, 8, 9, 10, OR 11 AND THE CURRENT PRIORITY LEVEL IS
                                0023   120 ; LOWER THAN THE PENDING LEVEL. THE STATE OF THE STACK ON ENTRY IS:
                                0023   121 ;
                                0023   122 ;       00(SP) = INTERRUPT PC.
                                0023   123 ;       04(SP) = INTERRUPT PSL.
                                0023   124 ;
                                0023   125 ; FOR EACH OF THE LEVELS 6, 7, 8, 9, 10, AND 11, THERE EXISTS A QUEUE OF FORK
                                0023   126 ; BLOCKS WAITING TO BE PROCESSED. WHEN A FORK BLOCK IS ENTERED IN ITS
                                0023   127 ; CORRESPONDING QUEUE AND IT IS THE FIRST TO BE ENTERED (I. E. THE QUEUE
                                0023   128 ; WAS PREVIOUSLY EMPTY), A SOFTWARE INTERRUPT IS REQUESTED FOR THAT LEVEL.
                                0023   129 ; THE FORK DISPATCHER GAINS CONTROL AND EMPTIES THE QUEUE ONE ENTRY AT A
                                0023   130 ; TIME. AS EACH FORK IS DISPATCHED, REGISTERS R3 AND R4 ARE RESTORED FROM
                                0023   131 ; THE FORK BLOCK AND THE FORK PROCESS IS CALLED VIA A JSB INSTRUCTION.
                                0023   132 ; ON RETURN, THE FORK DISPATCHER RETRIEVES THE NEXT ENTRY FROM THE QUEUE
                                0023   133 ; AND REPEATS THE DISPATCHING OPERATION. THIS PROCESS CONTINUES UNTIL
                                0023   134 ; THERE ARE NO MORE FORKS TO DISPATCH AT WHICH TIME THE INTERRUPT IS
                                0023   135 ; DISMISSED.
                                0023   136 ;-
                                0023   137
                                0023   138          .ALIGN  LONG
                                0024   139 EXE$FORKDSPTH::                             ;SOFTWARE INTERRUPT FORK DISPATCHER
                          3F    BB    0024   140          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ;SAVE FORK PROCESS REGISTER SET
                       50 12    DB    0026   141 10$:     MFPR    #PR$_IPL,R0             ;READ CURRENT IPL
                51 FFFFFFD0'EF40    7E    0029   142          MOVAQ   L^SWI$GL_FQFL-<6*8>[R0], R1     ; Get address of fork queue
                       55 91    0F    0031   143          REMQUE  @(R1)+,R5               ;REMOVE NEXT ENTRY FROM FORK QUEUE
                          2D    1D    0034   144          BVS     20$                     ;IF VS NO ENTRY REMOVED
                    53 10 A5    7D    0036   145          MOVQ    FKB$L_FR3(R5),R3        ;RESTORE REGISTERS R3 AND R4
                                003A   146
                                003A   147 ;
                                003A   148 ; DISPATCH FORK PROCESS WITH:
                                003A   149 ;
                                003A   150 ;       R0 THRU R2 = SCRATCH REGISTERS.
                                003A   151 ;       R3 AND R4 = RESTORED FROM FORK BLOCK.
                                003A   152 ;       R5 = ADDRESS OF FORK BLOCK.
                                003A   153 ;
                                003A   154
                    51 0C A5    D0    003A   155          MOVL    FKB$L_FPC(R5),R1        ;SAVE DISPATCH ADDRESS
                          03    BB    003E   156          PUSHR   #^M<R0,R1>              ;SAVE IPL AND DISPATCH ADDRESS
                          61    16    0040   157          JSB     (R1)                    ;DISPATCH FORK
                          03    BA    0042   158          POPR    #^M<R0,R1>              ;RESTORE IPL, DISPATCH ADDRESS
                       52 12    DB    0044   159          MFPR    #PR$_IPL,R2             ;GET EXIT IPL
                       52 50    D1    0047   160          CMPL    R0,R2                   ;EXIT IPL MUST EQUAL ENTRY
                          DA    13    004A   161          BEQL    10$                     ;GO AGAIN
                                004C   162
                                004C   163          BUG_CHECK BADFORKIPL,FATAL      ;BAD FORK EXIT INTERRUPT PRIORITY LEVEL
                00000000'9F     16    004C                JSB     @#BUG$CHECK
2C 4C 50 49 4B 52 4F 46 44 41 42 00' 0052                .ASCIC  "BADFORKIPL,FATAL"
                   4C 41 54 41 46    005E
                                0052   164
                          3F    BA    0063   165 20$:     POPR    #^M<R0,R1,R2,R3,R4,R5>  ;RESTORE FORK PROCESS REGISTER SET
                                0065   166          REI                             ;
```

```
                        0066   166
                        0066   167          .END
```

```
BUG$CHECK           ********   X   02      UCB$L_CPID          0000005C    D       UCB$W_DIRSEQ        00000088    D
EXE$FORK            00000005 RG  D  02      UCB$L_CRB           00000020    D       UCB$W_DSTADDR       00000018    D
EXE$FORKDSPTH       00000024 RG  D  02      UCB$L_DDB           00000024    D       UCB$W_DX_BCR        000000A4    D
EXE$IOFORK          00000000 RG  D  02      UCB$L_DEVCHAR       00000034    D       UCB$W_EC1           00000090    D
FKB$B_FIPL          0000000B     D          UCB$L_DEVDEPEND     0000003C    D       UCB$W_EC2           00000092    D
FKB$B_TYPE          0000000A     D          UCB$L_DPC           00000080    D       UCB$W_ERRCNT        00000072    D
FKB$C_LENGTH        00000018     D          UCB$L_DUETIM        0000005C    D       UCB$W_FUNC          0000007E    D
FKB$K_LENGTH        00000018     D          UCB$L_DX_BFPNT      0000009C    D       UCB$W_MB_SEED       00000000    D
FKB$L_FPC           0000000C     D          UCB$L_DX_BUF        00000098    D       UCB$W_MSGCNT        00000016    D
FKB$L_FQBL          00000004     D          UCB$L_DX_RXDB       000000A0    D       UCB$W_MSGMAX        00000014    D
FKB$L_FQFL          00000000     D          UCB$L_EMB           00000078    D       UCB$W_NT_CHAN       0000007C    D
FKB$L_FR3           00000010     D          UCB$L_FIRST         00000014    D       UCB$W_OFFSET        0000008A    D
FKB$L_FR4           00000014     D          UCB$L_FPC           0000000C    D       UCB$W_REFC          00000050    D
FKB$W_SIZE          00000008     D          UCB$L_FQBL          00000004    D       UCB$W_SIZE          00000008    D
PR$_IPL         =   00000012     D          UCB$L_FQFL          00000000    D       UCB$W_SRCADDR       0000001A    D
PR$_SIRR        =   00000014     D          UCB$L_FR3           00000010    D       UCB$W_STS           00000058    D
SIZ...          =   00000002     D          UCB$L_FR4           00000014    D       UCB$W_TT_DESIZE     000000A5    D
SWI$GL_FQFL         ********   X   02      UCB$L_IOQBL         00000044    D       UCB$W_UNIT          00000048    D
UCB$B_AMOD          00000053     D          UCB$L_IOQFL         00000040    D       UCB$W_VPROT         0000001A    D
UCB$B_CEX           00000077     D          UCB$L_IRP           0000004C    D
UCB$B_CM1           0000004A     D          UCB$L_LINK          0000002C    D
UCB$B_CM2           0000004B     D          UCB$L_LOGADR        00000064    D
UCB$B_DEVCLASS      00000038     D          UCB$L_MAXBLOCK      00000084    D
UCB$B_DEVTYPE       00000039     D          UCB$L_MB_MBX        0000007C    D
UCB$B_DIPL          00000052     D          UCB$L_MB_PORT       0000008C    D
UCB$B_DX_SCTCNT     000000A6     D          UCB$L_MB_RAST       00000078    D
UCB$B_ERTCNT        00000070     D          UCB$L_MB_SHB        00000080    D
UCB$B_ERTMAX        00000071     D          UCB$L_MB_WAST       00000074    D
UCB$B_FEX           00000076     D          UCB$L_MB_WIOQBL     00000088    D
UCB$B_FIPL          0000000B     D          UCB$L_MB_WIOQFL     00000084    D
UCB$B_LOCSRV        0000003C     D          UCB$L_MEDIA         0000008C    D
UCB$B_OFFNDX        00000094     D          UCB$L_NT_DATSSB     00000074    D
UCB$B_OFFRTC        00000095     D          UCB$L_NT_INTSSB     00000078    D
UCB$B_REMSRV        0000003D     D          UCB$L_OPCNT         00000060    D
UCB$B_SECTORS       0000003C     D          UCB$L_OWNUIC        0000001C    D
UCB$B_SLAVE         00000074     D          UCB$L_PID           00000028    D
UCB$B_SPR           00000075     D          UCB$L_RQBL          00000004    D
UCB$B_STATE         00000052     D          UCB$L_RQFL          00000000    D
UCB$B_TRACKS        0000003D     D          UCB$L_SVAPTE        00000068    D
UCB$B_TT_CRFILL     0000009D     D          UCB$L_SVPN          00000064    D
UCB$B_TT_DECRF      000000A1     D          UCB$L_TT_DECHAR     000000A8    D
UCB$B_TT_DELFF      000000A2     D          UCB$L_TT_RDUE       0000008C    D
UCB$B_TT_DESPEE     000000A0     D          UCB$L_TT_RTIMOU     000000B8    D
UCB$B_TT_DETYPE     000000A4     D          UCB$L_VCB           00000030    D
UCB$B_TT_LFFILL     0000009E     D          UCB$T_PARTNER       0000000C    D
UCB$B_TT_SPEED      0000009C     D          UCB$V_TIM       =   00000000    D
UCB$B_TYPE          0000000A     D          UCB$W_BCNT          0000006E    D
UCB$B_VERTSZ        0000003F     D          UCB$W_BCR           00000096    D
UCB$C_LENGTH        00000074     D          UCB$W_BOFF          0000006C    D
UCB$C_MB_LENGTH     00000090     D          UCB$W_BUFQUO        00000018    D
UCB$C_TT_LENGTH     000000BC     D          UCB$W_BYTESTOGO     0000003E    D
UCB$K_LENGTH        00000074     D          UCB$W_CHARGE        0000004A    D
UCB$K_MB_LENGTH     00000090     D          UCB$W_CYLINDERS     0000003E    D
UCB$K_TT_LENGTH     000000BC     D          UCB$W_DA            0000008C    D
UCB$L_AMB           00000054     D          UCB$W_DC            0000008E    D
UCB$L_ASTQBL        00000010     D          UCB$W_DEVBUFSIZ     0000003A    D
UCB$L_ASTQFL        0000000C     D          UCB$W_DEVSTS        0000005A    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000066 (102.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BUG$CHECK	00000000-XR		163 (3)
EXE$FORK	00000005-R	103 (2)	#-83 (1)
EXE$FORKDSPTH	00000024-R	139 (3)	
EXE$IOFORK	00000000-R	82 (1)	
FKB$B_FIPL	0000000B		#-106 (2)
FKB$L_FPC	0000000C		#-105 (2) #-155 (3)
FKB$L_FR3	00000010		#-104 (2) #-145 (3)
PR$_IPL	=00000012		#-141 (3) #-159 (3)
PR$_SIRR	=00000014		#-110 (2)
SWI$GL_FQFL	00000000-XR		107 (2) 142 (3)
UCB$V_TIM	=00000000		#-83 (1)
UCB$W_STS	00000058		83 (1)

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEFINI         1           53      (1)     53      (1)     54      (1)     55      (1)
$FKBDEF         1           53      (1)     53      (1)
$PRDEF          5           54      (1)     54      (1)
$UCBDEF         10          55      (1)     55      (1)
BUG_CHECK       1           163     (3)     163     (3)
SOFTINT         1           110     (2)     110     (2)
```

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
BBCC            00E5      83      (1)
BEQL            0013      161     (3)
BNEQ            0012      109     (2)
BVS             001D      144     (3)
CMPL            00D1      160     (3)
INSQUE          000E      108     (2)
JSB             0016      157     (3)     163     (3)
MFPR            00DB      141     (3)     159     (3)
MOVAQ           007E      107     (2)     142     (3)
MOVL            00D0      155     (3)
MOVQ            007D      104     (2)     145     (3)
MOVZBL          009A      106     (2)
MTPR            00DA      110     (2)
POPL            8ED0      105     (2)
POPR            00BA      158     (3)     164     (3)
PUSHR           00BB      140     (3)     156     (3)
REI             0002      165     (3)
REMQUE          000F      143     (3)
RSB             0005      111     (2)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ALIGN          138     (3)
.ASCIC          163     (3)
.END            167     (3)
.ENDM           110     (2)     163     (3)     53      (1)     54      (1)     55      (1)
.IDENT          5       (1)
.LIST           51      (1)     57      (1)     58      (1)     6       (1)
.NLIST          49      (1)     50      (1)     56      (1)     7       (1)
.NOCROSS        53      (1)     54      (1)     55      (1)
.PAGE           63      (1)
.PSECT          61      (1)
.RESTORE        53      (1)     54      (1)     55      (1)
.SAVE           53      (1)     54      (1)     55      (1)
.SBTTL          113     (3)     64      (1)     85      (2)
.TITLE          4       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	7	#-140	(3)	#-141	(3)	142	(3)	#-156	(3)	#-158	(3)	#-160	(3)
		#-164	(3)										
R1	8	#-140	(3)	#-142	(3)	143	(3)	#-155	(3)	#-156	(3)	157	(3)
		#-158	(3)	#-164	(3)								
R2	4	#-140	(3)	#-159	(3)	#-160	(3)	#-164	(3)				
R3	6	#-104	(2)	#-107	(2)	108	(2)	#-140	(3)	#-145	(3)	#-164	(3)
R4	5	#-106	(2)	107	(2)	#-110	(2)	#-140	(3)	#-164	(3)		
R5	10	#-104	(2)	#-105	(2)	#-106	(2)	108	(2)	#-140	(3)	#-143	(3)
		#-145	(3)	#-155	(3)	#-164	(3)	83	(1)				

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.02	00:00:00.20
Command processing	877	00:00:00.21	00:00:00.68
Pass 1	332	00:00:00.97	00:00:01.91
Symbol table sort	0	00:00:00.08	00:00:00.08
Pass 2	9	00:00:00.29	00:00:00.50
Symbol table output	0	00:00:00.02	00:00:00.06
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	2	00:00:00.05	00:00:00.12
Assembler run totals	1251	00:00:01.64	00:00:03.55

The working set limit was 2900 pages.
55458 bytes (109 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 302 non-local and 3 local symbols.
167 source lines were read in Pass 1, producing 29 object records in Pass 2.
27 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	2
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	1
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	10

503 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:FRKCTL.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W.DS/LIB+DS$R$W:
```

Table of contents

(2) 44 GET TIME

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element GETTIM.MAR
                        0000     2 ; *1     6-FEB-1986 15:18:41 DS ""
                        0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element GETTIM.MAR
```

```
                        0000     5          .TITLE  GETTIM - SYSTEM SERVICE GET TIME
                        0000     6          .IDENT  /01/
                        0000     7
                        0000     8 ;
                        0000     9 ; COPYRIGHT (C) 1976, 1980
                        0000    10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        0000    11 ;
                        0000    12 ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
                        0000    13 ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
                        0000    14 ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
                        0000    15 ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
                        0000    16 ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
                        0000    17 ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
                        0000    18 ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
                        0000    19 ;
                        0000    20 ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
                        0000    21 ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
                        0000    22 ; EQUIPMENT CORPORATION.
                        0000    23 ;
                        0000    24 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
                        0000    25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    26 ;
                        0000    27 ; D. N. CUTLER 30-SEP-76
                        0000    28 ;
                        0000    29 ; SYSTEM SERVICE GET TIME
                        0000    30 ;
                        0000    31 ; MACRO LIBRARY CALLS
                        0000    32 ;
                        0000    33
                        0000    34          $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .RESTORE
                        0000    35
                        0000    36 ;
                        0000    37 ; LOCAL SYMBOLS
                        0000    38 ;
                        0000    39 ; ARGUMENT LIST OFFSET DEFINITIONS
                        0000    40 ;
                        0000    41
00000004                0000    42 TIMADR=4                                ;ADDRESS OF QUADWORD TO RECEIVE TIME
```

```
                                0000     44          .SBTTL  GET TIME
                                0000     45 ;+
                                0000     46 ; EXE$GETTIM - GET TIME
                                0000     47 ;
                                0000     48 ; THIS SERVICE PROVIDES THE CAPABILITY TO RETRIEVE THE CURRENT SYSTEM TIME
                                0000     49 ; IN 64 BIT FORMAT.
                                0000     50 ;
                                0000     51 ; INPUTS:
                                0000     52 ;
                                0000     53 ;       TIMADR(AP) = ADDRESS OF QUADWORD THAT IS TO RECEIVE TIME.
                                0000     54 ;
                                0000     55 ; OUTPUTS:
                                0000     56 ;
                                0000     57 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO RETRIEVE SYSTEM TIME.
                                0000     58 ;
                                0000     59 ;              R0 = SS$_ACCVIO - QUADWORD TO RECEIVE TIME CANNOT BE
                                0000     60 ;                   WRITTEN BY CALLING ACCESS MODE.
                                0000     61 ;
                                0000     62 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0000     63 ;
                                0000     64 ;              R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0000     65 ;-
                                0000     66
                             00000000    67          .PSECT  SEP, SHR, WRT, EXE, LONG
                                0000     68 EXE$GETTIM::                             ;GET TIME
                           0000  0000     69          .WORD   0                       ;ENTRY MASK
                  51    04 AC    D0  0002     70          MOVL    TIMADR(AP),R1       ;GET ADDRESS OF QUADWORD TO RECEIVE TIME
                     50    0C    3C  0006     71          MOVZWL  #SS$_ACCVIO,R0      ;ASSUME QUADWORD NOT WRITABLE
                                     0009     72          IFNOWRT #8,(R1),10$         ;CAN QUADWORD BE WRITTEN?
                61    08    00    0D  0009                 PROBEW  #0,#8,(R1)
                                0A   13  000D                 BEQL    10$
                61  00000000'EF    7D  000F     73          MOVQ    EXE$GQ_SYSTIME,(R1) ;STORE SYSTEM TIME IN QUADWORD
                     50    01    3C  0016     74          MOVZWL  #SS$_NORMAL,R0      ;SET NORMAL COMPLETION
                                04   19  0019     75 10$:     RET                     ;
                                     001A     76
                                     001A     77          .END
```

```
EXE$GETTIM                      00000000 RG  D  02
EXE$GQ_SYSTIME                  ********   X   02
SS$_ACCVIO                    = 0000000C       D
SS$_NORMAL                    = 00000001       D
TIMADR                        = 00000004       D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	0000001A (26.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
EXE$GETTIM	00000000-R	68 (2)	
EXE$GQ_SYSTIME	00000000-XR		#-73 (2)
SS$_ACCVIO	=0000000C		#-71 (2)
SS$_NORMAL	=00000001		#-74 (2)
TIMADR	=00000004	42 (2)	#-70 (2)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEFINI	1	34 (2)	34 (2)
$SSDEF	21	34 (2)	34 (2)
IFNOWRT	1	72 (2)	72 (2)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...			
BEQL	0013	72	(2)		
MOVL	00D0	70	(2)		
MOVQ	007D	73	(2)		
MOVZWL	003C	71	(2)	74	(2)
PROBEW	000D	72	(2)		
RET	0004	75	(2)		

Directives Cross Reference

DIRECTIVE	REFERENCES...			
.END	77	(2)		
.ENDM	34	(2)	72	(2)
.IDENT	6	(2)		
.NOCROSS	34	(2)		
.PAGE	43	(2)		
.PSECT	67	(2)		
.RESTORE	34	(2)		
.SAVE	34	(2)		
.SBTTL	44	(2)		
.TITLE	5	(2)		
.WORD	69	(2)		

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...					
AP	1	#-70	(2)				
R0	2	#-71	(2)	#-74	(2)		
R1	3	#-70	(2)	72	(2)	#-73	(2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.01	00:00:00.19
Command processing	880	00:00:00.22	00:00:00.66
Pass 1	284	00:00:00.73	00:00:01.51
Symbol table sort	0	00:00:00.12	00:00:00.12
Pass 2	15	00:00:00.20	00:00:00.33
Symbol table output	0	00:00:00.00	00:00:00.00
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	2	00:00:00.04	00:00:00.09
Assembler run totals	1206	00:00:01.32	00:00:02.90

The working set limit was 2400 pages.
39972 bytes (79 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 413 non-local and 1 local symbols.
77 source lines were read in Pass 1, producing 35 object records in Pass 2.
10 pages of virtual memory were used to define 8 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	1
SYS$COMMON:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	5

473 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:GETTIM.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 56 GET I/O CHANNEL DEVICE INFORMATION

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element GTCHAN.MAR
                        0000     2 ; *1     6-FEB-1986 15:18:43 DS ""
                        0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element GTCHAN.MAR
                        0000     4         .TITLE  GTCHAN  *** GTCHAN Get channel information
                        0000     5         .IDENT  /02-01/
                        0000     6
                        0000     7 ;
                        0000     8 ; COPYRIGHT (C) 1977,, 1980
                        0000     9 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        0000    10 ;
                        0000    11 ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
                        0000    12 ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
                        0000    13 ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
                        0000    14 ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
                        0000    15 ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
                        0000    16 ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
                        0000    17 ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
                        0000    18 ;
                        0000    19 ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
                        0000    20 ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
                        0000    21 ; EQUIPMENT CORPORATION.
                        0000    22 ;
                        0000    23 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
                        0000    24 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    25 ;
                        0000    26 ; D. N. CUTLER 22-SEP-76
                        0000    27 ;
                        0000    28 ; MODIFICATION HISTORY
                        0000    29 ;               N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
                        0000    30 ;       01      DIAGNOSTIC SUPERVISOR INTEGRATION
                        0000    31 ;
                        0000    32 ; SYSTEM SERVICE GET I/O CHANNEL DEVICE INFORMATION
                        0000    33
                        0000    34 ;
                        0000    35 ; MACRO LIBRARY CALLS
                        0000    36 ;
                        0000    37
                        0000    38         $CCBDEF                             ;DEFINE CCB OFFSETS
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
00000004                0000                       .BLKL   1
                        0004               .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
00000008                0004                       .BLKL   1
                        0008               .IIF    NB,CCB$B_STS,   CCB$B_STS:
00000009                0008                       .BLKB   1
                        0009               .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
0000000A                0009                       .BLKB   1
                        000A               .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
0000000C                000A                       .BLKW   1
                        000C               .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
00000010                000C                       .BLKL   1
                        0010               .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                        0010               .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                        0000               .RESTORE
                        0000    39         $DDBDEF                             ;DEFINE DDB OFFSETS
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
```

```
                    00000000  0010              .=0
                              0000              .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
                    00000004  0000                      .BLKL   1
                              0004              .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
                    00000008  0004                      .BLKL   1
                              0008              .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
                    0000000A  0008                      .BLKW   1
                              000A              .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
                    0000000B  000A                      .BLKB   1
                    0000000C  000B                      .BLKB   1
                              000C              .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
                    00000010  000C                      .BLKL   1
                              0010              .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
                    00000013  0010                      .BLKB   3
                              0013              .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
                    00000014  0013                      .BLKB   1
                              0014              .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
                    00000015  0014                      .BLKB   1
                    00000024  0015                      .BLKB   15
                              0024              .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
                    00000025  0024                      .BLKB   1
                    00000034  0025                      .BLKB   15
                              0034              .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                              0034              .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                              0000              .RESTORE
                              0000      40      $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  0034              .=0
                              0000              .RESTORE
                              0000      41      $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  0034              .=0
                              0000              .RESTORE
                              0000      42      $UCBDEF                         ;DEFINE UCB OFFSETS
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  0034              .=0
                              0000              .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                              0000              .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                    00000004  0000                      .BLKL   1
                              0004              .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                              0004              .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                    00000008  0004                      .BLKL   1
                              0008              .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                    0000000A  0008                      .BLKW   1
                              000A              .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                    0000000B  000A                      .BLKB   1
                              000B              .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                    0000000C  000B                      .BLKB   1
                              000C              .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                              000C              .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                              000C              .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                    0000000D  000C                      .BLKB   1
                    00000010  000D                      .BLKB   3
                              0010              .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
```

```
                    0010            .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014            0010                    .BLKL   1
                    0014            .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014            .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014            .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016            0014                    .BLKW   1
                    0016            .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018            0016                    .BLKW   1
                    0018            .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018            .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
0000001A            0018                    .BLKW   1
                    001A            .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A            .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
0000001C            001A                    .BLKW   1
                    001C            .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
00000020            001C                    .BLKL   1
                    0020            .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
00000024            0020                    .BLKL   1
                    0024            .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
00000028            0024                    .BLKL   1
                    0028            .IIF    NB,UCB$L_PID,   UCB$L_PID:
0000002C            0028                    .BLKL   1
                    002C            .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030            002C                    .BLKL   1
                    0030            .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034            0030                    .BLKL   1
                    0034            .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038            0034                    .BLKL   1
                    0038            .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039            0038                    .BLKB   1
                    0039            .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A            0039                    .BLKB   1
                    003A            .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C            003A                    .BLKW   1
                    003C            .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C            .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C            .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D            003C                    .BLKB   1
                    003D            .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                    003D            .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E            003D                    .BLKB   1
                    003E            .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                    003E            .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F            003E                    .BLKB   1
                    003F            .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040            003F                    .BLKB   1
                    0040            .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044            0040                    .BLKL   1
                    0044            .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048            0044                    .BLKL   1
                    0048            .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A            0048                    .BLKW   1
                    004A            .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                    004A            .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B            004A                    .BLKB   1
                    004B            .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C            004B                    .BLKB   1
```

```
                        004C            .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                 .BLKL   1
                        0050            .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                 .BLKW   1
                        0052            .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                        0052            .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                 .BLKB   1
                        0053            .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                 .BLKB   1
                        0054            .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                 .BLKL   1
                        0058            .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                 .BLKW   1
                        005A            .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                 .BLKW   1
                        005C            .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                        005C            .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                 .BLKL   1
                        0060            .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                                 .BLKL   1
                        0064            .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                        0064            .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                                 .BLKL   1
                        0068            .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                                 .BLKL   1
                        006C            .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                                 .BLKW   1
                        006E            .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                                 .BLKW   1
                        0070            .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                                 .BLKB   1
                        0071            .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                                 .BLKB   1
                        0072            .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                                 .BLKW   1
                        0074            .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                        0074            .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074          .  = 0
                        0000            .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                                 .BLKW   1
00000074  0002          .  = 116
                        0074            .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                                 .BLKL   1
                        0078            .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                                 .BLKL   1
                        007C            .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                                 .BLKL   1
                        0080            .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                                 .BLKL   1
                        0084            .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                                 .BLKL   1
                        0088            .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                                 .BLKL   1
                        008C            .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                                 .BLKL   1
                        0090            .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                        0090            .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
```

```
00000074  0090          . = 116
          0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                          .BLKB   1
          0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                          .BLKB   1
          0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                          .BLKB   1
          0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                          .BLKB   1
          0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                          .BLKL   1
0000007E  007C                          .BLKW   1
          007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                          .BLKW   1
          0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                          .BLKL   1
00000080  0084          . = 128
00000084  0080                          .BLKL   1
          0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                          .BLKW   1
          008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                          .BLKW   1
          008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                          .BLKW   1
          008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                          .BLKW   1
          0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                          .BLKW   1
          0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                          .BLKW   1
          0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                          .BLKB   1
          0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                          .BLKB   1
          0096          .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                          .BLKW   1
00000096  0098          . = 150
00000098  0096                          .BLKW   1
          0098          .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                          .BLKL   1
          009C          .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                          .BLKL   1
          00A0          .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                          .BLKL   1
          00A4          .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                          .BLKW   1
          00A6          .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                          .BLKB   1
000000A8  00A7                          .BLKB   1
00000074  00A8          . = 116
00000078  0074                          .BLKL   1
0000007C  0078                          .BLKL   1
00000080  007C                          .BLKL   1
00000084  0080                          .BLKL   1
```

```
00000088  0084                                .BLKL   1
0000008C  0088                                .BLKL   1
          008C              .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                .BLKL   1
00000094  0090                                .BLKL   1
00000098  0094                                .BLKL   1
0000009C  0098                                .BLKL   1
          009C              .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                .BLKB   1
          009D              .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                .BLKB   1
          009E              .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                .BLKB   1
000000A0  009F                                .BLKB   1
          00A0              .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                .BLKB   1
          00A1              .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                .BLKB   1
          00A2              .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                .BLKB   1
000000A4  00A3                                .BLKB   1
          00A4              .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                .BLKB   1
          00A5              .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                .BLKW   1
000000A8  00A7                                .BLKB   1
          00A8              .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                .BLKL   1
000000B0  00AC                                .BLKL   1
000000B4  00B0                                .BLKL   1
000000B8  00B4                                .BLKL   1
          00B8              .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                .BLKL   1
          00BC              .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC              .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC              . = 116
          0074              .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                                .BLKL   1
          0078              .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                .BLKL   1
          007C              .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                .BLKW   1
00000080  007E                                .BLKW   1
          0000              .RESTORE
          0000     43
          0000     44 ;
          0000     45 ; LOCAL SYMBOLS
          0000     46 ;
          0000     47 ; ARGUMENT LIST OFFSET DEFINITIONS
          0000     48 ;
          0000     49
00000004  0000     50 CHAN=4                                  ;I/O CHANNEL NUMBER
00000008  0000     51 PRILEN=8                                ;ADDRESS TO STORE LENGTH OF PRIMARY STRING
0000000C  0000     52 PRIBUF=12                               ;ADDRESS OF PRIMARY BUFFER DESCRIPTOR
00000010  0000     53 SCDLEN=16                               ;ADDRESS TO STORE LENGTH OF SECONDARY STRING
00000014  0000     54 SCDBUF=20                               ;ADDRESS OF SECONDARY BUFFER DESCRIPTOR
```

```
                                        .SBTTL  GET I/O CHANNEL DEVICE INFORMATION
                                ;+
                                ; EXE$GTCHAN - GET I/O CHANNEL DEVICE INFORMATION
                                ;
                                ; THIS SERVICE PROVIDES THE CAPABILITY TO RETRIEVE INFORMATION ABOUT A
                                ; DEVICE THAT IS ASSIGNED TO A CHANNEL AND ITS ASSOCIATED DEVICE IF ANY.
                                ;
                                ; INPUTS:
                                ;
                                ;       CHAN(AP) = I/O CHANNEL NUMBER.
                                ;       PRILEN(AP) = ADDRESS TO STORE LENGTH OF PRIMARY STRING.
                                ;       PRIBUF(AP) = ADDRESS OF PRIMARY BUFFER DESCRIPTOR.
                                ;       SCDLEN(AP) = ADDRESS TO STORE LENGTH OF SECONDARY STRING.
                                ;       SCDBUF(AP) = ADDRESS OF SECONDARY BUFFER DESCRIPTOR.
                                ;
                                ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                ;
                                ; OUTPUTS:
                                ;
                                ;       R0 LOW BIT CLEAR INDICATES FAILURE TO RETRIEVE DEVICE INFORMATION.
                                ;
                                ;               R0 = SS$_ACCVIO - PRIMARY OR SECONDARY BUFFER DESCRIPTOR
                                ;                    CANNOT BE READ BY CALLING ACCESS MODE, OR PRIMARY
                                ;                    OR SECONDARY BUFFER CANNOT BE WRITTEN BY CALLING
                                ;                    ACCESS MODE.
                                ;
                                ;               R0 = SS$_IVCHAN - INVALID CHANNEL NUMBER SPECIFIED.
                                ;
                                ;               R0 = SS$_NOPRIV - SPECIFIED CHANNEL IS NOT ASSIGNED TO A
                                ;                    DEVICE OR THE CALLING ACCESS MODE DOES NOT HAVE
                                ;                    PRIVILEGE TO ACCESS THE CHANNEL.
                                ;
                                ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                ;
                                ;               R0 = SS$_BUFFEROVF - NORMAL COMPLETION, ALL CHARACTERISTIC
                                ;                    INFORMATION DID NOT FIT IN SPECIFIED BUFFER(S).
                                ;
                                ;               R0 = SS$_NORMAL - NORMAL COMPLETION, ALL CHARACTERISTIC
                                ;                    INFORMATION TRANSFERED.
                                ;-

                  00000000                      .PSECT  SEP,    SHR,    EXE,    WRT,    LONG

                  01FC  0000    .ENTRY  EXE$GTCHAN,^M<R2,R3,R4,R5,R6,R7,R8>
                        0002

50    04 AC     3C    0002            MOVZWL  CHAN(AP),R0             ;GET CHANNEL NUMBER
       FFF7'    30    0006            BSBW    IOC$VERIFYCHAN          ;VERIFY CHANNEL NUMBER
     2F 50      E9    0009            BLBC    R0,40$                  ;IF LBC INVALID CHANNEL
  55    61      D0    000C            MOVL    CCB$L_UCB(R1),R5        ;GET DEVICE UNIT UCB ADDRESS
  54    55      D0    000F            MOVL    R5,R4                   ;ASSUME ASSOCIATED UCB NOT SPOOL DEVICE
04 34 A5    06    E1    0012            BBC     #DEV$V_SPL,UCB$L_DEVCHAR(R5),10$ ;IF CLR, THEN DEVICE NOT SPOOLED
  54    54 A5    D0    0017            MOVL    UCB$L_AMB(R5),R4        ;GET INTERMEDIATE DEVICE UCB ADDRESS
  56    01       3C    001B    10$:    MOVZWL  #SS$_NORMAL,R6          ;ASSUME ALL INFO CAN BE RETURNED
  57    08 AC    7D    001E            MOVQ    PRILEN(AP),R7           ;GET PRIMARY BUFFER PARAMETERS
        18       10    0022            BSBB    FILBUF                  ;FILL PRIMARY BUFFER
  54    54 A5    D0    0024            MOVL    UCB$L_AMB(R5),R4        ;ANY ASSOCIATED UCB?
        05       13    0028            BEQL    20$                     ;IF EQL NO
```

```
                03 34 A5    06   E1  002A   113          BBC     #DEV$V_SPL,UCB$L_DEVCHAR(R5),30$ ;IF CLR, THEN DEVICE NOT SPOOLED
                      54    55   D0  002F   114 20$:      MOVL    R5,R4                            ;SET SECONDARY DEVICE UCB ADDRESS
                   57   10 AC   7D  0032   115 30$:      MOVQ    SCDLEN(AP),R7                    ;GET SECONDARY BUFFER PARAMETERS
                         04    10  0036   116            BSBB    FILBUF                           ;FILL SECONDARY BUFFER
                      50   56   D0  0038   117            MOVL    R6,R0                            ;SET FINAL REQUEST STATUS
                         04    04  003B   118 40$:      RET                                      ;
                                   003C   119
                                   003C   120 ;
                                   003C   121 ; SUBROUTINE TO FILL CHARACTERISTIC BUFFER
                                   003C   122 ;
                                   003C   123
                         58    D5  003C   124 FILBUF: TSTL    R8                               ;ANY BUFFER SPECIFIED?
                         43    13  003E   125            BEQL    50$                              ;IF EQL NO
                                   0040   126 ;          IFNORD  #8,(R8),60$                      ;CAN OUTPUT BUFFER DESCRIPTOR BE READ?
                      52   68    3C  0040   127            MOVZWL  (R8),R2                          ;GET SIZE OF OUTPUT BUFFER
                         31    13  0043   128            BEQL    30$                              ;IF EQL NULL
                   53   04 A8    D0  0045   129            MOVL    4(R8),R3                         ;GET ADDRESS OF OUTPUT BUFFER
                                   0049   130 ;          IFNOWRT R2,(R3),60$                      ;CAN ENTIRE BUFFER BE WRITTEN?
                   51   34 A4    DE  0049   131            MOVAL   UCB$L_DEVCHAR(R4),R1             ;GET ADDRESS OF CHARACTERISTICS
                      50   0C    D0  004D   132            MOVL    #12,R0                           ;SET BYTE COUNT
                         16    10  0050   133            BSBB    10$                              ;MOVE CHARACTERISTICS TO BUFFER
                   51   48 A4    DE  0052   134            MOVAL   UCB$W_UNIT(R4),R1                ;GET ADDRESS OF UNIT NUMBER
                      50   02    D0  0056   135            MOVL    #2,R0                            ;SET BYTE COUNT
                         0D    10  0059   136            BSBB    10$                              ;MOVE UNIT NUMBER OT BUFFER
                   51   24 A4    D0  005B   137            MOVL    UCB$L_DDB(R4),R1                 ;GET ADDRESS OF DDB
                   51   14 A1    DE  005F   138            MOVAL   DDB$T_NAME(R1),R1                ;GET ADDRESS OF GENERIC DEVICE NAME
                      50   61    9A  0063   139            MOVZBL  (R1),R0                          ;GET LENGTH OF NAME IN BYTES
                         50    D6  0066   140            INCL    R0                               ;ACCOUNT FOR COUNT BYTE
                         52    D7  0068   141 10$:      DECL    R2                               ;ANY SPACE LEFT IN BUFFER?
                         08    19  006A   142            BLSS    20$                              ;IF LSS NO
                   83    81    90  006C   143            MOVB    (R1)+,(R3)+                      ;MOVE BYTE TO BUFFER
                     F6  50    F5  006F   144            SOBGTR  R0,10$                           ;ANY MORE BYTES TO MOVE?
                         07    11  0072   145            BRB     40$                              ;
                         52    D6  0074   146 20$:      INCL    R2                               ;ADJUST COUNT OF REMAINING BYTES
             50    0601 8F    3C  0076   147 30$:      MOVZWL  #SS$_BUFFEROVF,R0                ;SET BUFFER OVERFLOW
                         57    D5  007B   148 40$:      TSTL    R7                               ;ADDRESS SPECIFIED TO STORE RESULT LENGTH?
                         04    13  007D   149            BEQL    50$                              ;IF EQL NO
                                   007F   150 ;          IFNOWRT #2,(R7),60$                      ;CAN LENGTH BE WRITTEN?
             67    68    52    A3  007F   151            SUBW3   R2,(R8),(R7)                     ;CALCULATE LENGTH OF RESULT STRING
                         05    05  0083   152 50$:      RSB                                      ;
                      50   0C    3C  0084   153 60$:      MOVZWL  #SS$_ACCVIO,R0                   ;SET ACCESS VIOLATION
                         04    04  0087   154            RET                                      ;
                                   0088   155
                                   0088   156            .END
```

```
CCB$B_AMOD                      00000009    D
CCB$B_STS                       00000008    D
CCB$C_LENGTH                    00000010    D
CCB$K_LENGTH                    00000010    D
CCB$L_DIRP                      0000000C    D
CCB$L_UCB                       00000000    D
CCB$L_WIND                      00000004    D
CCB$W_IOC                       0000000A    D
CHAN                          = 00000004    D
DDB$B_ACPCLASS                  00000013    D
DDB$B_TYPE                      0000000A    D
DDB$C_LENGTH                    00000034    D
DDB$K_LENGTH                    00000034    D
DDB$L_ACPD                      00000010    D
DDB$L_DDT                       0000000C    D
DDB$L_LINK                      00000000    D
DDB$L_UCB                       00000004    D
DDB$T_DRVNAME                   00000024    D
DDB$T_NAME                      00000014    D
DDB$W_SIZE                      00000008    D
DEV$V_SPL                     = 00000006    D
EXE$GTCHAN                      00000000 RG D   02
FILBUF                          0000003C R  D   02
IOC$VERIFYCHAN                  ********    X   02
PRIBUF                        = 0000000C    D
PRILEN                        = 00000008    D
SCDBUF                        = 00000014    D
SCDLEN                        = 00000010    D
SIZ...                        = 00000002    D
SS$_ACCVIO                    = 0000000C    D
SS$_BUFFEROVF                 = 00000601    D
SS$_NORMAL                    = 00000001    D
UCB$B_AMOD                      00000053    D
UCB$B_CEX                       00000077    D
UCB$B_CM1                       0000004A    D
UCB$B_CM2                       0000004B    D
UCB$B_DEVCLASS                  00000038    D
UCB$B_DEVTYPE                   00000039    D
UCB$B_DIPL                      00000052    D
UCB$B_DX_SCTCNT                 000000A6    D
UCB$B_ERTCNT                    00000070    D
UCB$B_ERTMAX                    00000071    D
UCB$B_FEX                       00000076    D
UCB$B_FIPL                      0000000B    D
UCB$B_LOCSRV                    0000003C    D
UCB$B_OFFNDX                    00000094    D
UCB$B_OFFRTC                    00000095    D
UCB$B_REMSRV                    0000003D    D
UCB$B_SECTORS                   0000003C    D
UCB$B_SLAVE                     00000074    D
UCB$B_SPR                       00000075    D
UCB$B_STATE                     00000052    D
UCB$B_TRACKS                    0000003D    D
UCB$B_TT_CRFILL                 0000009D    D
UCB$B_TT_DECRF                  000000A1    D
UCB$B_TT_DELFF                  000000A2    D
UCB$B_TT_DESPEE                 000000A0    D
UCB$B_TT_DETYPE                 000000A4    D
UCB$B_TT_LFFILL                 0000009E    D
UCB$B_TT_SPEED                  0000009C    D
UCB$B_TYPE                      0000000A    D
UCB$B_VERTSZ                    0000003F    D
UCB$C_LENGTH                    00000074    D
UCB$C_MB_LENGTH                 00000090    D
UCB$C_TT_LENGTH                 000000BC    D
UCB$K_LENGTH                    00000074    D
UCB$K_MB_LENGTH                 00000090    D
UCB$K_TT_LENGTH                 000000BC    D
UCB$L_AMB                       00000054    D
UCB$L_ASTQBL                    00000010    D
UCB$L_ASTQFL                    0000000C    D
UCB$L_CPID                      0000005C    D
UCB$L_CRB                       00000020    D
UCB$L_DDB                       00000024    D
UCB$L_DEVCHAR                   00000034    D
UCB$L_DEVDEPEND                 0000003C    D
UCB$L_DPC                       00000080    D
UCB$L_DUETIM                    0000005C    D
UCB$L_DX_BFPNT                  0000009C    D
UCB$L_DX_BUF                    00000098    D
UCB$L_DX_RXDB                   000000A0    D
UCB$L_EMB                       00000078    D
UCB$L_FIRST                     00000014    D
UCB$L_FPC                       0000000C    D
UCB$L_FQBL                      00000004    D
UCB$L_FQFL                      00000000    D
UCB$L_FR3                       00000010    D
UCB$L_FR4                       00000014    D
UCB$L_IOQBL                     00000044    D
UCB$L_IOQFL                     00000040    D
UCB$L_IRP                       0000004C    D
UCB$L_LINK                      0000002C    D
UCB$L_LOGADR                    00000064    D
UCB$L_MAXBLOCK                  00000084    D
UCB$L_MB_MBX                    0000007C    D
UCB$L_MB_PORT                   0000008C    D
UCB$L_MB_RAST                   00000078    D
UCB$L_MB_SHB                    00000080    D
UCB$L_MB_WAST                   00000074    D
UCB$L_MB_WIOQBL                 00000088    D
UCB$L_MB_WIOQFL                 00000084    D
UCB$L_MEDIA                     0000008C    D
UCB$L_NT_DATSSB                 00000074    D
UCB$L_NT_INTSSB                 00000078    D
UCB$L_OPCNT                     00000060    D
UCB$L_OWNUIC                    0000001C    D
UCB$L_PID                       00000028    D
UCB$L_RQBL                      00000004    D
UCB$L_RQFL                      00000000    D
UCB$L_SVAPTE                    00000068    D
UCB$L_SVPN                      00000064    D
UCB$L_TT_DECHAR                 000000A8    D
UCB$L_TT_RDUE                   0000008C    D
UCB$L_TT_RTIMOU                 000000B8    D
```

```
UCB$L_VCB                       00000030    D
UCB$T_PARTNER                   0000000C    D
UCB$W_BCNT                      0000006E    D
UCB$W_BCR                       00000096    D
UCB$W_BOFF                      0000006C    D
UCB$W_BUFQUO                    00000018    D
UCB$W_BYTESTOGO                 0000003E    D
UCB$W_CHARGE                    0000004A    D
UCB$W_CYLINDERS                 0000003E    D
UCB$W_DA                        0000008C    D
UCB$W_DC                        0000008E    D
UCB$W_DEVBUFSIZ                 0000003A    D
UCB$W_DEVSTS                    0000005A    D
UCB$W_DIRSEQ                    00000088    D
UCB$W_DSTADDR                   00000018    D
UCB$W_DX_BCR                    000000A4    D
UCB$W_EC1                       00000090    D
UCB$W_EC2                       00000092    D
UCB$W_ERRCNT                    00000072    D
UCB$W_FUNC                      0000007E    D
UCB$W_MB_SEED                   00000000    D
UCB$W_MSGCNT                    00000016    D
UCB$W_MSGMAX                    00000014    D
UCB$W_NT_CHAN                   0000007C    D
UCB$W_OFFSET                    0000008A    D
UCB$W_REFC                      00000050    D
UCB$W_SIZE                      00000008    D
UCB$W_SRCADDR                   0000001A    D
UCB$W_STS                       00000058    D
UCB$W_TT_DESIZE                 000000A5    D
UCB$W_UNIT                      00000048    D
UCB$W_VPROT                     0000001A    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000088 (136.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
CCB$L_UCB	00000000		#-104 (1)
CHAN	=00000004	50 (1)	#-101 (1)
DDB$T_NAME	00000014		138 (1)
DEV$V_SPL	=00000006		#-106 (1) #-113 (1)
EXE$GTCHAN	00000000-R	99 (1)	
FILBUF	0000003C-R	124 (1)	#-110 (1) #-116 (1)
IOC$VERIFYCHAN	00000000-XR		#-102 (1)
PRIBUF	=0000000C	52 (1)	
PRILEN	=00000008	51 (1)	#-109 (1)
SCDBUF	=00000014	54 (1)	
SCDLEN	=00000010	53 (1)	#-115 (1)
SS$_ACCVIO	=0000000C		#-153 (1)
SS$_BUFFEROVF	=00000601		#-147 (1)
SS$_NORMAL	=00000001		#-108 (1)
UCB$L_AMB	00000054		#-107 (1) #-111 (1)
UCB$L_DDB	00000024		#-137 (1)
UCB$L_DEVCHAR	00000034		106 (1) 113 (1) 131 (1)
UCB$W_UNIT	00000048		134 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CCBDEF	1	38 (1)	38 (1)
$DDBDEF	1	39 (1)	39 (1)
$DEFINI	1	38 (1)	38 (1) 39 (1) 40 (1) 41 (1) 42 (1)
$DEVDEF	1	40 (1)	40 (1)
$SSDEF	21	41 (1)	41 (1)
$UCBDEF	10	42 (1)	42 (1)

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
BBC             00E1      106     (1)     113     (1)
BEQL            0013      112     (1)     125     (1)     128     (1)     149     (1)
BLBC            00E9      103     (1)
BLSS            0019      142     (1)
BRB             0011      145     (1)
BSBB            0010      110     (1)     116     (1)     133     (1)     136     (1)
BSBW            0030      102     (1)
DECL            00D7      141     (1)
INCL            00D6      140     (1)     146     (1)
MOVAL           00DE      131     (1)     134     (1)     138     (1)
MOVB            0090      143     (1)
MOVL            00D0      104     (1)     105     (1)     107     (1)     111     (1)     114     (1)     117     (1)     129     (1)
                          132     (1)     135     (1)     137     (1)
MOVQ            007D      109     (1)     115     (1)
MOVZBL          009A      139     (1)
MOVZWL          003C      101     (1)     108     (1)     127     (1)     147     (1)     153     (1)
RET             0004      118     (1)     154     (1)
RSB             0005      152     (1)
SOBGTR          00F5      144     (1)
SUBW3           00A3      151     (1)
TSTL            00D5      124     (1)     148     (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...									
.END	156	(1)								
.ENDM	38	(1)	39	(1)	40	(1)	41	(1)	42	(1)
.ENTRY	99	(1)								
.IDENT	5	(1)								
.NOCROSS	38	(1)	39	(1)	40	(1)	41	(1)	42	(1)
.PAGE	55	(1)								
.PSECT	97	(1)								
.RESTORE	38	(1)	39	(1)	40	(1)	41	(1)	42	(1)
.SAVE	38	(1)	39	(1)	40	(1)	41	(1)	42	(1)
.SBTTL	56	(1)								
.TITLE	4	(1)								

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	3	#-101 (1) #-109 (1) #-115 (1)
R0	10	#-101 (1) #-103 (1) #-117 (1) #-132 (1) #-135 (1) #-139 (1)
		#-140 (1) #-144 (1) #-147 (1) #-153 (1)
R1	8	#-104 (1) #-131 (1) #-134 (1) #-137 (1) 138 (1) #-139 (1)
		#-143 (1)
R2	5	#-127 (1) #-141 (1) #-146 (1) #-151 (1) 99 (1)
R3	3	#-129 (1) #-143 (1) 99 (1)
R4	8	#-105 (1) #-107 (1) #-111 (1) #-114 (1) 131 (1) 134 (1)
		#-137 (1) 99 (1)
R5	8	#-104 (1) #-105 (1) 106 (1) #-107 (1) #-111 (1) 113 (1)
		#-114 (1) 99 (1)
R6	3	#-108 (1) #-117 (1) 99 (1)
R7	5	#-109 (1) #-115 (1) #-148 (1) #-151 (1) 99 (1)
R8	5	#-124 (1) #-127 (1) #-129 (1) #-151 (1) 99 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.19
Command processing	891	00:00:00.26	00:00:00.65
Pass 1	565	00:00:01.69	00:00:02.58
Symbol table sort	0	00:00:00.18	00:00:00.18
Pass 2	40	00:00:00.42	00:00:00.72
Symbol table output	2	00:00:00.03	00:00:00.15
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	4	00:00:00.05	00:00:00.09
Assembler run totals	1528	00:00:02.65	00:00:04.56

The working set limit was 2400 pages.
94152 bytes (184 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 671 non-local and 10 local symbols.
156 source lines were read in Pass 1, producing 51 object records in Pass 2.
35 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	10

868 GETS were required to define 10 macros.

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 10 Frame J10        Sequence 1980
GTCHAN                      *** GTCHAN Get channel information    11-NOV-1987 17:36:09  VAX/VMS Macro V04-00        Page  16
VAX-11 Macro Run Statistics                                       3-JAN-1985 16:51:54  GTCHAN.MAR;1                      (1)

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:GTCHAN.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element HELP.B32
;    0002  0      ! *2    17-SEP-1987 09:19:30 MEIER "ADDED L AND J AS VALID LETTERS FOR HELP FILE NAMES"
;    0003  0      ! *1     6-FEB-1986 15:18:45 DS ""
;    0004  0      ! DEC/CMS REPLACEMENT HISTORY, Element HELP.B32
;    0005  0      %TITLE '*** HELP Handle HELP command'
;    0006  0      MODULE help (                                   ! Supervisor HELP facility
;    0007  0                   IDENT = '10.10-2'                                                  !G:2
;    0008  0                   ) =
;    0009  0      !
;    0010  0      ! Copyright (c) 1980, 1982, 1983, 1984, 1985, 1986, 1987                       !G:2
;    0011  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0012  0      !
;    0013  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0014  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0015  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0016  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0017  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0018  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0019  0      ! REMAIN IN DEC.
;    0020  0      !
;    0021  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0022  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0023  0      ! CORPORATION.
;    0024  0      !
;    0025  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0026  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
```

```
0
0       !++
0       ! FACILITY: DIAGNOSTIC SUPERVISOR
0       !
0       ! ABSTRACT:
0       !
0       !       This module implements a Diagnostic Supervisor HELP command in a
0       !       manner more-or-less compatible with VMS.
0       !
0       ! AUTHOR:        Dave Butenhof   22-jun-1980
0       ! MODIFIED BY:
0       !
0       !               Dave Butenhof 8-sep-1980, version 6.0
0       !       1       Convert to use RMS calls for file read.
0       !
0       !               Dave Butenhof 2-oct-1980, version 6.1
0       !       2       Rename entry point to DSX$HELP, since DSV prefix routines
0       !               should be JSB entry, whereas DSX is CALLable.
0       !               Dave Butenhof, 8-apr-1981, version 6.3
0       !       3       On printing other help for not-found key, be sure to skip
0       !               the record SAVERFA is pointing to (lest is cause infinite
0       !               loop)!
0       !              -Dave Butenhof 02-Jun-1981, version 6.4
0       !       04      Redefine file specs to make library access more flexible
0       !
0       !       05      - Dave Butenhof, 03-Feb-1982, Version 6.6
0       !               Make KEY_EQUAL be global (as alias DSR$KEY_EQUAL) so
0       !               Directory can use it.  Also make it CALL linkage to avoid
0       !               complications.
0       !
0       !       [06]    Dave Butenhof, 29-Mar-1982, version 6.6
0       !               Fix bug in READ handling of no-attribute.  Don't check
0       !               'last two' characters unless they're really there.
0       !       07      Dave butenhof, 20-Apr-1982
0       !               Change name HLP$V_HELP to HLP$V_HELP_KEY; LIB now openly defines
0       !               something called HLP$V_HELP.
0       !
0       !       08      Jack Stansbury, 16-Dec-1982, Version 6.11
0       !               Changed the field width for qualifiers and other key words to
0       !               be 16 rather than 8. This improves the Help output some.
0       !
0       !       09      M. Baggett,     18-Nov-1983     Version 6.13
0       !               The restriction on the second character has been removed.
0       !
0       !       10      Bob Bergazzi    May 31, 1984    Version 7.0
0       !               Edit 09 broke HELP EXAMINE, so fix it and also add support
0       !               for all new processors.
0       !
0       !       11      M. Baggett      3-Jun-1985      Version 8.3
0       !               Fixed compare keys routine so ULTRIX text is converted to upper case.
0       !               Change AURORA help file to have 'A'.
0       !                                                                                       !G:2
0       !       (Edit #'s are cms generation #'s now)                                           !G:2
0       !                                                                                       !G:2
0       !       2       Stephen Meier   16-Sept-87      Version 10.10                           !G:2
0       !               Added "L" to the list of valid second letters of a help file name.      !G:2
0       !--
```

```
;   0085  0
```

```
;    0087  1      BEGIN
;    0088  1      !+
;    0089  1      ! INCLUDE FILES:
;    0090  1      !-
;    0091  1
;    0092  1      LIBRARY '$diag';                    !                                   [04]
;    0093  1
;    0094  1      LIBRARY '$ds';                      !                                   [04]
;    0095  1
;    0096  1      LIBRARY 'sys$library:starlet.l32';
;    0097  1
```

```
;    0099  1      !+
;    0100  1      ! linkages
;    0101  1      !-
;    0102  1
;    0103  1      LINKAGE
;    0104  1          jsb_2 = JSB (REGISTER = 0, REGISTER = 1),
;    0105  1          jsb_3 = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2);
;    0106  1
```

```
;    0108  1      !+
;    0109  1      ! TABLE OF CONTENTS:
;    0110  1      !-
;    0111  1
;    0112  1      FORWARD ROUTINE
;    0113  1          dsx$help,                          ! Main program
;    0114  1          do_help,                           ! Recursive file search
;    0115  1          is_key_on_line,                    ! Check for key record
;    0116  1          print_keys : NOVALUE,              ! Print keys level 1 - x
;    0117  1          print_text,                        ! Print help text for level
;    0118  1          print_otherhelp,                   ! List all keys at level
;    0119  1          READ,                              ! Read random RFA
;    0120  1          dsr$key_equal,                     ! Compare keys (wildcard, partial match, etc)
;    0121  1          skip_blanks,                       ! Skip spaces/tabs
;    0122  1          scan_word,                         ! Find length of word
;    0123  1          find_char : jsb_2,                 ! Find character on line
;    0124  1          print_crlf : jsb_none NOVALUE;     ! Cause a blank line to magically appear
;    0125  1
```

```
;    0127  1      !+
;    0128  1      ! PSECT definitions:
;    0129  1      !-
;    0130  1      PSECT
;    0131  1          PLIT = data(SHARE, ADDRESSING_MODE (LONG_RELATIVE));
;    0132  1
;    0133  1      PSECT
;    0134  1          GLOBAL = work(NOSHARE, ADDRESSING_MODE (LONG_RELATIVE));
;    0135  1
;    0136  1      PSECT
;    0137  1          CODE = CODE(EXECUTE, SHARE, NOWRITE);
;    0138  1
```

```
;   0140  1     !+
;   0141  1     ! Literals
;   0142  1     !-
;   0143  1
;   0144  1     BIND
;   0145  1         help_help = $ascid ('HELP');                ! Default first level key
;   0146  1      literal
;   0147  1      ultrix_v_mode = 0;                             ! ultrix flag                  [11]
;   0148  1     !
;   0149  1     ! External
;   0150  1     !
;   0151  1      external
;   0152  1             Ultrix_flags    :Byte addressing_mode (long_relative);  ! Flags for ultrix disk struc. [11]
```

```
;    0154  1      !+
;    0155  1      ! OWN storage:
;    0156  1      !-
;    0157  1
;    0158  1      GLOBAL
;    0159  1          helpinfo : bblock [hlp$c_size];          ! Help processing data block
;    0160  1
```

```
;    0162  1     %SBTTL 'Help control routine'
;    0163  1
;    0164  1     GLOBAL ROUTINE dsx$help (desc) =
;    0165  1
;    0166  1     !++
;    0167  1     ! FUNCTIONAL DESCRIPTION
;    0168  1     !       Process HELP command.  Scans appropriate file to extract help text
;    0169  1     !       for keywords
;    0170  1     !
;    0171  1     ! INPUTS
;    0172  1     !       desc    address of descriptor for remainder of command line
;    0173  1     !
;    0174  1     ! IMPLICIT INPUTS
;    0175  1     !       none
;    0176  1     !
;    0177  1     ! OUTPUTS
;    0178  1     !       desc    descriptor is updated to end of line
;    0179  1     !
;    0180  1     ! IMPLICIT OUTPUTS
;    0181  1     !       typeout to console
;    0182  1     !
;    0183  1     ! SIDE EFFECTS
;    0184  1     !       none
;    0185  1     !
;    0186  1     ! RETURN CODES
;    0187  1     !       none
;    0188  1     !--
;    0189  1
```

```
;    0191  2          BEGIN
;    0192  2
;    0193  2          LOCAL
;    0194  2              helpfab : $fab_decl,                    ! FAB block
;    0195  2              helprab : $rab_decl,                    ! RAB block
;    0196  2              startrfa : bblock [rfa$c_length],       ! Beginning of file
;    0197  2              buffer : bblock [hlp$c_maxrecsiz],      ! Allocate input buffer
;    0198  2              nextkey : VECTOR [hlp$c_maxrecsiz, BYTE],
;    0199  2              nextdesc : bblock [dsc$c_s_bln],
;    0200  2              level,
;    0201  2              elipsis,
;    0202  2              keylist : bblock [hlp$c_maxkeys*dsc$c_s_bln];   ! Keyword descriptors
;    0203  2
;    0204  2          MAP
;    0205  2              desc : REF bblock [dsc$c_s_bln];
;    0206  2
;    0207  2          print_crlf ();                              ! Leading blank line
;    0208  2          elipsis = '....';                           ! Three dots
;    0209  2          CH$FILL (0, hlp$c_size, helpinfo);          ! Clear data table
;    0210  2          helpinfo [hlp$l_allhelp] = hlp$c_maxkeys + 1;       ! Init 'allhelp' to deactivate it
;    0211  2          CH$FILL (0, hlp$c_maxkeys*dsc$c_s_bln, keylist);    ! Clear key descriptors
;    0212  2          helpinfo [hlp$l_keylist] = keylist;         ! Set up pointer to keys
;    0213  2          level = 0;
;    0214  2          helpinfo [hlp$l_rab] = helprab;             ! Set up pointer to RMS
;    0215  2              IF      .Ultrix_flags<ultrix_v_mode,1>
;    0216  2               THEN
;    0217  3                  $fab_init (fab = helpfab, dnm = '.hlp', fnm = 'evsaa.hlp')  ! Initialize FAB for Ultrix [11]
;    0218  2              ELSE
;    0219  2                  $fab_init (fab = helpfab, dnm = '.HLP', fnm = 'EVSAA');     ! Initialize FAB
;    0220  2          $rab_init (rab = helprab, fab = helpfab, ubf = buffer, usz = hlp$c_maxrecsiz);      ! Initialize RAB
;    0221  2
;    0222  2          WHILE 1 DO                                  ! Isolate the keywords
;    0223  3              BEGIN
;    0224  3
;    0225  3              BIND
;    0226  3                  curkey = keylist + dsc$c_s_bln*.level : bblock,
;    0227  3                  wildflags = helpinfo [hlp$b_wildflags] : BITVECTOR,
;    0228  3                  keytext = curkey [dsc$a_pointer] : REF VECTOR [, BYTE];
;    0229  3
;    0230  3              LOCAL
;    0231  3                  next_qual,
;    0232  3                  ptr;
;    0233  3
;    0234  3              IF NOT skip_blanks (.desc) THEN EXITLOOP;       ! End of line
;    0235  3
;    0236  3              curkey [dsc$a_pointer] = .desc [dsc$a_pointer]; ! Beginning of a word
;    0237  3              next_qual = find_char (.desc, %C'/');
;    0238  3
;    0239  3              IF .next_qual EQL 0                     ! If we're lookin' at it,
;    0240  3              THEN
;    0241  3                  next_qual = .desc [dsc$w_length];   !  it's a start, not an end
;    0242  3
;    0243  3              curkey [dsc$w_length] =                 ! Find length of word
;    0244  3              MIN (.next_qual, find_char (.desc, %C' '), find_char (.desc, %C'        '));
;    0245  3              desc [dsc$w_length] = .desc [dsc$w_length] -    ! Update descriptor
;    0246  3              .curkey [dsc$w_length];
;    0247  3              desc [dsc$a_pointer] = .desc [dsc$a_pointer] + .curkey [dsc$w_length];
```

```
;    0248  3              ptr = CH$FIND_SUB (.curkey [dsc$w_length], .curkey [dsc$a_pointer], 3, elipsis);
;    0249  3                                                    ! Look for '...' in key
;    0250  3
;    0251  4              IF NOT CH$FAIL (.ptr) AND .ptr EQL (.curkey [dsc$a_pointer] + .curkey [dsc$w_length] - 3)
;    0252  3              THEN                                  ! "<word>..."
;    0253  4                  BEGIN
;    0254  4                  helpinfo [hlp$l_allhelp] = .level + 2;      ! All lower keys wildcarded
;    0255  4                  curkey [dsc$w_length] = .curkey [dsc$w_length] - 3; ! Discard '...'
;    0256  4
;    0257  4                  IF .curkey [dsc$w_length] EQL 0     ! If just '...', wilcard this level, too
;    0258  4                  THEN
;    0259  4                      helpinfo [hlp$l_allhelp] = .helpinfo [hlp$l_allhelp] - 1
;    0260  4                  ELSE
;    0261  4                      level = .level + 1;             ! Count level if '...' had company
;    0262  4
;    0263  4                  INCR i FROM .helpinfo [hlp$l_allhelp] TO hlp$c_maxkeys DO
;    0264  4                      wildflags [.i - 1] = 1;         ! Set the wildcard flags
;    0265  4
;    0266  4                  EXITLOOP                            ! Don't need more keys
;    0267  3                  END;
;    0268  3
;    0269  3              INCR i FROM 0 TO .curkey [dsc$w_length] - 1 DO
;    0270  3
;    0271  3                  IF .keytext [.i] EQL %C'%' OR .keytext [.i] EQL %C'*' THEN wildflags [.level] = 1;
;    0272  3
;    0273  3                                                    ! Set flag if wildcard char.
;    0274  3              level = .level + 1
;    0275  2              END;
;    0276  2
;    0277  2          helpinfo [hlp$l_realkeys] = .level;       ! Number of keys
;    0278  2
;    0279  2          IF .level GTR 0
;    0280  2          THEN                                      ! If there's a key,
;    0281  3              BEGIN                                 ! Check for diagnostic help
;    0282  3
;    0283  3              BIND
;    0284  3                  keyltext = keylist [dsc$a_pointer] : REF VECTOR [, BYTE];
;    0285  3
;    0286  4              IF ((.keyltext [0] EQL %C'E') OR (.keyltext [0] EQL %C'e'))     !                    [11]
;    0287  4                                  AND (.KEY1TEXT [1] EQL %C'V' OR             !                    [10]
;    0288  4                                       .KEY1TEXT [1] EQL %C'v' OR             !                    [11]
;    0289  4                                       .KEY1TEXT [1] EQL %C'S' OR             !                    [10]
;    0290  4                                       .KEY1TEXT [1] EQL %C's' OR             !                    [11]
;    0291  4                                       .KEY1TEXT [1] EQL %C'C' OR             !                    [10]
;    0292  4                                       .KEY1TEXT [1] EQL %C'c' OR             !                    [11]
;    0293  4                                       .KEY1TEXT [1] EQL %C'N' OR             !                    [10]
;    0294  4                                       .KEY1TEXT [1] EQL %C'n' OR             !                    [11]
;    0295  4                                       .KEY1TEXT [1] EQL %C'T' OR             !                    [10]
;    0296  4                                       .KEY1TEXT [1] EQL %C't' OR             !                    [11]
;    0297  4                                       .KEY1TEXT [1] EQL %C'D' OR             !                    [10]
;    0298  4                                       .KEY1TEXT [1] EQL %C'd' OR             !                    [11]
;    0299  4                                       .KEY1TEXT [1] EQL %C'A' OR             !                    [11]
;    0300  4                                       .KEY1TEXT [1] EQL %C'a' OR             !                    [11]
;    0301  4                                       .KEY1TEXT [1] EQL %C'B' OR             !                    [10]
;    0302  4                                       .KEY1TEXT [1] EQL %C'b' OR             !                    [11]
;    0303  4                                       .KEY1TEXT [1] EQL %C'L' OR             !                    [10] !G:2
;    0304  4                                       .KEY1TEXT [1] EQL %C'l' OR             !                    [11] !G:2
```

```
;   0305  4                                   .KEY1TEXT [1] EQL %C'J' OR      !                                [10] !G:2
;   0306  4                                   .KEY1TEXT [1] EQL %C'j' OR      !                                [11] !G:2
;   0307  4                                   .KEY1TEXT [1] EQL %C'Z' OR      !                                [10]
;   0308  4                                   .KEY1TEXT [1] EQL %C'z')        !                                [11]
;   0309  3             THEN
;   0310  4                 BEGIN                                 ! Key 1 is actually file specifier for diagnostic help file
;   0311  4                 helpfab [fab$l_fna] = .keylist [dsc$a_pointer];     ! Change filename
;   0312  4                 helpfab [fab$b_fns] = .keylist [dsc$w_length];
;   0313  4                 CH$MOVE ((hlp$c_maxkeys - 1)*dsc$c_s_bln, keylist + dsc$c_s_bln, keylist);
;   0314  4                                                       ! Move other keys down one
;   0315  4                 helpinfo [hlp$b_wildflags] = .helpinfo [hlp$b_wildflags]^-1;        ! Move flags down one bit
;   0316  4
;   0317  4                 IF .helpinfo [hlp$l_allhelp] LEQ hlp$c_maxkeys
;   0318  4                 THEN                                  ! Correct ALLHELP level
;   0319  4                     helpinfo [hlp$l_allhelp] = .helpinfo [hlp$l_allhelp] - 1;
;   0320  4
;   0321  4                 helpinfo [hlp$l_realkeys] = .helpinfo [hlp$l_realkeys] - 1
;   0322  4                 END
;   0323  4
;   0324  2             END;
;   0325  2
;   0326  2         IF .helpinfo [hlp$l_realkeys] EQL 0 OR .keylist [dsc$w_length] EQL 0
;   0327  2         THEN                                          ! If null first key, make it
;   0328  3             BEGIN                                     ! 'HELP' by default
;   0329  3             CH$MOVE (dsc$c_s_bln, help_help, keylist);
;   0330  3             helpinfo [hlp$l_realkeys] = 1             ! Set one key found
;   0331  2             END;
;   0332  2
;   0333  2         IF NOT .(helpinfo [hlp$b_wildflags])<0, 1, 0>       !  If key 1 not wild
;   0334  2         THEN
;   0335  2
;   0336  3             IF (
;   0337  3
;   0338  3                 LOCAL
;   0339  3                     w;
;   0340  3
;   0341  3                 dsr$key_equal (help_help, keylist, w))      !  and it matches 'HELP'
;   0342  2             THEN
;   0343  2                 helpinfo [hlp$v_help_key] = 1;      !  then set flag                                  [07]
;   0344  2
;   0345  2         startrfa [rfa$l_vbn] = 1;                     ! Start of file
;   0346  2         startrfa [rfa$w_offset] = 0;
;   0347  3         (                                             ! Open the file
;   0348  3
;   0349  3         LOCAL
;   0350  3             stat;
;   0351  3
;   0352  2         stat = $open (fab = helpfab); IF NOT .stat THEN RETURN .stat);
;   0353  2
;   0354  3         IF (.helpfab [fab$b_rat] NEQ 0) OR  ( .Ultrix_flags<ultrix_v_mode,1> EQL 1 )
;   0355  3                                             ! If not no-attribute or ULTRIX         [11]
;   0356  2         THEN
;   0357  2             helpinfo [hlp$v_cr] = 1;                  ! Need explicit CR on typeout
;   0358  2
;   0359  3         (
;   0360  3
;   0361  3         LOCAL
```

```
;    0362  3              stat;
;    0363  3
;    0364  3          stat = $connect (rab = helprab);           ! Connect record stream
;    0365  2          IF NOT .stat THEN RETURN .stat);
;    0366  2          nextdesc [dsc$a_pointer] = nextkey;        ! Set key space for key 1
;    0367  3          (
;    0368  3
;    0369  3          LOCAL
;    0370  3              stat;
;    0371  3
;    0372  3          stat = do_help (1, startrfa, nextdesc);    ! Do everything, recursively
;    0373  2          IF NOT .stat THEN RETURN .stat);
;    0374  3          BEGIN                                      ! Check for tail-end stuff
;    0375  3
;    0376  3          LOCAL
;    0377  3              noerr;
;    0378  3
;    0379  3          startrfa [rfa$l_vbn] = 1;                  ! Reset RFA
;    0380  3          startrfa [rfa$w_offset] = 0;
;    0381  3
;    0382  4          IF NOT (noerr = .helpinfo [hlp$v_foundhelp])       ! If no help found
;    0383  3              OR .helpinfo [hlp$v_help_key]          !  or key 1 was 'HELP'                                  [07]
;    0384  3          THEN
;    0385  4              (
;    0386  4
;    0387  4              LOCAL
;    0388  4                  stat;
;    0389  4
;    0390  4              stat = print_otherhelp (1, startrfa, .noerr);   ! print error and/or otherhelp
;    0391  4              IF NOT .stat THEN RETURN .stat)
;    0392  4
;    0393  2          END;
;    0394  2          print_crlf ();                             ! Cause a trailing CRLF
;    0395  3          $close (fab = helpfab)                     ! Close the file
;    0396  1          END;                                       ! of help


                                                                .TITLE  HELP *** HELP Handle HELP command
                                                                .IDENT  \10.10-2\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                    50  4C  45  48  00000 P.AAB:  .ASCII  \HELP\                                          ;
                                          00000004  00004 P.AAA:  .LONG   4                                             ;
                                          00000000' 00008         .ADDRESS P.AAB                                        ;
                        70  6C  68  2E  61  61  73  76  65  0000C P.AAC:  .ASCII  \evsaa.hlp\                            ;
                                        70  6C  68  2E  00015 P.AAD:  .ASCII  \.hlp\                                    ;
                                    41  41  53  56  45  00019 P.AAE:  .ASCII  \EVSAA\                                   ;
                                        50  4C  48  2E  0001E P.AAF:  .ASCII  \.HLP\                                    ;

                                                                .PSECT  WORK,NOEXE,2

                                                        00000 HELPINFO::
                                                                .BLKB   46

                                                            HELP_HELP=          P.AAA
                                                            WILDFLAGS=          HELPINFO+44
```

```
                                                                .EXTRN  ULTRIX_FLAGS, SYS$OPEN
                                                                .EXTRN  SYS$CONNECT, SYS$CLOSE

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                        0FFC 00000      .ENTRY  DSX$HELP, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0164
                                                                                R11                                         ;
                                                    5B      0000V   CF  9E 00002    MOVAB   FIND_CHAR, R11                  ;
                                                    5A 00000000G    EF  9E 00007    MOVAB   ULTRIX_FLAGS, R10               ;
                                                    59 00000000'    EF  9E 0000E    MOVAB   HELP_HELP, R9                   ;
                                                    58 00000000'    EF  9E 00015    MOVAB   HELPINFO, R8                    ;
                                                    5E      FD2C    CE  9E 0001C    MOVAB   -724(SP), SP                    ;
                                                                0000V 30  00021    BSBW    PRINT_CRLF                      ; 0207
                                                    6E  2E2E2E2E    8F  D0 00024    MOVL    #774778414, ELIPSIS             ; 0208
            2E              00                      6E              00  2C 0002B    MOVC5   #0, (SP), #0, #46, HELPINFO     ; 0209
                                                                    68     00030                                            ;
                                                    68              06  D0 00031    MOVL    #6, HELPINFO                    ; 0210
            28              00                      6E              00  2C 00034    MOVC5   #0, (SP), #0, #40, KEYLIST      ; 0211
                                                                08      AE     00039                                        ;
                                                24  A8          08      AE  9E 0003B    MOVAB   KEYLIST, HELPINFO+36            ; 0212
                                                                    56  D4 00040    CLRL    LEVEL                           ; 0213
                                                04  A8      FF6C      CD  9E 00042    MOVAB   HELPRAB, HELPINFO+4             ; 0214
                                                    27              6A  E9 00048    BLBC    ULTRIX_FLAGS, 1$                ; 0215
    0050    8F              00                      6E              00  2C 0004B    MOVC5   #0, (SP), #0, #80, $RMS_PTR     ; 0217
                                                                B0      AD     00052                                        ;
                                                B0  AD      5003        8F  B0 00054    MOVW    #20483, $RMS_PTR                ;
                                                C6  AD              02  90 0005A    MOVB    #2, $RMS_PTR+22                 ;
                                                CF  AD              02  90 0005E    MOVB    #2, $RMS_PTR+31                 ;
                                                DC  AD          08      A9  9E 00062    MOVAB   P.AAC, $RMS_PTR+44              ;
                                                E0  AD          11      A9  9E 00067    MOVAB   P.AAD, $RMS_PTR+48              ;
                                                E4  AD                  09  90 0006C    MOVB    #9, $RMS_PTR+52                 ;
                                                                    25  11 00070    BRB     2$                              ;
    0050    8F              00                      6E              00  2C 00072 1$:    MOVC5   #0, (SP), #0, #80, $RMS_PTR     ; 0219
                                                                B0      AD     00079                                        ;
                                                B0  AD      5003        8F  B0 0007B    MOVW    #20483, $RMS_PTR                ;
                                                C6  AD              02  90 00081    MOVB    #2, $RMS_PTR+22                 ;
                                                CF  AD              02  90 00085    MOVB    #2, $RMS_PTR+31                 ;
                                                DC  AD          15      A9  9E 00089    MOVAB   P.AAE, $RMS_PTR+44              ;
                                                E0  AD          1A      A9  9E 0008E    MOVAB   P.AAF, $RMS_PTR+48              ;
                                                E4  AD                  05  90 00093    MOVB    #5, $RMS_PTR+52                 ;
                                                E5  AD                  04  90 00097 2$:    MOVB    #4, $RMS_PTR+53             ; 0217
    0044    8F              00                      6E              00  2C 0009B    MOVC5   #0, (SP), #0, #68, $RMS_PTR     ; 0220
                                                            FF6C        CD     000A2                                        ;
                                            FF6C    CD      4401        8F  B0 000A5    MOVW    #17409, $RMS_PTR                ;
                                              8C    AD      0100        8F  B0 000AC    MOVW    #256, $RMS_PTR+32               ;
                                              90    AD      0138        CE  9E 000B2    MOVAB   BUFFER, $RMS_PTR+36             ;
                                              A8    AD        B0        AD  9E 000B8    MOVAB   HELPFAB, $RMS_PTR+60            ;
                                                    55        04        AC  D0 000BD    MOVL    DESC, R5                        ; 0234
                                                    54        08  AE46  7E  C1 000C1 3$:    MOVAQ   KEYLIST[LEVEL], R4      ; 0226
                                                    57        04        A4  9E 000C6    MOVAB   4(R4), R7                       ; 0228
                                                              55        DD 000CA        PUSHL   R5                              ; 0234
                                            0000V   CF                  01  FB 000CC    CALLS   #1, SKIP_BLANKS             ;
                                                    03                  50  E8 000D1    BLBS    R0, 4$                      ;
                                                                    00A9    31 000D4    BRW     17$                         ;
                                              04    A4        04        A5  D0 000D7 4$:    MOVL    4(R5), 4(R4)            ; 0236
                                                    51                  2F  D0 000DC    MOVL    #47, R1                     ; 0237
                                                    50                  55  D0 000DF    MOVL    R5, R0                      ;
```

```
                                                  6B  16 000E2          JSB     FIND_CHAR                                  ;
                                  52              50  D0 000E4          MOVL    R0, NEXT_QUAL                              ;
                                                  03  12 000E7          BNEQ    5$                                         ; 0239
                                  52              65  3C 000E9          MOVZWL  (R5), NEXT_QUAL                            ; 0241
                                  51              20  D0 000EC 5$:      MOVL    #32, R1                                    ; 0244
                                  50              55  D0 000EF          MOVL    R5, R0                                     ;
                                                  6B  16 000F2          JSB     FIND_CHAR                                  ;
                                  50              52  D1 000F4          CMPL    R2, R0                                     ;
                                                  03  15 000F7          BLEQ    6$                                         ;
                                  52              50  D0 000F9          MOVL    R0, R2                                     ;
                                  51              09  D0 000FC 6$:      MOVL    #9, R1                                     ;
                                  50              55  D0 000FF          MOVL    R5, R0                                     ;
                                                  6B  16 00102          JSB     FIND_CHAR                                  ;
                                  50              52  D1 00104          CMPL    R2, R0                                     ;
                                                  03  15 00107          BLEQ    7$                                         ;
                                  52              50  D0 00109          MOVL    R0, R2                                     ;
                                  64              52  B0 0010C 7$:      MOVW    R2, (R4)                                   ;
                                  65              64  A2 0010F          SUBW2   (R4), (R5)                                 ; 0246
                                  50              64  3C 00112          MOVZWL  (R4), R0                                   ; 0247
                            04    A5              50  C0 00115          ADDL2   R0, 4(R5)                                  ;
  04   B4          64             6E              03  39 00119          MATCHC  #3, ELIPSIS, (R4), @4(R4)                  ; 0248
                                                  03  13 0011F          BEQL    8$                                         ;
                                  53              03  D0 00121          MOVL    #3, R3                                     ;
                                  53              03  C2 00124 8$:      SUBL2   #3, R3                                     ;
                                                  33  13 00127          BEQL    13$                                        ; 0251
                                  50              64  3C 00129          MOVZWL  (R4), R0                                   ;
                                  50        04    A4  C0 0012C          ADDL2   4(R4), R0                                  ;
                                  50              03  C2 00130          SUBL2   #3, R0                                     ;
                                  50              53  D1 00133          CMPL    PTR, R0                                    ;
                                                  24  12 00136          BNEQ    13$                                        ;
                                  68        02    A6  9E 00138          MOVAB   2(R6), HELPINFO                            ; 0254
                                  64              03  A2 0013C          SUBW2   #3, (R4)                                   ; 0255
                                                  04  12 0013F          BNEQ    9$                                         ; 0257
                                                  68  D7 00141          DECL    HELPINFO                                   ; 0259
                                                  02  11 00143          BRB     10$                                        ;
                                                  56  D6 00145 9$:      INCL    LEVEL                                      ; 0261
                  50              68              01  C3 00147 10$:     SUBL3   #1, HELPINFO, I                            ; 0263
                                                  09  11 0014B          BRB     12$                                        ;
                                  51        FF    A0  9E 0014D 11$:     MOVAB   -1(R0), R1                                 ; 0264
                  00        2C    A8              51  E2 00151          BBSS    R1, WILDFLAGS, 12$                         ;
                  F3              50              05  F3 00156 12$:     AOBLEQ  #5, I, 11$                                 ;
                                                  24  11 0015A          BRB     17$                                        ; 0263
                                  51              64  3C 0015C 13$:     MOVZWL  (R4), R1                                   ; 0269
                                  50              01  CE 0015F          MNEGL   #1, I                                      ;
                                                  13  11 00162          BRB     16$                                        ;
                                  25        00 B740   91 00164 14$:     CMPB    @0(R7)[I], #37                             ; 0271
                                                  07  13 00169          BEQL    15$                                        ;
                                  2A        00 B740   91 0016B          CMPB    @0(R7)[I], #42                             ;
                                                  05  12 00170          BNEQ    16$                                        ;
                  00        2C    A8              56  E2 00172 15$:     BBSS    LEVEL, WILDFLAGS, 16$                      ;
                  E9              50              51  F2 00177 16$:     AOBLSS  R1, I, 14$                                 ;
                                                  56  D6 0017B          INCL    LEVEL                                      ; 0274
                                               FF41   31 0017D          BRW     3$                                         ;
                            28    A8              56  D0 00180 17$:     MOVL    LEVEL, HELPINFO+40                         ; 0277
                                                  03  14 00184          BGTR    19$                                        ; 0279
                                               00BE   31 00186 18$:     BRW     23$                                        ;
                                  51        0C    AE  D0 00189 19$:     MOVL    KEY1TEXT, R1                               ; 0286
```

```
                    45   8F          61  91 0018D        CMPB    (R1), #69                          ;
                                     06  13 00191        BEQL    20$                                ;
                    65   8F          61  91 00193        CMPB    (R1), #101                         ;
                                     ED  12 00197        BNEQ    18$                                ;
                         50     01   A1  9A 00199 20$:   MOVZBL  1(R1), R0                          ; 0287
                    56   8F          50  91 0019D        CMPB    R0, #86                            ;
                                     7E  13 001A1        BEQL    21$                                ;
                    76   8F          50  91 001A3        CMPB    R0, #118                           ; 0288
                                     78  13 001A7        BEQL    21$                                ;
                    53   8F          50  91 001A9        CMPB    R0, #83                            ; 0289
                                     72  13 001AD        BEQL    21$                                ;
                    73   8F          50  91 001AF        CMPB    R0, #115                           ; 0290
                                     6C  13 001B3        BEQL    21$                                ;
                    43   8F          50  91 001B5        CMPB    R0, #67                            ; 0291
                                     66  13 001B9        BEQL    21$                                ;
                    63   8F          50  91 001BB        CMPB    R0, #99                            ; 0292
                                     60  13 001BF        BEQL    21$                                ;
                    4E   8F          50  91 001C1        CMPB    R0, #78                            ; 0293
                                     5A  13 001C5        BEQL    21$                                ;
                    6E   8F          50  91 001C7        CMPB    R0, #110                           ; 0294
                                     54  13 001CB        BEQL    21$                                ;
                    54   8F          50  91 001CD        CMPB    R0, #84                            ; 0295
                                     4E  13 001D1        BEQL    21$                                ;
                    74   8F          50  91 001D3        CMPB    R0, #116                           ; 0296
                                     48  13 001D7        BEQL    21$                                ;
                    44   8F          50  91 001D9        CMPB    R0, #68                            ; 0297
                                     42  13 001DD        BEQL    21$                                ;
                    64   8F          50  91 001DF        CMPB    R0, #100                           ; 0298
                                     3C  13 001E3        BEQL    21$                                ;
                    41   8F          50  91 001E5        CMPB    R0, #65                            ; 0299
                                     36  13 001E9        BEQL    21$                                ;
                    61   8F          50  91 001EB        CMPB    R0, #97                            ; 0300
                                     30  13 001EF        BEQL    21$                                ;
                    42   8F          50  91 001F1        CMPB    R0, #66                            ; 0301
                                     2A  13 001F5        BEQL    21$                                ;
                    62   8F          50  91 001F7        CMPB    R0, #98                            ; 0302
                                     24  13 001FB        BEQL    21$                                ;
                    4C   8F          50  91 001FD        CMPB    R0, #76                            ; 0303
                                     1E  13 00201        BEQL    21$                                ;
                    6C   8F          50  91 00203        CMPB    R0, #108                           ; 0304
                                     18  13 00207        BEQL    21$                                ;
                    4A   8F          50  91 00209        CMPB    R0, #74                            ; 0305
                                     12  13 0020D        BEQL    21$                                ;
                    6A   8F          50  91 0020F        CMPB    R0, #106                           ; 0306
                                     0C  13 00213        BEQL    21$                                ;
                    5A   8F          50  91 00215        CMPB    R0, #90                            ; 0307
                                     06  13 00219        BEQL    21$                                ;
                    7A   8F          50  91 0021B        CMPB    R0, #122                           ; 0308
                                     26  12 0021F        BNEQ    23$                                ;
                    DC   AD          51  D0 00221 21$:   MOVL    R1, HELPFAB+44                     ; 0311
                    E4   AD     08   AE  90 00225        MOVB    KEYLIST, HELPFAB+52                ; 0312
        08   AE     10   AE          20  28 0022A        MOVC3   #32, KEYLIST+8, KEYLIST            ; 0313
                         50     2C   A8  9A 00230        MOVZBL  HELPINFO+44, R0                    ; 0315
             50          50     FF   8F  78 00234        ASHL    #-1, R0, R0                        ;
                    2C   A8          50  90 00239        MOVB    R0, HELPINFO+44                    ;
                         05          68  D1 0023D        CMPL    HELPINFO, #5                       ; 0317
                                     02  14 00240        BGTR    22$                                ;
```

```
                                        68  D7 00242          DECL    HELPINFO                                  ; 0319
                                    28  A8  D7 00244 22$:     DECL    HELPINFO+40                               ; 0321
                                    28  A8  D5 00247 23$:     TSTL    HELPINFO+40                               ; 0326
                                        05  13 0024A          BEQL    24$                                       ;
                                    08  AE  B5 0024C          TSTW    KEYLIST                                   ;
                                        09  12 0024F          BNEQ    25$                                       ;
          08   AE          69           08  28 00251 24$:     MOVC3   #8, HELP_HELP, KEYLIST                    ; 0329
                      28   A8           01  D0 00256          MOVL    #1, HELPINFO+40                           ; 0330
                           14      2C   A8  E8 0025A 25$:     BLBS    HELPINFO+44, 26$                          ; 0333
                                   04   AE  9F 0025E          PUSHAB  W                                         ; 0341
                                   0C   AE  9F 00261          PUSHAB  KEYLIST                                   ;
                                        59  DD 00264          PUSHL   R9                                        ;
                   0000V   CF           03  FB 00266          CALLS   #3, DSR$KEY_EQUAL                         ;
                           04           50  E9 0026B          BLBC    R0, 26$                                   ;
                      2D   A8           01  88 0026E          BISB2   #1, HELPINFO+45                           ; 0343
                    FF64   CD           01  D0 00272 26$:     MOVL    #1, STARTRFA                              ; 0345
                                 FF68   CD  B4 00277          CLRW    STARTRFA+4                                ; 0346
                                   B0   AD  9F 0027B          PUSHAB  HELPFAB                                   ; 0352
              00000000G    00           01  FB 0027E          CALLS   #1, SYS$OPEN                              ;
                           63           50  E9 00285          BLBC    STAT, 31$                                 ;
                                   CE   AD  95 00288          TSTB    HELPFAB+30                                ; 0354
                                        03  12 0028B          BNEQ    27$                                       ;
                           04           6A  E9 0028D          BLBC    ULTRIX_FLAGS, 28$                         ;
                      2D   A8           08  88 00290 27$:     BISB2   #8, HELPINFO+45                           ; 0357
                                 FF6C   CD  9F 00294 28$:     PUSHAB  HELPRAB                                   ; 0364
              00000000G    00           01  FB 00298          CALLS   #1, SYS$CONNECT                           ;
                           49           50  E9 0029F          BLBC    STAT, 31$                                 ; 0365
                      34   AE      38   AE  9E 002A2          MOVAB   NEXTKEY, NEXTDESC+4                       ; 0366
                                   30   AE  9F 002A7          PUSHAB  NEXTDESC                                  ; 0372
                                 FF64   CD  9F 002AA          PUSHAB  STARTRFA                                  ;
                                        01  DD 002AE          PUSHL   #1                                        ;
                   0000V   CF           03  FB 002B0          CALLS   #3, DO_HELP                               ;
                           33           50  E9 002B5          BLBC    STAT, 31$                                 ; 0373
                    FF64   CD           01  D0 002B8          MOVL    #1, STARTRFA                              ; 0379
                                 FF68   CD  B4 002BD          CLRW    STARTRFA+4                                ; 0380
     50      2D   A8          01           02  EF 002C1          EXTZV   #2, #1, HELPINFO+45, NOERR                ; 0382
                           04           50  E9 002C7          BLBC    NOERR, 29$                                ;
                           10      2D   A8  E9 002CA          BLBC    HELPINFO+45, 30$                          ; 0383
                                        50  DD 002CE 29$:     PUSHL   NOERR                                     ; 0390
                                 FF64   CD  9F 002D0          PUSHAB  STARTRFA                                  ;
                                        01  DD 002D4          PUSHL   #1                                        ;
                   0000V   CF           03  FB 002D6          CALLS   #3, PRINT_OTHERHELP                       ;
                           0D           50  E9 002DB          BLBC    STAT, 31$                                 ; 0391
                                     0000V 30 002DE 30$:     BSBW    PRINT_CRLF                                ; 0394
                                   B0   AD  9F 002E1          PUSHAB  HELPFAB                                   ; 0395
              00000000G    00           01  FB 002E4          CALLS   #1, SYS$CLOSE                             ;
                                            04 002EB 31$:     RET                                               ; 0396

; Routine Size:  748 bytes,    Routine Base:  CODE + 0000


;   0397  1
```

```
;    0399  1      %SBTTL 'do help stuff'
;    0400  1      ROUTINE do_help (level, firstrfa, key) =
;    0401  1
;    0402  1      !++
;    0403  1      ! FUNCTIONAL DESCRIPTION
;    0404  1      !       Recursive routine to scan for all help.  Each incarnation will call
;    0405  1      !       itself at each matched key of it's level, to scan lower levels of
;    0406  1      !       that key.  Matched keys are printed, lowest level help text is
;    0407  1      !       printed.  If there is no match, the message 'No help for...' is
;    0408  1      !       typed, and a list of other help available is typed.  This list is
;    0409  1      !       typed anyway at the first level if the first level key is 'HELP'.
;    0410  1      !
;    0411  1      ! INPUTS
;    0412  1      !       level           the level of help for which this incarnation is
;    0413  1      !                       to extract help.
;    0414  1      !
;    0415  1      !       firstrfa        The RFA to begin searching the file at.
;    0416  1      !       key             address of descriptor to save our key
;    0417  1      !
;    0418  1      ! IMPLICIT INPUTS
;    0419  1      !       none
;    0420  1      !
;    0421  1      ! OUTPUTS
;    0422  1      !       none
;    0423  1      !
;    0424  1      ! IMPLICIT OUTPUTS
;    0425  1      !       several flags in the HELPINFO block.
;    0426  1      !
;    0427  1      ! SIDE EFFECTS
;    0428  1      !       updates RFA block beyond text used at this level, unless Allhelped
;    0429  1      !
;    0430  1      ! RETURN CODES
;    0431  1      !       none
;    0432  1      !--
;    0433  1
```

```
;    0435  2          BEGIN
;    0436  2
;    0437  2          BIND
;    0438  2              rab = .helpinfo [hlp$l_rab] : bblock,
;    0439  2              curkey = (.helpinfo [hlp$l_keylist] + ((.level - 1)*dsc$c_s_bln)) : bblock;
;    0440  2
;    0441  2          MAP
;    0442  2              key : REF bblock,
;    0443  2              firstrfa : REF bblock;
;    0444  2
;    0445  2          LOCAL
;    0446  2              linelevel,                              ! level of current key line
;    0447  2              qualifier,                              ! Set if curkey begins w/ "/"
;    0448  2              char : BYTE,
;    0449  2              nextkey : VECTOR [hlp$c_maxrecsiz, BYTE],       ! space to hold lower key
;    0450  2              nextdesc : bblock [dsc$c_s_bln],        ! descriptor lower key
;    0451  2              keydesc : bblock [dsc$c_s_bln],
;    0452  2              bound_level,
;    0453  2              lastlevel,                              ! last key level found
;    0454  2              saverfa : bblock [rfa$c_length];        ! Where we started
;    0455  2
;    0456  3          IF .level GTR hlp$c_maxkeys OR (.level GTR .helpinfo [hlp$l_realkeys] AND .level LSS .helpinfo [hlp$l_allhelp
;    0457  3              ])
;    0458  2          THEN
;    0459  2              RETURN 1;                               ! We oughtn't to be here,
;    0460  2
;    0461  2          IF .curkey [dsc$a_pointer] NEQ 0            ! If key is not null
;    0462  2          THEN
;    0463  2              char = .(.curkey [dsc$a_pointer])<0, 8, 0>      ! Get first char of input key
;    0464  2          ELSE
;    0465  2              char = 0;                               ! otherwise, clear it
;    0466  2
;    0467  2          qualifier = .char EQL %C'/';
;    0468  2          bound_level = lastlevel = .level - 1;       ! Init last level found
;    0469  2          CH$MOVE (rfa$c_length, .firstrfa, rab [rab$w_rfa]); ! Set RFA to read
;    0470  2
;    0471  2          DO
;    0472  3              BEGIN                                   ! Find first key of right level
;    0473  3
;    0474  3              LOCAL
;    0475  3                  stat;                               ! status
;    0476  3
;    0477  3              CH$MOVE (rfa$c_length, rab [rab$w_rfa], saverfa);       ! Save start RFA
;    0478  3              stat = READ ();                         ! Read a record
;    0479  3
;    0480  3              IF NOT .stat                            ! If some nasty error hit us
;    0481  3              THEN
;    0482  3
;    0483  3                  IF .stat EQL rms$_eof THEN EXITLOOP ELSE RETURN .stat;      ! Say it didn't work out
;    0484  3
;    0485  3              linelevel = is_key_on_line (keydesc);   ! Get level and key from line
;    0486  3              END
;    0487  4          UNTIL (IF NOT .qualifier THEN (.linelevel LEQ .level) ELSE (.linelevel EQL hlp$c_maxkeys + 1 OR .linelevel
;    0488  3                  LEQ .bound_level))                  ! Qual. search can skip 1 level
;    0489  2              AND .linelevel NEQ 0;                   ! until we find our level, or won't
;    0490  2
;    0491  2          IF .linelevel LSS .level                    ! No help at this level
```

```
;    0492  2              AND NOT .qualifier
;    0493  2          THEN
;    0494  3              BEGIN
;    0495  3              CH$MOVE (rfa$c_length, rab [rab$w_rfa], .firstrfa);     ! Backup RFA
;    0496  3              RETURN 1
;    0497  2              END;
;    0498  2
```

```
;    0500  2
;    0501  2          !+
;    0502  2          ! Now process the file, from current position to the next key of higher level
;    0503  2          ! (i.e., level 2 processing ends when level 1 key is found, level 1
;    0504  2          ! processing ends when end-of-file is reached).
;    0505  2          !-
;    0506  2
;    0507  2          rab [rab$b_rac] = rab$c_rfa;                ! Set random for fst record
;    0508  2
;    0509  2          WHILE 1 DO
;    0510  3              BEGIN
;    0511  3
;    0512  3              LOCAL
;    0513  3                  wildkey,                            ! Flag for wildcarding key
;    0514  3                  stat,                               ! Status
;    0515  3                  linelevel;                          ! level of current line
;    0516  3
;    0517  3              stat = READ ();                         ! Read next record
;    0518  3
;    0519  3              IF NOT .stat
;    0520  3              THEN
;    0521  3
;    0522  3                  IF .stat EQL rms$_eof THEN EXITLOOP ELSE RETURN .stat;
;    0523  3
;    0524  3              linelevel = is_key_on_line (keydesc);   ! Find key on this line
;    0525  3
;    0526  3              IF .linelevel LEQ hlp$c_maxkeys         ! If this is numeric key
;    0527  3                  AND .linelevel GTR 0                !  level (not qualifier)
;    0528  3              THEN
;    0529  3                  lastlevel = .linelevel;             ! then remember it
;    0530  3
;    0531  3              IF .linelevel LEQ .bound_level AND .linelevel NEQ 0 THEN EXITLOOP;      ! Exit if found key we don't want
;    0532  3
;    0533  3              IF .linelevel EQL .level                ! If levels match,
;    0534  3                  OR .level GEQ .helpinfo [hlp$l_allhelp]     !  or ALLHELPed
;    0535  4                  OR (.qualifier                      !  or key is qualifier
;    0536  4                  AND .linelevel EQL hlp$c_maxkeys + 1        !   and so is line
;    0537  4                  AND .lastlevel EQL .bound_level)    !   and it's at right level
;    0538  3              THEN                                    ! This is a possible
;    0539  3
;    0540  3                  IF dsr$key_equal (keydesc, curkey, wildkey) ! Compare the keys
;    0541  3                      OR .level GEQ .helpinfo [hlp$l_allhelp] !  see if all help ('...') used
;    0542  3                  THEN
;    0543  4                      BEGIN                           ! They ARE equal...go to it
;    0544  4
;    0545  4                      BIND
;    0546  4                          keynames = helpinfo [hlp$l_keynames] : VECTOR [hlp$c_maxkeys, LONG];
;    0547  4
;    0548  4                      keynames [.bound_level] = .key; ! Set up static key pointer
;    0549  4                      CH$MOVE (key [dsc$w_length] =   ! Set up key text--all rest
;    0550  4                          .helpinfo [hlp$l_bufdesc] -         !  or record from key on
;    0551  4                          (.keydesc [dsc$a_pointer] - .(helpinfo [hlp$l_bufdesc] + 4)), .keydesc [dsc$a_pointer],
;    0552  4                          .key [dsc$a_pointer]);      ! Copy key to save it
;    0553  4                      helpinfo [hlp$v_qual] = .linelevel EQL hlp$c_maxkeys + 1;       ! Set flag if this is qualifier
;    0554  4
;    0555  4                      IF .level EQL .helpinfo [hlp$l_realkeys]        ! If this is lowest level
;    0556  4                          OR .level GEQ .helpinfo [hlp$l_allhelp]     !  or we want it all
```

```
;    0557  4                    THEN
;    0558  5                        BEGIN
;    0559  5                        helpinfo [hlp$v_foundhelp] = 1;     ! We've found something
;    0560  5                        print_keys (.level);            ! Print out keywords found
;    0561  5                        print_text (.linelevel, .level);    ! Print out text under key
;    0562  5                        rab [rab$b_rac] = rab$c_rfa ! Set random
;    0563  4                        END;
;    0564  4
;    0565  4                    nextdesc [dsc$a_pointer] = nextkey;     ! Set up lower key space
;    0566  5                    (
;    0567  5
;    0568  5                    LOCAL
;    0569  5                        stat;
;    0570  5
;    0571  5                    stat = do_help (.level + 1, rab [rab$w_rfa], nextdesc); ! Find lower level help
;    0572  4                    IF NOT .stat THEN RETURN .stat);
;    0573  4
;    0574  4                    IF .level EQL .helpinfo [hlp$l_realkeys]        ! If lowest level
;    0575  4                    THEN
;    0576  5                        (
;    0577  5
;    0578  5                        LOCAL
;    0579  5                            stat;
;    0580  5
;    0581  5                        stat = print_otherhelp (.level + 1, rab [rab$w_rfa], 1);    ! type lower keys
;    0582  4                        IF NOT .stat THEN RETURN .stat);
;    0583  4
;    0584  4                    IF NOT .wildkey AND .level LSS .helpinfo [hlp$l_allhelp] THEN RETURN 1;
;    0585  4
;    0586  4                    ! If this key wasn't wild, (even implicitly) don't continue searching
;    0587  4                    rab [rab$b_rac] = rab$c_rfa     ! Next loop read is rnd
;    0588  3                    END;                            ! Key found
;    0589  3
;    0590  2                END;                                ! Loop
;    0591  2
```

```
;   0593  2     !+
;   0594  2     ! Now, type out list of other keys at this level if:
;   0595  2     !       o There was no help found at this level (or a lower level)
;   0596  2     !         and this level is the highest wild-carded level or no levels
;   0597  2     !         are wildcarded.
;   0598  2     !       o If this is level 1 and the key was 'HELP' or if no text or
;   0599  2     !         other keys were printed out elsewhere.
;   0600  2     !-
;   0601  3         BEGIN
;   0602  3
;   0603  3         LOCAL
;   0604  3             position,
;   0605  3             size,
;   0606  3             first,
;   0607  3             wildpath;
;   0608  3
;   0609  3         BUILTIN
;   0610  3             FFS;
;   0611  3
;   0612  3         position = 0;
;   0613  3         size = .level - 1;
;   0614  3         wildpath = NOT FFS (position, size,          ! See if wild path here
;   0615  3             helpinfo [hlp$b_wildflags], first);
;   0616  3
;   0617  4         IF ( NOT .helpinfo [hlp$v_foundhelp]         ! If haven't found help
;   0618  5             AND NOT (.level GEQ .helpinfo [hlp$l_allhelp]   !  and not all-helped
;   0619  4             OR .wildpath))                           !  and not wild path
;   0620  3         THEN
;   0621  4             BEGIN                                    !                                                    [03]
;   0622  4
;   0623  4             LOCAL
;   0624  4                 stat;
;   0625  4
;   0626  4             rab [rab$b_rac] = rab$c_rfa;             ! Set to random access                               [03]
;   0627  4             CH$MOVE (6, saverfa, rab [rab$w_rfa]);   ! Copy begin RFA                                     [03]
;   0628  4             stat = READ ();                          ! Position record stream                             [03]
;   0629  4             stat = READ ();                          ! Advance past begin key                             [03]
;   0630  4
;   0631  4             IF NOT .stat THEN RETURN .stat;          !                                                    [03]
;   0632  4
;   0633  4             stat = print_otherhelp (.level, rab [rab$w_rfa], 0);    ! Type out other keys
;   0634  4
;   0635  4             IF NOT .stat THEN RETURN .stat
;   0636  4
;   0637  4             END                                      !                                                    [03]
;   0638  2         END;
;   0639  2
;   0640  2         1
;   0641  1         END;                                         ! of do_help


                                                  KEYNAMES=          HELPINFO+16


                                  0FFC 00000 DO_HELP:.WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0400
                     5E     FEDC   CE  9E 00002         MOVAB   -292(SP), SP                                ;
```

```
                                58 00000000'  EF  D0 00007          MOVL    HELPINFO+4, R8                          ; 0438
                                59        04  AC  D0 0000E          MOVL    LEVEL, R9                               ; 0439
                                5B 00000000'FF49  7E 00012          MOVAQ   @HELPINFO+36[R9], R11                   ;
                                5B            08  C2 0001A          SUBL2   #8, R11                                 ;
                                05            59  D1 0001D          CMPL    R9, #5                                  ; 0456
                                              03  15 00020          BLEQ    1$                                      ;
                                            01ED  31 00022          BRW     29$                                     ;
                     00000000'  EF            59  D1 00025 1$:      CMPL    R9, HELPINFO+40                         ;
                                              09  15 0002C          BLEQ    2$                                      ;
                     00000000'  EF            59  D1 0002E          CMPL    R9, HELPINFO                            ;
                                              73  19 00035          BLSS    12$                                     ;
                                        04    AB  D5 00037 2$:      TSTL    4(R11)                                  ; 0461
                                              06  13 0003A          BEQL    3$                                      ;
                                51      04    BB  90 0003C          MOVB    @4(R11), CHAR                           ; 0463
                                              02  11 00040          BRB     4$                                      ;
                                              51  94 00042 3$:      CLRB    CHAR                                    ; 0465
                                              50  D4 00044 4$:      CLRL    R0                                      ; 0467
                                2F            51  91 00046          CMPB    CHAR, #47                               ;
                                              02  12 00049          BNEQ    5$                                      ;
                                              50  D6 0004B          INCL    R0                                      ;
                                57            50  D0 0004D 5$:      MOVL    R0, QUALIFIER                           ;
                                56      FF    A9  9E 00050          MOVAB   -1(R9), LASTLEVEL                       ; 0468
                                6E            56  D0 00054          MOVL    LASTLEVEL, BOUND_LEVEL                  ;
          10    A8        08    BC            06  28 00057          MOVC3   #6, @FIRSTRFA, 16(R8)                   ; 0469
                          04    AE            57  D2 0005D          MCOML   QUALIFIER, 4(SP)                        ; 0487
          0C    AE        10    A8            06  28 00061 6$:      MOVC3   #6, 16(R8), SAVERFA                     ; 0477
                        0000V   CF            00  FB 00067          CALLS   #0, READ                                ; 0478
                                0A            50  E8 0006C          BLBS    STAT, 7$                                ; 0480
                     0001827A   8F            50  D1 0006F          CMPL    STAT, #98938                            ; 0483
                                              23  13 00076          BEQL    11$                                     ;
                                                  04 00078          RET                                             ;
                                        14    AE  9F 00079 7$:      PUSHAB  KEYDESC                                 ; 0485
                        0000V   CF            01  FB 0007C          CALLS   #1, IS_KEY_ON_LINE                      ;
                                5A            50  D0 00081          MOVL    R0, LINELEVEL                           ;
                                05      04    AE  E9 00084          BLBC    4(SP), 8$                               ; 0487
                                59            5A  D1 00088          CMPL    LINELEVEL, R9                           ;
                                              08  11 0008B          BRB     9$                                      ;
                                06            5A  D1 0008D 8$:      CMPL    LINELEVEL, #6                           ;
                                              05  13 00090          BEQL    10$                                     ;
                                6E            5A  D1 00092          CMPL    LINELEVEL, BOUND_LEVEL                  ; 0488
                                              CA  14 00095 9$:      BGTR    6$                                      ;
                                              5A  D5 00097 10$:     TSTL    LINELEVEL                               ; 0489
                                              C6  13 00099          BEQL    6$                                      ;
                                59            5A  D1 0009B 11$:     CMPL    LINELEVEL, R9                           ; 0491
                                              0D  18 0009E          BGEQ    13$                                     ;
                                09      04    AE  E9 000A0          BLBC    4(SP), 13$                              ; 0492
          08    BC        10    A8            06  28 000A4          MOVC3   #6, 16(R8), @FIRSTRFA                   ; 0495
                                            0165  31 000AA 12$:     BRW     29$                                     ; 0496
                          1E    A8            02  90 000AD 13$:     MOVB    #2, 30(R8)                              ; 0507
                        0000V   CF            00  FB 000B1 14$:     CALLS   #0, READ                                ; 0517
                                0D            50  E8 000B6          BLBS    STAT, 17$                               ; 0519
                     0001827A   8F            50  D1 000B9          CMPL    STAT, #98938                            ; 0522
                                              03  12 000C0          BNEQ    16$                                     ;
                                            00FB  31 000C2 15$:     BRW     27$                                     ;
                                                  04 000C5 16$:     RET                                             ;
                                        14    AE  9F 000C6 17$:     PUSHAB  KEYDESC                                 ; 0524
                        0000V   CF            01  FB 000C9          CALLS   #1, IS_KEY_ON_LINE                      ;
```

```
                                 5A          50  D0 000CE          MOVL    R0, LINELEVEL                          ;
                                 05          5A  D1 000D1          CMPL    LINELEVEL, #5                          ; 0526
                                             07  14 000D4          BGTR    18$                                    ;
                                             5A  D5 000D6          TSTL    LINELEVEL                              ; 0527
                                             03  15 000D8          BLEQ    18$                                    ;
                                 56          5A  D0 000DA          MOVL    LINELEVEL, LASTLEVEL                   ; 0529
                                 6E          5A  D1 000DD 18$:     CMPL    LINELEVEL, BOUND_LEVEL                 ; 0531
                                             04  14 000E0          BGTR    19$                                    ;
                                             5A  D5 000E2          TSTL    LINELEVEL                              ;
                                             DC  12 000E4          BNEQ    15$                                    ;
                                 59          5A  D1 000E6 19$:     CMPL    LINELEVEL, R9                          ; 0533
                                             16  13 000E9          BEQL    20$                                    ;
                     00000000'   EF          59  D1 000EB          CMPL    R9, HELPINFO                           ; 0534
                                             0D  18 000F2          BGEQ    20$                                    ;
                                 BA          57  E9 000F4          BLBC    QUALIFIER, 14$                         ; 0535
                                 06          5A  D1 000F7          CMPL    LINELEVEL, #6                          ; 0536
                                             B5  12 000FA          BNEQ    14$                                    ;
                                 6E          56  D1 000FC          CMPL    LASTLEVEL, BOUND_LEVEL                 ; 0537
                                             B0  12 000FF          BNEQ    14$                                    ;
                                       08    AE  9F 00101 20$:     PUSHAB  WILDKEY                                ; 0540
                                             5B  DD 00104          PUSHL   R11                                    ;
                                       1C    AE  9F 00106          PUSHAB  KEYDESC                                ;
                         0000V   CF          03  FB 00109          CALLS   #3, DSR$KEY_EQUAL                      ;
                                 09          50  E8 0010E          BLBS    R0, 21$                                ;
                     00000000'   EF          59  D1 00111          CMPL    R9, HELPINFO                           ; 0541
                                             97  19 00118          BLSS    14$                                    ;
                                 50    0C    AC  D0 0011A 21$:     MOVL    KEY, R0                                ; 0548
                                 51          6E  D0 0011E          MOVL    BOUND_LEVEL, R1                        ;
                     00000000'EF41           50  D0 00121          MOVL    R0, KEYNAMES[R1]                       ;
                 51  00000000'   EF    18    AE  C3 00129          SUBL3   KEYDESC+4, HELPINFO+12, R1             ; 0551
                                 51  00000000' EF  C0 00132          ADDL2   HELPINFO+8, R1                         ;
                                 60          51  B0 00139          MOVW    R1, (R0)                               ; 0550
             04  B0        18    BE          51  28 0013C          MOVC3   R1, @KEYDESC+4, @4(R0)                 ; 0552
                                             50  D4 00142          CLRL    R0                                     ; 0553
                                 06          5A  D1 00144          CMPL    LINELEVEL, #6                          ;
                                             02  12 00147          BNEQ    22$                                    ;
                                             50  D6 00149          INCL    R0                                     ;
00000000'  EF            01      01          50  F0 0014B 22$:     INSV    R0, #1, #1, HELPINFO+45                ;
                     00000000'   EF          59  D1 00154          CMPL    R9, HELPINFO+40                        ; 0555
                                             09  13 0015B          BEQL    23$                                    ;
                     00000000'   EF          59  D1 0015D          CMPL    R9, HELPINFO                           ; 0556
                                             1B  19 00164          BLSS    24$                                    ;
                     00000000'   EF          04  88 00166 23$:     BISB2   #4, HELPINFO+45                        ; 0559
                                             59  DD 0016D          PUSHL   R9                                     ; 0560
                         0000V   CF          01  FB 0016F          CALLS   #1, PRINT_KEYS                         ;
                                             59  DD 00174          PUSHL   R9                                     ; 0561
                                             5A  DD 00176          PUSHL   LINELEVEL                              ;
                         0000V   CF          02  FB 00178          CALLS   #2, PRINT_TEXT                         ;
                           1E    A8          02  90 0017D          MOVB    #2, 30(R8)                             ; 0562
                           20    AE    24    AE  9E 00181 24$:     MOVAB   NEXTKEY, NEXTDESC+4                    ; 0565
                                       1C    AE  9F 00186          PUSHAB  NEXTDESC                               ; 0571
                                       10    A8  9F 00189          PUSHAB  16(R8)                                 ;
                                       01    A9  9F 0018C          PUSHAB  1(R9)                                  ;
                         FE6C    CF          03  FB 0018F          CALLS   #3, DO_HELP                            ;
                                 7E          50  E9 00194          BLBC    STAT, 30$                              ; 0572
                     00000000'   EF          59  D1 00197          CMPL    R9, HELPINFO+40                        ; 0574
                                             10  12 0019E          BNEQ    25$                                    ;
```

```
                                                  01  DD 001A0          PUSHL   #1                                          ; 0581
                                            10    A8  9F 001A2          PUSHAB  16(R8)                                      ;
                                            01    A9  9F 001A5          PUSHAB  1(R9)                                       ;
                          0000V  CF               03  FB 001A8          CALLS   #3, PRINT_OTHERHELP                         ;
                                 65               50  E9 001AD          BLBC    STAT, 30$                                   ; 0582
                                 09         08    AE  E8 001B0 25$:     BLBS    WILDKEY, 26$                                ; 0584
                      00000000'  EF               59  D1 001B4          CMPL    R9, HELPINFO                                ;
                                                  55  19 001BB          BLSS    29$                                         ;
                                               FEED  31 001BD 26$:      BRW     13$                                         ; 0587
                                                  52  D4 001C0 27$:     CLRL    POSITION                                    ; 0612
                                 51         FF    A9  9E 001C2          MOVAB   -1(R9), SIZE                                ; 0613
                                                  50  D4 001C6          CLRL    R0                                          ; 0615
      53 00000000'  EF           51               52  EA 001C8          FFS     POSITION, SIZE, HELPINFO+44, FIRST          ;
                                                  02  12 001D1          BNEQ    28$                                         ;
                                                  50  D6 001D3          INCL    R0                                          ;
                                 50               50  D2 001D5 28$:     MCOML   R0, WILDPATH                                ; 0614
                   32 00000000'  EF               02  E0 001D8          BBS     #2, HELPINFO+45, 29$                        ; 0617
                      00000000'  EF               59  D1 001E0          CMPL    R9, HELPINFO                                ; 0618
                                                  29  18 001E7          BGEQ    29$                                         ;
                                 26               50  E8 001E9          BLBS    WILDPATH, 29$                               ; 0619
                            1E   A8               02  90 001EC          MOVB    #2, 30(R8)                                  ; 0626
              10    A8       0C   AE               06  28 001F0          MOVC3   #6, SAVERFA, 16(R8)                         ; 0627
                          0000V  CF               00  FB 001F6          CALLS   #0, READ                                    ; 0628
                          0000V  CF               00  FB 001FB          CALLS   #0, READ                                    ; 0629
                                 12               50  E9 00200          BLBC    STAT, 30$                                   ; 0631
                                                  7E  D4 00203          CLRL    -(SP)                                       ; 0633
                                            10    A8  9F 00205          PUSHAB  16(R8)                                      ;
                                                  59  DD 00208          PUSHL   R9                                          ;
                          0000V  CF               03  FB 0020A          CALLS   #3, PRINT_OTHERHELP                         ;
                                 03               50  E9 0020F          BLBC    STAT, 30$                                   ; 0635
                                 50               01  D0 00212 29$:     MOVL    #1, R0                                      ; 0641
                                                      04 00215 30$:     RET                                                 ;

; Routine Size:  534 bytes,    Routine Base:  CODE + 02EC
```

```
;    0643  1      %SBTTL 'check for key on line'
;    0644  1      ROUTINE is_key_on_line (key) =
;    0645  1
;    0646  1      !++
;    0647  1      ! FUNCTIONAL DESCRIPTION
;    0648  1      !       Scan the current line to see if it has a keyword on it.  The valid
;    0649  1      !       formats for a keyword line are "n key" and "/key" where n is a number
;    0650  1      !       from 1 to hlp$c_maxkeys, and key is a valid symbol (A-Z, a-z, $, _).
;    0651  1      !       It will return n, or hlp$c_maxkeys+1 if the qualifier form "/key" is
;    0652  1      !       found, or 0 if no key is found.
;    0653  1      !
;    0654  1      ! INPUTS
;    0655  1      !       key             address of descriptor of buffer for keyword
;    0656  1      !
;    0657  1      ! IMPLICIT INPUTS
;    0658  1      !       none
;    0659  1      !
;    0660  1      ! OUTPUTS
;    0661  1      !       none
;    0662  1      !
;    0663  1      ! IMPLICIT OUTPUTS
;    0664  1      !       none
;    0665  1      !
;    0666  1      ! SIDE EFFECTS
;    0667  1      !       none
;    0668  1      !
;    0669  1      ! RETURN CODES  (R0)
;    0670  1      !       level of key found
;    0671  1      !       hlp$c_maxkeys+1 if qualifier line
;    0672  1      !       0 if no key found
;    0673  1      !--
;    0674  1
```

```
;    0676  2          BEGIN
;    0677  2
;    0678  2          MAP
;    0679  2              key : REF bblock;
;    0680  2
;    0681  2          BIND
;    0682  2              line = helpinfo [hlp$l_bufdesc] : bblock;
;    0683  2
;    0684  2          LOCAL
;    0685  2              locdesc : bblock [dsc$c_s_bln],
;    0686  2              char : BYTE,
;    0687  2              level;
;    0688  2
;    0689  2          IF .line [dsc$w_length] EQL 0               ! If line is null,
;    0690  2          THEN
;    0691  2              RETURN 0;                               !  then can't be key on it
;    0692  2
;    0693  2          level = 0;                                  ! Initialize level
;    0694  2          CH$MOVE (dsc$c_s_bln, line, locdesc);       ! Initialize local descriptor
;    0695  2          char = CH$RCHAR (.locdesc [dsc$a_pointer]); ! Read first character
;    0696  2
;    0697  3          IF (.char LSSU %C'0'                        ! If not numeric
;    0698  2              OR .char GEQU %C'9') AND .char NEQ %C'/'        ! And not a qualifier line
;    0699  2          THEN
;    0700  2              RETURN 0                                !  then it's not a key line
;    0701  2          ELSE
;    0702  3              BEGIN                                   !  else it IS a key line
;    0703  3
;    0704  3          IF .char NEQ %C'/'                          ! If it's a normal keyword
;    0705  3          THEN
;    0706  4              BEGIN
;    0707  4
;    0708  4              LOCAL
;    0709  4                  ptr,
;    0710  4                  len;
;    0711  4
;    0712  4              ptr = .locdesc [dsc$a_pointer];     ! Get number ptr, len
;    0713  4              len = scan_word (locdesc);          !  and scan past it
;    0714  4
;    0715  4              WHILE .len GTR 0                    ! convert ascii to binary
;    0716  4                  AND CH$RCHAR (.ptr) GEQU %C'0' AND CH$RCHAR (.ptr) LEQU %C'9' DO
;    0717  5                  BEGIN
;    0718  5                  level = .level*10 + (CH$RCHAR_A (ptr) - %C'0');
;    0719  5                  len = .len - 1
;    0720  4                  END;
;    0721  4
;    0722  4              IF .level GTR hlp$c_maxkeys         ! If level is illegally high,
;    0723  4              THEN
;    0724  4                  RETURN 0;                       !  don't recognize it
;    0725  4
;    0726  4              skip_blanks (locdesc)               ! Skip to keyword
;    0727  4              END
;    0728  3          ELSE
;    0729  3              level = hlp$c_maxkeys + 1;          ! Specify qualifier
;    0730  3
;    0731  3          key [dsc$a_pointer] = .locdesc [dsc$a_pointer]; ! Remember start of word
;    0732  3          key [dsc$w_length] = MIN (scan_word (locdesc), hlp$c_maxrecsiz) ! Length of string
```

```
;   0733  2          END;
;   0734  2
;   0735  2      .level
;   0736  1      END;                                    ! of is_key_on_line


                                                  LINE=              HELPINFO+8


                                        00FC 00000 IS_KEY_ON_LINE:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7                  ; 0644
                    57 00000000'  EF  9E 00002            MOVAB   LINE, R7                                ;
                    5E            08  C2 00009            SUBL2   #8, SP                                  ;
                                  67  B5 0000C            TSTW    LINE                                    ; 0689
                                  4D  13 0000E            BEQL    5$                                      ;
                                  56  D4 00010            CLRL    LEVEL                                   ; 0693
          6E        67            08  28 00012            MOVC3   #8, LINE, LOCDESC                       ; 0694
                    50      04    BE  90 00016            MOVB    @LOCDESC+4, CHAR                        ; 0695
                    30            50  91 0001A            CMPB    CHAR, #48                               ; 0697
                                  05  1F 0001D            BLSSU   1$                                      ;
                    39            50  91 0001F            CMPB    CHAR, #57                               ; 0698
                                  05  1F 00022            BLSSU   2$                                      ;
                    2F            50  91 00024 1$:        CMPB    CHAR, #47                               ;
                                  34  12 00027            BNEQ    5$                                      ;
                    2F            50  91 00029 2$:        CMPB    CHAR, #47                               ; 0704
                                  3B  13 0002C            BEQL    7$                                      ;
                    52      04    AE  D0 0002E            MOVL    LOCDESC+4, PTR                          ; 0712
                                  5E  DD 00032            PUSHL   SP                                      ; 0713
              0000V CF            01  FB 00034            CALLS   #1, SCAN_WORD                           ;
                    53            50  D0 00039            MOVL    R0, LEN                                 ;
                                  1A  15 0003C 3$:        BLEQ    4$                                      ; 0715
                    30            62  91 0003E            CMPB    (PTR), #48                              ; 0716
                                  15  1F 00041            BLSSU   4$                                      ;
                    39            62  91 00043            CMPB    (PTR), #57                              ;
                                  10  1A 00046            BGTRU   4$                                      ;
          50        56            0A  C5 00048            MULL3   #10, LEVEL, R0                          ; 0718
                    51            82  9A 0004C            MOVZBL  (PTR)+, R1                              ;
                    56      D0 A140   9E 0004F            MOVAB   -48(R1)[R0], LEVEL                      ;
                                  53  D7 00054            DECL    LEN                                     ; 0719
                                  E4  11 00056            BRB     3$                                      ;
                    05            56  D1 00058 4$:        CMPL    LEVEL, #5                               ; 0722
                                  03  15 0005B            BLEQ    6$                                      ;
                                  50  D4 0005D 5$:        CLRL    R0                                      ; 0724
                                      04 0005F            RET                                             ;
                                  5E  DD 00060 6$:        PUSHL   SP                                      ; 0726
              0000V CF            01  FB 00062            CALLS   #1, SKIP_BLANKS                         ;
                                  03  11 00067            BRB     8$                                      ;
                    56            06  D0 00069 7$:        MOVL    #6, LEVEL                               ; 0729
                    52      04    AC  D0 0006C 8$:        MOVL    KEY, R2                                 ; 0731
                04  A2      04    AE  D0 00070            MOVL    LOCDESC+4, 4(R2)                        ;
                                  5E  DD 00075            PUSHL   SP                                      ; 0732
              0000V CF            01  FB 00077            CALLS   #1, SCAN_WORD                           ;
          00000100  8F            50  D1 0007C            CMPL    R0, #256                                ;
                                  05  15 00083            BLEQ    9$                                      ;
                    50    0100    8F  3C 00085            MOVZWL  #256, R0                                ;
                    62            50  B0 0008A 9$:        MOVW    R0, (R2)                                ;
```

```
                    50          56  D0 0008D        MOVL    LEVEL, R0                          ; 0736
                                    04 00090        RET                                        ;

; Routine Size:  145 bytes,    Routine Base:  CODE + 0502
```

```
;    0738  1     %SBTTL 'print key list'
;    0739  1     ROUTINE print_keys (level) : NOVALUE =
;    0740  1
;    0741  1     !++
;    0742  1     ! FUNCTIONAL DESCRIPTION
;    0743  1     !       A list of pointers to string descriptors for keys along the path
;    0744  1     !       to the current level (i.e., key1 descriptor, key2 descriptor, etc.)
;    0745  1     !       is stored in the helpinfo block.  This routine will print, with
;    0746  1     !       the correct indentation, all keys along that path.
;    0747  1     !
;    0748  1     ! INPUTS
;    0749  1     !       level           The current level (lowest key name to output)
;    0750  1     !
;    0751  1     ! IMPLICIT INPUTS
;    0752  1     !       none
;    0753  1     !
;    0754  1     ! OUTPUTS
;    0755  1     !       none
;    0756  1     !
;    0757  1     ! IMPLICIT OUTPUTS
;    0758  1     !       none
;    0759  1     !
;    0760  1     ! SIDE EFFECTS
;    0761  1     !       none
;    0762  1     !
;    0763  1     ! RETURN CODES
;    0764  1     !       none
;    0765  1     !--
;    0766  1
```

```
;   0768  2          BEGIN
;   0769  2
;   0770  2          BIND
;   0771  2              keynames = helpinfo [hlp$l_keynames] : VECTOR [, LONG];
;   0772  2
;   0773  2          INCR i FROM 1 TO .level DO
;   0774  2
;   0775  2              !+
;   0776  2              ! Print; with trailing CRLF unless this is lowest level key and
;   0777  2              ! it's a qualifier.
;   0778  2              !-
;   0779  2
; P 0780  2              $ds_printf ((IF .helpinfo [hlp$v_qual] AND .i EQL .level THEN $ascic ('!/!#* !AS') ELSE $ascic (
;   0781  2                          '!/!#* !AS!/')), .i*2, .keynames [.i - 1]);
;   0782  2
;   0783  1          END;


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                 53  41  21  20  2A  23  21  2F  21  09  00022 P.AAG:   .ASCII  <9>\!/!#* !AS\                          ;
          2F  21 53  41  21  20  2A  23  21  2F  21  0B  0002C P.AAH:   .ASCII  <11>\!/!#* !AS!/\                        ;

                                                                KEYNAMES=          HELPINFO+16
                                                                        .EXTRN  DS$PRINTF

                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                  0004 00000 PRINT_KEYS:
                                                                        .WORD   Save R2                                   ; 0739
                                                 52  D4 00002           CLRL    I                                         ; 0773
                                                 32  11 00004           BRB     4$                                        ;
                                  00000000'EF42  DD 00006 1$:           PUSHL   KEYNAMES-4[I]                             ; 0781
                 7E            52               01  78 0000D           ASHL    #1, I, -(SP)                              ;
                 0F 00000000'  EF               01  E1 00011           BBC     #1, HELPINFO+45, 2$                       ;
                          04   AC               52  D1 00019           CMPL    I, LEVEL                                  ;
                                                 09  12 0001D           BNEQ    2$                                        ;
                               50 00000000'     EF  9E 0001F           MOVAB   P.AAG, R0                                 ;
                                                 07  11 00026           BRB     3$                                        ;
                               50 00000000'     EF  9E 00028 2$:       MOVAB   P.AAH, R0                                 ;
                                                 50  DD 0002F 3$:       PUSHL   R0                                        ;
                    00000000G  9F               03  FB 00031           CALLS   #3, a#DS$PRINTF                           ;
                 C9            52        04     AC  F3 00038 4$:       AOBLEQ  LEVEL, I, 1$                              ;
                                                    04 0003D           RET                                               ; 0783

; Routine Size:  62 bytes,    Routine Base:  CODE + 0593
```

```
;    0785  1      %SBTTL 'print help text'
;    0786  1      ROUTINE print_text (level, indent) =
;    0787  1
;    0788  1      !++
;    0789  1      ! FUNCTIONAL DESCRIPTION
;    0790  1      !       Print out help text.  Scan from given RFA until another key
;    0791  1      !       line is found, printing each line.  A qualifier key line which
;    0792  1      !       is imbedded within the text for a level is printed out as text.
;    0793  1      !
;    0794  1      ! INPUTS
;    0795  1      !       level           The level for which help is wanted (this can be
;    0796  1      !                       1 to hlp$c_maxkeys; or hlp$c_maxkeys+1 for
;    0797  1      !                       qualifier help)
;    0798  1      !       indent          position of key in input list
;    0799  1      !
;    0800  1      ! IMPLICIT INPUTS
;    0801  1      !       none
;    0802  1      !
;    0803  1      ! OUTPUTS
;    0804  1      !
;    0805  1      ! IMPLICIT OUTPUTS
;    0806  1      !       none
;    0807  1      !
;    0808  1      ! SIDE EFFECTS
;    0809  1      !       none
;    0810  1      !
;    0811  1      ! RETURN CODES
;    0812  1      !       none
;    0813  1      !--
;    0814  1
```

```
;    0816  2          BEGIN
;    0817  2
;    0818  2          LOCAL
;    0819  2              stat,
;    0820  2              logtabs,                        ! Number of logical tabs to indent
;    0821  2              linelevel,                      ! Level of key record
;    0822  2              lastqual,                       ! Flag for qualifier help text
;    0823  2              keydesc : bblock [dsc$c_s_bln];         ! Descriptor for key
;    0824  2
;    0825  2          logtabs = .indent + (IF .level LEQ hlp$c_maxkeys THEN 1 ELSE 0);
;    0826  2
;    0827  2          !+
;    0828  2          ! Now print out lines until we find another qualifier.  Note that
;    0829  2          ! unless we are looking for qualifier help, or we are in ALLHELP
;    0830  2          ! mode, a qualifier key line counts as text!
;    0831  2          !-
;    0832  2
;    0833  2          lastqual = 1;                           ! Pre-set switch so it works
;    0834  2
;    0835  2          WHILE
;    0836  3              BEGIN
;    0837  3              stat = READ ();                     ! Read next record
;    0838  3
;    0839  3              IF NOT .stat AND .stat NEQ rms$_eof THEN RETURN .stat;
;    0840  3
;    0841  4              BEGIN
;    0842  4
;    0843  4              IF NOT .stat
;    0844  4              THEN
;    0845  4                  0                           ! If end of file, no more text
;    0846  4              ELSE
;    0847  5                  BEGIN
;    0848  5                  linelevel = is_key_on_line (keydesc);       ! Get help key
;    0849  6                  (.linelevel EQL 0                   ! continue if no key on line
;    0850  7                  OR (.level EQL hlp$c_maxkeys + 1 AND .lastqual)     ! or second qual in row
;    0851  7                  OR (.linelevel EQL hlp$c_maxkeys + 1        ! or qual line and not printing
;    0852  6                  AND .level LEQ hlp$c_maxkeys))      ! qualifier help
;    0853  5                  END
;    0854  5
;    0855  4              END
;    0856  3              END
;    0857  2          DO
;    0858  3              BEGIN
;    0859  3              lastqual = (IF .linelevel EQL hlp$c_maxkeys + 1 THEN 1 ELSE 0);
;    0860  4              $ds_printf ($ascic ('!/!#* !AS'), .logtabs*2, helpinfo [hlp$l_bufdesc])
;    0861  2              END;
;    0862  2
;    0863  2          print_crlf ();                          ! Force double space after text
;    0864  2          .stat
;    0865  1          END;                                    ! of print_text


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                 53  41  21  20  2A  23  21  2F  21  09  00038 P.AAI:  .ASCII  <9>\!/!#* !AS\                    ;
```

```
                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                  0FFC 00000 PRINT_TEXT:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0786
                          5E                08  C2 00002        SUBL2   #8, SP                                        ;
                                            56  D4 00005        CLRL    R6                                            ; 0825
                          05        04      AC  D1 00007        CMPL    LEVEL, #5                                     ;
                                            07  14 0000B        BGTR    1$                                            ;
                                            56  D6 0000D        INCL    R6                                            ;
                          50                01  D0 0000F        MOVL    #1, R0                                        ;
                                            02  11 00012        BRB     2$                                            ;
                                            50  D4 00014 1$:    CLRL    R0                                            ;
                          50        08      AC  C0 00016 2$:    ADDL2   INDENT, LOGTABS                               ;
                          52                01  D0 0001A        MOVL    #1, LASTQUAL                                  ; 0833
                 55       50                01  78 0001D        ASHL    #1, LOGTABS, R5                               ; 0860
                     0000V CF                00  FB 00021 3$:    CALLS   #0, READ                                      ; 0837
                          53                50  D0 00026        MOVL    R0, STAT                                      ;
                          0C                53  E8 00029        BLBS    STAT, 4$                                      ; 0839
                 0001827A 8F                53  D1 0002C        CMPL    STAT, #98938                                  ;
                                            46  12 00033        BNEQ    10$                                           ;
                          40                53  E9 00035        BLBC    STAT, 9$                                      ; 0843
                                            5E  DD 00038 4$:    PUSHL   SP                                            ; 0848
                     FEF2 CF                01  FB 0003A        CALLS   #1, IS_KEY_ON_LINE                            ;
                          54                50  D0 0003F        MOVL    R0, LINELEVEL                                 ;
                                            11  13 00042        BEQL    6$                                            ; 0849
                          06        04      AC  D1 00044        CMPL    LEVEL, #6                                     ; 0850
                                            03  12 00048        BNEQ    5$                                            ;
                          08                52  E8 0004A        BLBS    LASTQUAL, 6$                                  ;
                          06                54  D1 0004D 5$:    CMPL    LINELEVEL, #6                                 ; 0851
                                            26  12 00050        BNEQ    9$                                            ;
                          23                56  E9 00052        BLBC    R6, 9$                                        ; 0852
                          06                54  D1 00055 6$:    CMPL    LINELEVEL, #6                                 ; 0859
                                            05  12 00058        BNEQ    7$                                            ;
                          52                01  D0 0005A        MOVL    #1, LASTQUAL                                  ;
                                            02  11 0005D        BRB     8$                                            ;
                                            52  D4 0005F 7$:    CLRL    LASTQUAL                                      ;
                             00000000'      EF  9F 00061 8$:    PUSHAB  HELPINFO+8                                    ; 0860
                                            55  DD 00067        PUSHL   R5                                            ;
                             00000000'      EF  9F 00069        PUSHAB  P.AAI                                         ;
                 00000000G 9F               03  FB 0006F        CALLS   #3, @#DS$PRINTF                               ;
                                            A9  11 00076        BRB     3$                                            ;
                                       0000V 30  00078 9$:       BSBW    PRINT_CRLF                                    ; 0863
                          50                53  D0 0007B 10$:   MOVL    STAT, R0                                      ; 0865
                                                04 0007E        RET                                                   ;

; Routine Size:  127 bytes,    Routine Base:  CODE + 05D1
```

```
;    0867  1      %SBTTL 'List other keys'
;    0868  1      ROUTINE print_otherhelp (level, keyrfa, errflg) =
;    0869  1
;    0870  1      !++
;    0871  1      ! FUNCTIONAL DESCRIPTION
;    0872  1      !       When the search for a help key fails, or if the key is level 1, and
;    0873  1      !       is 'HELP', a list is printed of all 'other' help keys valid for
;    0874  1      !       that level.
;    0875  1      !
;    0876  1      ! INPUTS
;    0877  1      !       level           The level for which keys are to be listed
;    0878  1      !       keyrfa          The RFA to begin scan at (immediately following the
;    0879  1      !                       RFA for the next highest level)
;    0880  1      !       errflg          0 if 'no keys found' message should be typed
;    0881  1      !
;    0882  1      ! IMPLICIT INPUTS
;    0883  1      !       none
;    0884  1      !
;    0885  1      ! OUTPUTS
;    0886  1      !
;    0887  1      ! IMPLICIT OUTPUTS
;    0888  1      !       none
;    0889  1      !
;    0890  1      ! SIDE EFFECTS
;    0891  1      !       Updates RAB RFA past keys
;    0892  1      !
;    0893  1      ! RETURN CODES
;    0894  1      !       none
;    0895  1      !--
;    0896  1
```

```
;    0898  2          BEGIN
;    0899  2
;    0900  2          BIND
;    0901  2              rab = .helpinfo [hlp$l_rab] : bblock,
;    0902  2              fmt = $ascic ('!/  !AD'),                  ! Format output line
;    0903  2              otherinfo = $ascic ('!/  Additional information available:!/');
;    0904  2
;    0905  2          LOCAL
;    0906  2              qualine,                                   ! Flag for qualifier line
;    0907  2              no_qual_yet,                               ! Flag for qual. line found
;    0908  2              lastlevel,                                 ! Last key level found
;    0909  2              linelevel,                                 ! Level of current record
;    0910  2              first,                                     ! First-time flag
;    0911  2              linecol,                                   ! Current column of line
;    0912  2              ptr,                                       ! Pointer to output buffer
;    0913  2              keydesc : bblock [dsc$c_s_bln],            ! Key descriptor
;    0914  2              outbuf : VECTOR [hlp$c_maxrecsiz, BYTE];       ! Output buffer
;    0915  2
```

```
;   0917  2     %SBTTL 'PrintHeader'
;   0918  2         ROUTINE printheader (err, lev) : jsb_2 NOVALUE =
;   0919  3             BEGIN
;   0920  3
;   0921  3             BIND
;   0922  3                 nodocmsg = $ascic ('!/  Sorry, no documentation on !AS !AS !AS !AS !AS!/'),
;   0923  3                 keylist = .helpinfo [hlp$l_keylist] : VECTOR [, LONG];
;   0924  3
;   0925  3             IF NOT .err
;   0926  3             THEN
;   0927  4                 BEGIN
;   0928  4                 helpinfo [hlp$v_foundhelp] = 1;      ! Say we found something.
;   0929  4                 print_keys (.lev - 1);               ! Print keys to next higher
;   0930  5                 $ds_printf (nodocmsg, keylist [0], keylist [2], keylist [4], keylist [6], keylist [8])
;   0931  5                                                      ! Print out "no help" msg
;   0932  3                 END;
;   0933  3
;   0934  4             $ds_printf (otherinfo)
;   0935  2             END;


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                44  41  21  20  20  2F  21  07  00042 P.AAJ:  .ASCII  <7>\!/  !AD\                                ;
6C  61  6E  6F  69  74  69  64  64  41  20  20  2F  21  27  0004A P.AAK:  .ASCII  \'!/  Additional information available:!/\  ;
76  61  20  6E  6F  69  74  61  6D  72  6F  66  6E  69  20  00059                                                                ;
                    2F  21  3A  65  6C  62  61  6C  69  61  00068                                                                ;
20  6F  6E  20  2C  79  72  72  6F  53  20  20  2F  21  34  00072 P.AAL:  .ASCII  \4!/  Sorry, no documentation on !AS !AS \  ;
6F  20  6E  6F  69  74  61  74  6E  65  6D  75  63  6F  64  00081                                                                ;
                    20  53  41  21  20  53  41  21  20  6E  00090                                                                ;
        2F  21  53  41  21  20  53  41  21  20  53  41  21  0009A         .ASCII  \!AS !AS !AS!/\                                ;

                                                                FMT=                P.AAJ
                                                                OTHERINFO=          P.AAK
                                                                NODOCMSG=           P.AAL


                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                    52  DD 00000 PRINTHEADER:
                                                                        PUSHL   R2                                          ; 0918
                                52 00000000'        EF  D0 00002        MOVL    HELPINFO+36, R2                             ; 0923
                                2A                  50  E8 00009        BLBS    ERR, 1$                                     ; 0925
                    00000000'   EF                  04  88 0000C        BISB2   #4, HELPINFO+45                             ; 0928
                                            FF      A1  9F 00013        PUSHAB  -1(LEV)                                     ; 0929
                        FF28    CF                  01  FB 00016        CALLS   #1, PRINT_KEYS                              ;
                                            20      A2  9F 0001B        PUSHAB  32(R2)                                      ; 0930
                                            18      A2  9F 0001E        PUSHAB  24(R2)                                      ;
                                            10      A2  9F 00021        PUSHAB  16(R2)                                      ;
                                            08      A2  9F 00024        PUSHAB  8(R2)                                       ;
                                                    52  DD 00027        PUSHL   R2                                          ;
                                    00000000'       EF  9F 00029        PUSHAB  NODOCMSG                                    ;
                    00000000G   9F                  06  FB 0002F        CALLS   #6, a#DS$PRINTF                             ;
                                    00000000'       EF  9F 00036 1$:    PUSHAB  OTHERINFO                                   ; 0934
                    00000000G   9F                  01  FB 0003C        CALLS   #1, a#DS$PRINTF                             ;
                                                    04  BA 00043        POPR    #^M<R2>                                     ; 0935
```

```
                                        05 00045        RSB                                             ;

; Routine Size:  70 bytes,    Routine Base:  CODE + 0650
```

```
;    0937  2      %SBTTL 'print_otherhelp main code'
;    0938  2
;    0939  2          IF .level GEQ .helpinfo [hlp$l_allhelp] OR .helpinfo [hlp$v_qual] THEN RETURN 1;
;    0940  2
;    0941  2          first = 1;                                  ! Set first-time flag
;    0942  2          lastlevel = .level - 1;                     ! Init last key found
;    0943  2          no_qual_yet = 1;                            ! Haven't found qualifier yet
;    0944  2          ptr = CH$PTR (outbuf);
;    0945  2          linecol = 0;                                ! Set column pointer
;    0946  2          CH$MOVE (rfa$c_length, .keyrfa, rab [rab$w_rfa]);   ! Initialize local RFA
;    0947  2          rab [rab$b_rac] = rab$c_rfa;                ! Set random access
;    0948  2
;    0949  2          WHILE
;    0950  3              BEGIN
;    0951  3
;    0952  3              LOCAL
;    0953  3                  stat;
;    0954  3
;    0955  3              stat = READ ();                         ! Read next record
;    0956  3              rab [rab$b_rac] = rab$c_seq;            ! Set back to sequential
;    0957  3
;    0958  3              IF NOT .stat AND .stat NEQ rms$_eof THEN RETURN .stat;
;    0959  3
;    0960  4              BEGIN
;    0961  4
;    0962  4              IF NOT .stat
;    0963  4              THEN
;    0964  5                  (0)
;    0965  4              ELSE
;    0966  5                  BEGIN
;    0967  5                  linelevel = is_key_on_line (keydesc);
;    0968  5
;    0969  5                  IF (.linelevel GTR 0) AND (.linelevel LEQ hlp$c_maxkeys) THEN lastlevel = .linelevel;
;    0970  5
;    0971  5                  qualine = (IF (.linelevel LEQ hlp$c_maxkeys) OR (.lastlevel NEQ .level) THEN 0 ELSE 1);
;    0972  6                  ((.linelevel GEQ .level) OR (.linelevel EQL 0))
;    0973  5                  END
;    0974  5
;    0975  4              END
;    0976  3              END
;    0977  2          DO
;    0978  2
;    0979  2              IF (.linelevel EQL .level) OR .qualine  ! Type keys & qualifiers
;    0980  2              THEN
;    0981  3                  BEGIN
;    0982  3
;    0983  3                  LOCAL
;    0984  3                      length,
;    0985  3                      reallen,
;    0986  3                      outlen;
;    0987  3
;    0988  3                  reallen = .helpinfo [hlp$l_bufdesc] - (.keydesc [dsc$a_pointer] - .(helpinfo [hlp$l_bufdesc] + 4));
;    0989  3                                                      ! Length of rest of record
;    0990  3                  !+
;    0991  3                  ! At this point, the following are true:
;    0992  3                  !   RealLen = the length of the key word or the qualifier to be printed
;    0993  3                  !   LineCol = Current column in the line (starting at 0)
```

```
;    0994  3                !-
;    0995  3
;    0996  3                !+
;    0997  3                ! Now set Length to the length of the key word plus at least one space
;    0998  3                ! and make the Length modulo 15. That is, if the RealLen of the key word
;    0999  3                ! is 8, the Length will be 15 (15 mod 15 = 0). If the RealLen of the key
;    1000  3                ! word is 15, the Length will be 30 (30 mod 15 = 0). In this second
;    1001  3                ! example, the one space required after the key word makes it 16 plus the
;    1002  3                ! additional spaces (for which 14 wuld be needed to bring it to the next
;    1003  3                ! field).
;    1004  3                !-
;    1005  3
;    1006  3                length = ((.reallen + 15) / 15) * 15;
;    1007  3
;    1008  3                outlen = CH$DIFF (.ptr, CH$PTR (outbuf));
;    1009  3
;    1010  4                IF ((.outlen + .length) GTR 78) OR (.no_qual_yet AND .qualine)
;    1011  3                THEN
;    1012  4                    BEGIN
;    1013  4
;    1014  4                    IF .qualine THEN no_qual_yet = 0;
;    1015  4
;    1016  4                    IF .first THEN printheader (.errflg, .level);
;    1017  4
;    1018  4                    first = 0;
;    1019  4                    $ds_printf (fmt, .outlen, outbuf);
;    1020  4                    ptr = CH$PTR (outbuf);
;    1021  4                    linecol = 0
;    1022  3                    END;
;    1023  3
;    1024  3                ptr = CH$COPY (.reallen, .keydesc [dsc$a_pointer], %C' ', .length, .ptr);
;    1025  3                                                ! Copy key into output descriptor
;    1026  3                linecol = .linecol + .length    ! Update column count
;    1027  2                END;
;    1028  2
;    1029  2            IF .linecol GTR 0                   ! If we have a partial line left,
;    1030  2            THEN
;    1031  3                BEGIN
;    1032  3
;    1033  3                IF .first THEN printheader (.errflg, .level);
;    1034  3
;    1035  4                $ds_printf ($ascic ('!/  !AD!/'), CH$DIFF (.ptr, CH$PTR (outbuf)), outbuf)
;    1036  2                END;
;    1037  2
;    1038  2            1
;    1039  1            END;                                ! of print_otherhelp


                                                            .PSECT  DATA,NOWRT,NOEXE,  SHR,2

            2F  21  44  41  21  20  20  2F  21  09  000A7 P.AAM:  .ASCII  <9>\!/  !AD!/\                    ;


                                                            .PSECT  CODE,NOWRT,  SHR,2
```

```
                                                  0FFC 00000 PRINT_OTHERHELP:
                                                                  .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0868
                          5E      FEE8   CE  9E 00002             MOVAB   -280(SP), SP                                  ;
                          56 00000000'   EF  D0 00007             MOVL    HELPINFO+4, R6                                ; 0901
                          57        04   AC  D0 0000E             MOVL    LEVEL, R7                                     ; 0939
                00000000'  EF            57  D1 00012             CMPL    R7, HELPINFO                                  ;
                                         03  19 00019             BLSS    2$                                            ;
                                       011A  31 0001B 1$:         BRW     18$                                           ;
            F5 00000000'  EF             01  E0 0001E 2$:         BBS     #1, HELPINFO+45, 1$                           ;
                     04   AE             01  D0 00026             MOVL    #1, FIRST                                     ; 0941
                     08   AE        FF   A7  9E 0002A             MOVAB   -1(R7), LASTLEVEL                             ; 0942
                     0C   AE             01  D0 0002F             MOVL    #1, NO_QUAL_YET                               ; 0943
                          5A        10   AE  9E 00033             MOVAB   OUTBUF, PTR                                   ; 0944
                                         6E  D4 00037             CLRL    LINECOL                                       ; 0945
        10   A6       08   BC             06  28 00039             MOVC3   #6, @KEYRFA, 16(R6)                           ; 0946
                     1E   A6             02  90 0003F             MOVB    #2, 30(R6)                                    ; 0947
                  0000V   CF             00  FB 00043 3$:         CALLS   #0, READ                                      ; 0955
                                    1E   A6  94 00048             CLRB    30(R6)                                        ; 0956
                          10             50  E8 0004B             BLBS    STAT, 5$                                      ; 0958
                0001827A  8F             50  D1 0004E             CMPL    STAT, #98938                                  ;
                                         01  13 00055             BEQL    4$                                            ;
                                             04 00057             RET                                                   ;
                          03             50  E8 00058 4$:         BLBS    STAT, 5$                                      ; 0962
                                       00B0  31 0005B             BRW     16$                                           ;
                                    F8   AD  9F 0005E 5$:         PUSHAB  KEYDESC                                       ; 0967
                   FE06   CF             01  FB 00061             CALLS   #1, IS_KEY_ON_LINE                            ;
                          59             50  D0 00066             MOVL    R0, LINELEVEL                                 ;
                                         09  15 00069             BLEQ    6$                                            ; 0969
                          05             59  D1 0006B             CMPL    LINELEVEL, #5                                 ;
                                         04  14 0006E             BGTR    6$                                            ;
                     08   AE             59  D0 00070             MOVL    LINELEVEL, LASTLEVEL                          ;
                          05             59  D1 00074 6$:         CMPL    LINELEVEL, #5                                 ; 0971
                                         06  15 00077             BLEQ    7$                                            ;
                          57        08   AE  D1 00079             CMPL    LASTLEVEL, R7                                 ;
                                         04  13 0007D             BEQL    8$                                            ;
                                         5B  D4 0007F 7$:         CLRL    QUALINE                                       ;
                                         03  11 00081             BRB     9$                                            ;
                          5B             01  D0 00083 8$:         MOVL    #1, QUALINE                                   ;
                          57             59  D1 00086 9$:         CMPL    LINELEVEL, R7                                 ; 0972
                                         04  18 00089             BGEQ    10$                                           ;
                                         59  D5 0008B             TSTL    LINELEVEL                                     ;
                                         7F  12 0008D             BNEQ    16$                                           ;
                          57             59  D1 0008F 10$:        CMPL    LINELEVEL, R7                                 ; 0979
                                         03  13 00092             BEQL    11$                                           ;
                          AC             5B  E9 00094             BLBC    QUALINE, 3$                                   ;
            50 00000000'  EF        FC   AD  C3 00097 11$:        SUBL3   KEYDESC+4, HELPINFO+12, R0                    ; 0988
            53            50 00000000'   EF  C1 000A0             ADDL3   HELPINFO+8, R0, REALLEN                       ;
                          50        0F   A3  9E 000A8             MOVAB   15(R3), R0                                    ; 1006
                          50             0F  C6 000AC             DIVL2   #15, R0                                       ;
            58            50             0F  C5 000AF             MULL3   #15, R0, LENGTH                               ;
                          50        10   AE  9E 000B3             MOVAB   OUTBUF, R0                                    ; 1008
            52            5A             50  C3 000B7             SUBL3   R0, PTR, OUTLEN                               ;
            50            52             58  C1 000BB             ADDL3   LENGTH, OUTLEN, R0                            ; 1010
                0000004E  8F             50  D1 000BF             CMPL    R0, #78                                       ;
                                         07  14 000C6             BGTR    12$                                           ;
                          32        0C   AE  E9 000C8             BLBC    NO_QUAL_YET, 15$                              ;
                          2F             5B  E9 000CC             BLBC    QUALINE, 15$                                  ;
```

```
                                  03          5B  E9 000CF 12$:    BLBC    QUALINE, 13$                                  ; 1014
                                        0C    AE  D4 000D2         CLRL    NO_QUAL_YET                                   ;
                                  0A    04    AE  E9 000D5 13$:    BLBC    FIRST, 14$                                    ; 1016
                                  51          57  D0 000D9         MOVL    R7, R1                                        ;
                                  50    0C    AC  D0 000DC         MOVL    ERRFLG, R0                                    ;
                                            FED7  30 000E0         BSBW    PRINTHEADER                                   ;
                                        04    AE  D4 000E3 14$:    CLRL    FIRST                                         ; 1018
                                        10    AE  9F 000E6         PUSHAB  OUTBUF                                        ; 1019
                                              52  DD 000E9         PUSHL   OUTLEN                                        ;
                                 00000000'    EF  9F 000EB         PUSHAB  FMT                                           ;
                     00000000G  9F            03  FB 000F1         CALLS   #3, a#DS$PRINTF                               ;
                                  5A    10    AE  9E 000F8         MOVAB   OUTBUF, PTR                                   ; 1020
                                              6E  D4 000FC         CLRL    LINECOL                                       ; 1021
        58          20       FC   BD          53  2C 000FE 15$:    MOVC5   REALLEN, aKEYDESC+4, #32, LENGTH, (PTR)       ; 1024
                                              6A     00104                                                              ;
                                  5A          53  D0 00105         MOVL    R3, PTR                                       ;
                                  6E          58  C0 00108         ADDL2   LENGTH, LINECOL                               ; 1026
                                            FF35  31 0010B         BRW     3$                                            ;
                                              6E  D5 0010E 16$:    TSTL    LINECOL                                       ; 1029
                                              26  15 00110         BLEQ    18$                                           ;
                                  0A    04    AE  E9 00112         BLBC    FIRST, 17$                                    ; 1033
                                  51          57  D0 00116         MOVL    R7, R1                                        ;
                                  50    0C    AC  D0 00119         MOVL    ERRFLG, R0                                    ;
                                            FE9A  30 0011D         BSBW    PRINTHEADER                                   ;
                                        10    AE  9F 00120 17$:    PUSHAB  OUTBUF                                        ; 1035
                                  50    14    AE  9E 00123         MOVAB   OUTBUF, R0                                    ;
                     7E           5A          50  C3 00127         SUBL3   R0, PTR, -(SP)                                ;
                                 00000000'    EF  9F 0012B         PUSHAB  P.AAM                                         ;
                     00000000G  9F            03  FB 00131         CALLS   #3, a#DS$PRINTF                               ;
                                  50          01  D0 00138 18$:    MOVL    #1, R0                                        ; 1039
                                                  04 0013B         RET                                                   ;

; Routine Size:  316 bytes,    Routine Base:  CODE + 0696
```

```
;    1041  1      %SBTTL 'Read random record'
;    1042  1      ROUTINE READ =
;    1043  1
;    1044  1      !++
;    1045  1      ! FUNCTIONAL DESCRIPTION
;    1046  1      !       Read a logical record from the current help file
;    1047  1      !
;    1048  1      ! INPUTS
;    1049  1      !
;    1050  1      ! IMPLICIT INPUTS
;    1051  1      !       helpinfo        HELP module storage block
;    1052  1      !       RAB
;    1053  1      !
;    1054  1      ! OUTPUTS
;    1055  1      !       none
;    1056  1      !
;    1057  1      ! IMPLICIT OUTPUTS
;    1058  1      !       helpinfo [hlp$l_bufdesc] is set to describe the record read.
;    1059  1      !
;    1060  1      ! SIDE EFFECTS
;    1061  1      !       none
;    1062  1      !
;    1063  1      ! RETURN CODES  (R0)
;    1064  1      !       SS$_NORMAL
;    1065  1      !       any RMS error code
;    1066  1      !--
;    1067  1
```

```
;    1069  2          BEGIN
;    1070  2
;    1071  2          LOCAL
;    1072  2              len,
;    1073  2              ptr,
;    1074  2              stat;
;    1075  2
;    1076  2          BIND
;    1077  2              rab = .helpinfo [hlp$l_rab] : bblock,   ! Map to RAB
;    1078  2              fab = .rab [rab$l_fab] : bblock;        ! Map to FAB
;    1079  2
;    1080  2          stat = $get (rab = rab);                    ! Read a record
;    1081  2          rab [rab$b_rac] = rab$c_seq;                ! Restore sequential access
;    1082  2          len = .rab [rab$w_rsz];                     ! Get length
;    1083  2          ptr = .rab [rab$l_rbf];                     !  and address
;    1084  2              IF      .Ultrix_flags<ultrix_v_mode,1>
;    1085  2              THEN
;    1086  2              ELSE
;    1087  3                 BEGIN
;    1088  3          IF .fab [fab$b_rat] EQL fab$m_ftn           ! If fortran format
;    1089  3              AND .len GTR 0                          !  and not null record
;    1090  3          THEN
;    1091  4              BEGIN
;    1092  4              len = .len - 1;                         ! Lose the CR byte
;    1093  4              ptr = .ptr + 1
;    1094  3              END;
;    1095  3
;    1096  3          IF .fab [fab$b_rat] EQL 0                   ! If no attribute
;    1097  3              AND .len GEQ 2                          ! .. and there are two chars                    [06]
;    1098  3              AND .(.ptr + .len - 2)<0, 16> EQL %X'0a0d'      ! .. and last chars are CRLF
;    1099  3          THEN
;    1100  3              len = .len - 2;                         ! Slice off CRLF
;    1101  2              END;
;    1102  2          helpinfo [hlp$l_bufdesc] = .len;            ! Length read
;    1103  2          helpinfo [hlp$l_bufdesc] + 4 = .ptr;
;    1104  2
;    1105  2          IF .stat EQL rms$_rfa THEN stat = rms$_eof; ! Fake error type if bad RFA
;    1106  2
;    1107  2          .stat
;    1108  1          END;                                        ! of read


                                                        .EXTRN  SYS$GET

                                        000C 00000 READ:    .WORD   Save R2,R3                          ; 1042
                          52 00000000'  EF   D0 00002       MOVL    HELPINFO+4, R2                      ; 1077
                          53       3C   A2   D0 00009       MOVL    60(R2), R3                          ; 1078
                                        52   DD 0000D       PUSHL   R2                                  ; 1080
                00000000G 00            01   FB 0000F       CALLS   #1, SYS$GET                         ;
                                     1E   A2   94 00016       CLRB    30(R2)                              ; 1081
                          51         22   A2   3C 00019       MOVZWL  34(R2), LEN                         ; 1082
                          52         28   A2   D0 0001D       MOVL    40(R2), PTR                         ; 1083
                          26 00000000G    EF   E8 00021       BLBS    ULTRIX_FLAGS, 2$                    ; 1084
                          01         1E   A3   91 00028       CMPB    30(R3), #1                          ; 1088
                                          08   12 0002C       BNEQ    1$                                  ;
                                          51   D5 0002E       TSTL    LEN                                 ; 1089
```

```
                                        04  15 00030        BLEQ    1$                                  ;
                                        51  D7 00032        DECL    LEN                                 ; 1092
                                        52  D6 00034        INCL    PTR                                 ; 1093
                                  1E    A3  95 00036 1$:    TSTB    30(R3)                              ; 1096
                                        13  12 00039        BNEQ    2$                                  ;
                        02              51  D1 0003B        CMPL    LEN, #2                             ; 1097
                                        0E  19 0003E        BLSS    2$                                  ;
                                  FE A142   9F 00040        PUSHAB  -2(LEN)[PTR]                        ; 1098
                  0A0D  8F              9E  B1 00044        CMPW    @(SP)+, #2573                       ;
                                        03  12 00049        BNEQ    2$                                  ;
                        51              02  C2 0004B        SUBL2   #2, LEN                             ; 1100
            00000000'   EF              51  7D 0004E 2$:    MOVQ    LEN, HELPINFO+8                     ; 1102
            0001865C    8F              50  D1 00055        CMPL    STAT, #99932                        ; 1105
                                        07  12 0005C        BNEQ    3$                                  ;
                        50 0001827A     8F  D0 0005E        MOVL    #98938, STAT                        ;
                                            04 00065 3$:    RET                                         ; 1108

; Routine Size:  102 bytes,    Routine Base:  CODE + 07D2
```

```
;    1110  1      %SBTTL 'compare keys'
;    1111  1
;    1112  1      GLOBAL ROUTINE dsr$key_equal (key1, key2, wildcompare) =
;    1113  1
;    1114  1      !++
;    1115  1      ! FUNCTIONAL DESCRIPTION
;    1116  1      !       Using wildcard characters ('*' and '%') compare two strings.
;    1117  1      !
;    1118  1      ! INPUTS
;    1119  1      !       key1            The non-wildcard match string
;    1120  1      !       key2            The pattern string (possibly wildcarded)
;    1121  1      !       wildcompare     Address of flag to set if wildcarded compare
;    1122  1      !
;    1123  1      ! IMPLICIT INPUTS
;    1124  1      !       none
;    1125  1      !
;    1126  1      ! OUTPUTS
;    1127  1      !       none
;    1128  1      !
;    1129  1      ! IMPLICIT OUTPUTS
;    1130  1      !       none
;    1131  1      !
;    1132  1      ! SIDE EFFECTS
;    1133  1      !       none
;    1134  1      !
;    1135  1      ! RETURN CODES  (R0)
;    1136  1      !       0       strings do not match
;    1137  1      !       1       strings match
;    1138  1      !--
;    1139  1
```

```
;    1141  2          BEGIN
;    1142  2
;    1143  2          MAP
;    1144  2              key1 : REF bblock,
;    1145  2              key2 : REF bblock;
;    1146  2
;    1147  2          LABEL
;    1148  2              loop;
;    1149  2
;    1150  2          LOCAL
;    1151  2              wild,
;    1152  2              len1,
;    1153  2              len2,
;    1154  2              ptr1,
;    1155  2              ptr2,
;    1156  2              savlen1,
;    1157  2              savlen2,
;    1158  2              savptr1,
;    1159  2              savptr2;
;    1160  2
;    1161  2          .wildcompare = wild = 0;                    ! Init wild flag
;    1162  2          savlen1 = 0;                                ! Use savelen1 as 'flag' to see if saved
;    1163  2          len1 = .key1 [dsc$w_length];
;    1164  2          len2 = .key2 [dsc$w_length];
;    1165  2          ptr1 = .key1 [dsc$a_pointer];
;    1166  2          ptr2 = .key2 [dsc$a_pointer];
;    1167  2
;    1168  2          WHILE 1 DO
;    1169  2      loop :
;    1170  3              BEGIN
;    1171  3
;    1172  3              LOCAL
;    1173  3                  char : BYTE;
;    1174  3
;    1175  3              len2 = .len2 - 1;
;    1176  3
;    1177  3              IF .len2 LSS 0
;    1178  3              THEN
;    1179  3
;    1180  3                  IF .len1 EQL 0
;    1181  3                  THEN
;    1182  3                      RETURN 1
;    1183  3                  ELSE
;    1184  4                      BEGIN
;    1185  4
;    1186  4                      IF (savlen1 = .savlen1 - 1) LSS 0       ! Partial match
;    1187  4                      THEN
;    1188  4
;    1189  4                          IF .wild
;    1190  4                          THEN
;    1191  4                              RETURN 0                ! Don't take partial, if wild
;    1192  4                          ELSE
;    1193  5                              RETURN (.wildcompare = 1)       ! Partial OK, elsewise
;    1194  4                      ELSE
;    1195  4
;    1196  5                          BEGIN                       ! Restore saved valued--backup
;    1197  5                          ptr1 = (savptr1 = .savptr1 + 1);    ! Advance to next char.
```

```
;    1198  5                      ptr2 = .savptr2;
;    1199  5                      len1 = .savlen1;
;    1200  5                      len2 = .savlen2
;    1201  4                      END;
;    1202  4
;    1203  4                  LEAVE loop                          ! Proceed to next iteration of loop
;    1204  3                  END;
;    1205  3
;    1206  3              char = CH$RCHAR_A (ptr2);               ! Get next pattern character
;    1207  3              IF      .Ultrix_flags<ultrix_v_mode,1>
;    1208  3              THEN
;    1209  4                BEGIN
;    1210  5                char = .char - (IF .char GEQU %C'a' AND .char LEQU %C'z'      ! convert to upper case
;    1211  4                     THEN %C'a' - %C'A' ELSE 0);
;    1212  3                 END;
;    1213  3              IF .char EQL %C'*'
;    1214  3              THEN
;    1215  4                  BEGIN                               ! If wildcard char, save descriptors for later backup.
;    1216  4                  wild = 1;                           ! Set wildcard flag
;    1217  4
;    1218  4                  IF .len2 EQL 0 THEN RETURN (.wildcompare = 1);      ! '*' at end matches all
;    1219  4
;    1220  4                  savptr1 = .ptr1;
;    1221  4                  savptr2 = .ptr2;
;    1222  4                  savlen1 = .len1;
;    1223  4                  savlen2 = .len2
;    1224  4                  END
;    1225  3              ELSE
;    1226  3
;    1227  3                  IF
;    1228  4                      BEGIN
;    1229  4                      len1 = .len1 - 1;
;    1230  5                      BEGIN
;    1231  5
;    1232  5                      LOCAL
;    1233  5                          ch : BYTE;
;    1234  5
;    1235  5                      ch = CH$RCHAR_A (ptr1);
;    1236  7                      (.ch - (IF .ch GEQU %C'a' AND .ch LEQU %C'z'    ! convert to upper case
;    1237  6                      THEN %C'a' - %C'A' ELSE 0))
;    1238  5                      END
;    1239  4                      END
;    1240  3                      NEQ .char                       ! If not a match
;    1241  3                  THEN
;    1242  3
;    1243  3                      IF .char NEQ %C'%'              ! and not a character wildcard
;    1244  3                      THEN
;    1245  3
;    1246  3                          IF (savlen1 = .savlen1 - 1) LSS 0
;    1247  3                          THEN
;    1248  3                              RETURN 0
;    1249  3                          ELSE
;    1250  4                              BEGIN                   ! Restore values
;    1251  4                              ptr1 = (savptr1 = .savptr1 + 1);        ! Advance to next char
;    1252  4                              ptr2 = .savptr2;
;    1253  4                              len1 = .savlen1;
;    1254  4                              len2 = .savlen2
```

```
;   1255  4                    END
;   1256  4
;   1257  3                ELSE
;   1258  3                    .wildcompare = 1
;   1259  3
;   1260  2            END;                                  ! loop
;   1261  2
;   1262  2        0                                         ! If we leave loop, no good.
;   1263  1        END;                                      ! of Dsr$Key_Equal



                                        0FFC 00000          .ENTRY  DSR$KEY_EQUAL, Save R2,R3,R4,R5,R6,R7,R8,-  ; 1112
                                                                    R9,R10,R11                                  ;
                        5E          04  C2 00002            SUBL2   #4, SP                                      ;
                                    5B  D4 00005            CLRL    WILD                                        ; 1161
                              0C    BC  D4 00007            CLRL    @WILDCOMPARE                                ;
                                    53  D4 0000A            CLRL    SAVLEN1                                     ; 1162
                        51    04    AC  D0 0000C            MOVL    KEY1, R1                                    ; 1163
                        55          61  3C 00010            MOVZWL  (R1), LEN1                                  ;
                        50    08    AC  D0 00013            MOVL    KEY2, R0                                    ; 1164
                        54          60  3C 00017            MOVZWL  (R0), LEN2                                  ;
                        58    04    A1  D0 0001A            MOVL    4(R1), PTR1                                 ; 1165
                        57    04    A0  D0 0001E            MOVL    4(R0), PTR2                                 ; 1166
                        0D          54  F4 00022 1$:        SOBGEQ  LEN2, 2$                                    ; 1175
                                    55  D5 00025            TSTL    LEN1                                        ; 1180
                                    39  13 00027            BEQL    7$                                          ;
                        79          53  F4 00029            SOBGEQ  SAVLEN1, 13$                                ; 1186
                        2F          5B  E9 0002C            BLBC    WILD, 6$                                    ; 1189
                                  0089  31 0002F            BRW     15$                                         ; 1193
                        52          87  90 00032 2$:        MOVB    (PTR2)+, CHAR                               ; 1206
                        16 00000000G  EF  E9 00035          BLBC    ULTRIX_FLAGS, 5$                            ; 1207
                  61    8F          52  91 0003C            CMPB    CHAR, #97                                   ; 1210
                                    0B  1F 00040            BLSSU   3$                                          ;
                  7A    8F          52  91 00042            CMPB    CHAR, #122                                  ;
                                    05  1A 00046            BGTRU   3$                                          ;
                        50          20  D0 00048            MOVL    #32, R0                                     ; 1211
                                    02  11 0004B            BRB     4$                                          ;
                                    50  D4 0004D 3$:        CLRL    R0                                          ; 1210
                        52          50  82 0004F 4$:        SUBB2   R0, CHAR                                    ;
                        2A          52  91 00052 5$:        CMPB    CHAR, #42                                   ; 1213
                                    1D  12 00055            BNEQ    10$                                         ;
                        5B          01  D0 00057            MOVL    #1, WILD                                    ; 1216
                                    54  D5 0005A            TSTL    LEN2                                        ; 1218
                                    08  12 0005C            BNEQ    8$                                          ;
                  0C    BC          01  D0 0005E 6$:        MOVL    #1, @WILDCOMPARE                            ;
                        50          01  D0 00062 7$:        MOVL    #1, R0                                      ;
                                        04 00065            RET                                                 ;
                        56          58  D0 00066 8$:        MOVL    PTR1, SAVPTR1                               ; 1220
                        59          57  D0 00069            MOVL    PTR2, SAVPTR2                               ; 1221
                        53          55  D0 0006C            MOVL    LEN1, SAVLEN1                               ; 1222
                        5A          54  D0 0006F            MOVL    LEN2, SAVLEN2                               ; 1223
                                    AE  11 00072 9$:        BRB     1$                                          ;
                                    55  D7 00074 10$:       DECL    LEN1                                        ; 1229
                        51          88  90 00076            MOVB    (PTR1)+, CH                                 ; 1235
```

```
                        61   8F      51  91 00079          CMPB    CH, #97                          ; 1236
                                     0B  1F 0007D          BLSSU   11$                              ;
                        7A   8F      51  91 0007F          CMPB    CH, #122                         ;
                                     05  1A 00083          BGTRU   11$                              ;
                             50      20  D0 00085          MOVL    #32, R0                          ; 1237
                                     02  11 00088          BRB     12$                              ;
                                     50  D4 0008A 11$:     CLRL    R0                               ; 1236
                             6E      51  9A 0008C 12$:     MOVZBL  CH, (SP)                         ;
                             50      6E  C2 0008F          SUBL2   (SP), R0                         ;
                             50      50  CE 00092          MNEGL   R0, R0                           ;
     50          52          08      00  ED 00095          CMPZV   #0, #8, CHAR, R0                 ; 1240
                                     86  13 0009A          BEQL    1$                               ;
                             25      52  91 0009C          CMPB    CHAR, #37                        ; 1243
                                     14  13 0009F          BEQL    14$                              ;
                                     53  D7 000A1          DECL    SAVLEN1                          ; 1246
                                     16  19 000A3          BLSS    15$                              ;
                                     56  D6 000A5 13$:     INCL    SAVPTR1                          ; 1251
                             58      56  D0 000A7          MOVL    SAVPTR1, PTR1                    ;
                             57      59  D0 000AA          MOVL    SAVPTR2, PTR2                    ; 1252
                             55      53  D0 000AD          MOVL    SAVLEN1, LEN1                    ; 1253
                             54      5A  D0 000B0          MOVL    SAVLEN2, LEN2                    ; 1254
                                     BD  11 000B3          BRB     9$                               ; 1246
                        0C   BC      01  D0 000B5 14$:     MOVL    #1, @WILDCOMPARE                 ; 1258
                                     B7  11 000B9          BRB     9$                               ; 1243
                                     50  D4 000BB 15$:     CLRL    R0                               ; 1263
                                         04 000BD          RET                                      ;

; Routine Size:  190 bytes,    Routine Base:  CODE + 0838

;   1264  1
```

```
%SBTTL 'skip_blanks'
ROUTINE skip_blanks (desc) =

!++
! FUNCTIONAL DESCRIPTION
!       Advance line descriptor past blanks
!
! INPUTS
!       desc    pointer to descriptor of text
!
! IMPLICIT INPUTS
!       none
!
! OUTPUTS
!       desc    descriptor is updated past blanks
!
! IMPLICIT OUTPUTS
!       none
!
! SIDE EFFECTS
!       none
!
! RETURN CODES
!       0       if end-of-line detected
!       1       if non-blank character found
!--
```

```
;   1294  2         BEGIN
;   1295  2
;   1296  2         MAP
;   1297  2             desc : REF bblock;
;   1298  2
;   1299  2         LOCAL
;   1300  2             char : BYTE;
;   1301  2
;   1302  2         WHILE .desc [dsc$w_length] GTR 0 DO
;   1303  3             BEGIN
;   1304  3             char = CH$RCHAR (.desc [dsc$a_pointer]);        ! Get next character
;   1305  3
;   1306  3             IF .char EQL %C'!' THEN RETURN 0;       ! Comment delimits line
;   1307  3
;   1308  3             IF .char NEQ %C' ' AND .char NEQ %C'    ' THEN RETURN 1;        ! Done
;   1309  3
;   1310  3             desc [dsc$a_pointer] = .desc [dsc$a_pointer] + 1;       ! Advance descriptor
;   1311  3             desc [dsc$w_length] = .desc [dsc$w_length] - 1
;   1312  2             END;
;   1313  2
;   1314  2         0                                           ! Hit end of line
;   1315  1         END;                                        ! of skip_blanks



                                  0000 00000 SKIP_BLANKS:
                                                     .WORD   Save nothing                        ; 1267
                 50       04   AC  D0 00002          MOVL    DESC, R0                            ; 1302
                               60  B5 00006 1$:      TSTW    (R0)                                ;
                               1E  13 00008          BEQL    3$                                  ;
                 51       04   B0  90 0000A          MOVB    @4(R0), CHAR                        ; 1304
                 21            51  91 0000E          CMPB    CHAR, #33                           ; 1306
                               15  13 00011          BEQL    3$                                  ;
                 20            51  91 00013          CMPB    CHAR, #32                           ; 1308
                               09  13 00016          BEQL    2$                                  ;
                 09            51  91 00018          CMPB    CHAR, #9                            ;
                               04  13 0001B          BEQL    2$                                  ;
                 50            01  D0 0001D          MOVL    #1, R0                              ;
                                   04 00020          RET                                         ;
                          04   A0  D6 00021 2$:      INCL    4(R0)                               ; 1310
                               60  B7 00024          DECW    (R0)                                ; 1311
                               DE  11 00026          BRB     1$                                  ;
                               50  D4 00028 3$:      CLRL    R0                                  ; 1315
                                   04 0002A          RET                                         ;

; Routine Size:  43 bytes,    Routine Base:  CODE + 08F6
```

```
;    1317  1      %SBTTL 'scan_word'
;    1318  1      ROUTINE scan_word (desc) =
;    1319  1
;    1320  1      !++
;    1321  1      ! FUNCTIONAL DESCRIPTION
;    1322  1      !       scan descriptor over a word
;    1323  1      !
;    1324  1      ! INPUTS
;    1325  1      !       desc    address of descriptor for string
;    1326  1      !
;    1327  1      ! IMPLICIT INPUTS
;    1328  1      !       none
;    1329  1      !
;    1330  1      ! OUTPUTS
;    1331  1      !       desc    descriptor is updated past word scanned
;    1332  1      !       R0      length of word scanned
;    1333  1      !
;    1334  1      ! IMPLICIT OUTPUTS
;    1335  1      !       none
;    1336  1      !
;    1337  1      ! SIDE EFFECTS
;    1338  1      !       none
;    1339  1      !
;    1340  1      ! RETURN CODES
;    1341  1      !       none    (R0 if function value)
;    1342  1      !--
;    1343  1
```

```
;   1345  2          BEGIN
;   1346  2
;   1347  2          MAP
;   1348  2              desc : REF bblock;
;   1349  2
;   1350  2          LOCAL
;   1351  2              firstchar,                          ! flag
;   1352  2              startpointer;                       ! Point to initial character
;   1353  2
;   1354  2          startpointer = .desc [dsc$a_pointer];
;   1355  2          firstchar = 1;                          ! On first character
;   1356  2
;   1357  2          WHILE .desc [dsc$w_length] GTR 0 DO     ! Scan string
;   1358  3              BEGIN
;   1359  3
;   1360  3              LOCAL
;   1361  3                  char : BYTE;
;   1362  3
;   1363  3              char = CH$RCHAR (.desc [dsc$a_pointer]);        ! Read next character
;   1364  3              !
;   1365  3              ! If it's not a legal character to have inside a word, exit
;   1366  3              !
;   1367  3
;   1368  5              IF NOT ((.char GEQU %C'A' AND .char LEQU %C'Z') OR (.char GEQU %C'a' AND .char LEQU %C'z') OR (.char GEQU
;   1369  5                  %C'0' AND .char LEQU %C'9') OR (.char EQL %C'$')    ! dollar sign
;   1370  5                  OR (.char EQL %C'_')                ! underscore
;   1371  5                  OR (.char EQL %C'-')                ! hyphen
;   1372  5                  OR (.char EQL %C'.')                ! period
;   1373  5                  OR (.char EQL %C'%')                ! wildcard
;   1374  5                  OR (.char EQL %C'*')                ! wildcard
;   1375  4                  OR (.firstchar AND (.char EQL %C'/')))      ! slash (for qualifier)
;   1376  3              THEN
;   1377  3                  EXITLOOP;                           ! Exit if done with word
;   1378  3
;   1379  3              firstchar = 0;                          ! Reset flag
;   1380  3              desc [dsc$a_pointer] = .desc [dsc$a_pointer] + 1;
;   1381  3              desc [dsc$w_length] = .desc [dsc$w_length] - 1
;   1382  2              END;
;   1383  2
;   1384  3          (CH$DIFF (.desc [dsc$a_pointer], .startpointer))
;   1385  1          END;                                    ! of scan_word


                                        000C 00000 SCAN_WORD:
                                                        .WORD   Save R2,R3                          ; 1318
                      50      04   AC  D0 00002         MOVL    DESC, R0                            ; 1354
                      53      04   A0  D0 00006         MOVL    4(R0), STARTPOINTER                 ;
                      52           01  D0 0000A         MOVL    #1, FIRSTCHAR                       ; 1355
                                   60  B5 0000D 1$:     TSTW    (R0)                                ; 1357
                                   56  13 0000F         BEQL    6$                                  ;
                      51      04   B0  90 00011         MOVB    @4(R0), CHAR                        ; 1363
                41    8F           51  91 00015         CMPB    CHAR, #65                           ; 1368
                                   06  1F 00019         BLSSU   2$                                  ;
                5A    8F           51  91 0001B         CMPB    CHAR, #90                           ;
```

```
                                  3D  1B 0001F        BLEQU   5$                                  ;
                 61   8F          51  91 00021 2$:    CMPB    CHAR, #97                           ;
                                  06  1F 00025        BLSSU   3$                                  ;
                 7A   8F          51  91 00027        CMPB    CHAR, #122                          ;
                                  31  1B 0002B        BLEQU   5$                                  ;
                      30          51  91 0002D 3$:    CMPB    CHAR, #48                           ;
                                  05  1F 00030        BLSSU   4$                                  ;
                      39          51  91 00032        CMPB    CHAR, #57                           ; 1369
                                  27  1B 00035        BLEQU   5$                                  ;
                      24          51  91 00037 4$:    CMPB    CHAR, #36                           ;
                                  22  13 0003A        BEQL    5$                                  ;
                 5F   8F          51  91 0003C        CMPB    CHAR, #95                           ; 1370
                                  1C  13 00040        BEQL    5$                                  ;
                      2D          51  91 00042        CMPB    CHAR, #45                           ; 1371
                                  17  13 00045        BEQL    5$                                  ;
                      2E          51  91 00047        CMPB    CHAR, #46                           ; 1372
                                  12  13 0004A        BEQL    5$                                  ;
                      25          51  91 0004C        CMPB    CHAR, #37                           ; 1373
                                  0D  13 0004F        BEQL    5$                                  ;
                      2A          51  91 00051        CMPB    CHAR, #42                           ; 1374
                                  08  13 00054        BEQL    5$                                  ;
                      0E          52  E9 00056        BLBC    FIRSTCHAR, 6$                       ; 1375
                      2F          51  91 00059        CMPB    CHAR, #47                           ;
                                  09  12 0005C        BNEQ    6$                                  ;
                                  52  D4 0005E 5$:    CLRL    FIRSTCHAR                           ; 1379
                            04    A0  D6 00060        INCL    4(R0)                               ; 1380
                                  60  B7 00063        DECW    (R0)                                ; 1381
                                  A6  11 00065        BRB     1$                                  ;
          50      04   A0         53  C3 00067 6$:    SUBL3   STARTPOINTER, 4(R0), R0             ; 1384
                                      04 0006C        RET                                         ; 1385

; Routine Size:  109 bytes,    Routine Base:  CODE + 0921
```

```
;    1387  1      %SBTTL 'find character'
;    1388  1      ROUTINE find_char (desc, character) : jsb_2 =
;    1389  1
;    1390  1      !++
;    1391  1      ! FUNCTIONAL DESCRIPTION
;    1392  1      !       find distance to character in string
;    1393  1      !
;    1394  1      ! INPUTS
;    1395  1      !       desc            address of string descriptor to search
;    1396  1      !       character       the character to look for
;    1397  1      !
;    1398  1      ! IMPLICIT INPUTS
;    1399  1      !       none
;    1400  1      !
;    1401  1      ! OUTPUTS
;    1402  1      !       R0              distance in string to character
;    1403  1      !
;    1404  1      ! IMPLICIT OUTPUTS
;    1405  1      !       none
;    1406  1      !
;    1407  1      ! SIDE EFFECTS
;    1408  1      !       none
;    1409  1      !
;    1410  1      ! RETURN CODES
;    1411  1      !       R0 is function value
;    1412  1      !--
;    1413  1
```

```
;   1415  2         BEGIN
;   1416  2
;   1417  2         LOCAL
;   1418  2             ptr,
;   1419  2             len;
;   1420  2
;   1421  2         MAP
;   1422  2             desc : REF bblock,
;   1423  2             character : BYTE;
;   1424  2
;   1425  2         len = .desc [dsc$w_length];
;   1426  2         ptr = .desc [dsc$a_pointer];
;   1427  2
;   1428  2         WHILE .len GTR 0 DO
;   1429  3             BEGIN
;   1430  3             len = .len - 1;
;   1431  3
;   1432  3             IF CH$RCHAR (.ptr) EQL .character THEN EXITLOOP;        ! If we found it
;   1433  3
;   1434  3             ptr = .ptr + 1
;   1435  2             END;
;   1436  2
;   1437  2         CH$DIFF (.ptr, .desc [dsc$a_pointer])       ! Return distance
;   1438  1         END;                                        ! of find_char



                                    0C  BB 00000 FIND_CHAR:
                                                        PUSHR   #^M<R2,R3>                          ; 1388
                        53          60  3C 00002        MOVZWL  (DESC), LEN                         ; 1425
                        52      04  A0  D0 00005        MOVL    4(DESC), PTR                        ; 1426
                                    53  D5 00009 1$:    TSTL    LEN                                 ; 1428
                                    0B  15 0000B        BLEQ    2$                                  ;
                                    53  D7 0000D        DECL    LEN                                 ; 1430
                        51          62  91 0000F        CMPB    (PTR), CHARACTER                    ; 1432
                                    04  13 00012        BEQL    2$                                  ;
                                    52  D6 00014        INCL    PTR                                 ; 1434
                                    F1  11 00016        BRB     1$                                  ;
                50      52      04  A0  C3 00018 2$:    SUBL3   4(DESC), PTR, R0                    ; 1437
                                    0C  BA 0001D        POPR    #^M<R2,R3>                          ; 1438
                                        05 0001F        RSB                                         ;

; Routine Size:  32 bytes,    Routine Base:  CODE + 098E
```

```
;    1440  1     %SBTTL 'print blank line'
;    1441  1     ROUTINE print_crlf : jsb_none NOVALUE =
;    1442  1
;    1443  1     !++
;    1444  1     ! Print a blank line
;    1445  1     !--
;    1446  1
;    1447  2         BEGIN
;    1448  3         $ds_printf ($ascic ('!/'))
;    1449  1         END;                                    ! of print_crlf


                                                            .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                    2F  21  02  000B1 P.AAN:  .ASCII  <2>\!/\                                   ;


                                                            .PSECT  CODE,NOWRT,  SHR,2

                              00000000'  EF  9F 00000 PRINT_CRLF:
                                                            PUSHAB  P.AAN                                       ; 1448
                         00000000G  9F        01  FB 00006        CALLS   #1, a#DS$PRINTF                        ;
                                              05 0000D           RSB                                            ; 1449

; Routine Size:  14 bytes,    Routine Base:  CODE + 09AE
```

```
;   1451  1     END
;   1452  1
;   1453  0     ELUDOM



;                                  PSECT SUMMARY
;
;       Name                    Bytes                        Attributes
;
;  DATA                           180  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                            46  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          2492  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



Symbol                 Type  Defined  Referenced ...
------------------ -------- ----- ----------------
$ASCIC             Macro    Lib01   780     860     902     903     922    1035    1448
$ASCID             Macro    Lib01   145
$CLOSE             KeyWMacr Lib03   395
$CONNECT           KeyWMacr Lib03   364
$DS_PRINTF         Macro    Lib01   780     860     930     934    1019    1035    1448
$FAB_DECL          Macro    Lib03   194
$FAB_INIT          KeyWMacr Lib03   217     219
$GET               KeyWMacr Lib03  1080
$OPEN              KeyWMacr Lib03   352
$RAB_DECL          Macro    Lib03   195
$RAB_INIT          KeyWMacr Lib03   220
$RMS_BITFLD_INI    Macro    Lib03   217     219     220
$RMS_BITS          Macro    Lib03   217     219
$RMS_CODFLD_INI    Macro    Lib03   217     219     220
$RMS_OR            Macro    Lib03   217     219
$RMS_PTR           Unbound          220
                   Bind       217   217=
                   Bind       219   219=
                   Bind       220   220=
$RMS_VALFLD_INI    Unbound          220
                   Macro    Lib03   217     219     220
BBLOCK             Structur Lib02   159     196     197     199     202     205     226     438     439     442     443     450
                                    451     454     679     682     685     823     901     913    1077    1078    1144    1145
                                   1297    1348    1422
BOUND_LEVEL        Local      452   468=    488.    531.    537.    548.
BUFFER             Local      197   220a
CH                 Local     1233  1235=   1236.
CHAR               Local      448   463=    465=    467.
                   Local      686   695=    697.    698.    704.
                   Local     1173  1206=   1210=   1210.   1213.   1240.   1243.
                   Local     1300  1304=   1306.   1308.
                   Local     1361  1363=   1368.   1369.   1370.   1371.   1372.   1373.   1374.   1375.
CHARACTER          RoutForm  1388  1423m   1432.
CURKEY             Bind       226   228a    236=    243=    246.    247.    248.    248a.   251.    255=    255.    257.    269.
```

Symbol	Type	Defined	Referenced ...										
	Bind	439	461.	463a.	540a								
DESC	RoutForm	164	205m	234.	236a.	237.	241a.	244.	245a=	245a.	247a=	247a.	
	RoutForm	1267	1297m	1302a.	1304a.	1310a=	1310a.	1311a=	1311a.				
	RoutForm	1318	1348m	1354a.	1357a.	1363a.	1380a=	1380a.	1381a=	1381a.	1384a.		
	RoutForm	1388	1422m	1425a.	1426a.	1437a.							
DO_HELP	Routine	400	114f	372c	571c								
DS$PRINTF	ExtRout	781	781c	860e	860c	930e	930c	934e	934c	1019e	1019c	1035e	1035c	1448e
			1448c										
DSC$A_POINTER	Macro	Lib03	228	236	247	248	251	284	311	366	461	463	551	552
			565	695	712	731	988	1024	1165	1166	1304	1310	1354	1363
			1380	1384	1426	1437								
DSC$C_S_BLN	Literal	Lib03	199	202	205	211	226	313	329	439	450	451	685	694
			823	913										
DSC$W_LENGTH	Macro	Lib03	241	243	245	246	247	248	251	255	257	269	312	326
			549	689	732	1163	1164	1302	1311	1357	1381	1425		
DSR$KEY_EQUAL	GlobRout	1112	120f	341c	540c								
DSX$HELP	GlobRout	164	113f										
ELIPSIS	Local	201	208=	248.									
ERR	RoutForm	918	925.										
ERRFLG	RoutForm	868	1016.	1033.									
FAB	Bind	1078	1088.	1096.									
FAB$B_BID	Macro	Lib03	217	219									
FAB$B_BKS	Macro	Lib03	217	219									
FAB$B_BLN	Macro	Lib03	217	219									
FAB$B_DNS	Macro	Lib03	217	219									
FAB$B_FAC	Macro	Lib03	217	219									
FAB$B_FNS	Macro	Lib03	217	219	312								
FAB$B_FSZ	Macro	Lib03	217	219									
FAB$B_JOURNAL	Macro	Lib03	217	219									
FAB$B_ORG	Macro	Lib03	217	219									
FAB$B_RAT	Macro	Lib03	217	219	354	1088	1096						
FAB$B_RFM	Macro	Lib03	217	219									
FAB$B_RTV	Macro	Lib03	217	219									
FAB$B_RU_FACILITY	Macro	Lib03	217	219									
FAB$B_SHR	Macro	Lib03	217	219									
FAB$C_BID	Literal	Lib03	217	219									
FAB$C_BLN	Literal	Lib03	194	217	219								
FAB$C_VAR	Literal	Lib03	217	219									
FAB$L_ALQ	Macro	Lib03	217	219									
FAB$L_CTX	Macro	Lib03	217	219									
FAB$L_DNA	Macro	Lib03	217	219									
FAB$L_FNA	Macro	Lib03	217	219	311								
FAB$L_FOP	Macro	Lib03	217	219									
FAB$L_MRN	Macro	Lib03	217	219									
FAB$L_NAM	Macro	Lib03	217	219									
FAB$L_XAB	Macro	Lib03	217	219									
FAB$M_FTN	Literal	Lib03	1088										
FAB$M_GET	Literal	Lib03	217	219									
FAB$V_CHAN_MODE	Macro	Lib03	217	219									
FAB$V_FILE_MODE	Macro	Lib03	217	219									
FAB$V_LNM_MODE	Macro	Lib03	217	219									
FAB$W_BLS	Macro	Lib03	217	219									
FAB$W_DEQ	Macro	Lib03	217	219									
FAB$W_GBC	Macro	Lib03	217	219									
FAB$W_MRS	Macro	Lib03	217	219									

Symbol	Type	Defined	Referenced ...											
FFS	Builtin	610	614											
FIND_CHAR	Routine	1388	123f	237c	244c									
FIRST	Local	606	615=											
	Local	910	941=	1016.	1018=	1033.								
FIRSTCHAR	Local	1351	1355=	1375.	1379=									
FIRSTRFA	RoutForm	400	443m	469a.	495a=									
FMT	Bind	902	1019a											
HELPFAB	Local	194	217a	219a	220a	311=	312=	352a	354.	395a				
HELPINFO	Global	159	209=	210=	212=	214=	227a	254=	259=	259.	263.	277=	315=	315.
			317.	319=	319.	321=	321.	326.	330=	333.	343=	357=	382.	383.
			438.	439.	456.	457.	534.	541.	546a	550.	551.	553=	555.	556.
			559=	574.	584.	615.	617.	618.	682a	771a	781.	860a	901.	923.
			928=	939.	988.	1077.	1102=	1103=						
HELPRAB	Local	195	214a	220a	364a									
HELP_HELP	Bind	145	329.	341a										
HLP$B_WILDFLAGS	Macro	Lib02	227	315	333	615								
HLP$C_MAXKEYS	Literal	Lib02	202	210	211	263	313	317	456	487	526	536	546	553
			722	729	825	850	851	852	859	969	971			
HLP$C_MAXRECSIZ	Literal	Lib02	197	198	220	449	732	914						
HLP$C_SIZE	Literal	Lib02	159	209										
HLP$L_ALLHELP	Macro	Lib02	210	254	259	263	317	319	456	534	541	556	584	618
			939											
HLP$L_BUFDESC	Macro	Lib02	550	551	682	860	988	1102	1103					
HLP$L_KEYLIST	Macro	Lib02	212	439	923									
HLP$L_KEYNAMES	Macro	Lib02	546	771										
HLP$L_RAB	Macro	Lib02	214	438	901	1077								
HLP$L_REALKEYS	Macro	Lib02	277	321	326	330	456	555	574					
HLP$V_CR	Macro	Lib02	357											
HLP$V_FOUNDHELP	Macro	Lib02	382	559	617	928								
HLP$V_HELP_KEY	Macro	Lib02	343	383										
HLP$V_QUAL	Macro	Lib02	553	780	939									
I	Local	263	264.											
	Local	269	271.											
	Local	773	781.											
INDENT	RoutForm	786	825.											
IS_KEY_ON_LINE	Routine	644	115f	485c	524c	848c	967c							
JSB_2	Linkage	104	123	918	1388									
JSB_3	Linkage	105												
JSB_NONE	Linkage	Lib02	124	1441										
KEY	RoutForm	400	442m	548.	549a=	552a=								
	RoutForm	644	679m	731a=	732a=									
KEY1	RoutForm	1112	1144m	1163a.	1165a.									
KEY1TEXT	Bind	284	286a.	287a.	288a.	289a.	290a.	291a.	292a.	293a.	294a.	295a.	296a.	297a.
			298a.	299a.	300a.	301a.	302a.	303a.	304a.	305a.	306a.	307a.	308a.	
KEY2	RoutForm	1112	1145m	1164a.	1166a.									
KEYDESC	Local	451	485a	524a	540a	551.	551a.							
	Local	823	848a											
	Local	913	967a	988.	1024a.									
KEYLIST	Local	202	211=	212a	226a	284a	311.	312.	313.	313=	326.	329=	341a	
	Bind	923	930a											
KEYNAMES	Bind	546	548=											
	Bind	771	781.											
KEYRFA	RoutForm	868	946a.											
KEYTEXT	Bind	228	271a.											
LASTLEVEL	Local	453	468=	529=	537.									

Symbol	Type	Defined	Referenced ...										
	Local	908	942=	969=	971.								
LASTQUAL	Local	822	833=	850.	859=								
LEN	Local	710	713=	715.	719=	719.							
	Local	1072	1082=	1089.	1092=	1092.	1097.	1098.	1100=	1100.	1102.		
	Local	1419	1425=	1428.	1430=	1430.							
LEN1	Local	1152	1163=	1180.	1199=	1222.	1229=	1229.	1253=				
LEN2	Local	1153	1164=	1175=	1175.	1177.	1200=	1218.	1223.	1254=			
LENGTH	Local	984	1006=	1010.	1024.	1026.							
LEV	RoutForm	918	929.										
LEVEL	Local	200	213=	226.	254.	261=	261.	271.	274=	274.	277.	279.	
	RoutForm	400	439.	456.	468.	487.	491.	533.	534.	541.	555.	556.	560. 561.
			571.	574.	581.	584.	613.	618.	633.				
	Local	687	693=	718=	718.	722.	729=	735.					
	RoutForm	739	773.	781.									
	RoutForm	786	825.	850.	852.								
	RoutForm	868	939.	942.	971.	972.	979.	1016.	1033.				
LINE	Bind	682	689.	694.									
LINECOL	Local	911	945=	1021=	1026=	1026.	1029.						
LINELEVEL	Local	446	485=	487.	489.	491.							
	Local	515	524=	526.	527.	529.	531.	533.	536.	553.	561.		
	Local	821	848=	849.	851.	859.							
	Local	909	967=	969.	971.	972.	979.						
LOCDESC	Local	685	694=	695a.	712.	713a	726a	731.	732a				
LOGTABS	Local	820	825=	860.									
LOOP	Label	1148	1169	1203									
NEXTDESC	Local	199	366=	372a									
	Local	450	565=	571a									
NEXTKEY	Local	198	366a										
	Local	449	565a										
NEXT_QUAL	Local	231	237=	239.	241=	244.							
NODOCMSG	Bind	922	930a										
NOERR	Local	377	382=	390.									
NO_QUAL_YET	Local	907	943=	1010.	1014=								
OTHERINFO	Bind	903	934a										
OUTBUF	Local	914	944a	1008a	1019a	1020a	1035a						
OUTLEN	Local	986	1008=	1010.	1019.								
POSITION	Local	604	612=	614.									
PRINTHEADER	Routine	918	1016c	1033c									
PRINT_CRLF	Routine	1441	124f	207c	394c	863c							
PRINT_KEYS	Routine	739	116f	560c	929c								
PRINT_OTHERHELP	Routine	868	118f	390c	581c	633c							
PRINT_TEXT	Routine	786	117f	561c									
PTR	Local	232	248=	251.									
	Local	709	712=	716a.	718.	718=							
	Local	912	944=	1008.	1020=	1024=	1024a=	1035.					
	Local	1073	1083=	1093=	1093.	1098.	1103.						
	Local	1418	1426=	1432a.	1434=	1434.	1437.						
PTR1	Local	1154	1165=	1197=	1220.	1235.	1235=	1251=					
PTR2	Local	1155	1166=	1198=	1206.	1206=	1221.	1252=					
QUALIFIER	Local	447	467=	487.	492.	535.							
QUALINE	Local	906	971=	979.	1010.	1014.							
RAB	Bind	438	469=	477.	495.	507=	562=	571a	581a	587=	626=	627=	633a
	Bind	901	946=	947=	956=								
	Bind	1077	1078.	1080a	1081=	1082.	1083.						
RAB$B_BID	Macro	Lib03	220										

Symbol	Type	Defined	Referenced ...						
RAB$B_BLN	Macro	Lib03	220						
RAB$B_KRF	Macro	Lib03	220						
RAB$B_KSZ	Unbound		220						
RAB$B_MBC	Macro	Lib03	220						
RAB$B_MBF	Macro	Lib03	220						
RAB$B_RAC	Macro	Lib03	220	507	562	587	626	947	956	1081
RAB$B_TMO	Macro	Lib03	220						
RAB$C_BID	Literal	Lib03	220						
RAB$C_BLN	Literal	Lib03	195	220					
RAB$C_RFA	Literal	Lib03	507	562	587	626	947		
RAB$C_SEQ	Literal	Lib03	956	1081					
RAB$L_BKT	Macro	Lib03	220						
RAB$L_CTX	Macro	Lib03	220						
RAB$L_FAB	Macro	Lib03	220	1078					
RAB$L_KBF	Unbound		220						
RAB$L_RBF	Macro	Lib03	220	1083					
RAB$L_RHB	Macro	Lib03	220						
RAB$L_ROP	Macro	Lib03	220						
RAB$L_UBF	Macro	Lib03	220						
RAB$L_XAB	Macro	Lib03	220						
RAB$W_RFA	Macro	Lib03	469	477	495	571	581	627	633	946
RAB$W_RSZ	Macro	Lib03	220	1082					
RAB$W_USZ	Macro	Lib03	220						
READ	Routine	1042	119f	478c	517c	628c	629c	837c	955c	
REALLEN	Local	985	988=	1006.	1024.					
RFA$C_LENGTH	Literal	Lib02	196	454	469	477	495	946		
RFA$L_VBN	Macro	Lib02	345	379						
RFA$W_OFFSET	Macro	Lib02	346	380						
RMS$_EOF	Literal	Lib03	483	522	839	958	1105			
RMS$_RFA	Literal	Lib03	1105							
SAVERFA	Local	454	477=	627.						
SAVLEN1	Local	1156	1162=	1186=	1186.	1199.	1222=	1246=	1246.	1253.
SAVLEN2	Local	1157	1200.	1223=	1254.					
SAVPTR1	Local	1158	1197=	1197.	1220=	1251=	1251.			
SAVPTR2	Local	1159	1198.	1221=	1252.					
SCAN_WORD	Routine	1318	122f	713c	732c					
SDL$STARLET_CONCAT	Macro	Lib03	352	364	395	1080				
SDL$STARLET_OPT	Macro	Lib03	352	364	395	1080				
SDL$STARLET_REQ	Macro	Lib03	352	364	395	1080				
SIZE	Local	605	613=	614.						
SKIP_BLANKS	Routine	1267	121f	234c	726c					
STARTPOINTER	Local	1352	1354=	1384.						
STARTRFA	Local	196	345=	346=	372a	379=	380=	390a		
STAT	Local	350	352=	352.						
	Local	362	364=	365.						
	Local	370	372=	373.						
	Local	388	390=	391.						
	Local	475	478=	480.	483.					
	Local	514	517=	519.	522.					
	Local	569	571=	572.						
	Local	579	581=	582.						
	Local	624	628=	629=	631.	633=	635.			
	Local	819	837=	839.	843.	864.				
	Local	953	955=	958.	962.					
	Local	1074	1080=	1105.	1105=	1107.				

Symbol	Type	Defined	Referenced ...			
SYS$CLOSE	ExtRout	395	395c			
SYS$CONNECT	ExtRout	364	364c			
SYS$GET	ExtRout	1080	1080c			
SYS$OPEN	ExtRout	352	352c			
ULTRIX_FLAGS	External	152	215.	354.	1084.	1207.
ULTRIX_V_MODE	Literal	147	215	354	1084	1207
W	Local	339	341a			
WILD	Local	1151	1161=	1189.	1216=	
WILDCOMPARE	RoutForm	1112	1161a=	1193a=	1218a=	1258a=
WILDFLAGS	Bind	227	264=	271=		
WILDKEY	Local	513	540a	584.		
WILDPATH	Local	607	614=	619.		

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]HELP.B32;1
6	Module HELP	
92	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
94	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
96	Library #3	SYS$COMMON:[SYSLIB]STARLET.L32;2
1453	Eludom HELP	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
a	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	3	0	96	00:00.0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	19	2	47	00:00.0
SYS$COMMON:[SYSLIB]STARLET.L32;2	10593	77	0	624	00:00.5

```
;                                   COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:HELP.B32

; Size:         2492 code + 226 data bytes
; Run Time:        00:10.7
; Elapsed Time:    00:13.8
; Lines/CPU Min:    8132
; Lexemes/CPU-Min: 90906
; Memory Used:  301 pages
; Compilation Complete
```

Table of contents

(2) 167 DECLARATIONS
(3) 210 Device description database

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ICODE.MAR
0000     2 ; *9    12-NOV-1987 14:27:39 MI "ADDED A FEW MACROS FOR COMPILE AND LINK"
0000     3 ; *8    11-NOV-1987 16:45:28 MI "FIXED THE COMPILATON PROBLEM"
0000     4 ; *7    11-NOV-1987 15:02:13 MI "ELIMINATED MACROS DEFINED IN DIAG.MLB ALREADY"
0000     5 ; *6    16-MAR-1987 15:30:37 BARANSKI "Fix undefined LN01 label"
0000     6 ; *5     4-MAR-1987 14:06:24 BERGAZZI "ADDED RA70, LG01,LG02,LG03"
0000     7 ; *4    11-MAR-1986 15:52:02 BARANSKI "<no reason given>"
0000     8 ; *3    19-FEB-1986 20:16:03 BARANSKI "Fixing CMS error."
0000     9 ; *2    12-FEB-1986 13:25:52 BARANSKI "<no reason given>"
0000    10 ; *1     6-FEB-1986 15:18:51 DS ""
0000    11 ; DEC/CMS REPLACEMENT HISTORY, Element ICODE.MAR
0000    12         .TITLE  ICODE - Interpreted code for TSP
0000    13         .IDENT  "10.10-08"                                              ;G:9
0000    14         .LIST   MEB
0000    15         .NLIST  CND
0000    16
0000    17 ;
0000    18 ; Copyright (c) 1979, 1982, 1983, 1984, 1985, 1987                      ;G:7
0000    19 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    20 ;
0000    21 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    22 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    23 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    24 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    25 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    26 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    27 ; REMAIN IN DEC.
0000    28 ;
0000    29 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    30 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    31 ; CORPORATION.
0000    32 ;
0000    33 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    34 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    35
0000    36 ;++
0000    37 ; FACILITY:     TSP
0000    38 ;
0000    39 ; ABSTRACT:     This module is used to generate the necessary driver
0000    40 ;               for the APT Attach command interpreter.  It is really
0000    41 ;               part of TSP.
0000    42 ;
0000    43 ; ENVIRONMENT:
0000    44 ;
0000    45 ; AUTHOR:       R. Riggs        15-Jul-1979
0000    46 ;
0000    47 ; MODIFIED BY:
0000    48 ;               Dave Butenhof, 2-apr-1981 (with Supervisor V6.3)
0000    49 ;       01      Add PTdescriptors for ML11, KA730, LP25, TM78 and TU78
0000    50 ;
0000    51 ;       02      Dave Butenhof, 13-Nov-1981, version 6.5
0000    52 ;               Add Ptable descriptors for DW730, DMF32[A,P,S, ], UDA50,
0000    53 ;               RB730, R80, and RA80.
0000    54 ;
0000    55 ;       [03] Dave butenhof, 09-Apr-1982, version 6.7
0000    56 ;               Support $DS_$NAME directive (ignore it)
0000    57 ;       [04] Dave Butenhof, 03-May-1982, version 6.7
```

```
                                0000     58 ;               Add RA81 and RA60 devices
                                0000     59 ;
                                0000     60 ;       05      M. Baggett       24-Sept-1982 Version 6.9
                                0000     61 ;               added CI750
                                0000     62 ;
                                0000     63 ;       06      Bob Bergazzi    2-Nov-1982      Version 6.9
                                0000     64 ;               added UNA11
                                0000     65 ;
                                0000     66 ;       07      M. Baggett       3-Nov-1982     Version 6.9
                                0000     67 ;               Added CI750 macro call.
                                0000     68 ;
                                0000     69 ;       08      M. BAGGETT       1-Dec-1982     Version 6.10
                                0000     70 ;               Added TU80 and DZ32.
                                0000     71 ;
                                0000     72 ;       09      M. BAGGETT       8-DEC-1982     Version 6.10
                                0000     73 ;               Added TU81.
                                0000     74 ;
                                0000     75 ;       10      M. BAGGETT       10-DEC-1982    Version 6.10
                                0000     76 ;               Added VS100.
                                0000     77 ;
                                0000     78 ;       11      M. BAGGETT       14-JAN-1983    Version 6.11
                                0000     79 ;               Added RC25.
                                0000     80 ;
                                0000     81 ;       12      M. Baggett       23-Feb-1983    Version 6.11
                                0000     82 ;               Added CI_NODE and RCF25,VS300.
                                0000     83 ;
                                0000     84 ;       13      M. Baggett       14-Mar-1983    Version 6.11
                                0000     85 ;               Added p-table for LESI controller.
                                0000     86 ;
                                0000     87 ;       14      M. Baggett       21-Mar-1983    Version 6.11
                                0000     88 ;               Added VT220,VT240,LA12,LA100,LN01.
                                0000     89 ;
                                0000     90 ;       15      Richard Brown   18-July-83      Version 6.12
                                0000     91 ;               Added CONSOLE, KAXXX, KAYYY, LP07, LP26, LP27,
                                0000     92 ;               RC11, SBIA.
                                0000     93 ;
                                0000     94 ;       16      M. Baggett       8-Aug-1983     Version 6.12
                                0000     95 ;               Added IEU11A.
                                0000     96 ;
                                0000     97 ;       17      M. Baggett       7-Sep-1983     Version 6.13
                                0000     98 ;               Added VS125.
                                0000     99 ;
                                0000    100 ;       18      M. Baggett       22-Nov-1983    Version 6.13
                                0000    101 ;               Added TS05,DISK.
                                0000    102 ;
                                0000    103 ;       19      Richard Brown   5-Mar-84        Version 6.14
                                0000    104 ;               Added DEQNA, DLVJ1, DZV11, RQDX1, RX50, RD51, RD52
                                0000    105 ;               for MicroVAX I.
                                0000    106 ;
                                0000    107 ;       20      Richard Brown   3-Apr-84        Version 7.0
                                0000    108 ;               Corrected typo which included RD51 twice and
                                0000    109 ;               excluded RD52.  Now both RD51 and  RD52  are
                                0000    110 ;               there.
                                0000    111 ;
                                0000    112 ;       21      M. Baggett       21-Jun-84      Version 7.0
                                0000    113 ;               Added DMZ32.
                                0000    114 ;
```

```
0000    115 ;       22      Richard Brown   7-Jul-84        Version 7.0
0000    116 ;               Removed RC11, since it is really LESI.
0000    117 ;
0000    118 ;       23      Amy Iorio       1-Oct-84        Version 7.1
0000    119 ;               Changed KAXXX to KA86.
0000    120 ;
0000    121 ;       24      Amy Iorio       11-Oct-84       Version 7.1
0000    122 ;               Added KAXXX.
0000    123 ;
0000    124 ;       25      M. Baggett      2-NOV-1984      Version 7.1
0000    125 ;               Added TK50.
0000    126 ;
0000    127 ;       26      M. Baggett      15-Nov-1984     Version 7.1
0000    128 ;               Added RUX50,RX50,RX31,RX32,RX180.
0000    129 ;
0000    130 ;       27      Bob Bergazzi    Nov. 29,1984    Version 8.0
0000    131 ;               Removed KAXXX. Added KA810.
0000    132 ;
0000    133 ;       28      M. Baggett      18-Dec-1984     Version 8.0
0000    134 ;               Added VT101, VT102, VT125.
0000    135 ;
0000    136 ;       29      A. Iorio        18-Jan-1985     Version 8.1
0000    137 ;               Added DHU11, DR11C.
0000    138 ;
0000    139 ;       30      A. Iorio        5-Mar-1985      Version 8.1
0000    140 ;               Added DX20,TU70,TU72.
0000    141 ;                                                                   ;G:9
0000    142 ;       31      A. Iorio        12-Mar-1985     Version 8.1
0000    143 ;               Added DHV11.
0000    144 ;
0000    145 ;       32      A. Iorio        12-Aug-1985     Version 8.3
0000    146 ;               Added LN03, LA210, VT61, VT71, VT131, VT132.
0000    147 ;
0000    148 ;       33      J. Baranski     14-Oct-1985     Version 9.0
0000    149 ;               Added RA82 disk, and LUA11, removed KA810 and RC11.
0000    150 ;
0000    151 ;       34      J. Baranski     30-Jan-1986     Version 9.1
0000    152 ;               Fix $DS_$HEX Macro to match Macro in DIAG.
0000    153 ;                                                                   ;G:4
0000    154 ;       35      J.Baranski      11-Mar-1986     Version 10.0        ;G:4
0000    155 ;               Add RRD50, Compact Disc Rom Drive.                  ;G:4
0000    156 ;                                                                   ;G:5
0000    157 ;       36      Bob Bergazzi    4-Mar-1986      Version 10.7        ;G:5
0000    158 ;               Added LG01,LG02,LG03,RA70.                          ;G:5
0000    159 ;                                                                   ;G:7
0000    160 ;               Jason Mi        11-Nov-1987     Version 10.10       ;G:7
0000    161 ;               Eliminated the MACRO's which are already defined in DIAG.MLB.;G:7 ;G
0000    162 ;                                                                   ;G:9
0000    163 ;               Jason Mi        12-Nov-1987     Version 10.10       ;G:9
0000    164 ;               Keep the old Macros except for $ds_disk $ds_ra90    ;G:9
0000    165 ;--                                                                 ;G:9
```

```
                        0000   167          .SBTTL  DECLARATIONS
                        0000   168 ;
                        0000   169 ; INCLUDE FILES:
                        0000   170 ;
                        0000   171
                        0000   172 .LIBRARY        /$DIAG/
                        0000   173
                        0000   174          $DS_HPODEF                                                    ;G:8
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
                00000008 0000              .IIF    NB,..BLKQ,      .BLKQ   1
                        0008               .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
                0000000A 0008              .IIF    NB,..BLKW,      .BLKW   1
                        000A               .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
                0000000B 000A              .IIF    NB,..BLKB,      .BLKB   1
                        000B               .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
                0000000C 000B              .IIF    NB,..BLKB,      .BLKB   1
                        000C               .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
                00000018 000C              .IIF    NB,..BLKB,      .BLKB   12
                        0018               .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
                0000001C 0018              .IIF    NB,..BLKL,      .BLKL   1
                        001C               .IIF    NB,HP$A_DVA,    HP$A_DVA:
                00000020 001C              .IIF    NB,..BLKL,      .BLKL   1
                        0020               .IIF    NB,HP$A_LINK,   HP$A_LINK:
                00000024 0020              .IIF    NB,..BLKL,      .BLKL   1
                        0024               .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
                00000026 0024              .IIF    NB,..BLKW,      .BLKW   1
                        0026               .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
                00000032 0026              .IIF    NB,..BLKB,      .BLKB   12
                        0032               .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                        0000               .RESTORE
                        0000   175          $DS_HPEODEF                                                   ;G:8
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                00000000 0032              .=0
                        0000               .IIF    NB,HPE$T_DEVICE,        HPE$T_DEVICE:
                00000014 0000              .IIF    NB,..BLKL,      .BLKL   5
                        0014               .IIF    NB,HPE$W_EXT_DRIVE,     HPE$W_EXT_DRIVE:
                00000016 0014              .IIF    NB,..BLKW,      .BLKW   1
                        0016               .IIF    NB,HPE$A_EPB,   HPE$A_EPB:
                0000001A 0016              .IIF    NB,..BLKL,      .BLKL   1
                        0000               .RESTORE
                        0000   176          $DS_DHUV_DEF                                                  ;G:9
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                        0032               .IIF    NB,HP$L_DHUV_CSR,       HP$L_DHUV_CSR:
                00000036 0032              .IIF    NB,..BLKL,      .BLKL   1
                        0036               .IIF    NB,HP$W_DHUV_VEC,       HP$W_DHUV_VEC:
                00000038 0036              .IIF    NB,..BLKW,      .BLKW   1
                        0038               .IIF    NB,HP$B_DHUV_BR,        HP$B_DHUV_BR:
                00000039 0038              .IIF    NB,..BLKB,      .BLKB   1
                        0039               .IIF    NB,HP$K_DHUV_LEN,       HP$K_DHUV_LEN:
                        0000               .RESTORE
                        0000   177          $DS_CI_DEF                                                    ;G:9
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                00000032 0039              .=HP$A_DEPENDENT
```

```
                     0032                .IIF    NB,HP$B_CI_TR,  HP$B_CI_TR:
                     0032                .IIF    NB,CI$B_BI_ID,  CI$B_BI_ID:
                     0032                .IIF    NB,CI$B_TR,     CI$B_TR:
            00000033 0032                .IIF    NB,..BLKB,      .BLKB   1
                     0033                .IIF    NB,HP$B_CI_BR,  HP$B_CI_BR:
                     0033                .IIF    NB,CI$B_BR,     CI$B_BR:
            00000034 0033                .IIF    NB,..BLKB,      .BLKB   1
                     0034                .IIF    NB,HP$B_CI_NODE,        HP$B_CI_NODE:
                     0034                .IIF    NB,CI$B_NODE,   CI$B_NODE:
            00000035 0034                .IIF    NB,..BLKB,      .BLKB   1
                     0035                .IIF    NB,HP$L_CI_FUNC,        HP$L_CI_FUNC:
                     0035                .IIF    NB,Ci$L_Func,   Ci$L_Func:
            00000039 0035                .IIF    NB,..BlkL,      .BlkL   1
                     0039                .IIF    NB,HP$L_CI_MAINT_ID,    HP$L_CI_MAINT_ID:
                     0039                .IIF    NB,Ci$L_Maint_Id,       Ci$L_Maint_Id:
            0000003D 0039                .IIF    NB,..BlkL,      .BlkL   1
                     003D                .IIF    NB,HP$L_CI_INDEX,       HP$L_CI_INDEX:
                     003D                .IIF    NB,Ci$L_Index,  Ci$L_Index:
            00000041 003D                .IIF    NB,..BlkL,      .BlkL   1
                     0041                .IIF    NB,HP$K_CI_LEN, HP$K_CI_LEN:
                     0041                .IIF    NB,CI$K_LEN,    CI$K_LEN:
                     0000                .RESTORE
                     0000   178                                                                      ;G:9
                     0000   179          .MACRO  $DS_DISK_DEF    $GBL                    ;           [163
                     0000   180          $DEFINI DISK,$GBL,HP$A_DEPENDENT                            ;G:9
                     0000   181          $DEF    HP$K_DISK_LEN                                       ;G:9
                     0000   182          $DEF    DISK$K_LEN                                          ;G:9
                     0000   183          $DEFEND DISK,$GBL,DEF                                       ;G:9
                     0000   184 .ENDM    $DS_DISK_DEF                            ;           [163]   ;G:9
                     0000   185                                                                      ;G:9
                     0000   186 .MACRO   $DS_DISK                                                    ;G:9
                     0000   187          $DS_DISK_DEF                                                ;G:9
                     0000   188          $DS_$INITIALIZE DISK,DISK$K_LEN,255                         ;G:77
                     0000   189          $DS_$Name       Ptd$M_DEVICE, DU        ;           [164]   ;G:9
                     0000   190          $DS_$END                                                    ;G:9
                     0000   191 .ENDM    $DS_DISK                                ;           [163]   ;G:9
                     0000   192                                                                      ;G:9
                     0000   193 .MACRO   $DS_RA90_DEF    $GBL                                        ;G:77
                     0000   194          $DEFINI RA90,$GBL,HP$A_DEPENDENT                            ;G:77
                     0000   195          $DEF    HP$K_RA90_LEN                                       ;G:77
                     0000   196          $DEF    RA90$K_LEN                                          ;G:77
                     0000   197          $DEFEND RA90,$GBL,DEF                                       ;G:77
                     0000   198 .ENDM    $DS_RA90_DEF                                                ;G:77
                     0000   199                                                                      ;G:77
                     0000   200 .MACRO   $DS_RA90                                                   ;G:77
                     0000   201          $DS_RA90_DEF                                                ;G:77
                     0000   202          $DS_$INITIALIZE RA90,RA90$K_LEN,255                         ;G:77
                     0000   203          $DS_$NAME       PTD$M_ENDDEVICE, DU                         ;G:77
                     0000   204          $DS_$LITERAL    HP$M_ALLOC                                  ;G:77
                     0000   205          $DS_$STORE      HP$B_FLAGS,0,8                              ;G:77
                     0000   206          $DS_$END                                                    ;G:77
                     0000   207 .ENDM    $DS_RA90                                                    ;G:77
                     0000   208                                                                      ;G:8
```

```
                     0000   210          .SBTTL  Device description database
                     0000   211 ;
                     0000   212 ;       The following list contains the descriptor for each type
                     0000   213 ;       of device the supervisor known about.
                     0000   214 ;
                     0000   215
                     0000   216 DS$GA_PTDESC::
                     0000   217          $DS_DEVTYP      <>,<AA11K,AD11K,CI_NODE,CI780,CI750,CONSOLE,-   ; [15]
                     0000   218                  CR11,DEQNA,DHU11,DHV11,DISK,DL11,DLVJ1 -                ; [18][19][2
                     0000   219                  DMC11,DMF32,DMF32A,DMF32P,DMF32S,-                      ; [02]
                     0000   220                  DMP11,DMR11,DMZ32,DR11B,DR11C,-                         ; [21][29]
                     0000   221                  DR11K,DR11W,DR750,DR780, -
                     0000   222                  DUP11,DW730,DW750,DW780,DX20,DZ11,DZ32,DZV11,-           ; [02][19][3
                     0000   223                  IEU11A, -                                                       ;G:2
                     0000   224                  KA730,KA750,KA780,KA785,KA86, -                          ; [01][15][2
                     0000   225                  KMC11,KW11K,LA12,LA34,LA36,LA38, -                       ; [14] ;G:2
                     0000   226                  LA100,LA120,LA180,LA210, -                                      ;G:5
                     0000   227                  LESI, -                                                  ; [13] ;G:5
                     0000   228                  LG01,LG02,LG03, -                                        ; [36] ;G:5
                     0000   229                  LN01,LN03,LP04, -                                        ; [32] ;G:5
                     0000   230                  LP05,LP06,LP07,LP11,LP14, -                              ; [15]
                     0000   231                  LP25,LP26,LP27,LPA11K,LUA11,MA780,MBE,-                  ; [01][15][3
                     0000   232                  ML11,MS750,MS780,NBIA,NBIB,PCL11,-                       ; [01]
                     0000   233                  PX780,-                                                  ;
                     0000   234                  R80,RA60,RA70,RA80,RA81,RA82,RA90,RB730,RCF25,RC25, -    ; [12][15][2
                     0000   235                  RD51,RD52,RH750,RH780,RL01,RL02, -                       ; [19][20]
                     0000   236                  RL11,RK06,RK07,RK611, -
                     0000   237                  RM03,RM05,RM80, -
                     0000   238                  RP04,RP05,RP06,RP07,RQDX1,RRD50,RUX50,RX02, -            ; [19][26][3
                     0000   239                  RX211,RX31,RX32,RX50,RX180,-                             ; [26]
                     0000   240                  SBIA,TE16,TK50,TM03,TM78,-                               ; [01][15][1
                     0000   241                  TS04,TS05,TS11,TU45,TU58,TU70,TU72,TU77,TU78,TU80,TU81-  ; [18][30]
                     0000   242                  UBE,UDA50,UNA11,-                                        ; [06]
                     0000   243                  VS100,VS125,VS300,VT50,VT52,VT55,VT61,VT71,-             ; [12][32]
                     0000   244                  VT100,VT101,VT102,VT125,VT131,VT132,VT220,-              ; [32]
                     0000   245                  VT240>
00000094             0000           AL_DEVTYP:      .LONG   $$N
00000254'            0004                           .LONG   AA11K
000002A5'            0008                           .LONG   AD11K
000002F6'            000C                           .LONG   CI_NODE
00000491'            0010                           .LONG   CI780
00000412'            0014                           .LONG   CI750
00000518'            0018                           .LONG   CONSOLE
00000526'            001C                           .LONG   CR11
0000057D'            0020                           .LONG   DEQNA
000005A5'            0024                           .LONG   DHU11
000005F3'            0028                           .LONG   DHV11
00000641'            002C                           .LONG   DISK
00000651'            0030                           .LONG   DL11
0000069E'            0034                           .LONG   DLVJ1
00000752'            0038                           .LONG   DMC11
000007A0'            003C                           .LONG   DMF32
000007F3'            0040                           .LONG   DMF32A
00000915'            0044                           .LONG   DMF32P
0000098F'            0048                           .LONG   DMF32S
00000A03'            004C                           .LONG   DMP11
00000ACC'            0050                           .LONG   DMR11
```

```
00000B1A' 0054          .LONG   DMZ32
00000C78' 0058          .LONG   DR11B
00000CC6' 005C          .LONG   DR11C
00000D0F' 0060          .LONG   DR11K
00000D60' 0064          .LONG   DR11W
00000DAE' 0068          .LONG   DR750
00000E39' 006C          .LONG   DR780
00000ECC' 0070          .LONG   DUP11
00000F67' 0074          .LONG   DW730
00000F96' 0078          .LONG   DW750
00000FDA' 007C          .LONG   DW780
0000104E' 0080          .LONG   DX20
0000107C' 0084          .LONG   DZ11
000010E5' 0088          .LONG   DZ32
000011D6' 008C          .LONG   DZV11
00001213' 0090          .LONG   IEU11A
0000125E' 0094          .LONG   KA730
00001339' 0098          .LONG   KA750
00001405' 009C          .LONG   KA780
00001556' 00A0          .LONG   KA785
000014AE' 00A4          .LONG   KA86
000015FF' 00A8          .LONG   KMC11
0000164D' 00AC          .LONG   KW11K
0000169E' 00B0          .LONG   LA12
00001709' 00B4          .LONG   LA34
00001723' 00B8          .LONG   LA36
0000173D' 00BC          .LONG   LA38
000016B8' 00C0          .LONG   LA100
000016D3' 00C4          .LONG   LA120
000016EE' 00C8          .LONG   LA180
00001757' 00CC          .LONG   LA210
00001772' 00D0          .LONG   LESI
000017BE' 00D4          .LONG   LG01
000017D8' 00D8          .LONG   LG02
000017F2' 00DC          .LONG   LG03
0000180C' 00E0          .LONG   LN01
00001826' 00E4          .LONG   LN03
0000183B' 00E8          .LONG   LP04
00001855' 00EC          .LONG   LP05
0000186F' 00F0          .LONG   LP06
00001889' 00F4          .LONG   LP07
000018A3' 00F8          .LONG   LP11
000018F0' 00FC          .LONG   LP14
0000190A' 0100          .LONG   LP25
00001924' 0104          .LONG   LP26
0000193E' 0108          .LONG   LP27
00001958' 010C          .LONG   LPA11K
000019B1' 0110          .LONG   LUA11
00001AD2' 0114          .LONG   MA780
00001B4D' 0118          .LONG   MBE
00001B61' 011C          .LONG   ML11
00001BC3' 0120          .LONG   MS750
00001BD4' 0124          .LONG   MS780
000019FF' 0128          .LONG   NBIA
00001A6F' 012C          .LONG   NBIB
00001C24' 0130          .LONG   PCL11
00001C6D' 0134          .LONG   PX780
```

```
00001CD6' 0138          .LONG   R80
00001CEF' 013C          .LONG   RA60
00001D09' 0140          .LONG   RA70
00001D23' 0144          .LONG   RA80
00001D3D' 0148          .LONG   RA81
00001D57' 014C          .LONG   RA82
00001D71' 0150          .LONG   RA90
00001D8B' 0154          .LONG   RB730
00001DBA' 0158          .LONG   RCF25
00001DD5' 015C          .LONG   RC25
00001DEF' 0160          .LONG   RD51
00001E09' 0164          .LONG   RD52
00001E23' 0168          .LONG   RH750
00001E7C' 016C          .LONG   RH780
00001EE1' 0170          .LONG   RL01
00001EFB' 0174          .LONG   RL02
00001F15' 0178          .LONG   RL11
00001F62' 017C          .LONG   RK06
00001F7C' 0180          .LONG   RK07
00001F96' 0184          .LONG   RK611
00001FE4' 0188          .LONG   RM03
00002008' 018C          .LONG   RM05
0000202C' 0190          .LONG   RM80
00002050' 0194          .LONG   RP04
00002074' 0198          .LONG   RP05
00002098' 019C          .LONG   RP06
000020BC' 01A0          .LONG   RP07
000020E0' 01A4          .LONG   RQDX1
00002107' 01A8          .LONG   RRD50
00002163' 01AC          .LONG   RUX50
000021AB' 01B0          .LONG   RX02
000021C5' 01B4          .LONG   RX211
00002213' 01B8          .LONG   RX31
00002223' 01BC          .LONG   RX32
00002233' 01C0          .LONG   RX50
00002248' 01C4          .LONG   RX180
00002259' 01C8          .LONG   SBIA
0000229B' 01CC          .LONG   TE16
000022B5' 01D0          .LONG   TK50
000022CF' 01D4          .LONG   TM03
000022FD' 01D8          .LONG   TM78
0000232B' 01DC          .LONG   TS04
00002382' 01E0          .LONG   TS05
000023EC' 01E4          .LONG   TS11
00002443' 01E8          .LONG   TU45
0000245D' 01EC          .LONG   TU58
00002477' 01F0          .LONG   TU70
00002491' 01F4          .LONG   TU72
000024AB' 01F8          .LONG   TU77
000024C5' 01FC          .LONG   TU78
000024DF' 0200          .LONG   TU80
0000254E' 0204          .LONG   TU81
000025A5' 0208          .LONG   UBE
000025E0' 020C          .LONG   UDA50
00002649' 0210          .LONG   UNA11
00002697' 0214          .LONG   VS100
000026EF' 0218          .LONG   VS125
```

```
                        00002747' 021C                  .LONG   VS300
                        0000279F' 0220                  .LONG   VT50
                        000027B9' 0224                  .LONG   VT52
                        000027D3' 0228                  .LONG   VT55
                        000027ED' 022C                  .LONG   VT61
                        00002807' 0230                  .LONG   VT71
                        00002821' 0234                  .LONG   VT100
                        0000283C' 0238                  .LONG   VT101
                        00002857' 023C                  .LONG   VT102
                        00002872' 0240                  .LONG   VT125
                        0000288D' 0244                  .LONG   VT131
                        000028A8' 0248                  .LONG   VT132
                        000028C3' 024C                  .LONG   VT220
                        000028DE' 0250                  .LONG   VT240
```

```
                                0254   247          .LIST   MEB,MC
                                0254   248
                                0254   249 AA11K:   $DS_AA11K
                                0254                .SAVE   LOCAL_BLOCK
                                0254                .PSECT  $ABS$,ABS
                      00000032  0041                .=HP$A_DEPENDENT
                                0032                .IIF    NB,AA11K$B_BR,  AA11K$B_BR:
                      00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                                0033                .IIF    NB,AA11K$K_LEN, AA11K$K_LEN:
                          00000254                  .RESTORE
                  4B 31 31 41 41 00' 0254           .ASCIC  "AA11K" ; AA11K name
                            05  0254
                            33  025A                .BYTE   AA11K$K_LEN             ; AA11K$K_LEN of resultant P-table
                            04  025B                .BYTE   4                       ; Maximum unit number                   ;G:65
                          0000  025C                .WORD   ^A""    ; Two character mnemonic for QIO AA11K
                            80  025E                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                            8D  025F                .Byte   Pd$_Name        ; Directive op-code
                            03  0260                .Byte   Ptd$M_Device            ; Enforced AA codes
                        41 41 00' 0261              .Ascic  "AA"    ; Enforced device name prefix
                            02  0261
                            83  0264                .BYTE   Pd$_Octal       ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 0265           .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                            0B  0265
                    0003FFFE 0003E000  0271         .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                            88  0279                .BYTE   Pd$_Store       ; Indicate store operation
                          0018  027A                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                            00  027C                .BYTE   0               ; 0 HP$A_DEVICE to data
                            12  027D                .BYTE   18              ; 18 of data field
                            83  027E                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 4F 54 43 45 56 00' 027F        .ASCIC  "VECTOR"        ; VECTOR string
                            06  027F
                    000001FE 00000002  0286         .LONG   ^O2,^O776       ; 2 , 776 limits on input
                            88  028E                .BYTE   Pd$_Store       ; Indicate store operation
                          0024  028F                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                            00  0291                .BYTE   0               ; 0 HP$W_VECTOR to data
                            09  0292                .BYTE   9               ; 9 of data field
                            82  0293                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                        52 42 00' 0294              .ASCIC  "BR"    ; BR string
                            02  0294
                    00000007 00000004  0297         .LONG   4,7     ; 4 , 7 limits on input
                            88  029F                .BYTE   Pd$_Store       ; Indicate store operation
                          0032  02A0                .WORD   AA11K$B_BR              ; Byte AA11K$B_BR to data
                            00  02A2                .BYTE   0               ; 0 AA11K$B_BR to data
                            08  02A3                .BYTE   8               ; 8 of data field
                            81  02A4                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                02A5   250 AD11K:   $DS_AD11K                       ;
                                02A5                .SAVE   LOCAL_BLOCK
                                02A5                .PSECT  $ABS$,ABS
                      00000032  0041                .=HP$A_DEPENDENT
                                0032                .IIF    NB,AD11K$B_BR,  AD11K$B_BR:
                      00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                                0033                .IIF    NB,AD11K$K_LEN, AD11K$K_LEN:
                          000002A5                  .RESTORE
                  4B 31 31 44 41 00' 02A5           .ASCIC  "AD11K" ; AD11K name
                            05  02A5
                            33  02AB                .BYTE   AD11K$K_LEN             ; AD11K$K_LEN of resultant P-table
                            04  02AC                .BYTE   4               ; Maximum unit number                   ;G:65
```

```
                          0000  02AD                .WORD   ^A""     ; Two character mnemonic for QIO AD11K
                            80  02AF                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                            8D  02B0                .Byte   Pd$_Name        ; Directive op-code
                            03  02B1                .Byte   Ptd$M_Device            ; Enforced AD codes
                     44 41 00'  02B2                .Ascic  "AD"    ; Enforced device name prefix
                            02  02B2
                            83  02B5                .BYTE   Pd$_Octal       ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00'  02B6          .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                            0B  02B6
                    0003FFFE 0003E000  02C2         .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                            88  02CA                .BYTE   Pd$_Store       ; Indicate store operation
                          0018  02CB                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                            00  02CD                .BYTE   0               ; 0 HP$A_DEVICE to data
                            12  02CE                .BYTE   18              ; 18 of data field
                            83  02CF                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 4F 54 43 45 56 00'  02D0       .ASCIC  "VECTOR"        ; VECTOR string
                            06  02D0
                    000001FE 00000002  02D7         .LONG   ^O2,^O776       ; 2 , 776 limits on input
                            88  02DF                .BYTE   Pd$_Store       ; Indicate store operation
                          0024  02E0                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                            00  02E2                .BYTE   0               ; 0 HP$W_VECTOR to data
                            09  02E3                .BYTE   9               ; 9 of data field
                            82  02E4                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                     52 42 00'  02E5                .ASCIC  "BR"    ; BR string
                            02  02E5
                    00000007 00000004  02E8         .LONG   4,7     ; 4 , 7 limits on input
                            88  02F0                .BYTE   Pd$_Store       ; Indicate store operation
                          0032  02F1                .WORD   AD11K$B_BR              ; Byte AD11K$B_BR to data
                            00  02F3                .BYTE   0               ; 0 AD11K$B_BR to data
                            08  02F4                .BYTE   8               ; 8 of data field
                            81  02F5                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                02F6  251 CI_NODE:$DS_CI_NODE                   ;               [12]
                  45 44 4F 4E 5F 49 43 00'  02F6    .ASCIC  "CI_NODE"       ; CI_NODE name
                            07  02F6
                            41  02FE                .BYTE   CI$K_LEN                ; CI$K_LEN of resultant P-table
                            00  02FF                .BYTE   0               ; Maximum unit number                   ;G:65
                          0000  0300                .WORD   ^A""    ; Two character mnemonic for QIO CI_NODE
                            80  0302                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                            8D  0303                .Byte   Pd$_Name        ; Directive op-code
                            04  0304                .Byte   Ptd$M_NAME              ; Enforced  codes
                            00' 0305                .Ascic  ""      ; Enforced device name prefix
                            00  0305
                            87  0306                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                          0018  0307                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                            00  0309                .BYTE   0               ; 0 HP$A_DEVICE to data
                            20  030A                .BYTE   32              ; 32 of data field
                            88  030B                .BYTE   Pd$_Store       ; Indicate store operation
                          001C  030C                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                            00  030E                .BYTE   0               ; 0 HP$A_DVA to data
                            20  030F                .BYTE   32              ; 32 of data field
                            85  0310                .BYTE   Pd$_String      ; Indicate scan and verify string
65 70 79 74 5F 65 64 6F 4E 00'  0311                .ASCIC  "Node_type"     ; Node_type string
                            09  0311
                         00 01  031B                .BYTE   1, 0
                         00 01  031D                .BYTE   1, 0
                  30 38 37 58 41 56 00'  031F       .ASCIC  "VAX780"
                            06  031F
```

```
                30 35 37 58 41 56 00' 0326              .ASCIC  "VAX750"
                                  06  0326
                   30 35 43 53 48 00' 032D              .ASCIC  "HSC50"
                                  05  032D
                               00 01  0333              .BYTE   1, 0
                      30 31 4C 4B 00' 0335              .ASCIC  "KL10"
                                  04  0335
                      54 4E 49 43 00' 033A              .ASCIC  "CINT"
                                  04  033A
                   35 38 37 58 41 56 00' 033F           .ASCIC  "VAX785"
                                  06  033F
             30 30 36 38 58 41 56 00' 0346              .ASCIC  "VAX8600"
                                  07  0346
             30 30 32 38 58 41 56 00' 034E              .ASCIC  "VAX8200"
                                  07  034E
             30 30 38 38 58 41 56 00' 0356              .ASCIC  "VAX8800"
                                  07  0356
                                  00  035E              .BYTE   0                       ; Null length is end of valid ,,
                                  88  035F              .BYTE   Pd$_Store               ; Indicate store operation
                               003D  0360              .WORD   Ci$L_Index                      ; Byte Ci$L_Index to data
                                  00  0362              .BYTE   0                       ; 0 Ci$L_Index to data
                                  20  0363              .BYTE   32                      ; 32 of data field
                                  8C  0364              .BYTE   Pd$_Case                ; Indicate case operation
                                  05  0365              .BYTE   $$_S                    ; Count of pairs
          00000002 00000003       0366                  .LONG   ^X3,                            ^X2             ; Store the pair
          00000002 00000008       036E                  .LONG   ^X8,                            ^X2             ; Store the pair
          00000002 00000009       0376                  .LONG   ^X9,                            ^X2             ; Store the pair
          00000002 0000000A       037E                  .LONG   ^XA,                            ^X2             ; Store the pair
          00000002 0000000B       0386                  .LONG   ^XB,                            ^X2             ; Store the pair
                                  88  038E              .BYTE   Pd$_Store               ; Indicate store operation
                               0039  038F              .WORD   Ci$L_Maint_Id                   ; Byte Ci$L_Maint_Id to data
                                  00  0391              .BYTE   0                       ; 0 Ci$L_Maint_Id to data
                                  20  0392              .BYTE   32                      ; 32 of data field
                                  87  0393              .BYTE   Pd$_Fetch               ; Indicate fetch operation
                               003D  0394              .WORD   Ci$L_index                      ; Byte Ci$L_index to data
                                  00  0396              .BYTE   0                       ; 0 Ci$L_index to data
                                  20  0397              .BYTE   32                      ; 32 of data field
                                  8C  0398              .BYTE   Pd$_Case                ; Indicate case operation
                                  09  0399              .BYTE   $$_S                    ; Count of pairs
          FFFF0F00 00000002       039A                  .LONG   ^X2,                            ^XFFFF0F00              ; Store the pair
          FFFF0F00 00000003       03A2                  .LONG   ^X3,                            ^XFFFF0F00              ; Store the pair
          4F710200 00000004       03AA                  .LONG   ^X4,                            ^X4F710200              ; Store the pair
          A3FFAD00 00000006       03B2                  .LONG   ^X6,                            ^XA3FFAD00              ; Store the pair
          FFFFFF00 00000007       03BA                  .LONG   ^X7,                            ^XFFFFFF00              ; Store the pair
          FFFF0F00 00000008       03C2                  .LONG   ^X8,                            ^XFFFF0F00              ; Store the pair
          FFFF0F00 00000009       03CA                  .LONG   ^X9,                            ^XFFFF0F00              ; Store the pair
          FFFF0F00 0000000A       03D2                  .LONG   ^XA,                            ^XFFFF0F00              ; Store the pair
          FFFF0F00 0000000B       03DA                  .LONG   ^XB,                            ^XFFFF0F00              ; Store the pair
                                  88  03E2              .BYTE   Pd$_Store               ; Indicate store operation
                               0035  03E3              .WORD   Ci$L_Func                       ; Byte Ci$L_Func to data
                                  00  03E5              .BYTE   0                       ; 0 Ci$L_Func to data
                                  20  03E6              .BYTE   32                      ; 32 of data field
                                  82  03E7              .BYTE   Pd$_Decimal             ; Indicate scan decimal
73 65 72 64 64 61 5F 65 64 6F 4E 00' 03E8              .ASCIC  "Node_address"          ; Node_address string
                                  73  03F4
                                  0C  03E8
          000000FF 00000000       03F5                  .LONG   0,255   ; 0 , 255 limits on input
```

```
                      88  03FD                .BYTE   Pd$_Store       ; Indicate store operation
                    0034  03FE                .WORD   CI$B_NODE               ; Byte CI$B_NODE to data
                      00  0400                .BYTE   0               ; 0 CI$B_NODE to data
                      08  0401                .BYTE   8               ; 8 of data field
                      88  0402                .BYTE   Pd$_Store       ; Indicate store operation
                    000B  0403                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                      00  0405                .BYTE   0               ; 0 HP$B_DRIVE to data
                      08  0406                .BYTE   8               ; 8 of data field
                      86  0407                .BYTE   Pd$_Literal     ; Indicate literal
                80000000  0408                .LONG   ^X80000000              ; Constant
                      8A  040C                .BYTE   Pd$_Add         ; Indicate add value to field operation
                    0039  040D                .WORD   CI$L_MAINT_ID           ; Byte CI$L_MAINT_ID to data
                      00  040F                .BYTE   0               ; 0 CI$L_MAINT_ID to data
                      20  0410                .BYTE   32              ; 32 of data field
                      81  0411                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          0412   252 CI750:    $DS_CI750                       ;               [07]
          30 35 37 49 43 00' 0412                .ASCIC  "CI750" ; CI750 name
                      05  0412
                      41  0418                .BYTE   CI$K_LEN                ; CI$K_LEN of resultant P-table
                      08  0419                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  041A                .WORD   ^A""    ; Two character mnemonic for QIO CI750
                      80  041C                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  041D                .Byte   Pd$_Name        ; Directive op-code
                      03  041E                .Byte   Ptd$M_Device            ; Enforced PA codes
                41 50 00' 041F                .Ascic  "PA"    ; Enforced device name prefix
                      02  041F
                      86  0422                .BYTE   Pd$_Literal     ; Indicate literal
                40F30000  0423                .LONG   ^X40F30000              ; Constant
                      88  0427                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0428                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  042A                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  042B                .BYTE   32              ; 32 of data field
                      82  042C                .BYTE   Pd$_Decimal     ; Indicate scan decimal
             74 6F 6C 53 00' 042D                .ASCIC  "Slot"  ; Slot string
                      04  042D
            0000000F 0000000A 0432                .LONG   10,15   ; 10 , 15 limits on input
                      88  043A                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  043B                .WORD   CI$B_TR         ; Byte CI$B_TR to data
                      00  043D                .BYTE   0               ; 0 CI$B_TR to data
                      08  043E                .BYTE   8               ; 8 of data field
                      88  043F                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0440                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      0D  0442                .BYTE   13              ; 13 HP$A_DEVICE to data
                      04  0443                .BYTE   4               ; 4 of data field
                      88  0444                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  0445                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      02  0447                .BYTE   2               ; 2 HP$W_VECTOR to data
                      04  0448                .BYTE   4               ; 4 of data field
                      87  0449                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  044A                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  044C                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  044D                .BYTE   32              ; 32 of data field
                      88  044E                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  044F                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      00  0451                .BYTE   0               ; 0 HP$A_DVA to data
                      20  0452                .BYTE   32              ; 32 of data field
                      82  0453                .BYTE   Pd$_Decimal     ; Indicate scan decimal
```

```
                    52 42 00' 0454                  .ASCIC  "BR"    ; BR string
                       02     0454
              00000007 00000004  0457                .LONG   4,7     ; 4 , 7 limits on input
                       88       045F                .BYTE   Pd$_Store       ; Indicate store operation
                     0033       0460                .WORD   CI$B_BR         ; Byte CI$B_BR to data
                       00       0462                .BYTE   0               ; 0 CI$B_BR to data
                       08       0463                .BYTE   8               ; 8 of data field
                       88       0464                .BYTE   Pd$_Store       ; Indicate store operation
                     0024       0465                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       06       0467                .BYTE   6               ; 6 HP$W_VECTOR to data
                       03       0468                .BYTE   3               ; 3 of data field
                       82       0469                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 65 64 6F 4E 00' 046A                .ASCIC  "Node"  ; Node string
                       04       046A
              000000FF 00000000  046F                .LONG   0,255   ; 0 , 255 limits on input
                       88       0477                .BYTE   Pd$_Store       ; Indicate store operation
                     0034       0478                .WORD   CI$B_NODE               ; Byte CI$B_NODE to data
                       00       047A                .BYTE   0               ; 0 CI$B_NODE to data
                       08       047B                .BYTE   8               ; 8 of data field
                       86       047C                .BYTE   Pd$_Literal     ; Indicate literal
                 FFFF0F00       047D                .LONG   ^XFFFF0F00              ; Constant
                       88       0481                .BYTE   Pd$_Store       ; Indicate store operation
                     0035       0482                .WORD   Ci$L_Func               ; Byte Ci$L_Func to data
                       00       0484                .BYTE   0               ; 0 Ci$L_Func to data
                       20       0485                .BYTE   32              ; 32 of data field
                       86       0486                .BYTE   Pd$_Literal     ; Indicate literal
                 80000002       0487                .LONG   ^X80000002              ; Constant
                       88       048B                .BYTE   Pd$_Store       ; Indicate store operation
                     0039       048C                .WORD   Ci$L_Maint_Id           ; Byte Ci$L_Maint_Id to data
                       00       048E                .BYTE   0               ; 0 Ci$L_Maint_Id to data
                       20       048F                .BYTE   32              ; 32 of data field
                       81       0490                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                0491  253 CI780:     $DS_CI780                       ;               [02]
           30 38 37 49 43 00' 0491                  .ASCIC  "CI780" ; CI780 name
                       05       0491
                       41       0497                .BYTE   CI$K_LEN                ; CI$K_LEN of resultant P-table
                       08       0498                .BYTE   8               ; Maximum unit number                   ;G:65
                     0000       0499                .WORD   ^A""    ; Two character mnemonic for QIO CI780
                       80       049B                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D       049C                .Byte   Pd$_Name        ; Directive op-code
                       03       049D                .Byte   Ptd$M_Device            ; Enforced PA codes
                 41 50 00' 049E                     .Ascic  "PA"    ; Enforced device name prefix
                       02       049E
                       82       04A1                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 54 00' 04A2                     .ASCIC  "TR"    ; TR string
                       02       04A2
              0000000F 00000001  04A5                .LONG   1,15    ; 1 , 15 limits on input
                       88       04AD                .BYTE   Pd$_Store       ; Indicate store operation
                     0032       04AE                .WORD   CI$B_TR         ; Byte CI$B_TR to data
                       00       04B0                .BYTE   0               ; 0 CI$B_TR to data
                       08       04B1                .BYTE   8               ; 8 of data field
                       88       04B2                .BYTE   Pd$_Store       ; Indicate store operation
                     0018       04B3                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       0D       04B5                .BYTE   13              ; 13 HP$A_DEVICE to data
                       04       04B6                .BYTE   4               ; 4 of data field
                       88       04B7                .BYTE   Pd$_Store       ; Indicate store operation
                     0024       04B8                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
```

```
                      02  04BA                .BYTE   2               ; 2 HP$W_VECTOR to data
                      04  04BB                .BYTE   4               ; 4 of data field
                      82  04BC                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 04BD                .ASCIC  "BR"    ; BR string
                      02  04BD
          00000007 00000004  04C0                .LONG   4,7     ; 4 , 7 limits on input
                      88  04C8                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  04C9                .WORD   CI$B_BR         ; Byte CI$B_BR to data
                      00  04CB                .BYTE   0               ; 0 CI$B_BR to data
                      08  04CC                .BYTE   8               ; 8 of data field
                      88  04CD                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  04CE                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      06  04D0                .BYTE   6               ; 6 HP$W_VECTOR to data
                      02  04D1                .BYTE   2               ; 2 of data field
                      86  04D2                .BYTE   Pd$_Literal     ; Indicate literal
                00000006  04D3                .LONG   6               ; Constant
                      88  04D7                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  04D8                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      1C  04DA                .BYTE   28              ; 28 HP$A_DEVICE to data
                      04  04DB                .BYTE   4               ; 4 of data field
                      87  04DC                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  04DD                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  04DF                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  04E0                .BYTE   32              ; 32 of data field
                      88  04E1                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  04E2                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      00  04E4                .BYTE   0               ; 0 HP$A_DVA to data
                      20  04E5                .BYTE   32              ; 32 of data field
                      86  04E6                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  04E7                .LONG   1               ; Constant
                      88  04EB                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  04EC                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      08  04EE                .BYTE   8               ; 8 HP$W_VECTOR to data
                      01  04EF                .BYTE   1               ; 1 of data field
                      82  04F0                .BYTE   Pd$_Decimal     ; Indicate scan decimal
             65 64 6F 4E 00' 04F1                .ASCIC  "Node"  ; Node string
                      04  04F1
          000000FF 00000000  04F6                .LONG   0,255   ; 0 , 255 limits on input
                      88  04FE                .BYTE   Pd$_Store       ; Indicate store operation
                    0034  04FF                .WORD   CI$B_NODE               ; Byte CI$B_NODE to data
                      00  0501                .BYTE   0               ; 0 CI$B_NODE to data
                      08  0502                .BYTE   8               ; 8 of data field
                      86  0503                .BYTE   Pd$_Literal     ; Indicate literal
                FFFF0F00  0504                .LONG   ^XFFFF0F00              ; Constant
                      88  0508                .BYTE   Pd$_Store       ; Indicate store operation
                    0035  0509                .WORD   Ci$L_Func               ; Byte Ci$L_Func to data
                      00  050B                .BYTE   0               ; 0 Ci$L_Func to data
                      20  050C                .BYTE   32              ; 32 of data field
                      86  050D                .BYTE   Pd$_Literal     ; Indicate literal
                80000002  050E                .LONG   ^X80000002              ; Constant
                      88  0512                .BYTE   Pd$_Store       ; Indicate store operation
                    0039  0513                .WORD   Ci$L_Maint_Id           ; Byte Ci$L_Maint_Id to data
                      00  0515                .BYTE   0               ; 0 Ci$L_Maint_Id to data
                      20  0516                .BYTE   32              ; 32 of data field
                      81  0517                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          0518    254 CONSOLE:$DS_CONSOLE                     ;               [15]
                          0518                .SAVE   LOCAL_BLOCK
```

```
                        0518                    .PSECT  $ABS$,ABS
                00000032  0041                    .=Hp$A_Dependent
                          0032                    .IIF    NB,Console$K_Len,       Console$K_Len:
                00000518  0518                    .RESTORE
   65 6C 6F 73 6E 6F 63 00' 0518                    .ASCIC  "console"       ; console name
                      07  0518
                      32  0520                    .BYTE   Console$K_Len           ; Console$K_Len of resultant P-table
                      01  0521                    .BYTE   1                       ; Maximum unit number                   ;G:65
                    0000  0522                    .WORD   ^A""    ; Two character mnemonic for QIO console
                      80  0524                    .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      81  0525                    .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          0526    255 CR11:       $DS_CR11
                          0526                    .SAVE   LOCAL_BLOCK
                          0526                    .PSECT  $ABS$,ABS
                00000032  0041                    .=HP$A_DEPENDENT
                          0032                    .IIF    NB,CR11$L_CSR,  CR11$L_CSR:
                00000036  0032                    .IIF    NB,.BLKL,       .BLKL   1
                          0036                    .IIF    NB,CR11$B_BR,   CR11$B_BR:
                00000037  0036                    .IIF    NB,.BLKB,       .BLKB   1
                          0037                    .IIF    NB,CR11$K_LEN,  CR11$K_LEN:
                00000526  0526                    .RESTORE
      31 31 52 43 00' 0526                    .ASCIC  "CR11"  ; CR11 name
                      04  0526
                      37  052B                    .BYTE   CR11$K_LEN              ; CR11$K_LEN of resultant P-table
                      01  052C                    .BYTE   1                       ; Maximum unit number                   ;G:65
                    0000  052D                    .WORD   ^A""    ; Two character mnemonic for QIO CR11
                      80  052F                    .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  0530                    .Byte   Pd$_Name        ; Directive op-code
                      03  0531                    .Byte   Ptd$M_Device            ; Enforced CR codes
                52 43 00' 0532                    .Ascic  "CR"    ; Enforced device name prefix
                      02  0532
                      86  0535                    .BYTE   Pd$_Literal     ; Indicate literal
                00000001  0536                    .LONG   HP$M_ALLOC              ; Constant
                      88  053A                    .BYTE   Pd$_Store       ; Indicate store operation
                    000A  053B                    .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  053D                    .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  053E                    .BYTE   8                       ; 8 of data field
                      83  053F                    .BYTE   Pd$_Octal       ; Indicate scan octal
            52 53 43 00' 0540                    .ASCIC  "CSR"   ; CSR string
                      03  0540
        0003FFFE 0003E000  0544                    .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  054C                    .BYTE   Pd$_Store       ; Indicate store operation
                    0032  054D                    .WORD   CR11$L_CSR              ; Byte CR11$L_CSR to data
                      00  054F                    .BYTE   0                       ; 0 CR11$L_CSR to data
                      20  0550                    .BYTE   32                      ; 32 of data field
                      88  0551                    .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0552                    .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  0554                    .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  0555                    .BYTE   18                      ; 18 of data field
                      83  0556                    .BYTE   Pd$_Octal       ; Indicate scan octal
   52 4F 54 43 45 56 00' 0557                    .ASCIC  "VECTOR"        ; VECTOR string
                      06  0557
        000001FE 00000002  055E                    .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  0566                    .BYTE   Pd$_Store       ; Indicate store operation
                    0024  0567                    .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  0569                    .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  056A                    .BYTE   9                       ; 9 of data field
```

```
                      82  056B                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 056C                .ASCIC  "BR"    ; BR string
                      02  056C
          00000007 00000004  056F             .LONG   4,7     ; 4 , 7 limits on input
                      88  0577                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  0578                .WORD   CR11$B_BR               ; Byte CR11$B_BR to data
                      00  057A                .BYTE   0               ; 0 CR11$B_BR to data
                      08  057B                .BYTE   8               ; 8 of data field
                      81  057C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          057D    256 DEQNA:  $DS_DEQNA                       ;               [19]
                          057D                .SAVE   LOCAL_BLOCK
                          057D                .PSECT  $ABS$,ABS
                00000032  0041                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DEQNA$L_CSR, DEQNA$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,DEQNA$K_LEN, DEQNA$K_LEN:
                      0000057D                .RESTORE
          41 4E 51 45 44 00' 057D             .ASCIC  "DEQNA" ; DEQNA name
                      05  057D
                      36  0583                .BYTE   DEQNA$K_LEN             ; DEQNA$K_LEN of resultant P-table
                      00  0584                .BYTE   0               ; Maximum unit number                   ;G:65
                    0000  0585                .WORD   ^A""    ; Two character mnemonic for QIO DEQNA
                      80  0587                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  0588                .Byte   Pd$_Name        ; Directive op-code
                      02  0589                .Byte   Ptd$M_Controller                ; Enforced XQ codes
                51 58 00' 058A                .Ascic  "XQ"    ; Enforced device name prefix
                      02  058A
                      83  058D                .BYTE   Pd$_Octal       ; Indicate scan octal
             52 53 43 00' 058E                .ASCIC  "CSR"   ; CSR string
                      03  058E
          0003FFFC 0003E000  0592             .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                      88  059A                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  059B                .WORD   DEQNA$L_CSR             ; Byte DEQNA$L_CSR to data
                      00  059D                .BYTE   0               ; 0 DEQNA$L_CSR to data
                      20  059E                .BYTE   32              ; 32 of data field
                      88  059F                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  05A0                .WORD   Hp$A_Device             ; Byte Hp$A_Device to data
                      00  05A2                .BYTE   0               ; 0 Hp$A_Device to data
                      12  05A3                .BYTE   18              ; 18 of data field
                      81  05A4                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          05A5    257 DHU11:  $DS_DHU11                       ;               [29]
          31 31 55 48 44 00' 05A5             .ASCIC  "DHU11" ; DHU11 name
                      05  05A5
                      39  05AB                .BYTE   HP$K_DHUV_LEN           ; HP$K_DHUV_LEN of resultant P-table
                      08  05AC                .BYTE   8               ; Maximum unit number                   ;G:65
                    5854  05AD                .WORD   ^A"TX"  ; Two character mnemonic for QIO DHU11 TX
                      80  05AF                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  05B0                .Byte   Pd$_Name        ; Directive op-code
                      02  05B1                .Byte   Ptd$M_Controller                ; Enforced TX codes
                58 54 00' 05B2                .Ascic  "TX"    ; Enforced device name prefix
                      02  05B2
                      83  05B5                .BYTE   Pd$_Octal       ; Indicate scan octal
             52 53 43 00' 05B6                .ASCIC  "CSR"   ; CSR string
                      03  05B6
          0003FFFE 0003E000  05BA             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  05C2                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  05C3                .WORD   HP$L_DHUV_CSR           ; Byte HP$L_DHUV_CSR to data
```

```
                      00  05C5                .BYTE   0                       ; 0 HP$L_DHUV_CSR to data
                      20  05C6                .BYTE   32                      ; 32 of data field
                      88  05C7                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  05C8                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  05CA                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  05CB                .BYTE   18                      ; 18 of data field
                      83  05CC                .BYTE   Pd$_Octal               ; Indicate scan octal
          52 4F 54 43 45 56 00' 05CD          .ASCIC  "VECTOR"                ; VECTOR string
                      06  05CD
     000001FE 00000000    05D4                .LONG   ^O0,^O776               ; 0 , 776 limits on input
                      88  05DC                .BYTE   Pd$_Store               ; Indicate store operation
                    0036  05DD                .WORD   HP$W_DHUV_VEC                   ; Byte HP$W_DHUV_VEC to data
                      00  05DF                .BYTE   0                       ; 0 HP$W_DHUV_VEC to data
                      10  05E0                .BYTE   16                      ; 16 of data field
                      82  05E1                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                 52 42 00' 05E2               .ASCIC  "BR"    ; BR string
                      02  05E2
     00000007 00000004    05E5                .LONG   4,7     ; 4 , 7 limits on input
                      88  05ED                .BYTE   Pd$_Store               ; Indicate store operation
                    0038  05EE                .WORD   HP$B_DHUV_BR                    ; Byte HP$B_DHUV_BR to data
                      00  05F0                .BYTE   0                       ; 0 HP$B_DHUV_BR to data
                      08  05F1                .BYTE   8                       ; 8 of data field
                      81  05F2                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          05F3    258 DHV11:  $DS_DHV11                               ;               [32]
          31 31 56 48 44 00' 05F3             .ASCIC  "DHV11" ; DHV11 name
                      05  05F3
                      39  05F9                .BYTE   HP$K_DHUV_LEN           ; HP$K_DHUV_LEN of resultant P-table
                      08  05FA                .BYTE   8                       ; Maximum unit number                   ;G:65
                    5854  05FB                .WORD   ^A"TX"  ; Two character mnemonic for QIO DHV11 TX
                      80  05FD                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  05FE                .Byte   Pd$_Name                ; Directive op-code
                      02  05FF                .Byte   Ptd$M_Controller                ; Enforced TX codes
                 58 54 00' 0600               .Ascic  "TX"    ; Enforced device name prefix
                      02  0600
                      83  0603                .BYTE   Pd$_Octal               ; Indicate scan octal
                 52 53 43 00' 0604            .ASCIC  "CSR"   ; CSR string
                      03  0604
     0003FFFE 0003E000    0608                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  0610                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  0611                .WORD   HP$L_DHUV_CSR                   ; Byte HP$L_DHUV_CSR to data
                      00  0613                .BYTE   0                       ; 0 HP$L_DHUV_CSR to data
                      20  0614                .BYTE   32                      ; 32 of data field
                      88  0615                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  0616                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  0618                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  0619                .BYTE   18                      ; 18 of data field
                      83  061A                .BYTE   Pd$_Octal               ; Indicate scan octal
          52 4F 54 43 45 56 00' 061B          .ASCIC  "VECTOR"                ; VECTOR string
                      06  061B
     000001FE 00000000    0622                .LONG   ^O0,^O776               ; 0 , 776 limits on input
                      88  062A                .BYTE   Pd$_Store               ; Indicate store operation
                    0036  062B                .WORD   HP$W_DHUV_VEC                   ; Byte HP$W_DHUV_VEC to data
                      00  062D                .BYTE   0                       ; 0 HP$W_DHUV_VEC to data
                      10  062E                .BYTE   16                      ; 16 of data field
                      82  062F                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                 52 42 00' 0630               .ASCIC  "BR"    ; BR string
                      02  0630
```

```
                00000007 00000004  0633                .LONG   4,7     ; 4 , 7 limits on input
                              88   063B                .BYTE   Pd$_Store       ; Indicate store operation
                            0038   063C                .WORD   HP$B_DHUV_BR            ; Byte HP$B_DHUV_BR to data
                              00   063E                .BYTE   0               ; 0 HP$B_DHUV_BR to data
                              08   063F                .BYTE   8               ; 8 of data field
                              81   0640                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                   0641    259 DISK:    $DS_DISK                        ;               [18]
                                   0641                .SAVE   LOCAL_BLOCK
                                   0641                .PSECT  $ABS$,ABS
                          00000032 0041                .=HP$A_DEPENDENT
                                   0032                .IIF    NB,HP$K_DISK_LEN,       HP$K_DISK_LEN:
                                   0032                .IIF    NB,DISK$K_LEN,  DISK$K_LEN:
                          00000641                     .RESTORE
                   4B 53 49 44 00' 0641                .ASCIC  "DISK"  ; DISK name
                              04   0641
                              32   0646                .BYTE   DISK$K_LEN              ; DISK$K_LEN of resultant P-table
                              FF   0647                .BYTE   255             ; Maximum unit number                   ;G:65
                            0000   0648                .WORD   ^A""    ; Two character mnemonic for QIO DISK
                              80   064A                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              8D   064B                .Byte   Pd$_Name        ; Directive op-code
                              03   064C                .Byte   Ptd$M_DEVICE            ; Enforced DU codes
                          55 44 00' 064D                .Ascic  "DU"    ; Enforced device name prefix
                              02   064D
                              81   0650                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                   0651    260 DL11:    $DS_DL11
                                   0651                .SAVE   LOCAL_BLOCK
                                   0651                .PSECT  $ABS$,ABS
                          00000032 0041                .=HP$A_DEPENDENT
                                   0032                .IIF    NB,DL11$L_CSR,  DL11$L_CSR:
                          00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                                   0036                .IIF    NB,DL11$B_BR,   DL11$B_BR:
                          00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                                   0037                .IIF    NB,DL11$K_LEN,  DL11$K_LEN:
                          00000651                     .RESTORE
                31 31 4C 44 00' 0651                .ASCIC  "DL11"  ; DL11 name
                              04   0651
                              37   0656                .BYTE   DL11$K_LEN              ; DL11$K_LEN of resultant P-table
                              00   0657                .BYTE   0               ; Maximum unit number                   ;G:65
                            0000   0658                .WORD   ^A""    ; Two character mnemonic for QIO DL11
                              80   065A                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              8D   065B                .Byte   Pd$_Name        ; Directive op-code
                              02   065C                .Byte   Ptd$M_Controller                ; Enforced TT codes
                          54 54 00' 065D                .Ascic  "TT"    ; Enforced device name prefix
                              02   065D
                              83   0660                .BYTE   Pd$_Octal       ; Indicate scan octal
                       52 53 43 00' 0661                .ASCIC  "CSR"   ; CSR string
                              03   0661
                00003FFFE 0003E000  0665                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                              88   066D                .BYTE   Pd$_Store       ; Indicate store operation
                            0032   066E                .WORD   DL11$L_CSR              ; Byte DL11$L_CSR to data
                              00   0670                .BYTE   0               ; 0 DL11$L_CSR to data
                              20   0671                .BYTE   32              ; 32 of data field
                              88   0672                .BYTE   Pd$_Store       ; Indicate store operation
                            0018   0673                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              00   0675                .BYTE   0               ; 0 HP$A_DEVICE to data
                              12   0676                .BYTE   18              ; 18 of data field
                              83   0677                .BYTE   Pd$_Octal       ; Indicate scan octal
```

```
                52 4F 54 43 45 56 00' 0678                      .ASCIC  "VECTOR"        ; VECTOR string
                                   06 0678
                   000001FE 00000002  067F                      .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                   88  0687                      .BYTE   Pd$_Store       ; Indicate store operation
                                 0024  0688                      .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                   00  068A                      .BYTE   0               ; 0 HP$W_VECTOR to data
                                   09  068B                      .BYTE   9               ; 9 of data field
                                   82  068C                      .BYTE   Pd$_Decimal     ; Indicate scan decimal
                              52 42 00' 068D                      .ASCIC  "BR"    ; BR string
                                   02  068D
                   00000007 00000004  0690                      .LONG   4,7     ; 4 , 7 limits on input
                                   88  0698                      .BYTE   Pd$_Store       ; Indicate store operation
                                 0036  0699                      .WORD   DL11$B_BR               ; Byte DL11$B_BR to data
                                   00  069B                      .BYTE   0               ; 0 DL11$B_BR to data
                                   08  069C                      .BYTE   8               ; 8 of data field
                                   81  069D                      .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                       069E    261 DLVJ1:      $DS_DLVJ1                       ;               [19]
                                       069E                    .SAVE   LOCAL_BLOCK
                                       069E                    .PSECT  $ABS$,ABS
                              00000032  0041                    .=HP$A_DEPENDENT
                                       0032                    .IIF    NB,DLVJ1$L_CSR, DLVJ1$L_CSR:
                              00000036  0032                    .IIF    NB,..BLKL,      .BLKL   1
                                       0036                    .IIF    NB,DLVJ1$B_FLAGS,       DLVJ1$B_FLAGS:
                              00000037  0036                    .IIF    NB,..BLKB,      .BLKB   1
                                       0037                    .IIF    NB,DLVJ1$K_LEN, DLVJ1$K_LEN:
                                   0000069E                    .RESTORE
                   31 4A 56 4C 44 00' 069E                      .ASCIC  "DLVJ1" ; DLVJ1 name
                                   05  069E
                                   37  06A4                      .BYTE   DLVJ1$K_LEN             ; DLVJ1$K_LEN of resultant P-table
                                   00  06A5                      .BYTE   0               ; Maximum unit number                   ;G:65
                                 0000  06A6                      .WORD   ^A""    ; Two character mnemonic for QIO DLVJ1
                                   80  06A8                      .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                   8D  06A9                      .Byte   Pd$_Name        ; Directive op-code
                                   02  06AA                      .Byte   Ptd$M_Controller                ; Enforced TT codes
                              54 54 00' 06AB                      .Ascic  "TT"    ; Enforced device name prefix
                                   02  06AB
                                   83  06AE                      .BYTE   Pd$_Octal       ; Indicate scan octal
                              52 53 43 00' 06AF                   .ASCIC  "CSR"   ; CSR string
                                   03  06AF
                   0003FFFE 0003E000  06B3                      .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                   88  06BB                      .BYTE   Pd$_Store       ; Indicate store operation
                                 0032  06BC                      .WORD   DLVJ1$L_CSR             ; Byte DLVJ1$L_CSR to data
                                   00  06BE                      .BYTE   0               ; 0 DLVJ1$L_CSR to data
                                   20  06BF                      .BYTE   32              ; 32 of data field
                                   88  06C0                      .BYTE   Pd$_Store       ; Indicate store operation
                                 0018  06C1                      .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                   00  06C3                      .BYTE   0               ; 0 HP$A_DEVICE to data
                                   12  06C4                      .BYTE   18              ; 18 of data field
                                   83  06C5                      .BYTE   Pd$_Octal       ; Indicate scan octal
                52 4F 54 43 45 56 00' 06C6                      .ASCIC  "VECTOR"        ; VECTOR string
                                   06  06C6
                   000001FE 00000002  06CD                      .LONG   ^O2,^O776       ; 2 , 776 limits on input
                                   88  06D5                      .BYTE   Pd$_Store       ; Indicate store operation
                                 0024  06D6                      .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                   00  06D8                      .BYTE   0               ; 0 HP$W_VECTOR to data
                                   09  06D9                      .BYTE   9               ; 9 of data field
                                   85  06DA                      .BYTE   Pd$_String      ; Indicate scan and verify string
```

```
64 20 66 6F 20 72 65 62 6D 75 4E 00' 06DB                .ASCIC  "Number of data bits"   ; Number of data bits string
            73 74 69 62 20 61 74 61  06E7
                                  13  06DB
                              37 00' 06EF                .ASCIC  "7"
                                  01  06EF
                              38 00' 06F1                .ASCIC  "8"
                                  01  06F1
                                  00  06F3                .BYTE   0                ; Null length is end of valid 7,8
                                  88  06F4                .BYTE   Pd$_Store        ; Indicate store operation
                               0036   06F5                .WORD   DLVJ1$B_FLAGS            ; Byte DLVJ1$B_FLAGS to data
                                  00  06F7                .BYTE   DLVJ1$V_NDATA            ; DLVJ1$V_NDATA DLVJ1$B_FLAGS to data
                                  01  06F8                .BYTE   1                ; 1 of data field
                                  85  06F9                .BYTE   Pd$_String       ; Indicate scan and verify string
73 20 66 6F 20 72 65 62 6D 75 4E 00' 06FA                .ASCIC  "Number of stop bits"   ; Number of stop bits string
            73 74 69 62 20 70 6F 74  0706
                                  13  06FA
                              31 00' 070E                .ASCIC  "1"
                                  01  070E
                              32 00' 0710                .ASCIC  "2"
                                  01  0710
                                  00  0712                .BYTE   0                ; Null length is end of valid 1,2
                                  88  0713                .BYTE   Pd$_Store        ; Indicate store operation
                               0036   0714                .WORD   DLVJ1$B_FLAGS            ; Byte DLVJ1$B_FLAGS to data
                                  01  0716                .BYTE   DLVJ1$V_NSTOP            ; DLVJ1$V_NSTOP DLVJ1$B_FLAGS to data
                                  01  0717                .BYTE   1                ; 1 of data field
                                  8B  0718                .BYTE   Pd$_Logical      ; Indicate logical value operation
65 74 65 64 20 79 74 69 72 61 50 00' 0719                .ASCIC  \Parity detection enabled\      ; Parity detection enabled string
65 6C 62 61 6E 65 20 6E 6F 69 74 63  0725
                                  64  0731
                                  18  0719
                                  88  0732                .BYTE   Pd$_Store        ; Indicate store operation
                               0036   0733                .WORD   DLVJ1$B_FLAGS            ; Byte DLVJ1$B_FLAGS to data
                                  02  0735                .BYTE   DLVJ1$V_PARITY           ; DLVJ1$V_PARITY DLVJ1$B_FLAGS to data
                                  01  0736                .BYTE   1                ; 1 of data field
                                  8B  0737                .BYTE   Pd$_Logical      ; Indicate logical value operation
79 74 69 72 61 70 20 6E 65 76 45 00' 0738                .ASCIC  \Even parity enabled\   ; Even parity enabled string
            64 65 6C 62 61 6E 65 20  0744
                                  13  0738
                                  88  074C                .BYTE   Pd$_Store        ; Indicate store operation
                               0036   074D                .WORD   DLVJ1$B_FLAGS            ; Byte DLVJ1$B_FLAGS to data
                                  03  074F                .BYTE   DLVJ1$V_EVEPAR           ; DLVJ1$V_EVEPAR DLVJ1$B_FLAGS to data
                                  01  0750                .BYTE   1                ; 1 of data field
                                  81  0751                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                      0752    262 DMC11:  $DS_DMC11
                                      0752                .SAVE   LOCAL_BLOCK
                                      0752                .PSECT  $ABS$,ABS
                            00000032  0041                .=HP$A_DEPENDENT
                                      0032                .IIF    NB,DMC11$L_CSR, DMC11$L_CSR:
                            00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                      0036                .IIF    NB,DMC11$B_BR,  DMC11$B_BR:
                            00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                      0037                .IIF    NB,DMC11$K_LEN, DMC11$K_LEN:
                                  00000752                .RESTORE
                     31 31 43 4D 44 00' 0752                .ASCIC  "DMC11" ; DMC11 name
                                  05  0752
                                  37  0758                .BYTE   DMC11$K_LEN              ; DMC11$K_LEN of resultant P-table
                                  00  0759                .BYTE   0                ; Maximum unit number                          ;G:65
```

```
                      4D58  075A                .WORD   ^A"XM"  ; Two character mnemonic for QIO DMC11 XM
                        80  075C                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  075D                .Byte   Pd$_Name        ; Directive op-code
                        03  075E                .Byte   Ptd$M_Device            ; Enforced XM codes
                   4D 58 00' 075F                .Ascic  "XM"    ; Enforced device name prefix
                        02  075F
                        83  0762                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 53 43 00' 0763             .ASCIC  "CSR"   ; CSR string
                        03  0763
               0003FFFE 0003E000  0767          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  076F                .BYTE   Pd$_Store       ; Indicate store operation
                      0032  0770                .WORD   DMC11$L_CSR             ; Byte DMC11$L_CSR to data
                        00  0772                .BYTE   0               ; 0 DMC11$L_CSR to data
                        20  0773                .BYTE   32              ; 32 of data field
                        88  0774                .BYTE   Pd$_Store       ; Indicate store operation
                      0018  0775                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        00  0777                .BYTE   0               ; 0 HP$A_DEVICE to data
                        12  0778                .BYTE   18              ; 18 of data field
                        83  0779                .BYTE   Pd$_Octal       ; Indicate scan octal
            52 4F 54 43 45 56 00' 077A          .ASCIC  "VECTOR"        ; VECTOR string
                        06  077A
               000001FE 00000002  0781          .LONG   ^O2,^O776       ; 2 , 776 limits on input
                        88  0789                .BYTE   Pd$_Store       ; Indicate store operation
                      0024  078A                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                        00  078C                .BYTE   0               ; 0 HP$W_VECTOR to data
                        09  078D                .BYTE   9               ; 9 of data field
                        82  078E                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                  52 42 00' 078F                .ASCIC  "BR"    ; BR string
                        02  078F
            00000007 00000004  0792             .LONG   4,7     ; 4 , 7 limits on input
                        88  079A                .BYTE   Pd$_Store       ; Indicate store operation
                      0036  079B                .WORD   DMC11$B_BR              ; Byte DMC11$B_BR to data
                        00  079D                .BYTE   0               ; 0 DMC11$B_BR to data
                        08  079E                .BYTE   8               ; 8 of data field
                        81  079F                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            07A0  263 DMF32:    $DS_DMF32                       ;               [02]
                            07A0                .SAVE   LOCAL_BLOCK
                            07A0                .PSECT  $ABS$,ABS
                      00000032  0041            .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$B_DMF32_COMMON,   HP$B_DMF32_COMMON:
                            0032                .IIF    NB,DMF32$B_COMMON,      DMF32$B_COMMON:
                      00000033  0032            .IIF    NB,,BLKB,       .BLKB   1
                            0033                .IIF    NB,HP$B_DMF32_FLAGS,    HP$B_DMF32_FLAGS:
                            0033                .IIF    NB,DMF32$B_FLAGS,       DMF32$B_FLAGS:
                      00000034  0033            .IIF    NB,,BLKB,       .BLKB   1
                            0034                .IIF    NB,HP$B_DMF32_BR,       HP$B_DMF32_BR:
                            0034                .IIF    NB,DMF32$B_BR,  DMF32$B_BR:
                      00000035  0034            .IIF    NB,,BLKB,       .BLKB   1
                            0035                .IIF    NB,HP$L_DMF32_CSR,      HP$L_DMF32_CSR:
                            0035                .IIF    NB,DMF32$L_CSR, DMF32$L_CSR:
                      00000039  0035            .IIF    NB,,BLKL,       .BLKL   1
                            0039                .IIF    NB,HP$K_DMF32_LEN,      HP$K_DMF32_LEN:
                            0039                .IIF    NB,DMF32$K_LEN, DMF32$K_LEN:
                        000007A0                .RESTORE
            32 33 46 4D 44 00' 07A0             .ASCIC  "DMF32" ; DMF32 name
                        05  07A0
                        39  07A6                .BYTE   DMF32$K_LEN             ; DMF32$K_LEN of resultant P-table
```

```
                    05  07A7          .BYTE   5                   ; Maximum unit number                    ;G:65
                  434C  07A8          .WORD   ^A"LC"   ; Two character mnemonic for QIO DMF32 LC
                    80  07AA          .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                    83  07AB          .BYTE   Pd$_Octal           ; Indicate scan octal
          52 53 43 00'  07AC          .ASCIC  "CSR"    ; CSR string
                    03  07AC
     0003FFFE 0003E000  07B0          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                    88  07B8          .BYTE   Pd$_Store           ; Indicate store operation
                  0035  07B9          .WORD   DMF32$L_CSR             ; Byte DMF32$L_CSR to data
                    00  07BB          .BYTE   0                   ; 0 DMF32$L_CSR to data
                    20  07BC          .BYTE   32                  ; 32 of data field
                    88  07BD          .BYTE   Pd$_Store           ; Indicate store operation
                  0018  07BE          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  07C0          .BYTE   0                   ; 0 HP$A_DEVICE to data
                    12  07C1          .BYTE   18                  ; 18 of data field
                    83  07C2          .BYTE   Pd$_Octal           ; Indicate scan octal
  72 6F 74 63 65 56 00'  07C3          .ASCIC  "Vector"            ; Vector string
                    06  07C3
     000001FE 000000C0  07CA          .LONG   ^O300,^O776         ; 300 , 776 limits on input
                    88  07D2          .BYTE   Pd$_Store           ; Indicate store operation
                  0024  07D3          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  07D5          .BYTE   0                   ; 0 HP$W_VECTOR to data
                    09  07D6          .BYTE   9                   ; 9 of data field
                    82  07D7          .BYTE   Pd$_Decimal         ; Indicate scan decimal
              52 42 00'  07D8          .ASCIC  "BR"     ; BR string
                    02  07D8
     00000007 00000004  07DB          .LONG   4,7      ; 4 , 7 limits on input
                    88  07E3          .BYTE   Pd$_Store           ; Indicate store operation
                  0034  07E4          .WORD   DMF32$B_BR              ; Byte DMF32$B_BR to data
                    00  07E6          .BYTE   0                   ; 0 DMF32$B_BR to data
                    08  07E7          .BYTE   8                   ; 8 of data field
                    86  07E8          .BYTE   Pd$_Literal         ; Indicate literal
              00000001  07E9          .LONG   1                   ; Constant
                    88  07ED          .BYTE   Pd$_Store           ; Indicate store operation
                  0032  07EE          .WORD   DMF32$B_COMMON          ; Byte DMF32$B_COMMON to data
                    00  07F0          .BYTE   0                   ; 0 DMF32$B_COMMON to data
                    08  07F1          .BYTE   8                   ; 8 of data field
                    81  07F2          .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                        07F3   264 DMF32A: $DS_DMF32A                   ;                [02]
                        07F3          .SAVE   LOCAL_BLOCK
                        07F3          .PSECT  $ABS$,ABS
              00000032  0041          .=HP$A_DEPENDENT
                        0032          .IIF    NB,HP$B_DMF32A_BR,      HP$B_DMF32A_BR:
                        0032          .IIF    NB,DMF32A$B_BR, DMF32A$B_BR:
              00000033  0032          .IIF    NB,.BLKB,       .BLKB   1
                        0033          .IIF    NB,HP$B_DMF32A_ACTIV,   HP$B_DMF32A_ACTIV:
                        0033          .IIF    NB,DMF32A$B_ACTIV,      DMF32A$B_ACTIV:
              00000034  0033          .IIF    NB,.BLKB,       .BLKB   1
                        0034          .IIF    NB,HP$B_DMF32A_SPEED,   HP$B_DMF32A_SPEED:
                        0034          .IIF    NB,DMF32A$B_SPEED,      DMF32A$B_SPEED:
              00000035  0034          .IIF    NB,.BLKB,       .BLKB   1
                        0035          .IIF    NB,HP$W_DMF32A_FLAGS,   HP$W_DMF32A_FLAGS:
                        0035          .IIF    NB,DMF32A$W_FLAGS,      DMF32A$W_FLAGS:
              00000037  0035          .IIF    NB,.BLKW,       .BLKW   1
                        0037          .IIF    NB,HP$B_DMF32A_JUMP,    HP$B_DMF32A_JUMP:
                        0037          .IIF    NB,DMF32A$B_JUMP,       DMF32A$B_JUMP:
              00000038  0037          .IIF    NB,.BLKB,       .BLKB   1
```

```
                            0038                .IIF    NB,HP$K_DMF32A_LEN,     HP$K_DMF32A_LEN:
                            0038                .IIF    NB,DMF32A$K_LEN,        DMF32A$K_LEN:
                        000007F3                .RESTORE
          41 32 33 46 4D 44 00' 07F3            .ASCIC  "DMF32A"            ; DMF32A name
                              06  07F3
                              38  07FA            .BYTE   DMF32A$K_LEN                ; DMF32A$K_LEN of resultant P-table
                              00  07FB            .BYTE   0                   ; Maximum unit number                       ;G:65
                            5854  07FC            .WORD   ^A"TX"  ; Two character mnemonic for QIO DMF32A TX
                              80  07FE            .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                              8D  07FF            .Byte   Pd$_Name            ; Directive op-code
                              02  0800            .Byte   Ptd$M_Controller                ; Enforced TX codes
                       58 54 00' 0801            .Ascic  "TX"    ; Enforced device name prefix
                              02  0801
                              83  0804            .BYTE   Pd$_Octal           ; Indicate scan octal
                    52 53 43 00' 0805            .ASCIC  "CSR"   ; CSR string
                              03  0805
             0003FFFE 0003E000  0809            .LONG   ^O00760000,^O00777776       ; 0760000 , 0777776 limits on input
                              88  0811            .BYTE   Pd$_Store           ; Indicate store operation
                            0018  0812            .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                              00  0814            .BYTE   0                   ; 0 HP$A_DEVICE to data
                              12  0815            .BYTE   18                  ; 18 of data field
                              83  0816            .BYTE   Pd$_Octal           ; Indicate scan octal
          72 6F 74 63 65 56 00' 0817            .ASCIC  "Vector"            ; Vector string
                              06  0817
             000001FE 000000C0  081E            .LONG   ^O300,^O776         ; 300 , 776 limits on input
                              88  0826            .BYTE   Pd$_Store           ; Indicate store operation
                            0024  0827            .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                              00  0829            .BYTE   0                   ; 0 HP$W_VECTOR to data
                              09  082A            .BYTE   9                   ; 9 of data field
                              82  082B            .BYTE   Pd$_Decimal         ; Indicate scan decimal
                       52 42 00' 082C            .ASCIC  "BR"    ; BR string
                              02  082C
                    00000007 00000004  082F            .LONG   4,7     ; 4 , 7 limits on input
                              88  0837            .BYTE   Pd$_Store           ; Indicate store operation
                            0032  0838            .WORD   DMF32A$B_BR                 ; Byte DMF32A$B_BR to data
                              00  083A            .BYTE   0                   ; 0 DMF32A$B_BR to data
                              08  083B            .BYTE   8                   ; 8 of data field
                              83  083C            .BYTE   Pd$_Octal           ; Indicate scan octal
65 6E 69 4C 20 65 76 69 74 63 41 00' 083D            .ASCIC  "Active Lines"      ; Active Lines string
                              73  0849
                              0C  083D
                    000000FF 00000001  084A            .LONG   ^O1,^O377           ; 1 , 377 limits on input
                              88  0852            .BYTE   Pd$_Store           ; Indicate store operation
                            0033  0853            .WORD   DMF32A$B_ACTIV              ; Byte DMF32A$B_ACTIV to data
                              00  0855            .BYTE   0                   ; 0 DMF32A$B_ACTIV to data
                              08  0856            .BYTE   8                   ; 8 of data field
                              85  0857            .BYTE   Pd$_String          ; Indicate scan and verify string
    65 74 61 52 20 64 75 61 42 00' 0858            .ASCIC  "Baud Rate"         ; Baud Rate string
                              09  0858
                       30 35 00' 0862            .ASCIC  "50"
                              02  0862
                       35 37 00' 0865            .ASCIC  "75"
                              02  0865
                    30 31 31 00' 0868            .ASCIC  "110"
                              03  0868
                 35 2E 34 33 31 00' 086C            .ASCIC  "134.5"
                              05  086C
```

```
                    30 35 31 00' 0872               .ASCIC  "150"
                       03  0872
                    30 30 33 00' 0876               .ASCIC  "300"
                       03  0876
                    30 30 36 00' 087A               .ASCIC  "600"
                       03  087A
                 30 30 32 31 00' 087E               .ASCIC  "1200"
                       04  087E
                 30 30 38 31 00' 0883               .ASCIC  "1800"
                       04  0883
                 30 30 30 32 00' 0888               .ASCIC  "2000"
                       04  0888
                 30 30 34 32 00' 088D               .ASCIC  "2400"
                       04  088D
                 30 30 36 33 00' 0892               .ASCIC  "3600"
                       04  0892
                 30 30 38 34 00' 0897               .ASCIC  "4800"
                       04  0897
                 30 30 32 37 00' 089C               .ASCIC  "7200"
                       04  089C
                 30 30 36 39 00' 08A1               .ASCIC  "9600"
                       04  08A1
              30 30 32 39 31 00' 08A6               .ASCIC  "19200"
                       05  08A6
                       00  08AC                    .BYTE   0               ; Null length is end of valid 50,75,110,134.5,150,30
                       88  08AD                    .BYTE   Pd$_Store       ; Indicate store operation
                     0034  08AE                    .WORD   DMF32A$B_SPEED          ; Byte DMF32A$B_SPEED to data
                       00  08B0                    .BYTE   0               ; 0 DMF32A$B_SPEED to data
                       08  08B1                    .BYTE   8               ; 8 of data field
                       85  08B2                    .BYTE   Pd$_String      ; Indicate scan and verify string
79 54 20 6B 63 61 62 70 6F 6F 4C 00' 08B3          .ASCIC  "Loopback Type" ; Loopback Type string
                          65 70  08BF
                       0D  08B3
          4C 41 4E 52 45 54 4E 49 00' 08C1         .ASCIC  "INTERNAL"
                       08  08C1
                    38 34 32 33 48 00' 08CA         .ASCIC  "H3248"
                       05  08CA
4D 45 44 4F 4D 5F 4C 41 43 4F 4C 00' 08D0          .ASCIC  "LOCAL_MODEM"
                       0B  08D0
4C 42 41 4D 4D 41 52 47 4F 52 50 00' 08DC          .ASCIC  "PROGRAMMABLE_MODEM"
                 4D 45 44 4F 4D 5F 45  08E8
                       12  08DC
                    39 34 32 33 48 00' 08EF         .ASCIC  "H3249"
                       05  08EF
                       00  08F5                    .BYTE   0               ; Null length is end of valid INTERNAL,<H3248>,LOCAL
                       88  08F6                    .BYTE   Pd$_Store       ; Indicate store operation
                     0035  08F7                    .WORD   DMF32A$W_FLAGS          ; Byte DMF32A$W_FLAGS to data
                       00  08F9                    .BYTE   0               ; 0 DMF32A$W_FLAGS to data
                       05  08FA                    .BYTE   5               ; 5 of data field
                       8B  08FB                    .BYTE   Pd$_Logical     ; Indicate logical value operation
74 69 6E 49 20 73 75 62 69 6E 55 00' 08FC          .ASCIC  \Unibus Init Jumper\    ; Unibus Init Jumper string
                 72 65 70 6D 75 4A 20  0908
                       12  08FC
                       88  090F                    .BYTE   Pd$_Store       ; Indicate store operation
                     0037  0910                    .WORD   DMF32A$B_JUMP           ; Byte DMF32A$B_JUMP to data
                       00  0912                    .BYTE   0               ; 0 DMF32A$B_JUMP to data
                       01  0913                    .BYTE   1               ; 1 of data field
```

```
                    81  0914                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                        0915   265 DMF32P: $DS_DMF32P                        ;                [02]
                        0915                .SAVE   LOCAL_BLOCK
                        0915                .PSECT  $ABS$,ABS
              00000032  0041                .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$B_DMF32P_COMMON,  HP$B_DMF32P_COMMON:
                        0032                .IIF    NB,DMF32P$B_COMMON,     DMF32P$B_COMMON:
              00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                        0033                .IIF    NB,HP$B_DMF32P_FLAGS,   HP$B_DMF32P_FLAGS:
                        0033                .IIF    NB,DMF32P$B_FLAGS,      DMF32P$B_FLAGS:
              00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                        0034                .IIF    NB,HP$B_DMF32P_BR,      HP$B_DMF32P_BR:
                        0034                .IIF    NB,DMF32P$B_BR, DMF32P$B_BR:
              00000035  0034                .IIF    NB,.BLKB,       .BLKB   1
                        0035                .IIF    NB,HP$L_DMF32P_CSR,     HP$L_DMF32P_CSR:
                        0035                .IIF    NB,DMF32P$L_CSR,        DMF32P$L_CSR:
              00000039  0035                .IIF    NB,.BLKL,       .BLKL   1
                        0039                .IIF    NB,HP$K_DMF32P_LEN,     HP$K_DMF32P_LEN:
                        0039                .IIF    NB,DMF32P$K_LEN,        DMF32P$K_LEN:
                    00000915                .RESTORE
   50 32 33 46 4D 44 00' 0915                .ASCIC  "DMF32P"        ; DMF32P name
                    06  0915
                    39  091C                .BYTE   DMF32P$K_LEN            ; DMF32P$K_LEN of resultant P-table
                    00  091D                .BYTE   0               ; Maximum unit number                    ;G:65
                  434C  091E                .WORD   ^A"LC"  ; Two character mnemonic for QIO DMF32P LC
                    80  0920                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  0921                .Byte   Pd$_Name        ; Directive op-code
                    02  0922                .Byte   Ptd$M_Controller                ; Enforced LC codes
              43 4C 00' 0923                .Ascic  "LC"    ; Enforced device name prefix
                    02  0923
                    83  0926                .BYTE   Pd$_Octal       ; Indicate scan octal
           52 53 43 00' 0927                .ASCIC  "CSR"   ; CSR string
                    03  0927
     0003FFFE 0003E000  092B                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                    88  0933                .BYTE   Pd$_Store       ; Indicate store operation
                  0035  0934                .WORD   DMF32P$L_CSR            ; Byte DMF32P$L_CSR to data
                    00  0936                .BYTE   0               ; 0 DMF32P$L_CSR to data
                    20  0937                .BYTE   32              ; 32 of data field
                    88  0938                .BYTE   Pd$_Store       ; Indicate store operation
                  0018  0939                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  093B                .BYTE   0               ; 0 HP$A_DEVICE to data
                    12  093C                .BYTE   18              ; 18 of data field
                    83  093D                .BYTE   Pd$_Octal       ; Indicate scan octal
   72 6F 74 63 65 56 00' 093E                .ASCIC  "Vector"        ; Vector string
                    06  093E
     000001FE 000000C0  0945                .LONG   ^O300,^O776     ; 300 , 776 limits on input
                    88  094D                .BYTE   Pd$_Store       ; Indicate store operation
                  0024  094E                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  0950                .BYTE   0               ; 0 HP$W_VECTOR to data
                    09  0951                .BYTE   9               ; 9 of data field
                    82  0952                .BYTE   Pd$_Decimal     ; Indicate scan decimal
              52 42 00' 0953                .ASCIC  "BR"    ; BR string
                    02  0953
     00000007 00000004  0956                .LONG   4,7     ; 4 , 7 limits on input
                    88  095E                .BYTE   Pd$_Store       ; Indicate store operation
                  0034  095F                .WORD   DMF32P$B_BR             ; Byte DMF32P$B_BR to data
                    00  0961                .BYTE   0               ; 0 DMF32P$B_BR to data
```

```
                                  08  0962                .BYTE   8                   ; 8 of data field
                                  85  0963                .BYTE   Pd$_String          ; Indicate scan and verify string
65 63 69 76 65 44 20 74 72 6F 50 00' 0964                .ASCIC  "Port Device"       ; Port Device string
                                  0B  0964
                 52 45 48 54 4F 00' 0970                .ASCIC  "OTHER"
                                  05  0970
                           50 4C 00' 0976                .ASCIC  "LP"
                                  02  0976
                     50 41 52 57 00' 0979                .ASCIC  "WRAP"
                                  04  0979
                                  00  097E                .BYTE   0                   ; Null length is end of valid OTHER,LP,WRAP
                                  88  097F                .BYTE   Pd$_Store           ; Indicate store operation
                                0033  0980                .WORD   DMF32P$B_FLAGS              ; Byte DMF32P$B_FLAGS to data
                                  00  0982                .BYTE   0                   ; 0 DMF32P$B_FLAGS to data
                                  08  0983                .BYTE   8                   ; 8 of data field
                                  86  0984                .BYTE   Pd$_Literal         ; Indicate literal
                           00000001  0985                .LONG   1                   ; Constant
                                  88  0989                .BYTE   Pd$_Store           ; Indicate store operation
                                0032  098A                .WORD   DMF32P$B_COMMON             ; Byte DMF32P$B_COMMON to data
                                  00  098C                .BYTE   0                   ; 0 DMF32P$B_COMMON to data
                                  08  098D                .BYTE   8                   ; 8 of data field
                                  81  098E                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                                      098F    266 DMF32S: $DS_DMF32S                          ;                [02]
                                      098F                .SAVE   LOCAL_BLOCK
                                      098F                .PSECT  $ABS$,ABS
                           00000032  0041                .=HP$A_DEPENDENT
                                      0032                .IIF    NB,HP$B_DMF32S_COMMON,  HP$B_DMF32S_COMMON:
                                      0032                .IIF    NB,DMF32S$B_COMMON,     DMF32S$B_COMMON:
                           00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                                      0033                .IIF    NB,HP$B_DMF32S_FLAGS,   HP$B_DMF32S_FLAGS:
                                      0033                .IIF    NB,DMF32S$B_FLAGS,      DMF32S$B_FLAGS:
                           00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                                      0034                .IIF    NB,HP$B_DMF32S_BR,      HP$B_DMF32S_BR:
                                      0034                .IIF    NB,DMF32S$B_BR, DMF32S$B_BR:
                           00000035  0034                .IIF    NB,.BLKB,       .BLKB   1
                                      0035                .IIF    NB,HP$L_DMF32S_CSR,     HP$L_DMF32S_CSR:
                                      0035                .IIF    NB,DMF32S$L_CSR,        DMF32S$L_CSR:
                           00000039  0035                .IIF    NB,.BLKL,       .BLKL   1
                                      0039                .IIF    NB,HP$K_DMF32S_LEN,     HP$K_DMF32S_LEN:
                                      0039                .IIF    NB,DMF32S$K_LEN,        DMF32S$K_LEN:
                                0000098F                .RESTORE
             53 32 33 46 4D 44 00' 098F                .ASCIC  "DMF32S"            ; DMF32S name
                                  06  098F
                                  39  0996                .BYTE   DMF32$K_LEN                 ; DMF32$K_LEN of resultant P-table
                                  01  0997                .BYTE   1                   ; Maximum unit number                    ;G:65
                                4758  0998                .WORD   ^A"XG"  ; Two character mnemonic for QIO DMF32S XG
                                  80  099A                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                                  8D  099B                .Byte   Pd$_Name            ; Directive op-code
                                  03  099C                .Byte   Ptd$M_Device                ; Enforced XG codes
                           47 58 00' 099D                .Ascic  "XG"    ; Enforced device name prefix
                                  02  099D
                                  83  09A0                .BYTE   Pd$_Octal           ; Indicate scan octal
                        52 53 43 00' 09A1                .ASCIC  "CSR"   ; CSR string
                                  03  09A1
                 0003FFFE 0003E000  09A5                .LONG   ^O760000,^O777776           ; 760000 , 777776 limits on input
                                  88  09AD                .BYTE   Pd$_Store           ; Indicate store operation
                                0035  09AE                .WORD   DMF32S$L_CSR                ; Byte DMF32S$L_CSR to data
```

```
                                 00  09B0          .BYTE   0                 ; 0 DMF32S$L_CSR to data
                                 20  09B1          .BYTE   32                ; 32 of data field
                                 88  09B2          .BYTE   Pd$_Store         ; Indicate store operation
                               0018  09B3          .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                                 00  09B5          .BYTE   0                 ; 0 HP$A_DEVICE to data
                                 12  09B6          .BYTE   18                ; 18 of data field
                                 83  09B7          .BYTE   Pd$_Octal         ; Indicate scan octal
              72 6F 74 63 65 56 00' 09B8          .ASCIC  "Vector"          ; Vector string
                                 06  09B8
                 000001FE 00000002  09BF          .LONG   ^O2,^O776         ; 2 , 776 limits on input
                                 88  09C7          .BYTE   Pd$_Store         ; Indicate store operation
                               0024  09C8          .WORD   HP$W_VECTOR               ; Byte HP$W_VECTOR to data
                                 00  09CA          .BYTE   0                 ; 0 HP$W_VECTOR to data
                                 09  09CB          .BYTE   9                 ; 9 of data field
                                 82  09CC          .BYTE   Pd$_Decimal       ; Indicate scan decimal
                           52 42 00' 09CD          .ASCIC  "BR"     ; BR string
                                 02  09CD
                 00000007 00000004  09D0          .LONG   4,7      ; 4 , 7 limits on input
                                 88  09D8          .BYTE   Pd$_Store         ; Indicate store operation
                               0034  09D9          .WORD   DMF32S$B_BR               ; Byte DMF32S$B_BR to data
                                 00  09DB          .BYTE   0                 ; 0 DMF32S$B_BR to data
                                 08  09DC          .BYTE   8                 ; 8 of data field
                                 85  09DD          .BYTE   Pd$_String        ; Indicate scan and verify string
72 57 20 6C 61 6E 72 65 74 78 45 00' 09DE          .ASCIC  "External Wrap"   ; External Wrap string
                              70 61  09EA
                                 0D  09DE
                    45 4E 4F 4E 00' 09EC          .ASCIC  "NONE"
                                 04  09EC
                 4D 45 44 4F 4D 00' 09F1          .ASCIC  "MODEM"
                                 05  09F1
                    50 41 52 57 00' 09F7          .ASCIC  "WRAP"
                                 04  09F7
                                 00  09FC          .BYTE   0                 ; Null length is end of valid NONE,MODEM,WRAP
                                 88  09FD          .BYTE   Pd$_Store         ; Indicate store operation
                               0033  09FE          .WORD   DMF32S$B_FLAGS            ; Byte DMF32S$B_FLAGS to data
                                 00  0A00          .BYTE   0                 ; 0 DMF32S$B_FLAGS to data
                                 02  0A01          .BYTE   2                 ; 2 of data field
                                 81  0A02          .BYTE   Pd$_End           ; Indicate end of PT-descriptor
                                     0A03  267 DMP11:  $DS_DMP11
                                     0A03          .SAVE   LOCAL_BLOCK
                                     0A03          .PSECT  $ABS$,ABS
                           00000032  0041          .=HP$A_DEPENDENT
                                     0032          .IIF    NB,DMP11$B_COMMON,      DMP11$B_COMMON:
                           00000033  0032          .IIF    NB,.BLKB,        .BLKB   1
                                     0033          .IIF    NB,DMP11$B_FLAGS,       DMP11$B_FLAGS:
                           00000034  0033          .IIF    NB,.BLKB,        .BLKB   1
                                     0034          .IIF    NB,DMP11$B_BR,  DMP11$B_BR:
                           00000035  0034          .IIF    NB,.BLKB,        .BLKB   1
                                     0035          .IIF    NB,DMP11$L_CSR, DMP11$L_CSR:
                           00000039  0035          .IIF    NB,.BLKL,        .BLKL   1
                                     0039          .IIF    NB,DMP11$K_LEN, DMP11$K_LEN:
                                 00000A03          .RESTORE
                 31 31 50 4D 44 00' 0A03          .ASCIC  "DMP11" ; DMP11 name
                                 05  0A03
                                 39  0A09          .BYTE   DMP11$K_LEN               ; DMP11$K_LEN of resultant P-table
                                 00  0A0A          .BYTE   0                 ; Maximum unit number                    ;G:65
                               4458  0A0B          .WORD   ^A"XD"  ; Two character mnemonic for QIO DMP11 XD
```

```
                            80  0A0D                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                            8D  0A0E                .Byte   Pd$_Name        ; Directive op-code
                            03  0A0F                .Byte   Ptd$M_Device            ; Enforced XD codes
                     44 58 00'  0A10                .Ascic  "XD"    ; Enforced device name prefix
                            02  0A10
                            83  0A13                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 53 43 00'  0A14                .ASCIC  "CSR"   ; CSR string
                            03  0A14
               0003FFFE 0003E000  0A18                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                            88  0A20                .BYTE   Pd$_Store       ; Indicate store operation
                          0035  0A21                .WORD   DMP11$L_CSR             ; Byte DMP11$L_CSR to data
                            00  0A23                .BYTE   0               ; 0 DMP11$L_CSR to data
                            20  0A24                .BYTE   32              ; 32 of data field
                            88  0A25                .BYTE   Pd$_Store       ; Indicate store operation
                          0018  0A26                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                            00  0A28                .BYTE   0               ; 0 HP$A_DEVICE to data
                            12  0A29                .BYTE   18              ; 18 of data field
                            83  0A2A                .BYTE   Pd$_Octal       ; Indicate scan octal
            52 4F 54 43 45 56 00'  0A2B                .ASCIC  "VECTOR"        ; VECTOR string
                            06  0A2B
               000001FE 00000002  0A32                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                            88  0A3A                .BYTE   Pd$_Store       ; Indicate store operation
                          0024  0A3B                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                            00  0A3D                .BYTE   0               ; 0 HP$W_VECTOR to data
                            09  0A3E                .BYTE   9               ; 9 of data field
                            82  0A3F                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                     52 42 00'  0A40                .ASCIC  "BR"    ; BR string
                            02  0A40
               00000007 00000004  0A43                .LONG   4,7     ; 4 , 7 limits on input
                            88  0A4B                .BYTE   Pd$_Store       ; Indicate store operation
                          0034  0A4C                .WORD   DMP11$B_BR              ; Byte DMP11$B_BR to data
                            00  0A4E                .BYTE   0               ; 0 DMP11$B_BR to data
                            08  0A4F                .BYTE   8               ; 8 of data field
                            85  0A50                .BYTE   Pd$_String      ; Indicate scan and verify string
61 74 53 20 6C 6F 72 74 6E 6F 43 00'  0A51                .ASCIC  "Control Station"       ; Control Station string
                   6E 6F 69 74  0A5D
                            0F  0A51
                        4F 4E 00'  0A61                .ASCIC  "NO"
                            02  0A61
                     53 45 59 00'  0A64                .ASCIC  "YES"
                            03  0A64
                            00  0A68                .BYTE   0               ; Null length is end of valid NO,YES
                            88  0A69                .BYTE   Pd$_Store       ; Indicate store operation
                          0033  0A6A                .WORD   DMP11$B_FLAGS           ; Byte DMP11$B_FLAGS to data
                            00  0A6C                .BYTE   0               ; 0 DMP11$B_FLAGS to data
                            01  0A6D                .BYTE   1               ; 1 of data field
                            85  0A6E                .BYTE   Pd$_String      ; Indicate scan and verify string
      79 72 61 74 75 62 69 72 54 00'  0A6F                .ASCIC  "Tributary"     ; Tributary string
                            09  0A6F
                        4F 4E 00'  0A79                .ASCIC  "NO"
                            02  0A79
                     53 45 59 00'  0A7C                .ASCIC  "YES"
                            03  0A7C
                            00  0A80                .BYTE   0               ; Null length is end of valid NO,YES
                            88  0A81                .BYTE   Pd$_Store       ; Indicate store operation
                          0033  0A82                .WORD   DMP11$B_FLAGS           ; Byte DMP11$B_FLAGS to data
                            01  0A84                .BYTE   1               ; 1 DMP11$B_FLAGS to data
```

```
                            01  0A85                .BYTE   1                   ; 1 of data field
                            85  0A86                .BYTE   Pd$_String          ; Indicate scan and verify string
69 6C 69 62 61 74 61 70 6D 6F 43 00' 0A87           .ASCIC  "Compatability Mode"    ; Compatability Mode string
                65 64 6F 4D 20 79 74  0A93
                            12  0A87
                         4F 4E 00'  0A9A            .ASCIC  "NO"
                            02  0A9A
                      53 45 59 00'  0A9D            .ASCIC  "YES"
                            03  0A9D
                            00  0AA1                .BYTE   0                   ; Null length is end of valid NO,YES
                            88  0AA2                .BYTE   Pd$_Store           ; Indicate store operation
                          0033  0AA3                .WORD   DMP11$B_FLAGS               ; Byte DMP11$B_FLAGS to data
                            02  0AA5                .BYTE   2                   ; 2 DMP11$B_FLAGS to data
                            01  0AA6                .BYTE   1                   ; 1 of data field
                            85  0AA7                .BYTE   Pd$_String          ; Indicate scan and verify string
78 65 6C 70 75 44 20 66 6C 61 48 00' 0AA8           .ASCIC  "Half Duplex"   ; Half Duplex string
                            0B  0AA8
                         4F 4E 00'  0AB4            .ASCIC  "NO"
                            02  0AB4
                      53 45 59 00'  0AB7            .ASCIC  "YES"
                            03  0AB7
                            00  0ABB                .BYTE   0                   ; Null length is end of valid NO,YES
                            88  0ABC                .BYTE   Pd$_Store           ; Indicate store operation
                          0033  0ABD                .WORD   DMP11$B_FLAGS               ; Byte DMP11$B_FLAGS to data
                            03  0ABF                .BYTE   3                   ; 3 DMP11$B_FLAGS to data
                            01  0AC0                .BYTE   1                   ; 1 of data field
                            86  0AC1                .BYTE   Pd$_Literal         ; Indicate literal
                      00000001  0AC2                .LONG   1                   ; Constant
                            88  0AC6                .BYTE   Pd$_Store           ; Indicate store operation
                          0032  0AC7                .WORD   DMP11$B_COMMON              ; Byte DMP11$B_COMMON to data
                            00  0AC9                .BYTE   0                   ; 0 DMP11$B_COMMON to data
                            08  0ACA                .BYTE   8                   ; 8 of data field
                            81  0ACB                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                                0ACC    268 DMR11:  $DS_DMR11
                                0ACC                .SAVE   LOCAL_BLOCK
                                0ACC                .PSECT  $ABS$,ABS
                      00000032  0041                .=HP$A_DEPENDENT
                                0032                .IIF    NB,DMR11$L_CSR, DMR11$L_CSR:
                      00000036  0032                .IIF    NB,..BLKL,      .BLKL   1
                                0036                .IIF    NB,DMR11$B_BR,  DMR11$B_BR:
                      00000037  0036                .IIF    NB,..BLKB,      .BLKB   1
                                0037                .IIF    NB,DMR11$K_LEN, DMR11$K_LEN:
                            00000ACC                .RESTORE
31 31 52 4D 44 00' 0ACC                             .ASCIC  "DMR11" ; DMR11 name
                            05  0ACC
                            37  0AD2                .BYTE   DMR11$K_LEN                 ; DMR11$K_LEN of resultant P-table
                            00  0AD3                .BYTE   0                   ; Maximum unit number                       ;G:65
                          4D58  0AD4                .WORD   ^A"XM"  ; Two character mnemonic for QIO DMR11 XM
                            80  0AD6                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                            8D  0AD7                .Byte   Pd$_Name            ; Directive op-code
                            03  0AD8                .Byte   Ptd$M_Device                ; Enforced XM codes
                      4D 58 00' 0AD9                .Ascic  "XM"    ; Enforced device name prefix
                            02  0AD9
                            83  0ADC                .BYTE   Pd$_Octal           ; Indicate scan octal
                   52 53 43 00' 0ADD                .ASCIC  "CSR"   ; CSR string
                            03  0ADD
0003FFFE 0003E000           0AE1                    .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
```

```
                      88  0AE9                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  0AEA                .WORD   DMR11$L_CSR             ; Byte DMR11$L_CSR to data
                      00  0AEC                .BYTE   0               ; 0 DMR11$L_CSR to data
                      20  0AED                .BYTE   32              ; 32 of data field
                      88  0AEE                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0AEF                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  0AF1                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  0AF2                .BYTE   18              ; 18 of data field
                      83  0AF3                .BYTE   Pd$_Octal       ; Indicate scan octal
           52 4F 54 43 45 56 00' 0AF4         .ASCIC  "VECTOR"        ; VECTOR string
                      06  0AF4
              000001FE 00000002  0AFB         .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  0B03                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  0B04                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  0B06                .BYTE   0               ; 0 HP$W_VECTOR to data
                      09  0B07                .BYTE   9               ; 9 of data field
                      82  0B08                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                   52 42 00' 0B09             .ASCIC  "BR"    ; BR string
                      02  0B09
              00000007 00000004  0B0C         .LONG   4,7     ; 4 , 7 limits on input
                      88  0B14                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  0B15                .WORD   DMR11$B_BR              ; Byte DMR11$B_BR to data
                      00  0B17                .BYTE   0               ; 0 DMR11$B_BR to data
                      08  0B18                .BYTE   8               ; 8 of data field
                      81  0B19                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          0B1A    269 DMZ32:  $DS_DMZ32                       ;               [21]
                          0B1A                .SAVE   LOCAL_BLOCK
                          0B1A                .PSECT  $ABS$,ABS
                    00000032  0041            .=HP$A_DEPENDENT
                              0032            .IIF    NB,HP$B_DMZ32_BR,       HP$B_DMZ32_BR:
                    00000033  0032            .IIF    NB,..BLKB,      .BLKB   1
                              0033            .IIF    NB,HP$B_DMZ32_OCTET0,   HP$B_DMZ32_OCTET0:
                    00000034  0033            .IIF    NB,..BLKB,      .BLKB   1
                              0034            .IIF    NB,HP$B_DMZ32_OCTET1,   HP$B_DMZ32_OCTET1:
                    00000035  0034            .IIF    NB,..BLKB,      .BLKB   1
                              0035            .IIF    NB,HP$B_DMZ32_OCTET2,   HP$B_DMZ32_OCTET2:
                    00000036  0035            .IIF    NB,..BLKB,      .BLKB   1
                              0036            .IIF    NB,HP$B_DMZ32_SPEED,    HP$B_DMZ32_SPEED:
                    00000037  0036            .IIF    NB,..BLKB,      .BLKB   1
                              0037            .IIF    NB,HP$W_DMZ32_LOOP,     HP$W_DMZ32_LOOP:
                    00000039  0037            .IIF    NB,..BLKW,      .BLKW   1
                              0039            .IIF    NB,HP$B_DMZ32_MODEM,    HP$B_DMZ32_MODEM:
                    0000003A  0039            .IIF    NB,..BLKB,      .BLKB   1
                              003A            .IIF    NB,HP$K_DMZ32_LEN,      HP$K_DMZ32_LEN:
                        00000B1A              .RESTORE
              32 33 5A 4D 44 00' 0B1A         .ASCIC  "DMZ32" ; DMZ32 name
                      05  0B1A
                      3A  0B20                .BYTE   HP$K_DMZ32_LEN          ; HP$K_DMZ32_LEN of resultant P-table
                      00  0B21                .BYTE   0               ; Maximum unit number                   ;G:65
                    0000  0B22                .WORD   ^A""    ; Two character mnemonic for QIO DMZ32
                      80  0B24                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  0B25                .Byte   Pd$_Name        ; Directive op-code
                      02  0B26                .Byte   PTD$M_CONTROLLER                ; Enforced TX codes
                   58 54 00' 0B27             .Ascic  "TX"    ; Enforced device name prefix
                      02  0B27
                      83  0B2A                .BYTE   Pd$_Octal       ; Indicate scan octal
                52 53 43 00' 0B2B             .ASCIC  "CSR"   ; CSR string
```

```
                                  03  0B2B
                  0003FFFE 0003E000  0B2F          .LONG   ^O760000,^O0777776      ; 760000 , 0777776 limits on input
                                  88  0B37          .BYTE   Pd$_Store       ; Indicate store operation
                                0018  0B38          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                  00  0B3A          .BYTE   0               ; 0 HP$A_DEVICE to data
                                  12  0B3B          .BYTE   18              ; 18 of data field
                                  83  0B3C          .BYTE   Pd$_Octal       ; Indicate scan octal
              72 6F 74 63 65 56 00'  0B3D          .ASCIC  "Vector"        ; Vector string
                                  06  0B3D
                  000001FE 000000C0  0B44          .LONG   ^O300,^O776     ; 300 , 776 limits on input
                                  88  0B4C          .BYTE   Pd$_Store       ; Indicate store operation
                                0024  0B4D          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                  00  0B4F          .BYTE   0               ; 0 HP$W_VECTOR to data
                                  09  0B50          .BYTE   9               ; 9 of data field
                                  82  0B51          .BYTE   Pd$_Decimal     ; Indicate scan decimal
                           52 42 00'  0B52          .ASCIC  "BR"    ; BR string
                                  02  0B52
                  00000006 00000005  0B55          .LONG   5,6     ; 5 , 6 limits on input
                                  88  0B5D          .BYTE   Pd$_Store       ; Indicate store operation
                                0032  0B5E          .WORD   HP$B_DMZ32_BR           ; Byte HP$B_DMZ32_BR to data
                                  00  0B60          .BYTE   0               ; 0 HP$B_DMZ32_BR to data
                                  08  0B61          .BYTE   8               ; 8 of data field
                                  83  0B62          .BYTE   Pd$_Octal       ; Indicate scan octal
20 53 45 4E 49 4C 20 54 53 45 54 00'  0B63          .ASCIC  "TEST LINES (OCTET0)"   ; TEST LINES (OCTET0) string
           29 30 54 45 54 43 4F 28  0B6F
                                  13  0B63
                  000000FF 00000000  0B77          .LONG   ^O0,^O377       ; 0 , 377 limits on input
                                  88  0B7F          .BYTE   Pd$_Store       ; Indicate store operation
                                0033  0B80          .WORD   HP$B_DMZ32_OCTET0               ; Byte HP$B_DMZ32_OCTET0 to data
                                  00  0B82          .BYTE   0               ; 0 HP$B_DMZ32_OCTET0 to data
                                  08  0B83          .BYTE   8               ; 8 of data field
                                  83  0B84          .BYTE   Pd$_Octal       ; Indicate scan octal
4C 4E 4F 20 33 20 4C 45 56 45 4C 00'  0B85          .ASCIC  "LEVEL 3 ONLY - TEST LINES (OCTET1)"    ; LEVEL 3 ONLY - TEST LINES
4E 49 4C 20 54 53 45 54 20 2D 20 59  0B91
   29 31 54 45 54 43 4F 28 20 53 45  0B9D
                                  22  0B85
                  000000FF 00000000  0BA8          .LONG   ^O0,^O377       ; 0 , 377 limits on input
                                  88  0BB0          .BYTE   Pd$_Store       ; Indicate store operation
                                0034  0BB1          .WORD   HP$B_DMZ32_OCTET1               ; Byte HP$B_DMZ32_OCTET1 to data
                                  00  0BB3          .BYTE   0               ; 0 HP$B_DMZ32_OCTET1 to data
                                  08  0BB4          .BYTE   8               ; 8 of data field
                                  83  0BB5          .BYTE   Pd$_Octal       ; Indicate scan octal
4C 4E 4F 20 33 20 4C 45 56 45 4C 00'  0BB6          .ASCIC  "LEVEL 3 ONLY - TEST LINES (OCTET2)"    ; LEVEL 3 ONLY - TEST LINES
4E 49 4C 20 54 53 45 54 20 2D 20 59  0BC2
   29 32 54 45 54 43 4F 28 20 53 45  0BCE
                                  22  0BB6
                  000000FF 00000000  0BD9          .LONG   ^O0,^O377       ; 0 , 377 limits on input
                                  88  0BE1          .BYTE   Pd$_Store       ; Indicate store operation
                                0035  0BE2          .WORD   HP$B_DMZ32_OCTET2               ; Byte HP$B_DMZ32_OCTET2 to data
                                  00  0BE4          .BYTE   0               ; 0 HP$B_DMZ32_OCTET2 to data
                                  08  0BE5          .BYTE   8               ; 8 of data field
                                  85  0BE6          .BYTE   Pd$_String      ; Indicate scan and verify string
      65 74 61 52 20 64 75 61 42 00'  0BE7          .ASCIC  "Baud Rate"     ; Baud Rate string
                                  09  0BE7
                           30 35 00'  0BF1          .ASCIC  "50"
                                  02  0BF1
                           35 37 00'  0BF4          .ASCIC  "75"
```

```
                        02  0BF4
                30 31 31 00' 0BF7               .ASCIC  "110"
                        03  0BF7
             35 2E 34 33 31 00' 0BFB            .ASCIC  "134.5"
                        05  0BFB
                30 35 31 00' 0C01               .ASCIC  "150"
                        03  0C01
                30 30 33 00' 0C05               .ASCIC  "300"
                        03  0C05
                30 30 36 00' 0C09               .ASCIC  "600"
                        03  0C09
             30 30 32 31 00' 0C0D               .ASCIC  "1200"
                        04  0C0D
             30 30 38 31 00' 0C12               .ASCIC  "1800"
                        04  0C12
             30 30 30 32 00' 0C17               .ASCIC  "2000"
                        04  0C17
             30 30 34 32 00' 0C1C               .ASCIC  "2400"
                        04  0C1C
             30 30 38 34 00' 0C21               .ASCIC  "4800"
                        04  0C21
             30 30 36 39 00' 0C26               .ASCIC  "9600"
                        04  0C26
          30 30 32 39 31 00' 0C2B               .ASCIC  "19200"
                        05  0C2B
                        00  0C31                .BYTE   0               ; Null length is end of valid 50,75,110,134.5,150,30
                        88  0C32                .BYTE   Pd$_Store       ; Indicate store operation
                      0036  0C33                .WORD   HP$B_DMZ32_SPEED                ; Byte HP$B_DMZ32_SPEED to data
                        00  0C35                .BYTE   0               ; 0 HP$B_DMZ32_SPEED to data
                        08  0C36                .BYTE   8               ; 8 of data field
                        85  0C37                .BYTE   Pd$_String      ; Indicate scan and verify string
79 54 20 6B 63 61 62 70 6F 6F 4C 00' 0C38       .ASCIC  "Loopback Type" ; Loopback Type string
                     65 70  0C44
                        0D  0C38
                     30 00' 0C46                .ASCIC  "0"
                        01  0C46
                     31 00' 0C48                .ASCIC  "1"
                        01  0C48
                     32 00' 0C4A                .ASCIC  "2"
                        01  0C4A
                     33 00' 0C4C                .ASCIC  "3"
                        01  0C4C
                     34 00' 0C4E                .ASCIC  "4"
                        01  0C4E
                     35 00' 0C50                .ASCIC  "5"
                        01  0C50
                     36 00' 0C52                .ASCIC  "6"
                        01  0C52
                     37 00' 0C54                .ASCIC  "7"
                        01  0C54
                     38 00' 0C56                .ASCIC  "8"
                        01  0C56
                        00  0C58                .BYTE   0               ; Null length is end of valid 0,1,2,3,4,5,6,7,8
                        88  0C59                .BYTE   Pd$_Store       ; Indicate store operation
                      0037  0C5A                .WORD   HP$W_DMZ32_LOOP         ; Byte HP$W_DMZ32_LOOP to data
                        00  0C5C                .BYTE   0               ; 0 HP$W_DMZ32_LOOP to data
                        05  0C5D                .BYTE   5               ; 5 of data field
```

```
                          8B  0C5E                .BYTE   Pd$_Logical     ; Indicate logical value operation
72 74 6E 6F 43 20 6D 65 64 6F 4D 00' 0C5F        .ASCIC  \Modem Control\ ; Modem Control string
                       6C 6F  0C6B
                          0D  0C5F
                          88  0C6D                .BYTE   Pd$_Store       ; Indicate store operation
                        0039  0C6E                .WORD   HP$B_DMZ32_MODEM                 ; Byte HP$B_DMZ32_MODEM to data
                          00  0C70                .BYTE   0               ; 0 HP$B_DMZ32_MODEM to data
                          02  0C71                .BYTE   2               ; 2 of data field
                          86  0C72                .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  0C73                .LONG   1               ; Constant
                          81  0C77                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              0C78    270 DR11B:  $DS_DR11B
                              0C78                .SAVE   LOCAL_BLOCK
                              0C78                .PSECT  $ABS$,ABS
                    00000032  0041                .=HP$A_DEPENDENT
                              0032                .IIF    NB,DR11B$L_CSR, DR11B$L_CSR:
                    00000036  0032                .IIF    NB,..BLKL,      .BLKL   1
                              0036                .IIF    NB,DR11B$B_BR,  DR11B$B_BR:
                    00000037  0036                .IIF    NB,..BLKB,      .BLKB   1
                              0037                .IIF    NB,DR11B$K_LEN, DR11B$K_LEN:
                          00000C78                .RESTORE
                 42 31 31 52 44 00' 0C78        .ASCIC  "DR11B" ; DR11B name
                          05  0C78
                          37  0C7E                .BYTE   DR11B$K_LEN             ; DR11B$K_LEN of resultant P-table
                          00  0C7F                .BYTE   0               ; Maximum unit number                    ;G:65
                        0000  0C80                .WORD   ^A""    ; Two character mnemonic for QIO DR11B
                          80  0C82                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  0C83                .Byte   Pd$_Name        ; Directive op-code
                          03  0C84                .Byte   Ptd$M_Device            ; Enforced XB codes
                    42 58 00' 0C85                .Ascic  "XB"    ; Enforced device name prefix
                          02  0C85
                          83  0C88                .BYTE   Pd$_Octal       ; Indicate scan octal
                 52 53 43 00' 0C89                .ASCIC  "CSR"   ; CSR string
                          03  0C89
              0003FFFE 0003E000  0C8D             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  0C95                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  0C96                .WORD   DR11B$L_CSR             ; Byte DR11B$L_CSR to data
                          00  0C98                .BYTE   0               ; 0 DR11B$L_CSR to data
                          20  0C99                .BYTE   32              ; 32 of data field
                          88  0C9A                .BYTE   Pd$_Store       ; Indicate store operation
                        0018  0C9B                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  0C9D                .BYTE   0               ; 0 HP$A_DEVICE to data
                          12  0C9E                .BYTE   18              ; 18 of data field
                          83  0C9F                .BYTE   Pd$_Octal       ; Indicate scan octal
           52 4F 54 43 45 56 00' 0CA0             .ASCIC  "VECTOR"        ; VECTOR string
                          06  0CA0
              000001FE 00000002  0CA7             .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  0CAF                .BYTE   Pd$_Store       ; Indicate store operation
                        0024  0CB0                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  0CB2                .BYTE   0               ; 0 HP$W_VECTOR to data
                          09  0CB3                .BYTE   9               ; 9 of data field
                          82  0CB4                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                    52 42 00' 0CB5                .ASCIC  "BR"    ; BR string
                          02  0CB5
              00000007 00000004  0CB8             .LONG   4,7     ; 4 , 7 limits on input
                          88  0CC0                .BYTE   Pd$_Store       ; Indicate store operation
                        0036  0CC1                .WORD   DR11B$B_BR              ; Byte DR11B$B_BR to data
```

```
                        00  0CC3                .BYTE   0                       ; 0 DR11B$B_BR to data
                        08  0CC4                .BYTE   8                       ; 8 of data field
                        81  0CC5                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                            0CC6    271 DR11C:  $DS_DR11C                               ;               [29]
                            0CC6                .SAVE   LOCAL_BLOCK
                            0CC6                .PSECT  $ABS$,ABS
                    00000032  0041              .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$B_DR11C_BR,       HP$B_DR11C_BR:
                    00000033  0032              .IIF    NB,.BLKB,       .BLKB   1
                            0033                .IIF    NB,HP$K_DR11C_LEN,      HP$K_DR11C_LEN:
                        00000CC6                .RESTORE
                43 31 31 52 44 00' 0CC6         .ASCIC  "DR11C" ; DR11C name
                        05  0CC6
                        33  0CCC                .BYTE   HP$K_DR11C_LEN          ; HP$K_DR11C_LEN of resultant P-table
                        00  0CCD                .BYTE   0                       ; Maximum unit number                   ;G:65
                      414F  0CCE                .WORD   ^A"OA"  ; Two character mnemonic for QIO DR11C OA
                        80  0CD0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  0CD1                .Byte   Pd$_Name        ; Directive op-code
                        02  0CD2                .Byte   Ptd$M_Controller                ; Enforced OA codes
                    41 4F 00' 0CD3              .Ascic  "OA"    ; Enforced device name prefix
                        02  0CD3
                        83  0CD6                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 53 43 00' 0CD7             .ASCIC  "CSR"   ; CSR string
                        03  0CD7
                0003FFFE 0003E000  0CDB         .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  0CE3                .BYTE   Pd$_Store       ; Indicate store operation
                      0018  0CE4                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        00  0CE6                .BYTE   0                       ; 0 HP$A_DEVICE to data
                        12  0CE7                .BYTE   18                      ; 18 of data field
                        83  0CE8                .BYTE   Pd$_Octal       ; Indicate scan octal
             52 4F 54 43 45 56 00' 0CE9         .ASCIC  "VECTOR"        ; VECTOR string
                        06  0CE9
                000001FE 00000000  0CF0         .LONG   ^O0,^O776       ; 0 , 776 limits on input
                        88  0CF8                .BYTE   Pd$_Store       ; Indicate store operation
                      0024  0CF9                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                        00  0CFB                .BYTE   0                       ; 0 HP$W_VECTOR to data
                        10  0CFC                .BYTE   16                      ; 16 of data field
                        82  0CFD                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                  52 42 00' 0CFE                .ASCIC  "BR"    ; BR string
                        02  0CFE
                00000007 00000004  0D01         .LONG   4,7     ; 4 , 7 limits on input
                        88  0D09                .BYTE   Pd$_Store       ; Indicate store operation
                      0032  0D0A                .WORD   HP$B_DR11C_BR           ; Byte HP$B_DR11C_BR to data
                        00  0D0C                .BYTE   0                       ; 0 HP$B_DR11C_BR to data
                        08  0D0D                .BYTE   8                       ; 8 of data field
                        81  0D0E                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            0D0F    272 DR11K:  $DS_DR11K
                            0D0F                .SAVE   LOCAL_BLOCK
                            0D0F                .PSECT  $ABS$,ABS
                    00000032  0041              .=HP$A_DEPENDENT
                            0032                .IIF    NB,DR11K$B_BR,  DR11K$B_BR:
                    00000033  0032              .IIF    NB,.BLKB,       .BLKB   1
                            0033                .IIF    NB,DR11K$K_LEN, DR11K$K_LEN:
                        00000D0F                .RESTORE
                4B 31 31 52 44 00' 0D0F         .ASCIC  "DR11K" ; DR11K name
                        05  0D0F
                        33  0D15                .BYTE   DR11K$K_LEN             ; DR11K$K_LEN of resultant P-table
```

```
                                 06  0D16                .BYTE   6                  ; Maximum unit number                          ;G:65
                               0000  0D17                .WORD   ^A""     ; Two character mnemonic for QIO DR11K
                                 80  0D19                .BYTE   Pd$_Start          ; Constant to identify start of descriptor
                                 8D  0D1A                .Byte   Pd$_Name           ; Directive op-code
                                 03  0D1B                .Byte   Ptd$M_Device               ; Enforced XR codes
                          52 58 00'  0D1C                .Ascic  "XR"     ; Enforced device name prefix
                                 02  0D1C
                                 83  0D1F                .BYTE   Pd$_Octal          ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00'  0D20                .ASCIC  "I/O BUS CSR"      ; I/O BUS CSR string
                                 0B  0D20
                  0003FFFE 0003E000  0D2C                .LONG   ^O760000,^O777776          ; 760000 , 777776 limits on input
                                 88  0D34                .BYTE   Pd$_Store          ; Indicate store operation
                               0018  0D35                .WORD   HP$A_DEVICE                ; Byte HP$A_DEVICE to data
                                 00  0D37                .BYTE   0                  ; 0 HP$A_DEVICE to data
                                 12  0D38                .BYTE   18                 ; 18 of data field
                                 83  0D39                .BYTE   Pd$_Octal          ; Indicate scan octal
                 52 4F 54 43 45 56 00'  0D3A                .ASCIC  "VECTOR"           ; VECTOR string
                                 06  0D3A
                  000001FE 00000002  0D41                .LONG   ^O2,^O776          ; 2 , 776 limits on input
                                 88  0D49                .BYTE   Pd$_Store          ; Indicate store operation
                               0024  0D4A                .WORD   HP$W_VECTOR                ; Byte HP$W_VECTOR to data
                                 00  0D4C                .BYTE   0                  ; 0 HP$W_VECTOR to data
                                 09  0D4D                .BYTE   9                  ; 9 of data field
                                 82  0D4E                .BYTE   Pd$_Decimal        ; Indicate scan decimal
                          52 42 00'  0D4F                .ASCIC  "BR"     ; BR string
                                 02  0D4F
                  00000007 00000004  0D52                .LONG   4,7      ; 4 , 7 limits on input
                                 88  0D5A                .BYTE   Pd$_Store          ; Indicate store operation
                               0032  0D5B                .WORD   DR11K$B_BR                 ; Byte DR11K$B_BR to data
                                 00  0D5D                .BYTE   0                  ; 0 DR11K$B_BR to data
                                 08  0D5E                .BYTE   8                  ; 8 of data field
                                 81  0D5F                .BYTE   Pd$_End            ; Indicate end of PT-descriptor
                                     0D60   273 DR11W:   $DS_DR11W
                                     0D60                .SAVE   LOCAL_BLOCK
                                     0D60                .PSECT  $ABS$,ABS
                          00000032  0041                .=HP$A_DEPENDENT
                                     0032                .IIF    NB,DR11W$L_CSR, DR11W$L_CSR:
                          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                     0036                .IIF    NB,DR11W$B_BR,  DR11W$B_BR:
                          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                     0037                .IIF    NB,DR11W$K_LEN, DR11W$K_LEN:
                                 00000D60                .RESTORE
                  57 31 31 52 44 00'  0D60                .ASCIC  "DR11W" ; DR11W name
                                 05  0D60
                                 37  0D66                .BYTE   DR11W$K_LEN                ; DR11W$K_LEN of resultant P-table
                                 00  0D67                .BYTE   0                  ; Maximum unit number                          ;G:65
                               4158  0D68                .WORD   ^A"XA"   ; Two character mnemonic for QIO DR11W XA
                                 80  0D6A                .BYTE   Pd$_Start          ; Constant to identify start of descriptor
                                 8D  0D6B                .Byte   Pd$_Name           ; Directive op-code
                                 03  0D6C                .Byte   Ptd$M_Device               ; Enforced XA codes
                          41 58 00'  0D6D                .Ascic  "XA"     ; Enforced device name prefix
                                 02  0D6D
                                 83  0D70                .BYTE   Pd$_Octal          ; Indicate scan octal
                       52 53 43 00'  0D71                .ASCIC  "CSR"    ; CSR string
                                 03  0D71
                  0003FFFE 0003E000  0D75                .LONG   ^O760000,^O777776          ; 760000 , 777776 limits on input
                                 88  0D7D                .BYTE   Pd$_Store          ; Indicate store operation
```

```
                        0032  0D7E                .WORD   DR11W$L_CSR             ; Byte DR11W$L_CSR to data
                          00  0D80                .BYTE   0                   ; 0 DR11W$L_CSR to data
                          20  0D81                .BYTE   32                  ; 32 of data field
                          88  0D82                .BYTE   Pd$_Store           ; Indicate store operation
                        0018  0D83                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                          00  0D85                .BYTE   0                   ; 0 HP$A_DEVICE to data
                          12  0D86                .BYTE   18                  ; 18 of data field
                          83  0D87                .BYTE   Pd$_Octal           ; Indicate scan octal
           52 4F 54 43 45 56 00' 0D88             .ASCIC  "VECTOR"            ; VECTOR string
                          06  0D88
              000001FE 00000002  0D8F             .LONG   ^O2,^O776           ; 2 , 776 limits on input
                          88  0D97                .BYTE   Pd$_Store           ; Indicate store operation
                        0024  0D98                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                          00  0D9A                .BYTE   0                   ; 0 HP$W_VECTOR to data
                          09  0D9B                .BYTE   9                   ; 9 of data field
                          82  0D9C                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                     52 42 00' 0D9D               .ASCIC  "BR"    ; BR string
                          02  0D9D
              00000007 00000004  0DA0             .LONG   4,7     ; 4 , 7 limits on input
                          88  0DA8                .BYTE   Pd$_Store           ; Indicate store operation
                        0036  0DA9                .WORD   DR11W$B_BR                  ; Byte DR11W$B_BR to data
                          00  0DAB                .BYTE   0                   ; 0 DR11W$B_BR to data
                          08  0DAC                .BYTE   8                   ; 8 of data field
                          81  0DAD                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                              0DAE    274 DR750:  $DS_DR750
                              0DAE                .SAVE   LOCAL_BLOCK
                              0DAE                .PSECT  $ABS$,ABS
                     00000032 0041                .=HP$A_DEPENDENT
                              0032                .IIF    NB,DR750$B_SLOT,        DR750$B_SLOT:
                     00000033 0032                .IIF    NB,,.BLKB,      .BLKB   1
                              0033                .IIF    NB,DR750$B_BR,  DR750$B_BR:
                     00000034 0033                .IIF    NB,,.BLKB,      .BLKB   1
                              0034                .IIF    NB,DR750$B_SELF,        DR750$B_SELF:
                     00000035 0034                .IIF    NB,,.BLKB,      .BLKB   1
                              0035                .IIF    NB,DR750$B_CONFIG,      DR750$B_CONFIG:
                     00000036 0035                .IIF    NB,,.BLKB,      .BLKB   1
                              0036                .IIF    NB,DR750$B_UUT, DR750$B_UUT:
                     00000037 0036                .IIF    NB,,.BLKB,      .BLKB   1
                              0037                .IIF    NB,DR750$K_LEN, DR750$K_LEN:
                        00000DAE                  .RESTORE
           30 35 37 52 44 00' 0DAE                .ASCIC  "DR750" ; DR750 name
                          05  0DAE
                          37  0DB4                .BYTE   DR750$K_LEN                 ; DR750$K_LEN of resultant P-table
                          FF  0DB5                .BYTE   255                 ; Maximum unit number                           ;G:65
                        4658  0DB6                .WORD   ^A"XF"  ; Two character mnemonic for QIO DR750 XF
                          80  0DB8                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                          8D  0DB9                .Byte   Pd$_Name            ; Directive op-code
                          03  0DBA                .Byte   Ptd$M_Device                ; Enforced XF codes
                     46 58 00' 0DBB               .Ascic  "XF"    ; Enforced device name prefix
                          02  0DBB
                          86  0DBE                .BYTE   Pd$_Literal         ; Indicate literal
                    40F30000  0DBF                .LONG   ^X40F30000                  ; Constant
                          88  0DC3                .BYTE   Pd$_Store           ; Indicate store operation
                        0018  0DC4                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                          00  0DC6                .BYTE   0                   ; 0 HP$A_DEVICE to data
                          20  0DC7                .BYTE   32                  ; 32 of data field
                          82  0DC8                .BYTE   Pd$_Decimal         ; Indicate scan decimal
```

```
                    54 4F 4C 53 00' 0DC9              .ASCIC  "SLOT"  ; SLOT string
                                 04 0DC9
                  0000000F 0000000A 0DCE              .LONG   10,15   ; 10 , 15 limits on input
                                 88 0DD6              .BYTE   Pd$_Store       ; Indicate store operation
                               0032 0DD7              .WORD   DR750$B_SLOT            ; Byte DR750$B_SLOT to data
                                 00 0DD9              .BYTE   0               ; 0 DR750$B_SLOT to data
                                 08 0DDA              .BYTE   8               ; 8 of data field
                                 88 0DDB              .BYTE   Pd$_Store       ; Indicate store operation
                               0018 0DDC              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                 0D 0DDE              .BYTE   13              ; 13 HP$A_DEVICE to data
                                 04 0DDF              .BYTE   4               ; 4 of data field
                                 88 0DE0              .BYTE   Pd$_Store       ; Indicate store operation
                               0024 0DE1              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                 02 0DE3              .BYTE   2               ; 2 HP$W_VECTOR to data
                                 04 0DE4              .BYTE   4               ; 4 of data field
                                 82 0DE5              .BYTE   Pd$_Decimal     ; Indicate scan decimal
                            52 42 00' 0DE6            .ASCIC  "BR"    ; BR string
                                 02 0DE6
                  00000007 00000004 0DE9              .LONG   4,7     ; 4 , 7 limits on input
                                 88 0DF1              .BYTE   Pd$_Store       ; Indicate store operation
                               0033 0DF2              .WORD   DR750$B_BR              ; Byte DR750$B_BR to data
                                 00 0DF4              .BYTE   0               ; 0 DR750$B_BR to data
                                 08 0DF5              .BYTE   8               ; 8 of data field
                                 88 0DF6              .BYTE   Pd$_Store       ; Indicate store operation
                               0024 0DF7              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                 06 0DF9              .BYTE   6               ; 6 HP$W_VECTOR to data
                                 03 0DFA              .BYTE   3               ; 3 of data field
                                 8B 0DFB              .BYTE   Pd$_Logical     ; Indicate logical value operation
        54 53 45 54 20 46 4C 45 53 00' 0DFC           .ASCIC  \SELF TEST\     ; SELF TEST string
                                 09 0DFC
                                 88 0E06              .BYTE   Pd$_Store       ; Indicate store operation
                               0034 0E07              .WORD   DR750$B_SELF            ; Byte DR750$B_SELF to data
                                 00 0E09              .BYTE   0               ; 0 DR750$B_SELF to data
                                 08 0E0A              .BYTE   8               ; 8 of data field
                                 85 0E0B              .BYTE   Pd$_String      ; Indicate scan and verify string
47 49 46 4E 4F 43 20 54 53 45 54 00' 0E0C             .ASCIC  "TEST CONFIG"   ; TEST CONFIG string
                                 0B 0E0C
                        55 50 43 31 00' 0E18          .ASCIC  "1CPU"
                                 04 0E18
                        55 50 43 32 00' 0E1D          .ASCIC  "2CPU"
                                 04 0E1D
                                 00 0E22              .BYTE   0               ; Null length is end of valid 1CPU,2CPU
                                 88 0E23              .BYTE   Pd$_Store       ; Indicate store operation
                               0035 0E24              .WORD   DR750$B_CONFIG          ; Byte DR750$B_CONFIG to data
                                 00 0E26              .BYTE   0               ; 0 DR750$B_CONFIG to data
                                 08 0E27              .BYTE   8               ; 8 of data field
                                 8B 0E28              .BYTE   Pd$_Logical     ; Indicate logical value operation
     54 49 4E 55 20 54 53 45 54 00' 0E29              .ASCIC  \TEST UNIT\     ; TEST UNIT string
                                 09 0E29
                                 88 0E33              .BYTE   Pd$_Store       ; Indicate store operation
                               0036 0E34              .WORD   DR750$B_UUT             ; Byte DR750$B_UUT to data
                                 00 0E36              .BYTE   0               ; 0 DR750$B_UUT to data
                                 08 0E37              .BYTE   8               ; 8 of data field
                                 81 0E38              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    0E39  275 DR780:  $DS_DR780
                                    0E39              .SAVE   LOCAL_BLOCK
                                    0E39              .PSECT  $ABS$,ABS
```

```
                    00000032  0041              .=HP$A_DEPENDENT
                              0032              .IIF    NB,DR780$B_TR,  DR780$B_TR:
                    00000033  0032              .IIF    NB,.BLKB,       .BLKB   1
                              0033              .IIF    NB,DR780$B_BR,  DR780$B_BR:
                    00000034  0033              .IIF    NB,.BLKB,       .BLKB   1
                              0034              .IIF    NB,DR780$B_SELF,        DR780$B_SELF:
                    00000035  0034              .IIF    NB,.BLKB,       .BLKB   1
                              0035              .IIF    NB,DR780$B_CONFIG,      DR780$B_CONFIG:
                    00000036  0035              .IIF    NB,.BLKB,       .BLKB   1
                              0036              .IIF    NB,DR780$B_UUT, DR780$B_UUT:
                    00000037  0036              .IIF    NB,.BLKB,       .BLKB   1
                              0037              .IIF    NB,DR780$K_LEN, DR780$K_LEN:
                         00000E39               .RESTORE
                30 38 37 52 44 00' 0E39         .ASCIC  "DR780" ; DR780 name
                         05  0E39
                         37  0E3F               .BYTE   DR780$K_LEN             ; DR780$K_LEN of resultant P-table
                         FF  0E40               .BYTE   255             ; Maximum unit number                   ;G:65
                       4658  0E41               .WORD   ^A"XF"  ; Two character mnemonic for QIO DR780 XF
                         80  0E43               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                         8D  0E44               .Byte   Pd$_Name        ; Directive op-code
                         03  0E45               .Byte   Ptd$M_Device            ; Enforced XF codes
                    46 58 00' 0E46              .Ascic  "XF"    ; Enforced device name prefix
                         02  0E46
                         82  0E49               .BYTE   Pd$_Decimal     ; Indicate scan decimal
                    52 54 00' 0E4A              .ASCIC  "TR"    ; TR string
                         02  0E4A
                0000000F 00000001  0E4D         .LONG   1,15    ; 1 , 15 limits on input
                         88  0E55               .BYTE   Pd$_Store       ; Indicate store operation
                       0032  0E56               .WORD   DR780$B_TR              ; Byte DR780$B_TR to data
                         00  0E58               .BYTE   0               ; 0 DR780$B_TR to data
                         08  0E59               .BYTE   8               ; 8 of data field
                         88  0E5A               .BYTE   Pd$_Store       ; Indicate store operation
                       0018  0E5B               .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                         0D  0E5D               .BYTE   13              ; 13 HP$A_DEVICE to data
                         04  0E5E               .BYTE   4               ; 4 of data field
                         88  0E5F               .BYTE   Pd$_Store       ; Indicate store operation
                       0024  0E60               .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                         02  0E62               .BYTE   2               ; 2 HP$W_VECTOR to data
                         04  0E63               .BYTE   4               ; 4 of data field
                         82  0E64               .BYTE   Pd$_Decimal     ; Indicate scan decimal
                    52 42 00' 0E65              .ASCIC  "BR"    ; BR string
                         02  0E65
                00000007 00000004  0E68         .LONG   4,7     ; 4 , 7 limits on input
                         88  0E70               .BYTE   Pd$_Store       ; Indicate store operation
                       0033  0E71               .WORD   DR780$B_BR              ; Byte DR780$B_BR to data
                         00  0E73               .BYTE   0               ; 0 DR780$B_BR to data
                         08  0E74               .BYTE   8               ; 8 of data field
                         88  0E75               .BYTE   Pd$_Store       ; Indicate store operation
                       0024  0E76               .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                         06  0E78               .BYTE   6               ; 6 HP$W_VECTOR to data
                         02  0E79               .BYTE   2               ; 2 of data field
                         86  0E7A               .BYTE   Pd$_Literal     ; Indicate literal
                   00000006  0E7B               .LONG   6               ; Constant
                         88  0E7F               .BYTE   Pd$_Store       ; Indicate store operation
                       0018  0E80               .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                         1C  0E82               .BYTE   28              ; 28 HP$A_DEVICE to data
                         04  0E83               .BYTE   4               ; 4 of data field
```

```
                                 86  0E84                .BYTE   Pd$_Literal      ; Indicate literal
                         00000001    0E85                .LONG   1               ; Constant
                                 88  0E89                .BYTE   Pd$_Store       ; Indicate store operation
                               0024  0E8A                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                 08  0E8C                .BYTE   8               ; 8 HP$W_VECTOR to data
                                 01  0E8D                .BYTE   1               ; 1 of data field
                                 8B  0E8E                .BYTE   Pd$_Logical     ; Indicate logical value operation
      54 53 45 54 20 46 4C 45 53 00' 0E8F                .ASCIC  \SELF TEST\     ; SELF TEST string
                                 09  0E8F
                                 88  0E99                .BYTE   Pd$_Store       ; Indicate store operation
                               0034  0E9A                .WORD   DR780$B_SELF            ; Byte DR780$B_SELF to data
                                 00  0E9C                .BYTE   0               ; 0 DR780$B_SELF to data
                                 08  0E9D                .BYTE   8               ; 8 of data field
                                 85  0E9E                .BYTE   Pd$_String      ; Indicate scan and verify string
47 49 46 4E 4F 43 20 54 53 45 54 00' 0E9F                .ASCIC  "TEST CONFIG"   ; TEST CONFIG string
                                 0B  0E9F
                     55 50 43 31 00' 0EAB                .ASCIC  "1CPU"
                                 04  0EAB
                     55 50 43 32 00' 0EB0                .ASCIC  "2CPU"
                                 04  0EB0
                                 00  0EB5                .BYTE   0               ; Null length is end of valid 1CPU,2CPU
                                 88  0EB6                .BYTE   Pd$_Store       ; Indicate store operation
                               0035  0EB7                .WORD   DR780$B_CONFIG          ; Byte DR780$B_CONFIG to data
                                 00  0EB9                .BYTE   0               ; 0 DR780$B_CONFIG to data
                                 08  0EBA                .BYTE   8               ; 8 of data field
                                 8B  0EBB                .BYTE   Pd$_Logical     ; Indicate logical value operation
      54 49 4E 55 20 54 53 45 54 00' 0EBC                .ASCIC  \TEST UNIT\     ; TEST UNIT string
                                 09  0EBC
                                 88  0EC6                .BYTE   Pd$_Store       ; Indicate store operation
                               0036  0EC7                .WORD   DR780$B_UUT             ; Byte DR780$B_UUT to data
                                 00  0EC9                .BYTE   0               ; 0 DR780$B_UUT to data
                                 08  0ECA                .BYTE   8               ; 8 of data field
                                 81  0ECB                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                     0ECC    276 DUP11:  $DS_DUP11
                                     0ECC                .SAVE   LOCAL_BLOCK
                                     0ECC                .PSECT  $ABS$,ABS
                         00000032    0041                .=HP$A_DEPENDENT
                                     0032                .IIF    NB,DUP11$L_CSR, DUP11$L_CSR:
                         00000036    0032                .IIF    NB,.BLKL,       .BLKL   1
                                     0036                .IIF    NB,DUP11$B_BR,  DUP11$B_BR:
                         00000037    0036                .IIF    NB,.BLKB,       .BLKB   1
                                     0037                .IIF    NB,DUP11$K_LEN, DUP11$K_LEN:
                                 0000ECC                 .RESTORE
                  31 31 50 55 44 00' 0ECC                .ASCIC  "DUP11" ; DUP11 name
                                 05  0ECC
                                 37  0ED2                .BYTE   DUP11$K_LEN             ; DUP11$K_LEN of resultant P-table
                                 00  0ED3                .BYTE   0               ; Maximum unit number                   ;G:65
                               0000  0ED4                .WORD   ^A""    ; Two character mnemonic for QIO DUP11
                                 80  0ED6                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                 8D  0ED7                .Byte   Pd$_Name        ; Directive op-code
                                 03  0ED8                .Byte   Ptd$M_Device            ; Enforced XW codes
                        57 58 00'    0ED9                .Ascic  "XW"    ; Enforced device name prefix
                                 02  0ED9
                                 83  0EDC                .BYTE   Pd$_Octal       ; Indicate scan octal
                           52 53 43 00' 0EDD             .ASCIC  "CSR"   ; CSR string
                                 03  0EDD
                  0003FFFE 0003E000  0EE1                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
```

```
                      88  0EE9                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  0EEA                .WORD   DUP11$L_CSR             ; Byte DUP11$L_CSR to data
                      00  0EEC                .BYTE   0               ; 0 DUP11$L_CSR to data
                      20  0EED                .BYTE   32              ; 32 of data field
                      88  0EEE                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0EEF                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  0EF1                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  0EF2                .BYTE   18              ; 18 of data field
                      83  0EF3                .BYTE   Pd$_Octal       ; Indicate scan octal
           52 4F 54 43 45 56 00'  0EF4        .ASCIC  "VECTOR"        ; VECTOR string
                      06  0EF4
              000001FE 00000002  0EFB        .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  0F03                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  0F04                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  0F06                .BYTE   0               ; 0 HP$W_VECTOR to data
                      09  0F07                .BYTE   9               ; 9 of data field
                      82  0F08                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00'  0F09        .ASCIC  "BR"    ; BR string
                      02  0F09
              00000007 00000004  0F0C        .LONG   4,7     ; 4 , 7 limits on input
                      88  0F14                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  0F15                .WORD   DUP11$B_BR              ; Byte DUP11$B_BR to data
                      00  0F17                .BYTE   0               ; 0 DUP11$B_BR to data
                      08  0F18                .BYTE   8               ; 8 of data field
                      81  0F19                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          0F1A    277 DV11:   $DS_DV11
                          0F1A                .SAVE   LOCAL_BLOCK
                          0F1A                .PSECT  $ABS$,ABS
                  00000032  0041              .=HP$A_DEPENDENT
                            0032              .IIF    NB,DV11$L_CSR,  DV11$L_CSR:
                  00000036  0032              .IIF    NB,.BLKL,       .BLKL   1
                            0036              .IIF    NB,DV11$B_BR,   DV11$B_BR:
                  00000037  0036              .IIF    NB,.BLKB,       .BLKB   1
                            0037              .IIF    NB,DV11$K_LEN,  DV11$K_LEN:
                        00000F1A              .RESTORE
              31 31 56 44 00'  0F1A           .ASCIC  "DV11"  ; DV11 name
                      04  0F1A
                      37  0F1F                .BYTE   DV11$K_LEN              ; DV11$K_LEN of resultant P-table
                      00  0F20                .BYTE   0               ; Maximum unit number                   ;G:65
                    0000  0F21                .WORD   ^A""    ; Two character mnemonic for QIO DV11
                      80  0F23                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  0F24                .Byte   Pd$_Name        ; Directive op-code
                      03  0F25                .Byte   Ptd$M_Device            ; Enforced XV codes
                 56 58 00'  0F26        .Ascic  "XV"    ; Enforced device name prefix
                      02  0F26
                      83  0F29                .BYTE   Pd$_Octal       ; Indicate scan octal
                52 53 43 00'  0F2A        .ASCIC  "CSR"   ; CSR string
                      03  0F2A
              0003FFFE 0003E000  0F2E        .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  0F36                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  0F37                .WORD   DV11$L_CSR              ; Byte DV11$L_CSR to data
                      00  0F39                .BYTE   0               ; 0 DV11$L_CSR to data
                      20  0F3A                .BYTE   32              ; 32 of data field
                      88  0F3B                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0F3C                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  0F3E                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  0F3F                .BYTE   18              ; 18 of data field
```

```
                      83  0F40                .BYTE   Pd$_Octal          ; Indicate scan octal
          52 4F 54 43 45 56 00' 0F41          .ASCIC  "VECTOR"           ; VECTOR string
                      06  0F41
             000001FE 00000002  0F48          .LONG   ^O2,^O776          ; 2 , 776 limits on input
                      88  0F50                .BYTE   Pd$_Store          ; Indicate store operation
                    0024  0F51                .WORD   HP$W_VECTOR                ; Byte HP$W_VECTOR to data
                      00  0F53                .BYTE   0                  ; 0 HP$W_VECTOR to data
                      09  0F54                .BYTE   9                  ; 9 of data field
                      82  0F55                .BYTE   Pd$_Decimal        ; Indicate scan decimal
                   52 42 00' 0F56             .ASCIC  "BR"     ; BR string
                      02  0F56
             00000007 00000004  0F59          .LONG   4,7      ; 4 , 7 limits on input
                      88  0F61                .BYTE   Pd$_Store          ; Indicate store operation
                    0036  0F62                .WORD   DV11$B_BR                  ; Byte DV11$B_BR to data
                      00  0F64                .BYTE   0                  ; 0 DV11$B_BR to data
                      08  0F65                .BYTE   8                  ; 8 of data field
                      81  0F66                .BYTE   Pd$_End            ; Indicate end of PT-descriptor
                          0F67    278 DW730:  $DS_DW730                          ;                [02]
                          0F67                .SAVE   LOCAL_BLOCK
                          0F67                .PSECT  $ABS$,ABS
                   00000032  0041             .=HP$A_DEPENDENT
                             0032             .IIF    NB,DW730$K_LEN, DW730$K_LEN:
                          0F67                .RESTORE
          30 33 37 57 44 00' 0F67             .ASCIC  "DW730" ; DW730 name
                      05  0F67
                      32  0F6D                .BYTE   DW730$K_LEN                ; DW730$K_LEN of resultant P-table
                      01  0F6E                .BYTE   1                  ; Maximum unit number                        ;G:65
                    0000  0F6F                .WORD   ^A""     ; Two character mnemonic for QIO DW730
                      80  0F71                .BYTE   Pd$_Start          ; Constant to identify start of descriptor
                      8D  0F72                .Byte   Pd$_Name           ; Directive op-code
                      01  0F73                .Byte   Ptd$M_Unit                 ; Enforced DW codes
                   57 44 00' 0F74             .Ascic  "DW"     ; Enforced device name prefix
                      02  0F74
                      86  0F77                .BYTE   Pd$_Literal        ; Indicate literal
                40F26000  0F78                .LONG   ^X40F26000                 ; Constant
                      88  0F7C                .BYTE   Pd$_Store          ; Indicate store operation
                    0018  0F7D                .WORD   HP$A_DEVICE                ; Byte HP$A_DEVICE to data
                      00  0F7F                .BYTE   0                  ; 0 HP$A_DEVICE to data
                      20  0F80                .BYTE   32                 ; 32 of data field
                      86  0F81                .BYTE   Pd$_Literal        ; Indicate literal
                40FC0000  0F82                .LONG   ^X40FC0000                 ; Constant
                      88  0F86                .BYTE   Pd$_Store          ; Indicate store operation
                    001C  0F87                .WORD   HP$A_DVA                   ; Byte HP$A_DVA to data
                      00  0F89                .BYTE   0                  ; 0 HP$A_DVA to data
                      20  0F8A                .BYTE   32                 ; 32 of data field
                      86  0F8B                .BYTE   Pd$_Literal        ; Indicate literal
                00000200  0F8C                .LONG   ^X200              ; Constant
                      88  0F90                .BYTE   Pd$_Store          ; Indicate store operation
                    0024  0F91                .WORD   HP$W_VECTOR                ; Byte HP$W_VECTOR to data
                      00  0F93                .BYTE   0                  ; 0 HP$W_VECTOR to data
                      10  0F94                .BYTE   16                 ; 16 of data field
                      81  0F95                .BYTE   Pd$_End            ; Indicate end of PT-descriptor
                          0F96    279 DW750:  $DS_DW750
                          0F96                .SAVE   LOCAL_BLOCK
                          0F96                .PSECT  $ABS$,ABS
                   00000032  0041             .=HP$A_DEPENDENT
                             0032             .IIF    NB,DW750$K_LEN, DW750$K_LEN:
```

```
                    00000F96                    .RESTORE
          30 35 37 57 44 00' 0F96               .ASCIC  "DW750" ; DW750 name
                       05  0F96
                       32  0F9C                .BYTE   DW750$K_LEN             ; DW750$K_LEN of resultant P-table
                       04  0F9D                .BYTE   4               ; Maximum unit number                   ;G:65
                     0000  0F9E                .WORD   ^A""    ; Two character mnemonic for QIO DW750
                       80  0FA0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  0FA1                .Byte   Pd$_Name        ; Directive op-code
                       01  0FA2                .Byte   Ptd$M_Unit              ; Enforced DW codes
                    57 44 00' 0FA3              .Ascic  "DW"    ; Enforced device name prefix
                       02  0FA3
                       86  0FA6                .BYTE   Pd$_Literal     ; Indicate literal
                 40F30000  0FA7                .LONG   ^X40F30000              ; Constant
                       88  0FAB                .BYTE   Pd$_Store       ; Indicate store operation
                     0018  0FAC                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  0FAE                .BYTE   0               ; 0 HP$A_DEVICE to data
                       20  0FAF                .BYTE   32              ; 32 of data field
                       86  0FB0                .BYTE   Pd$_Literal     ; Indicate literal
                 40F80000  0FB1                .LONG   ^X40F80000              ; Constant
                       88  0FB5                .BYTE   Pd$_Store       ; Indicate store operation
                     001C  0FB6                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       00  0FB8                .BYTE   0               ; 0 HP$A_DVA to data
                       20  0FB9                .BYTE   32              ; 32 of data field
                       86  0FBA                .BYTE   Pd$_Literal     ; Indicate literal
                 00000200  0FBB                .LONG   ^X200           ; Constant
                       88  0FBF                .BYTE   Pd$_Store       ; Indicate store operation
                     0024  0FC0                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       00  0FC2                .BYTE   0               ; 0 HP$W_VECTOR to data
                       10  0FC3                .BYTE   16              ; 16 of data field
                       87  0FC4                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                     000B  0FC5                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                       00  0FC7                .BYTE   0               ; 0 HP$B_DRIVE to data
                       08  0FC8                .BYTE   8               ; 8 of data field
                       88  0FC9                .BYTE   Pd$_Store       ; Indicate store operation
                     0018  0FCA                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       0D  0FCC                .BYTE   13              ; 13 HP$A_DEVICE to data
                       01  0FCD                .BYTE   1               ; 1 of data field
                       8A  0FCE                .BYTE   Pd$_Add         ; Indicate add value to field operation
                     0024  0FCF                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       09  0FD1                .BYTE   9               ; 9 HP$W_VECTOR to data
                       02  0FD2                .BYTE   2               ; 2 of data field
                       89  0FD3                .BYTE   PJ$_Complement  ; Indicate complement value operation
                       88  0FD4                .BYTE   Pd$_Store       ; Indicate store operation
                     001C  0FD5                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       12  0FD7                .BYTE   18              ; 18 HP$A_DVA to data
                       01  0FD8                .BYTE   1               ; 1 of data field
                       81  0FD9                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                           0FDA    280 DW780:   $DS_DW780
                           0FDA                .SAVE   LOCAL_BLOCK
                           0FDA                .PSECT  $ABS$,ABS
                 00000032  0041                .=HP$A_DEPENDENT
                           0032                .IIF    NB,DW780$B_TR,  DW780$B_TR:
                 00000033  0032                .IIF    NB,,BLKB,       .BLKB   1
                           0033                .IIF    NB,DW780$B_BR,  DW780$B_BR:
                 00000034  0033                .IIF    NB,,BLKB,       .BLKB   1
                           0034                .IIF    NB,DW780$K_LEN, DW780$K_LEN:
                    00000FDA                    .RESTORE
```

```
                30 38 37 57 44 00' 0FDA              .ASCIC  "DW780" ; DW780 name
                                05  0FDA
                                34  0FE0              .BYTE   DW780$K_LEN             ; DW780$K_LEN of resultant P-table
                                08  0FE1              .BYTE   8                       ; Maximum unit number                   ;G:65
                              0000  0FE2              .WORD   ^A""    ; Two character mnemonic for QIO DW780
                                80  0FE4              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  0FE5              .Byte   Pd$_Name        ; Directive op-code
                                01  0FE6              .Byte   Ptd$M_Unit              ; Enforced DW codes
                            57 44 00' 0FE7              .Ascic  "DW"    ; Enforced device name prefix
                                02  0FE7
                                82  0FEA              .BYTE   Pd$_Decimal     ; Indicate scan decimal
                            52 54 00' 0FEB              .ASCIC  "TR"    ; TR string
                                02  0FEB
                        0000000F 00000001  0FEE              .LONG   1,15    ; 1 , 15 limits on input
                                88  0FF6              .BYTE   Pd$_Store       ; Indicate store operation
                              0032  0FF7              .WORD   DW780$B_TR              ; Byte DW780$B_TR to data
                                00  0FF9              .BYTE   0               ; 0 DW780$B_TR to data
                                08  0FFA              .BYTE   8               ; 8 of data field
                                88  0FFB              .BYTE   Pd$_Store       ; Indicate store operation
                              0018  0FFC              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                0D  0FFE              .BYTE   13              ; 13 HP$A_DEVICE to data
                                04  0FFF              .BYTE   4               ; 4 of data field
                                88  1000              .BYTE   Pd$_Store       ; Indicate store operation
                              0024  1001              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                02  1003              .BYTE   2               ; 2 HP$W_VECTOR to data
                                04  1004              .BYTE   4               ; 4 of data field
                                82  1005              .BYTE   Pd$_Decimal     ; Indicate scan decimal
                            52 42 00' 1006              .ASCIC  "BR"    ; BR string
                                02  1006
                        00000007 00000004  1009              .LONG   4,7     ; 4 , 7 limits on input
                                88  1011              .BYTE   Pd$_Store       ; Indicate store operation
                              0033  1012              .WORD   DW780$B_BR              ; Byte DW780$B_BR to data
                                00  1014              .BYTE   0               ; 0 DW780$B_BR to data
                                08  1015              .BYTE   8               ; 8 of data field
                                88  1016              .BYTE   Pd$_Store       ; Indicate store operation
                              0024  1017              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                06  1019              .BYTE   6               ; 6 HP$W_VECTOR to data
                                02  101A              .BYTE   2               ; 2 of data field
                                87  101B              .BYTE   Pd$_Fetch       ; Indicate fetch operation
                              000B  101C              .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                                00  101E              .BYTE   0               ; 0 HP$B_DRIVE to data
                                08  101F              .BYTE   8               ; 8 of data field
                                88  1020              .BYTE   Pd$_Store       ; Indicate store operation
                              001C  1021              .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                                12  1023              .BYTE   18              ; 18 HP$A_DVA to data
                                02  1024              .BYTE   2               ; 2 of data field
                                86  1025              .BYTE   Pd$_Literal     ; Indicate literal
                        00000006  1026              .LONG   6               ; Constant
                                88  102A              .BYTE   Pd$_Store       ; Indicate store operation
                              0018  102B              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                1C  102D              .BYTE   28              ; 28 HP$A_DEVICE to data
                                04  102E              .BYTE   4               ; 4 of data field
                                88  102F              .BYTE   Pd$_Store       ; Indicate store operation
                              001C  1030              .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                                1C  1032              .BYTE   28              ; 28 HP$A_DVA to data
                                04  1033              .BYTE   4               ; 4 of data field
                                87  1034              .BYTE   Pd$_Fetch       ; Indicate fetch operation
```

```
                     0018  1035                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       19  1037                .BYTE   25              ; 25 HP$A_DEVICE to data
                       02  1038                .BYTE   2               ; 2 of data field
                       88  1039                .BYTE   Pd$_Store       ; Indicate store operation
                     001C  103A                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       19  103C                .BYTE   25              ; 25 HP$A_DVA to data
                       02  103D                .BYTE   2               ; 2 of data field
                       86  103E                .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  103F                .LONG   1               ; Constant
                       88  1043                .BYTE   Pd$_Store       ; Indicate store operation
                     001C  1044                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       14  1046                .BYTE   20              ; 20 HP$A_DVA to data
                       01  1047                .BYTE   1               ; 1 of data field
                       88  1048                .BYTE   Pd$_Store       ; Indicate store operation
                     0024  1049                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       08  104B                .BYTE   8               ; 8 HP$W_VECTOR to data
                       01  104C                .BYTE   1               ; 1 of data field
                       81  104D                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                           104E  281 DX20:      $DS_DX20                        ;               [30]
                           104E                .SAVE   LOCAL_BLOCK
                           104E                .PSECT  $ABS$,ABS
                 00000032  0041                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$B_DX20_DRIVE,     HP$B_DX20_DRIVE::
                           0032                .IIF    NB,DX20$B_DRIVE,        DX20$B_DRIVE::
                 00000033  0032                .IIF    NB,,BLKB,       .BLKB   1
                           0033                .IIF    NB,HP$K_DX20_LEN,       HP$K_DX20_LEN::
                           0033                .IIF    NB,DX20$K_LEN,  DX20$K_LEN::
                       0000104E                .RESTORE
           30 32 58 44 00' 104E                .ASCIC  "DX20"  ; DX20 name
                       04  104E
                       33  1053                .BYTE   DX20$K_LEN              ; DX20$K_LEN of resultant P-table
                       08  1054                .BYTE   8               ; Maximum unit number                   ;G:65
                     0000  1055                .WORD   ^A""    ; Two character mnemonic for QIO DX20
                       80  1057                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  1058                .Byte   Pd$_Name        ; Directive op-code
                       02  1059                .Byte   PTD$M_CONTROLLER                ; Enforced MR codes
                 52 4D 00' 105A                .Ascic  "MR"    ; Enforced device name prefix
                       02  105A
                       82  105D                .BYTE   Pd$_Decimal     ; Indicate scan decimal
           45 56 49 52 44 00' 105E             .ASCIC  "DRIVE" ; DRIVE string
                       05  105E
 00000007 00000000     1064                .LONG   0,7     ; 0 . 7 limits on input
                       88  106C                .BYTE   Pd$_Store       ; Indicate store operation
                     0032  106D                .WORD   DX20$B_DRIVE            ; Byte DX20$B_DRIVE to data
                       00  106F                .BYTE   0               ; 0 DX20$B_DRIVE to data
                       08  1070                .BYTE   8               ; 8 of data field
                       88  1071                .BYTE   Pd$_Store       ; Indicate store operation
                     000B  1072                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                       00  1074                .BYTE   0               ; 0 HP$B_DRIVE to data
                       08  1075                .BYTE   8               ; 8 of data field
                       88  1076                .BYTE   Pd$_Store       ; Indicate store operation
                     0018  1077                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       07  1079                .BYTE   7               ; 7 HP$A_DEVICE to data
                       03  107A                .BYTE   3               ; 3 of data field
                       81  107B                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                           107C  282 DZ11:      $DS_DZ11
                           107C                .SAVE   LOCAL_BLOCK
```

```
                        107C                .PSECT  $ABS$,ABS
                    00000032  0041          .=HP$A_DEPENDENT
                              0032          .IIF    NB,DZ11$L_CSR,  DZ11$L_CSR:
                    00000036  0032          .IIF    NB,.BLKL,       .BLKL   1
                              0036          .IIF    NB,DZ11$B_BR,   DZ11$B_BR:
                    00000037  0036          .IIF    NB,.BLKB,       .BLKB   1
                              0037          .IIF    NB,DZ11$B_MTYPE,        DZ11$B_MTYPE:
                    00000038  0037          .IIF    NB,.BLKB,       .BLKB   1
                              0038          .IIF    NB,DZ11$K_LEN,  DZ11$K_LEN:
                         0000107C           .RESTORE
                31 31 5A 44 00' 107C        .ASCIC  "DZ11"  ; DZ11 name
                          04  107C
                          38  1081          .BYTE   DZ11$K_LEN              ; DZ11$K_LEN of resultant P-table
                          00  1082          .BYTE   0                       ; Maximum unit number                   ;G:65
                        5454  1083          .WORD   ^A"TT"  ; Two character mnemonic for QIO DZ11 TT
                          80  1085          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  1086          .Byte   Pd$_Name        ; Directive op-code
                          02  1087          .Byte   Ptd$M_Controller                ; Enforced TT codes
                     54 54 00' 1088         .Ascic  "TT"    ; Enforced device name prefix
                          02  1088
                          83  108B          .BYTE   Pd$_Octal       ; Indicate scan octal
                   52 53 43 00' 108C        .ASCIC  "CSR"   ; CSR string
                          03  108C
                0003FFFE 0003E000  1090     .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  1098          .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1099          .WORD   DZ11$L_CSR              ; Byte DZ11$L_CSR to data
                          00  109B          .BYTE   0                       ; 0 DZ11$L_CSR to data
                          20  109C          .BYTE   32                      ; 32 of data field
                          88  109D          .BYTE   Pd$_Store       ; Indicate store operation
                        0018  109E          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  10A0          .BYTE   0                       ; 0 HP$A_DEVICE to data
                          12  10A1          .BYTE   18                      ; 18 of data field
                          83  10A2          .BYTE   Pd$_Octal       ; Indicate scan octal
             52 4F 54 43 45 56 00' 10A3     .ASCIC  "VECTOR"        ; VECTOR string
                          06  10A3
                000001FE 00000002  10AA     .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  10B2          .BYTE   Pd$_Store       ; Indicate store operation
                        0024  10B3          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  10B5          .BYTE   0                       ; 0 HP$W_VECTOR to data
                          09  10B6          .BYTE   9                       ; 9 of data field
                          82  10B7          .BYTE   Pd$_Decimal     ; Indicate scan decimal
                     52 42 00' 10B8         .ASCIC  "BR"    ; BR string
                          02  10B8
                00000007 00000004  10BB     .LONG   4,7     ; 4 , 7 limits on input
                          88  10C3          .BYTE   Pd$_Store       ; Indicate store operation
                        0036  10C4          .WORD   DZ11$B_BR               ; Byte DZ11$B_BR to data
                          00  10C6          .BYTE   0                       ; 0 DZ11$B_BR to data
                          08  10C7          .BYTE   8                       ; 8 of data field
                          85  10C8          .BYTE   Pd$_String      ; Indicate scan and verify string
45 50 59 54 20 45 4C 55 44 4F 4D 00' 10C9   .ASCIC  "MODULE TYPE"   ; MODULE TYPE string
                          0B  10C9
                   41 49 45 00' 10D5        .ASCIC  "EIA"
                          03  10D5
                41 4D 30 32 00' 10D9        .ASCIC  "20MA"
                          04  10D9
                          00  10DE          .BYTE   0                       ; Null length is end of valid EIA,20MA
                          88  10DF          .BYTE   Pd$_Store       ; Indicate store operation
```

```
                     0037  10E0                .WORD   DZ11$B_MTYPE            ; Byte DZ11$B_MTYPE to data
                       00  10E2                .BYTE   0                ; 0 DZ11$B_MTYPE to data
                       08  10E3                .BYTE   8                ; 8 of data field
                       81  10E4                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                           10E5    283 DZ32:    $DS_DZ32                         ;              [08]
                           10E5                .SAVE   LOCAL_BLOCK
                           10E5                .PSECT  $ABS$,ABS
                 00000032  0041                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$B_DZ32_BRLVL,      HP$B_DZ32_BRLVL:
                           0032                .IIF    NB,DZ32$B_BRLVL,         DZ32$B_BRLVL:
                 00000033  0032                .IIF    NB,,BLKB,        .BLKB   1
                           0033                .IIF    NB,HP$B_DZ32_ACTIV,      HP$B_DZ32_ACTIV:
                           0033                .IIF    NB,DZ32$B_ACTIV,         DZ32$B_ACTIV:
                 00000034  0033                .IIF    NB,,BLKB,        .BLKB   1
                           0034                .IIF    NB,HP$B_DZ32_SPEED,      HP$B_DZ32_SPEED:
                           0034                .IIF    NB,DZ32$B_SPEED,         DZ32$B_SPEED:
                 00000035  0034                .IIF    NB,,BLKB,        .BLKB   1
                           0035                .IIF    NB,HP$W_DZ32_FLAGS,      HP$W_DZ32_FLAGS:
                           0035                .IIF    NB,DZ32$W_FLAGS,         DZ32$W_FLAGS:
                 00000037  0035                .IIF    NB,,BLKW,        .BLKW   1
                           0037                .IIF    NB,HP$K_LEN,     HP$K_LEN:
                           0037                .IIF    NB,DZ32$_LEN,    DZ32$_LEN:
                       000010E5                .RESTORE
              32 33 5A 44 00' 10E5             .ASCIC  "DZ32"   ; DZ32 name
                       04  10E5
                       37  10EA                .BYTE   DZ32$_LEN                ; DZ32$_LEN of resultant P-table
                       00  10EB                .BYTE   0                ; Maximum unit number                          ;G:65
                     0000  10EC                .WORD   ^A""     ; Two character mnemonic for QIO DZ32
                       80  10EE                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                       8D  10EF                .Byte   Pd$_Name         ; Directive op-code
                       02  10F0                .Byte   Ptd$M_Controller                 ; Enforced TT codes
                 54 54 00' 10F1                .Ascic  "TT"     ; Enforced device name prefix
                       02  10F1
                       83  10F4                .BYTE   Pd$_Octal        ; Indicate scan octal
                 52 53 43 00' 10F5             .ASCIC  "CSR"    ; CSR string
                       03  10F5
              0003FFFE 0003E000  10F9          .LONG   ^O760000,^O777776        ; 760000 , 777776 limits on input
                       88  1101                .BYTE   Pd$_Store        ; Indicate store operation
                     0018  1102                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                       00  1104                .BYTE   0                ; 0 HP$A_DEVICE to data
                       12  1105                .BYTE   18               ; 18 of data field
                       83  1106                .BYTE   Pd$_Octal        ; Indicate scan octal
           52 4F 54 43 45 56 00' 1107          .ASCIC  "VECTOR"         ; VECTOR string
                       06  1107
              000001FE 00000002  110E          .LONG   ^O2,^O776        ; 2 , 776 limits on input
                       88  1116                .BYTE   Pd$_Store        ; Indicate store operation
                     0024  1117                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                       00  1119                .BYTE   0                ; 0 HP$W_VECTOR to data
                       10  111A                .BYTE   16               ; 16 of data field
                       83  111B                .BYTE   Pd$_Octal        ; Indicate scan octal
                 52 42 00' 111C                .ASCIC  "BR"     ; BR string
                       02  111C
              00000007 00000004  111F          .LONG   ^O4,^O7 ; 4 , 7 limits on input
                       88  1127                .BYTE   Pd$_Store        ; Indicate store operation
                     0032  1128                .WORD   DZ32$B_BRLVL             ; Byte DZ32$B_BRLVL to data
                       00  112A                .BYTE   0                ; 0 DZ32$B_BRLVL to data
                       08  112B                .BYTE   8                ; 8 of data field
```

```
                                  83  112C          .BYTE   Pd$_Octal        ; Indicate scan octal
45 4E 49 4C 20 45 56 49 54 43 41 00'  112D          .ASCIC  "ACTIVE LINES"   ; ACTIVE LINES string
                                  53  1139
                                  0C  112D
                 000000FF 00000000  113A          .LONG   ^O0,^O377        ; 0 , 377 limits on input
                                  88  1142          .BYTE   Pd$_Store        ; Indicate store operation
                                0033  1143          .WORD   DZ32$B_ACTIV             ; Byte DZ32$B_ACTIV to data
                                  00  1145          .BYTE   0                ; 0 DZ32$B_ACTIV to data
                                  08  1146          .BYTE   8                ; 8 of data field
                                  83  1147          .BYTE   Pd$_Octal        ; Indicate scan octal
      45 54 41 52 20 44 55 41 42 00'  1148          .ASCIC  "BAUD RATE"      ; BAUD RATE string
                                  09  1148
                 0000000E 00000000  1152          .LONG   ^O0,^O16         ; 0 , 16 limits on input
                                  88  115A          .BYTE   Pd$_Store        ; Indicate store operation
                                0034  115B          .WORD   DZ32$B_SPEED             ; Byte DZ32$B_SPEED to data
                                  00  115D          .BYTE   0                ; 0 DZ32$B_SPEED to data
                                  08  115E          .BYTE   8                ; 8 of data field
                                  83  115F          .BYTE   Pd$_Octal        ; Indicate scan octal
59 54 20 4B 43 41 42 50 4F 4F 4C 00'  1160          .ASCIC  "LOOPBACK TYPE"  ; LOOPBACK TYPE string
                               45 50  116C
                                  0D  1160
                 00000004 00000000  116E          .LONG   ^O0,^O4 ; 0 , 4 limits on input
                                  88  1176          .BYTE   Pd$_Store        ; Indicate store operation
                                0035  1177          .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                                  00  1179          .BYTE   0                ; 0 DZ32$W_FLAGS to data
                                  03  117A          .BYTE   3                ; 3 of data field
                                  83  117B          .BYTE   Pd$_Octal        ; Indicate scan octal
53 20 54 45 53 45 52 20 48 4E 49 00'  117C          .ASCIC  "INH RESET SWITCH"       ; INH RESET SWITCH string
                     48 43 54 49 57  1188
                                  10  117C
                 00000001 00000000  118D          .LONG   ^O0,^O1 ; 0 , 1 limits on input
                                  88  1195          .BYTE   Pd$_Store        ; Indicate store operation
                                0035  1196          .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                                  03  1198          .BYTE   3                ; 3 DZ32$W_FLAGS to data
                                  01  1199          .BYTE   1                ; 1 of data field
                                  83  119A          .BYTE   Pd$_Octal        ; Indicate scan octal
48 43 54 49 57 53 20 45 44 4F 4D 00'  119B          .ASCIC  "MODE SWITCH"    ; MODE SWITCH string
                                  0B  119B
                 00000001 00000000  11A7          .LONG   ^O0,^O1 ; 0 , 1 limits on input
                                  88  11AF          .BYTE   Pd$_Store        ; Indicate store operation
                                0035  11B0          .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                                  04  11B2          .BYTE   4                ; 4 DZ32$W_FLAGS to data
                                  01  11B3          .BYTE   1                ; 1 of data field
                                  83  11B4          .BYTE   Pd$_Octal        ; Indicate scan octal
44 45 45 50 53 20 54 49 4C 50 53 00'  11B5          .ASCIC  "SPLIT SPEED JUMPER"     ; SPLIT SPEED JUMPER string
              52 45 50 4D 55 4A 20  11C1
                                  12  11B5
                 00000002 00000000  11C8          .LONG   ^O0,^O2 ; 0 , 2 limits on input
                                  88  11D0          .BYTE   Pd$_Store        ; Indicate store operation
                                0035  11D1          .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                                  05  11D3          .BYTE   5                ; 5 DZ32$W_FLAGS to data
                                  02  11D4          .BYTE   2                ; 2 of data field
                                  81  11D5          .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                      11D6  284 DZV11:  $DS_DZV11                        ;                [19]
                                      11D6          .SAVE   LOCAL_BLOCK
                                      11D6          .PSECT  $ABS$,ABS
                          00000032  0041          .=HP$A_DEPENDENT
```

```
                          0032                .IIF    NB,DZV11$L_CSR, DZV11$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,DZV11$K_LEN, DZV11$K_LEN:
                 000011D6                     .RESTORE
      31 31 56 5A 44 00'  11D6                .ASCIC  "DZV11" ; DZV11 name
                       05 11D6
                       36 11DC                .BYTE   DZV11$K_LEN             ; DZV11$K_LEN of resultant P-table
                       00 11DD                .BYTE   0                       ; Maximum unit number                   ;G:65
                     5454 11DE                .WORD   ^A"TT"  ; Two character mnemonic for QIO DZV11 TT
                       80 11E0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D 11E1                .Byte   Pd$_Name        ; Directive op-code
                       02 11E2                .Byte   Ptd$M_Controller                ; Enforced TT codes
                  54 54 00' 11E3                .Ascic  "TT"    ; Enforced device name prefix
                       02 11E3
                       83 11E6                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 53 43 00' 11E7                .ASCIC  "CSR"   ; CSR string
                       03 11E7
            0003FFFE 0003E000  11EB                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                       88 11F3                .BYTE   Pd$_Store       ; Indicate store operation
                     0032 11F4                .WORD   DZV11$L_CSR             ; Byte DZV11$L_CSR to data
                       00 11F6                .BYTE   0                       ; 0 DZV11$L_CSR to data
                       20 11F7                .BYTE   32                      ; 32 of data field
                       88 11F8                .BYTE   Pd$_Store       ; Indicate store operation
                     0018 11F9                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00 11FB                .BYTE   0                       ; 0 HP$A_DEVICE to data
                       12 11FC                .BYTE   18                      ; 18 of data field
                       83 11FD                .BYTE   Pd$_Octal       ; Indicate scan octal
   52 4F 54 43 45 56 00' 11FE                .ASCIC  "VECTOR"        ; VECTOR string
                       06 11FE
      000001FE 00000002  1205                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                       88 120D                .BYTE   Pd$_Store       ; Indicate store operation
                     0024 120E                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       00 1210                .BYTE   0                       ; 0 HP$W_VECTOR to data
                       09 1211                .BYTE   9                       ; 9 of data field
                       81 1212                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1213  285 IEU11A: $DS_IEU11A                        ;               [16]
                          1213                .SAVE   LOCAL_BLOCK
                          1213                .PSECT  $ABS$,ABS
                 00000032 0041                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_IEU11A_BR,      HP$B_IEU11A_BR:
                          0032                .IIF    NB,IEU11A$B_BR, IEU11A$B_BR:
                 00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_IEU11A_LEN,     HP$K_IEU11A_LEN:
                          0033                .IIF    NB,IEU11A$K_LEN,        IEU11A$K_LEN:
                 00001213                     .RESTORE
   41 31 31 55 45 49 00' 1213                .ASCIC  "IEU11A"        ; IEU11A name
                       06 1213
                       33 121A                .BYTE   IEU11A$K_LEN            ; IEU11A$K_LEN of resultant P-table
                       01 121B                .BYTE   1                       ; Maximum unit number                   ;G:65
                     0000 121C                .WORD   ^A""    ; Two character mnemonic for QIO IEU11A
                       80 121E                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D 121F                .Byte   Pd$_Name        ; Directive op-code
                       02 1220                .Byte   PTD$M_CONTROLLER                ; Enforced IX codes
                  58 49 00' 1221                .Ascic  "IX"    ; Enforced device name prefix
                       02 1221
                       83 1224                .BYTE   Pd$_Octal       ; Indicate scan octal
                65 73 61 42 00' 1225                .ASCIC  "Base"  ; Base string
```

```
                    04  1225
     0003FFF0 0003E000  122A          .LONG   ^O760000,^O777760       ; 760000 , 777760 limits on input
                    88  1232          .BYTE   Pd$_Store       ; Indicate store operation
                  0018  1233          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  1235          .BYTE   0               ; 0 HP$A_DEVICE to data
                    12  1236          .BYTE   18              ; 18 of data field
                    83  1237          .BYTE   Pd$_Octal       ; Indicate scan octal
  72 6F 74 63 65 56 00' 1238          .ASCIC  "Vector"        ; Vector string
                    06  1238
     000001FC 00000000  123F          .LONG   ^O0,^O774       ; 0 , 774 limits on input
                    88  1247          .BYTE   Pd$_Store       ; Indicate store operation
                  0024  1248          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  124A          .BYTE   0               ; 0 HP$W_VECTOR to data
                    10  124B          .BYTE   16              ; 16 of data field
                    82  124C          .BYTE   Pd$_Decimal     ; Indicate scan decimal
              52 42 00' 124D          .ASCIC  "BR"    ; BR string
                    02  124D
     00000007 00000004  1250          .LONG   4,7     ; 4 , 7 limits on input
                    88  1258          .BYTE   Pd$_Store       ; Indicate store operation
                  0032  1259          .WORD   IEU11A$B_BR             ; Byte IEU11A$B_BR to data
                    00  125B          .BYTE   0               ; 0 IEU11A$B_BR to data
                    08  125C          .BYTE   8               ; 8 of data field
                    81  125D          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        125E   286    $DS_KA_DEF
                        125E          .SAVE   LOCAL_BLOCK
                        125E          .PSECT  $ABS$,ABS
              00000032  0041          .=HP$A_Dependent
                        0032          .IIF    NB,KA$L_Flags,  KA$L_Flags::
              00000036  0032          .IIF    NB,.Blkl,       .Blkl   1
                        0036          .IIF    NB,KA$W_WCS,    KA$W_WCS::
              00000038  0036          .IIF    NB,.Blkw,       .Blkw   1
                        0038          .IIF    NB,KA$W_Reserved,       KA$W_Reserved::
              0000003A  0038          .IIF    NB,.Blkw,       .Blkw   1
                        003A          .IIF    NB,KA$L_SID,    KA$L_SID::
              0000003D  003A          .IIF    NB,.Blkb,       .Blkb   3
                        003D          .IIF    NB,KA$B_SID_Type,       KA$B_SID_Type::
              0000003E  003D          .IIF    NB,.Blkb,       .Blkb   1
                        003E          .IIF    NB,KA$W_Latency,        KA$W_Latency::
              00000040  003E          .IIF    NB,.Blkw,       .Blkw   1
                        0040          .IIF    NB,KA$W_Progress,       KA$W_Progress::
              00000042  0040          .IIF    NB,.Blkw,       .Blkw   1
                        0042          .IIF    NB,KA$B_Acc_Type,       KA$B_Acc_Type::
              00000043  0042          .IIF    NB,.Blkb,       .Blkb   1
                        0043          .IIF    NB,KA$B_Reserved,       KA$B_Reserved::
              00000044  0043          .IIF    NB,.Blkb,       .Blkb   1
                        0044          .IIF    NB,KA$W_Mem_Size,       KA$W_Mem_Size::
              00000046  0044          .IIF    NB,.Blkw,       .Blkw   1
                        0046          .IIF    NB,KA$B_Node_ID,        KA$B_Node_ID::
              00000047  0046          .IIF    NB,.Blkb,       .Blkb   1
                        0047          .IIF    NB,KA$L_Mem_Size,       KA$L_Mem_Size::
              0000004B  0047          .IIF    NB,.Blkl,       .Blkl   1
                        004B          .IIF    NB,HP$L_KAA_Slot,       HP$L_KAA_Slot::
              0000004F  004B          .IIF    NB,.Blkl,       .Blkl   1
                        004F          .IIF    NB,HP$B_Cpu_ID, HP$B_Cpu_ID::
              00000050  004F          .IIF    NB,.Blkb,       .Blkb   1
                        0050          .IIF    NB,KA$K_Len,    KA$K_Len::
                    0000125E          .RESTORE
```

```
                                  125E   287 KA730:  $DS_KA730                          ;                [01]
                30 33 37 41 4B 00'  125E              .ASCIC  "KA730" ; KA730 name
                                 05  125E
                                 50  1264              .BYTE   KA$K_LEN                  ; KA$K_LEN of resultant P-table
                                 04  1265              .BYTE   4                 ; Maximum unit number                          ;G:65
                               0000  1266              .WORD   ^A""     ; Two character mnemonic for QIO KA730
                                 80  1268              .BYTE   Pd$_Start         ; Constant to identify start of descriptor
                                 8D  1269              .Byte   Pd$_Name          ; Directive op-code
                                 01  126A              .Byte   Ptd$M_Unit                ; Enforced KA codes
                           41 4B 00'  126B              .Ascic  "KA"     ; Enforced device name prefix
                                 02  126B
                                 86  126E              .BYTE   Pd$_Literal       ; Indicate literal
                           40F30000  126F              .LONG   ^X40F30000                ; Constant
                                 88  1273              .BYTE   Pd$_Store         ; Indicate store operation
                               0018  1274              .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                                 00  1276              .BYTE   0                 ; 0 HP$A_DEVICE to data
                                 20  1277              .BYTE   32                ; 32 of data field
                                 86  1278              .BYTE   Pd$_Literal       ; Indicate literal
                           000001FF  1279              .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                    KA$M_CRC +
                                 88  127D              .BYTE   Pd$_Store         ; Indicate store operation
                               0032  127E              .WORD   KA$L_FLAGS                ; Byte KA$L_FLAGS to data
                                 00  1280              .BYTE   0                 ; 0 KA$L_FLAGS to data
                                 20  1281              .BYTE   32                ; 32 of data field
                                 8B  1282              .BYTE   Pd$_Logical       ; Indicate logical value operation
61 65 79 2D 66 6F 2D 65 6D 69 54 00'  1283              .ASCIC  \Time-of-year clock\     ; Time-of-year clock string
              6B 63 6F 6C 63 20 72  128F
                                 12  1283
                                 88  1296              .BYTE   Pd$_Store         ; Indicate store operation
                               0032  1297              .WORD   KA$L_FLAGS                ; Byte KA$L_FLAGS to data
                                 10  1299              .BYTE   KA$V_TODR                 ; KA$V_TODR KA$L_FLAGS to data
                                 01  129A              .BYTE   1                 ; 1 of data field
                                 84  129B              .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'  129C              .ASCIC  "WCS last address"       ; WCS last address string
                     73 73 65 72 64  12A8
                                 10  129C
                 0000FFFF 00000000  12AD              .LONG   ^X0,^X1@16-1     ; 0 , 1@16-1 limits on input
                                 88  12B5              .BYTE   Pd$_Store         ; Indicate store operation
                               0036  12B6              .WORD   KA$W_WCS                  ; Byte KA$W_WCS to data
                                 00  12B8              .BYTE   0                 ; 0 KA$W_WCS to data
                                 10  12B9              .BYTE   16                ; 16 of data field
                                 86  12BA              .BYTE   Pd$_Literal       ; Indicate literal
                           00002710  12BB              .LONG   10000             ; Constant
                                 88  12BF              .BYTE   Pd$_Store         ; Indicate store operation
                               003E  12C0              .WORD   KA$W_LATENCY              ; Byte KA$W_LATENCY to data
                                 00  12C2              .BYTE   0                 ; 0 KA$W_LATENCY to data
                                 20  12C3              .BYTE   32                ; 32 of data field
                                 86  12C4              .BYTE   Pd$_Literal       ; Indicate literal
                           000003E8  12C5              .LONG   1000              ; Constant
                                 88  12C9              .BYTE   Pd$_Store         ; Indicate store operation
                               0040  12CA              .WORD   KA$W_PROGRESS             ; Byte KA$W_PROGRESS to data
                                 00  12CC              .BYTE   0                 ; 0 KA$W_PROGRESS to data
                                 20  12CD              .BYTE   32                ; 32 of data field
                                 82  12CE              .BYTE   Pd$_Decimal       ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00'  12CF              .ASCIC  "Accelerator type"       ; Accelerator type string
                     65 70 79 74 20  12DB
                                 10  12CF
                 000000FF 00000000  12E0              .LONG   0,255    ; 0 , 255 limits on input
```

```
                                88  12E8                .BYTE   Pd$_Store       ; Indicate store operation
                              0042  12E9                .WORD   KA$B_ACC_TYPE           ; Byte KA$B_ACC_TYPE to data
                                00  12EB                .BYTE   0               ; 0 KA$B_ACC_TYPE to data
                                08  12EC                .BYTE   8               ; 8 of data field
                                82  12ED                .BYTE   Pd$_Decimal     ; Indicate scan decimal
20 66 6F 20 73 65 74 79 62 2D 4B 00' 12EE                .ASCIC  "K-bytes of Main Memory"        ; K-bytes of Main Memory string
   79 72 6F 6D 65 4D 20 6E 69 61 4D  12FA
                                16  12EE
                          00001400 00000000  1305        .LONG   0,5120  ; 0 , 5120 limits on input
                                88  130D                .BYTE   Pd$_Store       ; Indicate store operation
                              0044  130E                .WORD   KA$W_MEM_SIZE           ; Byte KA$W_MEM_SIZE to data
                                00  1310                .BYTE   0               ; 0 KA$W_MEM_SIZE to data
                                10  1311                .BYTE   16              ; 16 of data field
                                8B  1312                .BYTE   Pd$_Logical     ; Indicate logical value operation
6F 6C 20 53 43 57 20 72 65 73 55 00' 1313                .ASCIC  \User WCS loaded\        ; User WCS loaded string
                     64 65 64 61  131F
                                0F  1313
                                88  1323                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  1324                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                18  1326                .BYTE   KA$V_WCS_LD             ; KA$V_WCS_LD KA$L_FLAGS to data
                                01  1327                .BYTE   1               ; 1 of data field
                                8B  1328                .BYTE   Pd$_Logical     ; Indicate logical value operation
      73 72 6F 72 72 65 20 42 53 00' 1329                .ASCIC  \SB errors\     ; SB errors string
                                09  1329
                                88  1333                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  1334                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                19  1336                .BYTE   KA$V_SB_ERR             ; KA$V_SB_ERR KA$L_FLAGS to data
                                01  1337                .BYTE   1               ; 1 of data field
                                81  1338                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    1339   288 KA750:   $DS_KA750
                 30 35 37 41 4B 00' 1339                .ASCIC  "KA750" ; KA750 name
                                05  1339
                                50  133F                .BYTE   KA$K_LEN                ; KA$K_LEN of resultant P-table
                                04  1340                .BYTE   4               ; Maximum unit number                   ;G:65
                              0000  1341                .WORD   ^A""    ; Two character mnemonic for QIO KA750
                                80  1343                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  1344                .Byte   Pd$_Name        ; Directive op-code
                                01  1345                .Byte   Ptd$M_Unit              ; Enforced KA codes
                           41 4B 00' 1346                .Ascic  "KA"    ; Enforced device name prefix
                                02  1346
                                86  1349                .BYTE   Pd$_Literal     ; Indicate literal
                          40F30000  134A                .LONG   ^X40F30000              ; Constant
                                88  134E                .BYTE   Pd$_Store       ; Indicate store operation
                              0018  134F                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                00  1351                .BYTE   0               ; 0 HP$A_DEVICE to data
                                20  1352                .BYTE   32              ; 32 of data field
                                86  1353                .BYTE   Pd$_Literal     ; Indicate literal
                          000001F3  1354                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                                88  1358                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  1359                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                00  135B                .BYTE   0               ; 0 KA$L_FLAGS to data
                                20  135C                .BYTE   32              ; 32 of data field
                                8B  135D                .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 47 00' 135E                .ASCIC  \G-floating instructions\        ; G-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  136A
                                17  135E
                                88  1376                .BYTE   Pd$_Store       ; Indicate store operation
```

```
                                0032  1377                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  02  1379                .BYTE   KA$V_G_FLOAT            ; KA$V_G_FLOAT KA$L_FLAGS to data
                                  01  137A                .BYTE   1                   ; 1 of data field
                                  8B  137B                .BYTE   Pd$_Logical      ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 48 00' 137C                .ASCIC  \H-floating instructions\       ; H-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  1388
                                  17  137C
                                  88  1394                .BYTE   Pd$_Store        ; Indicate store operation
                                0032  1395                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  03  1397                .BYTE   KA$V_H_FLOAT            ; KA$V_H_FLOAT KA$L_FLAGS to data
                                  01  1398                .BYTE   1                   ; 1 of data field
                                  8B  1399                .BYTE   Pd$_Logical      ; Indicate logical value operation
61 65 79 2D 66 6F 2D 65 6D 69 54 00' 139A                .ASCIC  \Time-of-year clock\    ; Time-of-year clock string
                 6B 63 6F 6C 63 20 72  13A6
                                  12  139A
                                  88  13AD                .BYTE   Pd$_Store        ; Indicate store operation
                                0032  13AE                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  10  13B0                .BYTE   KA$V_TODR               ; KA$V_TODR KA$L_FLAGS to data
                                  01  13B1                .BYTE   1                   ; 1 of data field
                                  84  13B2                .BYTE   Pd$_Hexadecimal  ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00' 13B3                .ASCIC  "WCS last address"      ; WCS last address string
                     73 73 65 72 64  13BF
                                  10  13B3
                        0000FFFF 00000000  13C4                .LONG   ^X0,^X1a16-1    ; 0 , 1a16-1 limits on input
                                  88  13CC                .BYTE   Pd$_Store        ; Indicate store operation
                                0036  13CD                .WORD   KA$W_WCS                ; Byte KA$W_WCS to data
                                  00  13CF                .BYTE   0                   ; 0 KA$W_WCS to data
                                  10  13D0                .BYTE   16                  ; 16 of data field
                                  86  13D1                .BYTE   Pd$_Literal      ; Indicate literal
                            00002710  13D2                .LONG   10000               ; Constant
                                  88  13D6                .BYTE   Pd$_Store        ; Indicate store operation
                                003E  13D7                .WORD   KA$W_LATENCY             ; Byte KA$W_LATENCY to data
                                  00  13D9                .BYTE   0                   ; 0 KA$W_LATENCY to data
                                  20  13DA                .BYTE   32                  ; 32 of data field
                                  86  13DB                .BYTE   Pd$_Literal      ; Indicate literal
                            000003E8  13DC                .LONG   1000                ; Constant
                                  88  13E0                .BYTE   Pd$_Store        ; Indicate store operation
                                0040  13E1                .WORD   KA$W_PROGRESS            ; Byte KA$W_PROGRESS to data
                                  00  13E3                .BYTE   0                   ; 0 KA$W_PROGRESS to data
                                  20  13E4                .BYTE   32                  ; 32 of data field
                                  82  13E5                .BYTE   Pd$_Decimal      ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00' 13E6                .ASCIC  "Accelerator type"      ; Accelerator type string
                        65 70 79 74 20  13F2
                                  10  13E6
                        000000FF 00000000  13F7                .LONG   0,255   ; 0 , 255 limits on input
                                  88  13FF                .BYTE   Pd$_Store        ; Indicate store operation
                                0042  1400                .WORD   KA$B_ACC_TYPE            ; Byte KA$B_ACC_TYPE to data
                                  00  1402                .BYTE   0                   ; 0 KA$B_ACC_TYPE to data
                                  08  1403                .BYTE   8                   ; 8 of data field
                                  81  1404                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                      1405    289 KA780:  $DS_KA780
                     30 38 37 41 4B 00' 1405                .ASCIC  "KA780" ; KA780 name
                                  05  1405
                                  50  140B                .BYTE   KA$K_LEN                ; KA$K_LEN of resultant P-table
                                  04  140C                .BYTE   4                   ; Maximum unit number                   ;G:65
                                0000  140D                .WORD   ^A""    ; Two character mnemonic for QIO KA780
                                  80  140F                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
```

```
                                  8D  1410          .Byte   Pd$_Name        ; Directive op-code
                                  01  1411          .Byte   Ptd$M_Unit              ; Enforced KA codes
                          41 4B 00'  1412          .Ascic  "KA"    ; Enforced device name prefix
                                  02  1412
                                  86  1415          .BYTE   Pd$_Literal     ; Indicate literal
                          000101F3  1416          .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                                  88  141A          .BYTE   Pd$_Store       ; Indicate store operation
                                0032  141B          .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  00  141D          .BYTE   0               ; 0 KA$L_FLAGS to data
                                  20  141E          .BYTE   32              ; 32 of data field
                                  8B  141F          .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 47 00'  1420          .ASCIC  \G-floating instructions\       ; G-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  142C
                                  17  1420
                                  88  1438          .BYTE   Pd$_Store       ; Indicate store operation
                                0032  1439          .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  02  143B          .BYTE   KA$V_G_FLOAT            ; KA$V_G_FLOAT KA$L_FLAGS to data
                                  01  143C          .BYTE   1               ; 1 of data field
                                  8B  143D          .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 48 00'  143E          .ASCIC  \H-floating instructions\       ; H-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  144A
                                  17  143E
                                  88  1456          .BYTE   Pd$_Store       ; Indicate store operation
                                0032  1457          .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  03  1459          .BYTE   KA$V_H_FLOAT            ; KA$V_H_FLOAT KA$L_FLAGS to data
                                  01  145A          .BYTE   1               ; 1 of data field
                                  84  145B          .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'  145C          .ASCIC  "WCS last address"      ; WCS last address string
                     73 73 65 72 64  1468
                                  10  145C
                  0000FFFF 00000000  146D          .LONG   ^X0,^X1@16-1    ; 0 , 1@16-1 limits on input
                                  88  1475          .BYTE   Pd$_Store       ; Indicate store operation
                                0036  1476          .WORD   KA$W_WCS                ; Byte KA$W_WCS to data
                                  00  1478          .BYTE   0               ; 0 KA$W_WCS to data
                                  10  1479          .BYTE   16              ; 16 of data field
                                  86  147A          .BYTE   Pd$_Literal     ; Indicate literal
                          00002710  147B          .LONG   10000           ; Constant
                                  88  147F          .BYTE   Pd$_Store       ; Indicate store operation
                                003E  1480          .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                                  00  1482          .BYTE   0               ; 0 KA$W_LATENCY to data
                                  20  1483          .BYTE   32              ; 32 of data field
                                  86  1484          .BYTE   Pd$_Literal     ; Indicate literal
                          000003E8  1485          .LONG   1000            ; Constant
                                  88  1489          .BYTE   Pd$_Store       ; Indicate store operation
                                0040  148A          .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                                  00  148C          .BYTE   0               ; 0 KA$W_PROGRESS to data
                                  20  148D          .BYTE   32              ; 32 of data field
                                  82  148E          .BYTE   Pd$_Decimal     ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00'  148F          .ASCIC  "Accelerator type"      ; Accelerator type string
                     65 70 79 74 20  149B
                                  10  148F
                  000000FF 00000000  14A0          .LONG   0,255   ; 0 , 255 limits on input
                                  88  14A8          .BYTE   Pd$_Store       ; Indicate store operation
                                0042  14A9          .WORD   KA$B_ACC_TYPE           ; Byte KA$B_ACC_TYPE to data
                                  00  14AB          .BYTE   0               ; 0 KA$B_ACC_TYPE to data
                                  08  14AC          .BYTE   8               ; 8 of data field
                                  81  14AD          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
```

```
                                  14AE   290 KA86:   $DS_KA86                          ;                [23][50]
                 36 38 41 4B 00'  14AE               .ASCIC  "KA86"  ; KA86 name
                              04  14AE
                              50  14B3               .BYTE   KA$K_LEN                  ; KA$K_LEN of resultant P-table
                              04  14B4               .BYTE   4                 ; Maximum unit number                        ;G:65
                            0000  14B5               .WORD   ^A""     ; Two character mnemonic for QIO KA86
                              80  14B7               .BYTE   Pd$_Start         ; Constant to identify start of descriptor
                              8D  14B8               .Byte   Pd$_Name          ; Directive op-code
                              01  14B9               .Byte   Ptd$M_Unit                ; Enforced KA codes
                        41 4B 00'  14BA               .Ascic  "KA"     ; Enforced device name prefix
                              02  14BA
                              86  14BD               .BYTE   Pd$_Literal       ; Indicate literal
                        000101F3  14BE               .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                    KA$M_CRC +
                              88  14C2               .BYTE   Pd$_Store         ; Indicate store operation
                            0032  14C3               .WORD   KA$L_FLAGS                ; Byte KA$L_FLAGS to data
                              00  14C5               .BYTE   0                 ; 0 KA$L_FLAGS to data
                              20  14C6               .BYTE   32                ; 32 of data field
                              8B  14C7               .BYTE   Pd$_Logical       ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 47 00'  14C8           .ASCIC  \G-floating instructions\        ; G-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  14D4
                              17  14C8
                              88  14E0               .BYTE   Pd$_Store         ; Indicate store operation
                            0032  14E1               .WORD   KA$L_FLAGS                ; Byte KA$L_FLAGS to data
                              02  14E3               .BYTE   KA$V_G_FLOAT              ; KA$V_G_FLOAT KA$L_FLAGS to data
                              01  14E4               .BYTE   1                 ; 1 of data field
                              8B  14E5               .BYTE   Pd$_Logical       ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 48 00'  14E6           .ASCIC  \H-floating instructions\        ; H-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  14F2
                              17  14E6
                              88  14FE               .BYTE   Pd$_Store         ; Indicate store operation
                            0032  14FF               .WORD   KA$L_FLAGS                ; Byte KA$L_FLAGS to data
                              03  1501               .BYTE   KA$V_H_FLOAT              ; KA$V_H_FLOAT KA$L_FLAGS to data
                              01  1502               .BYTE   1                 ; 1 of data field
                              84  1503               .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'  1504           .ASCIC  "WCS last address"        ; WCS last address string
                 73 73 65 72 64  1510
                              10  1504
              0000FFFF 00000000  1515               .LONG   ^X0,^X1@16-1      ; 0 , 1@16-1 limits on input
                              88  151D               .BYTE   Pd$_Store         ; Indicate store operation
                            0036  151E               .WORD   KA$W_WCS                  ; Byte KA$W_WCS to data
                              00  1520               .BYTE   0                 ; 0 KA$W_WCS to data
                              10  1521               .BYTE   16                ; 16 of data field
                              86  1522               .BYTE   Pd$_Literal       ; Indicate literal
                        00002710  1523               .LONG   10000             ; Constant
                              88  1527               .BYTE   Pd$_Store         ; Indicate store operation
                            003E  1528               .WORD   KA$W_LATENCY              ; Byte KA$W_LATENCY to data
                              00  152A               .BYTE   0                 ; 0 KA$W_LATENCY to data
                              20  152B               .BYTE   32                ; 32 of data field
                              86  152C               .BYTE   Pd$_Literal       ; Indicate literal
                        000003E8  152D               .LONG   1000              ; Constant
                              88  1531               .BYTE   Pd$_Store         ; Indicate store operation
                            0040  1532               .WORD   KA$W_PROGRESS             ; Byte KA$W_PROGRESS to data
                              00  1534               .BYTE   0                 ; 0 KA$W_PROGRESS to data
                              20  1535               .BYTE   32                ; 32 of data field
                              82  1536               .BYTE   Pd$_Decimal       ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00'  1537           .ASCIC  "Accelerator type"        ; Accelerator type string
                 65 70 79 74 20  1543
```

```
                                  10
                 000000FF 00000000
                                  88
                                0042
                                  00
                                  08
                                  81
                                      291 KA785:
                 35 38 37 41 4B 00'
                                  05
                                  50
                                  04
                                0000
                                  80
                                  8D
                                  01
                           41 4B 00'
                                  02
                                  86
                           000101F3
                                  88
                                0032
                                  00
                                  20
                                  8B
20 67 6E 69 74 61 6F 6C 66 2D 47 00'
73 6E 6F 69 74 63 75 72 74 73 6E 69
                                  17
                                  88
                                0032
                                  02
                                  01
                                  8B
20 67 6E 69 74 61 6F 6C 66 2D 48 00'
73 6E 6F 69 74 63 75 72 74 73 6E 69
                                  17
                                  88
                                0032
                                  03
                                  01
                                  84
64 61 20 74 73 61 6C 20 53 43 4A 00'
                     73 73 65 72 64
                                  10
                 0000FFFF 00000000
                                  88
                                0036
                                  00
                                  10
                                  86
                           00002710
                                  88
                                003E
                                  00
                                  20
                                  86
                           000003E8
```

```
                .LONG   0,255   ; 0 , 255 limits on input
                .BYTE   Pd$_Store       ; Indicate store operation
                .WORD   KA$B_ACC_TYPE           ; Byte KA$B_ACC_TYPE to data
                .BYTE   0               ; 0 KA$B_ACC_TYPE to data
                .BYTE   8               ; 8 of data field
                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                $DS_KA785                       ;                [15]
                .ASCIC  "KA785" ; KA785 name

                .BYTE   KA$K_LEN                ; KA$K_LEN of resultant P-table
                .BYTE   4               ; Maximum unit number                   ;G:65
                .WORD   ^A""    ; Two character mnemonic for QIO KA785
                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                .Byte   Pd$_Name        ; Directive op-code
                .Byte   Ptd$M_Unit              ; Enforced KA codes
                .Ascic  "KA"    ; Enforced device name prefix

                .BYTE   Pd$_Literal     ; Indicate literal
                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                .BYTE   Pd$_Store       ; Indicate store operation
                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                .BYTE   0               ; 0 KA$L_FLAGS to data
                .BYTE   32              ; 32 of data field
                .BYTE   Pd$_Logical     ; Indicate logical value operation
                .ASCIC  \G-floating instructions\       ; G-floating instructions string

                .BYTE   Pd$_Store       ; Indicate store operation
                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                .BYTE   KA$V_G_FLOAT            ; KA$V_G_FLOAT KA$L_FLAGS to data
                .BYTE   1               ; 1 of data field
                .BYTE   Pd$_Logical     ; Indicate logical value operation
                .ASCIC  \H-floating instructions\       ; H-floating instructions string

                .BYTE   Pd$_Store       ; Indicate store operation
                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                .BYTE   KA$V_H_FLOAT            ; KA$V_H_FLOAT KA$L_FLAGS to data
                .BYTE   1               ; 1 of data field
                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
                .ASCIC  "JCS last address"      ; JCS last address string

                .LONG   ^X0,^X1@16-1    ; 0 , 1@16-1 limits on input
                .BYTE   Pd$_Store       ; Indicate store operation
                .WORD   KA$W_WCS                ; Byte KA$W_WCS to data
                .BYTE   0               ; 0 KA$W_WCS to data
                .BYTE   16              ; 16 of data field
                .BYTE   Pd$_Literal     ; Indicate literal
                .LONG   10000           ; Constant
                .BYTE   Pd$_Store       ; Indicate store operation
                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                .BYTE   0               ; 0 KA$W_LATENCY to data
                .BYTE   32              ; 32 of data field
                .BYTE   Pd$_Literal     ; Indicate literal
                .LONG   1000            ; Constant
```

```
                          88  15DA                .BYTE   Pd$_Store       ; Indicate store operation
                        0040  15DB                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                          00  15DD                .BYTE   0               ; 0 KA$W_PROGRESS to data
                          20  15DE                .BYTE   32              ; 32 of data field
                          82  15DF                .BYTE   Pd$_Decimal     ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00' 15E0         .ASCIC  "Accelerator type"      ; Accelerator type string
                65 70 79 74 20    15EC
                          10  15E0
                000000FF 00000000  15F1         .LONG   0,255   ; 0 , 255 limits on input
                          88  15F9              .BYTE   Pd$_Store       ; Indicate store operation
                        0042  15FA              .WORD   KA$B_ACC_TYPE           ; Byte KA$B_ACC_TYPE to data
                          00  15FC              .BYTE   0               ; 0 KA$B_ACC_TYPE to data
                          08  15FD              .BYTE   8               ; 8 of data field
                          81  15FE              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              15FF  292 KMC11:  $DS_KMC11
                              15FF              .SAVE   LOCAL_BLOCK
                              15FF              .PSECT  $ABS$,ABS
                    00000032  0050              .=HP$A_DEPENDENT
                              0032              .IIF    NB,KMC11$L_CSR, KMC11$L_CSR:
                    00000036  0032              .IIF    NB,.BLKL,       .BLKL   1
                              0036              .IIF    NB,KMC11$B_BR,  KMC11$B_BR:
                    00000037  0036              .IIF    NB,.BLKB,       .BLKB   1
                              0037              .IIF    NB,KMC11$K_LEN, KMC11$K_LEN:
                              15FF              .RESTORE
                31 31 43 4D 4B 00' 15FF         .ASCIC  "KMC11" ; KMC11 name
                          05  15FF
                          37  1605              .BYTE   KMC11$K_LEN             ; KMC11$K_LEN of resultant P-table
                          00  1606              .BYTE   0               ; Maximum unit number                   ;G:65
                        0000  1607              .WORD   ^A""    ; Two character mnemonic for QIO KMC11
                          80  1609              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  160A              .Byte   Pd$_Name        ; Directive op-code
                          03  160B              .Byte   Ptd$M_Device            ; Enforced XK codes
                     4B 58 00' 160C             .Ascic  "XK"    ; Enforced device name prefix
                          02  160C
                          83  160F              .BYTE   Pd$_Octal       ; Indicate scan octal
                   52 53 43 00' 1610            .ASCIC  "CSR"   ; CSR string
                          03  1610
                0003FFFE 0003E000  1614         .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  161C              .BYTE   Pd$_Store       ; Indicate store operation
                        0032  161D              .WORD   KMC11$L_CSR             ; Byte KMC11$L_CSR to data
                          00  161F              .BYTE   0               ; 0 KMC11$L_CSR to data
                          20  1620              .BYTE   32              ; 32 of data field
                          88  1621              .BYTE   Pd$_Store       ; Indicate store operation
                        0018  1622              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  1624              .BYTE   0               ; 0 HP$A_DEVICE to data
                          12  1625              .BYTE   18              ; 18 of data field
                          83  1626              .BYTE   Pd$_Octal       ; Indicate scan octal
          52 4F 54 43 45 56 00' 1627            .ASCIC  "VECTOR"        ; VECTOR string
                          06  1627
                000001FE 00000002  162E         .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  1636              .BYTE   Pd$_Store       ; Indicate store operation
                        0024  1637              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  1639              .BYTE   0               ; 0 HP$W_VECTOR to data
                          09  163A              .BYTE   9               ; 9 of data field
                          82  163B              .BYTE   Pd$_Decimal     ; Indicate scan decimal
                   52 42 00' 163C             .ASCIC  "BR"    ; BR string
                          02  163C
```

```
                 00000007 00000004  163F                .LONG   4,7      ; 4 , 7 limits on input
                               88  1647                .BYTE   Pd$_Store        ; Indicate store operation
                             0036  1648                .WORD   KMC11$B_BR               ; Byte KMC11$B_BR to data
                               00  164A                .BYTE   0                ; 0 KMC11$B_BR to data
                               08  164B                .BYTE   8                ; 8 of data field
                               81  164C                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                   164D   293 KW11K:   $DS_KW11K
                                   164D                .SAVE   LOCAL_BLOCK
                                   164D                .PSECT  $ABS$,ABS
                         00000032  0050                .=HP$A_DEPENDENT
                                   0032                .IIF    NB,KW11K$B_BR,  KW11K$B_BR:
                         00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                                   0033                .IIF    NB,KW11K$K_LEN, KW11K$K_LEN:
                               0000164D                .RESTORE
                 4B 31 31 57 4B 00' 164D                .ASCIC  "KW11K" ; KW11K name
                               05  164D
                               33  1653                .BYTE   KW11K$K_LEN              ; KW11K$K_LEN of resultant P-table
                               02  1654                .BYTE   2                ; Maximum unit number                          ;G:65
                             0000  1655                .WORD   ^A""     ; Two character mnemonic for QIO KW11K
                               80  1657                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                               8D  1658                .Byte   Pd$_Name         ; Directive op-code
                               03  1659                .Byte   Ptd$M_Device             ; Enforced KW codes
                         57 4B 00' 165A                .Ascic  "KW"     ; Enforced device name prefix
                               02  165A
                               83  165D                .BYTE   Pd$_Octal        ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 165E                .ASCIC  "I/O BUS CSR"    ; I/O BUS CSR string
                               0B  165E
                 0003FFFE 0003E000  166A                .LONG   ^O760000,^O777776        ; 760000 , 777776 limits on input
                               88  1672                .BYTE   Pd$_Store        ; Indicate store operation
                             0018  1673                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                               00  1675                .BYTE   0                ; 0 HP$A_DEVICE to data
                               12  1676                .BYTE   18               ; 18 of data field
                               83  1677                .BYTE   Pd$_Octal        ; Indicate scan octal
             52 4F 54 43 45 56 00' 1678                .ASCIC  "VECTOR"         ; VECTOR string
                               06  1678
                 000001FE 00000002  167F                .LONG   ^O2,^O776        ; 2 , 776 limits on input
                               88  1687                .BYTE   Pd$_Store        ; Indicate store operation
                             0024  1688                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                               00  168A                .BYTE   0                ; 0 HP$W_VECTOR to data
                               09  168B                .BYTE   9                ; 9 of data field
                               82  168C                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                         52 42 00' 168D                .ASCIC  "BR"     ; BR string
                               02  168D
                 00000007 00000004  1690                .LONG   4,7      ; 4 , 7 limits on input
                               88  1698                .BYTE   Pd$_Store        ; Indicate store operation
                             0032  1699                .WORD   KW11K$B_BR               ; Byte KW11K$B_BR to data
                               00  169B                .BYTE   0                ; 0 KW11K$B_BR to data
                               08  169C                .BYTE   8                ; 8 of data field
                               81  169D                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                   169E   294 LA12:    $DS_LA12                         ;                [14]
                                   169E                .SAVE   LOCAL_BLOCK
                                   169E                .PSECT  $ABS$,ABS
                         00000032  0050                .=HP$A_DEPENDENT
                                   0032                .IIF    NB,HP$K_LA12_LEN,       HP$K_LA12_LEN:
                                   0032                .IIF    NB,LA12$K_LEN,  LA12$K_LEN:
                               0000169E                .RESTORE
                    32 31 41 4C 00' 169E                .ASCIC  "LA12"   ; LA12 name
```

```
                      04  169E
                      32  16A3                .BYTE   LA12$K_LEN                ; LA12$K_LEN of resultant P-table
                      FF  16A4                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  16A5                .WORD   ^A""    ; Two character mnemonic for QIO LA12
                      80  16A7                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  16A8                .Byte   Pd$_Name            ; Directive op-code
                      1B  16A9                .Byte   Ptd$M_EndDevice             ; Enforced TT codes
                54 54 00' 16AA                .Ascic  "TT"    ; Enforced device name prefix
                      02  16AA
                      86  16AD                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  16AE                .LONG   HP$M_ALLOC                  ; Constant
                      88  16B2                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  16B3                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  16B5                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  16B6                .BYTE   8                   ; 8 of data field
                      81  16B7                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          16B8    295 LA100:  $DS_LA100                           ;                   [14]
                          16B8                .SAVE   LOCAL_BLOCK
                          16B8                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_LA100_LEN,      HP$K_LA100_LEN:
                          0032                .IIF    NB,LA100$K_LEN, LA100$K_LEN:
                      000016B8                .RESTORE
          30 30 31 41 4C 00' 16B8               .ASCIC  "LA100" ; LA100 name
                      05  16B8
                      32  16BE                .BYTE   LA100$K_LEN                ; LA100$K_LEN of resultant P-table
                      FF  16BF                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  16C0                .WORD   ^A""    ; Two character mnemonic for QIO LA100
                      80  16C2                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  16C3                .Byte   Pd$_Name            ; Directive op-code
                      1B  16C4                .Byte   Ptd$M_EndDevice             ; Enforced TT codes
                54 54 00' 16C5                .Ascic  "TT"    ; Enforced device name prefix
                      02  16C5
                      86  16C8                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  16C9                .LONG   HP$M_ALLOC                  ; Constant
                      88  16CD                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  16CE                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  16D0                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  16D1                .BYTE   8                   ; 8 of data field
                      81  16D2                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          16D3    296 LA120:  $DS_LA120
                          16D3                .SAVE   LOCAL_BLOCK
                          16D3                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LA120$K_LEN, LA120$K_LEN:
                      000016D3                .RESTORE
          30 32 31 41 4C 00' 16D3               .ASCIC  "LA120" ; LA120 name
                      05  16D3
                      32  16D9                .BYTE   LA120$K_LEN                ; LA120$K_LEN of resultant P-table
                      FF  16DA                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  16DB                .WORD   ^A""    ; Two character mnemonic for QIO LA120
                      80  16DD                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  16DE                .Byte   Pd$_Name            ; Directive op-code
                      1B  16DF                .Byte   Ptd$M_EndDevice             ; Enforced TT codes
                54 54 00' 16E0                .Ascic  "TT"    ; Enforced device name prefix
                      02  16E0
                      86  16E3                .BYTE   Pd$_Literal         ; Indicate literal
```

```
                    00000001  16E4                .LONG   HP$M_ALLOC                      ; Constant
                          88  16E8                .BYTE   Pd$_Store           ; Indicate store operation
                        000A  16E9                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                          00  16EB                .BYTE   0                   ; 0 HP$B_FLAGS to data
                          08  16EC                .BYTE   8                   ; 8 of data field
                          81  16ED                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                              16EE    297 LA180:  $DS_LA180
                              16EE                .SAVE   LOCAL_BLOCK
                              16EE                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,LA180$K_LEN, LA180$K_LEN:
                          0000 16EE               .RESTORE
                 30 38 31 41 4C 00' 16EE          .ASCIC  "LA180" ; LA180 name
                          05  16EE
                          32  16F4                .BYTE   LA180$K_LEN                     ; LA180$K_LEN of resultant P-table
                          FF  16F5                .BYTE   255                 ; Maximum unit number                       ;G:65
                        0000  16F6                .WORD   ^A""    ; Two character mnemonic for QIO LA180
                          80  16F8                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                          8D  16F9                .Byte   Pd$_Name            ; Directive op-code
                          1B  16FA                .Byte   Ptd$M_EndDevice             ; Enforced LP codes
                    50 4C 00' 16FB                .Ascic  "LP"    ; Enforced device name prefix
                          02  16FB
                          86  16FE                .BYTE   Pd$_Literal         ; Indicate literal
                    00000001  16FF                .LONG   HP$M_ALLOC                      ; Constant
                          88  1703                .BYTE   Pd$_Store           ; Indicate store operation
                        000A  1704                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                          00  1706                .BYTE   0                   ; 0 HP$B_FLAGS to data
                          08  1707                .BYTE   8                   ; 8 of data field
                          81  1708                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                              1709    298 LA34:   $DS_LA34
                              1709                .SAVE   LOCAL_BLOCK
                              1709                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,LA34$K_LEN,  LA34$K_LEN:
                          0000 1709               .RESTORE
                 34 33 41 4C 00' 1709             .ASCIC  "LA34"  ; LA34 name
                          04  1709
                          32  170E                .BYTE   LA34$K_LEN                      ; LA34$K_LEN of resultant P-table
                          FF  170F                .BYTE   255                 ; Maximum unit number                       ;G:65
                        0000  1710                .WORD   ^A""    ; Two character mnemonic for QIO LA34
                          80  1712                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                          8D  1713                .Byte   Pd$_Name            ; Directive op-code
                          1B  1714                .Byte   Ptd$M_EndDevice             ; Enforced TT codes
                    54 54 00' 1715                .Ascic  "TT"    ; Enforced device name prefix
                          02  1715
                          86  1718                .BYTE   Pd$_Literal         ; Indicate literal
                    00000001  1719                .LONG   HP$M_ALLOC                      ; Constant
                          88  171D                .BYTE   Pd$_Store           ; Indicate store operation
                        000A  171E                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                          00  1720                .BYTE   0                   ; 0 HP$B_FLAGS to data
                          08  1721                .BYTE   8                   ; 8 of data field
                          81  1722                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                              1723    299 LA36:   $DS_LA36
                              1723                .SAVE   LOCAL_BLOCK
                              1723                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,LA36$K_LEN,  LA36$K_LEN:
```

```
                    00001723                .RESTORE
         36 33 41 4C 00' 1723               .ASCIC  "LA36"  ; LA36 name
                    04  1723
                    32  1728                .BYTE   LA36$K_LEN              ; LA36$K_LEN of resultant P-table
                    FF  1729                .BYTE   255             ; Maximum unit number                   ;G:65
                  0000  172A                .WORD   ^A""    ; Two character mnemonic for QIO LA36
                    80  172C                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  172D                .Byte   Pd$_Name        ; Directive op-code
                    1B  172E                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
              54 54 00' 172F                .Ascic  "TT"    ; Enforced device name prefix
                    02  172F
                    86  1732                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  1733                .LONG   HP$M_ALLOC              ; Constant
                    88  1737                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  1738                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  173A                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  173B                .BYTE   8               ; 8 of data field
                    81  173C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        173D    300 LA38:   $DS_LA38
                        173D                .SAVE   LOCAL_BLOCK
                        173D                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,LA38$K_LEN,  LA38$K_LEN:
                    0000173D                .RESTORE
         38 33 41 4C 00' 173D               .ASCIC  "LA38"  ; LA38 name
                    04  173D
                    32  1742                .BYTE   LA38$K_LEN              ; LA38$K_LEN of resultant P-table
                    FF  1743                .BYTE   255             ; Maximum unit number                   ;G:65
                  0000  1744                .WORD   ^A""    ; Two character mnemonic for QIO LA38
                    80  1746                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  1747                .Byte   Pd$_Name        ; Directive op-code
                    1B  1748                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
              54 54 00' 1749                .Ascic  "TT"    ; Enforced device name prefix
                    02  1749
                    86  174C                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  174D                .LONG   HP$M_ALLOC              ; Constant
                    88  1751                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  1752                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  1754                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  1755                .BYTE   8               ; 8 of data field
                    81  1756                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        1757    301 LA210:  $DS_LA210                       ;               [32]
                        1757                .SAVE   LOCAL_BLOCK
                        1757                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_LA210_LEN,      HP$K_LA210_LEN:
                        0032                .IIF    NB,LA210$K_LEN, LA210$K_LEN:
                    00001757                .RESTORE
      30 31 32 41 4C 00' 1757               .ASCIC  "LA210" ; LA210 name
                    05  1757
                    32  175D                .BYTE   LA210$K_LEN             ; LA210$K_LEN of resultant P-table
                    08  175E                .BYTE   8               ; Maximum unit number                   ;G:65
                  0000  175F                .WORD   ^A""    ; Two character mnemonic for QIO LA210
                    80  1761                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  1762                .Byte   Pd$_Name        ; Directive op-code
                    1B  1763                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
              54 54 00' 1764                .Ascic  "TT"    ; Enforced device name prefix
```

```
                      02  1764
                      86  1767                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1768                .LONG   HP$M_ALLOC              ; Constant
                      88  176C                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  176D                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  176F                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1770                .BYTE   8               ; 8 of data field
                      81  1771                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1772    302 LESI:   $DS_LESI                ;               [13]
                          1772                .SAVE   LOCAL_BLOCK
                          1772                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_LESI_IP,        HP$L_LESI_IP:
                          0032                .IIF    NB,LESI$L_IP,   LESI$L_IP:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_LESI_BR,        HP$B_LESI_BR:
                          0036                .IIF    NB,LESI$B_BR,   LESI$B_BR:
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_LESI_LEN,       HP$K_LESI_LEN:
                          0037                .IIF    NB,LESI$K_LEN,  LESI$K_LEN:
                      00001772                .RESTORE
             49 53 45 4C 00' 1772                .ASCIC  "LESI"  ; LESI name
                      04  1772
                      37  1777                .BYTE   HP$K_LESI_LEN           ; HP$K_LESI_LEN of resultant P-table
                      00  1778                .BYTE   0               ; Maximum unit number                    ;G:65
                    0000  1779                .WORD   ^A""    ; Two character mnemonic for QIO LESI
                      80  177B                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  177C                .Byte   Pd$_Name        ; Directive op-code
                      02  177D                .Byte   Ptd$M_Controller                ; Enforced DA codes
                41 44 00' 177E                .Ascic  "DA"    ; Enforced device name prefix
                      02  177E
                      83  1781                .BYTE   Pd$_Octal       ; Indicate scan octal
                50 49 00' 1782                .ASCIC  "IP"    ; IP string
                      02  1782
            0003FFFC 0003E000  1785           .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                      88  178D                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  178E                .WORD   HP$L_LESI_IP            ; Byte HP$L_LESI_IP to data
                      00  1790                .BYTE   0               ; 0 HP$L_LESI_IP to data
                      20  1791                .BYTE   32              ; 32 of data field
                      88  1792                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1793                .WORD   Hp$A_Device              ; Byte Hp$A_Device to data
                      00  1795                .BYTE   0               ; 0 Hp$A_Device to data
                      12  1796                .BYTE   18              ; 18 of data field
                      83  1797                .BYTE   Pd$_Octal       ; Indicate scan octal
          72 6F 74 63 65 56 00' 1798          .ASCIC  "Vector"        ; Vector string
                      06  1798
            000001FC 00000004  179F           .LONG   ^O4,^O774       ; 4 , 774 limits on input
                      88  17A7                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  17A8                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  17AA                .BYTE   0               ; 0 HP$W_VECTOR to data
                      08  17AB                .BYTE   8               ; 8 of data field
                      82  17AC                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 17AD                .ASCIC  "BR"    ; BR string
                      02  17AD
            00000007 00000004  17B0           .LONG   4,7     ; 4 , 7 limits on input
                      88  17B8                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  17B9                .WORD   HP$B_LESI_BR            ; Byte HP$B_LESI_BR to data
```

```
                      00  17BB                .BYTE   0                   ; 0 HP$B_LESI_BR to data
                      08  17BC                .BYTE   8                   ; 8 of data field
                      81  17BD                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          17BE    303 LG01:    $DS_LG01                            ;               [36]            ;G:5
                          17BE                .SAVE   LOCAL_BLOCK
                          17BE                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LG01$K_LEN,  LG01$K_LEN:
                  000017BE                    .RESTORE
             31 30 47 4C 00' 17BE             .ASCIC  "LG01"  ; LG01 name
                      04  17BE
                      32  17C3                .BYTE   LG01$K_LEN              ; LG01$K_LEN of resultant P-table
                      01  17C4                .BYTE   1                   ; Maximum unit number               ;G:65
                    0000  17C5                .WORD   ^A""    ; Two character mnemonic for QIO LG01
                      80  17C7                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  17C8                .Byte   Pd$_Name            ; Directive op-code
                      1B  17C9                .Byte   Ptd$M_EndDevice         ; Enforced LG codes
                 47 4C 00' 17CA               .Ascic  "LG"    ; Enforced device name prefix
                      02  17CA
                      86  17CD                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  17CE                .LONG   HP$M_ALLOC              ; Constant
                      88  17D2                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  17D3                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  17D5                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  17D6                .BYTE   8                   ; 8 of data field
                      81  17D7                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          17D8    304 LG02:    $DS_LG02                            ;               [36]            ;G:5
                          17D8                .SAVE   LOCAL_BLOCK
                          17D8                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LG02$K_LEN,  LG02$K_LEN:
                  000017D8                    .RESTORE
             32 30 47 4C 00' 17D8             .ASCIC  "LG02"  ; LG02 name
                      04  17D8
                      32  17DD                .BYTE   LG02$K_LEN              ; LG02$K_LEN of resultant P-table
                      01  17DE                .BYTE   1                   ; Maximum unit number               ;G:65
                    0000  17DF                .WORD   ^A""    ; Two character mnemonic for QIO LG02
                      80  17E1                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  17E2                .Byte   Pd$_Name            ; Directive op-code
                      1B  17E3                .Byte   Ptd$M_EndDevice         ; Enforced LG codes
                 47 4C 00' 17E4               .Ascic  "LG"    ; Enforced device name prefix
                      02  17E4
                      86  17E7                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  17E8                .LONG   HP$M_ALLOC              ; Constant
                      88  17EC                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  17ED                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  17EF                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  17F0                .BYTE   8                   ; 8 of data field
                      81  17F1                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          17F2    305 LG03:    $DS_LG03                            ;               [36]            ;G:5
                          17F2                .SAVE   LOCAL_BLOCK
                          17F2                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LG03$K_LEN,  LG03$K_LEN:
                  000017F2                    .RESTORE
             33 30 47 4C 00' 17F2             .ASCIC  "LG03"  ; LG03 name
                      04  17F2
```

```
                      32  17F7                .BYTE   LG03$K_LEN              ; LG03$K_LEN of resultant P-table
                      01  17F8                .BYTE   1                       ; Maximum unit number                   ;G:65
                    0000  17F9                .WORD   ^A""    ; Two character mnemonic for QIO LG03
                      80  17FB                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  17FC                .Byte   Pd$_Name        ; Directive op-code
                      1B  17FD                .Byte   Ptd$M_EndDevice         ; Enforced LG codes
                 47 4C 00' 17FE               .Ascic  "LG"    ; Enforced device name prefix
                      02  17FE
                      86  1801                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1802                .LONG   HP$M_ALLOC              ; Constant
                      88  1806                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1807                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1809                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  180A                .BYTE   8                       ; 8 of data field
                      81  180B                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          180C    306 LN01:   $DS_LN01                        ;               [14]                    ;G:6
                          180C                .SAVE   LOCAL_BLOCK
                          180C                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LN01$K_LEN,  LN01$K_LEN:
                      0000180C                .RESTORE
             31 30 4E 4C 00' 180C             .ASCIC  "LN01"  ; LN01 name
                      04  180C
                      32  1811                .BYTE   LN01$K_LEN              ; LN01$K_LEN of resultant P-table
                      FF  1812                .BYTE   255                     ; Maximum unit number                   ;G:65
                    0000  1813                .WORD   ^A""    ; Two character mnemonic for QIO LN01
                      80  1815                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1816                .Byte   Pd$_Name        ; Directive op-code
                      1B  1817                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                 50 4C 00' 1818               .Ascic  "LP"    ; Enforced device name prefix
                      02  1818
                      86  181B                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  181C                .LONG   HP$M_ALLOC              ; Constant
                      88  1820                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1821                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1823                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  1824                .BYTE   8                       ; 8 of data field
                      81  1825                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          1826    307 LN03:   $DS_LN03                        ;               [32]
                          1826                .SAVE   LOCAL_BLOCK
                          1826                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LN03$K_LEN,  LN03$K_LEN:
                      00001826                .RESTORE
             33 30 4E 4C 00' 1826             .ASCIC  "LN03"  ; LN03 name
                      04  1826
                      32  182B                .BYTE   LN03$K_LEN              ; LN03$K_LEN of resultant P-table
                      FF  182C                .BYTE   255                     ; Maximum unit number                   ;G:65
                    0000  182D                .WORD   ^A""    ; Two character mnemonic for QIO LN03
                      80  182F                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      86  1830                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1831                .LONG   HP$M_ALLOC              ; Constant
                      88  1835                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1836                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1838                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  1839                .BYTE   8                       ; 8 of data field
                      81  183A                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
```

```
                        183B   308 LP04:   $DS_LP04
                        183B               .SAVE   LOCAL_BLOCK
                        183B               .PSECT  $ABS$,ABS
              00000032  0050               .=HP$A_DEPENDENT
                        0032               .IIF    NB,LP04$K_LEN,  LP04$K_LEN:
                    0000183B               .RESTORE
        34 30 50 4C 00' 183B               .ASCIC  "LP04"  ; LP04 name
                    04  183B
                    32  1840               .BYTE   LP04$K_LEN              ; LP04$K_LEN of resultant P-table
                    01  1841               .BYTE   1               ; Maximum unit number                   ;G:65
                  0000  1842               .WORD   ^A""    ; Two character mnemonic for QIO LP04
                    80  1844               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  1845               .Byte   Pd$_Name        ; Directive op-code
                    1B  1846               .Byte   Ptd$M_EndDevice         ; Enforced LP codes
              50 4C 00' 1847               .Ascic  "LP"    ; Enforced device name prefix
                    02  1847
                    86  184A               .BYTE   Pd$_Literal     ; Indicate literal
              00000001  184B               .LONG   HP$M_ALLOC              ; Constant
                    88  184F               .BYTE   Pd$_Store       ; Indicate store operation
                  000A  1850               .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  1852               .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  1853               .BYTE   8               ; 8 of data field
                    81  1854               .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        1855   309 LP05:   $DS_LP05
                        1855               .SAVE   LOCAL_BLOCK
                        1855               .PSECT  $ABS$,ABS
              00000032  0050               .=HP$A_DEPENDENT
                        0032               .IIF    NB,LP05$K_LEN,  LP05$K_LEN:
                    00001855               .RESTORE
        35 30 50 4C 00' 1855               .ASCIC  "LP05"  ; LP05 name
                    04  1855
                    32  185A               .BYTE   LP05$K_LEN              ; LP05$K_LEN of resultant P-table
                    01  185B               .BYTE   1               ; Maximum unit number                   ;G:65
                  0000  185C               .WORD   ^A""    ; Two character mnemonic for QIO LP05
                    80  185E               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  185F               .Byte   Pd$_Name        ; Directive op-code
                    1B  1860               .Byte   Ptd$M_EndDevice         ; Enforced LP codes
              50 4C 00' 1861               .Ascic  "LP"    ; Enforced device name prefix
                    02  1861
                    86  1864               .BYTE   Pd$_Literal     ; Indicate literal
              00000001  1865               .LONG   HP$M_ALLOC              ; Constant
                    88  1869               .BYTE   Pd$_Store       ; Indicate store operation
                  000A  186A               .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  186C               .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  186D               .BYTE   8               ; 8 of data field
                    81  186E               .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        186F   310 LP06:   $DS_LP06
                        186F               .SAVE   LOCAL_BLOCK
                        186F               .PSECT  $ABS$,ABS
              00000032  0050               .=HP$A_DEPENDENT
                        0032               .IIF    NB,LP06$K_LEN,  LP06$K_LEN:
                    0000186F               .RESTORE
        36 30 50 4C 00' 186F               .ASCIC  "LP06"  ; LP06 name
                    04  186F
                    32  1874               .BYTE   LP06$K_LEN              ; LP06$K_LEN of resultant P-table
                    01  1875               .BYTE   1               ; Maximum unit number                   ;G:65
                  0000  1876               .WORD   ^A""    ; Two character mnemonic for QIO LP06
```

```
                        80  1878                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                        8D  1879                .Byte   Pd$_Name         ; Directive op-code
                        1B  187A                .Byte   Ptd$M_EndDevice          ; Enforced LP codes
                  50 4C 00' 187B                .Ascic  "LP"     ; Enforced device name prefix
                        02  187B
                        86  187E                .BYTE   Pd$_Literal      ; Indicate literal
                  00000001  187F                .LONG   HP$M_ALLOC               ; Constant
                        88  1883                .BYTE   Pd$_Store        ; Indicate store operation
                      000A  1884                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                        00  1886                .BYTE   0                ; 0 HP$B_FLAGS to data
                        08  1887                .BYTE   8                ; 8 of data field
                        81  1888                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                            1889    311 LP07:   $DS_LP07                ;                [15]
                            1889                .SAVE   LOCAL_BLOCK
                            1889                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,LP07$K_LEN,  LP07$K_LEN:
                            1889                .RESTORE
               37 30 50 4C 00' 1889             .ASCIC  "LP07"   ; LP07 name
                        04  1889
                        32  188E                .BYTE   LP07$K_LEN               ; LP07$K_LEN of resultant P-table
                        01  188F                .BYTE   1                ; Maximum unit number                          ;G:65
                      0000  1890                .WORD   ^A""     ; Two character mnemonic for QIO LP07
                        80  1892                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                        8D  1893                .Byte   Pd$_Name         ; Directive op-code
                        1B  1894                .Byte   Ptd$M_EndDevice          ; Enforced LP codes
                  50 4C 00' 1895                .Ascic  "LP"     ; Enforced device name prefix
                        02  1895
                        86  1898                .BYTE   Pd$_Literal      ; Indicate literal
                  00000001  1899                .LONG   HP$M_ALLOC               ; Constant
                        88  189D                .BYTE   Pd$_Store        ; Indicate store operation
                      000A  189E                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                        00  18A0                .BYTE   0                ; 0 HP$B_FLAGS to data
                        08  18A1                .BYTE   8                ; 8 of data field
                        81  18A2                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                            18A3    312 LP11:   $DS_LP11
                            18A3                .SAVE   LOCAL_BLOCK
                            18A3                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,LP11$L_CSR,  LP11$L_CSR:
                  00000036  0032                .IIF    NB,,BLKL,       .BLKL   1
                            0036                .IIF    NB,LP11$B_BR,   LP11$B_BR:
                  00000037  0036                .IIF    NB,,BLKB,       .BLKB   1
                            0037                .IIF    NB,LP11$K_LEN,  LP11$K_LEN:
                        000018A3                .RESTORE
            31 31 50 4C 00' 18A3                .ASCIC  "LP11"   ; LP11 name
                        04  18A3
                        37  18A8                .BYTE   LP11$K_LEN               ; LP11$K_LEN of resultant P-table
                        00  18A9                .BYTE   0                ; Maximum unit number                          ;G:65
                      504C  18AA                .WORD   ^A"LP"   ; Two character mnemonic for QIO LP11 LP
                        80  18AC                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                        8D  18AD                .Byte   Pd$_Name         ; Directive op-code
                        02  18AE                .Byte   Ptd$M_Controller                 ; Enforced LP codes
                  50 4C 00' 18AF                .Ascic  "LP"     ; Enforced device name prefix
                        02  18AF
                        83  18B2                .BYTE   Pd$_Octal        ; Indicate scan octal
               52 53 43 00' 18B3                .ASCIC  "CSR"    ; CSR string
```

```
                    03  18B3
          0003FFFE 0003E000  18B7                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                    88  18BF                .BYTE   Pd$_Store       ; Indicate store operation
                  0032  18C0                .WORD   LP11$L_CSR              ; Byte LP11$L_CSR to data
                    00  18C2                .BYTE   0               ; 0 LP11$L_CSR to data
                    20  18C3                .BYTE   32              ; 32 of data field
                    88  18C4                .BYTE   Pd$_Store       ; Indicate store operation
                  0018  18C5                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  18C7                .BYTE   0               ; 0 HP$A_DEVICE to data
                    12  18C8                .BYTE   18              ; 18 of data field
                    83  18C9                .BYTE   Pd$_Octal       ; Indicate scan octal
       52 4F 54 43 45 56 00'  18CA          .ASCIC  "VECTOR"        ; VECTOR string
                    06  18CA
          000001FE 00000002  18D1           .LONG   ^O2,^O776       ; 2 , 776 limits on input
                    88  18D9                .BYTE   Pd$_Store       ; Indicate store operation
                  0024  18DA                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  18DC                .BYTE   0               ; 0 HP$W_VECTOR to data
                    09  18DD                .BYTE   9               ; 9 of data field
                    82  18DE                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00'  18DF            .ASCIC  "BR"    ; BR string
                    02  18DF
          00000007 00000004  18E2           .LONG   4,7     ; 4 , 7 limits on input
                    88  18EA                .BYTE   Pd$_Store       ; Indicate store operation
                  0036  18EB                .WORD   LP11$B_BR               ; Byte LP11$B_BR to data
                    00  18ED                .BYTE   0               ; 0 LP11$B_BR to data
                    08  18EE                .BYTE   8               ; 8 of data field
                    81  18EF                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        18F0    313 LP14:   $DS_LP14
                        18F0                .SAVE   LOCAL_BLOCK
                        18F0                .PSECT  $ABS$,ABS
                00000032  0050              .=HP$A_DEPENDENT
                        0032                .IIF    NB,LP14$K_LEN,  LP14$K_LEN:
                    000018F0                .RESTORE
       34 31 50 4C 00'  18F0                .ASCIC  "LP14"  ; LP14 name
                    04  18F0
                    32  18F5                .BYTE   LP14$K_LEN              ; LP14$K_LEN of resultant P-table
                    01  18F6                .BYTE   1               ; Maximum unit number                           ;G:65
                  0000  18F7                .WORD   ^A""    ; Two character mnemonic for QIO LP14
                    80  18F9                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  18FA                .Byte   Pd$_Name        ; Directive op-code
                    1B  18FB                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                50 4C 00'  18FC             .Ascic  "LP"    ; Enforced device name prefix
                    02  18FC
                    86  18FF                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1900              .LONG   HP$M_ALLOC              ; Constant
                    88  1904                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  1905                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  1907                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  1908                .BYTE   8               ; 8 of data field
                    81  1909                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        190A    314 LP25:   $DS_LP25                        ;                       [01]
                        190A                .SAVE   LOCAL_BLOCK
                        190A                .PSECT  $ABS$,ABS
                00000032  0050              .=HP$A_DEPENDENT
                        0032                .IIF    NB,LP25$K_LEN,  LP25$K_LEN:
                    0000190A                .RESTORE
       35 32 50 4C 00'  190A                .ASCIC  "LP25"  ; LP25 name
```

```
                      04  190A
                      32  190F                .BYTE   LP25$K_LEN              ; LP25$K_LEN of resultant P-table
                      01  1910                .BYTE   1                       ; Maximum unit number                   ;G:65
                    0000  1911                .WORD   ^A""     ; Two character mnemonic for QIO LP25
                      80  1913                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  1914                .Byte   Pd$_Name                ; Directive op-code
                      1B  1915                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                 50 4C 00' 1916                .Ascic  "LP"    ; Enforced device name prefix
                      02  1916
                      86  1919                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  191A                .LONG   HP$M_ALLOC              ; Constant
                      88  191E                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  191F                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1921                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  1922                .BYTE   8                       ; 8 of data field
                      81  1923                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          1924    315 LP26:   $DS_LP26                        ;                       [15]
                          1924                .SAVE   LOCAL_BLOCK
                          1924                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LP26$K_LEN,  LP26$K_LEN:
                      0000192A                .RESTORE
              36 32 50 4C 00' 1924                .ASCIC  "LP26"  ; LP26 name
                      04  1924
                      32  1929                .BYTE   LP26$K_LEN              ; LP26$K_LEN of resultant P-table
                      01  192A                .BYTE   1                       ; Maximum unit number                   ;G:65
                    0000  192B                .WORD   ^A""     ; Two character mnemonic for QIO LP26
                      80  192D                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  192E                .Byte   Pd$_Name                ; Directive op-code
                      1B  192F                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                 50 4C 00' 1930                .Ascic  "LP"    ; Enforced device name prefix
                      02  1930
                      86  1933                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  1934                .LONG   HP$M_ALLOC              ; Constant
                      88  1938                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  1939                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  193B                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  193C                .BYTE   8                       ; 8 of data field
                      81  193D                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          193E    316 LP27:   $DS_LP27                        ;                       [15]
                          193E                .SAVE   LOCAL_BLOCK
                          193E                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LP27$K_LEN,  LP27$K_LEN:
                      0000193E                .RESTORE
              37 32 50 4C 00' 193E                .ASCIC  "LP27"  ; LP27 name
                      04  193E
                      32  1943                .BYTE   LP27$K_LEN              ; LP27$K_LEN of resultant P-table
                      01  1944                .BYTE   1                       ; Maximum unit number                   ;G:65
                    0000  1945                .WORD   ^A""     ; Two character mnemonic for QIO LP27
                      80  1947                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  1948                .Byte   Pd$_Name                ; Directive op-code
                      1B  1949                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                 50 4C 00' 194A                .Ascic  "LP"    ; Enforced device name prefix
                      02  194A
                      86  194D                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  194E                .LONG   HP$M_ALLOC              ; Constant
```

```
                      88  1952                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1953                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1955                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1956                .BYTE   8               ; 8 of data field
                      81  1957                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1958    317 LPA11K: $DS_LPA11K
                          1958                .SAVE   LOCAL_BLOCK
                          1958                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LPA11K$L_CSR,        LPA11K$L_CSR:
                 00000036 0032                .IIF    NB,..BLKL,      .BLKL   1
                          0036                .IIF    NB,LPA11K$B_BR, LPA11K$B_BR:
                 00000037 0036                .IIF    NB,..BLKB,      .BLKB   1
                          0037                .IIF    NB,LPA11K$K_LEN,        LPA11K$K_LEN:
                      00001958                .RESTORE
           4B 31 31 41 50 4C 00' 1958         .ASCIC  "LPA11K"        ; LPA11K name
                      06  1958
                      37  195F                .BYTE   LPA11K$K_LEN            ; LPA11K$K_LEN of resultant P-table
                      01  1960                .BYTE   1               ; Maximum unit number                   ;G:65
                    414C  1961                .WORD   ^A"LA"  ; Two character mnemonic for QIO LPA11K LA
                      80  1963                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1964                .Byte   Pd$_Name        ; Directive op-code
                      03  1965                .Byte   Ptd$M_Device            ; Enforced LA codes
                 41 4C 00' 1966                .Ascic  "LA"    ; Enforced device name prefix
                      02  1966
                      86  1969                .BYTE   Pd$_Literal     ; Indicate literal
                 00000001 196A                .LONG   HP$M_ALLOC              ; Constant
                      88  196E                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  196F                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1971                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1972                .BYTE   8               ; 8 of data field
                      83  1973                .BYTE   Pd$_Octal       ; Indicate scan octal
              52 53 43 00' 1974                .ASCIC  "CSR"   ; CSR string
                      03  1974
           0003FFFE 0003E000  1978             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  1980                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1981                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  1983                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  1984                .BYTE   18              ; 18 of data field
                      88  1985                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1986                .WORD   LPA11K$L_CSR            ; Byte LPA11K$L_CSR to data
                      00  1988                .BYTE   0               ; 0 LPA11K$L_CSR to data
                      20  1989                .BYTE   32              ; 32 of data field
                      83  198A                .BYTE   Pd$_Octal       ; Indicate scan octal
        52 4F 54 43 45 56 00' 198B            .ASCIC  "VECTOR"        ; VECTOR string
                      06  198B
           000001FE 00000002  1992             .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  199A                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  199B                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  199D                .BYTE   0               ; 0 HP$W_VECTOR to data
                      09  199E                .BYTE   9               ; 9 of data field
                      82  199F                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 19A0                .ASCIC  "BR"    ; BR string
                      02  19A0
           00000007 00000004  19A3             .LONG   4,7     ; 4 , 7 limits on input
                      88  19AB                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  19AC                .WORD   LPA11K$B_BR             ; Byte LPA11K$B_BR to data
```

```
                      00  19AE                .BYTE   0                       ; 0 LPA11K$B_BR to data
                      08  19AF                .BYTE   8                       ; 8 of data field
                      81  19B0                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          19B1    318 LUA11:  $DS_LUA11                               ;                   [33]
                          19B1                .SAVE   LOCAL_BLOCK
                          19B1                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LUA11$B_BR,  LUA11$B_BR:
                 00000033 0032                .IIF    NB,,BLKB,       .BLKB   1
                          0033                .IIF    NB,LUA11$L_CSR, LUA11$L_CSR:
                 00000037 0033                .IIF    NB,,BLKL,       .BLKL   1
                          0037                .IIF    NB,LUA11$K_LEN, LUA11$K_LEN:
                          000019B1            .RESTORE
           31 31 41 55 4C 00' 19B1            .ASCIC  "LUA11" ; LUA11 name
                      05  19B1
                      37  19B7                .BYTE   LUA11$K_LEN             ; LUA11$K_LEN of resultant P-table
                      08  19B8                .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  19B9                .WORD   ^A""    ; Two character mnemonic for QIO LUA11
                      80  19BB                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  19BC                .Byte   Pd$_Name                ; Directive op-code
                      03  19BD                .Byte   PTD$M_DEVICE            ; Enforced XE codes
                 45 58 00' 19BE                .Ascic  "XE"    ; Enforced device name prefix
                      02  19BE
                      83  19C1                .BYTE   Pd$_Octal               ; Indicate scan octal
                52 53 43 00' 19C2                .ASCIC  "CSR"   ; CSR string
                      03  19C2
            0003FFFE 0003E000 19C6                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  19CE                .BYTE   Pd$_Store               ; Indicate store operation
                    0033  19CF                .WORD   LUA11$L_CSR             ; Byte LUA11$L_CSR to data
                      00  19D1                .BYTE   0                       ; 0 LUA11$L_CSR to data
                      20  19D2                .BYTE   32                      ; 32 of data field
                      88  19D3                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  19D4                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  19D6                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  19D7                .BYTE   18                      ; 18 of data field
                      83  19D8                .BYTE   Pd$_Octal               ; Indicate scan octal
        52 4F 54 43 45 56 00' 19D9            .ASCIC  "VECTOR"                ; VECTOR string
                      06  19D9
            000001FE 00000002 19E0            .LONG   ^O2,^O776               ; 2 , 776 limits on input
                      88  19E8                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  19E9                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  19EB                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  19EC                .BYTE   9                       ; 9 of data field
                      82  19ED                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                52 42 00' 19EE                .ASCIC  "BR"    ; BR string
                      02  19EE
            00000007 00000004 19F1            .LONG   4,7     ; 4 , 7 limits on input
                      88  19F9                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  19FA                .WORD   LUA11$B_BR              ; Byte LUA11$B_BR to data
                      00  19FC                .BYTE   0                       ; 0 LUA11$B_BR to data
                      08  19FD                .BYTE   8                       ; 8 of data field
                      61  19FE                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          19FF    319 NBIA:   $DS_NBIA
                          19FF                .SAVE   LOCAL_BLOCK
                          19FF                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_NBIA_COMMON,    HP$B_NBIA_COMMON:
```

```
                        00000033  0032                .IIF    NB,,BLKB,       .BLKB   1
                                  0033                .IIF    NB,HP$L_NBIA_CSR,       HP$L_NBIA_CSR:
                        00000037  0033                .IIF    NB,,BLKL,       .BLKL   1
                                  0037                .IIF    NB,HP$B_BR,     HP$B_BR:
                        00000038  0037                .IIF    NB,,BLKB,       .BLKB   1
                                  0038                .IIF    NB,HP$K_NBIA_LEN,       HP$K_NBIA_LEN:
                           000019FF                 .RESTORE
                     41 49 42 4E 00' 19FF           .ASCIC  "NBIA"  ; NBIA name
                              04  19FF
                              38  1A04              .BYTE   HP$K_NBIA_LEN           ; HP$K_NBIA_LEN of resultant P-table
                              02  1A05              .BYTE   2                       ; Maximum unit number                   ;G:65
                            0000  1A06              .WORD   ^A""    ; Two character mnemonic for QIO NBIA
                              80  1A08              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              8D  1A09              .Byte   Pd$_Name        ; Directive op-code
                              01  1A0A              .Byte   PTD$M_UNIT              ; Enforced NBIA codes
                     41 49 42 4E 00' 1A0B           .Ascic  "NBIA"  ; Enforced device name prefix
                              04  1A0B
                              86  1A10              .BYTE   Pd$_Literal     ; Indicate literal
                        60080000  1A11              .LONG   ^X60080000              ; Constant
                              88  1A15              .BYTE   Pd$_Store       ; Indicate store operation
                            0018  1A16              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              00  1A18              .BYTE   0                       ; 0 HP$A_DEVICE to data
                              20  1A19              .BYTE   32                      ; 32 of data field
                              82  1A1A              .BYTE   Pd$_Decimal     ; Indicate scan decimal
41 44 41 20 4C 41 47 49 47 4F 4C 00' 1A1B           .ASCIC  "LOGIGAL ADAPTER # "    ; LOGIGAL ADAPTER #  string
            20 23 20 52 45 54 50  1A27
                              12  1A1B
                        00000001 00000000  1A2E     .LONG   0,1     ; 0 , 1 limits on input
                              88  1A36              .BYTE   Pd$_Store       ; Indicate store operation
                            0018  1A37              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              1A  1A39              .BYTE   26                      ; 26 HP$A_DEVICE to data
                              01  1A3A              .BYTE   1                       ; 1 of data field
                              8C  1A3B              .BYTE   Pd$_Case        ; Indicate case operation
                              02  1A3C              .BYTE   $$_$            ; Count of pairs
                        00000120 00000000  1A3D     .LONG   0,^X120         ; Store the pair
                        00000130 00000001  1A45     .LONG   1,^X130         ; Store the pair
                              88  1A4D              .BYTE   Pd$_Store       ; Indicate store operation
                            0024  1A4E              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                              00  1A50              .BYTE   0                       ; 0 HP$W_VECTOR to data
                              10  1A51              .BYTE   16                      ; 16 of data field
                              87  1A52              .BYTE   Pd$_Fetch       ; Indicate fetch operation
                            0018  1A53              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              1A  1A55              .BYTE   26                      ; 26 HP$A_DEVICE to data
                              01  1A56              .BYTE   1                       ; 1 of data field
                              8C  1A57              .BYTE   Pd$_Case        ; Indicate case operation
                              02  1A58              .BYTE   $$_$            ; Count of pairs
                        60000000 00000000  1A59     .LONG   0,^X60000000            ; Store the pair
                        64000000 00000001  1A61     .LONG   1,^X64000000            ; Store the pair
                              88  1A69              .BYTE   Pd$_Store       ; Indicate store operation
                            001C  1A6A              .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                              00  1A6C              .BYTE   0                       ; 0 HP$A_DVA to data
                              20  1A6D              .BYTE   32                      ; 32 of data field
                              81  1A6E              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                  1A6F  320 NBIB:   $DS_NBIB
                                  1A6F              .SAVE   LOCAL_BLOCK
                                  1A6F              .PSECT  $ABS$,ABS
                        00000032  0050              .=HP$A_DEPENDENT
```

```
                          0032                .IIF    NB,HP$B_NBIB_COMMON,    HP$B_NBIB_COMMON:
                 00000033 0032                .IIF    NB,,BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$B_BI_NODE,        HP$B_BI_NODE:
                 00000034 0033                .IIF    NB,,BLKB,       .BLKB   1
                          0034                .IIF    NB,HP$L_NBIB_CSR,       HP$L_NBIB_CSR:
                 00000038 0034                .IIF    NB,,BLKL,       .BLKL   1
                          0038                .IIF    NB,HP$B_BI_NUM, HP$B_BI_NUM:
                 00000039 0038                .IIF    NB,,BLKB,       .BLKB   1
                          0039                .IIF    NB,HP$K_NBIB_LEN,       HP$K_NBIB_LEN:
                   00001A6F                  .RESTORE
            42 49 42 4E 00' 1A6F              .ASCIC  "NBIB"  ; NBIB name
                       04   1A6F
                       39   1A74              .BYTE   HP$K_NBIB_LEN           ; HP$K_NBIB_LEN of resultant P-table
                       06   1A75              .BYTE   6                       ; Maximum unit number                   ;G:65
                     0000   1A76              .WORD   ^A""    ; Two character mnemonic for QIO NBIB
                       80   1A78              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D   1A79              .Byte   Pd$_Name        ; Directive op-code
                       01   1A7A              .Byte   PTD$M_UNIT              ; Enforced NBIB codes
            42 49 42 4E 00' 1A7B              .Ascic  "NBIB"  ; Enforced device name prefix
                       04   1A7B
                       82   1A80              .BYTE   Pd$_Decimal     ; Indicate scan decimal
         20 23 20 49 42 00' 1A81              .ASCIC  "BI # " ; BI #  string
                       05   1A81
         00000001 00000000  1A87              .LONG   0,1     ; 0 , 1 limits on input
                       88   1A8F              .BYTE   Pd$_Store       ; Indicate store operation
                     0038   1A90              .WORD   HP$B_BI_NUM             ; Byte HP$B_BI_NUM to data
                       00   1A92              .BYTE   0               ; 0 HP$B_BI_NUM to data
                       02   1A93              .BYTE   2               ; 2 of data field
                       88   1A94              .BYTE   Pd$_Store       ; Indicate store operation
                     0018   1A95              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       19   1A97              .BYTE   25              ; 25 HP$A_DEVICE to data
                       01   1A98              .BYTE   1               ; 1 of data field
                       87   1A99              .BYTE   Pd$_Fetch       ; Indicate fetch operation
                     0018   1A9A              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00   1A9C              .BYTE   0               ; 0 HP$A_DEVICE to data
                       20   1A9D              .BYTE   32              ; 32 of data field
                       88   1A9E              .BYTE   Pd$_Store       ; Indicate store operation
                     001C   1A9F              .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       00   1AA1              .BYTE   0               ; 0 HP$A_DVA to data
                       20   1AA2              .BYTE   32              ; 32 of data field
                       84   1AA3              .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 1AA4     .ASCIC  "BI Node Number (HEX) " ; BI Node Number (HEX)  string
      20 29 58 45 48 28 20 72 65 62  1AB0
                       15   1AA4
                 0000000F 00000000  1ABA      .LONG   ^X0,^XF ; 0 , F limits on input
                       88   1AC2              .BYTE   Pd$_Store       ; Indicate store operation
                     0033   1AC3              .WORD   HP$B_BI_NODE            ; Byte HP$B_BI_NODE to data
                       00   1AC5              .BYTE   0               ; 0 HP$B_BI_NODE to data
                       08   1AC6              .BYTE   8               ; 8 of data field
                       87   1AC7              .BYTE   Pd$_Fetch       ; Indicate fetch operation
                     0033   1AC8              .WORD   HP$B_BI_NODE            ; Byte HP$B_BI_NODE to data
                       00   1ACA              .BYTE   0               ; 0 HP$B_BI_NODE to data
                       08   1ACB              .BYTE   8               ; 8 of data field
                       88   1ACC              .BYTE   Pd$_Store       ; Indicate store operation
                     0018   1ACD              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       0D   1ACF              .BYTE   13              ; 13 HP$A_DEVICE to data
                       04   1AD0              .BYTE   4               ; 4 of data field
```

```
                      81  1AD1                        .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1AD2   321 MA780:   $DS_MA780
                          1AD2               .SAVE   LOCAL_BLOCK
                          1AD2               .PSECT  $ABS$,ABS
                 00000032 0050               .=HP$A_DEPENDENT
                          0032               .IIF    NB,MA780$B_TR,  MA780$B_TR:
                 00000033 0032               .IIF    NB,..BLKB,      .BLKB   1
                          0033               .IIF    NB,MA780$B_BR,  MA780$B_BR:
                 00000034 0033               .IIF    NB,..BLKB,      .BLKB   1
                          0034               .IIF    NB,MA780$B_MPM, MA780$B_MPM:
                 00000035 0034               .IIF    NB,..BLKB,      .BLKB   1
                          0035               .IIF    NB,MA780$B_PORT,        MA780$B_PORT:
                 00000036 0035               .IIF    NB,..BLKB,      .BLKB   1
                          0036               .IIF    NB,MA780$K_LEN, MA780$K_LEN:
                      00001AD2               .RESTORE
           30 38 37 41 4D 00' 1AD2           .ASCIC  "MA780" ; MA780 name
                      05  1AD2
                      36  1AD8               .BYTE   MA780$K_LEN             ; MA780$K_LEN of resultant P-table
                      08  1AD9               .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  1ADA               .WORD   ^A""    ; Two character mnemonic for QIO MA780
                      80  1ADC               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1ADD               .Byte   Pd$_Name        ; Directive op-code
                      01  1ADE               .Byte   Ptd$M_Unit              ; Enforced MA codes
                 41 4D 00' 1ADF               .Ascic  "MA"    ; Enforced device name prefix
                      02  1ADF
                      82  1AE2               .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 54 00' 1AE3               .ASCIC  "TR"    ; TR string
                      02  1AE3
                00000001  1AE6               .LONG   1,15    ; 1 , 15 limits on input
                      88  1AEE               .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1AEF               .WORD   MA780$B_TR              ; Byte MA780$B_TR to data
                      00  1AF1               .BYTE   0               ; 0 MA780$B_TR to data
                      08  1AF2               .BYTE   8               ; 8 of data field
                      88  1AF3               .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1AF4               .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      0D  1AF6               .BYTE   13              ; 13 HP$A_DEVICE to data
                      04  1AF7               .BYTE   4               ; 4 of data field
                      88  1AF8               .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1AF9               .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      02  1AFB               .BYTE   2               ; 2 HP$W_VECTOR to data
                      04  1AFC               .BYTE   4               ; 4 of data field
                      82  1AFD               .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 1AFE               .ASCIC  "BR"    ; BR string
                      02  1AFE
                00000004  1B01               .LONG   4,7     ; 4 , 7 limits on input
                      88  1B09               .BYTE   Pd$_Store       ; Indicate store operation
                    0033  1B0A               .WORD   MA780$B_BR              ; Byte MA780$B_BR to data
                      00  1B0C               .BYTE   0               ; 0 MA780$B_BR to data
                      08  1B0D               .BYTE   8               ; 8 of data field
                      88  1B0E               .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1B0F               .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      06  1B11               .BYTE   6               ; 6 HP$W_VECTOR to data
                      02  1B12               .BYTE   2               ; 2 of data field
                      86  1B13               .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1B14               .LONG   1               ; Constant
                      88  1B18               .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1B19               .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
```

```
                      08  1B1B                .BYTE   8               ; 8 HP$W_VECTOR to data
                      01  1B1C                .BYTE   1               ; 1 of data field
                      86  1B1D                .BYTE   Pd$_Literal     ; Indicate literal
                00000006  1B1E                .LONG   6               ; Constant
                      88  1B22                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1B23                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      1C  1B25                .BYTE   28              ; 28 HP$A_DEVICE to data
                      04  1B26                .BYTE   4               ; 4 of data field
                      82  1B27                .BYTE   Pd$_Decimal     ; Indicate scan decimal
             4D 50 4D 00' 1B28                .ASCIC  "MPM"   ; MPM string
                      03  1B28
            00000003 00000000  1B2C           .LONG   0,3     ; 0 . 3 limits on input
                      88  1B34                .BYTE   Pd$_Store       ; Indicate store operation
                    0034  1B35                .WORD   MA780$B_MPM             ; Byte MA780$B_MPM to data
                      00  1B37                .BYTE   0               ; 0 MA780$B_MPM to data
                      08  1B38                .BYTE   8               ; 8 of data field
                      82  1B39                .BYTE   Pd$_Decimal     ; Indicate scan decimal
          54 52 4F 50 00' 1B3A                .ASCIC  "PORT"  ; PORT string
                      04  1B3A
            00000003 00000000  1B3F           .LONG   0,3     ; 0 . 3 limits on input
                      88  1B47                .BYTE   Pd$_Store       ; Indicate store operation
                    0035  1B48                .WORD   MA780$B_PORT            ; Byte MA780$B_PORT to data
                      00  1B4A                .BYTE   0               ; 0 MA780$B_PORT to data
                      08  1B4B                .BYTE   8               ; 8 of data field
                      81  1B4C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1B4D    322 MBE:    $DS_MBE
                          1B4D                .SAVE   LOCAL_BLOCK
                          1B4D                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,MBE$K_LEN,   MBE$K_LEN:
                      00001B4D                .RESTORE
             45 42 4D 00' 1B4D                .ASCIC  "MBE"   ; MBE name
                      03  1B4D
                      32  1B51                .BYTE   MBE$K_LEN               ; MBE$K_LEN of resultant P-table
                      08  1B52                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  1B53                .WORD   ^A""    ; Two character mnemonic for QIO MBE
                      80  1B55                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      87  1B56                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    000B  1B57                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                      00  1B59                .BYTE   0               ; 0 HP$B_DRIVE to data
                      08  1B5A                .BYTE   8               ; 8 of data field
                      88  1B5B                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1B5C                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      07  1B5E                .BYTE   7               ; 7 HP$A_DEVICE to data
                      03  1B5F                .BYTE   3               ; 3 of data field
                      81  1B60                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1B61    323 ML11:   $DS_ML11                        ;               [01]
                          1B61                .SAVE   LOCAL_BLOCK
                          1B61                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,ML11$B_ARRAY,        ML11$B_ARRAY:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,ML11$B_CHIP, ML11$B_CHIP:
                00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,ML11$K_LEN,  ML11$K_LEN:
                      00001B61                .RESTORE
          31 31 4C 4D 00' 1B61                .ASCIC  "ML11"  ; ML11 name
```

```
                          04  1B61
                          34  1B66                .BYTE   ML11$K_LEN              ; ML11$K_LEN of resultant P-table
                          08  1B67                .BYTE   8                       ; Maximum unit number                   ;G:65
                        4D45  1B68                .WORD   ^A"EM"  ; Two character mnemonic for QIO ML11 EM
                          80  1B6A                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                          8D  1B6B                .Byte   Pd$_Name                ; Directive op-code
                          03  1B6C                .Byte   Ptd$M_Device            ; Enforced EM codes
                 4D 45 00'    1B6D                .Ascic  "EM"    ; Enforced device name prefix
                          02  1B6D
                          86  1B70                .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  1B71                .LONG   HP$M_ALLOC              ; Constant
                          88  1B75                .BYTE   Pd$_Store       ; Indicate store operation
                        000A  1B76                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  1B78                .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  1B79                .BYTE   8               ; 8 of data field
                          87  1B7A                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                        000B  1B7B                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                          00  1B7D                .BYTE   0               ; 0 HP$B_DRIVE to data
                          08  1B7E                .BYTE   8               ; 8 of data field
                          88  1B7F                .BYTE   Pd$_Store       ; Indicate store operation
                        0018  1B80                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          07  1B82                .BYTE   7               ; 7 HP$A_DEVICE to data
                          03  1B83                .BYTE   3               ; 3 of data field
                          82  1B84                .BYTE   Pd$_Decimal     ; Indicate scan decimal
61 20 66 6F 20 72 65 62 6D 75 4E 00'  1B85        .ASCIC  "Number of array boards"        ; Number of array boards string
   73 64 72 61 6F 62 20 79 61 72 72   1B91
                          16  1B85
                00000010 00000001  1B9C                .LONG   1,16    ; 1 , 16 limits on input
                          88  1BA4                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1BA5                .WORD   ML11$B_ARRAY            ; Byte ML11$B_ARRAY to data
                          00  1BA7                .BYTE   0               ; 0 ML11$B_ARRAY to data
                          08  1BA8                .BYTE   8               ; 8 of data field
                          85  1BA9                .BYTE   Pd$_String      ; Indicate scan and verify string
65 7A 69 73 20 70 69 68 43 00'  1BAA              .ASCIC  "Chip size"     ; Chip size string
                          09  1BAA
                 4B 36 31 00'  1BB4                .ASCIC  "16K"
                          03  1BB4
                 4B 34 36 00'  1BB8                .ASCIC  "64K"
                          03  1BB8
                          00  1BBC                .BYTE   0               ; Null length is end of valid 16K,64K
                          88  1BBD                .BYTE   Pd$_Store       ; Indicate store operation
                        0033  1BBE                .WORD   ML11$B_CHIP             ; Byte ML11$B_CHIP to data
                          00  1BC0                .BYTE   0               ; 0 ML11$B_CHIP to data
                          08  1BC1                .BYTE   8               ; 8 of data field
                          81  1BC2                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              1BC3    324 MS750:  $DS_MS750
                              1BC3                .SAVE   LOCAL_BLOCK
                              1BC3                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,MS750$K_LEN, MS750$K_LEN:
                          0000 1BC3               .RESTORE
             30 35 37 53 4D 00'  1BC3             .ASCIC  "MS750" ; MS750 name
                          05  1BC3
                          32  1BC9                .BYTE   MS750$K_LEN             ; MS750$K_LEN of resultant P-table
                          08  1BCA                .BYTE   8                       ; Maximum unit number                   ;G:65
                        0000  1BCB                .WORD   ^A""    ; Two character mnemonic for QIO MS750
                          80  1BCD                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
```

```
                          8D  1BCE                .Byte   Pd$_Name         ; Directive op-code
                          01  1BCF                .Byte   Ptd$M_Unit               ; Enforced MS codes
                    53 4D 00' 1BD0                .Ascic  "MS"    ; Enforced device name prefix
                          02  1BD0
                          81  1BD3                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                              1BD4    325 MS780:  $DS_MS780
                              1BD4                .SAVE   LOCAL_BLOCK
                              1BD4                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,MS780$B_TR,  MS780$B_TR:
                    00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                              0033                .IIF    NB,ms780$b_arrays,      ms780$b_arrays:
                    00000034  0033                .IIF    NB,.blkb,       .blkb   1
                              0034                .IIF    NB,MS780$K_LEN, MS780$K_LEN:
                          00001BD4                .RESTORE
                 30 38 37 53 4D 00' 1BD4          .ASCIC  "MS780" ; MS780 name
                          05  1BD4
                          34  1BDA                .BYTE   MS780$K_LEN              ; MS780$K_LEN of resultant P-table
                          08  1BDB                .BYTE   8                ; Maximum unit number                          ;G:65
                        0000  1BDC                .WORD   ^A""     ; Two character mnemonic for QIO MS780
                          80  1BDE                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                          8D  1BDF                .Byte   Pd$_Name         ; Directive op-code
                          01  1BE0                .Byte   Ptd$M_Unit               ; Enforced MS codes
                    53 4D 00' 1BE1                .Ascic  "MS"    ; Enforced device name prefix
                          02  1BE1
                          82  1BE4                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                       52 54 00' 1BE5             .ASCIC  "TR"     ; TR string
                          02  1BE5
                    0000000F 00000001  1BE8       .LONG   1,15     ; 1 , 15 limits on input
                          88  1BF0                .BYTE   Pd$_Store        ; Indicate store operation
                        0032  1BF1                .WORD   MS780$B_TR               ; Byte MS780$B_TR to data
                          00  1BF3                .BYTE   0                ; 0 MS780$B_TR to data
                          08  1BF4                .BYTE   8                ; 8 of data field
                          88  1BF5                .BYTE   Pd$_Store        ; Indicate store operation
                        0018  1BF6                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                          0D  1BF8                .BYTE   13               ; 13 HP$A_DEVICE to data
                          04  1BF9                .BYTE   4                ; 4 of data field
                          86  1BFA                .BYTE   Pd$_Literal      ; Indicate literal
                    00000006  1BFB                .LONG   6                ; Constant
                          88  1BFF                .BYTE   Pd$_Store        ; Indicate store operation
                        0018  1C00                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                          1C  1C02                .BYTE   28               ; 28 HP$A_DEVICE to data
                          04  1C03                .BYTE   4                ; 4 of data field
                          82  1C04                .BYTE   Pd$_Decimal      ; Indicate scan decimal
61 20 66 6F 20 72 65 62 6D 75 4E 00' 1C05         .ASCIC  "Number of arrays"       ; Number of arrays string
                    73 79 61 72 72  1C11
                          10  1C05
                    00000010 00000000  1C16       .LONG   0,16     ; 0 , 16 limits on input
                          88  1C1E                .BYTE   Pd$_Store        ; Indicate store operation
                        0033  1C1F                .WORD   ms780$b_arrays           ; Byte ms780$b_arrays to data
                          00  1C21                .BYTE   0                ; 0 ms780$b_arrays to data
                          08  1C22                .BYTE   8                ; 8 of data field
                          81  1C23                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                              1C24    326 PCL11:  $DS_PCL11
                              1C24                .SAVE   LOCAL_BLOCK
                              1C24                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
```

```
                                0032          .IIF    NB,PCL11$L_CSR, PCL11$L_CSR:
                        00000036 0032          .IIF    NB,.BLKL,       .BLKL   1
                                0036          .IIF    NB,PCL11$B_BR,  PCL11$B_BR:
                        00000037 0036          .IIF    NB,.BLKB,       .BLKB   1
                                0037          .IIF    NB,PCL11$K_LEN, PCL11$K_LEN:
                            00001C24          .RESTORE
                31 31 4C 43 50 00' 1C24       .ASCIC  "PCL11" ; PCL11 name
                              05 1C24
                              37 1C2A          .BYTE   PCL11$K_LEN             ; PCL11$K_LEN of resultant P-table
                              20 1C2B          .BYTE   32              ; Maximum unit number                    ;G:65
                            0000 1C2C          .WORD   ^A""    ; Two character mnemonic for QIO PCL11
                              80 1C2E          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              83 1C2F          .BYTE   Pd$_Octal       ; Indicate scan octal
                     52 53 43 00' 1C30         .ASCIC  "CSR"   ; CSR string
                              03 1C30
                0003FFFE 0003E000 1C34         .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                              88 1C3C          .BYTE   Pd$_Store       ; Indicate store operation
                            0032 1C3D          .WORD   PCL11$L_CSR             ; Byte PCL11$L_CSR to data
                              00 1C3F          .BYTE   0               ; 0 PCL11$L_CSR to data
                              20 1C40          .BYTE   32              ; 32 of data field
                              88 1C41          .BYTE   Pd$_Store       ; Indicate store operation
                            0018 1C42          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              00 1C44          .BYTE   0               ; 0 HP$A_DEVICE to data
                              12 1C45          .BYTE   18              ; 18 of data field
                              83 1C46          .BYTE   Pd$_Octal       ; Indicate scan octal
             52 4F 54 43 45 56 00' 1C47        .ASCIC  "VECTOR"        ; VECTOR string
                              06 1C47
                000001FE 00000002 1C4E         .LONG   ^O2,^O776       ; 2 , 776 limits on input
                              88 1C56          .BYTE   Pd$_Store       ; Indicate store operation
                            0024 1C57          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                              00 1C59          .BYTE   0               ; 0 HP$W_VECTOR to data
                              09 1C5A          .BYTE   9               ; 9 of data field
                              82 1C5B          .BYTE   Pd$_Decimal     ; Indicate scan decimal
                        52 42 00' 1C5C         .ASCIC  "BR"    ; BR string
                              02 1C5C
                00000007 00000004 1C5F         .LONG   4,7     ; 4 , 7 limits on input
                              88 1C67          .BYTE   Pd$_Store       ; Indicate store operation
                            0036 1C68          .WORD   PCL11$B_BR              ; Byte PCL11$B_BR to data
                              00 1C6A          .BYTE   0               ; 0 PCL11$B_BR to data
                              08 1C6B          .BYTE   8               ; 8 of data field
                              81 1C6C          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                1C6D  327 PX780: $DS_PX780
                                1C6D          .SAVE   LOCAL_BLOCK
                                1C6D          .PSECT  $ABS$,ABS
                        00000032 0050          .=HP$A_DEPENDENT
                                0032          .IIF    NB,PX780$B_TR,  PX780$B_TR:
                        00000033 0032          .IIF    NB,.BLKB,       .BLKB   1
                                0033          .IIF    NB,PX780$B_BR,  PX780$B_BR:
                        00000034 0033          .IIF    NB,.BLKB,       .BLKB   1
                                0034          .IIF    NB,PX780$B_NODE,        PX780$B_NODE:
                        00000035 0034          .IIF    NB,.BLKB,       .BLKB   1
                                0035          .IIF    NB,PX780$K_LEN, PX780$K_LEN:
                            00001C6D          .RESTORE
                30 38 37 58 50 00' 1C6D       .ASCIC  "PX780" ; PX780 name
                              05 1C6D
                              35 1C73          .BYTE   PX780$K_LEN             ; PX780$K_LEN of resultant P-table
                              01 1C74          .BYTE   1               ; Maximum unit number                    ;G:65
```

```
                    5850  1C75                .WORD   ^A"PX"  ; Two character mnemonic for QIO PX780 PX
                      80  1C77                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1C78                .Byte   Pd$_Name        ; Directive op-code
                      03  1C79                .Byte   Ptd$M_Device            ; Enforced PX codes
                58 50 00' 1C7A                .Ascic  "PX"    ; Enforced device name prefix
                      02  1C7A
                      82  1C7D                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 54 00' 1C7E                .ASCIC  "TR"    ; TR string
                      02  1C7E
            0000000F 00000001  1C81           .LONG   1,15    ; 1 , 15 limits on input
                      88  1C89                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1C8A                .WORD   PX780$B_TR                  ; Byte PX780$B_TR to data
                      00  1C8C                .BYTE   0                   ; 0 PX780$B_TR to data
                      08  1C8D                .BYTE   8                   ; 8 of data field
                      88  1C8E                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1C8F                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      0D  1C91                .BYTE   13                  ; 13 HP$A_DEVICE to data
                      04  1C92                .BYTE   4                   ; 4 of data field
                      88  1C93                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1C94                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      02  1C96                .BYTE   2                   ; 2 HP$W_VECTOR to data
                      04  1C97                .BYTE   4                   ; 4 of data field
                      82  1C98                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 1C99                .ASCIC  "BR"    ; BR string
                      02  1C99
            00000007 00000004  1C9C           .LONG   4,7     ; 4 , 7 limits on input
                      88  1CA4                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  1CA5                .WORD   PX780$B_BR                  ; Byte PX780$B_BR to data
                      00  1CA7                .BYTE   0                   ; 0 PX780$B_BR to data
                      08  1CA8                .BYTE   8                   ; 8 of data field
                      88  1CA9                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1CAA                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      06  1CAC                .BYTE   6                   ; 6 HP$W_VECTOR to data
                      02  1CAD                .BYTE   2                   ; 2 of data field
                      82  1CAE                .BYTE   Pd$_Decimal     ; Indicate scan decimal
             65 64 6F 4E 00' 1CAF             .ASCIC  "Node"  ; Node string
                      04  1CAF
            000000FF 00000000  1CB4           .LONG   0,255   ; 0 , 255 limits on input
                      88  1CBC                .BYTE   Pd$_Store       ; Indicate store operation
                    0034  1CBD                .WORD   PX780$B_NODE                ; Byte PX780$B_NODE to data
                      00  1CBF                .BYTE   0                   ; 0 PX780$B_NODE to data
                      08  1CC0                .BYTE   8                   ; 8 of data field
                      86  1CC1                .BYTE   Pd$_Literal      ; Indicate literal
                00000006  1CC2                .LONG   6                   ; Constant
                      88  1CC6                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1CC7                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      1C  1CC9                .BYTE   28                  ; 28 HP$A_DEVICE to data
                      04  1CCA                .BYTE   4                   ; 4 of data field
                      86  1CCB                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  1CCC                .LONG   1                   ; Constant
                      88  1CD0                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1CD1                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      08  1CD3                .BYTE   8                   ; 8 HP$W_VECTOR to data
                      01  1CD4                .BYTE   1                   ; 1 of data field
                      81  1CD5                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1CD6    328 R80:    $DS_R80                             ;                   [02]
                          1CD6                .SAVE   LOCAL_BLOCK
```

```
                        1CD6                    .PSECT  $ABS$,ABS
                    00000032  0050              .=HP$A_DEPENDENT
                              0032              .IIF    NB,R80$K_LEN,   R80$K_LEN:
                          00001CD6              .RESTORE
                 30 38 52 00' 1CD6              .ASCIC  "R80"   ; R80 name
                              03  1CD6
                              32  1CDA          .BYTE   R80$K_LEN               ; R80$K_LEN of resultant P-table
                              08  1CDB          .BYTE   8               ; Maximum unit number                   ;G:65
                            0000  1CDC          .WORD   ^A""    ; Two character mnemonic for QIO R80
                              80  1CDE          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              8D  1CDF          .Byte   Pd$_Name        ; Directive op-code
                              1B  1CE0          .Byte   Ptd$M_EndDevice         ; Enforced DQ codes
                        51 44 00' 1CE1          .Ascic  "DQ"    ; Enforced device name prefix
                              02  1CE1
                              86  1CE4          .BYTE   Pd$_Literal     ; Indicate literal
                        00000001  1CE5          .LONG   HP$M_ALLOC              ; Constant
                              88  1CE9          .BYTE   Pd$_Store       ; Indicate store operation
                            000A  1CEA          .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                              00  1CEC          .BYTE   0               ; 0 HP$B_FLAGS to data
                              08  1CED          .BYTE   8               ; 8 of data field
                              81  1CEE          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                  1CEF  329 RA60:   $DS_RA60
                                  1CEF          .SAVE   LOCAL_BLOCK
                                  1CEF          .PSECT  $ABS$,ABS
                        00000032  0050          .=HP$A_DEPENDENT
                                  0032          .IIF    NB,HP$K_RA60_LEN,       HP$K_RA60_LEN:
                                  0032          .IIF    NB,RA60$K_LEN,  RA60$K_LEN:
                          00001CEF              .RESTORE
                 30 36 41 52 00' 1CEF           .ASCIC  "RA60"  ; RA60 name
                              04  1CEF
                              32  1CF4          .BYTE   RA60$K_LEN              ; RA60$K_LEN of resultant P-table
                              FF  1CF5          .BYTE   255             ; Maximum unit number                   ;G:65
                            0000  1CF6          .WORD   ^A""    ; Two character mnemonic for QIO RA60
                              80  1CF8          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              8D  1CF9          .Byte   Pd$_Name        ; Directive op-code
                              13  1CFA          .Byte   Ptd$M_UNIT!PTD$M_CONTROLLER!PTD$M_INHERIT_CON           ; Enforced D
                        4A 44 00' 1CFB          .Ascic  "DJ"    ; Enforced device name prefix
                              02  1CFB
                              86  1CFE          .BYTE   Pd$_Literal     ; Indicate literal
                        00000001  1CFF          .LONG   HP$M_ALLOC              ; Constant
                              88  1D03          .BYTE   Pd$_Store       ; Indicate store operation
                            000A  1D04          .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                              00  1D06          .BYTE   0               ; 0 HP$B_FLAGS to data
                              08  1D07          .BYTE   8               ; 8 of data field
                              81  1D08          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                  1D09  330 RA70:   $DS_RA70                        ;               [36]            ;G:5
                                  1D09          .SAVE   LOCAL_BLOCK
                                  1D09          .PSECT  $ABS$,ABS
                        00000032  0050          .=HP$A_DEPENDENT
                                  0032          .IIF    NB,HP$K_RA70_LEN,       HP$K_RA70_LEN:
                                  0032          .IIF    NB,RA70$K_LEN,  RA70$K_LEN:
                          00001D09              .RESTORE
                 30 37 41 52 00' 1D09           .ASCIC  "RA70"  ; RA70 name
                              04  1D09
                              32  1D0E          .BYTE   RA70$K_LEN              ; RA70$K_LEN of resultant P-table
                              FF  1D0F          .BYTE   255             ; Maximum unit number                   ;G:65
                            0000  1D10          .WORD   ^A""    ; Two character mnemonic for QIO RA70
```

```
                      80  1D12                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1D13                .Byte   Pd$_Name        ; Directive op-code
                      1B  1D14                .Byte   PTD$M_ENDDEVICE         ; Enforced DU codes
                   55 44 00' 1D15             .Ascic  "DU"    ; Enforced device name prefix
                      02  1D15
                      86  1D18                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1D19                .LONG   HP$M_ALLOC              ; Constant
                      88  1D1D                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1D1E                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1D20                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1D21                .BYTE   8               ; 8 of data field
                      81  1D22                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1D23    331 RA80:   $DS_RA80                ;               [02]
                          1D23                .SAVE   LOCAL_BLOCK
                          1D23                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RA80_LEN,       HP$K_RA80_LEN:
                          0032                .IIF    NB,RA80$K_LEN,  RA80$K_LEN:
                      00001D23                .RESTORE
             30 38 41 52 00' 1D23             .ASCIC  "RA80"  ; RA80 name
                      04  1D23
                      32  1D28                .BYTE   RA80$K_LEN              ; RA80$K_LEN of resultant P-table
                      FF  1D29                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  1D2A                .WORD   ^A""    ; Two character mnemonic for QIO RA80
                      80  1D2C                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1D2D                .Byte   Pd$_Name        ; Directive op-code
                      1B  1D2E                .Byte   PTD$M_ENDDEVICE         ; Enforced DU codes
                   55 44 00' 1D2F             .Ascic  "DU"    ; Enforced device name prefix
                      02  1D2F
                      86  1D32                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1D33                .LONG   HP$M_ALLOC              ; Constant
                      88  1D37                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1D38                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1D3A                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1D3B                .BYTE   8               ; 8 of data field
                      81  1D3C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1D3D    332 RA81:   $DS_RA81
                          1D3D                .SAVE   LOCAL_BLOCK
                          1D3D                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RA81_LEN,       HP$K_RA81_LEN:
                          0032                .IIF    NB,RA81$K_LEN,  RA81$K_LEN:
                      00001D3D                .RESTORE
             31 38 41 52 00' 1D3D             .ASCIC  "RA81"  ; RA81 name
                      04  1D3D
                      32  1D42                .BYTE   RA81$K_LEN              ; RA81$K_LEN of resultant P-table
                      FF  1D43                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  1D44                .WORD   ^A""    ; Two character mnemonic for QIO RA81
                      80  1D46                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1D47                .Byte   Pd$_Name        ; Directive op-code
                      1B  1D48                .Byte   PTD$M_ENDDEVICE         ; Enforced DU codes
                   55 44 00' 1D49             .Ascic  "DU"    ; Enforced device name prefix
                      02  1D49
                      86  1D4C                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1D4D                .LONG   HP$M_ALLOC              ; Constant
                      88  1D51                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1D52                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
```

```
                      00  1D54                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  1D55                .BYTE   8                   ; 8 of data field
                      81  1D56                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          1D57    333 RA82:    $DS_RA82                            ;               [33]
                          1D57                .SAVE   LOCAL_BLOCK
                          1D57                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RA82_LEN,       HP$K_RA82_LEN:
                          0032                .IIF    NB,RA82$K_LEN,  RA82$K_LEN:
                      0000 1D57                .RESTORE
               32 38 41 52 00' 1D57                .ASCIC  "RA82"  ; RA82 name
                      04  1D57
                      32  1D5C                .BYTE   RA82$K_LEN              ; RA82$K_LEN of resultant P-table
                      FF  1D5D                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  1D5E                .WORD   ^A""    ; Two character mnemonic for QIO RA82
                      80  1D60                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1D61                .Byte   Pd$_Name        ; Directive op-code
                      1B  1D62                .Byte   PTD$M_ENDDEVICE         ; Enforced DU codes
                55 44 00' 1D63                .Ascic  "DU"    ; Enforced device name prefix
                      02  1D63
                      86  1D66                .BYTE   Pd$_Literal     ; Indicate literal
                  00000001  1D67                .LONG   HP$M_ALLOC              ; Constant
                      88  1D6B                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1D6C                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1D6E                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  1D6F                .BYTE   8                   ; 8 of data field
                      81  1D70                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          1D71    334 RA90:    $DS_RA90                                                ;G:7
                          1D71                .SAVE   LOCAL_BLOCK
                          1D71                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RA90_LEN,       HP$K_RA90_LEN:
                          0032                .IIF    NB,RA90$K_LEN,  RA90$K_LEN:
                      0000 1D71                .RESTORE
               30 39 41 52 00' 1D71                .ASCIC  "RA90"  ; RA90 name
                      04  1D71
                      32  1D76                .BYTE   RA90$K_LEN              ; RA90$K_LEN of resultant P-table
                      FF  1D77                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  1D78                .WORD   ^A""    ; Two character mnemonic for QIO RA90
                      80  1D7A                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1D7B                .Byte   Pd$_Name        ; Directive op-code
                      1B  1D7C                .Byte   PTD$M_ENDDEVICE         ; Enforced DU codes
                55 44 00' 1D7D                .Ascic  "DU"    ; Enforced device name prefix
                      02  1D7D
                      86  1D80                .BYTE   Pd$_Literal     ; Indicate literal
                  00000001  1D81                .LONG   HP$M_ALLOC              ; Constant
                      88  1D85                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1D86                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1D88                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  1D89                .BYTE   8                   ; 8 of data field
                      81  1D8A                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          1D8B    335 RB730:   $DS_RB730                           ;               [02]
                          1D8B                .SAVE   LOCAL_BLOCK
                          1D8B                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_RB730_CSR,      HP$L_RB730_CSR:
                          0032                .IIF    NB,RB730$L_CSR, RB730$L_CSR:
```

```
                00000036  0032                .IIF    NB,..BLKL,        .BLKL   1
                          0036                .IIF    NB,HP$B_RB730_BR,       HP$B_RB730_BR:
                          0036                .IIF    NB,RB730$B_BR,  RB730$B_BR:
                00000037  0036                .IIF    NB,..BLKB,        .BLKB   1
                          0037                .IIF    NB,HP$K_RB730_LEN,      HP$K_RB730_LEN:
                          0037                .IIF    NB,RB730$K_LEN, RB730$K_LEN:
                   00001D8B                   .RESTORE
          30 33 37 42 52 00' 1D8B             .ASCIC  "RB730" ; RB730 name
                      05  1D8B
                      37  1D91                .BYTE   RB730$K_LEN             ; RB730$K_LEN of resultant P-table
                      00  1D92                .BYTE   0                       ; Maximum unit number                   ;G:65
                    5144  1D93                .WORD   ^A"DQ"  ; Two character mnemonic for QIO RB730 DQ
                      80  1D95                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1D96                .Byte   Pd$_Name        ; Directive op-code
                      02  1D97                .Byte   Ptd$M_Controller                ; Enforced DQ codes
                51 44 00' 1D98                .Ascic  "DQ"    ; Enforced device name prefix
                      02  1D98
                      86  1D9B                .BYTE   Pd$_Literal     ; Indicate literal
                00F26200  1D9C                .LONG   ^xF26200                ; Constant
                      88  1DA0                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1DA1                .WORD   hp$a_device             ; Byte hp$a_device to data
                      00  1DA3                .BYTE   0                       ; 0 hp$a_device to data
                      1D  1DA4                .BYTE   29                      ; 29 of data field
                      86  1DA5                .BYTE   Pd$_Literal     ; Indicate literal
                000000A8  1DA6                .LONG   ^o250           ; Constant
                      88  1DAA                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1DAB                .WORD   hp$w_vector             ; Byte hp$w_vector to data
                      00  1DAD                .BYTE   0                       ; 0 hp$w_vector to data
                      09  1DAE                .BYTE   9                       ; 9 of data field
                      86  1DAF                .BYTE   Pd$_Literal     ; Indicate literal
                00000005  1DB0                .LONG   5                       ; Constant
                      88  1DB4                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  1DB5                .WORD   rb730$b_br              ; Byte rb730$b_br to data
                      00  1DB7                .BYTE   0                       ; 0 rb730$b_br to data
                      08  1DB8                .BYTE   8                       ; 8 of data field
                      81  1DB9                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1DBA    336 RCF25:  $DS_RCF25                       ;               [12][22]
                          1DBA                .SAVE   LOCAL_BLOCK
                          1DBA                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RCF25_LEN,      HP$K_RCF25_LEN:
                          0032                .IIF    NB,RCF25$K_LEN, RCF25$K_LEN:
                   00001DBA                   .RESTORE
          35 32 46 43 52 00' 1DBA             .ASCIC  "RCF25" ; RCF25 name
                      05  1DBA
                      32  1DC0                .BYTE   RCF25$K_LEN             ; RCF25$K_LEN of resultant P-table
                      FF  1DC1                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  1DC2                .WORD   ^A""    ; Two character mnemonic for QIO RCF25
                      80  1DC4                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1DC5                .Byte   Pd$_Name        ; Directive op-code
                      1B  1DC6                .Byte   Ptd$M_EndDevice         ; Enforced DA codes
                41 44 00' 1DC7                .Ascic  "DA"    ; Enforced device name prefix
                      02  1DC7
                      86  1DCA                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1DCB                .LONG   HP$M_ALLOC              ; Constant
                      88  1DCF                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1DD0                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
```

```
                      00  1DD2                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1DD3                .BYTE   8               ; 8 of data field
                      81  1DD4                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1DD5    337 RC25:    $DS_RC25                        ;               [11]
                          1DD5                .SAVE   LOCAL_BLOCK
                          1DD5                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RC25_LEN,       HP$K_RC25_LEN:
                          0032                .IIF    NB,RC25$K_LEN,  RC25$K_LEN:
                      0000 1DD5                .RESTORE
               35 32 43 52 00' 1DD5                .ASCIC  "RC25"  ; RC25 name
                      04  1DD5
                      32  1DDA                .BYTE   RC25$K_LEN              ; RC25$K_LEN of resultant P-table
                      FF  1DDB                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  1DDC                .WORD   ^A""    ; Two character mnemonic for QIO RC25
                      80  1DDE                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1DDF                .Byte   Pd$_Name        ; Directive op-code
                      1B  1DE0                .Byte   Ptd$M_EndDevice         ; Enforced DA codes
                  41 44 00' 1DE1                .Ascic  "DA"    ; Enforced device name prefix
                      02  1DE1
                      86  1DE4                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1DE5                .LONG   HP$M_ALLOC              ; Constant
                      88  1DE9                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1DEA                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1DEC                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1DED                .BYTE   8               ; 8 of data field
                      81  1DEE                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1DEF    338 RD51:    $DS_RD51                        ;               [19]
                          1DEF                .SAVE   LOCAL_BLOCK
                          1DEF                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RD51$K_LEN,  RD51$K_LEN:
                      0000 1DEF                .RESTORE
               31 35 44 52 00' 1DEF                .ASCIC  "RD51"  ; RD51 name
                      04  1DEF
                      32  1DF4                .BYTE   RD51$K_LEN              ; RD51$K_LEN of resultant P-table
                      FF  1DF5                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  1DF6                .WORD   ^A""    ; Two character mnemonic for QIO RD51
                      80  1DF8                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1DF9                .Byte   Pd$_Name        ; Directive op-code
                      1B  1DFA                .Byte   Ptd$M_EndDevice         ; Enforced DU codes
                  55 44 00' 1DFB                .Ascic  "DU"    ; Enforced device name prefix
                      02  1DFB
                      86  1DFE                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1DFF                .LONG   HP$M_ALLOC              ; Constant
                      88  1E03                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1E04                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1E06                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1E07                .BYTE   8               ; 8 of data field
                      81  1E08                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1E09    339 RD52:    $DS_RD52                        ;               [19]
                          1E09                .SAVE   LOCAL_BLOCK
                          1E09                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RD52$K_LEN,  RD52$K_LEN:
                      0000 1E09                .RESTORE
               32 35 44 52 00' 1E09                .ASCIC  "RD52"  ; RD52 name
```

```
                      04  1E09
                      32  1E0E                .BYTE   RD52$K_LEN                ; RD52$K_LEN of resultant P-table
                      FF  1E0F                .BYTE   255                 ; Maximum unit number                           ;G:65
                    0000  1E10                .WORD   ^A""    ; Two character mnemonic for QIO RD52
                      80  1E12                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  1E13                .Byte   Pd$_Name            ; Directive op-code
                      1B  1E14                .Byte   Ptd$M_EndDevice             ; Enforced DU codes
                55 44 00' 1E15                .Ascic  "DU"    ; Enforced device name prefix
                      02  1E15
                      86  1E18                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  1E19                .LONG   HP$M_ALLOC                  ; Constant
                      88  1E1D                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  1E1E                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  1E20                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  1E21                .BYTE   8                   ; 8 of data field
                      81  1E22                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          1E23    340 RH750:  $DS_RH750
                          1E23                .SAVE   LOCAL_BLOCK
                          1E23                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_RH750_BR,       HP$B_RH750_BR:
                          0032                .IIF    NB,RH750$B_BR,  RH750$B_BR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_RH750_LEN,      HP$K_RH750_LEN:
                          0033                .IIF    NB,RH750$K_LEN, RH750$K_LEN:
                      0001E23                 .RESTORE
          30 35 37 48 52 00' 1E23             .ASCIC  "RH750" ; RH750 name
                      05  1E23
                      33  1E29                .BYTE   RH750$K_LEN                 ; RH750$K_LEN of resultant P-table
                      08  1E2A                .BYTE   8                   ; Maximum unit number                           ;G:65
                    0000  1E2B                .WORD   ^A""    ; Two character mnemonic for QIO RH750
                      80  1E2D                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  1E2E                .Byte   Pd$_Name            ; Directive op-code
                      01  1E2F                .Byte   Ptd$M_Unit                  ; Enforced RH codes
                48 52 00' 1E30                .Ascic  "RH"    ; Enforced device name prefix
                      02  1E30
                      86  1E33                .BYTE   Pd$_Literal         ; Indicate literal
                40F28000  1E34                .LONG   ^X40F28000                  ; Constant
                      88  1E38                .BYTE   Pd$_Store           ; Indicate store operation
                    0018  1E39                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      00  1E3B                .BYTE   0                   ; 0 HP$A_DEVICE to data
                      20  1E3C                .BYTE   32                  ; 32 of data field
                      87  1E3D                .BYTE   Pd$_Fetch           ; Indicate fetch operation
                    000B  1E3E                .WORD   HP$B_DRIVE                  ; Byte HP$B_DRIVE to data
                      00  1E40                .BYTE   0                   ; 0 HP$B_DRIVE to data
                      08  1E41                .BYTE   8                   ; 8 of data field
                      88  1E42                .BYTE   Pd$_Store           ; Indicate store operation
                    0018  1E43                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      0D  1E45                .BYTE   13                  ; 13 HP$A_DEVICE to data
                      02  1E46                .BYTE   2                   ; 2 of data field
                      87  1E47                .BYTE   Pd$_Fetch           ; Indicate fetch operation
                    0018  1E48                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      0D  1E4A                .BYTE   13                  ; 13 HP$A_DEVICE to data
                      04  1E4B                .BYTE   4                   ; 4 of data field
                      88  1E4C                .BYTE   Pd$_Store           ; Indicate store operation
                    0024  1E4D                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      02  1E4F                .BYTE   2                   ; 2 HP$W_VECTOR to data
```

```
                      04  1E50                .BYTE   4                   ; 4 of data field
                      82  1E51                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                52 42 00' 1E52                .ASCIC  "BR"        ; BR string
                      02  1E52
          00000007 00000004  1E55             .LONG   4,7         ; 4 , 7 limits on input
                      88  1E5D                .BYTE   Pd$_Store           ; Indicate store operation
                    0032  1E5E                .WORD   RH750$B_BR                  ; Byte RH750$B_BR to data
                      00  1E60                .BYTE   0                   ; 0 RH750$B_BR to data
                      08  1E61                .BYTE   8                   ; 8 of data field
                      88  1E62                .BYTE   Pd$_Store           ; Indicate store operation
                    0024  1E63                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      06  1E65                .BYTE   6                   ; 6 HP$W_VECTOR to data
                      03  1E66                .BYTE   3                   ; 3 of data field
                      87  1E67                .BYTE   Pd$_Fetch           ; Indicate fetch operation
                    0018  1E68                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      00  1E6A                .BYTE   0                   ; 0 HP$A_DEVICE to data
                      20  1E6B                .BYTE   32                  ; 32 of data field
                      88  1E6C                .BYTE   Pd$_Store           ; Indicate store operation
                    001C  1E6D                .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                      00  1E6F                .BYTE   0                   ; 0 HP$A_DVA to data
                      20  1E70                .BYTE   32                  ; 32 of data field
                      86  1E71                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  1E72                .LONG   1                   ; Constant
                      88  1E76                .BYTE   Pd$_Store           ; Indicate store operation
                    001C  1E77                .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                      0A  1E79                .BYTE   10                  ; 10 HP$A_DVA to data
                      01  1E7A                .BYTE   1                   ; 1 of data field
                      81  1E7B                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          1E7C    341 RH780:  $DS_RH780
                          1E7C                .SAVE   LOCAL_BLOCK
                          1E7C                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_RH780_TR,       HP$B_RH780_TR:
                          0032                .IIF    NB,RH780$B_TR,  RH780$B_TR:
                00000033  0032                .IIF    NB,,BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$B_RH780_BR,       HP$B_RH780_BR:
                          0033                .IIF    NB,RH780$B_BR,  RH780$B_BR:
                00000034  0033                .IIF    NB,,BLKB,       .BLKB   1
                          0034                .IIF    NB,HP$K_RH780_LEN,      HP$K_RH780_LEN:
                          0034                .IIF    NB,RH780$K_LEN, RH780$K_LEN:
                      0001E7C                 .RESTORE
          30 38 37 48 52 00' 1E7C             .ASCIC  "RH780" ; RH780 name
                      05  1E7C
                      34  1E82                .BYTE   RH780$K_LEN                 ; RH780$K_LEN of resultant P-table
                      08  1E83                .BYTE   8                   ; Maximum unit number                       ;G:65
                    0000  1E84                .WORD   ^A""        ; Two character mnemonic for QIO RH780
                      80  1E86                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  1E87                .Byte   Pd$_Name            ; Directive op-code
                      01  1E88                .Byte   Ptd$M_Unit                  ; Enforced RH codes
                48 52 00' 1E89                .Ascic  "RH"        ; Enforced device name prefix
                      02  1E89
                      82  1E8C                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                52 54 00' 1E8D                .ASCIC  "TR"        ; TR string
                      02  1E8D
          0000000F 00000001  1E90             .LONG   1,15        ; 1 , 15 limits on input
                      88  1E98                .BYTE   Pd$_Store           ; Indicate store operation
                    0032  1E99                .WORD   RH780$B_TR                  ; Byte RH780$B_TR to data
```

```
                      00  1E9B                .BYTE   0               ; 0 RH780$B_TR to data
                      08  1E9C                .BYTE   8               ; 8 of data field
                      88  1E9D                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1E9E                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      0D  1EA0                .BYTE   13              ; 13 HP$A_DEVICE to data
                      04  1EA1                .BYTE   4               ; 4 of data field
                      88  1EA2                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1EA3                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      02  1EA5                .BYTE   2               ; 2 HP$W_VECTOR to data
                      04  1EA6                .BYTE   4               ; 4 of data field
                      82  1EA7                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 1EA8                .ASCIC  "BR"    ; BR string
                      02  1EA8
             00000007 00000004  1EAB          .LONG   4,7     ; 4 , 7 limits on input
                      88  1EB3                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  1EB4                .WORD   RH780$B_BR              ; Byte RH780$B_BR to data
                      00  1EB6                .BYTE   0               ; 0 RH780$B_BR to data
                      08  1EB7                .BYTE   8               ; 8 of data field
                      88  1EB8                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1EB9                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      06  1EBB                .BYTE   6               ; 6 HP$W_VECTOR to data
                      02  1EBC                .BYTE   2               ; 2 of data field
                      86  1EBD                .BYTE   Pd$_Literal     ; Indicate literal
                00000006  1EBE                .LONG   6               ; Constant
                      88  1EC2                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1EC3                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      1C  1EC5                .BYTE   28              ; 28 HP$A_DEVICE to data
                      04  1EC6                .BYTE   4               ; 4 of data field
                      87  1EC7                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  1EC8                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  1ECA                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  1ECB                .BYTE   32              ; 32 of data field
                      88  1ECC                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  1ECD                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      00  1ECF                .BYTE   0               ; 0 HP$A_DVA to data
                      20  1ED0                .BYTE   32              ; 32 of data field
                      86  1ED1                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1ED2                .LONG   1               ; Constant
                      88  1ED6                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  1ED7                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      0A  1ED9                .BYTE   10              ; 10 HP$A_DVA to data
                      01  1EDA                .BYTE   1               ; 1 of data field
                      88  1EDB                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1EDC                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      08  1EDE                .BYTE   8               ; 8 HP$W_VECTOR to data
                      01  1EDF                .BYTE   1               ; 1 of data field
                      81  1EE0                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1EE1    342 RL01:    $DS_RL01
                          1EE1                .SAVE   LOCAL_BLOCK
                          1EE1                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RL01_LEN,       HP$K_RL01_LEN:
                          0032                .IIF    NB,RL01$K_LEN,  RL01$K_LEN:
                      0001EE1                 .RESTORE
           31 30 4C 52 00' 1EE1                .ASCIC  "RL01"  ; RL01 name
                      04  1EE1
                      32  1EE6                .BYTE   RL01$K_LEN              ; RL01$K_LEN of resultant P-table
```

```
                 08  1EE7                .BYTE   8                   ; Maximum unit number                    ;G:65
               0000  1EE8                .WORD   ^A""      ; Two character mnemonic for QIO RL01
                 80  1EEA                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                 8D  1EEB                .Byte   Pd$_Name            ; Directive op-code
                 1B  1EEC                .Byte   Ptd$M_EndDevice             ; Enforced DL codes
          4C 44 00'  1EED                .Ascic  "DL"      ; Enforced device name prefix
                 02  1EED
                 86  1EF0                .BYTE   Pd$_Literal         ; Indicate literal
          00000001  1EF1                .LONG   HP$M_ALLOC                  ; Constant
                 88  1EF5                .BYTE   Pd$_Store           ; Indicate store operation
               000A  1EF6                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                 00  1EF8                .BYTE   0                   ; 0 HP$B_FLAGS to data
                 08  1EF9                .BYTE   8                   ; 8 of data field
                 81  1EFA                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                     1EFB   343 RL02:    $DS_RL02
                     1EFB                .SAVE   LOCAL_BLOCK
                     1EFB                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                     0032                .IIF    NB,HP$K_RL02_LEN,       HP$K_RL02_LEN:
                     0032                .IIF    NB,RL02$K_LEN,  RL02$K_LEN:
                 00001EFB                .RESTORE
   32 30 4C 52 00'  1EFB                .ASCIC  "RL02"  ; RL02 name
                 04  1EFB
                 32  1F00                .BYTE   RL02$K_LEN                  ; RL02$K_LEN of resultant P-table
                 08  1F01                .BYTE   8                   ; Maximum unit number                    ;G:65
               0000  1F02                .WORD   ^A""      ; Two character mnemonic for QIO RL02
                 80  1F04                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                 8D  1F05                .Byte   Pd$_Name            ; Directive op-code
                 1B  1F06                .Byte   Ptd$M_EndDevice             ; Enforced DL codes
          4C 44 00'  1F07                .Ascic  "DL"      ; Enforced device name prefix
                 02  1F07
                 86  1F0A                .BYTE   Pd$_Literal         ; Indicate literal
          00000001  1F0B                .LONG   HP$M_ALLOC                  ; Constant
                 88  1F0F                .BYTE   Pd$_Store           ; Indicate store operation
               000A  1F10                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                 00  1F12                .BYTE   0                   ; 0 HP$B_FLAGS to data
                 08  1F13                .BYTE   8                   ; 8 of data field
                 81  1F14                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                     1F15   344 RL11:    $DS_RL11
                     1F15                .SAVE   LOCAL_BLOCK
                     1F15                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                     0032                .IIF    NB,HP$L_RL11_CSR,       HP$L_RL11_CSR:
                     0032                .IIF    NB,RL11$L_CSR,  RL11$L_CSR:
          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                     0036                .IIF    NB,HP$B_RL11_BR,        HP$B_RL11_BR:
                     0036                .IIF    NB,RL11$B_BR,   RL11$B_BR:
          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                     0037                .IIF    NB,HP$K_RL11_LEN,       HP$K_RL11_LEN:
                     0037                .IIF    NB,RL11$K_LEN,  RL11$K_LEN:
                 00001F15                .RESTORE
   31 31 4C 52 00'  1F15                .ASCIC  "RL11"  ; RL11 name
                 04  1F15
                 37  1F1A                .BYTE   RL11$K_LEN                  ; RL11$K_LEN of resultant P-table
                 00  1F1B                .BYTE   0                   ; Maximum unit number                    ;G:65
               4C44  1F1C                .WORD   ^A"DL"  ; Two character mnemonic for QIO RL11 DL
                 80  1F1E                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
```

```
                        8D  1F1F                .Byte   Pd$_Name         ; Directive op-code
                        02  1F20                .Byte   Ptd$M_Controller                ; Enforced DL codes
                  4C 44 00' 1F21                .Ascic  "DL"    ; Enforced device name prefix
                        02  1F21
                        83  1F24                .BYTE   Pd$_Octal        ; Indicate scan octal
                  52 53 43 00' 1F25             .ASCIC  "CSR"   ; CSR string
                        03  1F25
               0003FFFE 0003E000  1F29          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  1F31                .BYTE   Pd$_Store        ; Indicate store operation
                      0032  1F32                .WORD   RL11$L_CSR               ; Byte RL11$L_CSR to data
                        00  1F34                .BYTE   0                ; 0 RL11$L_CSR to data
                        20  1F35                .BYTE   32               ; 32 of data field
                        88  1F36                .BYTE   Pd$_Store        ; Indicate store operation
                      0018  1F37                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                        00  1F39                .BYTE   0                ; 0 HP$A_DEVICE to data
                        12  1F3A                .BYTE   18               ; 18 of data field
                        83  1F3B                .BYTE   Pd$_Octal        ; Indicate scan octal
            52 4F 54 43 45 56 00' 1F3C          .ASCIC  "VECTOR"         ; VECTOR string
                        06  1F3C
               000001FE 00000002  1F43          .LONG   ^O2,^O776        ; 2 , 776 limits on input
                        88  1F4B                .BYTE   Pd$_Store        ; Indicate store operation
                      0024  1F4C                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                        00  1F4E                .BYTE   0                ; 0 HP$W_VECTOR to data
                        09  1F4F                .BYTE   9                ; 9 of data field
                        82  1F50                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                  52 42 00' 1F51                .ASCIC  "BR"    ; BR string
                        02  1F51
               00000007 00000004  1F54          .LONG   4,7      ; 4 , 7 limits on input
                        88  1F5C                .BYTE   Pd$_Store        ; Indicate store operation
                      0036  1F5D                .WORD   RL11$B_BR                ; Byte RL11$B_BR to data
                        00  1F5F                .BYTE   0                ; 0 RL11$B_BR to data
                        08  1F60                .BYTE   8                ; 8 of data field
                        81  1F61                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                            1F62    345 RK06:   $DS_RK06
                            1F62                .SAVE   LOCAL_BLOCK
                            1F62                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_RK06_LEN,       HP$K_RK06_LEN:
                            0032                .IIF    NB,RK06$K_LEN,  RK06$K_LEN:
                        00001F62                .RESTORE
               36 30 4B 52 00' 1F62             .ASCIC  "RK06"   ; RK06 name
                        04  1F62
                        32  1F67                .BYTE   RK06$K_LEN               ; RK06$K_LEN of resultant P-table
                        08  1F68                .BYTE   8                ; Maximum unit number                          ;G:65
                      0000  1F69                .WORD   ^A""     ; Two character mnemonic for QIO RK06
                        80  1F6B                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                        8D  1F6C                .Byte   Pd$_Name         ; Directive op-code
                        1B  1F6D                .Byte   Ptd$M_EndDevice          ; Enforced DM codes
                  4D 44 00' 1F6E                .Ascic  "DM"    ; Enforced device name prefix
                        02  1F6E
                        86  1F71                .BYTE   Pd$_Literal      ; Indicate literal
                  00000001  1F72                .LONG   HP$M_ALLOC               ; Constant
                        88  1F76                .BYTE   Pd$_Store        ; Indicate store operation
                      000A  1F77                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                        00  1F79                .BYTE   0                ; 0 HP$B_FLAGS to data
                        08  1F7A                .BYTE   8                ; 8 of data field
                        81  1F7B                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
```

```
                          1F7C   346 RK07:   $DS_RK07
                          1F7C               .SAVE   LOCAL_BLOCK
                          1F7C               .PSECT  $ABS$,ABS
                 00000032 0050               .=HP$A_DEPENDENT
                          0032               .IIF    NB,HP$K_RK07_LEN,       HP$K_RK07_LEN:
                          0032               .IIF    NB,RK07$K_LEN,  RK07$K_LEN:
                      00001F7C               .RESTORE
           37 30 4B 52 00' 1F7C               .ASCIC  "RK07"  ; RK07 name
                       04  1F7C
                       32  1F81               .BYTE   RK07$K_LEN              ; RK07$K_LEN of resultant P-table
                       08  1F82               .BYTE   8                       ; Maximum unit number                   ;G:65
                     0000  1F83               .WORD   ^A""    ; Two character mnemonic for QIO RK07
                       80  1F85               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  1F86               .Byte   Pd$_Name        ; Directive op-code
                       1B  1F87               .Byte   Ptd$M_EndDevice         ; Enforced DM codes
                 4D 44 00' 1F88               .Ascic  "DM"    ; Enforced device name prefix
                       02  1F88
                       86  1F8B               .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  1F8C               .LONG   HP$M_ALLOC              ; Constant
                       88  1F90               .BYTE   Pd$_Store       ; Indicate store operation
                     000A  1F91               .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                       00  1F93               .BYTE   0                       ; 0 HP$B_FLAGS to data
                       08  1F94               .BYTE   8                       ; 8 of data field
                       81  1F95               .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1F96   347 RK611:   $DS_RK611
                          1F96               .SAVE   LOCAL_BLOCK
                          1F96               .PSECT  $ABS$,ABS
                 00000032 0050               .=HP$A_DEPENDENT
                          0032               .IIF    NB,HP$L_RK611_CSR,      HP$L_RK611_CSR:
                          0032               .IIF    NB,RK611$L_CSR, RK611$L_CSR:
                 00000036 0032               .IIF    NB,,.BLKL,      .BLKL   1
                          0036               .IIF    NB,HP$B_RK611_BR,       HP$B_RK611_BR:
                          0036               .IIF    NB,RK611$B_BR,  RK611$B_BR:
                 00000037 0036               .IIF    NB,,.BLKB,      .BLKB   1
                          0037               .IIF    NB,HP$K_RK611_LEN,      HP$K_RK611_LEN:
                          0037               .IIF    NB,RK611$K_LEN, RK611$K_LEN:
                      00001F96               .RESTORE
        31 31 36 4B 52 00' 1F96               .ASCIC  "RK611" ; RK611 name
                       05  1F96
                       37  1F9C               .BYTE   RK611$K_LEN             ; RK611$K_LEN of resultant P-table
                       00  1F9D               .BYTE   0                       ; Maximum unit number                   ;G:65
                     4D44  1F9E               .WORD   ^A"DM"  ; Two character mnemonic for QIO RK611 DM
                       80  1FA0               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  1FA1               .Byte   Pd$_Name        ; Directive op-code
                       02  1FA2               .Byte   Ptd$M_Controller                ; Enforced DM codes
                 4D 44 00' 1FA3               .Ascic  "DM"    ; Enforced device name prefix
                       02  1FA3
                       83  1FA6               .BYTE   Pd$_Octal       ; Indicate scan octal
              52 53 43 00' 1FA7               .ASCIC  "CSR"   ; CSR string
                       03  1FA7
          0003FFFE 0003E000  1FAB               .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                       88  1FB3               .BYTE   Pd$_Store       ; Indicate store operation
                     0032  1FB4               .WORD   RK611$L_CSR             ; Byte RK611$L_CSR to data
                       00  1FB6               .BYTE   0                       ; 0 RK611$L_CSR to data
                       20  1FB7               .BYTE   32                      ; 32 of data field
                       88  1FB8               .BYTE   Pd$_Store       ; Indicate store operation
                     0018  1FB9               .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
```

```
                      00  1FBB                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  1FBC                .BYTE   18                      ; 18 of data field
                      83  1FBD                .BYTE   Pd$_Octal               ; Indicate scan octal
          52 4F 54 43 45 56 00' 1FBE           .ASCIC  "VECTOR"                ; VECTOR string
                      06  1FBE
             000001FE 00000002  1FC5           .LONG   ^O2,^O776               ; 2 , 776 limits on input
                      88  1FCD                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  1FCE                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  1FD0                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  1FD1                .BYTE   9                       ; 9 of data field
                      82  1FD2                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                   52 42 00' 1FD3             .ASCIC  "BR"    ; BR string
                      02  1FD3
             00000007 00000004  1FD6           .LONG   4,7     ; 4 , 7 limits on input
                      88  1FDE                .BYTE   Pd$_Store               ; Indicate store operation
                    0036  1FDF                .WORD   RK611$B_BR                      ; Byte RK611$B_BR to data
                      00  1FE1                .BYTE   0                       ; 0 RK611$B_BR to data
                      08  1FE2                .BYTE   8                       ; 8 of data field
                      81  1FE3                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          1FE4    348 RM03:    $DS_RM03
                          1FE4                .SAVE   LOCAL_BLOCK
                          1FE4                .PSECT  $ABS$,ABS
                   00000032  0050             .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RM03_LEN,       HP$K_RM03_LEN:
                          0032                .IIF    NB,RM03$K_LEN,  RM03$K_LEN:
                      00001FE4                .RESTORE
          33 30 4D 52 00' 1FE4                .ASCIC  "RM03"  ; RM03 name
                      04  1FE4
                      32  1FE9                .BYTE   RM03$K_LEN                      ; RM03$K_LEN of resultant P-table
                      08  1FEA                .BYTE   8                       ; Maximum unit number                   ;G:65
                    5244  1FEB                .WORD   ^A"DR"  ; Two character mnemonic for QIO RM03 DR
                      80  1FED                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  1FEE                .Byte   Pd$_Name                ; Directive op-code
                      03  1FEF                .Byte   Ptd$M_Device                    ; Enforced DR codes
                   52 44 00' 1FF0             .Ascic  "DR"    ; Enforced device name prefix
                      02  1FF0
                      86  1FF3                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  1FF4                .LONG   HP$M_ALLOC                      ; Constant
                      88  1FF8                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  1FF9                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  1FFB                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  1FFC                .BYTE   8                       ; 8 of data field
                      87  1FFD                .BYTE   Pd$_Fetch               ; Indicate fetch operation
                    000B  1FFE                .WORD   HP$B_DRIVE                      ; Byte HP$B_DRIVE to data
                      00  2000                .BYTE   0                       ; 0 HP$B_DRIVE to data
                      08  2001                .BYTE   8                       ; 8 of data field
                      88  2002                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  2003                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      07  2005                .BYTE   7                       ; 7 HP$A_DEVICE to data
                      03  2006                .BYTE   3                       ; 3 of data field
                      81  2007                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          2008    349 RM05:    $DS_RM05
                          2008                .SAVE   LOCAL_BLOCK
                          2008                .PSECT  $ABS$,ABS
                   00000032  0050             .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RM05_LEN,       HP$K_RM05_LEN:
                          0032                .IIF    NB,RM05$K_LEN,  RM05$K_LEN:
```

```
                    00002008                    .RESTORE
          35 30 4D 52 00' 2008                  .ASCIC  "RM05"  ; RM05 name
                      04  2008
                      32  200D                  .BYTE   RM05$K_LEN                  ; RM05$K_LEN of resultant P-table
                      08  200E                  .BYTE   8                   ; Maximum unit number                       ;G:65
                    5244  200F                  .WORD   ^A"DR"  ; Two character mnemonic for QIO RM05 DR
                      80  2011                  .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  2012                  .Byte   Pd$_Name            ; Directive op-code
                      03  2013                  .Byte   Ptd$M_Device                ; Enforced DR codes
                   52 44 00' 2014               .Ascic  "DR"    ; Enforced device name prefix
                      02  2014
                      86  2017                  .BYTE   Pd$_Literal     ; Indicate literal
                00000001  2018                  .LONG   HP$M_ALLOC                  ; Constant
                      88  201C                  .BYTE   Pd$_Store           ; Indicate store operation
                    000A  201D                  .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  201F                  .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  2020                  .BYTE   8                   ; 8 of data field
                      87  2021                  .BYTE   Pd$_Fetch           ; Indicate fetch operation
                    000B  2022                  .WORD   HP$B_DRIVE                  ; Byte HP$B_DRIVE to data
                      00  2024                  .BYTE   0                   ; 0 HP$B_DRIVE to data
                      08  2025                  .BYTE   8                   ; 8 of data field
                      88  2026                  .BYTE   Pd$_Store           ; Indicate store operation
                    0018  2027                  .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      07  2029                  .BYTE   7                   ; 7 HP$A_DEVICE to data
                      03  202A                  .BYTE   3                   ; 3 of data field
                      81  202B                  .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          202C    350 RM80:     $DS_RM80
                          202C                  .SAVE   LOCAL_BLOCK
                          202C                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,HP$K_RM80_LEN,       HP$K_RM80_LEN:
                          0032                  .IIF    NB,RM80$K_LEN,  RM80$K_LEN:
                    0000202C                    .RESTORE
          30 38 4D 52 00' 202C                  .ASCIC  "RM80"  ; RM80 name
                      04  202C
                      32  2031                  .BYTE   RM80$K_LEN                  ; RM80$K_LEN of resultant P-table
                      08  2032                  .BYTE   8                   ; Maximum unit number                       ;G:65
                    5244  2033                  .WORD   ^A"DR"  ; Two character mnemonic for QIO RM80 DR
                      80  2035                  .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  2036                  .Byte   Pd$_Name            ; Directive op-code
                      03  2037                  .Byte   Ptd$M_Device                ; Enforced DR codes
                   52 44 00' 2038               .Ascic  "DR"    ; Enforced device name prefix
                      02  2038
                      86  203B                  .BYTE   Pd$_Literal     ; Indicate literal
                00000001  203C                  .LONG   HP$M_ALLOC                  ; Constant
                      88  2040                  .BYTE   Pd$_Store           ; Indicate store operation
                    000A  2041                  .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  2043                  .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  2044                  .BYTE   8                   ; 8 of data field
                      87  2045                  .BYTE   Pd$_Fetch           ; Indicate fetch operation
                    000B  2046                  .WORD   HP$B_DRIVE                  ; Byte HP$B_DRIVE to data
                      00  2048                  .BYTE   0                   ; 0 HP$B_DRIVE to data
                      08  2049                  .BYTE   8                   ; 8 of data field
                      88  204A                  .BYTE   Pd$_Store           ; Indicate store operation
                    0018  204B                  .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      07  204D                  .BYTE   7                   ; 7 HP$A_DEVICE to data
                      03  204E                  .BYTE   3                   ; 3 of data field
```

```
                      81  204F                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2050    351 RP04:   $DS_RP04
                          2050                .SAVE   LOCAL_BLOCK
                          2050                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RP04_LEN,       HP$K_RP04_LEN:
                          0032                .IIF    NB,RP04$K_LEN,  RP04$K_LEN:
                    0000205O                .RESTORE
          34 30 50 52 00' 2050                .ASCIC  "RP04"  ; RP04 name
                      04  2050
                      32  2055                .BYTE   RP04$K_LEN              ; RP04$K_LEN of resultant P-table
                      08  2056                .BYTE   8                       ; Maximum unit number                   ;G:65
                    4244  2057                .WORD   ^A"DB"  ; Two character mnemonic for QIO RP04 DB
                      80  2059                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  205A                .Byte   Pd$_Name        ; Directive op-code
                      03  205B                .Byte   Ptd$M_Device            ; Enforced DB codes
                42 44 00' 205C                .Ascic  "DB"    ; Enforced device name prefix
                      02  205C
                      86  205F                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  2060                .LONG   HP$M_ALLOC              ; Constant
                      88  2064                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  2065                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  2067                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2068                .BYTE   8                       ; 8 of data field
                      87  2069                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    000B  206A                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                      00  206C                .BYTE   0                       ; 0 HP$B_DRIVE to data
                      08  206D                .BYTE   8                       ; 8 of data field
                      88  206E                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  206F                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      07  2071                .BYTE   7                       ; 7 HP$A_DEVICE to data
                      03  2072                .BYTE   3                       ; 3 of data field
                      81  2073                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2074    352 RP05:   $DS_RP05
                          2074                .SAVE   LOCAL_BLOCK
                          2074                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RP05_LEN,       HP$K_RP05_LEN:
                          0032                .IIF    NB,RP05$K_LEN,  RP05$K_LEN:
                    0000207O                .RESTORE
          35 30 50 52 00' 2074                .ASCIC  "RP05"  ; RP05 name
                      04  2074
                      32  2079                .BYTE   RP05$K_LEN              ; RP05$K_LEN of resultant P-table
                      08  207A                .BYTE   8                       ; Maximum unit number                   ;G:65
                    4244  207B                .WORD   ^A"DB"  ; Two character mnemonic for QIO RP05 DB
                      80  207D                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  207E                .Byte   Pd$_Name        ; Directive op-code
                      03  207F                .Byte   Ptd$M_Device            ; Enforced DB codes
                42 44 00' 2080                .Ascic  "DB"    ; Enforced device name prefix
                      02  2080
                      86  2083                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  2084                .LONG   HP$M_ALLOC              ; Constant
                      88  2088                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  2089                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  208B                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  208C                .BYTE   8                       ; 8 of data field
                      87  208D                .BYTE   Pd$_Fetch       ; Indicate fetch operation
```

```
                     000B  208E                .WORD   HP$B_DRIVE                ; Byte HP$B_DRIVE to data
                       00  2090                .BYTE   0                   ; 0 HP$B_DRIVE to data
                       08  2091                .BYTE   8                   ; 8 of data field
                       88  2092                .BYTE   Pd$_Store           ; Indicate store operation
                     0018  2093                .WORD   HP$A_DEVICE                ; Byte HP$A_DEVICE to data
                       07  2095                .BYTE   7                   ; 7 HP$A_DEVICE to data
                       03  2096                .BYTE   3                   ; 3 of data field
                       81  2097                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           2098    353 RP06:   $DS_RP06
                           2098                .SAVE   LOCAL_BLOCK
                           2098                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_RP06_LEN,       HP$K_RP06_LEN:
                           0032                .IIF    NB,RP06$K_LEN,  RP06$K_LEN:
                       00002098                .RESTORE
              36 30 50 52 00' 2098             .ASCIC  "RP06"  ; RP06 name
                       04  2098
                       32  209D                .BYTE   RP06$K_LEN                 ; RP06$K_LEN of resultant P-table
                       08  209E                .BYTE   8                   ; Maximum unit number                       ;G:65
                     4244  209F                .WORD   ^A"DB"  ; Two character mnemonic for QIO RP06 DB
                       80  20A1                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  20A2                .Byte   Pd$_Name            ; Directive op-code
                       03  20A3                .Byte   Ptd$M_Device                ; Enforced DB codes
                  42 44 00' 20A4                .Asic   "DB"    ; Enforced device name prefix
                       02  20A4
                       86  20A7                .BYTE   Pd$_Literal         ; Indicate literal
                 00000001  20A8                .LONG   HP$M_ALLOC                 ; Constant
                       88  20AC                .BYTE   Pd$_Store           ; Indicate store operation
                     000A  20AD                .WORD   HP$B_FLAGS                 ; Byte HP$B_FLAGS to data
                       00  20AF                .BYTE   0                   ; 0 HP$B_FLAGS to data
                       08  20B0                .BYTE   8                   ; 8 of data field
                       87  20B1                .BYTE   Pd$_Fetch           ; Indicate fetch operation
                     000B  20B2                .WORD   HP$B_DRIVE                 ; Byte HP$B_DRIVE to data
                       00  20B4                .BYTE   0                   ; 0 HP$B_DRIVE to data
                       08  20B5                .BYTE   8                   ; 8 of data field
                       88  20B6                .BYTE   Pd$_Store           ; Indicate store operation
                     0018  20B7                .WORD   HP$A_DEVICE                ; Byte HP$A_DEVICE to data
                       07  20B9                .BYTE   7                   ; 7 HP$A_DEVICE to data
                       03  20BA                .BYTE   3                   ; 3 of data field
                       81  20BB                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           20BC    354 RP07:   $DS_RP07
                           20BC                .SAVE   LOCAL_BLOCK
                           20BC                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_RP07_LEN,       HP$K_RP07_LEN:
                           0032                .IIF    NB,RP07$K_LEN,  RP07$K_LEN:
                       000020BC                .RESTORE
              37 30 50 52 00' 20BC             .ASCIC  "RP07"  ; RP07 name
                       04  20BC
                       32  20C1                .BYTE   RP07$K_LEN                 ; RP07$K_LEN of resultant P-table
                       08  20C2                .BYTE   8                   ; Maximum unit number                       ;G:65
                     5244  20C3                .WORD   ^A"DR"  ; Two character mnemonic for QIO RP07 DR
                       80  20C5                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  20C6                .Byte   Pd$_Name            ; Directive op-code
                       03  20C7                .Byte   Ptd$M_Device                ; Enforced DR codes
                  52 44 00' 20C8                .Asic   "DR"    ; Enforced device name prefix
                       02  20C8
```

```
                      86  20CB                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  20CC                .LONG   HP$M_ALLOC               ; Constant
                      88  20D0                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  20D1                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  20D3                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  20D4                .BYTE   8               ; 8 of data field
                      87  20D5                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    000B  20D6                .WORD   HP$B_DRIVE               ; Byte HP$B_DRIVE to data
                      00  20D8                .BYTE   0               ; 0 HP$B_DRIVE to data
                      08  20D9                .BYTE   8               ; 8 of data field
                      88  20DA                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  20DB                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                      07  20DD                .BYTE   7               ; 7 HP$A_DEVICE to data
                      03  20DE                .BYTE   3               ; 3 of data field
                      81  20DF                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          20E0    355 RQDX1:    $DS_RQDX1                       ;[19]                          ;G:4
                          20E0                .SAVE   LOCAL_BLOCK
                          20E0                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RQDX1$L_IP,  RQDX1$L_IP:
                00000036  0032                .IIF    NB,..BLKL,      .BLKL   1
                          0036                .IIF    NB,RQDX1$K_LEN, RQDX1$K_LEN:
                      000020E0                .RESTORE
          31 58 44 51 52 00' 20E0             .ASCIC  "RQDX1" ; RQDX1 name
                      05  20E0
                      36  20E6                .BYTE   RQDX1$K_LEN             ; RQDX1$K_LEN of resultant P-table
                      00  20E7                .BYTE   0               ; Maximum unit number                  ;G:65
                    0000  20E8                .WORD   ^A""    ; Two character mnemonic for QIO RQDX1
                      80  20EA                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  20EB                .Byte   Pd$_Name        ; Directive op-code
                      02  20EC                .Byte   Ptd$M_Controller                ; Enforced DU codes
                55 44 00' 20ED                .Ascic  "DU"    ; Enforced device name prefix
                      02  20ED
                      83  20F0                .BYTE   Pd$_Octal       ; Indicate scan octal
                50 49 00' 20F1                .ASCIC  "IP"    ; IP string
                      02  20F1
          0003FFFC 0003E000  20F4             .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                      88  20FC                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  20FD                .WORD   RQDX1$L_IP              ; Byte RQDX1$L_IP to data
                      00  20FF                .BYTE   0               ; 0 RQDX1$L_IP to data
                      20  2100                .BYTE   32              ; 32 of data field
                      88  2101                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2102                .WORD   Hp$A_Device              ; Byte Hp$A_Device to data
                      00  2104                .BYTE   0               ; 0 Hp$A_Device to data
                      12  2105                .BYTE   18              ; 18 of data field
                      81  2106                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2107    356 RRD50:    $DS_RRD50                       ;[35]                          ;G:4
                          2107                .SAVE   LOCAL_BLOCK
                          2107                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RRD50$L_CSR, RRD50$L_CSR::
                00000036  0032                .IIF    NB,..BLKL,      .BLKL   1
                          0036                .IIF    NB,RRD50$B_BR,  RRD50$B_BR::
                00000037  0036                .IIF    NB,..BLKB,      .BLKB   1
                          0037                .IIF    NB,RRD50$K_LEN, RRD50$K_LEN::
                      00002107                .RESTORE
          30 35 44 52 52 00' 2107             .ASCIC  "RRD50" ; RRD50 name
```

```
                      05  2107
                      37  210D                .BYTE   RRD50$K_LEN             ; RRD50$K_LEN of resultant P-table
                      04  210E                .BYTE   4                       ; Maximum unit number                   ;G:65
                    0000  210F                .WORD   ^A""    ; Two character mnemonic for QIO RRD50
                      80  2111                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2112                .Byte   Pd$_Name        ; Directive op-code
                      03  2113                .Byte   PTD$M_DEVICE            ; Enforced DU codes
                55 44 00' 2114                .Ascic  "DU"    ; Enforced device name prefix
                      02  2114
                      86  2117                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  2118                .LONG   HP$M_ALLOC              ; Constant
                      88  211C                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  211D                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  211F                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2120                .BYTE   8                       ; 8 of data field
                      83  2121                .BYTE   Pd$_Octal       ; Indicate scan octal
     53 53 45 52 44 44 41 00' 2122                .ASCIC  "ADDRESS"       ; ADDRESS string
                      07  2122
            0003FFFE 0003E008  212A                .LONG   ^O760010,^O777776       ; 760010 , 777776 limits on input
                      88  2132                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  2133                .WORD   RRD50$L_CSR             ; Byte RRD50$L_CSR to data
                      00  2135                .BYTE   0                       ; 0 RRD50$L_CSR to data
                      20  2136                .BYTE   32                      ; 32 of data field
                      88  2137                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2138                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  213A                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  213B                .BYTE   18                      ; 18 of data field
                      83  213C                .BYTE   Pd$_Octal       ; Indicate scan octal
        52 4F 54 43 45 56 00' 213D                .ASCIC  "VECTOR"        ; VECTOR string
                      06  213D
            000001FC 00000040  2144                .LONG   ^O100,^O774     ; 100 , 774 limits on input
                      88  214C                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  214D                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  214F                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      10  2150                .BYTE   16                      ; 16 of data field
                      82  2151                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 2152                .ASCIC  "BR"    ; BR string
                      02  2152
            00000007 00000004  2155                .LONG   4,7     ; 4 , 7 limits on input
                      88  215D                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  215E                .WORD   RRD50$B_BR              ; Byte RRD50$B_BR to data
                      00  2160                .BYTE   0                       ; 0 RRD50$B_BR to data
                      08  2161                .BYTE   8                       ; 8 of data field
                      81  2162                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2163    357 RUX50:  $DS_RUX50                       ;[26]                                   ;G:4
                          2163                .SAVE   LOCAL_BLOCK
                          2163                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_RUX50_BR,       HP$B_RUX50_BR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_RUX50_LEN,      HP$K_RUX50_LEN:
                      0000216300              .RESTORE
        30 35 58 55 52 00' 2163                .ASCIC  "RUX50" ; RUX50 name
                      05  2163
                      33  2169                .BYTE   HP$K_RUX50_LEN          ; HP$K_RUX50_LEN of resultant P-table
                      04  216A                .BYTE   4                       ; Maximum unit number                   ;G:65
                    0000  216B                .WORD   ^A""    ; Two character mnemonic for QIO RUX50
```

```
                        80  216D                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  216E                .Byte   Pd$_Name        ; Directive op-code
                        02  216F                .Byte   PTD$M_CONTROLLER                ; Enforced DU codes
                 55 44 00'  2170                .Ascic  "DU"    ; Enforced device name prefix
                        02  2170
                        83  2173                .BYTE   Pd$_Octal       ; Indicate scan octal
                 50 49 00'  2174                .ASCIC  "IP"    ; IP string
                        02  2174
            0003FFFC 0003E000   2177            .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                        88  217F                .BYTE   Pd$_Store       ; Indicate store operation
                      0018  2180                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        00  2182                .BYTE   0               ; 0 HP$A_DEVICE to data
                        12  2183                .BYTE   18              ; 18 of data field
                        83  2184                .BYTE   Pd$_Octal       ; Indicate scan octal
     52 4F 54 43 45 56 00'  2185                .ASCIC  "VECTOR"        ; VECTOR string
                        06  2185
            000001FC 00000004   218C            .LONG   ^O4,^O774       ; 4 , 774 limits on input
                        88  2194                .BYTE   Pd$_Store       ; Indicate store operation
                      0024  2195                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                        00  2197                .BYTE   0               ; 0 HP$W_VECTOR to data
                        09  2198                .BYTE   9               ; 9 of data field
                        82  2199                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00'  219A                .ASCIC  "BR"    ; BR string
                        02  219A
            00000007 00000004   219D            .LONG   4,7     ; 4 , 7 limits on input
                        88  21A5                .BYTE   Pd$_Store       ; Indicate store operation
                      0032  21A6                .WORD   HP$B_RUX50_BR           ; Byte HP$B_RUX50_BR to data
                        00  21A8                .BYTE   0               ; 0 HP$B_RUX50_BR to data
                        08  21A9                .BYTE   8               ; 8 of data field
                        81  21AA                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            21AB    358 RX02:   $DS_RX02
                            21AB                .SAVE   LOCAL_BLOCK
                            21AB                .PSECT  $ABS$,ABS
                00000032    0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_RX02_LEN,       HP$K_RX02_LEN:
                            0032                .IIF    NB,RX02$K_LEN,  RX02$K_LEN:
                            21AB                .RESTORE
     32 30 58 52 00'  21AB                      .ASCIC  "RX02"  ; RX02 name
                        04  21AB
                        32  21B0                .BYTE   RX02$K_LEN              ; RX02$K_LEN of resultant P-table
                        02  21B1                .BYTE   2               ; Maximum unit number                   ;G:65
                      0000  21B2                .WORD   ^A""    ; Two character mnemonic for QIO RX02
                        80  21B4                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  21B5                .Byte   Pd$_Name        ; Directive op-code
                        1B  21B6                .Byte   Ptd$M_EndDevice         ; Enforced DY codes
                 59 44 00'  21B7                .Ascic  "DY"    ; Enforced device name prefix
                        02  21B7
                        86  21BA                .BYTE   Pd$_Literal     ; Indicate literal
                00000001    21BB                .LONG   HP$M_ALLOC              ; Constant
                        88  21BF                .BYTE   Pd$_Store       ; Indicate store operation
                      000A  21C0                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                        00  21C2                .BYTE   0               ; 0 HP$B_FLAGS to data
                        08  21C3                .BYTE   8               ; 8 of data field
                        81  21C4                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            21C5    359 RX211:  $DS_RX211
                            21C5                .SAVE   LOCAL_BLOCK
                            21C5                .PSECT  $ABS$,ABS
```

```
                  00000032  0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$L_RX211_CSR,      HP$L_RX211_CSR:
                            0032                .IIF    NB,RX211$L_CSR, RX211$L_CSR:
                  00000036  0032                .IIF    NB,,BLKL,               .BLKL   1
                            0036                .IIF    NB,HP$B_RX211_BR,       HP$B_RX211_BR:
                            0036                .IIF    NB,RX211$B_BR,  RX211$B_BR:
                  00000037  0036                .IIF    NB,,BLKB,               .BLKB   1
                            0037                .IIF    NB,HP$K_RX211_LEN,      HP$K_RX211_LEN:
                            0037                .IIF    NB,RX211$K_LEN, RX211$K_LEN:
                     000021C5                   .RESTORE
            31 31 32 58 52 00' 21C5             .ASCIC  "RX211" ; RX211 name
                        05  21C5
                        37  21CB                .BYTE   RX211$K_LEN             ; RX211$K_LEN of resultant P-table
                        00  21CC                .BYTE   0                       ; Maximum unit number                   ;G:65
                      5944  21CD                .WORD   ^A"DY"  ; Two character mnemonic for QIO RX211 DY
                        80  21CF                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  21D0                .Byte   Pd$_Name        ; Directive op-code
                        02  21D1                .Byte   Ptd$M_Controller                ; Enforced DY codes
                  59 44 00' 21D2                .Ascic  "DY"    ; Enforced device name prefix
                        02  21D2
                        83  21D5                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 53 43 00' 21D6             .ASCIC  "CSR"   ; CSR string
                        03  21D6
            0003FFFE 0003E000  21DA             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  21E2                .BYTE   Pd$_Store       ; Indicate store operation
                      0032  21E3                .WORD   RX211$L_CSR             ; Byte RX211$L_CSR to data
                        00  21E5                .BYTE   0               ; 0 RX211$L_CSR to data
                        20  21E6                .BYTE   32              ; 32 of data field
                        88  21E7                .BYTE   Pd$_Store       ; Indicate store operation
                      0018  21E8                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                        00  21EA                .BYTE   0               ; 0 HP$A_DEVICE to data
                        12  21EB                .BYTE   18              ; 18 of data field
                        83  21EC                .BYTE   Pd$_Octal       ; Indicate scan octal
      52 4F 54 43 45 56 00' 21ED                .ASCIC  "VECTOR"        ; VECTOR string
                        06  21ED
         000001FE 00000002  21F4                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                        88  21FC                .BYTE   Pd$_Store       ; Indicate store operation
                      0024  21FD                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                        00  21FF                .BYTE   0               ; 0 HP$W_VECTOR to data
                        09  2200                .BYTE   9               ; 9 of data field
                        82  2201                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                  52 42 00' 2202                .ASCIC  "BR"    ; BR string
                        02  2202
         00000007 00000004  2205                .LONG   4,7     ; 4 , 7 limits on input
                        88  220D                .BYTE   Pd$_Store       ; Indicate store operation
                      0036  220E                .WORD   RX211$B_BR              ; Byte RX211$B_BR to data
                        00  2210                .BYTE   0               ; 0 RX211$B_BR to data
                        08  2211                .BYTE   8               ; 8 of data field
                        81  2212                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            2213    360 RX31:   $DS_RX31                        ;[26]                                   ;G:4
                            2213                .SAVE   LOCAL_BLOCK
                            2213                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,RX31$K_LEN,  RX31$K_LEN:
                     00002213                   .RESTORE
               31 33 58 52 00' 2213             .ASCIC  "RX31"  ; RX31 name
                        04  2213
```

```
                      32  2218                .BYTE   RX31$K_LEN                ; RX31$K_LEN of resultant P-table
                      FF  2219                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  221A                .WORD   ^A""     ; Two character mnemonic for QIO RX31
                      80  221C                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  221D                .Byte   Pd$_Name            ; Directive op-code
                      1B  221E                .Byte   Ptd$M_EndDevice             ; Enforced DU codes
                55 44 00' 221F                .Ascic  "DU"     ; Enforced device name prefix
                      02  221F
                      81  2222                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          2223    361 RX32:   $DS_RX32                            ;[26]                       ;G:4
                          2223                .SAVE   LOCAL_BLOCK
                          2223                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RX32$K_LEN,  RX32$K_LEN:
                00002223                      .RESTORE
             32 33 58 52 00' 2223             .ASCIC  "RX32"   ; RX32 name
                      04  2223
                      32  2228                .BYTE   RX32$K_LEN                ; RX32$K_LEN of resultant P-table
                      FF  2229                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  222A                .WORD   ^A""     ; Two character mnemonic for QIO RX32
                      80  222C                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  222D                .Byte   Pd$_Name            ; Directive op-code
                      1B  222E                .Byte   Ptd$M_EndDevice             ; Enforced DU codes
                55 44 00' 222F                .Ascic  "DU"     ; Enforced device name prefix
                      02  222F
                      81  2232                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          2233    362 RX50:   $DS_RX50                            ;[19]                       ;G:4
                          2233                .SAVE   LOCAL_BLOCK
                          2233                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RX50$K_LEN,  RX50$K_LEN:
                00002233                      .RESTORE
             30 35 58 52 00' 2233             .ASCIC  "RX50"   ; RX50 name
                      04  2233
                      32  2238                .BYTE   RX50$K_LEN                ; RX50$K_LEN of resultant P-table
                      FF  2239                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  223A                .WORD   ^A""     ; Two character mnemonic for QIO RX50
                      80  223C                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      86  223D                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  223E                .LONG   HP$M_ALLOC                  ; Constant
                      88  2242                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  2243                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  2245                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  2246                .BYTE   8                   ; 8 of data field
                      81  2247                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          2248    363 RX180:  $DS_RX180                           ;[26]                       ;G:4
                          2248                .SAVE   LOCAL_BLOCK
                          2248                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RX180$K_LEN, RX180$K_LEN:
                00002248                      .RESTORE
          30 38 31 58 52 00' 2248             .ASCIC  "RX180" ; RX180 name
                      05  2248
                      32  224E                .BYTE   RX180$K_LEN               ; RX180$K_LEN of resultant P-table
                      FF  224F                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  2250                .WORD   ^A""     ; Two character mnemonic for QIO RX180
                      80  2252                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
```

```
                8D  2253                .Byte   Pd$_Name        ; Directive op-code
                1B  2254                .Byte   Ptd$M_EndDevice         ; Enforced DU codes
          55 44 00' 2255                .Ascic  "DU"    ; Enforced device name prefix
                02  2255
                81  2258                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                    2259   364 SBIA:    $DS_SBIA                        ;[15]                                   ;G:4
                    2259                .SAVE   LOCAL_BLOCK
                    2259                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                    0032                .IIF    NB,HP$K_SBIA_LEN,       HP$K_SBIA_LEN:
                    0032                .IIF    NB,SBIA$K_LEN,  SBIA$K_LEN:
                00002259                .RESTORE
    41 49 42 53 00' 2259                .ASCIC  "SBIA"  ; SBIA name
                04  2259
                32  225E                .BYTE   SBIA$K_LEN              ; SBIA$K_LEN of resultant P-table
                02  225F                .BYTE   2               ; Maximum unit number                           ;G:65
              0000  2260                .WORD   ^A""    ; Two character mnemonic for QIO SBIA
                80  2262                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                8D  2263                .Byte   Pd$_Name        ; Directive op-code
                01  2264                .Byte   Ptd$M_Unit              ; Enforced SI codes
          49 53 00' 2265                .Ascic  "SI"    ; Enforced device name prefix
                02  2265
                86  2268                .BYTE   Pd$_Literal     ; Indicate literal
          60000000  2269                .LONG   ^X60000000              ; Constant
                88  226D                .BYTE   Pd$_Store       ; Indicate store operation
              001C  226E                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                00  2270                .BYTE   0               ; 0 HP$A_DVA to data
                20  2271                .BYTE   32              ; 32 of data field
                86  2272                .BYTE   Pd$_Literal     ; Indicate literal
          60080000  2273                .LONG   ^X60080000              ; Constant
                88  2277                .BYTE   Pd$_Store       ; Indicate store operation
              0018  2278                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                00  227A                .BYTE   0               ; 0 HP$A_DEVICE to data
                20  227B                .BYTE   32              ; 32 of data field
                86  227C                .BYTE   Pd$_Literal     ; Indicate literal
          00000000  227D                .LONG   ^X000           ; Constant
                88  2281                .BYTE   Pd$_Store       ; Indicate store operation
              0024  2282                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                00  2284                .BYTE   0               ; 0 HP$W_VECTOR to data
                10  2285                .BYTE   16              ; 16 of data field
                87  2286                .BYTE   Pd$_Fetch       ; Indicate fetch operation
              000B  2287                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                00  2289                .BYTE   0               ; 0 HP$B_DRIVE to data
                08  228A                .BYTE   8               ; 8 of data field
                88  228B                .BYTE   Pd$_Store       ; Indicate store operation
              0018  228C                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                19  228E                .BYTE   25              ; 25 HP$A_DEVICE to data
                03  228F                .BYTE   3               ; 3 of data field
                88  2290                .BYTE   Pd$_Store       ; Indicate store operation
              001C  2291                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                19  2293                .BYTE   25              ; 25 HP$A_DVA to data
                03  2294                .BYTE   3               ; 3 of data field
                8A  2295                .BYTE   Pd$_Add         ; Indicate add value to field operation
              0024  2296                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                09  2298                .BYTE   9               ; 9 HP$W_VECTOR to data
                07  2299                .BYTE   7               ; 7 of data field
                81  229A                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
```

```
                        229B   365 TE16:   $DS_TE16
                        229B               .SAVE   LOCAL_BLOCK
                        229B               .PSECT  $ABS$,ABS
                00000032  0050             .=HP$A_DEPENDENT
                          0032             .IIF    NB,HP$K_TE16_LEN,       HP$K_TE16_LEN:
                          0032             .IIF    NB,TE16$K_LEN,  TE16$K_LEN:
                    0000229B               .RESTORE
          36 31 45 54 00' 229B             .ASCIC  "TE16"  ; TE16 name
                      04  229B
                      32  22A0             .BYTE   TE16$K_LEN              ; TE16$K_LEN of resultant P-table
                      08  22A1             .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  22A2             .WORD   ^A""    ; Two character mnemonic for QIO TE16
                      80  22A4             .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  22A5             .Byte   Pd$_Name        ; Directive op-code
                      1B  22A6             .Byte   Ptd$M_EndDevice         ; Enforced MT codes
                54 4D 00' 22A7             .Ascic  "MT"    ; Enforced device name prefix
                      02  22A7
                      86  22AA             .BYTE   Pd$_Literal     ; Indicate literal
                00000001  22AB             .LONG   HP$M_ALLOC              ; Constant
                      88  22AF             .BYTE   Pd$_Store       ; Indicate store operation
                    000A  22B0             .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  22B2             .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  22B3             .BYTE   8                       ; 8 of data field
                      81  22B4             .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        22B5   366 TK50:    $DS_TK50                        ;[25]                                   ;G:4
                        22B5               .SAVE   LOCAL_BLOCK
                        22B5               .PSECT  $ABS$,ABS
                00000032  0050             .=HP$A_DEPENDENT
                          0032             .IIF    NB,HP$K_TK50_LEN,       HP$K_TK50_LEN:
                    000022B5               .RESTORE
          30 35 4B 54 00' 22B5             .ASCIC  "TK50"  ; TK50 name
                      04  22B5
                      32  22BA             .BYTE   HP$K_TK50_LEN           ; HP$K_TK50_LEN of resultant P-table
                      10  22BB             .BYTE   16                      ; Maximum unit number                   ;G:65
                    554D  22BC             .WORD   ^A"MU"  ; Two character mnemonic for QIO TK50 MU
                      80  22BE             .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  22BF             .Byte   Pd$_Name        ; Directive op-code
                      03  22C0             .Byte   Ptd$M_Device            ; Enforced MU codes
                55 4D 00' 22C1             .Ascic  "MU"    ; Enforced device name prefix
                      02  22C1
                      86  22C4             .BYTE   Pd$_Literal     ; Indicate literal
                00000001  22C5             .LONG   HP$M_ALLOC              ; Constant
                      88  22C9             .BYTE   Pd$_Store       ; Indicate store operation
                    000A  22CA             .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  22CC             .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  22CD             .BYTE   8                       ; 8 of data field
                      81  22CE             .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        22CF   367 TM03:    $DS_TM03                        ;
                        22CF               .SAVE   LOCAL_BLOCK
                        22CF               .PSECT  $ABS$,ABS
                00000032  0050             .=HP$A_DEPENDENT
                          0032             .IIF    NB,HP$B_TM03_DRIVE,     HP$B_TM03_DRIVE:
                          0032             .IIF    NB,TM03$B_DRIVE,        TM03$B_DRIVE:
                00000033  0032             .IIF    NB,,.BLKB,      .BLKB   1
                          0033             .IIF    NB,HP$K_TM03_LEN,       HP$K_TM03_LEN:
                          0033             .IIF    NB,TM03$K_LEN,  TM03$K_LEN:
                    000022CF               .RESTORE
```

```
                33 30 4D 54 00' 22CF                .ASCIC  "TM03"  ; TM03 name
                            04  22CF
                            33  22D4                .BYTE   TM03$K_LEN              ; TM03$K_LEN of resultant P-table
                            00  22D5                .BYTE   0                       ; Maximum unit number                   ;G:65
                          4D54  22D6                .WORD   ^A"TM"  ; Two character mnemonic for QIO TM03 TM
                            80  22D8                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                            8D  22D9                .Byte   Pd$_Name        ; Directive op-code
                            02  22DA                .Byte   Ptd$M_Controller                ; Enforced MT codes
                     54 4D 00'  22DB                .Ascic  "MT"    ; Enforced device name prefix
                            02  22DB
                            82  22DE                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                45 56 49 52 44 00' 22DF             .ASCIC  "DRIVE" ; DRIVE string
                            05  22DF
                00000007 00000000  22E5             .LONG   0,7     ; 0 , 7 limits on input
                            88  22ED                .BYTE   Pd$_Store       ; Indicate store operation
                          0032  22EE                .WORD   TM03$B_DRIVE            ; Byte TM03$B_DRIVE to data
                            00  22F0                .BYTE   0                       ; 0 TM03$B_DRIVE to data
                            08  22F1                .BYTE   8                       ; 8 of data field
                            88  22F2                .BYTE   Pd$_Store       ; Indicate store operation
                          000B  22F3                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                            00  22F5                .BYTE   0                       ; 0 HP$B_DRIVE to data
                            08  22F6                .BYTE   8                       ; 8 of data field
                            88  22F7                .BYTE   Pd$_Store       ; Indicate store operation
                          0018  22F8                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                            07  22FA                .BYTE   7                       ; 7 HP$A_DEVICE to data
                            03  22FB                .BYTE   3                       ; 3 of data field
                            81  22FC                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                22FD    368 TM78:   $DS_TM78                        ;[01]                                   ;G:4
                                22FD                .SAVE   LOCAL_BLOCK
                                22FD                .PSECT  $ABS$,ABS
                        00000032  0050              .=HP$A_DEPENDENT
                                  0032              .IIF    NB,HP$B_TM78_DRIVE,     HP$B_TM78_DRIVE:
                                  0032              .IIF    NB,TM78$B_DRIVE,        TM78$B_DRIVE:
                        00000033  0032              .IIF    NB,.BLKB,       .BLKB   1
                                  0033              .IIF    NB,HP$K_TM78_LEN,       HP$K_TM78_LEN:
                                  0033              .IIF    NB,TM78$K_LEN,  TM78$K_LEN:
                               000022FD             .RESTORE
                   38 37 4D 54 00' 22FD             .ASCIC  "TM78"  ; TM78 name
                               04  22FD
                               33  2302             .BYTE   TM78$K_LEN              ; TM78$K_LEN of resultant P-table
                               00  2303             .BYTE   0                       ; Maximum unit number                   ;G:65
                             4654  2304             .WORD   ^A"TF"  ; Two character mnemonic for QIO TM78 TF
                               80  2306             .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                               8D  2307             .Byte   Pd$_Name        ; Directive op-code
                               02  2308             .Byte   Ptd$M_Controller                ; Enforced MF codes
                            46 4D 00' 2309          .Ascic  "MF"    ; Enforced device name prefix
                               02  2309
                               82  230C             .BYTE   Pd$_Decimal     ; Indicate scan decimal
                   45 56 49 52 44 00' 230D          .ASCIC  "DRIVE" ; DRIVE string
                               05  230D
                   00000007 00000000  2313          .LONG   0,7     ; 0 , 7 limits on input
                               88  231B             .BYTE   Pd$_Store       ; Indicate store operation
                             0032  231C             .WORD   TM78$B_DRIVE            ; Byte TM78$B_DRIVE to data
                               00  231E             .BYTE   0                       ; 0 TM78$B_DRIVE to data
                               08  231F             .BYTE   8                       ; 8 of data field
                               88  2320             .BYTE   Pd$_Store       ; Indicate store operation
                             000B  2321             .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
```

```
                      00  2323                .BYTE   0                       ; 0 HP$B_DRIVE to data
                      08  2324                .BYTE   8                       ; 8 of data field
                      88  2325                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  2326                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      07  2328                .BYTE   7                       ; 7 HP$A_DEVICE to data
                      03  2329                .BYTE   3                       ; 3 of data field
                      81  232A                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          232B    369 TS04:     $DS_TS04
                          232B                .SAVE   LOCAL_BLOCK
                          232B                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_TS04_CSR,       HP$L_TS04_CSR:
                          0032                .IIF    NB,TS04$L_CSR,  TS04$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_TS04_BR,        HP$B_TS04_BR:
                          0036                .IIF    NB,TS04$B_BR,   TS04$B_BR:
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_TS04_LEN,       HP$K_TS04_LEN:
                          0037                .IIF    NB,TS04$K_LEN,  TS04$K_LEN:
                  0000232B                    .RESTORE
          34 30 53 54 00' 232B                .ASCIC  "TS04"  ; TS04 name
                      04  232B
                      37  2330                .BYTE   TS04$K_LEN                      ; TS04$K_LEN of resultant P-table
                      01  2331                .BYTE   1                       ; Maximum unit number                   ;G:65
                    5354  2332                .WORD   ^A"TS"  ; Two character mnemonic for QIO TS04 TS
                      80  2334                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  2335                .Byte   Pd$_Name                ; Directive op-code
                      03  2336                .Byte   Ptd$M_Device                    ; Enforced MS codes
                53 4D 00' 2337                .Ascic  "MS"    ; Enforced device name prefix
                      02  2337
                      86  233A                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  233B                .LONG   HP$M_ALLOC                      ; Constant
                      88  233F                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  2340                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  2342                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2343                .BYTE   8                       ; 8 of data field
                      83  2344                .BYTE   Pd$_Octal               ; Indicate scan octal
             52 53 43 00' 2345                .ASCIC  "CSR"   ; CSR string
                      03  2345
            0003FFFE 0003E000  2349           .LONG   ^O760000,^O777776               ; 760000 , 777776 limits on input
                      88  2351                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  2352                .WORD   TS04$L_CSR                      ; Byte TS04$L_CSR to data
                      00  2354                .BYTE   0                       ; 0 TS04$L_CSR to data
                      20  2355                .BYTE   32                      ; 32 of data field
                      88  2356                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  2357                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  2359                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  235A                .BYTE   18                      ; 18 of data field
                      83  235B                .BYTE   Pd$_Octal               ; Indicate scan octal
   52 4F 54 43 45 56 00'  235C                .ASCIC  "VECTOR"                ; VECTOR string
                      06  235C
            000001FE 00000002  2363           .LONG   ^O2,^O776               ; 2 , 776 limits on input
                      88  236B                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  236C                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  236E                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  236F                .BYTE   9                       ; 9 of data field
                      82  2370                .BYTE   Pd$_Decimal             ; Indicate scan decimal
```

```
                    52 42 00'  2371              .ASCIC  "BR"    ; BR string
                       02      2371
              00000007 00000004  2374              .LONG   4,7     ; 4 , 7 limits on input
                       88      237C              .BYTE   Pd$_Store       ; Indicate store operation
                     0036      237D              .WORD   TS04$B_BR               ; Byte TS04$B_BR to data
                       00      237F              .BYTE   0               ; 0 TS04$B_BR to data
                       08      2380              .BYTE   8               ; 8 of data field
                       81      2381              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                               2382   370 TS05:   $DS_TS05                       ;[18]                   ;G:4
                               2382              .SAVE   LOCAL_BLOCK
                               2382              .PSECT  $ABS$,ABS
                    00000032   0050              .=HP$A_DEPENDENT
                               0032              .IIF    NB,TS05$B_BR,   TS05$B_BR:
                    00000033   0032              .IIF    NB,.BLKB,       .BLKB   1
                               0033              .IIF    NB,HP$K_TS05_LEN,       HP$K_TS05_LEN:
                               0033              .IIF    NB,TS05$K_LEN,  TS05$K_LEN:
                           00002382              .RESTORE
                 35 30 53 54 00'  2382              .ASCIC  "TS05"  ; TS05 name
                       04      2382
                       33      2387              .BYTE   TS05$K_LEN              ; TS05$K_LEN of resultant P-table
                       08      2388              .BYTE   8               ; Maximum unit number           ;G:65
                     0000      2389              .WORD   ^A""    ; Two character mnemonic for QIO TS05
                       80      238B              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D      238C              .Byte   Pd$_Name        ; Directive op-code
                       03      238D              .Byte   Ptd$M_Device            ; Enforced MS codes
                    53 4D 00'  238E              .Ascic  "MS"    ; Enforced device name prefix
                       02      238E
                       86      2391              .BYTE   Pd$_Literal     ; Indicate literal
                    00000001   2392              .LONG   HP$M_ALLOC              ; Constant
                       88      2396              .BYTE   Pd$_Store       ; Indicate store operation
                     000A      2397              .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                       00      2399              .BYTE   0               ; 0 HP$B_FLAGS to data
                       08      239A              .BYTE   8               ; 8 of data field
                       83      239B              .BYTE   Pd$_Octal       ; Indicate scan octal
44 41 5F 52 53 43 5F 35 30 53 54 00'  239C              .ASCIC  "TS05_CSR_ADDRESS"      ; TS05_CSR_ADDRESS string
                 53 53 45 52 44  23A8
                       10      239C
                 0003FFFE 0003E000  23AD              .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                       88      23B5              .BYTE   Pd$_Store       ; Indicate store operation
                     0018      23B6              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00      23B8              .BYTE   0               ; 0 HP$A_DEVICE to data
                       12      23B9              .BYTE   18              ; 18 of data field
                       83      23BA              .BYTE   Pd$_Octal       ; Indicate scan octal
52 4F 54 43 45 56 5F 35 30 53 54 00'  23BB              .ASCIC  "TS05_VECTOR"   ; TS05_VECTOR string
                       0B      23BB
                 000001FE 00000002  23C7              .LONG   ^O2,^O776       ; 2 , 776 limits on input
                       88      23CF              .BYTE   Pd$_Store       ; Indicate store operation
                     0024      23D0              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       00      23D2              .BYTE   0               ; 0 HP$W_VECTOR to data
                       10      23D3              .BYTE   16              ; 16 of data field
                       82      23D4              .BYTE   Pd$_Decimal     ; Indicate scan decimal
        4C 45 56 45 4C 5F 52 42 00'  23D5              .ASCIC  "BR_LEVEL"      ; BR_LEVEL string
                       08      23D5
                 00000007 00000004  23DE              .LONG   4,7     ; 4 , 7 limits on input
                       88      23E6              .BYTE   Pd$_Store       ; Indicate store operation
                     0032      23E7              .WORD   TS05$B_BR               ; Byte TS05$B_BR to data
                       00      23E9              .BYTE   0               ; 0 TS05$B_BR to data
```

```
                      08  23EA                .BYTE   8                       ; 8 of data field
                      81  23EB                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          23EC    371 TS11:    $DS_TS11
                          23EC                .SAVE   LOCAL_BLOCK
                          23EC                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_TS11_CSR,       HP$L_TS11_CSR:
                          0032                .IIF    NB,TS11$L_CSR,  TS11$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_TS11_BR,        HP$B_TS11_BR:
                          0036                .IIF    NB,TS11$B_BR,   TS11$B_BR:
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_TS11_LEN,       HP$K_TS11_LEN:
                          0037                .IIF    NB,TS11$K_LEN,  TS11$K_LEN:
                      000023EC                .RESTORE
             31 31 53 54 00' 23EC             .ASCIC  "TS11"  ; TS11 name
                      04  23EC
                      37  23F1                .BYTE   TS11$K_LEN              ; TS11$K_LEN of resultant P-table
                      01  23F2                .BYTE   1                       ; Maximum unit number                   ;G:65
                    5354  23F3                .WORD   ^A"TS"  ; Two character mnemonic for QIO TS11 TS
                      80  23F5                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  23F6                .Byte   Pd$_Name                ; Directive op-code
                      03  23F7                .Byte   Ptd$M_Device            ; Enforced MS codes
                53 4D 00' 23F8                .Asic   "MS"    ; Enforced device name prefix
                      02  23F8
                      86  23FB                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  23FC                .LONG   HP$M_ALLOC              ; Constant
                      88  2400                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  2401                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  2403                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2404                .BYTE   8                       ; 8 of data field
                      83  2405                .BYTE   Pd$_Octal               ; Indicate scan octal
             52 53 43 00' 2406                .ASCIC  "CSR"   ; CSR string
                      03  2406
            0003FFFE 0003E000  240A            .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  2412                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  2413                .WORD   TS11$L_CSR              ; Byte TS11$L_CSR to data
                      00  2415                .BYTE   0                       ; 0 TS11$L_CSR to data
                      20  2416                .BYTE   32                      ; 32 of data field
                      88  2417                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  2418                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  241A                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  241B                .BYTE   18                      ; 18 of data field
                      83  241C                .BYTE   Pd$_Octal               ; Indicate scan octal
   52 4F 54 43 45 56 00' 241D                .ASCIC  "VECTOR"                ; VECTOR string
                      06  241D
            000001FE 00000002  2424            .LONG   ^O2,^O776               ; 2 , 776 limits on input
                      88  242C                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  242D                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  242F                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  2430                .BYTE   9                       ; 9 of data field
                      82  2431                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                52 42 00' 2432                .ASCIC  "BR"    ; BR string
                      02  2432
            00000007 00000004  2435            .LONG   4,7     ; 4 , 7 limits on input
                      88  243D                .BYTE   Pd$_Store               ; Indicate store operation
                    0036  243E                .WORD   TS11$B_BR               ; Byte TS11$B_BR to data
```

```
                 00  2440                .BYTE   0                   ; 0 TS11$B_BR to data
                 08  2441                .BYTE   8                   ; 8 of data field
                 81  2442                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                     2443   372 TU45:    $DS_TU45
                     2443                .SAVE   LOCAL_BLOCK
                     2443                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                     0032                .IIF    NB,HP$K_TU45_LEN,       HP$K_TU45_LEN:
                     0032                .IIF    NB,TU45$K_LEN,  TU45$K_LEN:
                 0000 2443                .RESTORE
     35 34 55 54 00' 2443                .ASCIC  "TU45"  ; TU45 name
                 04  2443
                 32  2448                .BYTE   TU45$K_LEN                  ; TU45$K_LEN of resultant P-table
                 08  2449                .BYTE   8                   ; Maximum unit number                       ;G:65
               0000  244A                .WORD   ^A""    ; Two character mnemonic for QIO TU45
                 80  244C                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                 8D  244D                .Byte   Pd$_Name            ; Directive op-code
                 1B  244E                .Byte   Ptd$M_EndDevice             ; Enforced MT codes
          54 4D 00' 244F                .Ascic  "MT"    ; Enforced device name prefix
                 02  244F
                 86  2452                .BYTE   Pd$_Literal         ; Indicate literal
          00000001  2453                .LONG   HP$M_ALLOC                  ; Constant
                 88  2457                .BYTE   Pd$_Store           ; Indicate store operation
               000A  2458                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                 00  245A                .BYTE   0                   ; 0 HP$B_FLAGS to data
                 08  245B                .BYTE   8                   ; 8 of data field
                 81  245C                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                     245D   373 TU58:    $DS_TU58
                     245D                .SAVE   LOCAL_BLOCK
                     245D                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                     0032                .IIF    NB,HP$K_TU58_LEN,       HP$K_TU58_LEN:
                     0032                .IIF    NB,TU58$K_LEN,  TU58$K_LEN:
                 0000 245D                .RESTORE
     38 35 55 54 00' 245D                .ASCIC  "TU58"  ; TU58 name
                 04  245D
                 32  2462                .BYTE   TU58$K_LEN                  ; TU58$K_LEN of resultant P-table
                 02  2463                .BYTE   2                   ; Maximum unit number                       ;G:65
               4444  2464                .WORD   ^A"DD"  ; Two character mnemonic for QIO TU58 DD
                 80  2466                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                 8D  2467                .Byte   Pd$_Name            ; Directive op-code
                 03  2468                .Byte   Ptd$M_Device                ; Enforced DD codes
          44 44 00' 2469                .Ascic  "DD"    ; Enforced device name prefix
                 02  2469
                 86  246C                .BYTE   Pd$_Literal         ; Indicate literal
          00000001  246D                .LONG   HP$M_ALLOC                  ; Constant
                 88  2471                .BYTE   Pd$_Store           ; Indicate store operation
               000A  2472                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                 00  2474                .BYTE   0                   ; 0 HP$B_FLAGS to data
                 08  2475                .BYTE   8                   ; 8 of data field
                 81  2476                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                     2477   374 TU70:    $DS_TU70                            ;[30]                               ;G:4
                     2477                .SAVE   LOCAL_BLOCK
                     2477                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                     0032                .IIF    NB,HP$K_TU70_LEN,       HP$K_TU70_LEN::
                     0032                .IIF    NB,TU70$K_LEN,  TU70$K_LEN::
```

```
                    00002477                    .RESTORE
          30 37 55 54 00' 2477                  .ASCIC  "TU70"  ; TU70 name
                      04  2477
                      32  247C                  .BYTE   TU70$K_LEN              ; TU70$K_LEN of resultant P-table
                      10  247D                  .BYTE   16              ; Maximum unit number                   ;G:65
                    0000  247E                  .WORD   ^A""    ; Two character mnemonic for QIO TU70
                      80  2480                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2481                  .Byte   Pd$_Name        ; Directive op-code
                      1B  2482                  .Byte   PTD$M_ENDDEVICE         ; Enforced MR codes
                   52 4D 00' 2483               .Ascic  "MR"    ; Enforced device name prefix
                      02  2483
                      86  2486                  .BYTE   Pd$_Literal     ; Indicate literal
                00000001  2487                  .LONG   HP$M_ALLOC              ; Constant
                      88  248B                  .BYTE   Pd$_Store       ; Indicate store operation
                    000A  248C                  .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  248E                  .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  248F                  .BYTE   8               ; 8 of data field
                      81  2490                  .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2491    375 TU72:     $DS_TU72                        ;[30]                   ;G:4
                          2491                  .SAVE   LOCAL_BLOCK
                          2491                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,HP$K_TU72_LEN,       HP$K_TU72_LEN::
                          0032                  .IIF    NB,TU72$K_LEN,  TU72$K_LEN::
                    00002491                    .RESTORE
          32 37 55 54 00' 2491                  .ASCIC  "TU72"  ; TU72 name
                      04  2491
                      32  2496                  .BYTE   TU72$K_LEN              ; TU72$K_LEN of resultant P-table
                      10  2497                  .BYTE   16              ; Maximum unit number                   ;G:65
                    0000  2498                  .WORD   ^A""    ; Two character mnemonic for QIO TU72
                      80  249A                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  249B                  .Byte   Pd$_Name        ; Directive op-code
                      1B  249C                  .Byte   PTD$M_ENDDEVICE         ; Enforced MR codes
                   52 4D 00' 249D               .Ascic  "MR"    ; Enforced device name prefix
                      02  249D
                      86  24A0                  .BYTE   Pd$_Literal     ; Indicate literal
                00000001  24A1                  .LONG   HP$M_ALLOC              ; Constant
                      88  24A5                  .BYTE   Pd$_Store       ; Indicate store operation
                    000A  24A6                  .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  24A8                  .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  24A9                  .BYTE   8               ; 8 of data field
                      81  24AA                  .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          24AB    376 TU77:     $DS_TU77
                          24AB                  .SAVE   LOCAL_BLOCK
                          24AB                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,HP$K_TU77_LEN,       HP$K_TU77_LEN:
                          0032                  .IIF    NB,TU77$K_LEN,  TU77$K_LEN:
                    000024AB                    .RESTORE
          37 37 55 54 00' 24AB                  .ASCIC  "TU77"  ; TU77 name
                      04  24AB
                      32  24B0                  .BYTE   TU77$K_LEN              ; TU77$K_LEN of resultant P-table
                      08  24B1                  .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  24B2                  .WORD   ^A""    ; Two character mnemonic for QIO TU77
                      80  24B4                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  24B5                  .Byte   Pd$_Name        ; Directive op-code
                      1B  24B6                  .Byte   Ptd$M_EndDevice         ; Enforced MT codes
```

```
                    54 4D 00' 24B7                  .Ascic  "MT"    ; Enforced device name prefix
                          02  24B7
                          86  24BA                  .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  24BB                  .LONG   HP$M_ALLOC              ; Constant
                          88  24BF                  .BYTE   Pd$_Store       ; Indicate store operation
                        000A  24C0                  .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  24C2                  .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  24C3                  .BYTE   8               ; 8 of data field
                          81  24C4                  .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              24C5    377 TU78:     $DS_TU78                        ;[01]                   ;G:4
                              24C5                  .SAVE   LOCAL_BLOCK
                              24C5                  .PSECT  $ABS$,ABS
                    00000032  0050                  .=HP$A_DEPENDENT
                              0032                  .IIF    NB,HP$K_TU78_LEN,       HP$K_TU78_LEN:
                              0032                  .IIF    NB,TU78$K_LEN,  TU78$K_LEN:
                              000024C5              .RESTORE
                 38 37 55 54 00' 24C5               .ASCIC  "TU78"  ; TU78 name
                          04  24C5
                          32  24CA                  .BYTE   TU78$K_LEN              ; TU78$K_LEN of resultant P-table
                          08  24CB                  .BYTE   8               ; Maximum unit number           ;G:65
                        0000  24CC                  .WORD   ^A""    ; Two character mnemonic for QIO TU78
                          80  24CE                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  24CF                  .Byte   Pd$_Name        ; Directive op-code
                          1B  24D0                  .Byte   Ptd$M_EndDevice         ; Enforced MF codes
                    46 4D 00' 24D1                  .Ascic  "MF"    ; Enforced device name prefix
                          02  24D1
                          86  24D4                  .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  24D5                  .LONG   HP$M_ALLOC              ; Constant
                          88  24D9                  .BYTE   Pd$_Store       ; Indicate store operation
                        000A  24DA                  .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  24DC                  .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  24DD                  .BYTE   8               ; 8 of data field
                          81  24DE                  .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              24DF    378 TU80:     $DS_TU80                        ;[08]                   ;G:4
                              24DF                  .SAVE   LOCAL_BLOCK
                              24DF                  .PSECT  $ABS$,ABS
                    00000032  0050                  .=HP$A_DEPENDENT
                              0032                  .IIF    NB,HP$L_TU80_CSR,       HP$L_TU80_CSR:
                    00000036  0032                  .IIF    NB,.BLKL,       .BLKL   1
                              0036                  .IIF    NB,TU80$B_BR,   TU80$B_BR:
                              0036                  .IIF    NB,HP$B_TU80_BR,        HP$B_TU80_BR:
                    00000037  0036                  .IIF    NB,.BLKB,       .BLKB   1
                              0037                  .IIF    NB,HP$K_TU80_LEN,       HP$K_TU80_LEN:
                              0037                  .IIF    NB,TU80$K_LEN,  TU80$K_LEN:
                              000024DF              .RESTORE
                 30 38 55 54 00' 24DF               .ASCIC  "TU80"  ; TU80 name
                          04  24DF
                          37  24E4                  .BYTE   TU80$K_LEN              ; TU80$K_LEN of resultant P-table
                          08  24E5                  .BYTE   8               ; Maximum unit number           ;G:65
                        0000  24E6                  .WORD   ^A""    ; Two character mnemonic for QIO TU80
                          80  24E8                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  24E9                  .Byte   Pd$_Name        ; Directive op-code
                          03  24EA                  .Byte   Ptd$M_Device            ; Enforced MS codes
                    53 4D 00' 24EB                  .Ascic  "MS"    ; Enforced device name prefix
                          02  24EB
                          86  24EE                  .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  24EF                  .LONG   HP$M_ALLOC              ; Constant
```

```
                          88  24F3                .BYTE   Pd$_Store       ; Indicate store operation
                        000A  24F4                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  24F6                .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  24F7                .BYTE   8               ; 8 of data field
                          83  24F8                .BYTE   Pd$_Octal       ; Indicate scan octal
44 41 5F 52 53 43 5F 30 38 55 54 00' 24F9        .ASCIC  "TU80_CSR_ADDRESS"      ; TU80_CSR_ADDRESS string
                53 53 45 52 44  2505
                          10  24F9
                0003FFFE 0003E000  250A          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  2512                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  2513                .WORD   HP$L_TU80_CSR            ; Byte HP$L_TU80_CSR to data
                          00  2515                .BYTE   0               ; 0 HP$L_TU80_CSR to data
                          20  2516                .BYTE   32              ; 32 of data field
                          88  2517                .BYTE   Pd$_Store       ; Indicate store operation
                        0018  2518                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  251A                .BYTE   0               ; 0 HP$A_DEVICE to data
                          12  251B                .BYTE   18              ; 18 of data field
                          83  251C                .BYTE   Pd$_Octal       ; Indicate scan octal
52 4F 54 43 45 56 5F 30 38 55 54 00' 251D        .ASCIC  "TU80_VECTOR"   ; TU80_VECTOR string
                          0B  251D
                000001FE 00000002  2529          .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  2531                .BYTE   Pd$_Store       ; Indicate store operation
                        0024  2532                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  2534                .BYTE   0               ; 0 HP$W_VECTOR to data
                          10  2535                .BYTE   16              ; 16 of data field
                          82  2536                .BYTE   Pd$_Decimal     ; Indicate scan decimal
        4C 45 56 45 4C 5F 52 42 00' 2537         .ASCIC  "BR_LEVEL"      ; BR_LEVEL string
                          08  2537
                00000007 00000004  2540          .LONG   4,7     ; 4 , 7 limits on input
                          88  2548                .BYTE   Pd$_Store       ; Indicate store operation
                        0036  2549                .WORD   HP$B_TU80_BR             ; Byte HP$B_TU80_BR to data
                          00  254B                .BYTE   0               ; 0 HP$B_TU80_BR to data
                          08  254C                .BYTE   8               ; 8 of data field
                          81  254D                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              254E  379 TU81:     $DS_TU81                        ;[09]                           ;G:4
                              254E                .SAVE   LOCAL_BLOCK
                              254E                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$L_TU81_CSR,       HP$L_TU81_CSR:
                              0032                .IIF    NB,TU81$L_CSR,  TU81$L_CSR:
                    00000036  0032                .IIF    NB,,.BLKL       .BLKL   1
                              0036                .IIF    NB,HP$B_TU81_BR,        HP$B_TU81_BR:
                              0036                .IIF    NB,TU81$B_BR,   TU81$B_BR:
                    00000037  0036                .IIF    NB,,.BLKB       .BLKB   1
                              0037                .IIF    NB,HP$K_TU81_LEN,       HP$K_TU81_LEN:
                              0037                .IIF    NB,TU81$K_LEN,  TU81$K_LEN:
                          0000254E                .RESTORE
                  31 38 55 54 00' 254E            .ASCIC  "TU81"  ; TU81 name
                          04  254E
                          37  2553                .BYTE   TU81$K_LEN              ; TU81$K_LEN of resultant P-table
                          04  2554                .BYTE   4               ; Maximum unit number                   ;G:65
                        554D  2555                .WORD   ^A"MU"  ; Two character mnemonic for QIO TU81 MU
                          80  2557                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  2558                .Byte   Pd$_Name        ; Directive op-code
                          03  2559                .Byte   Ptd$M_Device            ; Enforced MU codes
                    55 4D 00' 255A                .Ascic  "MU"    ; Enforced device name prefix
                          02  255A
```

```
                      86  255D                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  255E                .LONG   HP$M_ALLOC               ; Constant
                      88  2562                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2563                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  2565                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  2566                .BYTE   8                ; 8 of data field
                      83  2567                .BYTE   Pd$_Octal        ; Indicate scan octal
             52 53 43 00' 2568                .ASCIC  "CSR"    ; CSR string
                      03  2568
          0003FFFE 0003E000  256C                .LONG   ^O760000,^O777776        ; 760000 , 777776 limits on input
                      88  2574                .BYTE   Pd$_Store        ; Indicate store operation
                    0032  2575                .WORD   TU81$L_CSR               ; Byte TU81$L_CSR to data
                      00  2577                .BYTE   0                ; 0 TU81$L_CSR to data
                      20  2578                .BYTE   32               ; 32 of data field
                      88  2579                .BYTE   Pd$_Store        ; Indicate store operation
                    0018  257A                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                      00  257C                .BYTE   0                ; 0 HP$A_DEVICE to data
                      12  257D                .BYTE   18               ; 18 of data field
                      83  257E                .BYTE   Pd$_Octal        ; Indicate scan octal
       52 4F 54 43 45 56 00' 257F                .ASCIC  "VECTOR"         ; VECTOR string
                      06  257F
          000001FE 00000002  2586                .LONG   ^O2,^O776        ; 2 , 776 limits on input
                      88  258E                .BYTE   Pd$_Store        ; Indicate store operation
                    0024  258F                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                      00  2591                .BYTE   0                ; 0 HP$W_VECTOR to data
                      09  2592                .BYTE   9                ; 9 of data field
                      82  2593                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                52 42 00' 2594                .ASCIC  "BR"     ; BR string
                      02  2594
          00000007 00000004  2597                .LONG   4,7      ; 4 , 7 limits on input
                      88  259F                .BYTE   Pd$_Store        ; Indicate store operation
                    0036  25A0                .WORD   TU81$B_BR                ; Byte TU81$B_BR to data
                      00  25A2                .BYTE   0                ; 0 TU81$B_BR to data
                      08  25A3                .BYTE   8                ; 8 of data field
                      81  25A4                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          25A5    380 UBE:    $DS_UBE
                          25A5                .SAVE   LOCAL_BLOCK
                          25A5                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_UBE_CSR,         HP$L_UBE_CSR:
                          0032                .IIF    NB,UBE$L_CSR,    UBE$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,        .BLKL   1
                          0036                .IIF    NB,HP$K_UBE_LEN,         HP$K_UBE_LEN:
                          0036                .IIF    NB,UBE$K_LEN,    UBE$K_LEN:
                    000025A5                .RESTORE
             45 42 55 00' 25A5                .ASCIC  "UBE"    ; UBE name
                      03  25A5
                      36  25A9                .BYTE   UBE$K_LEN                ; UBE$K_LEN of resultant P-table
                      08  25AA                .BYTE   8                ; Maximum unit number                          ;G:65
                    4555  25AB                .WORD   ^A"UE"   ; Two character mnemonic for QIO UBE UE
                      80  25AD                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  25AE                .Byte   Pd$_Name         ; Directive op-code
                      03  25AF                .Byte   Ptd$M_Device             ; Enforced UB codes
                42 55 00' 25B0                .Ascic  "UB"     ; Enforced device name prefix
                      02  25B0
                      83  25B3                .BYTE   Pd$_Octal        ; Indicate scan octal
             52 53 43 00' 25B4                .ASCIC  "CSR"    ; CSR string
```

```
                    03  25B4
     0003FFFE 0003E000  25B8          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                    88  25C0          .BYTE   Pd$_Store       ; Indicate store operation
                  0032  25C1          .WORD   UBE$L_CSR               ; Byte UBE$L_CSR to data
                    00  25C3          .BYTE   0               ; 0 UBE$L_CSR to data
                    20  25C4          .BYTE   32              ; 32 of data field
                    88  25C5          .BYTE   Pd$_Store       ; Indicate store operation
                  0018  25C6          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  25C8          .BYTE   0               ; 0 HP$A_DEVICE to data
                    12  25C9          .BYTE   18              ; 18 of data field
                    83  25CA          .BYTE   Pd$_Octal       ; Indicate scan octal
  52 4F 54 43 45 56 00' 25CB          .ASCIC  "VECTOR"        ; VECTOR string
                    06  25CB
     000001FE 00000002  25D2          .LONG   ^O2,^O776       ; 2 , 776 limits on input
                    88  25DA          .BYTE   Pd$_Store       ; Indicate store operation
                  0024  25DB          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  25DD          .BYTE   0               ; 0 HP$W_VECTOR to data
                    09  25DE          .BYTE   9               ; 9 of data field
                    81  25DF          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        25E0  381 UDA50:  $DS_UDA50
                        25E0          .SAVE   LOCAL_BLOCK
                        25E0          .PSECT  $ABS$,ABS
              00000032  0050          .=HP$A_DEPENDENT
                        0032          .IIF    NB,HP$L_UDA50_UDAIP,    HP$L_UDA50_UDAIP:
                        0032          .IIF    NB,UDA50$L_UDAIP,       UDA50$L_UDAIP:
              00000036  0032          .IIF    NB,.BLKL,       .BLKL   1
                        0036          .IIF    NB,HP$B_UDA50_BR,       HP$B_UDA50_BR:
                        0036          .IIF    NB,UDA50$B_BR,  UDA50$B_BR:
              00000037  0036          .IIF    NB,.BLKB,       .BLKB   1
                        0037          .IIF    NB,HP$B_UDA50_BURST,    HP$B_UDA50_BURST:
                        0037          .IIF    NB,UDA50$B_BURST,       UDA50$B_BURST:
              00000038  0037          .IIF    NB,.BLKB,       .BLKB   1
                        0038          .IIF    NB,HP$K_UDA50_LEN,      HP$K_UDA50_LEN:
                        0038          .IIF    NB,UDA50$K_LEN, UDA50$K_LEN:
                    000025E0          .RESTORE
     30 35 41 44 55 00' 25E0          .ASCIC  "UDA50" ; UDA50 name
                    05  25E0
                    38  25E6          .BYTE   HP$K_UDA50_LEN          ; HP$K_UDA50_LEN of resultant P-table
                    00  25E7          .BYTE   0               ; Maximum unit number                   ;G:65
                  0000  25E8          .WORD   ^A""    ; Two character mnemonic for QIO UDA50
                    80  25EA          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  25EB          .Byte   Pd$_Name        ; Directive op-code
                    02  25EC          .Byte   Ptd$M_Controller                ; Enforced DU codes
              55 44 00' 25ED          .Ascic  "DU"    ; Enforced device name prefix
                    02  25ED
                    83  25F0          .BYTE   Pd$_Octal       ; Indicate scan octal
     50 49 41 44 55 00' 25F1          .ASCIC  "UDAIP" ; UDAIP string
                    05  25F1
     0003FFFC 0003E000  25F7          .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                    88  25FF          .BYTE   Pd$_Store       ; Indicate store operation
                  0032  2600          .WORD   HP$L_UDA50_UDAIP                ; Byte HP$L_UDA50_UDAIP to data
                    00  2602          .BYTE   0               ; 0 HP$L_UDA50_UDAIP to data
                    20  2603          .BYTE   32              ; 32 of data field
                    88  2604          .BYTE   Pd$_Store       ; Indicate store operation
                  0018  2605          .WORD   Hp$A_Device             ; Byte Hp$A_Device to data
                    00  2607          .BYTE   0               ; 0 Hp$A_Device to data
                    12  2608          .BYTE   18              ; 18 of data field
```

```
                       83  2609                .BYTE   Pd$_Octal       ; Indicate scan octal
            72 6F 74 63 65 56 00' 260A         .ASCIC  "Vector"        ; Vector string
                       06  260A
               000001FC 00000004  2611         .LONG   ^O4,^O774       ; 4 , 774 limits on input
                       88  2619                .BYTE   Pd$_Store       ; Indicate store operation
                     0024  261A                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       00  261C                .BYTE   0               ; 0 HP$W_VECTOR to data
                       08  261D                .BYTE   8               ; 8 of data field
                       82  261E                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                  52 42 00' 261F                .ASCIC  "BR"    ; BR string
                       02  261F
               00000007 00000004  2622         .LONG   4,7     ; 4 , 7 limits on input
                       88  262A                .BYTE   Pd$_Store       ; Indicate store operation
                     0036  262B                .WORD   HP$B_UDA50_BR           ; Byte HP$B_UDA50_BR to data
                       00  262D                .BYTE   0               ; 0 HP$B_UDA50_BR to data
                       08  262E                .BYTE   8               ; 8 of data field
                       82  262F                .BYTE   Pd$_Decimal     ; Indicate scan decimal
   65 74 61 52 5F 74 73 72 75 42 00' 2630      .ASCIC  "Burst_Rate"    ; Burst_Rate string
                       0A  2630
               0000003F 00000000  263B         .LONG   0,63    ; 0 , 63 limits on input
                       88  2643                .BYTE   Pd$_Store       ; Indicate store operation
                     0037  2644                .WORD   HP$B_UDA50_BURST                ; Byte HP$B_UDA50_BURST to data
                       00  2646                .BYTE   0               ; 0 HP$B_UDA50_BURST to data
                       06  2647                .BYTE   6               ; 6 of data field
                       81  2648                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                           2649   382 UNA11:   $DS_UNA11                      ;[06]                          ;G:4
                           2649                .SAVE   LOCAL_BLOCK
                           2649                .PSECT  $ABS$,ABS
                  00000032 0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,UNA11$B_BR,  UNA11$B_BR:
                  00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                           0033                .IIF    NB,UNA11$L_CSR, UNA11$L_CSR:
                  00000037 0033                .IIF    NB,.BLKL,       .BLKL   1
                           0037                .IIF    NB,UNA11$K_LEN, UNA11$K_LEN:
                       00002649                .RESTORE
            31 31 41 4E 55 00' 2649             .ASCIC  "UNA11" ; UNA11 name
                       05  2649
                       37  264F                .BYTE   UNA11$K_LEN             ; UNA11$K_LEN of resultant P-table
                       08  2650                .BYTE   8               ; Maximum unit number                   ;G:65
                     0000  2651                .WORD   ^A""    ; Two character mnemonic for QIO UNA11
                       80  2653                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  2654                .Byte   Pd$_Name        ; Directive op-code
                       03  2655                .Byte   PTD$M_DEVICE            ; Enforced XE codes
                  45 58 00' 2656                .Ascic  "XE"    ; Enforced device name prefix
                       02  2656
                       83  2659                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 53 43 00' 265A             .ASCIC  "CSR"   ; CSR string
                       03  265A
               0003FFFE 0003E000  265E         .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                       88  2666                .BYTE   Pd$_Store       ; Indicate store operation
                     0033  2667                .WORD   UNA11$L_CSR             ; Byte UNA11$L_CSR to data
                       00  2669                .BYTE   0               ; 0 UNA11$L_CSR to data
                       20  266A                .BYTE   32              ; 32 of data field
                       88  266B                .BYTE   Pd$_Store       ; Indicate store operation
                     0018  266C                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  266E                .BYTE   0               ; 0 HP$A_DEVICE to data
                       12  266F                .BYTE   18              ; 18 of data field
```

```
                    83  2670                .BYTE   Pd$_Octal        ; Indicate scan octal
52 4F 54 43 45 56 00'  2671                .ASCIC  "VECTOR"         ; VECTOR string
                    06  2671
   000001FE 00000002   2678                .LONG   ^O2,^O776        ; 2 , 776 limits on input
                    88  2680                .BYTE   Pd$_Store        ; Indicate store operation
                  0024  2681                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                    00  2683                .BYTE   0                ; 0 HP$W_VECTOR to data
                    09  2684                .BYTE   9                ; 9 of data field
                    82  2685                .BYTE   Pd$_Decimal      ; Indicate scan decimal
             52 42 00'  2686                .ASCIC  "BR"     ; BR string
                    02  2686
   00000007 00000004   2689                .LONG   4,7      ; 4 , 7 limits on input
                    88  2691                .BYTE   Pd$_Store        ; Indicate store operation
                  0032  2692                .WORD   UNA11$B_BR               ; Byte UNA11$B_BR to data
                    00  2694                .BYTE   0                ; 0 UNA11$B_BR to data
                    08  2695                .BYTE   8                ; 8 of data field
                    81  2696                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                        2697   383 VS100:   $DS_VS100                        ;[10]                                  ;G:4
                        2697                .SAVE   LOCAL_BLOCK
                        2697                .PSECT  $ABS$,ABS
             00000032   0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$L_VS100_CSR,       HP$L_VS100_CSR:
                        0032                .IIF    NB,VS100$L_CSR, VS100$L_CSR:
             00000036   0032                .IIF    NB,.BLKL,        .BLKL   1
                        0036                .IIF    NB,HP$B_VS100_BR,        HP$B_VS100_BR:
                        0036                .IIF    NB,VS100$B_BR,  VS100$B_BR:
             00000037   0036                .IIF    NB,.BLKB,        .BLKB   1
                        0037                .IIF    NB,HP$K_VS100_LEN,       HP$K_VS100_LEN:
                        0037                .IIF    NB,VS100$K_LEN, VS100$K_LEN:
                    00002697                .RESTORE
   30 30 31 53 56 00'  2697                .ASCIC  "VS100" ; VS100 name
                    05  2697
                    37  269D                .BYTE   VS100$K_LEN              ; VS100$K_LEN of resultant P-table
                    01  269E                .BYTE   1                ; Maximum unit number                          ;G:65
                  5356  269F                .WORD   ^A"VS"   ; Two character mnemonic for QIO VS100 VS
                    80  26A1                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                    8D  26A2                .Byte   Pd$_Name         ; Directive op-code
                    1B  26A3                .Byte   Ptd$M_EndDevice          ; Enforced VB codes
             42 56 00'  26A4                .Ascic  "VB"     ; Enforced device name prefix
                    02  26A4
                    86  26A7                .BYTE   Pd$_Literal      ; Indicate literal
             00000001   26A8                .LONG   HP$M_ALLOC               ; Constant
                    88  26AC                .BYTE   Pd$_Store        ; Indicate store operation
                  000A  26AD                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                    00  26AF                .BYTE   0                ; 0 HP$B_FLAGS to data
                    08  26B0                .BYTE   8                ; 8 of data field
                    83  26B1                .BYTE   Pd$_Octal        ; Indicate scan octal
         52 53 43 00'  26B2                .ASCIC  "CSR"    ; CSR string
                    03  26B2
   0003FFFE 0003E000   26B6                .LONG   ^O760000,^O777776        ; 760000 , 777776 limits on input
                    88  26BE                .BYTE   Pd$_Store        ; Indicate store operation
                  0032  26BF                .WORD   VS100$L_CSR              ; Byte VS100$L_CSR to data
                    00  26C1                .BYTE   0                ; 0 VS100$L_CSR to data
                    20  26C2                .BYTE   32               ; 32 of data field
                    88  26C3                .BYTE   Pd$_Store        ; Indicate store operation
                  0018  26C4                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                    00  26C6                .BYTE   0                ; 0 HP$A_DEVICE to data
```

```
                      12  26C7                .BYTE   18                      ; 18 of data field
                      83  26C8                .BYTE   Pd$_Octal             ; Indicate scan octal
           52 4F 54 43 45 56 00' 26C9          .ASCIC  "VECTOR"              ; VECTOR string
                      06  26C9
      000001FE 00000002  26D0                  .LONG   ^O2,^O776             ; 2 , 776 limits on input
                      88  26D8                .BYTE   Pd$_Store             ; Indicate store operation
                    0024  26D9                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  26DB                .BYTE   0                     ; 0 HP$W_VECTOR to data
                      09  26DC                .BYTE   9                     ; 9 of data field
                      82  26DD                .BYTE   Pd$_Decimal           ; Indicate scan decimal
                 52 42 00' 26DE               .ASCIC  "BR"    ; BR string
                      02  26DE
      00000007 00000004  26E1                  .LONG   4,7     ; 4 , 7 limits on input
                      88  26E9                .BYTE   Pd$_Store             ; Indicate store operation
                    0036  26EA                .WORD   VS100$B_BR                      ; Byte VS100$B_BR to data
                      00  26EC                .BYTE   0                     ; 0 VS100$B_BR to data
                      08  26ED                .BYTE   8                     ; 8 of data field
                      81  26EE                .BYTE   Pd$_End               ; Indicate end of PT-descriptor
                          26EF   384 VS125:    $DS_VS125                               ;[17]                                   ;G:4
                          26EF                .SAVE   LOCAL_BLOCK
                          26EF                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_VS125_CSR,      HP$L_VS125_CSR:
                          0032                .IIF    NB,VS125$L_CSR, VS125$L_CSR:
                 00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_VS125_BR,       HP$B_VS125_BR:
                          0036                .IIF    NB,VS125$B_BR,  VS125$B_BR:
                 00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_VS125_LEN,      HP$K_VS125_LEN:
                          0037                .IIF    NB,VS125$K_LEN, VS125$K_LEN:
                        000026EF              .RESTORE
           35 32 31 53 56 00' 26EF            .ASCIC  "VS125" ; VS125 name
                      05  26EF
                      37  26F5                .BYTE   VS125$K_LEN                     ; VS125$K_LEN of resultant P-table
                      01  26F6                .BYTE   1                       ; Maximum unit number                   ;G:65
                    5356  26F7                .WORD   ^A"VS"  ; Two character mnemonic for QIO VS125 VS
                      80  26F9                .BYTE   Pd$_Start             ; Constant to identify start of descriptor
                      8D  26FA                .Byte   Pd$_Name              ; Directive op-code
                      1B  26FB                .Byte   Ptd$M_EndDevice               ; Enforced VB codes
                 42 56 00' 26FC               .Ascic  "VB"    ; Enforced device name prefix
                      02  26FC
                      86  26FF                .BYTE   Pd$_Literal           ; Indicate literal
                 00000001 2700                .LONG   HP$M_ALLOC                      ; Constant
                      88  2704                .BYTE   Pd$_Store             ; Indicate store operation
                    000A  2705                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  2707                .BYTE   0                     ; 0 HP$B_FLAGS to data
                      08  2708                .BYTE   8                     ; 8 of data field
                      83  2709                .BYTE   Pd$_Octal             ; Indicate scan octal
                 52 53 43 00' 270A            .ASCIC  "CSR"   ; CSR string
                      03  270A
      0003FFFE 0003E000  270E                  .LONG   ^O760000,^O777776             ; 760000 , 777776 limits on input
                      88  2716                .BYTE   Pd$_Store             ; Indicate store operation
                    0032  2717                .WORD   VS125$L_CSR                     ; Byte VS125$L_CSR to data
                      00  2719                .BYTE   0                     ; 0 VS125$L_CSR to data
                      20  271A                .BYTE   32                    ; 32 of data field
                      88  271B                .BYTE   Pd$_Store             ; Indicate store operation
                    0018  271C                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
```

```
                      00  271E                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  271F                .BYTE   18                      ; 18 of data field
                      83  2720                .BYTE   Pd$_Octal               ; Indicate scan octal
         52 4F 54 43 45 56 00' 2721           .ASCIC  "VECTOR"                ; VECTOR string
                      06  2721
            000001FE 00000002  2728           .LONG   ^O2,^O776               ; 2 , 776 limits on input
                      88  2730                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  2731                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  2733                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  2734                .BYTE   9                       ; 9 of data field
                      82  2735                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                   52 42 00' 2736             .ASCIC  "BR"    ; BR string
                      02  2736
            00000007 00000004  2739           .LONG   4,7     ; 4 , 7 limits on input
                      88  2741                .BYTE   Pd$_Store               ; Indicate store operation
                    0036  2742                .WORD   VS125$B_BR                      ; Byte VS125$B_BR to data
                      00  2744                .BYTE   0                       ; 0 VS125$B_BR to data
                      08  2745                .BYTE   8                       ; 8 of data field
                      81  2746                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          2747   385 VS300:    $DS_VS300                               ;[12]                                   ;G:4
                          2747                .SAVE   LOCAL_BLOCK
                          2747                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_VS300_CSR,      HP$L_VS300_CSR:
                          0032                .IIF    NB,VS300$L_CSR, VS300$L_CSR:
                 00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_VS300_BR,       HP$B_VS300_BR:
                          0036                .IIF    NB,VS300$B_BR,  VS300$B_BR:
                 00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_VS300_LEN,      HP$K_VS300_LEN:
                          0037                .IIF    NB,VS300$K_LEN, VS300$K_LEN:
                      00002747                .RESTORE
         30 30 33 53 56 00' 2747              .ASCIC  "VS300" ; VS300 name
                      05  2747
                      37  274D                .BYTE   VS300$K_LEN                     ; VS300$K_LEN of resultant P-table
                      01  274E                .BYTE   1                       ; Maximum unit number                   ;G:65
                    5356  274F                .WORD   ^A"VS"  ; Two character mnemonic for QIO VS300 VS
                      80  2751                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  2752                .Byte   Pd$_Name                ; Directive op-code
                      1B  2753                .Byte   Ptd$M_EndDevice                 ; Enforced VB codes
                   42 56 00' 2754             .Ascic  "VB"    ; Enforced device name prefix
                      02  2754
                      86  2757                .BYTE   Pd$_Literal             ; Indicate literal
                 00000001  2758                .LONG   HP$M_ALLOC                      ; Constant
                      88  275C                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  275D                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  275F                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2760                .BYTE   8                       ; 8 of data field
                      83  2761                .BYTE   Pd$_Octal               ; Indicate scan octal
                52 53 43 00' 2762             .ASCIC  "CSR"   ; CSR string
                      03  2762
            0003FFFE 0003E000  2766           .LONG   ^O760000,^O777776               ; 760000 , 777776 limits on input
                      88  276E                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  276F                .WORD   VS300$L_CSR                     ; Byte VS300$L_CSR to data
                      00  2771                .BYTE   0                       ; 0 VS300$L_CSR to data
                      20  2772                .BYTE   32                      ; 32 of data field
                      88  2773                .BYTE   Pd$_Store               ; Indicate store operation
```

```
                     0018  2774                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  2776                .BYTE   0                   ; 0 HP$A_DEVICE to data
                       12  2777                .BYTE   18                  ; 18 of data field
                       83  2778                .BYTE   Pd$_Octal           ; Indicate scan octal
           52 4F 54 43 45 56 00' 2779          .ASCIC  "VECTOR"            ; VECTOR string
                       06  2779
              000001FE 00000002  2780          .LONG   ^O2,^O776           ; 2 , 776 limits on input
                       88  2788                .BYTE   Pd$_Store           ; Indicate store operation
                     0024  2789                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                       00  278B                .BYTE   0                   ; 0 HP$W_VECTOR to data
                       09  278C                .BYTE   9                   ; 9 of data field
                       82  278D                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                  52 42 00' 278E               .ASCIC  "BR"    ; BR string
                       02  278E
              00000007 00000004  2791          .LONG   4,7     ; 4 , 7 limits on input
                       88  2799                .BYTE   Pd$_Store           ; Indicate store operation
                     0036  279A                .WORD   VS300$B_BR                  ; Byte VS300$B_BR to data
                       00  279C                .BYTE   0                   ; 0 VS300$B_BR to data
                       08  279D                .BYTE   8                   ; 8 of data field
                       81  279E                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           279F   386 VT50:    $DS_VT50
                           279F                .SAVE   LOCAL_BLOCK
                           279F                .PSECT  $ABS$,ABS
                  00000032  0050               .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_VT50_LEN,       HP$K_VT50_LEN:
                           0032                .IIF    NB,VT50$K_LEN,  VT50$K_LEN:
                      0000279F                 .RESTORE
        30 35 54 56 00' 279F                   .ASCIC  "VT50"  ; VT50 name
                       04  279F
                       32  27A4                .BYTE   VT50$K_LEN                  ; VT50$K_LEN of resultant P-table
                       FF  27A5                .BYTE   255                 ; Maximum unit number                   ;G:65
                     0000  27A6                .WORD   ^A""    ; Two character mnemonic for QIO VT50
                       80  27A8                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  27A9                .Byte   Pd$_Name            ; Directive op-code
                       1B  27AA                .Byte   Ptd$M_EndDevice             ; Enforced TT codes
                  54 54 00' 27AB               .Ascic  "TT"    ; Enforced device name prefix
                       02  27AB
                       86  27AE                .BYTE   Pd$_Literal         ; Indicate literal
                  00000001  27AF                .LONG   HP$M_ALLOC                  ; Constant
                       88  27B3                .BYTE   Pd$_Store           ; Indicate store operation
                     000A  27B4                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                       00  27B6                .BYTE   0                   ; 0 HP$B_FLAGS to data
                       08  27B7                .BYTE   8                   ; 8 of data field
                       81  27B8                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           27B9   387 VT52:    $DS_VT52
                           27B9                .SAVE   LOCAL_BLOCK
                           27B9                .PSECT  $ABS$,ABS
                  00000032  0050               .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_VT52_LEN,       HP$K_VT52_LEN:
                           0032                .IIF    NB,VT52$K_LEN,  VT52$K_LEN:
                      000027B9                 .RESTORE
        32 35 54 56 00' 27B9                   .ASCIC  "VT52"  ; VT52 name
                       04  27B9
                       32  27BE                .BYTE   VT52$K_LEN                  ; VT52$K_LEN of resultant P-table
                       FF  27BF                .BYTE   255                 ; Maximum unit number                   ;G:65
                     0000  27C0                .WORD   ^A""    ; Two character mnemonic for QIO VT52
                       80  27C2                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
```

```
                      8D  27C3                .Byte   Pd$_Name         ; Directive op-code
                      1B  27C4                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 27C5                .Ascic  "TT"    ; Enforced device name prefix
                      02  27C5
                      86  27C8                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  27C9                .LONG   HP$M_ALLOC               ; Constant
                      88  27CD                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  27CE                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  27D0                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  27D1                .BYTE   8                ; 8 of data field
                      81  27D2                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          27D3    388 VT55:   $DS_VT55
                          27D3                .SAVE   LOCAL_BLOCK
                          27D3                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT55_LEN,       HP$K_VT55_LEN:
                          0032                .IIF    NB,VT55$K_LEN,  VT55$K_LEN:
                      000027D3                .RESTORE
             35 35 54 56 00' 27D3             .ASCIC  "VT55"  ; VT55 name
                      04  27D3
                      32  27D8                .BYTE   VT55$K_LEN               ; VT55$K_LEN of resultant P-table
                      FF  27D9                .BYTE   255              ; Maximum unit number                  ;G:65
                    0000  27DA                .WORD   ^A""     ; Two character mnemonic for QIO VT55
                      80  27DC                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  27DD                .Byte   Pd$_Name         ; Directive op-code
                      1B  27DE                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 27DF                .Ascic  "TT"    ; Enforced device name prefix
                      02  27DF
                      86  27E2                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  27E3                .LONG   HP$M_ALLOC               ; Constant
                      88  27E7                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  27E8                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  27EA                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  27EB                .BYTE   8                ; 8 of data field
                      81  27EC                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          27ED    389 VT61:   $DS_VT61                         ;[32]                  ;G:4
                          27ED                .SAVE   LOCAL_BLOCK
                          27ED                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT61_LEN,       HP$K_VT61_LEN:
                          0032                .IIF    NB,VT61$K_LEN,  VT61$K_LEN:
                      000027ED                .RESTORE
             31 36 54 56 00' 27ED             .ASCIC  "VT61"  ; VT61 name
                      04  27ED
                      32  27F2                .BYTE   VT61$K_LEN               ; VT61$K_LEN of resultant P-table
                      08  27F3                .BYTE   8                ; Maximum unit number                  ;G:65
                    0000  27F4                .WORD   ^A""     ; Two character mnemonic for QIO VT61
                      80  27F6                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  27F7                .Byte   Pd$_Name         ; Directive op-code
                      1B  27F8                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 27F9                .Ascic  "TT"    ; Enforced device name prefix
                      02  27F9
                      86  27FC                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  27FD                .LONG   HP$M_ALLOC               ; Constant
                      88  2801                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2802                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  2804                .BYTE   0                ; 0 HP$B_FLAGS to data
```

```
                  08  2805                .BYTE   8                   ; 8 of data field
                  81  2806                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                      2807   390 VT71:    $DS_VT71                            ;[32]                                   ;G:4
                      2807                .SAVE   LOCAL_BLOCK
                      2807                .PSECT  $ABS$,ABS
          00000032    0050                .=HP$A_DEPENDENT
                      0032                .IIF    NB,HP$K_VT71_LEN,       HP$K_VT71_LEN:
                      0032                .IIF    NB,VT71$K_LEN,  VT71$K_LEN:
                  00002807                .RESTORE
   31 37 54 56 00'    2807                .ASCIC  "VT71"  ; VT71 name
                  04  2807
                  32  280C                .BYTE   VT71$K_LEN                  ; VT71$K_LEN of resultant P-table
                  08  280D                .BYTE   8                   ; Maximum unit number                           ;G:65
                0000  280E                .WORD   ^A""    ; Two character mnemonic for QIO VT71
                  80  2810                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                  8D  2811                .Byte   Pd$_Name            ; Directive op-code
                  1B  2812                .Byte   Ptd$M_EndDevice             ; Enforced TT codes
          54 54 00'   2813                .Ascic  "TT"    ; Enforced device name prefix
                  02  2813
                  86  2816                .BYTE   Pd$_Literal         ; Indicate literal
          00000001    2817                .LONG   HP$M_ALLOC                  ; Constant
                  88  281B                .BYTE   Pd$_Store           ; Indicate store operation
                000A  281C                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                  00  281E                .BYTE   0                   ; 0 HP$B_FLAGS to data
                  08  281F                .BYTE   8                   ; 8 of data field
                  81  2820                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                      2821   391 VT100:   $DS_VT100
                      2821                .SAVE   LOCAL_BLOCK
                      2821                .PSECT  $ABS$,ABS
          00000032    0050                .=HP$A_DEPENDENT
                      0032                .IIF    NB,HP$K_VT100_LEN,      HP$K_VT100_LEN:
                      0032                .IIF    NB,VT100$K_LEN, VT100$K_LEN:
                  00002821                .RESTORE
   30 30 31 54 56 00' 2821                .ASCIC  "VT100" ; VT100 name
                  05  2821
                  32  2827                .BYTE   VT100$K_LEN                 ; VT100$K_LEN of resultant P-table
                  FF  2828                .BYTE   255                 ; Maximum unit number                           ;G:65
                0000  2829                .WORD   ^A""    ; Two character mnemonic for QIO VT100
                  80  282B                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                  8D  282C                .Byte   Pd$_Name            ; Directive op-code
                  1B  282D                .Byte   Ptd$M_EndDevice             ; Enforced TT codes
          54 54 00'   282E                .Ascic  "TT"    ; Enforced device name prefix
                  02  282E
                  86  2831                .BYTE   Pd$_Literal         ; Indicate literal
          00000001    2832                .LONG   HP$M_ALLOC                  ; Constant
                  88  2836                .BYTE   Pd$_Store           ; Indicate store operation
                000A  2837                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                  00  2839                .BYTE   0                   ; 0 HP$B_FLAGS to data
                  08  283A                .BYTE   8                   ; 8 of data field
                  81  283B                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                      283C   392 VT101:   $DS_VT101
                      283C                .SAVE   LOCAL_BLOCK
                      283C                .PSECT  $ABS$,ABS
          00000032    0050                .=HP$A_DEPENDENT
                      0032                .IIF    NB,HP$K_VT101_LEN,      HP$K_VT101_LEN:
                      0032                .IIF    NB,VT101$K_LEN, VT101$K_LEN:
                  0000283C                .RESTORE
```

```
                31 30 31 54 56 00' 283C                 .ASCIC  "VT101" ; VT101 name
                                05  283C
                                32  2842                 .BYTE   VT101$K_LEN             ; VT101$K_LEN of resultant P-table
                                FF  2843                 .BYTE   255             ; Maximum unit number                   ;G:65
                              0000  2844                 .WORD   ^A""    ; Two character mnemonic for QIO VT101
                                80  2846                 .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  2847                 .Byte   Pd$_Name        ; Directive op-code
                                1B  2848                 .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                           54 54 00' 2849                 .Ascic  "TT"    ; Enforced device name prefix
                                02  2849
                                86  284C                 .BYTE   Pd$_Literal     ; Indicate literal
                          00000001  284D                 .LONG   HP$M_ALLOC              ; Constant
                                88  2851                 .BYTE   Pd$_Store       ; Indicate store operation
                              000A  2852                 .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                                00  2854                 .BYTE   0               ; 0 HP$B_FLAGS to data
                                08  2855                 .BYTE   8               ; 8 of data field
                                81  2856                 .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    2857    393 VT102:   $DS_VT102
                                    2857                 .SAVE   LOCAL_BLOCK
                                    2857                 .PSECT  $ABS$,ABS
                          00000032  0050                 .=HP$A_DEPENDENT
                                    0032                 .IIF    NB,HP$K_VT102_LEN,      HP$K_VT102_LEN:
                                    0032                 .IIF    NB,VT102$K_LEN, VT102$K_LEN:
                                00002857                 .RESTORE
                32 30 31 54 56 00' 2857                 .ASCIC  "VT102" ; VT102 name
                                05  2857
                                32  285D                 .BYTE   VT102$K_LEN             ; VT102$K_LEN of resultant P-table
                                FF  285E                 .BYTE   255             ; Maximum unit number                   ;G:65
                              0000  285F                 .WORD   ^A""    ; Two character mnemonic for QIO VT102
                                80  2861                 .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  2862                 .Byte   Pd$_Name        ; Directive op-code
                                1B  2863                 .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                           54 54 00' 2864                 .Ascic  "TT"    ; Enforced device name prefix
                                02  2864
                                86  2867                 .BYTE   Pd$_Literal     ; Indicate literal
                          00000001  2868                 .LONG   HP$M_ALLOC              ; Constant
                                88  286C                 .BYTE   Pd$_Store       ; Indicate store operation
                              000A  286D                 .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                                00  286F                 .BYTE   0               ; 0 HP$B_FLAGS to data
                                08  2870                 .BYTE   8               ; 8 of data field
                                81  2871                 .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    2872    394 VT125:   $DS_VT125
                                    2872                 .SAVE   LOCAL_BLOCK
                                    2872                 .PSECT  $ABS$,ABS
                          00000032  0050                 .=HP$A_DEPENDENT
                                    0032                 .IIF    NB,HP$K_VT125_LEN,      HP$K_VT125_LEN:
                                    0032                 .IIF    NB,VT125$K_LEN, VT125$K_LEN:
                                00002872                 .RESTORE
                35 32 31 54 56 00' 2872                 .ASCIC  "VT125" ; VT125 name
                                05  2872
                                32  2878                 .BYTE   VT125$K_LEN             ; VT125$K_LEN of resultant P-table
                                FF  2879                 .BYTE   255             ; Maximum unit number                   ;G:65
                              0000  287A                 .WORD   ^A""    ; Two character mnemonic for QIO VT125
                                80  287C                 .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  287D                 .Byte   Pd$_Name        ; Directive op-code
                                1B  287E                 .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                           54 54 00' 287F                 .Ascic  "TT"    ; Enforced device name prefix
```

```
                    02  287F
                    86  2882                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  2883                .LONG   HP$M_ALLOC              ; Constant
                    88  2887                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  2888                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  288A                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  288B                .BYTE   8               ; 8 of data field
                    81  288C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        288D   395 VT131:   $DS_VT131                       ;[32]                                   ;G:4
                        288D                .SAVE   LOCAL_BLOCK
                        288D                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_VT131_LEN,      HP$K_VT131_LEN:
                        0032                .IIF    NB,VT131$K_LEN, VT131$K_LEN:
                    0000288D                .RESTORE
           31 33 31 54 56 00' 288D          .ASCIC  "VT131" ; VT131 name
                    05  288D
                    32  2893                .BYTE   VT131$K_LEN             ; VT131$K_LEN of resultant P-table
                    08  2894                .BYTE   8               ; Maximum unit number                   ;G:65
                  0000  2895                .WORD   ^A""    ; Two character mnemonic for QIO VT131
                    80  2897                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  2898                .Byte   Pd$_Name        ; Directive op-code
                    1B  2899                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
              54 54 00' 289A                .Ascic  "TT"    ; Enforced device name prefix
                    02  289A
                    86  289D                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  289E                .LONG   HP$M_ALLOC              ; Constant
                    88  28A2                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  28A3                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  28A5                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  28A6                .BYTE   8               ; 8 of data field
                    81  28A7                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        28A8   396 VT132:   $DS_VT132                       ;[32]                                   ;G:4
                        28A8                .SAVE   LOCAL_BLOCK
                        28A8                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_VT132_LEN,      HP$K_VT132_LEN:
                        0032                .IIF    NB,VT132$K_LEN, VT132$K_LEN:
                    000028A8                .RESTORE
           32 33 31 54 56 00' 28A8          .ASCIC  "VT132" ; VT132 name
                    05  28A8
                    32  28AE                .BYTE   VT132$K_LEN             ; VT132$K_LEN of resultant P-table
                    08  28AF                .BYTE   8               ; Maximum unit number                   ;G:65
                  0000  28B0                .WORD   ^A""    ; Two character mnemonic for QIO VT132
                    80  28B2                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  28B3                .Byte   Pd$_Name        ; Directive op-code
                    1B  28B4                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
              54 54 00' 28B5                .Ascic  "TT"    ; Enforced device name prefix
                    02  28B5
                    86  28B8                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  28B9                .LONG   HP$M_ALLOC              ; Constant
                    88  28BD                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  28BE                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  28C0                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  28C1                .BYTE   8               ; 8 of data field
                    81  28C2                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        28C3   397 VT220:   $DS_VT220                       ;[14]                                   ;G:4
```

```
                        28C3                .SAVE   LOCAL_BLOCK
                        28C3                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_VT220_LEN,      HP$K_VT220_LEN:
                        0032                .IIF    NB,VT220$K_LEN, VT220$K_LEN:
                    000028C3                .RESTORE
          30 32 32 54 56 00' 28C3           .ASCIC  "VT220" ; VT220 name
                          05  28C3
                          32  28C9          .BYTE   VT220$K_LEN             ; VT220$K_LEN of resultant P-table
                          FF  28CA          .BYTE   255             ; Maximum unit number                   ;G:65
                        0000  28CB          .WORD   ^A""    ; Two character mnemonic for QIO VT220
                          80  28CD          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  28CE          .Byte   Pd$_Name        ; Directive op-code
                          1B  28CF          .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                  54 54 00'   28D0          .Ascic  "TT"    ; Enforced device name prefix
                          02  28D0
                          86  28D3          .BYTE   Pd$_Literal     ; Indicate literal
                  00000001    28D4          .LONG   HP$M_ALLOC              ; Constant
                          88  28D8          .BYTE   Pd$_Store       ; Indicate store operation
                        000A  28D9          .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  28DB          .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  28DC          .BYTE   8               ; 8 of data field
                          81  28DD          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        28DE    398 VT240:  $DS_VT240                       ;[14]                   ;G:4
                        28DE                .SAVE   LOCAL_BLOCK
                        28DE                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$K_VT240_LEN,      HP$K_VT240_LEN:
                        0032                .IIF    NB,VT240$K_LEN, VT240$K_LEN:
                    000028DE                .RESTORE
          30 34 32 54 56 00' 28DE           .ASCIC  "VT240" ; VT240 name
                          05  28DE
                          32  28E4          .BYTE   VT240$K_LEN             ; VT240$K_LEN of resultant P-table
                          FF  28E5          .BYTE   255             ; Maximum unit number                   ;G:65
                        0000  28E6          .WORD   ^A""    ; Two character mnemonic for QIO VT240
                          80  28E8          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  28E9          .Byte   Pd$_Name        ; Directive op-code
                          1B  28EA          .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                  54 54 00'   28EB          .Ascic  "TT"    ; Enforced device name prefix
                          02  28EB
                          86  28EE          .BYTE   Pd$_Literal     ; Indicate literal
                  00000001    28EF          .LONG   HP$M_ALLOC              ; Constant
                          88  28F3          .BYTE   Pd$_Store       ; Indicate store operation
                        000A  28F4          .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  28F6          .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  28F7          .BYTE   8               ; 8 of data field
                          81  28F8          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        28F9    399         .END
```

```
$$N                             = 00000094     D
$$_$                            = 00000002     D
AA11K                             00000254 R   D   01
AA11K$B_BR                        00000032     D
AA11K$K_LEN                       00000033     D
AD11K                             000002A5 R   D   01
AD11K$B_BR                        00000032     D
AD11K$K_LEN                       00000033     D
AL_DEVTYP                         00000000 R   D   01
BIT...                          = 00000005     D
CI$B_BI_ID                        00000032     D
CI$B_BR                           00000033     D
CI$B_NODE                         00000034     D
CI$B_TR                           00000032     D
CI$K_LEN                          00000041     D
CI$L_FUNC                         00000035     D
CI$L_INDEX                        0000003D     D
CI$L_MAINT_ID                     00000039     D
CI750                             00000412 R   D   01
CI780                             00000491 R   D   01
CI_NODE                           000002F6 R   D   01
CONSOLE                           00000518 R   D   01
CONSOLE$K_LEN                     00000032     D
CR11                              00000526 R   D   01
CR11$B_BR                         00000036     D
CR11$K_LEN                        00000037     D
CR11$L_CSR                        00000032     D
DEQNA                             0000057D R   D   01
DEQNA$K_LEN                       00000036     D
DEQNA$L_CSR                       00000032     D
DHU11                             000005A5 R   D   01
DHV11                             000005F3 R   D   01
DISK                              00000641 R   D   01
DISK$K_LEN                        00000032     D
DL11                              00000651 R   D   01
DL11$B_BR                         00000036     D
DL11$K_LEN                        00000037     D
DL11$L_CSR                        00000032     D
DLVJ1                             0000069E R   D   01
DLVJ1$B_FLAGS                     00000036     D
DLVJ1$K_LEN                       00000037     D
DLVJ1$L_CSR                       00000032     D
DLVJ1$V_EVEPAR                  = 00000003     D
DLVJ1$V_NDATA                   = 00000000     D
DLVJ1$V_NSTOP                   = 00000001     D
DLVJ1$V_PARITY                  = 00000002     D
DMC11                             00000752 R   D   01
DMC11$B_BR                        00000036     D
DMC11$K_LEN                       00000037     D
DMC11$L_CSR                       00000032     D
DMF32                             000007A0 R   D   01
DMF32$B_BR                        00000034     D
DMF32$B_COMMON                    00000032     D
DMF32$B_FLAGS                     00000033     D
DMF32$K_LEN                       00000039     D
DMF32$L_CSR                       00000035     D
DMF32A                            000007F3 R   D   01
DMF32A$B_ACTIV                    00000033     D
DMF32A$B_BR                       00000032     D
DMF32A$B_JUMP                     00000037     D
DMF32A$B_SPEED                    00000034     D
DMF32A$K_LEN                      00000038     D
DMF32A$W_FLAGS                    00000035     D
DMF32P                            00000915 R   D   01
DMF32P$B_BR                       00000034     D
DMF32P$B_COMMON                   00000032     D
DMF32P$B_FLAGS                    00000033     D
DMF32P$K_LEN                      00000039     D
DMF32P$L_CSR                      00000035     D
DMF32S                            0000098F R   D   01
DMF32S$B_BR                       00000034     D
DMF32S$B_COMMON                   00000032     D
DMF32S$B_FLAGS                    00000033     D
DMF32S$K_LEN                      00000039     D
DMF32S$L_CSR                      00000035     D
DMP11                             00000A03 R   D   01
DMP11$B_BR                        00000034     D
DMP11$B_COMMON                    00000032     D
DMP11$B_FLAGS                     00000033     D
DMP11$K_LEN                       00000039     D
DMP11$L_CSR                       00000035     D
DMR11                             00000ACC R   D   01
DMR11$B_BR                        00000036     D
DMR11$K_LEN                       00000037     D
DMR11$L_CSR                       00000032     D
DMZ32                             00000B1A R   D   01
DR11B                             00000C78 R   D   01
DR11B$B_BR                        00000036     D
DR11B$K_LEN                       00000037     D
DR11B$L_CSR                       00000032     D
DR11C                             00000CC6 R   D   01
DR11K                             00000D0F R   D   01
DR11K$B_BR                        00000032     D
DR11K$K_LEN                       00000033     D
DR11W                             00000D60 R   D   01
DR11W$B_BR                        00000036     D
DR11W$K_LEN                       00000037     D
DR11W$L_CSR                       00000032     D
DR750                             00000DAE R   D   01
DR750$B_BR                        00000033     D
DR750$B_CONFIG                    00000035     D
DR750$B_SELF                      00000034     D
DR750$B_SLOT                      00000032     D
DR750$B_UUT                       00000036     D
DR750$K_LEN                       00000037     D
DR780                             00000E39 R   D   01
DR780$B_BR                        00000033     D
DR780$B_CONFIG                    00000035     D
DR780$B_SELF                      00000034     D
DR780$B_TR                        00000032     D
DR780$B_UUT                       00000036     D
DR780$K_LEN                       00000037     D
DS$GA_PTDESC                      00000000 RG  D   01
DUP11                             00000ECC R   D   01
```

DUP11$B_BR	00000036		D		HP$B_DMZ32_BR	00000032		D
DUP11$K_LEN	00000037		D		HP$B_DMZ32_MODEM	00000039		D
DUP11$L_CSR	00000032		D		HP$B_DMZ32_OCTET0	00000033		D
DV11	00000F1A	R	D	01	HP$B_DMZ32_OCTET1	00000034		D
DV11$B_BR	00000036		D		HP$B_DMZ32_OCTET2	00000035		D
DV11$K_LEN	00000037		D		HP$B_DMZ32_SPEED	00000036		D
DV11$L_CSR	00000032		D		HP$B_DR11C_BR	00000032		D
DW730	00000F67	R	D	01	HP$B_DRIVE	0000000B		D
DW730$K_LEN	00000032		D		HP$B_DX20_DRIVE	00000032	G	D
DW750	00000F96	R	D	01	HP$B_DZ32_ACTIV	00000033		D
DW750$K_LEN	00000032		D		HP$B_DZ32_BRLVL	00000032		D
DW780	00000FDA	R	D	01	HP$B_DZ32_SPEED	00000034		D
DW780$B_BR	00000033		D		HP$B_FLAGS	0000000A		D
DW780$B_TR	00000032		D		HP$B_IEU11A_BR	00000032		D
DW780$K_LEN	00000034		D		HP$B_LESI_BR	00000036		D
DX20	0000104E	R	D	01	HP$B_NBIA_COMMON	00000032		D
DX20$B_DRIVE	00000032	G	D		HP$B_NBIB_COMMON	00000032		D
DX20$K_LEN	00000033	G	D		HP$B_RB730_BR	00000036		D
DZ11	0000107C	R	D	01	HP$B_RH750_BR	00000032		D
DZ11$B_BR	00000036		D		HP$B_RH780_BR	00000033		D
DZ11$B_MTYPE	00000037		D		HP$B_RH780_TR	00000032		D
DZ11$K_LEN	00000038		D		HP$B_RK611_BR	00000036		D
DZ11$L_CSR	00000032		D		HP$B_RL11_BR	00000036		D
DZ32	000010E5	R	D	01	HP$B_RUX50_BR	00000032		D
DZ32$B_ACTIV	00000033		D		HP$B_RX211_BR	00000036		D
DZ32$B_BRLVL	00000032		D		HP$B_TM03_DRIVE	00000032		D
DZ32$B_SPEED	00000034		D		HP$B_TM78_DRIVE	00000032		D
DZ32$W_FLAGS	00000035		D		HP$B_TS04_BR	00000036		D
DZ32$_LEN	00000037		D		HP$B_TS11_BR	00000036		D
DZV11	000011D6	R	D	01	HP$B_TU80_BR	00000036		D
DZV11$K_LEN	00000036		D		HP$B_TU81_BR	00000036		D
DZV11$L_CSR	00000032		D		HP$B_UDA50_BR	00000036		D
HP$A_DEPENDENT	00000032		D		HP$B_UDA50_BURST	00000037		D
HP$A_DEVICE	00000018		D		HP$B_VS100_BR	00000036		D
HP$A_DVA	0000001C		D		HP$B_VS125_BR	00000036		D
HP$A_LINK	00000020		D		HP$B_VS300_BR	00000036		D
HP$B_BI_NODE	00000033		D		HP$K_CI_LEN	00000041		D
HP$B_BI_NUM	00000038		D		HP$K_DHUV_LEN	00000039		D
HP$B_BR	00000037		D		HP$K_DISK_LEN	00000032		D
HP$B_CI_BR	00000033		D		HP$K_DMF32A_LEN	00000038		D
HP$B_CI_NODE	00000034		D		HP$K_DMF32P_LEN	00000039		D
HP$B_CI_TR	00000032		D		HP$K_DMF32S_LEN	00000039		D
HP$B_CPU_ID	0000004F	G	D		HP$K_DMF32_LEN	00000039		D
HP$B_DHUV_BR	00000038		D		HP$K_DMZ32_LEN	0000003A		D
HP$B_DMF32A_ACTIV	00000033		D		HP$K_DR11C_LEN	00000033		D
HP$B_DMF32A_BR	00000032		D		HP$K_DX20_LEN	00000033	G	D
HP$B_DMF32A_JUMP	00000037		D		HP$K_IEU11A_LEN	00000033		D
HP$B_DMF32A_SPEED	00000034		D		HP$K_LA100_LEN	00000032		D
HP$B_DMF32P_BR	00000034		D		HP$K_LA12_LEN	00000032		D
HP$B_DMF32P_COMMON	00000032		D		HP$K_LA210_LEN	00000032		D
HP$B_DMF32P_FLAGS	00000033		D		HP$K_LEN	00000037		D
HP$B_DMF32S_BR	00000034		D		HP$K_LESI_LEN	00000037		D
HP$B_DMF32S_COMMON	00000032		D		HP$K_NBIA_LEN	00000038		D
HP$B_DMF32S_FLAGS	00000033		D		HP$K_NBIB_LEN	00000039		D
HP$B_DMF32_BR	00000034		D		HP$K_RA60_LEN	00000032		D
HP$B_DMF32_COMMON	00000032		D		HP$K_RA70_LEN	00000032		D
HP$B_DMF32_FLAGS	00000033		D		HP$K_RA80_LEN	00000032		D

```
HP$K_RA81_LEN                   00000032    D
HP$K_RA82_LEN                   00000032    D
HP$K_RA90_LEN                   00000032    D
HP$K_RB730_LEN                  00000037    D
HP$K_RC25_LEN                   00000032    D
HP$K_RCF25_LEN                  00000032    D
HP$K_RH750_LEN                  00000033    D
HP$K_RH780_LEN                  00000034    D
HP$K_RK06_LEN                   00000032    D
HP$K_RK07_LEN                   00000032    D
HP$K_RK611_LEN                  00000037    D
HP$K_RL01_LEN                   00000032    D
HP$K_RL02_LEN                   00000032    D
HP$K_RL11_LEN                   00000037    D
HP$K_RM03_LEN                   00000032    D
HP$K_RM05_LEN                   00000032    D
HP$K_RM80_LEN                   00000032    D
HP$K_RP04_LEN                   00000032    D
HP$K_RP05_LEN                   00000032    D
HP$K_RP06_LEN                   00000032    D
HP$K_RP07_LEN                   00000032    D
HP$K_RUX50_LEN                  00000033    D
HP$K_RX02_LEN                   00000032    D
HP$K_RX211_LEN                  00000037    D
HP$K_SBIA_LEN                   00000032    D
HP$K_TE16_LEN                   00000032    D
HP$K_TK50_LEN                   00000032    D
HP$K_TM03_LEN                   00000033    D
HP$K_TM78_LEN                   00000033    D
HP$K_TS04_LEN                   00000037    D
HP$K_TS05_LEN                   00000033    D
HP$K_TS11_LEN                   00000037    D
HP$K_TU45_LEN                   00000032    D
HP$K_TU58_LEN                   00000032    D
HP$K_TU70_LEN                   00000032  G D
HP$K_TU72_LEN                   00000032  G D
HP$K_TU77_LEN                   00000032    D
HP$K_TU78_LEN                   00000032    D
HP$K_TU80_LEN                   00000037    D
HP$K_TU81_LEN                   00000037    D
HP$K_UBE_LEN                    00000036    D
HP$K_UDA50_LEN                  00000038    D
HP$K_VS100_LEN                  00000037    D
HP$K_VS125_LEN                  00000037    D
HP$K_VS300_LEN                  00000037    D
HP$K_VT100_LEN                  00000032    D
HP$K_VT101_LEN                  00000032    D
HP$K_VT102_LEN                  00000032    D
HP$K_VT125_LEN                  00000032    D
HP$K_VT131_LEN                  00000032    D
HP$K_VT132_LEN                  00000032    D
HP$K_VT220_LEN                  00000032    D
HP$K_VT240_LEN                  00000032    D
HP$K_VT50_LEN                   00000032    D
HP$K_VT52_LEN                   00000032    D
HP$K_VT55_LEN                   00000032    D
HP$K_VT61_LEN                   00000032    D
HP$K_VT71_LEN                   00000032    D
HP$L_CI_FUNC                    00000035    D
HP$L_CI_INDEX                   0000003D    D
HP$L_CI_MAINT_ID                00000039    D
HP$L_DHUV_CSR                   00000032    D
HP$L_DMF32P_CSR                 00000035    D
HP$L_DMF32S_CSR                 00000035    D
HP$L_DMF32_CSR                  00000035    D
HP$L_KAA_SLOT                   0000004B  G D
HP$L_LESI_IP                    00000032    D
HP$L_NBIA_CSR                   00000033    D
HP$L_NBIB_CSR                   00000034    D
HP$L_RB730_CSR                  00000032    D
HP$L_RK611_CSR                  00000032    D
HP$L_RL11_CSR                   00000032    D
HP$L_RX211_CSR                  00000032    D
HP$L_TS04_CSR                   00000032    D
HP$L_TS11_CSR                   00000032    D
HP$L_TU80_CSR                   00000032    D
HP$L_TU81_CSR                   00000032    D
HP$L_UBE_CSR                    00000032    D
HP$L_UDA50_UDAIP                00000032    D
HP$L_VS100_CSR                  00000032    D
HP$L_VS125_CSR                  00000032    D
HP$L_VS300_CSR                  00000032    D
HP$M_ALLOC                    = 00000001    D
HP$Q_DEVICE                     00000000    D
HP$T_DEVICE                     0000000C    D
HP$T_TYPE                       00000026    D
HP$W_DHUV_VEC                   00000036    D
HP$W_DMF32A_FLAGS               00000035    D
HP$W_DMZ32_LOOP                 00000037    D
HP$W_DZ32_FLAGS                 00000035    D
HP$W_SIZE                       00000008    D
HP$W_VECTOR                     00000024    D
HPE$A_EPB                       00000016    D
HPE$T_DEVICE                    00000000    D
HPE$W_EXT_DRIVE                 00000014    D
IEU11A                          00001213  R   D   01
IEU11A$B_BR                     00000032      D
IEU11A$K_LEN                    00000033      D
KA$B_ACC_TYPE                   00000042    G D
KA$B_NODE_ID                    00000046    G D
KA$B_RESERVED                   00000043    G D
KA$B_SID_TYPE                   0000003D    G D
KA$K_LEN                        00000050    G D
KA$L_FLAGS                      00000032    G D
KA$L_MEM_SIZE                   00000047    G D
KA$L_SID                        0000003A    G D
KA$M_CHAR                     = 00000100    G D
KA$M_CRC                      = 00000010    G D
KA$M_D_FLOAT                  = 00000002    G D
KA$M_EDIT                     = 00000080    G D
KA$M_F_FLOAT                  = 00000001    G D
KA$M_G_FLOAT                  = 00000004    G D
KA$M_H_FLOAT                  = 00000008    G D
KA$M_PACKED                   = 00000020    G D
```

```
KA$M_PACK_MUL                   = 00000040  G D
KA$M_SB_ERR                     = 02000000  G D
KA$M_TODR                       = 00010000  G D
KA$M_WCS_LD                     = 01000000  G D
KA$V_CHAR                       = 00000008  G D
KA$V_CRC                        = 00000004  G D
KA$V_D_FLOAT                    = 00000001  G D
KA$V_EDIT                       = 00000007  G D
KA$V_F_FLOAT                    = 00000000  G D
KA$V_G_FLOAT                    = 00000002  G D
KA$V_H_FLOAT                    = 00000003  G D
KA$V_PACKED                     = 00000005  G D
KA$V_PACK_MUL                   = 00000006  G D
KA$V_SB_ERR                     = 00000019  G D
KA$V_TODR                       = 00000010  G D
KA$V_WCS_LD                     = 00000018  G D
KA$W_LATENCY                      0000003E  G D
KA$W_MEM_SIZE                     00000044  G D
KA$W_PROGRESS                     00000040  G D
KA$W_RESERVED                     00000038  G D
KA$W_WCS                          00000036  G D
KA730                             0000125E R   D   01
KA750                             00001339 R   D   01
KA780                             00001405 R   D   01
KA785                             00001556 R   D   01
KA86                              000014AE R   D   01
KMC11                             000015FF R   D   01
KMC11$B_BR                        00000036     D
KMC11$K_LEN                       00000037     D
KMC11$L_CSR                       00000032     D
KW11K                             0000164D R   D   01
KW11K$B_BR                        00000032     D
KW11K$K_LEN                       00000033     D
LA100                             000016B8 R   D   01
LA100$K_LEN                       00000032     D
LA12                              0000169E R   D   01
LA12$K_LEN                        00000032     D
LA120                             000016D3 R   D   01
LA120$K_LEN                       00000032     D
LA180                             000016EE R   D   01
LA180$K_LEN                       00000032     D
LA210                             00001757 R   D   01
LA210$K_LEN                       00000032     D
LA34                              00001709 R   D   01
LA34$K_LEN                        00000032     D
LA36                              00001723 R   D   01
LA36$K_LEN                        00000032     D
LA38                              0000173D R   D   01
LA38$K_LEN                        00000032     D
LESI                              00001772 R   D   01
LESI$B_BR                         00000036     D
LESI$K_LEN                        00000037     D
LESI$L_IP                         00000032     D
LG01                              000017BE R   D   01
LG01$K_LEN                        00000032     D
LG02                              000017D8 R   D   01
LG02$K_LEN                        00000032     D
LG03                              000017F2 R   D   01
LG03$K_LEN                        00000032     D
LN01                              0000180C R   D   01
LN01$K_LEN                        00000032     D
LN03                              00001826 R   D   01
LN03$K_LEN                        00000032     D
LP04                              0000183B R   D   01
LP04$K_LEN                        00000032     D
LP05                              00001855 R   D   01
LP05$K_LEN                        00000032     D
LP06                              0000186F R   D   01
LP06$K_LEN                        00000032     D
LP07                              00001889 R   D   01
LP07$K_LEN                        00000032     D
LP11                              000018A3 R   D   01
LP11$B_BR                         00000036     D
LP11$K_LEN                        00000037     D
LP11$L_CSR                        00000032     D
LP14                              000018F0 R   D   01
LP14$K_LEN                        00000032     D
LP25                              0000190A R   D   01
LP25$K_LEN                        00000032     D
LP26                              00001924 R   D   01
LP26$K_LEN                        00000032     D
LP27                              0000193E R   D   01
LP27$K_LEN                        00000032     D
LPA11K                            00001958 R   D   01
LPA11K$B_BR                       00000036     D
LPA11K$K_LEN                      00000037     D
LPA11K$L_CSR                      00000032     D
LUA11                             000019B1 R   D   01
LUA11$B_BR                        00000032     D
LUA11$K_LEN                       00000037     D
LUA11$L_CSR                       00000033     D
MA780                             00001AD2 R   D   01
MA780$B_BR                        00000033     D
MA780$B_MPM                       00000034     D
MA780$B_PORT                      00000035     D
MA780$B_TR                        00000032     D
MA780$K_LEN                       00000036     D
MBE                               00001B4D R   D   01
MBE$K_LEN                         00000032     D
ML11                              00001B61 R   D   01
ML11$B_ARRAY                      00000032     D
ML11$B_CHIP                       00000033     D
ML11$K_LEN                        00000034     D
MS750                             00001BC3 R   D   01
MS750$K_LEN                       00000032     D
MS780                             00001BD4 R   D   01
MS780$B_ARRAYS                    00000033     D
MS780$B_TR                        00000032     D
MS780$K_LEN                       00000034     D
NBIA                              000019FF R   D   01
NBIB                              00001A6F R   D   01
PCL11                             00001C24 R   D   01
PCL11$B_BR                        00000036     D
PCL11$K_LEN                       00000037     D
```

```
PCL11$L_CSR                     00000032    D
PD$_ADD                       = 0000008A    D
PD$_CASE                      = 0000008C    D
PD$_COMPLEMENT                = 00000089    D
PD$_DECIMAL                   = 00000082    D
PD$_END                       = 00000081    D
PD$_FETCH                     = 00000087    D
PD$_HEXADECIMAL               = 00000084    D
PD$_LITERAL                   = 00000086    D
PD$_LOGICAL                   = 0000008B    D
PD$_NAME                      = 0000008D    D
PD$_OCTAL                     = 00000083    D
PD$_START                     = 00000080    D
PD$_STORE                     = 00000088    D
PD$_STRING                    = 00000085    D
PTD$M_CONTROLLER              = 00000002    D
PTD$M_DEVICE                  = 00000003    D
PTD$M_ENDDEVICE               = 0000001B    D
PTD$M_INHERIT                 = 00000018    D
PTD$M_INHERIT_CON             = 00000010    D
PTD$M_INHERIT_PRE             = 00000008    D
PTD$M_UNIT                    = 00000001    D
PX780                           00001C6D R  D  01
PX780$B_BR                      00000033    D
PX780$B_NODE                    00000034    D
PX780$B_TR                      00000032    D
PX780$K_LEN                     00000035    D
R80                             00001CD6 R  D  01
R80$K_LEN                       00000032    D
RA60                            00001CEF R  D  01
RA60$K_LEN                      00000032    D
RA70                            00001D09 R  D  01
RA70$K_LEN                      00000032    D
RA80                            00001D23 R  D  01
RA80$K_LEN                      00000032    D
RA81                            00001D3D R  D  01
RA81$K_LEN                      00000032    D
RA82                            00001D57 R  D  01
RA82$K_LEN                      00000032    D
RA90                            00001D71 R  D  01
RA90$K_LEN                      00000032    D
RB730                           00001D8B R  D  01
RB730$B_BR                      00000036    D
RB730$K_LEN                     00000037    D
RB730$L_CSR                     00000032    D
RC25                            00001DD5 R  D  01
RC25$K_LEN                      00000032    D
RCF25                           00001DBA R  D  01
RCF25$K_LEN                     00000032    D
RD51                            00001DEF R  D  01
RD51$K_LEN                      00000032    D
RD52                            00001E09 R  D  01
RD52$K_LEN                      00000032    D
RH750                           00001E23 R  D  01
RH750$B_BR                      00000032    D
RH750$K_LEN                     00000033    D
RH780                           00001E7C R  D  01
RH780$B_BR                      00000033    D
RH780$B_TR                      00000032    D
RH780$K_LEN                     00000034    D
RK06                            00001F62 R  D  01
RK06$K_LEN                      00000032    D
RK07                            00001F7C R  D  01
RK07$K_LEN                      00000032    D
RK611                           00001F96 R  D  01
RK611$B_BR                      00000036    D
RK611$K_LEN                     00000037    D
RK611$L_CSR                     00000032    D
RL01                            00001EE1 R  D  01
RL01$K_LEN                      00000032    D
RL02                            00001EFB R  D  01
RL02$K_LEN                      00000032    D
RL11                            00001F15 R  D  01
RL11$B_BR                       00000036    D
RL11$K_LEN                      00000037    D
RL11$L_CSR                      00000032    D
RM03                            00001FE4 R  D  01
RM03$K_LEN                      00000032    D
RM05                            00002008 R  D  01
RM05$K_LEN                      00000032    D
RM80                            0000202C R  D  01
RM80$K_LEN                      00000032    D
RP04                            00002050 R  D  01
RP04$K_LEN                      00000032    D
RP05                            00002074 R  D  01
RP05$K_LEN                      00000032    D
RP06                            00002098 R  D  01
RP06$K_LEN                      00000032    D
RP07                            000020BC R  D  01
RP07$K_LEN                      00000032    D
RQDX1                           000020E0 R  D  01
RQDX1$K_LEN                     00000036    D
RQDX1$L_IP                      00000032    D
RRD50                           00002107 R  D  01
RRD50$B_BR                      00000036  G D
RRD50$K_LEN                     00000037  G D
RRD50$L_CSR                     00000032  G D
RUX50                           00002163 R  D  01
RX02                            000021AB R  D  01
RX02$K_LEN                      00000032    D
RX180                           00002248 R  D  01
RX180$K_LEN                     00000032    D
RX211                           000021C5 R  D  01
RX211$B_BR                      00000036    D
RX211$K_LEN                     00000037    D
RX211$L_CSR                     00000032    D
RX31                            00002213 R  D  01
RX31$K_LEN                      00000032    D
RX32                            00002223 R  D  01
RX32$K_LEN                      00000032    D
RX50                            00002233 R  D  01
RX50$K_LEN                      00000032    D
SBIA                            00002259 R  D  01
SBIA$K_LEN                      00000032    D
```

```
SIZ...                  = 00000001    D
TE16                      0000229B R  D  01
TE16$K_LEN                00000032    D
TK50                      000022B5 R  D  01
TM03                      000022CF R  D  01
TM03$B_DRIVE              00000032    D
TM03$K_LEN                00000033    D
TM78                      000022FD R  D  01
TM78$B_DRIVE              00000032    D
TM78$K_LEN                00000033    D
TS04                      0000232B R  D  01
TS04$B_BR                 00000036    D
TS04$K_LEN                00000037    D
TS04$L_CSR                00000032    D
TS05                      00002382 R  D  01
TS05$B_BR                 00000032    D
TS05$K_LEN                00000033    D
TS11                      000023EC R  D  01
TS11$B_BR                 00000036    D
TS11$K_LEN                00000037    D
TS11$L_CSR                00000032    D
TU45                      00002443 R  D  01
TU45$K_LEN                00000032    D
TU58                      0000245D R  D  01
TU58$K_LEN                00000032    D
TU70                      00002477 R  D  01
TU70$K_LEN                00000032  G D
TU72                      00002491 R  D  01
TU72$K_LEN                00000032  G D
TU77                      000024AB R  D  01
TU77$K_LEN                00000032    D
TU78                      000024C5 R  D  01
TU78$K_LEN                00000032    D
TU80                      000024DF R  D  01
TU80$B_BR                 00000036    D
TU80$K_LEN                00000037    D
TU81                      0000254E R  D  01
TU81$B_BR                 00000036    D
TU81$K_LEN                00000037    D
TU81$L_CSR                00000032    D
UBE                       000025A5 R  D  01
UBE$K_LEN                 00000036    D
UBE$L_CSR                 00000032    D
UDA50                     000025E0 R  D  01
UDA50$B_BR                00000036    D
UDA50$B_BURST             00000037    D
UDA50$K_LEN               00000038    D
UDA50$L_UDAIP             00000032    D
UNA11                     00002649 R  D  01
UNA11$B_BR                00000032    D
UNA11$K_LEN               00000037    D
UNA11$L_CSR               00000033    D
VS100                     00002697 R  D  01
VS100$B_BR                00000036    D
VS100$K_LEN               00000037    D
VS100$L_CSR               00000032    D
VS125                     000026EF R  D  01
VS125$B_BR                00000036    D
VS125$K_LEN               00000037    D
VS125$L_CSR               00000032    D
VS300                     00002747 R  D  01
VS300$B_BR                00000036    D
VS300$K_LEN               00000037    D
VS300$L_CSR               00000032    D
VT100                     00002821 R  D  01
VT100$K_LEN               00000032    D
VT101                     0000283C R  D  01
VT101$K_LEN               00000032    D
VT102                     00002857 R  D  01
VT102$K_LEN               00000032    D
VT125                     00002872 R  D  01
VT125$K_LEN               00000032    D
VT131                     0000288D R  D  01
VT131$K_LEN               00000032    D
VT132                     000028A8 R  D  01
VT132$K_LEN               00000032    D
VT220                     000028C3 R  D  01
VT220$K_LEN               00000032    D
VT240                     000028DE R  D  01
VT240$K_LEN               00000032    D
VT50                      0000279F R  D  01
VT50$K_LEN                00000032    D
VT52                      000027B9 R  D  01
VT52$K_LEN                00000032    D
VT55                      000027D3 R  D  01
VT55$K_LEN                00000032    D
VT61                      000027ED R  D  01
VT61$K_LEN                00000032    D
VT71                      00002807 R  D  01
VT71$K_LEN                00000032    D
```

```
                                        +----------------+
                                        ! Psect synopsis !
                                        +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
. BLANK .                       000028F9  (10489.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$ABS$                           00000050  (    80.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000094	245 (3)	245 (3)
$$_$	=00000002	319 (4)	251 (4) 319 (4)
AA11K	00000254-R	249 (4)	245 (3)
AA11K$B_BR	00000032		249 (4)
AA11K$K_LEN	00000033		249 (4)
AD11K	000002A5-R	250 (4)	245 (3)
AD11K$B_BR	00000032		250 (4)
AD11K$K_LEN	00000033		250 (4)
AL_DEVTYP	00000000-R	245 (3)	
BIT...	=00000005	398 (4)	249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 258 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4) 264 (4) 265 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 278 (4) 279 (4) 280 (4) 281 (4) 282 (4) 283 (4) 284 (4) 285 (4) 287 (4) 288 (4) 289 (4) 290 (4) 291 (4) 292 (4) 293 (4) 294 (4) 295 (4) 296 (4) 297 (4) 298 (4) 299 (4) 300 (4) 301 (4) 302 (4) 303 (4) 304 (4) 305 (4) 306 (4) 307 (4) 308 (4) 309 (4) 310 (4) 311 (4) 312 (4) 313 (4) 314 (4) 315 (4) 316 (4) 317 (4) 318 (4) 319 (4) 320 (4) 321 (4) 322 (4) 323 (4) 324 (4) 325 (4) 326 (4) 327 (4) 328 (4) 329 (4) 330 (4) 331 (4) 332 (4) 333 (4) 334 (4) 335 (4) 336 (4) 337 (4) 338 (4) 339 (4) 340 (4) 341 (4) 342 (4) 343 (4) 344 (4) 345 (4) 346 (4) 347 (4) 348 (4) 349 (4) 350 (4) 351 (4) 352 (4) 353 (4) 354 (4) 355 (4) 356 (4) 357 (4) 358 (4) 359 (4) 360 (4) 361 (4) 362 (4) 363 (4) 364 (4) 365 (4) 366 (4) 367 (4) 368 (4) 369 (4) 370 (4) 371 (4) 372 (4) 373 (4) 374 (4) 375 (4) 376 (4) 377 (4) 378 (4) 379 (4) 380 (4) 381 (4) 382 (4) 383 (4) 384 (4) 385 (4) 386 (4) 387 (4) 388 (4) 389 (4) 390 (4) 391 (4) 392 (4) 393 (4) 394 (4) 395 (4) 396 (4) 397 (4) 398 (4)
CI$B_BR	00000033		252 (4) 253 (4)
CI$B_NODE	00000034		251 (4) 252 (4) 253 (4)
CI$B_TR	00000032		252 (4) 253 (4)

```
CI$K_LEN                  00000041                  251   (4)     252   (4)     253   (4)
CI$L_FUNC                 00000035                  251   (4)     252   (4)     253   (4)
CI$L_INDEX                0000003D                  251   (4)
CI$L_MAINT_ID             00000039                  251   (4)     252   (4)     253   (4)
CI750                     00000412-R   252   (4)    245   (3)
CI780                     00000491-R   253   (4)    245   (3)
CI_NODE                   000002F6-R   251   (4)    245   (3)
CONSOLE                   00000518-R   254   (4)    245   (3)
CONSOLE$K_LEN             00000032                  254   (4)
CR11                      00000526-R   255   (4)    245   (3)
CR11$B_BR                 00000036                  255   (4)
CR11$K_LEN                00000037                  255   (4)
CR11$L_CSR                00000032                  255   (4)
DEQNA                     0000057D-R   256   (4)    245   (3)
DEQNA$K_LEN               00000036                  256   (4)
DEQNA$L_CSR               00000032                  256   (4)
DHU11                     000005A5-R   257   (4)    245   (3)
DHV11                     000005F3-R   258   (4)    245   (3)
DISK                      00000641-R   259   (4)    245   (3)
DISK$K_LEN                00000032                  259   (4)
DL11                      00000651-R   260   (4)    245   (3)
DL11$B_BR                 00000036                  260   (4)
DL11$K_LEN                00000037                  260   (4)
DL11$L_CSR                00000032                  260   (4)
DLVJ1                     0000069E-R   261   (4)    245   (3)
DLVJ1$B_FLAGS             00000036                  261   (4)
DLVJ1$K_LEN               00000037                  261   (4)
DLVJ1$L_CSR               00000032                  261   (4)
DLVJ1$V_EVEPAR           =00000003                  261   (4)
DLVJ1$V_NDATA            =00000000                  261   (4)
DLVJ1$V_NSTOP            =00000001                  261   (4)
DLVJ1$V_PARITY           =00000002                  261   (4)
DMC11                     00000752-R   262   (4)    245   (3)
DMC11$B_BR                00000036                  262   (4)
DMC11$K_LEN               00000037                  262   (4)
DMC11$L_CSR               00000032                  262   (4)
DMF32                     000007A0-R   263   (4)    245   (3)
DMF32$B_BR                00000034                  263   (4)
DMF32$B_COMMON            00000032                  263   (4)
DMF32$K_LEN               00000039                  263   (4)     266   (4)
DMF32$L_CSR               00000035                  263   (4)
DMF32A                    000007F3-R   264   (4)    245   (3)
DMF32A$B_ACTIV            00000033                  264   (4)
DMF32A$B_BR               00000032                  264   (4)
DMF32A$B_JUMP             00000037                  264   (4)
DMF32A$B_SPEED            00000034                  264   (4)
DMF32A$K_LEN              00000038                  264   (4)
DMF32A$W_FLAGS            00000035                  264   (4)
DMF32P                    00000915-R   265   (4)    245   (3)
DMF32P$B_BR               00000034                  265   (4)
DMF32P$B_COMMON           00000032                  265   (4)
DMF32P$B_FLAGS            00000033                  265   (4)
DMF32P$K_LEN              00000039                  265   (4)
DMF32P$L_CSR              00000035                  265   (4)
DMF32S                    0000098F-R   266   (4)    245   (3)
DMF32S$B_BR               00000034                  266   (4)
DMF32S$B_FLAGS            00000033                  266   (4)
```

```
DMF32S$L_CSR                    00000035                        266     (4)
DMP11                           00000A03-R      267     (4)     245     (3)
DMP11$B_BR                      00000034                        267     (4)
DMP11$B_COMMON                  00000032                        267     (4)
DMP11$B_FLAGS                   00000033                        267     (4)
DMP11$K_LEN                     00000039                        267     (4)
DMP11$L_CSR                     00000035                        267     (4)
DMR11                           00000ACC-R      268     (4)     245     (3)
DMR11$B_BR                      00000036                        268     (4)
DMR11$K_LEN                     00000037                        268     (4)
DMR11$L_CSR                     00000032                        268     (4)
DMZ32                           00000B1A-R      269     (4)     245     (3)
DR11B                           00000C78-R      270     (4)     245     (3)
DR11B$B_BR                      00000036                        270     (4)
DR11B$K_LEN                     00000037                        270     (4)
DR11B$L_CSR                     00000032                        270     (4)
DR11C                           00000CC6-R      271     (4)     245     (3)
DR11K                           00000D0F-R      272     (4)     245     (3)
DR11K$B_BR                      00000032                        272     (4)
DR11K$K_LEN                     00000033                        272     (4)
DR11W                           00000D60-R      273     (4)     245     (3)
DR11W$B_BR                      00000036                        273     (4)
DR11W$K_LEN                     00000037                        273     (4)
DR11W$L_CSR                     00000032                        273     (4)
DR750                           00000DAE-R      274     (4)     245     (3)
DR750$B_BR                      00000033                        274     (4)
DR750$B_CONFIG                  00000035                        274     (4)
DR750$B_SELF                    00000034                        274     (4)
DR750$B_SLOT                    00000032                        274     (4)
DR750$B_UUT                     00000036                        274     (4)
DR750$K_LEN                     00000037                        274     (4)
DR780                           00000E39-R      275     (4)     245     (3)
DR780$B_BR                      00000033                        275     (4)
DR780$B_CONFIG                  00000035                        275     (4)
DR780$B_SELF                    00000034                        275     (4)
DR780$B_TR                      00000032                        275     (4)
DR780$B_UUT                     00000036                        275     (4)
DR780$K_LEN                     00000037                        275     (4)
DS$GA_PTDESC                    00000000-R      216     (3)
DUP11                           00000ECC-R      276     (4)     245     (3)
DUP11$B_BR                      00000036                        276     (4)
DUP11$K_LEN                     00000037                        276     (4)
DUP11$L_CSR                     00000032                        276     (4)
DV11                            00000F1A-R      277     (4)
DV11$B_BR                       00000036                        277     (4)
DV11$K_LEN                      00000037                        277     (4)
DV11$L_CSR                      00000032                        277     (4)
DW730                           00000F67-R      278     (4)     245     (3)
DW730$K_LEN                     00000032                        278     (4)
DW750                           00000F96-R      279     (4)     245     (3)
DW750$K_LEN                     00000032                        279     (4)
DW780                           00000FDA-R      280     (4)     245     (3)
DW780$B_BR                      00000033                        280     (4)
DW780$B_TR                      00000032                        280     (4)
DW780$K_LEN                     00000034                        280     (4)
DX20                            0000104E-R      281     (4)     245     (3)
DX20$B_DRIVE                    00000032                        281     (4)
```

DX20$K_LEN	00000033			281 (4)
DZ11	0000107C-R	282	(4)	245 (3)
DZ11$B_BR	00000036			282 (4)
DZ11$B_MTYPE	00000037			282 (4)
DZ11$K_LEN	00000038			282 (4)
DZ11$L_CSR	00000032			282 (4)
DZ32	000010E5-R	283	(4)	245 (3)
DZ32$B_ACTIV	00000033			283 (4)
DZ32$B_BRLVL	00000032			283 (4)
DZ32$B_SPEED	00000034			283 (4)
DZ32$W_FLAGS	00000035			283 (4)
DZ32$_LEN	00000037			283 (4)
DZV11	000011D6-R	284	(4)	245 (3)
DZV11$K_LEN	00000036			284 (4)
DZV11$L_CSR	00000032			284 (4)
HP$A_DEVICE	00000018			249 (4) 250 (4) 251 (4) 252 (4)
				253 (4) 255 (4) 256 (4) 257 (4)
				258 (4) 260 (4) 261 (4) 262 (4)
				263 (4) 264 (4) 265 (4) 266 (4)
				267 (4) 268 (4) 269 (4) 270 (4)
				271 (4) 272 (4) 273 (4) 274 (4)
				275 (4) 276 (4) 277 (4) 278 (4)
				279 (4) 280 (4) 281 (4) 282 (4)
				283 (4) 284 (4) 285 (4) 287 (4)
				288 (4) 292 (4) 293 (4) 302 (4)
				312 (4) 317 (4) 318 (4) 319 (4)
				320 (4) 321 (4) 322 (4) 323 (4)
				325 (4) 326 (4) 327 (4) 335 (4)
				340 (4) 341 (4) 344 (4) 347 (4)
				348 (4) 349 (4) 350 (4) 351 (4)
				352 (4) 353 (4) 354 (4) 355 (4)
				356 (4) 357 (4) 359 (4) 364 (4)
				367 (4) 368 (4) 369 (4) 370 (4)
				371 (4) 378 (4) 379 (4) 380 (4)
				381 (4) 382 (4) 383 (4) 384 (4)
				385 (4)
HP$A_DVA	0000001C			251 (4) 252 (4) 253 (4) 278 (4)
				279 (4) 280 (4) 319 (4) 320 (4)
				340 (4) 341 (4) 364 (4)
HP$B_BI_NODE	00000033			320 (4)
HP$B_BI_NUM	00000038			320 (4)
HP$B_DHUV_BR	00000038			257 (4) 258 (4)
HP$B_DMZ32_BR	00000032			269 (4)
HP$B_DMZ32_MODEM	00000039			269 (4)
HP$B_DMZ32_OCTET0	00000033			269 (4)
HP$B_DMZ32_OCTET1	00000034			269 (4)
HP$B_DMZ32_OCTET2	00000035			269 (4)
HP$B_DMZ32_SPEED	00000036			269 (4)
HP$B_DR11C_BR	00000032			271 (4)
HP$B_DRIVE	0000000B			251 (4) 279 (4) 280 (4) 281 (4)
				322 (4) 323 (4) 340 (4) 348 (4)
				349 (4) 350 (4) 351 (4) 352 (4)
				353 (4) 354 (4) 364 (4) 367 (4)
				368 (4)
HP$B_FLAGS	0000000A			255 (4) 294 (4) 295 (4) 296 (4)
				297 (4) 298 (4) 299 (4) 300 (4)
				301 (4) 303 (4) 304 (4) 305 (4)

		306	(4)	307	(4)	308	(4)	309	(4)	
		310	(4)	311	(4)	313	(4)	314	(4)	
		315	(4)	316	(4)	317	(4)	323	(4)	
		328	(4)	329	(4)	330	(4)	331	(4)	
		332	(4)	333	(4)	334	(4)	336	(4)	
		337	(4)	338	(4)	339	(4)	342	(4)	
		343	(4)	345	(4)	346	(4)	348	(4)	
		349	(4)	350	(4)	351	(4)	352	(4)	
		353	(4)	354	(4)	356	(4)	358	(4)	
		362	(4)	365	(4)	366	(4)	369	(4)	
		370	(4)	371	(4)	372	(4)	373	(4)	
		374	(4)	375	(4)	376	(4)	377	(4)	
		378	(4)	379	(4)	383	(4)	384	(4)	
		385	(4)	386	(4)	387	(4)	388	(4)	
		389	(4)	390	(4)	391	(4)	392	(4)	
		393	(4)	394	(4)	395	(4)	396	(4)	
		397	(4)	398	(4)					
HP$B_LESI_BR	00000036	302	(4)							
HP$B_RUX50_BR	00000032	357	(4)							
HP$B_TU80_BR	00000036	378	(4)							
HP$B_UDA50_BR	00000036	381	(4)							
HP$B_UDA50_BURST	00000037	381	(4)							
HP$K_DHUV_LEN	00000039	257	(4)	258	(4)					
HP$K_DMZ32_LEN	0000003A	269	(4)							
HP$K_DR11C_LEN	00000033	271	(4)							
HP$K_LESI_LEN	00000037	302	(4)							
HP$K_NBIA_LEN	00000038	319	(4)							
HP$K_NBIB_LEN	00000039	320	(4)							
HP$K_RUX50_LEN	00000033	357	(4)							
HP$K_TK50_LEN	00000032	366	(4)							
HP$K_UDA50_LEN	00000038	381	(4)							
HP$L_DHUV_CSR	00000032	257	(4)	258	(4)					
HP$L_LESI_IP	00000032	302	(4)							
HP$L_TU80_CSR	00000032	378	(4)							
HP$L_UDA50_UDAIP	00000032	381	(4)							
HP$M_ALLOC	=00000001	255	(4)	294	(4)	295	(4)	296	(4)	
		297	(4)	298	(4)	299	(4)	300	(4)	
		301	(4)	303	(4)	304	(4)	305	(4)	
		306	(4)	307	(4)	308	(4)	309	(4)	
		310	(4)	311	(4)	313	(4)	314	(4)	
		315	(4)	316	(4)	317	(4)	323	(4)	
		328	(4)	329	(4)	330	(4)	331	(4)	
		332	(4)	333	(4)	334	(4)	336	(4)	
		337	(4)	338	(4)	339	(4)	342	(4)	
		343	(4)	345	(4)	346	(4)	348	(4)	
		349	(4)	350	(4)	351	(4)	352	(4)	
		353	(4)	354	(4)	356	(4)	358	(4)	
		362	(4)	365	(4)	366	(4)	369	(4)	
		370	(4)	371	(4)	372	(4)	373	(4)	
		374	(4)	375	(4)	376	(4)	377	(4)	
		378	(4)	379	(4)	383	(4)	384	(4)	
		385	(4)	386	(4)	387	(4)	388	(4)	
		389	(4)	390	(4)	391	(4)	392	(4)	
		393	(4)	394	(4)	395	(4)	396	(4)	
		397	(4)	398	(4)					
HP$W_DHUV_VEC	00000036	257	(4)	258	(4)					
HP$W_DMZ32_LOOP	00000037	269	(4)							

```
HP$W_VECTOR                 00000024                 249   (4)    250   (4)    252   (4)    253   (4)
                                                     255   (4)    260   (4)    261   (4)    262   (4)
                                                     263   (4)    264   (4)    265   (4)    266   (4)
                                                     267   (4)    268   (4)    269   (4)    270   (4)
                                                     271   (4)    272   (4)    273   (4)    274   (4)
                                                     275   (4)    276   (4)    277   (4)    278   (4)
                                                     279   (4)    280   (4)    282   (4)    283   (4)
                                                     284   (4)    285   (4)    292   (4)    293   (4)
                                                     302   (4)    312   (4)    317   (4)    318   (4)
                                                     319   (4)    321   (4)    326   (4)    327   (4)
                                                     335   (4)    340   (4)    341   (4)    344   (4)
                                                     347   (4)    356   (4)    357   (4)    359   (4)
                                                     364   (4)    369   (4)    370   (4)    371   (4)
                                                     378   (4)    379   (4)    380   (4)    381   (4)
                                                     382   (4)    383   (4)    384   (4)    385   (4)
IEU11A                      00001213-R   285   (4)   245   (3)
IEU11A$B_BR                 00000032                 285   (4)
IEU11A$K_LEN                00000033                 285   (4)
KA$B_ACC_TYPE               00000042                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$K_LEN                    00000050                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$L_FLAGS                  00000032                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_CHAR                  =00000100                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_CRC                   =00000010                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_D_FLOAT               =00000002                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_EDIT                  =00000080                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_F_FLOAT               =00000001                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_G_FLOAT               =00000004                 287   (4)
KA$M_H_FLOAT               =00000008                 287   (4)
KA$M_PACKED                =00000020                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_PACK_MUL              =00000040                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$M_TODR                  =00010000                 289   (4)    290   (4)    291   (4)
KA$V_G_FLOAT               =00000002                 288   (4)    289   (4)    290   (4)    291   (4)
KA$V_H_FLOAT               =00000003                 288   (4)    289   (4)    290   (4)    291   (4)
KA$V_SB_ERR                =00000019                 287   (4)
KA$V_TODR                  =00000010                 287   (4)    288   (4)
KA$V_WCS_LD                =00000018                 287   (4)
KA$W_LATENCY                0000003E                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$W_MEM_SIZE               00000044                 287   (4)
KA$W_PROGRESS               00000040                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA$W_WCS                    00000036                 287   (4)    288   (4)    289   (4)    290   (4)
                                                     291   (4)
KA730                       0000125E-R   287   (4)   245   (3)
KA750                       00001339-R   288   (4)   245   (3)
KA780                       00001405-R   289   (4)   245   (3)
KA785                       00001556-R   291   (4)   245   (3)
```

KA86	000014AE-R	290	(4)	245	(3)
KMC11	000015FF-R	292	(4)	245	(3)
KMC11$B_BR	00000036			292	(4)
KMC11$K_LEN	00000037			292	(4)
KMC11$L_CSR	00000032			292	(4)
KW11K	0000164D-R	293	(4)	245	(3)
KW11K$B_BR	00000032			293	(4)
KW11K$K_LEN	00000033			293	(4)
LA100	000016B8-R	295	(4)	245	(3)
LA100$K_LEN	00000032			295	(4)
LA12	0000169E-R	294	(4)	245	(3)
LA12$K_LEN	00000032			294	(4)
LA120	000016D3-R	296	(4)	245	(3)
LA120$K_LEN	00000032			296	(4)
LA180	000016EE-R	297	(4)	245	(3)
LA180$K_LEN	00000032			297	(4)
LA210	00001757-R	301	(4)	245	(3)
LA210$K_LEN	00000032			301	(4)
LA34	00001709-R	298	(4)	245	(3)
LA34$K_LEN	00000032			298	(4)
LA36	00001723-R	299	(4)	245	(3)
LA36$K_LEN	00000032			299	(4)
LA38	0000173D-R	300	(4)	245	(3)
LA38$K_LEN	00000032			300	(4)
LESI	00001772-R	302	(4)	245	(3)
LG01	000017BE-R	303	(4)	245	(3)
LG01$K_LEN	00000032			303	(4)
LG02	000017D8-R	304	(4)	245	(3)
LG02$K_LEN	00000032			304	(4)
LG03	000017F2-R	305	(4)	245	(3)
LG03$K_LEN	00000032			305	(4)
LN01	0000180C-R	306	(4)	245	(3)
LN01$K_LEN	00000032			306	(4)
LN03	00001826-R	307	(4)	245	(3)
LN03$K_LEN	00000032			307	(4)
LP04	0000183B-R	308	(4)	245	(3)
LP04$K_LEN	00000032			308	(4)
LP05	00001855-R	309	(4)	245	(3)
LP05$K_LEN	00000032			309	(4)
LP06	0000186F-R	310	(4)	245	(3)
LP06$K_LEN	00000032			310	(4)
LP07	00001889-R	311	(4)	245	(3)
LP07$K_LEN	00000032			311	(4)
LP11	000018A3-R	312	(4)	245	(3)
LP11$B_BR	00000036			312	(4)
LP11$K_LEN	00000037			312	(4)
LP11$L_CSR	00000032			312	(4)
LP14	000018F0-R	313	(4)	245	(3)
LP14$K_LEN	00000032			313	(4)
LP25	0000190A-R	314	(4)	245	(3)
LP25$K_LEN	00000032			314	(4)
LP26	00001924-R	315	(4)	245	(3)
LP26$K_LEN	00000032			315	(4)
LP27	0000193E-R	316	(4)	245	(3)
LP27$K_LEN	00000032			316	(4)
LPA11K	00001958-R	317	(4)	245	(3)
LPA11K$B_BR	00000036			317	(4)

```
LPA11K$K_LEN                    00000037                        317     (4)
LPA11K$L_CSR                    00000032                        317     (4)
LUA11                           000019B1-R      318     (4)     245     (3)
LUA11$B_BR                      00000032                        318     (4)
LUA11$K_LEN                     00000037                        318     (4)
LUA11$L_CSR                     00000033                        318     (4)
MA780                           00001AD2-R      321     (4)     245     (3)
MA780$B_BR                      00000033                        321     (4)
MA780$B_MPM                     00000034                        321     (4)
MA780$B_PORT                    00000035                        321     (4)
MA780$B_TR                      00000032                        321     (4)
MA780$K_LEN                     00000036                        321     (4)
MBE                             00001B4D-R      322     (4)     245     (3)
MBE$K_LEN                       00000032                        322     (4)
ML11                            00001B61-R      323     (4)     245     (3)
ML11$B_ARRAY                    00000032                        323     (4)
ML11$B_CHIP                     00000033                        323     (4)
ML11$K_LEN                      00000034                        323     (4)
MS750                           00001BC3-R      324     (4)     245     (3)
MS750$K_LEN                     00000032                        324     (4)
MS780                           00001BD4-R      325     (4)     245     (3)
MS780$B_ARRAYS                  00000033                        325     (4)
MS780$B_TR                      00000032                        325     (4)
MS780$K_LEN                     00000034                        325     (4)
NBIA                            000019FF-R      319     (4)     245     (3)
NBIB                            00001A6F-R      320     (4)     245     (3)
PCL11                           00001C24-R      326     (4)     245     (3)
PCL11$B_BR                      00000036                        326     (4)
PCL11$K_LEN                     00000037                        326     (4)
PCL11$L_CSR                     00000032                        326     (4)
PD$_ADD                         =0000008A       398     (4)     251     (4)     279     (4)     364     (4)
PD$_CASE                        =0000008C       398     (4)     251     (4)     319     (4)
PD$_COMPLEMENT                  =00000089       398     (4)     279     (4)
PD$_DECIMAL                     =00000082       398     (4)     249     (4)     250     (4)     251     (4)     252     (4)
                                                                253     (4)     255     (4)     257     (4)     258     (4)
                                                                260     (4)     262     (4)     263     (4)     264     (4)
                                                                265     (4)     266     (4)     267     (4)     268     (4)
                                                                269     (4)     270     (4)     271     (4)     272     (4)
                                                                273     (4)     274     (4)     275     (4)     276     (4)
                                                                277     (4)     280     (4)     281     (4)     282     (4)
                                                                285     (4)     287     (4)     288     (4)     289     (4)
                                                                290     (4)     291     (4)     292     (4)     293     (4)
                                                                302     (4)     312     (4)     317     (4)     318     (4)
                                                                319     (4)     320     (4)     321     (4)     323     (4)
                                                                325     (4)     326     (4)     327     (4)     340     (4)
                                                                341     (4)     344     (4)     347     (4)     356     (4)
                                                                357     (4)     359     (4)     367     (4)     368     (4)
                                                                369     (4)     370     (4)     371     (4)     378     (4)
                                                                379     (4)     381     (4)     382     (4)     383     (4)
                                                                384     (4)     385     (4)
PD$_END                         =00000081       398     (4)     249     (4)     250     (4)     251     (4)     252     (4)
                                                                253     (4)     254     (4)     255     (4)     256     (4)
                                                                257     (4)     258     (4)     259     (4)     260     (4)
                                                                261     (4)     262     (4)     263     (4)     264     (4)
                                                                265     (4)     266     (4)     267     (4)     268     (4)
                                                                269     (4)     270     (4)     271     (4)     272     (4)
                                                                273     (4)     274     (4)     275     (4)     276     (4)
```

			277 (4) 278 (4) 279 (4) 280 (4)
			281 (4) 282 (4) 283 (4) 284 (4)
			285 (4) 287 (4) 288 (4) 289 (4)
			290 (4) 291 (4) 292 (4) 293 (4)
			294 (4) 295 (4) 296 (4) 297 (4)
			298 (4) 299 (4) 300 (4) 301 (4)
			302 (4) 303 (4) 304 (4) 305 (4)
			306 (4) 307 (4) 308 (4) 309 (4)
			310 (4) 311 (4) 312 (4) 313 (4)
			314 (4) 315 (4) 316 (4) 317 (4)
			318 (4) 319 (4) 320 (4) 321 (4)
			322 (4) 323 (4) 324 (4) 325 (4)
			326 (4) 327 (4) 328 (4) 329 (4)
			330 (4) 331 (4) 332 (4) 333 (4)
			334 (4) 335 (4) 336 (4) 337 (4)
			338 (4) 339 (4) 340 (4) 341 (4)
			342 (4) 343 (4) 344 (4) 345 (4)
			346 (4) 347 (4) 348 (4) 349 (4)
			350 (4) 351 (4) 352 (4) 353 (4)
			354 (4) 355 (4) 356 (4) 357 (4)
			358 (4) 359 (4) 360 (4) 361 (4)
			362 (4) 363 (4) 364 (4) 365 (4)
			366 (4) 367 (4) 368 (4) 369 (4)
			370 (4) 371 (4) 372 (4) 373 (4)
			374 (4) 375 (4) 376 (4) 377 (4)
			378 (4) 379 (4) 380 (4) 381 (4)
			382 (4) 383 (4) 384 (4) 385 (4)
			386 (4) 387 (4) 388 (4) 389 (4)
			390 (4) 391 (4) 392 (4) 393 (4)
			394 (4) 395 (4) 396 (4) 397 (4)
			398 (4)
PD$_EPB	=0000008E	398 (4)	
PD$_FETCH	=00000087	398 (4)	251 (4) 252 (4) 253 (4) 279 (4)
			280 (4) 319 (4) 320 (4) 322 (4)
			323 (4) 340 (4) 341 (4) 348 (4)
			349 (4) 350 (4) 351 (4) 352 (4)
			353 (4) 354 (4) 364 (4)
PD$_HEXADECIMAL	=00000084	398 (4)	287 (4) 288 (4) 289 (4) 290 (4)
			291 (4) 320 (4)
PD$_LITERAL	=00000086	398 (4)	251 (4) 252 (4) 253 (4) 255 (4)
			263 (4) 265 (4) 267 (4) 269 (4)
			274 (4) 275 (4) 278 (4) 279 (4)
			280 (4) 287 (4) 288 (4) 289 (4)
			290 (4) 291 (4) 294 (4) 295 (4)
			296 (4) 297 (4) 298 (4) 299 (4)
			300 (4) 301 (4) 303 (4) 304 (4)
			305 (4) 306 (4) 307 (4) 308 (4)
			309 (4) 310 (4) 311 (4) 313 (4)
			314 (4) 315 (4) 316 (4) 317 (4)
			319 (4) 321 (4) 323 (4) 325 (4)
			327 (4) 328 (4) 329 (4) 330 (4)
			331 (4) 332 (4) 333 (4) 334 (4)
			335 (4) 336 (4) 337 (4) 338 (4)
			339 (4) 340 (4) 341 (4) 342 (4)
			343 (4) 345 (4) 346 (4) 348 (4)
			349 (4) 350 (4) 351 (4) 352 (4)
			353 (4) 354 (4) 356 (4) 358 (4)

```
                                                  362   (4)   364   (4)   365   (4)   366   (4)
                                                  369   (4)   370   (4)   371   (4)   372   (4)
                                                  373   (4)   374   (4)   375   (4)   376   (4)
                                                  377   (4)   378   (4)   379   (4)   383   (4)
                                                  384   (4)   385   (4)   386   (4)   387   (4)
                                                  388   (4)   389   (4)   390   (4)   391   (4)
                                                  392   (4)   393   (4)   394   (4)   395   (4)
                                                  396   (4)   397   (4)   398   (4)
PD$_LOGICAL                =0000008B     398   (4)   261   (4)   264   (4)   269   (4)   274   (4)
                                                  275   (4)   287   (4)   288   (4)   289   (4)
                                                  290   (4)   291   (4)
PD$_NAME                   =0000008D     398   (4)   249   (4)   250   (4)   251   (4)   252   (4)
                                                  253   (4)   255   (4)   256   (4)   257   (4)
                                                  258   (4)   259   (4)   260   (4)   261   (4)
                                                  262   (4)   264   (4)   265   (4)   266   (4)
                                                  267   (4)   268   (4)   269   (4)   270   (4)
                                                  271   (4)   272   (4)   273   (4)   274   (4)
                                                  275   (4)   276   (4)   277   (4)   278   (4)
                                                  279   (4)   280   (4)   281   (4)   282   (4)
                                                  283   (4)   284   (4)   285   (4)   287   (4)
                                                  288   (4)   289   (4)   290   (4)   291   (4)
                                                  292   (4)   293   (4)   294   (4)   295   (4)
                                                  296   (4)   297   (4)   298   (4)   299   (4)
                                                  300   (4)   301   (4)   302   (4)   303   (4)
                                                  304   (4)   305   (4)   306   (4)   308   (4)
                                                  309   (4)   310   (4)   311   (4)   312   (4)
                                                  313   (4)   314   (4)   315   (4)   316   (4)
                                                  317   (4)   318   (4)   319   (4)   320   (4)
                                                  321   (4)   323   (4)   324   (4)   325   (4)
                                                  327   (4)   328   (4)   329   (4)   330   (4)
                                                  331   (4)   332   (4)   333   (4)   334   (4)
                                                  335   (4)   336   (4)   337   (4)   338   (4)
                                                  339   (4)   340   (4)   341   (4)   342   (4)
                                                  343   (4)   344   (4)   345   (4)   346   (4)
                                                  347   (4)   348   (4)   349   (4)   350   (4)
                                                  351   (4)   352   (4)   353   (4)   354   (4)
                                                  355   (4)   356   (4)   357   (4)   358   (4)
                                                  359   (4)   360   (4)   361   (4)   363   (4)
                                                  364   (4)   365   (4)   366   (4)   367   (4)
                                                  368   (4)   369   (4)   370   (4)   371   (4)
                                                  372   (4)   373   (4)   374   (4)   375   (4)
                                                  376   (4)   377   (4)   378   (4)   379   (4)
                                                  380   (4)   381   (4)   382   (4)   383   (4)
                                                  384   (4)   385   (4)   386   (4)   387   (4)
                                                  388   (4)   389   (4)   390   (4)   391   (4)
                                                  392   (4)   393   (4)   394   (4)   395   (4)
                                                  396   (4)   397   (4)   398   (4)
PD$_OCTAL                  =00000083     398   (4)   249   (4)   250   (4)   255   (4)   256   (4)
                                                  257   (4)   258   (4)   260   (4)   261   (4)
                                                  262   (4)   263   (4)   264   (4)   265   (4)
                                                  266   (4)   267   (4)   268   (4)   269   (4)
                                                  270   (4)   271   (4)   272   (4)   273   (4)
                                                  276   (4)   277   (4)   282   (4)   283   (4)
                                                  284   (4)   285   (4)   292   (4)   293   (4)
                                                  302   (4)   312   (4)   317   (4)   318   (4)
                                                  326   (4)   344   (4)   347   (4)   355   (4)
                                                  356   (4)   357   (4)   359   (4)   369   (4)
```

			370 (4)	371 (4) 378 (4) 379 (4)
			380 (4)	381 (4) 382 (4) 383 (4)
			384 (4)	385 (4)
PD$_START	=00000080	398 (4)	249 (4)	250 (4) 251 (4) 252 (4)
			253 (4)	254 (4) 255 (4) 256 (4)
			257 (4)	258 (4) 259 (4) 260 (4)
			261 (4)	262 (4) 263 (4) 264 (4)
			265 (4)	266 (4) 267 (4) 268 (4)
			269 (4)	270 (4) 271 (4) 272 (4)
			273 (4)	274 (4) 275 (4) 276 (4)
			277 (4)	278 (4) 279 (4) 280 (4)
			281 (4)	282 (4) 283 (4) 284 (4)
			285 (4)	287 (4) 288 (4) 289 (4)
			290 (4)	291 (4) 292 (4) 293 (4)
			294 (4)	295 (4) 296 (4) 297 (4)
			298 (4)	299 (4) 300 (4) 301 (4)
			302 (4)	303 (4) 304 (4) 305 (4)
			306 (4)	307 (4) 308 (4) 309 (4)
			310 (4)	311 (4) 312 (4) 313 (4)
			314 (4)	315 (4) 316 (4) 317 (4)
			318 (4)	319 (4) 320 (4) 321 (4)
			322 (4)	323 (4) 324 (4) 325 (4)
			326 (4)	327 (4) 328 (4) 329 (4)
			330 (4)	331 (4) 332 (4) 333 (4)
			334 (4)	335 (4) 336 (4) 337 (4)
			338 (4)	339 (4) 340 (4) 341 (4)
			342 (4)	343 (4) 344 (4) 345 (4)
			346 (4)	347 (4) 348 (4) 349 (4)
			350 (4)	351 (4) 352 (4) 353 (4)
			354 (4)	355 (4) 356 (4) 357 (4)
			358 (4)	359 (4) 360 (4) 361 (4)
			362 (4)	363 (4) 364 (4) 365 (4)
			366 (4)	367 (4) 368 (4) 369 (4)
			370 (4)	371 (4) 372 (4) 373 (4)
			374 (4)	375 (4) 376 (4) 377 (4)
			378 (4)	379 (4) 380 (4) 381 (4)
			382 (4)	383 (4) 384 (4) 385 (4)
			386 (4)	387 (4) 388 (4) 389 (4)
			390 (4)	391 (4) 392 (4) 393 (4)
			394 (4)	395 (4) 396 (4) 397 (4)
			398 (4)	
PD$_STORE	=00000088	398 (4)	249 (4)	250 (4) 251 (4) 252 (4)
			253 (4)	255 (4) 256 (4) 257 (4)
			258 (4)	260 (4) 261 (4) 262 (4)
			263 (4)	264 (4) 265 (4) 266 (4)
			267 (4)	268 (4) 269 (4) 270 (4)
			271 (4)	272 (4) 273 (4) 274 (4)
			275 (4)	276 (4) 277 (4) 278 (4)
			279 (4)	280 (4) 281 (4) 282 (4)
			283 (4)	284 (4) 285 (4) 287 (4)
			288 (4)	289 (4) 290 (4) 291 (4)
			292 (4)	293 (4) 294 (4) 295 (4)
			296 (4)	297 (4) 298 (4) 299 (4)
			300 (4)	301 (4) 302 (4) 303 (4)
			304 (4)	305 (4) 306 (4) 307 (4)
			308 (4)	309 (4) 310 (4) 311 (4)
			312 (4)	313 (4) 314 (4) 315 (4)

				316 (4) 317 (4) 318 (4) 319 (4)
				320 (4) 321 (4) 322 (4) 323 (4)
				325 (4) 326 (4) 327 (4) 328 (4)
				329 (4) 330 (4) 331 (4) 332 (4)
				333 (4) 334 (4) 335 (4) 336 (4)
				337 (4) 338 (4) 339 (4) 340 (4)
				341 (4) 342 (4) 343 (4) 344 (4)
				345 (4) 346 (4) 347 (4) 348 (4)
				349 (4) 350 (4) 351 (4) 352 (4)
				353 (4) 354 (4) 355 (4) 356 (4)
				357 (4) 358 (4) 359 (4) 362 (4)
				364 (4) 365 (4) 366 (4) 367 (4)
				368 (4) 369 (4) 370 (4) 371 (4)
				372 (4) 373 (4) 374 (4) 375 (4)
				376 (4) 377 (4) 378 (4) 379 (4)
				380 (4) 381 (4) 382 (4) 383 (4)
				384 (4) 385 (4) 386 (4) 387 (4)
				388 (4) 389 (4) 390 (4) 391 (4)
				392 (4) 393 (4) 394 (4) 395 (4)
				396 (4) 397 (4) 398 (4)
PD$_STRING	=00000085	398	(4)	251 (4) 261 (4) 264 (4) 265 (4)
				266 (4) 267 (4) 269 (4) 274 (4)
				275 (4) 282 (4) 323 (4)
PTD$M_CONTROLLER	=00000002	398	(4)	249 (4) 250 (4) 251 (4) 252 (4)
				253 (4) 254 (4) 255 (4) 256 (4)
				257 (4) 258 (4) 259 (4) 260 (4)
				261 (4) 262 (4) 263 (4) 264 (4)
				265 (4) 266 (4) 267 (4) 268 (4)
				269 (4) 270 (4) 271 (4) 272 (4)
				273 (4) 274 (4) 275 (4) 276 (4)
				277 (4) 278 (4) 279 (4) 280 (4)
				281 (4) 282 (4) 283 (4) 284 (4)
				285 (4) 287 (4) 288 (4) 289 (4)
				290 (4) 291 (4) 292 (4) 293 (4)
				294 (4) 295 (4) 296 (4) 297 (4)
				298 (4) 299 (4) 300 (4) 301 (4)
				302 (4) 303 (4) 304 (4) 305 (4)
				306 (4) 307 (4) 308 (4) 309 (4)
				310 (4) 311 (4) 312 (4) 313 (4)
				314 (4) 315 (4) 316 (4) 317 (4)
				318 (4) 319 (4) 320 (4) 321 (4)
				322 (4) 323 (4) 324 (4) 325 (4)
				326 (4) 327 (4) 328 (4) 329 (4)
				330 (4) 331 (4) 332 (4) 333 (4)
				334 (4) 335 (4) 336 (4) 337 (4)
				338 (4) 339 (4) 340 (4) 341 (4)
				342 (4) 343 (4) 344 (4) 345 (4)
				346 (4) 347 (4) 348 (4) 349 (4)
				350 (4) 351 (4) 352 (4) 353 (4)
				354 (4) 355 (4) 356 (4) 357 (4)
				358 (4) 359 (4) 360 (4) 361 (4)
				362 (4) 363 (4) 364 (4) 365 (4)
				366 (4) 367 (4) 368 (4) 369 (4)
				370 (4) 371 (4) 372 (4) 373 (4)
				374 (4) 375 (4) 376 (4) 377 (4)
				378 (4) 379 (4) 380 (4) 381 (4)
				382 (4) 383 (4) 384 (4) 385 (4)

			386	(4)
			390	(4)
			394	(4)
			398	(4)
PTD$M_DEVICE	=00000003	398	249	(4)
			255	(4)
			267	(4)
			273	(4)
			277	(4)
			318	(4)
			349	(4)
			353	(4)
			369	(4)
			378	(4)
PTD$M_ENDDEVICE	=0000001B	398	294	(4)
			298	(4)
			303	(4)
			308	(4)
			313	(4)
			328	(4)
			333	(4)
			338	(4)
			345	(4)
			361	(4)
			374	(4)
			383	(4)
			387	(4)
			391	(4)
			395	(4)
PTD$M_INHERIT	=00000018	398	249	(4)
			253	(4)
			257	(4)
			261	(4)
			265	(4)
			269	(4)
			273	(4)
			277	(4)
			281	(4)
			285	(4)
			290	(4)
			294	(4)
			298	(4)
			302	(4)
			306	(4)
			310	(4)
			314	(4)
			318	(4)
			322	(4)
			326	(4)
			330	(4)
			334	(4)
			338	(4)
			342	(4)
			346	(4)
			350	(4)
			354	(4)
			358	(4)

387 (4)	388 (4)	389 (4)	390 (4)
391 (4)	392 (4)	393 (4)	394 (4)
395 (4)	396 (4)	397 (4)	398 (4)
249 (4)	250 (4)	251 (4)	252 (4)
253 (4)	254 (4)	255 (4)	256 (4)
257 (4)	258 (4)	259 (4)	260 (4)
261 (4)	262 (4)	263 (4)	264 (4)
265 (4)	266 (4)	267 (4)	268 (4)
269 (4)	270 (4)	271 (4)	272 (4)
273 (4)	274 (4)	275 (4)	276 (4)
277 (4)	278 (4)	279 (4)	280 (4)
281 (4)	282 (4)	283 (4)	284 (4)
285 (4)	287 (4)	288 (4)	289 (4)
290 (4)	291 (4)	292 (4)	293 (4)
294 (4)	295 (4)	296 (4)	297 (4)
298 (4)	299 (4)	300 (4)	301 (4)
302 (4)	303 (4)	304 (4)	305 (4)
306 (4)	307 (4)	308 (4)	309 (4)
310 (4)	311 (4)	312 (4)	313 (4)
314 (4)	315 (4)	316 (4)	317 (4)
318 (4)	319 (4)	320 (4)	321 (4)
322 (4)	323 (4)	324 (4)	325 (4)
326 (4)	327 (4)	328 (4)	329 (4)
330 (4)	331 (4)	332 (4)	333 (4)
334 (4)	335 (4)	336 (4)	337 (4)
338 (4)	339 (4)	340 (4)	341 (4)
342 (4)	343 (4)	344 (4)	345 (4)
346 (4)	347 (4)	348 (4)	349 (4)
350 (4)	351 (4)	352 (4)	353 (4)
354 (4)	355 (4)	356 (4)	357 (4)
358 (4)	359 (4)	360 (4)	361 (4)

```
                                                        362     (4)     363     (4)     364     (4)     365     (4)
                                                        366     (4)     367     (4)     368     (4)     369     (4)
                                                        370     (4)     371     (4)     372     (4)     373     (4)
                                                        374     (4)     375     (4)     376     (4)     377     (4)
                                                        378     (4)     379     (4)     380     (4)     381     (4)
                                                        382     (4)     383     (4)     384     (4)     385     (4)
                                                        386     (4)     387     (4)     388     (4)     389     (4)
                                                        390     (4)     391     (4)     392     (4)     393     (4)
                                                        394     (4)     395     (4)     396     (4)     397     (4)
                                                        398     (4)
PTD$M_INHERITED                 =00000008       398     (4)
PTD$M_INHERIT_CON               =00000010       398     (4)     249     (4)     250     (4)     251     (4)     252     (4)
                                                        253     (4)     254     (4)     255     (4)     256     (4)
                                                        257     (4)     258     (4)     259     (4)     260     (4)
                                                        261     (4)     262     (4)     263     (4)     264     (4)
                                                        265     (4)     266     (4)     267     (4)     268     (4)
                                                        269     (4)     270     (4)     271     (4)     272     (4)
                                                        273     (4)     274     (4)     275     (4)     276     (4)
                                                        277     (4)     278     (4)     279     (4)     280     (4)
                                                        281     (4)     282     (4)     283     (4)     284     (4)
                                                        285     (4)     287     (4)     288     (4)     289     (4)
                                                        290     (4)     291     (4)     292     (4)     293     (4)
                                                        294     (4)     295     (4)     296     (4)     297     (4)
                                                        298     (4)     299     (4)     300     (4)     301     (4)
                                                        302     (4)     303     (4)     304     (4)     305     (4)
                                                        306     (4)     307     (4)     308     (4)     309     (4)
                                                        310     (4)     311     (4)     312     (4)     313     (4)
                                                        314     (4)     315     (4)     316     (4)     317     (4)
                                                        318     (4)     319     (4)     320     (4)     321     (4)
                                                        322     (4)     323     (4)     324     (4)     325     (4)
                                                        326     (4)     327     (4)     328     (4)     329     (4)
                                                        330     (4)     331     (4)     332     (4)     333     (4)
                                                        334     (4)     335     (4)     336     (4)     337     (4)
                                                        338     (4)     339     (4)     340     (4)     341     (4)
                                                        342     (4)     343     (4)     344     (4)     345     (4)
                                                        346     (4)     347     (4)     348     (4)     349     (4)
                                                        350     (4)     351     (4)     352     (4)     353     (4)
                                                        354     (4)     355     (4)     356     (4)     357     (4)
                                                        358     (4)     359     (4)     360     (4)     361     (4)
                                                        362     (4)     363     (4)     364     (4)     365     (4)
                                                        366     (4)     367     (4)     368     (4)     369     (4)
                                                        370     (4)     371     (4)     372     (4)     373     (4)
                                                        374     (4)     375     (4)     376     (4)     377     (4)
                                                        378     (4)     379     (4)     380     (4)     381     (4)
                                                        382     (4)     383     (4)     384     (4)     385     (4)
                                                        386     (4)     387     (4)     388     (4)     389     (4)
                                                        390     (4)     391     (4)     392     (4)     393     (4)
                                                        394     (4)     395     (4)     396     (4)     397     (4)
                                                        398     (4)
PTD$M_INHERIT_PRE               =00000008       398     (4)     249     (4)     250     (4)     251     (4)     252     (4)
                                                        253     (4)     254     (4)     255     (4)     256     (4)
                                                        257     (4)     258     (4)     259     (4)     260     (4)
                                                        261     (4)     262     (4)     263     (4)     264     (4)
                                                        265     (4)     266     (4)     267     (4)     268     (4)
                                                        269     (4)     270     (4)     271     (4)     272     (4)
                                                        273     (4)     274     (4)     275     (4)     276     (4)
                                                        277     (4)     278     (4)     279     (4)     280     (4)
```

```
                                                                  281   (4)   282   (4)   283   (4)   284   (4)
                                                                  285   (4)   287   (4)   288   (4)   289   (4)
                                                                  290   (4)   291   (4)   292   (4)   293   (4)
                                                                  294   (4)   295   (4)   296   (4)   297   (4)
                                                                  298   (4)   299   (4)   300   (4)   301   (4)
                                                                  302   (4)   303   (4)   304   (4)   305   (4)
                                                                  306   (4)   307   (4)   308   (4)   309   (4)
                                                                  310   (4)   311   (4)   312   (4)   313   (4)
                                                                  314   (4)   315   (4)   316   (4)   317   (4)
                                                                  318   (4)   319   (4)   320   (4)   321   (4)
                                                                  322   (4)   323   (4)   324   (4)   325   (4)
                                                                  326   (4)   327   (4)   328   (4)   329   (4)
                                                                  330   (4)   331   (4)   332   (4)   333   (4)
                                                                  334   (4)   335   (4)   336   (4)   337   (4)
                                                                  338   (4)   339   (4)   340   (4)   341   (4)
                                                                  342   (4)   343   (4)   344   (4)   345   (4)
                                                                  346   (4)   347   (4)   348   (4)   349   (4)
                                                                  350   (4)   351   (4)   352   (4)   353   (4)
                                                                  354   (4)   355   (4)   356   (4)   357   (4)
                                                                  358   (4)   359   (4)   360   (4)   361   (4)
                                                                  362   (4)   363   (4)   364   (4)   365   (4)
                                                                  366   (4)   367   (4)   368   (4)   369   (4)
                                                                  370   (4)   371   (4)   372   (4)   373   (4)
                                                                  374   (4)   375   (4)   376   (4)   377   (4)
                                                                  378   (4)   379   (4)   380   (4)   381   (4)
                                                                  382   (4)   383   (4)   384   (4)   385   (4)
                                                                  386   (4)   387   (4)   388   (4)   389   (4)
                                                                  390   (4)   391   (4)   392   (4)   393   (4)
                                                                  394   (4)   395   (4)   396   (4)   397   (4)
                                                                  398   (4)
PTD$M_NAME                    =00000004     398     (4)           251   (4)
PTD$M_UNIT                    =00000001     398     (4)           249   (4)   250   (4)   251   (4)   252   (4)
                                                                  253   (4)   254   (4)   255   (4)   256   (4)
                                                                  257   (4)   258   (4)   259   (4)   260   (4)
                                                                  261   (4)   262   (4)   263   (4)   264   (4)
                                                                  265   (4)   266   (4)   267   (4)   268   (4)
                                                                  269   (4)   270   (4)   271   (4)   272   (4)
                                                                  273   (4)   274   (4)   275   (4)   276   (4)
                                                                  277   (4)   278   (4)   279   (4)   280   (4)
                                                                  281   (4)   282   (4)   283   (4)   284   (4)
                                                                  285   (4)   287   (4)   288   (4)   289   (4)
                                                                  290   (4)   291   (4)   292   (4)   293   (4)
                                                                  294   (4)   295   (4)   296   (4)   297   (4)
                                                                  298   (4)   299   (4)   300   (4)   301   (4)
                                                                  302   (4)   303   (4)   304   (4)   305   (4)
                                                                  306   (4)   307   (4)   308   (4)   309   (4)
                                                                  310   (4)   311   (4)   312   (4)   313   (4)
                                                                  314   (4)   315   (4)   316   (4)   317   (4)
                                                                  318   (4)   319   (4)   320   (4)   321   (4)
                                                                  322   (4)   323   (4)   324   (4)   325   (4)
                                                                  326   (4)   327   (4)   328   (4)   329   (4)
                                                                  330   (4)   331   (4)   332   (4)   333   (4)
                                                                  334   (4)   335   (4)   336   (4)   337   (4)
                                                                  338   (4)   339   (4)   340   (4)   341   (4)
                                                                  342   (4)   343   (4)   344   (4)   345   (4)
                                                                  346   (4)   347   (4)   348   (4)   349   (4)
                                                                  350   (4)   351   (4)   352   (4)   353   (4)
```

```
                                                                354    (4)    355    (4)    356    (4)    357    (4)
                                                                358    (4)    359    (4)    360    (4)    361    (4)
                                                                362    (4)    363    (4)    364    (4)    365    (4)
                                                                366    (4)    367    (4)    368    (4)    369    (4)
                                                                370    (4)    371    (4)    372    (4)    373    (4)
                                                                374    (4)    375    (4)    376    (4)    377    (4)
                                                                378    (4)    379    (4)    380    (4)    381    (4)
                                                                382    (4)    383    (4)    384    (4)    385    (4)
                                                                386    (4)    387    (4)    388    (4)    389    (4)
                                                                390    (4)    391    (4)    392    (4)    393    (4)
                                                                394    (4)    395    (4)    396    (4)    397    (4)
                                                                398    (4)
PTD$V_CONTROLLER                =00000001     398    (4)
PTD$V_INHERIT_CON               =00000004     398    (4)
PTD$V_INHERIT_PRE               =00000003     398    (4)
PTD$V_NAME                      =00000002     398    (4)
PTD$V_UNIT                      =00000000     398    (4)
PX780                            00001C6D-R   327    (4)    245    (3)
PX780$B_BR                       00000033                   327    (4)
PX780$B_NODE                     00000034                   327    (4)
PX780$B_TR                       00000032                   327    (4)
PX780$K_LEN                      00000035                   327    (4)
R80                              00001CD6-R   328    (4)    245    (3)
R80$K_LEN                        00000032                   328    (4)
RA60                             00001CEF-R   329    (4)    245    (3)
RA60$K_LEN                       00000032                   329    (4)
RA70                             00001D09-R   330    (4)    245    (3)
RA70$K_LEN                       00000032                   330    (4)
RA80                             00001D23-R   331    (4)    245    (3)
RA80$K_LEN                       00000032                   331    (4)
RA81                             00001D3D-R   332    (4)    245    (3)
RA81$K_LEN                       00000032                   332    (4)
RA82                             00001D57-R   333    (4)    245    (3)
RA82$K_LEN                       00000032                   333    (4)
RA90                             00001D71-R   334    (4)    245    (3)
RA90$K_LEN                       00000032                   334    (4)
RB730                            00001D8B-R   335    (4)    245    (3)
RB730$B_BR                       00000036                   335    (4)
RB730$K_LEN                      00000037                   335    (4)
RC25                             00001DD5-R   337    (4)    245    (3)
RC25$K_LEN                       00000032                   337    (4)
RCF25                            00001DBA-R   336    (4)    245    (3)
RCF25$K_LEN                      00000032                   336    (4)
RD51                             00001DEF-R   338    (4)    245    (3)
RD51$K_LEN                       00000032                   338    (4)
RD52                             00001E09-R   339    (4)    245    (3)
RD52$K_LEN                       00000032                   339    (4)
RH750                            00001E23-R   340    (4)    245    (3)
RH750$B_BR                       00000032                   340    (4)
RH750$K_LEN                      00000033                   340    (4)
RH780                            00001E7C-R   341    (4)    245    (3)
RH780$B_BR                       00000033                   341    (4)
RH780$B_TR                       00000032                   341    (4)
RH780$K_LEN                      00000034                   341    (4)
RK06                             00001F62-R   345    (4)    245    (3)
RK06$K_LEN                       00000032                   345    (4)
RK07                             00001F7C-R   346    (4)    245    (3)
```

```
RK07$K_LEN                      00000032                        346     (4)
RK611                           00001F96-R      347     (4)     245     (3)
RK611$B_BR                      00000036                        347     (4)
RK611$K_LEN                     00000037                        347     (4)
RK611$L_CSR                     00000032                        347     (4)
RL01                            00001EE1-R      342     (4)     245     (3)
RL01$K_LEN                      00000032                        342     (4)
RL02                            00001EFB-R      343     (4)     245     (3)
RL02$K_LEN                      00000032                        343     (4)
RL11                            00001F15-R      344     (4)     245     (3)
RL11$B_BR                       00000036                        344     (4)
RL11$K_LEN                      00000037                        344     (4)
RL11$L_CSR                      00000032                        344     (4)
RM03                            00001FE4-R      348     (4)     245     (3)
RM03$K_LEN                      00000032                        348     (4)
RM05                            00002008-R      349     (4)     245     (3)
RM05$K_LEN                      00000032                        349     (4)
RM80                            0000202C-R      350     (4)     245     (3)
RM80$K_LEN                      00000032                        350     (4)
RP04                            00002050-R      351     (4)     245     (3)
RP04$K_LEN                      00000032                        351     (4)
RP05                            00002074-R      352     (4)     245     (3)
RP05$K_LEN                      00000032                        352     (4)
RP06                            00002098-R      353     (4)     245     (3)
RP06$K_LEN                      00000032                        353     (4)
RP07                            000020BC-R      354     (4)     245     (3)
RP07$K_LEN                      00000032                        354     (4)
RQDX1                           000020E0-R      355     (4)     245     (3)
RQDX1$K_LEN                     00000036                        355     (4)
RQDX1$L_IP                      00000032                        355     (4)
RRD50                           00002107-R      356     (4)     245     (3)
RRD50$B_BR                      00000036                        356     (4)
RRD50$K_LEN                     00000037                        356     (4)
RRD50$L_CSR                     00000032                        356     (4)
RUX50                           00002163-R      357     (4)     245     (3)
RX02                            000021AB-R      358     (4)     245     (3)
RX02$K_LEN                      00000032                        358     (4)
RX180                           00002248-R      363     (4)     245     (3)
RX180$K_LEN                     00000032                        363     (4)
RX211                           000021C5-R      359     (4)     245     (3)
RX211$B_BR                      00000036                        359     (4)
RX211$K_LEN                     00000037                        359     (4)
RX211$L_CSR                     00000032                        359     (4)
RX31                            00002213-R      360     (4)     245     (3)
RX31$K_LEN                      00000032                        360     (4)
RX32                            00002223-R      361     (4)     245     (3)
RX32$K_LEN                      00000032                        361     (4)
RX50                            00002233-R      362     (4)     245     (3)
RX50$K_LEN                      00000032                        362     (4)
SBIA                            00002259-R      364     (4)     245     (3)
SBIA$K_LEN                      00000032                        364     (4)
SIZ...                         =00000001        398     (4)     249     (4)     250     (4)     251     (4)     252     (4)
                                                                253     (4)     254     (4)     255     (4)     256     (4)
                                                                257     (4)     258     (4)     259     (4)     260     (4)
                                                                261     (4)     262     (4)     263     (4)     264     (4)
                                                                265     (4)     266     (4)     267     (4)     268     (4)
                                                                269     (4)     270     (4)     271     (4)     272     (4)
```

				273	(4)	274	(4)	275	(4)	276	(4)				
				277	(4)	278	(4)	279	(4)	280	(4)				
				281	(4)	282	(4)	283	(4)	284	(4)				
				285	(4)	287	(4)	288	(4)	289	(4)				
				290	(4)	291	(4)	292	(4)	293	(4)				
				294	(4)	295	(4)	296	(4)	297	(4)				
				298	(4)	299	(4)	300	(4)	301	(4)				
				302	(4)	303	(4)	304	(4)	305	(4)				
				306	(4)	307	(4)	308	(4)	309	(4)				
				310	(4)	311	(4)	312	(4)	313	(4)				
				314	(4)	315	(4)	316	(4)	317	(4)				
				318	(4)	319	(4)	320	(4)	321	(4)				
				322	(4)	323	(4)	324	(4)	325	(4)				
				326	(4)	327	(4)	328	(4)	329	(4)				
				330	(4)	331	(4)	332	(4)	333	(4)				
				334	(4)	335	(4)	336	(4)	337	(4)				
				338	(4)	339	(4)	340	(4)	341	(4)				
				342	(4)	343	(4)	344	(4)	345	(4)				
				346	(4)	347	(4)	348	(4)	349	(4)				
				350	(4)	351	(4)	352	(4)	353	(4)				
				354	(4)	355	(4)	356	(4)	357	(4)				
				358	(4)	359	(4)	360	(4)	361	(4)				
				362	(4)	363	(4)	364	(4)	365	(4)				
				366	(4)	367	(4)	368	(4)	369	(4)				
				370	(4)	371	(4)	372	(4)	373	(4)				
				374	(4)	375	(4)	376	(4)	377	(4)				
				378	(4)	379	(4)	380	(4)	381	(4)				
				382	(4)	383	(4)	384	(4)	385	(4)				
				386	(4)	387	(4)	388	(4)	389	(4)				
				390	(4)	391	(4)	392	(4)	393	(4)				
				394	(4)	395	(4)	396	(4)	397	(4)				
				398	(4)										
TE16		0000229B-R	365	(4)	245	(3)									
TE16$K_LEN		00000032			365	(4)									
TK50		000022B5-R	366	(4)	245	(3)									
TM03		000022CF-R	367	(4)	245	(3)									
TM03$B_DRIVE		00000032			367	(4)									
TM03$K_LEN		00000033			367	(4)									
TM78		000022FD-R	368	(4)	245	(3)									
TM78$B_DRIVE		00000032			368	(4)									
TM78$K_LEN		00000033			368	(4)									
TS04		0000232B-R	369	(4)	245	(3)									
TS04$B_BR		00000036			369	(4)									
TS04$K_LEN		00000037			369	(4)									
TS04$L_CSR		00000032			369	(4)									
TS05		00002382-R	370	(4)	245	(3)									
TS05$B_BR		00000032			370	(4)									
TS05$K_LEN		00000033			370	(4)									
TS11		000023EC-R	371	(4)	245	(3)									
TS11$B_BR		00000036			371	(4)									
TS11$K_LEN		00000037			371	(4)									
TS11$L_CSR		00000032			371	(4)									
TU45		00002443-R	372	(4)	245	(3)									
TU45$K_LEN		00000032			372	(4)									
TU58		0000245D-R	373	(4)	245	(3)									
TU58$K_LEN		00000032			373	(4)									
TU70		00002477-R	374	(4)	245	(3)									

```
TU70$K_LEN          00000032                    374    (4)
TU72                00002491-R   375    (4)     245    (3)
TU72$K_LEN          00000032                    375    (4)
TU77                000024AB-R   376    (4)     245    (3)
TU77$K_LEN          00000032                    376    (4)
TU78                000024C5-R   377    (4)     245    (3)
TU78$K_LEN          00000032                    377    (4)
TU80                000024DF-R   378    (4)     245    (3)
TU80$K_LEN          00000037                    378    (4)
TU81                0000254E-R   379    (4)     245    (3)
TU81$B_BR           00000036                    379    (4)
TU81$K_LEN          00000037                    379    (4)
TU81$L_CSR          00000032                    379    (4)
UBE                 000025A5-R   380    (4)     245    (3)
UBE$K_LEN           00000036                    380    (4)
UBE$L_CSR           00000032                    380    (4)
UDA50               000025E0-R   381    (4)     245    (3)
UNA11               00002649-R   382    (4)     245    (3)
UNA11$B_BR          00000032                    382    (4)
UNA11$K_LEN         00000037                    382    (4)
UNA11$L_CSR         00000033                    382    (4)
VS100               00002697-R   383    (4)     245    (3)
VS100$B_BR          00000036                    383    (4)
VS100$K_LEN         00000037                    383    (4)
VS100$L_CSR         00000032                    383    (4)
VS125               000026EF-R   384    (4)     245    (3)
VS125$B_BR          00000036                    384    (4)
VS125$K_LEN         00000037                    384    (4)
VS125$L_CSR         00000032                    384    (4)
VS300               00002747-R   385    (4)     245    (3)
VS300$B_BR          00000036                    385    (4)
VS300$K_LEN         00000037                    385    (4)
VS300$L_CSR         00000032                    385    (4)
VT100               00002821-R   391    (4)     245    (3)
VT100$K_LEN         00000032                    391    (4)
VT101               0000283C-R   392    (4)     245    (3)
VT101$K_LEN         00000032                    392    (4)
VT102               00002857-R   393    (4)     245    (3)
VT102$K_LEN         00000032                    393    (4)
VT125               00002872-R   394    (4)     245    (3)
VT125$K_LEN         00000032                    394    (4)
VT131               0000288D-R   395    (4)     245    (3)
VT131$K_LEN         00000032                    395    (4)
VT132               000028A8-R   396    (4)     245    (3)
VT132$K_LEN         00000032                    396    (4)
VT220               000028C3-R   397    (4)     245    (3)
VT220$K_LEN         00000032                    397    (4)
VT240               000028DE-R   398    (4)     245    (3)
VT240$K_LEN         00000032                    398    (4)
VT50                0000279F-R   386    (4)     245    (3)
VT50$K_LEN          00000032                    386    (4)
VT52                000027B9-R   387    (4)     245    (3)
VT52$K_LEN          00000032                    387    (4)
VT55                000027D3-R   388    (4)     245    (3)
VT55$K_LEN          00000032                    388    (4)
VT61                000027ED-R   389    (4)     245    (3)
VT61$K_LEN          00000032                    389    (4)
```

```
VT71                    00002807-R   390    (4)      245    (3)
VT71$K_LEN              00000032                     390    (4)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEF	1	398 (4)	
$DEFINI	1	174 (2)	174 (2) 175 (2) 176 (2) 177 (2) 249 (4) 250 (4) 254 (4) 255 (4) 256 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4) 264 (4) 265 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 278 (4) 279 (4) 280 (4) 281 (4) 282 (4) 283 (4) 284 (4) 285 (4) 286 (4) 292 (4) 293 (4) 294 (4) 295 (4) 296 (4) 297 (4) 298 (4) 299 (4) 300 (4) 301 (4) 302 (4) 303 (4) 304 (4) 305 (4) 306 (4) 307 (4) 308 (4) 309 (4) 310 (4) 311 (4) 312 (4) 313 (4) 314 (4) 315 (4) 316 (4) 317 (4) 318 (4) 319 (4) 320 (4) 321 (4) 322 (4) 323 (4) 324 (4) 325 (4) 326 (4) 327 (4) 328 (4) 329 (4) 330 (4) 331 (4) 332 (4) 333 (4) 334 (4) 335 (4) 336 (4) 337 (4) 338 (4) 339 (4) 340 (4) 341 (4) 342 (4) 343 (4) 344 (4) 345 (4) 346 (4) 347 (4) 348 (4) 349 (4) 350 (4) 351 (4) 352 (4) 353 (4) 354 (4) 355 (4) 356 (4) 357 (4) 358 (4) 359 (4) 360 (4) 361 (4) 362 (4) 363 (4) 364 (4) 365 (4) 366 (4) 367 (4) 368 (4) 369 (4) 370 (4) 371 (4) 372 (4) 373 (4) 374 (4) 375 (4) 376 (4) 377 (4) 378 (4) 379 (4) 380 (4) 381 (4) 382 (4) 383 (4) 384 (4) 385 (4) 386 (4) 387 (4) 388 (4) 389 (4) 390 (4) 391 (4) 392 (4) 393 (4) 394 (4) 395 (4) 396 (4) 397 (4) 398 (4)
$DS_$ADD	1	251 (4)	251 (4) 279 (4) 364 (4)
$DS_$CASE	1	251 (4)	251 (4) 319 (4)
$DS_$COMPLEMENT	1	279 (4)	279 (4)
$DS_$DECIMAL	1	249 (4)	249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 255 (4) 257 (4) 258 (4) 260 (4) 262 (4) 263 (4) 264 (4) 265 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 280 (4) 281 (4) 282 (4) 285 (4) 287 (4) 288 (4) 289 (4) 290 (4) 291 (4) 292 (4) 293 (4) 302 (4) 312 (4) 317 (4) 318 (4) 319 (4) 320 (4) 321 (4) 323 (4) 325 (4) 326 (4) 327 (4) 340 (4) 341 (4) 344 (4) 347 (4) 356 (4) 357 (4) 359 (4) 367 (4) 368 (4) 369 (4) 370 (4) 371 (4) 378 (4) 379 (4) 381 (4) 382 (4) 383 (4) 384 (4) 385 (4)
$DS_$END	1	249 (4)	249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 258 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4)

				264 (4) 265 (4) 266 (4) 267 (4) 268 (4)
				269 (4) 270 (4) 271 (4) 272 (4) 273 (4)
				274 (4) 275 (4) 276 (4) 277 (4) 278 (4)
				279 (4) 280 (4) 281 (4) 282 (4) 283 (4)
				284 (4) 285 (4) 287 (4) 288 (4) 289 (4)
				290 (4) 291 (4) 292 (4) 293 (4) 294 (4)
				295 (4) 296 (4) 297 (4) 298 (4) 299 (4)
				300 (4) 301 (4) 302 (4) 303 (4) 304 (4)
				305 (4) 306 (4) 307 (4) 308 (4) 309 (4)
				310 (4) 311 (4) 312 (4) 313 (4) 314 (4)
				315 (4) 316 (4) 317 (4) 318 (4) 319 (4)
				320 (4) 321 (4) 322 (4) 323 (4) 324 (4)
				325 (4) 326 (4) 327 (4) 328 (4) 329 (4)
				330 (4) 331 (4) 332 (4) 333 (4) 334 (4)
				335 (4) 336 (4) 337 (4) 338 (4) 339 (4)
				340 (4) 341 (4) 342 (4) 343 (4) 344 (4)
				345 (4) 346 (4) 347 (4) 348 (4) 349 (4)
				350 (4) 351 (4) 352 (4) 353 (4) 354 (4)
				355 (4) 356 (4) 357 (4) 358 (4) 359 (4)
				360 (4) 361 (4) 362 (4) 363 (4) 364 (4)
				365 (4) 366 (4) 367 (4) 368 (4) 369 (4)
				370 (4) 371 (4) 372 (4) 373 (4) 374 (4)
				375 (4) 376 (4) 377 (4) 378 (4) 379 (4)
				380 (4) 381 (4) 382 (4) 383 (4) 384 (4)
				385 (4) 386 (4) 387 (4) 388 (4) 389 (4)
				390 (4) 391 (4) 392 (4) 393 (4) 394 (4)
				395 (4) 396 (4) 397 (4) 398 (4)
$DS_$FETCH	1	251	(4)	251 (4) 252 (4) 253 (4) 279 (4) 280 (4)
				319 (4) 320 (4) 322 (4) 323 (4) 340 (4)
				341 (4) 348 (4) 349 (4) 350 (4) 351 (4)
				352 (4) 353 (4) 354 (4) 364 (4)
$DS_$HEX	1	287	(4)	287 (4) 288 (4) 289 (4) 290 (4) 291 (4)
				320 (4)
$DS_$INITIALIZE	2	249	(4)	249 (4) 250 (4) 251 (4) 252 (4) 253 (4)
				254 (4) 255 (4) 256 (4) 257 (4) 258 (4)
				259 (4) 260 (4) 261 (4) 262 (4) 263 (4)
				264 (4) 265 (4) 266 (4) 267 (4) 268 (4)
				269 (4) 270 (4) 271 (4) 272 (4) 273 (4)
				274 (4) 275 (4) 276 (4) 277 (4) 278 (4)
				279 (4) 280 (4) 281 (4) 282 (4) 283 (4)
				284 (4) 285 (4) 287 (4) 288 (4) 289 (4)
				290 (4) 291 (4) 292 (4) 293 (4) 294 (4)
				295 (4) 296 (4) 297 (4) 298 (4) 299 (4)
				300 (4) 301 (4) 302 (4) 303 (4) 304 (4)
				305 (4) 306 (4) 307 (4) 308 (4) 309 (4)
				310 (4) 311 (4) 312 (4) 313 (4) 314 (4)
				315 (4) 316 (4) 317 (4) 318 (4) 319 (4)
				320 (4) 321 (4) 322 (4) 323 (4) 324 (4)
				325 (4) 326 (4) 327 (4) 328 (4) 329 (4)
				330 (4) 331 (4) 332 (4) 333 (4) 334 (4)
				335 (4) 336 (4) 337 (4) 338 (4) 339 (4)
				340 (4) 341 (4) 342 (4) 343 (4) 344 (4)
				345 (4) 346 (4) 347 (4) 348 (4) 349 (4)
				350 (4) 351 (4) 352 (4) 353 (4) 354 (4)
				355 (4) 356 (4) 357 (4) 358 (4) 359 (4)
				360 (4) 361 (4) 362 (4) 363 (4) 364 (4)
				365 (4) 366 (4) 367 (4) 368 (4) 369 (4)

				370	(4)	371	(4)	372	(4)	373	(4)	374	(4)		
				375	(4)	376	(4)	377	(4)	378	(4)	379	(4)		
				380	(4)	381	(4)	382	(4)	383	(4)	384	(4)		
				385	(4)	386	(4)	387	(4)	388	(4)	389	(4)		
				390	(4)	391	(4)	392	(4)	393	(4)	394	(4)		
				395	(4)	396	(4)	397	(4)	398	(4)				
$DS_$LITERAL	1	251	(4)	251	(4)	252	(4)	253	(4)	255	(4)	263	(4)		
				265	(4)	267	(4)	269	(4)	274	(4)	275	(4)		
				278	(4)	279	(4)	280	(4)	287	(4)	288	(4)		
				289	(4)	290	(4)	291	(4)	294	(4)	295	(4)		
				296	(4)	297	(4)	298	(4)	299	(4)	300	(4)		
				301	(4)	303	(4)	304	(4)	305	(4)	306	(4)		
				307	(4)	308	(4)	309	(4)	310	(4)	311	(4)		
				313	(4)	314	(4)	315	(4)	316	(4)	317	(4)		
				319	(4)	321	(4)	323	(4)	325	(4)	327	(4)		
				328	(4)	329	(4)	330	(4)	331	(4)	332	(4)		
				333	(4)	334	(4)	335	(4)	336	(4)	337	(4)		
				338	(4)	339	(4)	340	(4)	341	(4)	342	(4)		
				343	(4)	345	(4)	346	(4)	348	(4)	349	(4)		
				350	(4)	351	(4)	352	(4)	353	(4)	354	(4)		
				356	(4)	358	(4)	362	(4)	364	(4)	365	(4)		
				366	(4)	369	(4)	370	(4)	371	(4)	372	(4)		
				373	(4)	374	(4)	375	(4)	376	(4)	377	(4)		
				378	(4)	379	(4)	383	(4)	384	(4)	385	(4)		
				386	(4)	387	(4)	388	(4)	389	(4)	390	(4)		
				391	(4)	392	(4)	393	(4)	394	(4)	395	(4)		
				396	(4)	397	(4)	398	(4)						
$DS_$LOGICAL	1	261	(4)	261	(4)	264	(4)	269	(4)	274	(4)	275	(4)		
				287	(4)	288	(4)	289	(4)	290	(4)	291	(4)		
$DS_$NAME	1	249	(4)	249	(4)	250	(4)	251	(4)	252	(4)	253	(4)		
				255	(4)	256	(4)	257	(4)	258	(4)	259	(4)		
				260	(4)	261	(4)	262	(4)	264	(4)	265	(4)		
				266	(4)	267	(4)	268	(4)	269	(4)	270	(4)		
				271	(4)	272	(4)	273	(4)	274	(4)	275	(4)		
				276	(4)	277	(4)	278	(4)	279	(4)	280	(4)		
				281	(4)	282	(4)	283	(4)	284	(4)	285	(4)		
				287	(4)	288	(4)	289	(4)	290	(4)	291	(4)		
				292	(4)	293	(4)	294	(4)	295	(4)	296	(4)		
				297	(4)	298	(4)	299	(4)	300	(4)	301	(4)		
				302	(4)	303	(4)	304	(4)	305	(4)	306	(4)		
				308	(4)	309	(4)	310	(4)	311	(4)	312	(4)		
				313	(4)	314	(4)	315	(4)	316	(4)	317	(4)		
				318	(4)	319	(4)	320	(4)	321	(4)	323	(4)		
				324	(4)	325	(4)	327	(4)	328	(4)	329	(4)		
				330	(4)	331	(4)	332	(4)	333	(4)	334	(4)		
				335	(4)	336	(4)	337	(4)	338	(4)	339	(4)		
				340	(4)	341	(4)	342	(4)	343	(4)	344	(4)		
				345	(4)	346	(4)	347	(4)	348	(4)	349	(4)		
				350	(4)	351	(4)	352	(4)	353	(4)	354	(4)		
				355	(4)	356	(4)	357	(4)	358	(4)	359	(4)		
				360	(4)	361	(4)	363	(4)	364	(4)	365	(4)		
				366	(4)	367	(4)	368	(4)	369	(4)	370	(4)		
				371	(4)	372	(4)	373	(4)	374	(4)	375	(4)		
				376	(4)	377	(4)	378	(4)	379	(4)	380	(4)		
				381	(4)	382	(4)	383	(4)	384	(4)	385	(4)		
				386	(4)	387	(4)	388	(4)	389	(4)	390	(4)		
				391	(4)	392	(4)	393	(4)	394	(4)	395	(4)		

				396	(4)	397	(4)	398	(4)						
$DS_$OCTAL	1	249	(4)	249	(4)	250	(4)	255	(4)	256	(4)	257	(4)		
				258	(4)	260	(4)	261	(4)	262	(4)	263	(4)		
				264	(4)	265	(4)	266	(4)	267	(4)	268	(4)		
				269	(4)	270	(4)	271	(4)	272	(4)	273	(4)		
				276	(4)	277	(4)	282	(4)	283	(4)	284	(4)		
				285	(4)	292	(4)	293	(4)	302	(4)	312	(4)		
				317	(4)	318	(4)	326	(4)	344	(4)	347	(4)		
				355	(4)	356	(4)	357	(4)	359	(4)	369	(4)		
				370	(4)	371	(4)	378	(4)	379	(4)	380	(4)		
				381	(4)	382	(4)	383	(4)	384	(4)	385	(4)		
$DS_$STORE	1	249	(4)	249	(4)	250	(4)	251	(4)	252	(4)	253	(4)		
				255	(4)	256	(4)	257	(4)	258	(4)	260	(4)		
				261	(4)	262	(4)	263	(4)	264	(4)	265	(4)		
				266	(4)	267	(4)	268	(4)	269	(4)	270	(4)		
				271	(4)	272	(4)	273	(4)	274	(4)	275	(4)		
				276	(4)	277	(4)	278	(4)	279	(4)	280	(4)		
				281	(4)	282	(4)	283	(4)	284	(4)	285	(4)		
				287	(4)	288	(4)	289	(4)	290	(4)	291	(4)		
				292	(4)	293	(4)	294	(4)	295	(4)	296	(4)		
				297	(4)	298	(4)	299	(4)	300	(4)	301	(4)		
				302	(4)	303	(4)	304	(4)	305	(4)	306	(4)		
				307	(4)	308	(4)	309	(4)	310	(4)	311	(4)		
				312	(4)	313	(4)	314	(4)	315	(4)	316	(4)		
				317	(4)	318	(4)	319	(4)	320	(4)	321	(4)		
				322	(4)	323	(4)	325	(4)	326	(4)	327	(4)		
				328	(4)	329	(4)	330	(4)	331	(4)	332	(4)		
				333	(4)	334	(4)	335	(4)	336	(4)	337	(4)		
				338	(4)	339	(4)	340	(4)	341	(4)	342	(4)		
				343	(4)	344	(4)	345	(4)	346	(4)	347	(4)		
				348	(4)	349	(4)	350	(4)	351	(4)	352	(4)		
				353	(4)	354	(4)	355	(4)	356	(4)	357	(4)		
				358	(4)	359	(4)	362	(4)	364	(4)	365	(4)		
				366	(4)	367	(4)	368	(4)	369	(4)	370	(4)		
				371	(4)	372	(4)	373	(4)	374	(4)	375	(4)		
				376	(4)	377	(4)	378	(4)	379	(4)	380	(4)		
				381	(4)	382	(4)	383	(4)	384	(4)	385	(4)		
				386	(4)	387	(4)	388	(4)	389	(4)	390	(4)		
				391	(4)	392	(4)	393	(4)	394	(4)	395	(4)		
				396	(4)	397	(4)	398	(4)						
$DS_$STRING	1	251	(4)	251	(4)	261	(4)	264	(4)	265	(4)	266	(4)		
				267	(4)	269	(4)	274	(4)	275	(4)	282	(4)		
				323	(4)										
$DS_AA11K	1	249	(4)	249	(4)										
$DS_AA11K_DEF	1	249	(4)	249	(4)										
$DS_AD11K	1	250	(4)	250	(4)										
$DS_AD11K_DEF	1	250	(4)	250	(4)										
$DS_CI750	2	252	(4)	252	(4)										
$DS_CI780	3	253	(4)	253	(4)										
$DS_CI_DEF	2	177	(2)	177	(2)										
$DS_CI_NODE	5	251	(4)	251	(4)										
$DS_CONSOLE	1	254	(4)	254	(4)										
$DS_CONSOLE_DEF	1	254	(4)	254	(4)										
$DS_CR11	1	255	(4)	255	(4)										
$DS_CR11_DEF	1	255	(4)	255	(4)										
$DS_DEQNA	1	256	(4)	256	(4)										
$DS_DEQNA_DEF	1	256	(4)	256	(4)										

$DS_DEVTYP	1	217	(3)	217	(3)
$DS_DHU11	1	257	(4)	257	(4)
$DS_DHUV_DEF	1	176	(2)	176	(2)
$DS_DHV11	1	258	(4)	258	(4)
$DS_DISK	1	186	(2)	259	(4)
$DS_DISK_DEF	1	179	(2)	259	(4)
$DS_DL11	1	260	(4)	260	(4)
$DS_DL11_DEF	1	260	(4)	260	(4)
$DS_DLVJ1	2	261	(4)	261	(4)
$DS_DLVJ1_DEF	1	261	(4)	261	(4)
$DS_DMC11	1	262	(4)	262	(4)
$DS_DMC11_DEF	1	262	(4)	262	(4)
$DS_DMF32	1	263	(4)	263	(4)
$DS_DMF32A	2	264	(4)	264	(4)
$DS_DMF32A_DEF	2	264	(4)	264	(4)
$DS_DMF32P	1	265	(4)	265	(4)
$DS_DMF32P_DEF	1	265	(4)	265	(4)
$DS_DMF32S	1	266	(4)	266	(4)
$DS_DMF32S_DEF	2	266	(4)	266	(4)
$DS_DMF32_DEF	1	263	(4)	263	(4)
$DS_DMP11	2	267	(4)	267	(4)
$DS_DMP11_DEF	1	267	(4)	267	(4)
$DS_DMR11	1	268	(4)	268	(4)
$DS_DMR11_DEF	1	268	(4)	268	(4)
$DS_DMZ32	3	269	(4)	269	(4)
$DS_DMZ32_DEF	2	269	(4)	269	(4)
$DS_DR11B	1	270	(4)	270	(4)
$DS_DR11B_DEF	1	270	(4)	270	(4)
$DS_DR11C	1	271	(4)	271	(4)
$DS_DR11C_DEF	1	271	(4)	271	(4)
$DS_DR11K	1	272	(4)	272	(4)
$DS_DR11K_DEF	1	272	(4)	272	(4)
$DS_DR11W	1	273	(4)	273	(4)
$DS_DR11W_DEF	1	273	(4)	273	(4)
$DS_DR750	2	274	(4)	274	(4)
$DS_DR750_DEF	1	274	(4)	274	(4)
$DS_DR780	2	275	(4)	275	(4)
$DS_DR780_DEF	1	275	(4)	275	(4)
$DS_DUP11	1	276	(4)	276	(4)
$DS_DUP11_DEF	1	276	(4)	276	(4)
$DS_DV11	1	277	(4)	277	(4)
$DS_DV11_DEF	1	277	(4)	277	(4)
$DS_DW730	1	278	(4)	278	(4)
$DS_DW730_DEF	1	278	(4)	278	(4)
$DS_DW750	1	279	(4)	279	(4)
$DS_DW750_DEF	1	279	(4)	279	(4)
$DS_DW780	2	280	(4)	280	(4)
$DS_DW780_DEF	1	280	(4)	280	(4)
$DS_DX20	1	281	(4)	281	(4)
$DS_DX20_DEF	1	281	(4)	281	(4)
$DS_DZ11	1	282	(4)	282	(4)
$DS_DZ11_DEF	1	282	(4)	282	(4)
$DS_DZ32	2	283	(4)	283	(4)
$DS_DZ32_DEF	1	283	(4)	283	(4)
$DS_DZV11	1	284	(4)	284	(4)
$DS_DZV11_DEF	1	284	(4)	284	(4)
$DS_HPEODEF	1	175	(2)	175	(2)

```
$DS_HPODEF        2          174    (2)       174    (2)
$DS_IEU11A        1          285    (4)       285    (4)
$DS_IEU11A_DEF    1          285    (4)       285    (4)
$DS_KA730         3          287    (4)       287    (4)
$DS_KA750         3          288    (4)       288    (4)
$DS_KA780         2          289    (4)       289    (4)
$DS_KA785         2          291    (4)       291    (4)
$DS_KA86          2          290    (4)       290    (4)
$DS_KA_DEF        4          286    (4)       286    (4)
$DS_KMC11         1          292    (4)       292    (4)
$DS_KMC11_DEF     1          292    (4)       292    (4)
$DS_KW11K         1          293    (4)       293    (4)
$DS_KW11K_DEF     1          293    (4)       293    (4)
$DS_LA100         1          295    (4)       295    (4)
$DS_LA100_DEF     1          295    (4)       295    (4)
$DS_LA12          1          294    (4)       294    (4)
$DS_LA120         1          296    (4)       296    (4)
$DS_LA120_DEF     1          296    (4)       296    (4)
$DS_LA12_DEF      1          294    (4)       294    (4)
$DS_LA180         1          297    (4)       297    (4)
$DS_LA180_DEF     1          297    (4)       297    (4)
$DS_LA210         1          301    (4)       301    (4)
$DS_LA210_DEF     1          301    (4)       301    (4)
$DS_LA34          1          298    (4)       298    (4)
$DS_LA34_DEF      1          298    (4)       298    (4)
$DS_LA36          1          299    (4)       299    (4)
$DS_LA36_DEF      1          299    (4)       299    (4)
$DS_LA38          1          300    (4)       300    (4)
$DS_LA38_DEF      1          300    (4)       300    (4)
$DS_LESI          1          302    (4)       302    (4)
$DS_LESI_DEF      1          302    (4)       302    (4)
$DS_LG01          1          303    (4)       303    (4)
$DS_LG01_DEF      1          303    (4)       303    (4)
$DS_LG02          1          304    (4)       304    (4)
$DS_LG02_DEF      1          304    (4)       304    (4)
$DS_LG03          1          305    (4)       305    (4)
$DS_LG03_DEF      1          305    (4)       305    (4)
$DS_LN01          1          306    (4)       306    (4)
$DS_LN01_DEF      1          306    (4)       306    (4)
$DS_LN03          1          307    (4)       307    (4)
$DS_LN03_DEF      1          307    (4)       307    (4)
$DS_LP04          1          308    (4)       308    (4)
$DS_LP04_DEF      1          308    (4)       308    (4)
$DS_LP05          1          309    (4)       309    (4)
$DS_LP05_DEF      1          309    (4)       309    (4)
$DS_LP06          1          310    (4)       310    (4)
$DS_LP06_DEF      1          310    (4)       310    (4)
$DS_LP07          1          311    (4)       311    (4)
$DS_LP07_DEF      1          311    (4)       311    (4)
$DS_LP11          1          312    (4)       312    (4)
$DS_LP11_DEF      1          312    (4)       312    (4)
$DS_LP14          1          313    (4)       313    (4)
$DS_LP14_DEF      1          313    (4)       313    (4)
$DS_LP25          1          314    (4)       314    (4)
$DS_LP25_DEF      1          314    (4)       314    (4)
$DS_LP26          1          315    (4)       315    (4)
$DS_LP26_DEF      1          315    (4)       315    (4)
```

```
$DS_LP27          1          316    (4)      316    (4)
$DS_LP27_DEF      1          316    (4)      316    (4)
$DS_LPA11K        1          317    (4)      317    (4)
$DS_LPA11K_DEF    1          317    (4)      317    (4)
$DS_LUA11         2          318    (4)      318    (4)
$DS_LUA11_DEF     1          318    (4)      318    (4)
$DS_MA780         2          321    (4)      321    (4)
$DS_MA780_DEF     1          321    (4)      321    (4)
$DS_MBE           1          322    (4)      322    (4)
$DS_MBE_DEF       1          322    (4)      322    (4)
$DS_ML11          1          323    (4)      323    (4)
$DS_ML11_DEF      1          323    (4)      323    (4)
$DS_MS750         1          324    (4)      324    (4)
$DS_MS750_DEF     1          324    (4)      324    (4)
$DS_MS780         1          325    (4)      325    (4)
$DS_MS780_DEF     1          325    (4)      325    (4)
$DS_NBIA          2          319    (4)      319    (4)
$DS_NBIA_DEF      1          319    (4)      319    (4)
$DS_NBIB          1          320    (4)      320    (4)
$DS_NBIB_DEF      1          320    (4)      320    (4)
$DS_PCL11         1          326    (4)      326    (4)
$DS_PCL11_DEF     1          326    (4)      326    (4)
$DS_PDDEF         1          249    (4)      249    (4)      250    (4)      251    (4)      252    (4)      253    (4)
                                            254    (4)      255    (4)      256    (4)      257    (4)      258    (4)
                                            259    (4)      260    (4)      261    (4)      262    (4)      263    (4)
                                            264    (4)      265    (4)      266    (4)      267    (4)      268    (4)
                                            269    (4)      270    (4)      271    (4)      272    (4)      273    (4)
                                            274    (4)      275    (4)      276    (4)      277    (4)      278    (4)
                                            279    (4)      280    (4)      281    (4)      282    (4)      283    (4)
                                            284    (4)      285    (4)      287    (4)      288    (4)      289    (4)
                                            290    (4)      291    (4)      292    (4)      293    (4)      294    (4)
                                            295    (4)      296    (4)      297    (4)      298    (4)      299    (4)
                                            300    (4)      301    (4)      302    (4)      303    (4)      304    (4)
                                            305    (4)      306    (4)      307    (4)      308    (4)      309    (4)
                                            310    (4)      311    (4)      312    (4)      313    (4)      314    (4)
                                            315    (4)      316    (4)      317    (4)      318    (4)      319    (4)
                                            320    (4)      321    (4)      322    (4)      323    (4)      324    (4)
                                            325    (4)      326    (4)      327    (4)      328    (4)      329    (4)
                                            330    (4)      331    (4)      332    (4)      333    (4)      334    (4)
                                            335    (4)      336    (4)      337    (4)      338    (4)      339    (4)
                                            340    (4)      341    (4)      342    (4)      343    (4)      344    (4)
                                            345    (4)      346    (4)      347    (4)      348    (4)      349    (4)
                                            350    (4)      351    (4)      352    (4)      353    (4)      354    (4)
                                            355    (4)      356    (4)      357    (4)      358    (4)      359    (4)
                                            360    (4)      361    (4)      362    (4)      363    (4)      364    (4)
                                            365    (4)      366    (4)      367    (4)      368    (4)      369    (4)
                                            370    (4)      371    (4)      372    (4)      373    (4)      374    (4)
                                            375    (4)      376    (4)      377    (4)      378    (4)      379    (4)
                                            380    (4)      381    (4)      382    (4)      383    (4)      384    (4)
                                            385    (4)      386    (4)      387    (4)      388    (4)      389    (4)
                                            390    (4)      391    (4)      392    (4)      393    (4)      394    (4)
                                            395    (4)      396    (4)      397    (4)      398    (4)
$DS_PTDDEF        1          249    (4)      249    (4)      250    (4)      251    (4)      252    (4)      253    (4)
                                            254    (4)      255    (4)      256    (4)      257    (4)      258    (4)
                                            259    (4)      260    (4)      261    (4)      262    (4)      263    (4)
                                            264    (4)      265    (4)      266    (4)      267    (4)      268    (4)
                                            269    (4)      270    (4)      271    (4)      272    (4)      273    (4)
```

				274	(4)	275	(4)	276	(4)	277	(4)	278	(4)
				279	(4)	280	(4)	281	(4)	282	(4)	283	(4)
				284	(4)	285	(4)	287	(4)	288	(4)	289	(4)
				290	(4)	291	(4)	292	(4)	293	(4)	294	(4)
				295	(4)	296	(4)	297	(4)	298	(4)	299	(4)
				300	(4)	301	(4)	302	(4)	303	(4)	304	(4)
				305	(4)	306	(4)	307	(4)	308	(4)	309	(4)
				310	(4)	311	(4)	312	(4)	313	(4)	314	(4)
				315	(4)	316	(4)	317	(4)	318	(4)	319	(4)
				320	(4)	321	(4)	322	(4)	323	(4)	324	(4)
				325	(4)	326	(4)	327	(4)	328	(4)	329	(4)
				330	(4)	331	(4)	332	(4)	333	(4)	334	(4)
				335	(4)	336	(4)	337	(4)	338	(4)	339	(4)
				340	(4)	341	(4)	342	(4)	343	(4)	344	(4)
				345	(4)	346	(4)	347	(4)	348	(4)	349	(4)
				350	(4)	351	(4)	352	(4)	353	(4)	354	(4)
				355	(4)	356	(4)	357	(4)	358	(4)	359	(4)
				360	(4)	361	(4)	362	(4)	363	(4)	364	(4)
				365	(4)	366	(4)	367	(4)	368	(4)	369	(4)
				370	(4)	371	(4)	372	(4)	373	(4)	374	(4)
				375	(4)	376	(4)	377	(4)	378	(4)	379	(4)
				380	(4)	381	(4)	382	(4)	383	(4)	384	(4)
				385	(4)	386	(4)	387	(4)	388	(4)	389	(4)
				390	(4)	391	(4)	392	(4)	393	(4)	394	(4)
				395	(4)	396	(4)	397	(4)	398	(4)		
$DS_PX780	2	327	(4)	327	(4)								
$DS_PX780_DEF	1	327	(4)	327	(4)								
$DS_R80	1	328	(4)	328	(4)								
$DS_R80_DEF	1	328	(4)	328	(4)								
$DS_RA60	1	329	(4)	329	(4)								
$DS_RA60_DEF	1	329	(4)	329	(4)								
$DS_RA70	1	330	(4)	330	(4)								
$DS_RA70_DEF	1	330	(4)	330	(4)								
$DS_RA80	1	331	(4)	331	(4)								
$DS_RA80_DEF	1	331	(4)	331	(4)								
$DS_RA81	1	332	(4)	332	(4)								
$DS_RA81_DEF	1	332	(4)	332	(4)								
$DS_RA82	1	333	(4)	333	(4)								
$DS_RA82_DEF	1	333	(4)	333	(4)								
$DS_RA90	1	200	(2)	334	(4)								
$DS_RA90_DEF	1	193	(2)	334	(4)								
$DS_RB730	2	335	(4)	335	(4)								
$DS_RB730_DEF	1	335	(4)	335	(4)								
$DS_RC25	1	337	(4)	337	(4)								
$DS_RC25_DEF	1	337	(4)	337	(4)								
$DS_RCF25	1	336	(4)	336	(4)								
$DS_RCF25_DEF	1	336	(4)	336	(4)								
$DS_RD51	1	338	(4)	338	(4)								
$DS_RD51_DEF	1	338	(4)	338	(4)								
$DS_RD52	1	339	(4)	339	(4)								
$DS_RD52_DEF	1	339	(4)	339	(4)								
$DS_RH750	2	340	(4)	340	(4)								
$DS_RH750_DEF	1	340	(4)	340	(4)								
$DS_RH780	2	341	(4)	341	(4)								
$DS_RH780_DEF	1	341	(4)	341	(4)								
$DS_RK06	1	345	(4)	345	(4)								
$DS_RK06_DEF	1	345	(4)	345	(4)								

```
$DS_RK07          1          346    (4)       346    (4)
$DS_RK07_DEF      1          346    (4)       346    (4)
$DS_RK611         1          347    (4)       347    (4)
$DS_RK611_DEF     1          347    (4)       347    (4)
$DS_RL01          1          342    (4)       342    (4)
$DS_RL01_DEF      1          342    (4)       342    (4)
$DS_RL02          1          343    (4)       343    (4)
$DS_RL02_DEF      1          343    (4)       343    (4)
$DS_RL11          1          344    (4)       344    (4)
$DS_RL11_DEF      1          344    (4)       344    (4)
$DS_RM03          1          348    (4)       348    (4)
$DS_RM03_DEF      1          348    (4)       348    (4)
$DS_RM05          1          349    (4)       349    (4)
$DS_RM05_DEF      1          349    (4)       349    (4)
$DS_RM80          1          350    (4)       350    (4)
$DS_RM80_DEF      1          350    (4)       350    (4)
$DS_RP04          1          351    (4)       351    (4)
$DS_RP04_DEF      1          351    (4)       351    (4)
$DS_RP05          1          352    (4)       352    (4)
$DS_RP05_DEF      1          352    (4)       352    (4)
$DS_RP06          1          353    (4)       353    (4)
$DS_RP06_DEF      1          353    (4)       353    (4)
$DS_RP07          1          354    (4)       354    (4)
$DS_RP07_DEF      1          354    (4)       354    (4)
$DS_RQDX1         1          355    (4)       355    (4)
$DS_RQDX1_DEF     1          355    (4)       355    (4)
$DS_RRD50         3          356    (4)       356    (4)
$DS_RRD50_DEF     1          356    (4)       356    (4)
$DS_RUX50         1          357    (4)       357    (4)
$DS_RUX50_DEF     1          357    (4)       357    (4)
$DS_RX02          1          358    (4)       358    (4)
$DS_RX02_DEF      1          358    (4)       358    (4)
$DS_RX180         1          363    (4)       363    (4)
$DS_RX180_DEF     1          363    (4)       363    (4)
$DS_RX211         1          359    (4)       359    (4)
$DS_RX211_DEF     1          359    (4)       359    (4)
$DS_RX31          1          360    (4)       360    (4)
$DS_RX31_DEF      1          360    (4)       360    (4)
$DS_RX32          1          361    (4)       361    (4)
$DS_RX32_DEF      1          361    (4)       361    (4)
$DS_RX50          1          362    (4)       362    (4)
$DS_RX50_DEF      1          362    (4)       362    (4)
$DS_SBIA          2          364    (4)       364    (4)
$DS_SBIA_DEF      1          364    (4)       364    (4)
$DS_TE16          1          365    (4)       365    (4)
$DS_TE16_DEF      1          365    (4)       365    (4)
$DS_TK50          1          366    (4)       366    (4)
$DS_TK50_DEF      1          366    (4)       366    (4)
$DS_TM03          1          367    (4)       367    (4)
$DS_TM03_DEF      1          367    (4)       367    (4)
$DS_TM78          1          368    (4)       368    (4)
$DS_TM78_DEF      1          368    (4)       368    (4)
$DS_TS04          1          369    (4)       369    (4)
$DS_TS04_DEF      1          369    (4)       369    (4)
$DS_TS05          2          370    (4)       370    (4)
$DS_TS05_DEF      1          370    (4)       370    (4)
$DS_TS11          1          371    (4)       371    (4)
```

$DS_TS11_DEF	1	371	(4)	371	(4)										
$DS_TU45	1	372	(4)	372	(4)										
$DS_TU45_DEF	1	372	(4)	372	(4)										
$DS_TU58	1	373	(4)	373	(4)										
$DS_TU58_DEF	1	373	(4)	373	(4)										
$DS_TU70	1	374	(4)	374	(4)										
$DS_TU70_DEF	1	374	(4)	374	(4)										
$DS_TU72	1	375	(4)	375	(4)										
$DS_TU72_DEF	1	375	(4)	375	(4)										
$DS_TU77	1	376	(4)	376	(4)										
$DS_TU77_DEF	1	376	(4)	376	(4)										
$DS_TU78	1	377	(4)	377	(4)										
$DS_TU78_DEF	1	377	(4)	377	(4)										
$DS_TU80	2	378	(4)	378	(4)										
$DS_TU80_DEF	1	378	(4)	378	(4)										
$DS_TU81	2	379	(4)	379	(4)										
$DS_TU81_DEF	1	379	(4)	379	(4)										
$DS_UBE	1	380	(4)	380	(4)										
$DS_UBE_DEF	1	380	(4)	380	(4)										
$DS_UDA50	1	381	(4)	381	(4)										
$DS_UDA50_DEF	1	381	(4)	381	(4)										
$DS_UNA11	2	382	(4)	382	(4)										
$DS_UNA11_DEF	1	382	(4)	382	(4)										
$DS_VS100	1	383	(4)	383	(4)										
$DS_VS100_DEF	1	383	(4)	383	(4)										
$DS_VS125	1	384	(4)	384	(4)										
$DS_VS125_DEF	1	384	(4)	384	(4)										
$DS_VS300	1	385	(4)	385	(4)										
$DS_VS300_DEF	1	385	(4)	385	(4)										
$DS_VT100	1	391	(4)	391	(4)										
$DS_VT100_DEF	1	391	(4)	391	(4)										
$DS_VT101	1	392	(4)	392	(4)										
$DS_VT101_DEF	1	392	(4)	392	(4)										
$DS_VT102	1	393	(4)	393	(4)										
$DS_VT102_DEF	1	393	(4)	393	(4)										
$DS_VT125	1	394	(4)	394	(4)										
$DS_VT125_DEF	1	394	(4)	394	(4)										
$DS_VT131	1	395	(4)	395	(4)										
$DS_VT131_DEF	1	395	(4)	395	(4)										
$DS_VT132	1	396	(4)	396	(4)										
$DS_VT132_DEF	1	396	(4)	396	(4)										
$DS_VT220	1	397	(4)	397	(4)										
$DS_VT220_DEF	1	397	(4)	397	(4)										
$DS_VT240	1	398	(4)	398	(4)										
$DS_VT240_DEF	1	398	(4)	398	(4)										
$DS_VT50	1	386	(4)	386	(4)										
$DS_VT50_DEF	1	386	(4)	386	(4)										
$DS_VT52	1	387	(4)	387	(4)										
$DS_VT52_DEF	1	387	(4)	387	(4)										
$DS_VT55	1	388	(4)	388	(4)										
$DS_VT55_DEF	1	388	(4)	388	(4)										
$DS_VT61	1	389	(4)	389	(4)										
$DS_VT61_DEF	1	389	(4)	389	(4)										
$DS_VT71	1	390	(4)	390	(4)										
$DS_VT71_DEF	1	390	(4)	390	(4)										
$EQU	1	398	(4)	249	(4)	250	(4)	251	(4)	252	(4)	253	(4)		
				254	(4)	255	(4)	256	(4)	257	(4)	258	(4)		

				259	(4)	260	(4)	261	(4)	262	(4)	263	(4)		
				264	(4)	265	(4)	266	(4)	267	(4)	268	(4)		
				269	(4)	270	(4)	271	(4)	272	(4)	273	(4)		
				274	(4)	275	(4)	276	(4)	277	(4)	278	(4)		
				279	(4)	280	(4)	281	(4)	282	(4)	283	(4)		
				284	(4)	285	(4)	287	(4)	288	(4)	289	(4)		
				290	(4)	291	(4)	292	(4)	293	(4)	294	(4)		
				295	(4)	296	(4)	297	(4)	298	(4)	299	(4)		
				300	(4)	301	(4)	302	(4)	303	(4)	304	(4)		
				305	(4)	306	(4)	307	(4)	308	(4)	309	(4)		
				310	(4)	311	(4)	312	(4)	313	(4)	314	(4)		
				315	(4)	316	(4)	317	(4)	318	(4)	319	(4)		
				320	(4)	321	(4)	322	(4)	323	(4)	324	(4)		
				325	(4)	326	(4)	327	(4)	328	(4)	329	(4)		
				330	(4)	331	(4)	332	(4)	333	(4)	334	(4)		
				335	(4)	336	(4)	337	(4)	338	(4)	339	(4)		
				340	(4)	341	(4)	342	(4)	343	(4)	344	(4)		
				345	(4)	346	(4)	347	(4)	348	(4)	349	(4)		
				350	(4)	351	(4)	352	(4)	353	(4)	354	(4)		
				355	(4)	356	(4)	357	(4)	358	(4)	359	(4)		
				360	(4)	361	(4)	362	(4)	363	(4)	364	(4)		
				365	(4)	366	(4)	367	(4)	368	(4)	369	(4)		
				370	(4)	371	(4)	372	(4)	373	(4)	374	(4)		
				375	(4)	376	(4)	377	(4)	378	(4)	379	(4)		
				380	(4)	381	(4)	382	(4)	383	(4)	384	(4)		
				385	(4)	386	(4)	387	(4)	388	(4)	389	(4)		
				390	(4)	391	(4)	392	(4)	393	(4)	394	(4)		
				395	(4)	396	(4)	397	(4)	398	(4)				
$EQULS1		1	398	(4)	249	(4)	250	(4)	251	(4)	252	(4)	253	(4)	
					254	(4)	255	(4)	256	(4)	257	(4)	258	(4)	
					259	(4)	260	(4)	261	(4)	262	(4)	263	(4)	
					264	(4)	265	(4)	266	(4)	267	(4)	268	(4)	
					269	(4)	270	(4)	271	(4)	272	(4)	273	(4)	
					274	(4)	275	(4)	276	(4)	277	(4)	278	(4)	
					279	(4)	280	(4)	281	(4)	282	(4)	283	(4)	
					284	(4)	285	(4)	287	(4)	288	(4)	289	(4)	
					290	(4)	291	(4)	292	(4)	293	(4)	294	(4)	
					295	(4)	296	(4)	297	(4)	298	(4)	299	(4)	
					300	(4)	301	(4)	302	(4)	303	(4)	304	(4)	
					305	(4)	306	(4)	307	(4)	308	(4)	309	(4)	
					310	(4)	311	(4)	312	(4)	313	(4)	314	(4)	
					315	(4)	316	(4)	317	(4)	318	(4)	319	(4)	
					320	(4)	321	(4)	322	(4)	323	(4)	324	(4)	
					325	(4)	326	(4)	327	(4)	328	(4)	329	(4)	
					330	(4)	331	(4)	332	(4)	333	(4)	334	(4)	
					335	(4)	336	(4)	337	(4)	338	(4)	339	(4)	
					340	(4)	341	(4)	342	(4)	343	(4)	344	(4)	
					345	(4)	346	(4)	347	(4)	348	(4)	349	(4)	
					350	(4)	351	(4)	352	(4)	353	(4)	354	(4)	
					355	(4)	356	(4)	357	(4)	358	(4)	359	(4)	
					360	(4)	361	(4)	362	(4)	363	(4)	364	(4)	
					365	(4)	366	(4)	367	(4)	368	(4)	369	(4)	
					370	(4)	371	(4)	372	(4)	373	(4)	374	(4)	
					375	(4)	376	(4)	377	(4)	378	(4)	379	(4)	
					380	(4)	381	(4)	382	(4)	383	(4)	384	(4)	
					385	(4)	386	(4)	387	(4)	388	(4)	389	(4)	
					390	(4)	391	(4)	392	(4)	393	(4)	394	(4)	

$EQULST 1 249 (4)

395 (4) 396 (4) 397 (4) 398 (4)
249 (4) 250 (4) 251 (4) 252 (4) 253 (4)
254 (4) 255 (4) 256 (4) 257 (4) 258 (4)
259 (4) 260 (4) 261 (4) 262 (4) 263 (4)
264 (4) 265 (4) 266 (4) 267 (4) 268 (4)
269 (4) 270 (4) 271 (4) 272 (4) 273 (4)
274 (4) 275 (4) 276 (4) 277 (4) 278 (4)
279 (4) 280 (4) 281 (4) 282 (4) 283 (4)
284 (4) 285 (4) 287 (4) 288 (4) 289 (4)
290 (4) 291 (4) 292 (4) 293 (4) 294 (4)
295 (4) 296 (4) 297 (4) 298 (4) 299 (4)
300 (4) 301 (4) 302 (4) 303 (4) 304 (4)
305 (4) 306 (4) 307 (4) 308 (4) 309 (4)
310 (4) 311 (4) 312 (4) 313 (4) 314 (4)
315 (4) 316 (4) 317 (4) 318 (4) 319 (4)
320 (4) 321 (4) 322 (4) 323 (4) 324 (4)
325 (4) 326 (4) 327 (4) 328 (4) 329 (4)
330 (4) 331 (4) 332 (4) 333 (4) 334 (4)
335 (4) 336 (4) 337 (4) 338 (4) 339 (4)
340 (4) 341 (4) 342 (4) 343 (4) 344 (4)
345 (4) 346 (4) 347 (4) 348 (4) 349 (4)
350 (4) 351 (4) 352 (4) 353 (4) 354 (4)
355 (4) 356 (4) 357 (4) 358 (4) 359 (4)
360 (4) 361 (4) 362 (4) 363 (4) 364 (4)
365 (4) 366 (4) 367 (4) 368 (4) 369 (4)
370 (4) 371 (4) 372 (4) 373 (4) 374 (4)
375 (4) 376 (4) 377 (4) 378 (4) 379 (4)
380 (4) 381 (4) 382 (4) 383 (4) 384 (4)
385 (4) 386 (4) 387 (4) 388 (4) 389 (4)
390 (4) 391 (4) 392 (4) 393 (4) 394 (4)
395 (4) 396 (4) 397 (4) 398 (4)

$GBLINI 2

249 (4) 250 (4) 251 (4) 252 (4) 253 (4)
254 (4) 255 (4) 256 (4) 257 (4) 258 (4)
259 (4) 260 (4) 261 (4) 262 (4) 263 (4)
264 (4) 265 (4) 266 (4) 267 (4) 268 (4)
269 (4) 270 (4) 271 (4) 272 (4) 273 (4)
274 (4) 275 (4) 276 (4) 277 (4) 278 (4)
279 (4) 280 (4) 281 (4) 282 (4) 283 (4)
284 (4) 285 (4) 287 (4) 288 (4) 289 (4)
290 (4) 291 (4) 292 (4) 293 (4) 294 (4)
295 (4) 296 (4) 297 (4) 298 (4) 299 (4)
300 (4) 301 (4) 302 (4) 303 (4) 304 (4)
305 (4) 306 (4) 307 (4) 308 (4) 309 (4)
310 (4) 311 (4) 312 (4) 313 (4) 314 (4)
315 (4) 316 (4) 317 (4) 318 (4) 319 (4)
320 (4) 321 (4) 322 (4) 323 (4) 324 (4)
325 (4) 326 (4) 327 (4) 328 (4) 329 (4)
330 (4) 331 (4) 332 (4) 333 (4) 334 (4)
335 (4) 336 (4) 337 (4) 338 (4) 339 (4)
340 (4) 341 (4) 342 (4) 343 (4) 344 (4)
345 (4) 346 (4) 347 (4) 348 (4) 349 (4)
350 (4) 351 (4) 352 (4) 353 (4) 354 (4)
355 (4) 356 (4) 357 (4) 358 (4) 359 (4)
360 (4) 361 (4) 362 (4) 363 (4) 364 (4)
365 (4) 366 (4) 367 (4) 368 (4) 369 (4)
370 (4) 371 (4) 372 (4) 373 (4) 374 (4)
375 (4) 376 (4) 377 (4) 378 (4) 379 (4)

				380	(4)	381	(4)	382	(4)	383	(4)	384	(4)
				385	(4)	386	(4)	387	(4)	388	(4)	389	(4)
				390	(4)	391	(4)	392	(4)	393	(4)	394	(4)
				395	(4)	396	(4)	397	(4)	398	(4)		
$VIELD	1			249	(4)	250	(4)	251	(4)	252	(4)	253	(4)
				254	(4)	255	(4)	256	(4)	257	(4)	258	(4)
				259	(4)	260	(4)	261	(4)	262	(4)	263	(4)
				264	(4)	265	(4)	266	(4)	267	(4)	268	(4)
				269	(4)	270	(4)	271	(4)	272	(4)	273	(4)
				274	(4)	275	(4)	276	(4)	277	(4)	278	(4)
				279	(4)	280	(4)	281	(4)	282	(4)	283	(4)
				284	(4)	285	(4)	287	(4)	288	(4)	289	(4)
				290	(4)	291	(4)	292	(4)	293	(4)	294	(4)
				295	(4)	296	(4)	297	(4)	298	(4)	299	(4)
				300	(4)	301	(4)	302	(4)	303	(4)	304	(4)
				305	(4)	306	(4)	307	(4)	308	(4)	309	(4)
				310	(4)	311	(4)	312	(4)	313	(4)	314	(4)
				315	(4)	316	(4)	317	(4)	318	(4)	319	(4)
				320	(4)	321	(4)	322	(4)	323	(4)	324	(4)
				325	(4)	326	(4)	327	(4)	328	(4)	329	(4)
				330	(4)	331	(4)	332	(4)	333	(4)	334	(4)
				335	(4)	336	(4)	337	(4)	338	(4)	339	(4)
				340	(4)	341	(4)	342	(4)	343	(4)	344	(4)
				345	(4)	346	(4)	347	(4)	348	(4)	349	(4)
				350	(4)	351	(4)	352	(4)	353	(4)	354	(4)
				355	(4)	356	(4)	357	(4)	358	(4)	359	(4)
				360	(4)	361	(4)	362	(4)	363	(4)	364	(4)
				365	(4)	366	(4)	367	(4)	368	(4)	369	(4)
				370	(4)	371	(4)	372	(4)	373	(4)	374	(4)
				375	(4)	376	(4)	377	(4)	378	(4)	379	(4)
				380	(4)	381	(4)	382	(4)	383	(4)	384	(4)
				385	(4)	386	(4)	387	(4)	388	(4)	389	(4)
				390	(4)	391	(4)	392	(4)	393	(4)	394	(4)
				395	(4)	396	(4)	397	(4)	398	(4)		
$VIELD1	1	398	(4)	249	(4)	250	(4)	251	(4)	252	(4)	253	(4)
				254	(4)	255	(4)	256	(4)	257	(4)	258	(4)
				259	(4)	260	(4)	261	(4)	262	(4)	263	(4)
				264	(4)	265	(4)	266	(4)	267	(4)	268	(4)
				269	(4)	270	(4)	271	(4)	272	(4)	273	(4)
				274	(4)	275	(4)	276	(4)	277	(4)	278	(4)
				279	(4)	280	(4)	281	(4)	282	(4)	283	(4)
				284	(4)	285	(4)	287	(4)	288	(4)	289	(4)
				290	(4)	291	(4)	292	(4)	293	(4)	294	(4)
				295	(4)	296	(4)	297	(4)	298	(4)	299	(4)
				300	(4)	301	(4)	302	(4)	303	(4)	304	(4)
				305	(4)	306	(4)	307	(4)	308	(4)	309	(4)
				310	(4)	311	(4)	312	(4)	313	(4)	314	(4)
				315	(4)	316	(4)	317	(4)	318	(4)	319	(4)
				320	(4)	321	(4)	322	(4)	323	(4)	324	(4)
				325	(4)	326	(4)	327	(4)	328	(4)	329	(4)
				330	(4)	331	(4)	332	(4)	333	(4)	334	(4)
				335	(4)	336	(4)	337	(4)	338	(4)	339	(4)
				340	(4)	341	(4)	342	(4)	343	(4)	344	(4)
				345	(4)	346	(4)	347	(4)	348	(4)	349	(4)
				350	(4)	351	(4)	352	(4)	353	(4)	354	(4)
				355	(4)	356	(4)	357	(4)	358	(4)	359	(4)
				360	(4)	361	(4)	362	(4)	363	(4)	364	(4)

365 (4) 366 (4) 367 (4) 368 (4) 369 (4)
370 (4) 371 (4) 372 (4) 373 (4) 374 (4)
375 (4) 376 (4) 377 (4) 378 (4) 379 (4)
380 (4) 381 (4) 382 (4) 383 (4) 384 (4)
385 (4) 386 (4) 387 (4) 388 (4) 389 (4)
390 (4) 391 (4) 392 (4) 393 (4) 394 (4)
395 (4) 396 (4) 397 (4) 398 (4)

Directives Cross Reference

DIRECTIVE	REFERENCES...
.ASCIC	249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 258 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4) 264 (4) 265 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 278 (4) 279 (4) 280 (4) 281 (4) 282 (4) 283 (4) 284 (4) 285 (4) 287 (4) 288 (4) 289 (4) 290 (4) 291 (4) 292 (4) 293 (4) 294 (4) 295 (4) 296 (4) 297 (4) 298 (4) 299 (4) 300 (4) 301 (4) 302 (4) 303 (4) 304 (4) 305 (4) 306 (4) 307 (4) 308 (4) 309 (4) 310 (4) 311 (4) 312 (4) 313 (4) 314 (4) 315 (4) 316 (4) 317 (4) 318 (4) 319 (4) 320 (4) 321 (4) 322 (4) 323 (4) 324 (4) 325 (4) 326 (4) 327 (4) 328 (4) 329 (4) 330 (4) 331 (4) 332 (4) 333 (4) 334 (4) 335 (4) 336 (4) 337 (4) 338 (4) 339 (4) 340 (4) 341 (4) 342 (4) 343 (4) 344 (4) 345 (4) 346 (4) 347 (4) 348 (4) 349 (4) 350 (4) 351 (4) 352 (4) 353 (4) 354 (4) 355 (4) 356 (4) 357 (4) 358 (4) 359 (4) 360 (4) 361 (4) 362 (4) 363 (4) 364 (4) 365 (4) 366 (4) 367 (4) 368 (4) 369 (4) 370 (4) 371 (4) 372 (4) 373 (4) 374 (4) 375 (4) 376 (4) 377 (4) 378 (4) 379 (4) 380 (4) 381 (4) 382 (4) 383 (4) 384 (4) 385 (4) 386 (4) 387 (4) 388 (4) 389 (4) 390 (4) 391 (4) 392 (4) 393 (4) 394 (4) 395 (4) 396 (4) 397 (4) 398 (4)
.BYTE	249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 258 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4) 264 (4) 265 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 278 (4) 279 (4) 280 (4) 281 (4) 282 (4) 283 (4) 284 (4) 285 (4) 287 (4) 288 (4) 289 (4) 290 (4) 291 (4) 292 (4) 293 (4) 294 (4) 295 (4) 296 (4) 297 (4) 298 (4) 299 (4) 300 (4) 301 (4) 302 (4) 303 (4) 304 (4) 305 (4) 306 (4) 307 (4) 308 (4) 309 (4) 310 (4) 311 (4) 312 (4) 313 (4) 314 (4) 315 (4) 316 (4) 317 (4) 318 (4) 319 (4) 320 (4) 321 (4) 322 (4) 323 (4) 324 (4) 325 (4) 326 (4) 327 (4) 328 (4) 329 (4) 330 (4) 331 (4) 332 (4) 333 (4) 334 (4) 335 (4) 336 (4) 337 (4) 338 (4) 339 (4) 340 (4) 341 (4) 342 (4) 343 (4) 344 (4) 345 (4) 346 (4) 347 (4) 348 (4) 349 (4) 350 (4) 351 (4) 352 (4) 353 (4) 354 (4) 355 (4) 356 (4) 357 (4) 358 (4) 359 (4) 360 (4) 361 (4) 362 (4) 363 (4) 364 (4) 365 (4) 366 (4) 367 (4) 368 (4) 369 (4) 370 (4) 371 (4) 372 (4) 373 (4) 374 (4) 375 (4) 376 (4) 377 (4) 378 (4) 379 (4) 380 (4) 381 (4) 382 (4) 383 (4) 384 (4) 385 (4) 386 (4) 387 (4) 388 (4) 389 (4) 390 (4) 391 (4) 392 (4) 393 (4) 394 (4) 395 (4) 396 (4) 397 (4) 398 (4)
.END	399 (4)
.ENDC	245 (3) 249 (4) 250 (4) 251 (4) 252 (4) 253 (4) 254 (4) 255 (4) 256 (4) 257 (4) 258 (4) 259 (4) 260 (4) 261 (4) 262 (4) 263 (4) 264 (4) 265 (4) 266 (4) 267 (4) 268 (4) 269 (4) 270 (4) 271 (4) 272 (4) 273 (4) 274 (4) 275 (4) 276 (4) 277 (4) 278 (4) 279 (4) 280 (4) 281 (4) 282 (4) 283 (4) 284 (4) 285 (4) 287 (4) 288 (4) 289 (4) 290 (4) 291 (4) 292 (4) 293 (4) 294 (4) 295 (4) 296 (4) 297 (4) 298 (4) 299 (4) 300 (4) 301 (4) 302 (4) 303 (4) 304 (4) 305 (4) 306 (4) 307 (4) 308 (4) 309 (4) 310 (4) 311 (4) 312 (4) 313 (4) 314 (4) 315 (4) 316 (4) 317 (4) 318 (4) 319 (4) 320 (4) 321 (4) 322 (4) 323 (4) 324 (4) 325 (4) 326 (4) 327 (4) 328 (4) 329 (4) 330 (4) 331 (4) 332 (4) 333 (4) 334 (4) 335 (4) 336 (4)

```
                337  (4)   338  (4)   339  (4)   340  (4)   341  (4)   342  (4)   343  (4)   344  (4)
                345  (4)   346  (4)   347  (4)   348  (4)   349  (4)   350  (4)   351  (4)   352  (4)
                353  (4)   354  (4)   355  (4)   356  (4)   357  (4)   358  (4)   359  (4)   360  (4)
                361  (4)   362  (4)   363  (4)   364  (4)   365  (4)   366  (4)   367  (4)   368  (4)
                369  (4)   370  (4)   371  (4)   372  (4)   373  (4)   374  (4)   375  (4)   376  (4)
                377  (4)   378  (4)   379  (4)   380  (4)   381  (4)   382  (4)   383  (4)   384  (4)
                385  (4)   386  (4)   387  (4)   388  (4)   389  (4)   390  (4)   391  (4)   392  (4)
                393  (4)   394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.ENDM           174  (2)   175  (2)   176  (2)   177  (2)   184  (2)   191  (2)   198  (2)   207  (2)
                217  (3)   249  (4)   250  (4)   251  (4)   252  (4)   253  (4)   254  (4)   255  (4)
                256  (4)   257  (4)   258  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)
                264  (4)   265  (4)   266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)
                272  (4)   273  (4)   274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)
                280  (4)   281  (4)   282  (4)   283  (4)   284  (4)   285  (4)   286  (4)   287  (4)
                288  (4)   289  (4)   290  (4)   291  (4)   292  (4)   293  (4)   294  (4)   295  (4)
                296  (4)   297  (4)   298  (4)   299  (4)   300  (4)   301  (4)   302  (4)   303  (4)
                304  (4)   305  (4)   306  (4)   307  (4)   308  (4)   309  (4)   310  (4)   311  (4)
                312  (4)   313  (4)   314  (4)   315  (4)   316  (4)   317  (4)   318  (4)   319  (4)
                320  (4)   321  (4)   322  (4)   323  (4)   324  (4)   325  (4)   326  (4)   327  (4)
                328  (4)   329  (4)   330  (4)   331  (4)   332  (4)   333  (4)   334  (4)   335  (4)
                336  (4)   337  (4)   338  (4)   339  (4)   340  (4)   341  (4)   342  (4)   343  (4)
                344  (4)   345  (4)   346  (4)   347  (4)   348  (4)   349  (4)   350  (4)   351  (4)
                352  (4)   353  (4)   354  (4)   355  (4)   356  (4)   357  (4)   358  (4)   359  (4)
                360  (4)   361  (4)   362  (4)   363  (4)   364  (4)   365  (4)   366  (4)   367  (4)
                368  (4)   369  (4)   370  (4)   371  (4)   372  (4)   373  (4)   374  (4)   375  (4)
                376  (4)   377  (4)   378  (4)   379  (4)   380  (4)   381  (4)   382  (4)   383  (4)
                384  (4)   385  (4)   386  (4)   387  (4)   388  (4)   389  (4)   390  (4)   391  (4)
                392  (4)   393  (4)   394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.ENDR           245  (3)   249  (4)   250  (4)   251  (4)   252  (4)   253  (4)   254  (4)   255  (4)
                256  (4)   257  (4)   258  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)
                264  (4)   265  (4)   266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)
                272  (4)   273  (4)   274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)
                280  (4)   281  (4)   282  (4)   283  (4)   284  (4)   285  (4)   287  (4)   288  (4)
                289  (4)   290  (4)   291  (4)   292  (4)   293  (4)   294  (4)   295  (4)   296  (4)
                297  (4)   298  (4)   299  (4)   300  (4)   301  (4)   302  (4)   303  (4)   304  (4)
                305  (4)   306  (4)   307  (4)   308  (4)   309  (4)   310  (4)   311  (4)   312  (4)
                313  (4)   314  (4)   315  (4)   316  (4)   317  (4)   318  (4)   319  (4)   320  (4)
                321  (4)   322  (4)   323  (4)   324  (4)   325  (4)   326  (4)   327  (4)   328  (4)
                329  (4)   330  (4)   331  (4)   332  (4)   333  (4)   334  (4)   335  (4)   336  (4)
                337  (4)   338  (4)   339  (4)   340  (4)   341  (4)   342  (4)   343  (4)   344  (4)
                345  (4)   346  (4)   347  (4)   348  (4)   349  (4)   350  (4)   351  (4)   352  (4)
                353  (4)   354  (4)   355  (4)   356  (4)   357  (4)   358  (4)   359  (4)   360  (4)
                361  (4)   362  (4)   363  (4)   364  (4)   365  (4)   366  (4)   367  (4)   368  (4)
                369  (4)   370  (4)   371  (4)   372  (4)   373  (4)   374  (4)   375  (4)   376  (4)
                377  (4)   378  (4)   379  (4)   380  (4)   381  (4)   382  (4)   383  (4)   384  (4)
                385  (4)   386  (4)   387  (4)   388  (4)   389  (4)   390  (4)   391  (4)   392  (4)
                393  (4)   394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.IDENT          13   (1)
.IF             245  (3)   249  (4)   250  (4)   251  (4)   252  (4)   253  (4)   254  (4)   255  (4)
                256  (4)   257  (4)   258  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)
                264  (4)   265  (4)   266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)
                272  (4)   273  (4)   274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)
                280  (4)   281  (4)   282  (4)   283  (4)   284  (4)   285  (4)   287  (4)   288  (4)
                289  (4)   290  (4)   291  (4)   292  (4)   293  (4)   294  (4)   295  (4)   296  (4)
                297  (4)   298  (4)   299  (4)   300  (4)   301  (4)   302  (4)   303  (4)   304  (4)
                305  (4)   306  (4)   307  (4)   308  (4)   309  (4)   310  (4)   311  (4)   312  (4)
                313  (4)   314  (4)   315  (4)   316  (4)   317  (4)   318  (4)   319  (4)   320  (4)
```

```
                321  (4)   322  (4)   323  (4)   324  (4)   325  (4)   326  (4)   327  (4)   328  (4)
                329  (4)   330  (4)   331  (4)   332  (4)   333  (4)   334  (4)   335  (4)   336  (4)
                337  (4)   338  (4)   339  (4)   340  (4)   341  (4)   342  (4)   343  (4)   344  (4)
                345  (4)   346  (4)   347  (4)   348  (4)   349  (4)   350  (4)   351  (4)   352  (4)
                353  (4)   354  (4)   355  (4)   356  (4)   357  (4)   358  (4)   359  (4)   360  (4)
                361  (4)   362  (4)   363  (4)   364  (4)   365  (4)   366  (4)   367  (4)   368  (4)
                369  (4)   370  (4)   371  (4)   372  (4)   373  (4)   374  (4)   375  (4)   376  (4)
                377  (4)   378  (4)   379  (4)   380  (4)   381  (4)   382  (4)   383  (4)   384  (4)
                385  (4)   386  (4)   387  (4)   388  (4)   389  (4)   390  (4)   391  (4)   392  (4)
                393  (4)   394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.IFF            249  (4)   250  (4)   251  (4)   252  (4)   253  (4)   254  (4)   255  (4)   256  (4)
                257  (4)   258  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)   264  (4)
                265  (4)   266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)   272  (4)
                273  (4)   274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)   280  (4)
                281  (4)   282  (4)   283  (4)   284  (4)   285  (4)   287  (4)   288  (4)   289  (4)
                290  (4)   291  (4)   292  (4)   293  (4)   294  (4)   295  (4)   296  (4)   297  (4)
                298  (4)   299  (4)   300  (4)   301  (4)   302  (4)   303  (4)   304  (4)   305  (4)
                306  (4)   307  (4)   308  (4)   309  (4)   310  (4)   311  (4)   312  (4)   313  (4)
                314  (4)   315  (4)   316  (4)   317  (4)   318  (4)   319  (4)   320  (4)   321  (4)
                322  (4)   323  (4)   324  (4)   325  (4)   326  (4)   327  (4)   328  (4)   329  (4)
                330  (4)   331  (4)   332  (4)   333  (4)   334  (4)   335  (4)   336  (4)   337  (4)
                338  (4)   339  (4)   340  (4)   341  (4)   342  (4)   343  (4)   344  (4)   345  (4)
                346  (4)   347  (4)   348  (4)   349  (4)   350  (4)   351  (4)   352  (4)   353  (4)
                354  (4)   355  (4)   356  (4)   357  (4)   358  (4)   359  (4)   360  (4)   361  (4)
                362  (4)   363  (4)   364  (4)   365  (4)   366  (4)   367  (4)   368  (4)   369  (4)
                370  (4)   371  (4)   372  (4)   373  (4)   374  (4)   375  (4)   376  (4)   377  (4)
                378  (4)   379  (4)   380  (4)   381  (4)   382  (4)   383  (4)   384  (4)   385  (4)
                386  (4)   387  (4)   388  (4)   389  (4)   390  (4)   391  (4)   392  (4)   393  (4)
                394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.IIF            249  (4)   250  (4)   251  (4)   252  (4)   253  (4)   254  (4)   255  (4)   256  (4)
                257  (4)   258  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)   264  (4)
                265  (4)   266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)   272  (4)
                273  (4)   274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)   280  (4)
                281  (4)   282  (4)   283  (4)   284  (4)   285  (4)   287  (4)   288  (4)   289  (4)
                290  (4)   291  (4)   292  (4)   293  (4)   294  (4)   295  (4)   296  (4)   297  (4)
                298  (4)   299  (4)   300  (4)   301  (4)   302  (4)   303  (4)   304  (4)   305  (4)
                306  (4)   307  (4)   308  (4)   309  (4)   310  (4)   311  (4)   312  (4)   313  (4)
                314  (4)   315  (4)   316  (4)   317  (4)   318  (4)   319  (4)   320  (4)   321  (4)
                322  (4)   323  (4)   324  (4)   325  (4)   326  (4)   327  (4)   328  (4)   329  (4)
                330  (4)   331  (4)   332  (4)   333  (4)   334  (4)   335  (4)   336  (4)   337  (4)
                338  (4)   339  (4)   340  (4)   341  (4)   342  (4)   343  (4)   344  (4)   345  (4)
                346  (4)   347  (4)   348  (4)   349  (4)   350  (4)   351  (4)   352  (4)   353  (4)
                354  (4)   355  (4)   356  (4)   357  (4)   358  (4)   359  (4)   360  (4)   361  (4)
                362  (4)   363  (4)   364  (4)   365  (4)   366  (4)   367  (4)   368  (4)   369  (4)
                370  (4)   371  (4)   372  (4)   373  (4)   374  (4)   375  (4)   376  (4)   377  (4)
                378  (4)   379  (4)   380  (4)   381  (4)   382  (4)   383  (4)   384  (4)   385  (4)
                386  (4)   387  (4)   388  (4)   389  (4)   390  (4)   391  (4)   392  (4)   393  (4)
                394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.IRP            245  (3)   249  (4)   250  (4)   251  (4)   252  (4)   253  (4)   254  (4)   255  (4)
                256  (4)   257  (4)   258  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)
                264  (4)   265  (4)   266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)
                272  (4)   273  (4)   274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)
                280  (4)   281  (4)   282  (4)   283  (4)   284  (4)   285  (4)   287  (4)   288  (4)
                289  (4)   290  (4)   291  (4)   292  (4)   293  (4)   294  (4)   295  (4)   296  (4)
                297  (4)   298  (4)   299  (4)   300  (4)   301  (4)   302  (4)   303  (4)   304  (4)
                305  (4)   306  (4)   307  (4)   308  (4)   309  (4)   310  (4)   311  (4)   312  (4)
                313  (4)   314  (4)   315  (4)   316  (4)   317  (4)   318  (4)   319  (4)   320  (4)
```

```
                321     (4)     322     (4)     323     (4)     324     (4)     325     (4)     326     (4)     327     (4)     328     (4)
                329     (4)     330     (4)     331     (4)     332     (4)     333     (4)     334     (4)     335     (4)     336     (4)
                337     (4)     338     (4)     339     (4)     340     (4)     341     (4)     342     (4)     343     (4)     344     (4)
                345     (4)     346     (4)     347     (4)     348     (4)     349     (4)     350     (4)     351     (4)     352     (4)
                353     (4)     354     (4)     355     (4)     356     (4)     357     (4)     358     (4)     359     (4)     360     (4)
                361     (4)     362     (4)     363     (4)     364     (4)     365     (4)     366     (4)     367     (4)     368     (4)
                369     (4)     370     (4)     371     (4)     372     (4)     373     (4)     374     (4)     375     (4)     376     (4)
                377     (4)     378     (4)     379     (4)     380     (4)     381     (4)     382     (4)     383     (4)     384     (4)
                385     (4)     386     (4)     387     (4)     388     (4)     389     (4)     390     (4)     391     (4)     392     (4)
                393     (4)     394     (4)     395     (4)     396     (4)     397     (4)     398     (4)
.LIBRARY        172     (2)
.LIST           14      (1)     247     (4)
.LONG           245     (3)     249     (4)     250     (4)     251     (4)     252     (4)     253     (4)     255     (4)     256     (4)
                257     (4)     258     (4)     260     (4)     261     (4)     262     (4)     263     (4)     264     (4)     265     (4)
                266     (4)     267     (4)     268     (4)     269     (4)     270     (4)     271     (4)     272     (4)     273     (4)
                274     (4)     275     (4)     276     (4)     277     (4)     278     (4)     279     (4)     280     (4)     281     (4)
                282     (4)     283     (4)     284     (4)     285     (4)     287     (4)     288     (4)     289     (4)     290     (4)
                291     (4)     292     (4)     293     (4)     294     (4)     295     (4)     296     (4)     297     (4)     298     (4)
                299     (4)     300     (4)     301     (4)     302     (4)     303     (4)     304     (4)     305     (4)     306     (4)
                307     (4)     308     (4)     309     (4)     310     (4)     311     (4)     312     (4)     313     (4)     314     (4)
                315     (4)     316     (4)     317     (4)     318     (4)     319     (4)     320     (4)     321     (4)     323     (4)
                325     (4)     326     (4)     327     (4)     328     (4)     329     (4)     330     (4)     331     (4)     332     (4)
                333     (4)     334     (4)     335     (4)     336     (4)     337     (4)     338     (4)     339     (4)     340     (4)
                341     (4)     342     (4)     343     (4)     344     (4)     345     (4)     346     (4)     347     (4)     348     (4)
                349     (4)     350     (4)     351     (4)     352     (4)     353     (4)     354     (4)     355     (4)     356     (4)
                357     (4)     358     (4)     359     (4)     362     (4)     364     (4)     365     (4)     366     (4)     367     (4)
                368     (4)     369     (4)     370     (4)     371     (4)     372     (4)     373     (4)     374     (4)     375     (4)
                376     (4)     377     (4)     378     (4)     379     (4)     380     (4)     381     (4)     382     (4)     383     (4)
                384     (4)     385     (4)     386     (4)     387     (4)     388     (4)     389     (4)     390     (4)     391     (4)
                392     (4)     393     (4)     394     (4)     395     (4)     396     (4)     397     (4)     398     (4)
.MACRO          179     (2)     186     (2)     193     (2)     200     (2)     249     (4)     250     (4)     251     (4)     252     (4)
                253     (4)     254     (4)     255     (4)     256     (4)     257     (4)     258     (4)     259     (4)     260     (4)
                261     (4)     262     (4)     263     (4)     264     (4)     265     (4)     266     (4)     267     (4)     268     (4)
                269     (4)     270     (4)     271     (4)     272     (4)     273     (4)     274     (4)     275     (4)     276     (4)
                277     (4)     278     (4)     279     (4)     280     (4)     281     (4)     282     (4)     283     (4)     284     (4)
                285     (4)     287     (4)     288     (4)     289     (4)     290     (4)     291     (4)     292     (4)     293     (4)
                294     (4)     295     (4)     296     (4)     297     (4)     298     (4)     299     (4)     300     (4)     301     (4)
                302     (4)     303     (4)     304     (4)     305     (4)     306     (4)     307     (4)     308     (4)     309     (4)
                310     (4)     311     (4)     312     (4)     313     (4)     314     (4)     315     (4)     316     (4)     317     (4)
                318     (4)     319     (4)     320     (4)     321     (4)     322     (4)     323     (4)     324     (4)     325     (4)
                326     (4)     327     (4)     328     (4)     329     (4)     330     (4)     331     (4)     332     (4)     333     (4)
                334     (4)     335     (4)     336     (4)     337     (4)     338     (4)     339     (4)     340     (4)     341     (4)
                342     (4)     343     (4)     344     (4)     345     (4)     346     (4)     347     (4)     348     (4)     349     (4)
                350     (4)     351     (4)     352     (4)     353     (4)     354     (4)     355     (4)     356     (4)     357     (4)
                358     (4)     359     (4)     360     (4)     361     (4)     362     (4)     363     (4)     364     (4)     365     (4)
                366     (4)     367     (4)     368     (4)     369     (4)     370     (4)     371     (4)     372     (4)     373     (4)
                374     (4)     375     (4)     376     (4)     377     (4)     378     (4)     379     (4)     380     (4)     381     (4)
                382     (4)     383     (4)     384     (4)     385     (4)     386     (4)     387     (4)     388     (4)     389     (4)
                390     (4)     391     (4)     392     (4)     393     (4)     394     (4)     395     (4)     396     (4)     397     (4)
                398     (4)
.MDELETE        245     (3)
.NLIST          15      (1)
.NOCROSS        174     (2)     175     (2)     176     (2)     177     (2)     249     (4)     250     (4)     254     (4)     255     (4)
                256     (4)     259     (4)     260     (4)     261     (4)     262     (4)     263     (4)     264     (4)     265     (4)
                266     (4)     267     (4)     268     (4)     269     (4)     270     (4)     271     (4)     272     (4)     273     (4)
                274     (4)     275     (4)     276     (4)     277     (4)     278     (4)     279     (4)     280     (4)     281     (4)
                282     (4)     283     (4)     284     (4)     285     (4)     286     (4)     292     (4)     293     (4)     294     (4)
```

```
                295  (4)   296  (4)   297  (4)   298  (4)   299  (4)   300  (4)   301  (4)   302  (4)
                303  (4)   304  (4)   305  (4)   306  (4)   307  (4)   308  (4)   309  (4)   310  (4)
                311  (4)   312  (4)   313  (4)   314  (4)   315  (4)   316  (4)   317  (4)   318  (4)
                319  (4)   320  (4)   321  (4)   322  (4)   323  (4)   324  (4)   325  (4)   326  (4)
                327  (4)   328  (4)   329  (4)   330  (4)   331  (4)   332  (4)   333  (4)   334  (4)
                335  (4)   336  (4)   337  (4)   338  (4)   339  (4)   340  (4)   341  (4)   342  (4)
                343  (4)   344  (4)   345  (4)   346  (4)   347  (4)   348  (4)   349  (4)   350  (4)
                351  (4)   352  (4)   353  (4)   354  (4)   355  (4)   356  (4)   357  (4)   358  (4)
                359  (4)   360  (4)   361  (4)   362  (4)   363  (4)   364  (4)   365  (4)   366  (4)
                367  (4)   368  (4)   369  (4)   370  (4)   371  (4)   372  (4)   373  (4)   374  (4)
                375  (4)   376  (4)   377  (4)   378  (4)   379  (4)   380  (4)   381  (4)   382  (4)
                383  (4)   384  (4)   385  (4)   386  (4)   387  (4)   388  (4)   389  (4)   390  (4)
                391  (4)   392  (4)   393  (4)   394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.RESTORE        174  (2)   175  (2)   176  (2)   177  (2)   249  (4)   250  (4)   254  (4)   255  (4)
                256  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)   264  (4)   265  (4)
                266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)   272  (4)   273  (4)
                274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)   280  (4)   281  (4)
                282  (4)   283  (4)   284  (4)   285  (4)   286  (4)   292  (4)   293  (4)   294  (4)
                295  (4)   296  (4)   297  (4)   298  (4)   299  (4)   300  (4)   301  (4)   302  (4)
                303  (4)   304  (4)   305  (4)   306  (4)   307  (4)   308  (4)   309  (4)   310  (4)
                311  (4)   312  (4)   313  (4)   314  (4)   315  (4)   316  (4)   317  (4)   318  (4)
                319  (4)   320  (4)   321  (4)   322  (4)   323  (4)   324  (4)   325  (4)   326  (4)
                327  (4)   328  (4)   329  (4)   330  (4)   331  (4)   332  (4)   333  (4)   334  (4)
                335  (4)   336  (4)   337  (4)   338  (4)   339  (4)   340  (4)   341  (4)   342  (4)
                343  (4)   344  (4)   345  (4)   346  (4)   347  (4)   348  (4)   349  (4)   350  (4)
                351  (4)   352  (4)   353  (4)   354  (4)   355  (4)   356  (4)   357  (4)   358  (4)
                359  (4)   360  (4)   361  (4)   362  (4)   363  (4)   364  (4)   365  (4)   366  (4)
                367  (4)   368  (4)   369  (4)   370  (4)   371  (4)   372  (4)   373  (4)   374  (4)
                375  (4)   376  (4)   377  (4)   378  (4)   379  (4)   380  (4)   381  (4)   382  (4)
                383  (4)   384  (4)   385  (4)   386  (4)   387  (4)   388  (4)   389  (4)   390  (4)
                391  (4)   392  (4)   393  (4)   394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.SAVE           174  (2)   175  (2)   176  (2)   177  (2)   249  (4)   250  (4)   254  (4)   255  (4)
                256  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)   264  (4)   265  (4)
                266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)   272  (4)   273  (4)
                274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)   280  (4)   281  (4)
                282  (4)   283  (4)   284  (4)   285  (4)   286  (4)   292  (4)   293  (4)   294  (4)
                295  (4)   296  (4)   297  (4)   298  (4)   299  (4)   300  (4)   301  (4)   302  (4)
                303  (4)   304  (4)   305  (4)   306  (4)   307  (4)   308  (4)   309  (4)   310  (4)
                311  (4)   312  (4)   313  (4)   314  (4)   315  (4)   316  (4)   317  (4)   318  (4)
                319  (4)   320  (4)   321  (4)   322  (4)   323  (4)   324  (4)   325  (4)   326  (4)
                327  (4)   328  (4)   329  (4)   330  (4)   331  (4)   332  (4)   333  (4)   334  (4)
                335  (4)   336  (4)   337  (4)   338  (4)   339  (4)   340  (4)   341  (4)   342  (4)
                343  (4)   344  (4)   345  (4)   346  (4)   347  (4)   348  (4)   349  (4)   350  (4)
                351  (4)   352  (4)   353  (4)   354  (4)   355  (4)   356  (4)   357  (4)   358  (4)
                359  (4)   360  (4)   361  (4)   362  (4)   363  (4)   364  (4)   365  (4)   366  (4)
                367  (4)   368  (4)   369  (4)   370  (4)   371  (4)   372  (4)   373  (4)   374  (4)
                375  (4)   376  (4)   377  (4)   378  (4)   379  (4)   380  (4)   381  (4)   382  (4)
                383  (4)   384  (4)   385  (4)   386  (4)   387  (4)   388  (4)   389  (4)   390  (4)
                391  (4)   392  (4)   393  (4)   394  (4)   395  (4)   396  (4)   397  (4)   398  (4)
.SBTTL          167  (2)   210  (3)
.TITLE           12  (1)
.WORD           249  (4)   250  (4)   251  (4)   252  (4)   253  (4)   254  (4)   255  (4)   256  (4)
                257  (4)   258  (4)   259  (4)   260  (4)   261  (4)   262  (4)   263  (4)   264  (4)
                265  (4)   266  (4)   267  (4)   268  (4)   269  (4)   270  (4)   271  (4)   272  (4)
                273  (4)   274  (4)   275  (4)   276  (4)   277  (4)   278  (4)   279  (4)   280  (4)
                281  (4)   282  (4)   283  (4)   284  (4)   285  (4)   287  (4)   288  (4)   289  (4)
                290  (4)   291  (4)   292  (4)   293  (4)   294  (4)   295  (4)   296  (4)   297  (4)
```

298	(4)	299	(4)	300	(4)	301	(4)	302	(4)	303	(4)	304	(4)	305	(4)
306	(4)	307	(4)	308	(4)	309	(4)	310	(4)	311	(4)	312	(4)	313	(4)
314	(4)	315	(4)	316	(4)	317	(4)	318	(4)	319	(4)	320	(4)	321	(4)
322	(4)	323	(4)	324	(4)	325	(4)	326	(4)	327	(4)	328	(4)	329	(4)
330	(4)	331	(4)	332	(4)	333	(4)	334	(4)	335	(4)	336	(4)	337	(4)
338	(4)	339	(4)	340	(4)	341	(4)	342	(4)	343	(4)	344	(4)	345	(4)
346	(4)	347	(4)	348	(4)	349	(4)	350	(4)	351	(4)	352	(4)	353	(4)
354	(4)	355	(4)	356	(4)	357	(4)	358	(4)	359	(4)	360	(4)	361	(4)
362	(4)	363	(4)	364	(4)	365	(4)	366	(4)	367	(4)	368	(4)	369	(4)
370	(4)	371	(4)	372	(4)	373	(4)	374	(4)	375	(4)	376	(4)	377	(4)
378	(4)	379	(4)	380	(4)	381	(4)	382	(4)	383	(4)	384	(4)	385	(4)
386	(4)	387	(4)	388	(4)	389	(4)	390	(4)	391	(4)	392	(4)	393	(4)
394	(4)	395	(4)	396	(4)	397	(4)	398	(4)						

```
+-------------------------+
! Performance indicators !
+-------------------------+
```

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.02	00:00:00.17
Command processing	843	00:00:00.12	00:00:00.58
Pass 1	7854	00:00:45.61	00:00:47.97
Symbol table sort	0	00:00:00.17	00:00:00.17
Pass 2	5113	00:00:05.87	00:00:09.64
Symbol table output	82	00:00:00.11	00:00:00.30
Psect synopsis output	5	00:00:00.00	00:00:00.00
Cross-reference output	905	00:00:02.60	00:00:03.67
Assembler run totals	14831	00:00:54.51	00:01:02.51

The working set limit was 4096 pages.
2258249 bytes (4411 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 697 non-local and 0 local symbols.
399 source lines were read in Pass 1, producing 93 object records in Pass 2.
1981 pages of virtual memory were used to define 462 macros.

```
+---------------------------+
! Macro library statistics !
+---------------------------+
```

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;8	306
SYS$COMMON:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	311

3118 GETS were required to define 311 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$$:/LIST=DS$$: DS$$:ICODE.MAR

Table of contents

(2) 383 DECLARATIONS
(5) 453 Device description database

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element IOBASE.MAR
0000     2 ;  50A1 11-NOV-1987 08:50:46 MEIER "FREEZE"
0000     3 ; *54   28-OCT-1987 09:07:20 GEARIN "CORRECT GENERATION"
0000     4 ; *53   27-OCT-1987 16:05:53 GEARIN "BACK UP GEN"
0000     5 ; *52   26-OCT-1987 12:50:35 GEARIN "FIX LIBRARY STATEMENT"
0000     6 ; *51   26-OCT-1987 12:30:17 GEARIN "ADDED RA90 PTABLE"
0000     7 ; *50   19-OCT-1987 08:52:00 MEIER "CHANGES TO DIAG"
0000     8 ; *49   14-OCT-1987 06:30:49 MEIER "RECOMPILED"
0000     9 ; *48    7-OCT-1987 20:10:01 MARTIN "ADDED A COMMENT FOR PREVIOUS FIX"
0000    10 ; *47    7-OCT-1987 19:49:43 MARTIN "REMOVED CALL TO $DS_DEVTYP"
0000    11 ; *46   29-SEP-1987 12:59:22 MEIER "CHANGES TO DIAG"
0000    12 ; *45   12-AUG-1987 10:00:54 BERGAZZI "CHANGED .LIB BACK TO $DIAG FROM WORK$:DIAG"
0000    13 ; *44   10-AUG-1987 16:52:16 BARANSKI "Merging Gen:43A1 and Gen:43B1 to Gen:44"
0000    14 ;  43B1 30-JUL-1987 16:58:33 MARTIN "KA888, AND DS_KA_DEF"
0000    15 ;  43A1 30-JUL-1987 01:13:50 MARTIN "KARRR TO KA888"
0000    16 ; *43   13-JUL-1987 09:13:10 GEARIN "ADDED RV20 PTABLE"
0000    17 ; *42    6-JUL-1987 11:04:52 GEARIN "ADDED RV20 P-TABLE"
0000    18 ; *41   22-MAY-1987 09:54:43 BERGAZZI "ADDED DEBNA, DEBNK PTABLES"
0000    19 ; *40    7-MAY-1987 14:46:14 MEIER "CHANGES TO DIAG"
0000    20 ; *39   10-APR-1987 16:18:44 BAGGETT "CHANGE FINISHED"
0000    21 ; *38   26-MAR-1987 11:05:48 MEIER "REMOVED REFERENCES TO LLL"
0000    22 ; *37   19-MAR-1987 15:55:36 MARTIN "TOOK OUT DIRECT REFERENCES TO JJJ"
0000    23 ; *36   19-MAR-1987 09:07:59 MEIER "RETRY"
0000    24 ; *35   18-MAR-1987 15:22:52 MEIER "RETRY"
0000    25 ; *34   18-MAR-1987 13:55:10 MEIER "CHANGES TO DIAG"
0000    26 ; *33   13-MAR-1987 13:54:17 MARTIN "TOO CONFUSING AND STUPID TO MENTION"
0000    27 ; *32   13-MAR-1987 13:45:06 MARTIN "FIXED MY LAST EDIT HISTORY"
0000    28 ; *31    4-MAR-1987 13:26:51 BERGAZZI "ADD RA70 PTABLE"
0000    29 ; *30   11-FEB-1987 09:10:26 GEARIN "ADDED PTABLE INFORMATION (DRB32)"
0000    30 ; *29    9-FEB-1987 16:07:04 MARTIN "RE-COMPILED"
0000    31 ; *28    4-FEB-1987 10:57:02 MARTIN "ADDED $DS_KARRR"
0000    32 ; *27   30-JAN-1987 09:03:40 GEARIN "ADDED PTABLE ENTRIES"
0000    33 ; *26   18-DEC-1986 17:27:36 BAGGETT "MODULE RECOMPILED FOR CIBCA CHANGES"
0000    34 ; *25   16-DEC-1986 16:55:11 BAGGETT "FINISHED"
0000    35 ; *24   14-NOV-1986 13:58:21 BAGGETT "<no reason given>"
0000    36 ; *23   14-NOV-1986 13:26:44 BAGGETT "INSERT NEW DEFINITION FOR CIBCA"
0000    37 ; *22   14-NOV-1986 12:14:14 BAGGETT "INSERT NEW P-TABLE DEFINITION FOR TU81"
0000    38 ; *21   31-OCT-1986 12:39:09 MEIER "CHANGES TO DIAG"
0000    39 ; *20   30-OCT-1986 10:30:32 MEIER "OOPS TRY AGAIN"
0000    40 ; *19   30-OCT-1986 09:01:28 MEIER "CHANGES TO DIAG"
0000    41 ; *18   10-OCT-1986 14:37:12 BAGGETT "RECOMPILE FOR NEW KA800 DEFINITIONS"
0000    42 ; *17   17-SEP-1986 15:06:24 BERGAZZI "FORMAT CHANGE FOR VSDP"
0000    43 ; *16   15-AUG-1986 10:57:54 BAGGETT "<no reason given>"
0000    44 ; *15   23-JUN-1986 09:11:27 MARTIN "DONE"
0000    45 ; *14   10-JUN-1986 14:19:39 BAGGETT "<no reason given>"
0000    46 ; *13   20-MAY-1986 10:43:52 MUSE "<no reason given>"
0000    47 ; *12    5-MAY-1986 16:56:44 BAGGETT "<no reason given>"
0000    48 ; *11   30-APR-1986 15:15:29 BERGAZZI "DEBNT FIX"
0000    49 ; *10   30-APR-1986 15:06:22 BAGGETT "FINISHED"
0000    50 ; *9    17-MAR-1986 11:05:48 MUSE "<no reason given>"
0000    51 ; *8    17-MAR-1986 10:05:48 BAGGETT "CHANGES FOR KFBTA AND DEBNT P-TABLES"
0000    52 ; *7    13-MAR-1986 14:02:12 MUSE "<no reason given>"
0000    53 ; *6    13-MAR-1986 13:29:58 MUSE "<no reason given>"
0000    54 ; *5    13-MAR-1986 13:22:17 MUSE "<no reason given>"
0000    55 ; *4    12-MAR-1986 16:52:14 BARANSKI "Add RRD50 PTable."
0000    56 ; *3    26-FEB-1986 16:39:32 MUSE "BECAUSE IT WAS THERE"
0000    57 ; *2    12-FEB-1986 16:59:26 BERGAZZI "CI FUNCTIONALITY CHANGE"
```

```
0000     58 ; *1     6-FEB-1986 15:18:54 DS ""
0000     59 ; DEC/CMS REPLACEMENT HISTORY, Element IOBASE.MAR
0000     60         .TITLE  IOBASE  *** IOBASE I/O data base
0000     61         .ident  "10.10-47"                                          ;G:47
0000     62         .LIST   MEB
0000     63         .DSABL  GBL
0000     64         .NLIST  CND
0000     65
0000     66 ;
0000     67 ;
0000     68 ; Copyright (c) 1977, 1982, 1983, 1984, 1985, 1986, 1987            ;G:27
0000     69 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
0000     70 ;
0000     71 ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
0000     72 ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
0000     73 ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
0000     74 ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
0000     75 ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
0000     76 ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
0000     77 ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
0000     78 ;
0000     79 ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
0000     80 ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
0000     81 ; EQUIPMENT CORPORATION.
0000     82 ;
0000     83 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000     84 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000     85 ;
0000     86 ; R.HEINEN 3-AUG-76
0000     87 ;
0000     88 ; MODIFICATION HISTORY:
0000     89 ;
0000     90 ;                   N. HOWGATE 3-FEB-78     VERSION 02 (ESSAA-4.00).
0000     91 ;         01        ALTERED TO REPRESENT A ONE (4) DBA: SYSTEM
0000     92 ;                   FOR INITIAL DIAGNOSTIC SUPERVISOR QIO DEBUG
0000     93 ;
0000     94 ;                   N. HOWGATE  20-APR-78  VERSION 03  (ESSAA-4.01).
0000     95 ;         02        ADDED DEVICE DATA FOR ONE DMA: SYSTEM (3 UNITS)
0000     96 ;
0000     97 ;                   TOM SOUTTER  12-MAY-78  VERSION 04  (ESSAA-4.02).
0000     98 ;         03        ADDED DEVICE DATA FOR ONE DRA: SYSTEM (4 UNITS)
0000     99 ;
0000    100 ;                   TOM SOUTTER  14-JUN-78  VERSION 05  (ESSAA-4.03).
0000    101 ;         04        ADDED DEVICE DATA FOR ONE MTA: SYSTEM (2 UNITS)
0000    102 ;
0000    103 ;                   NICK HOWGATE    22-JUN-78
0000    104 ;         05        ADD LONGWORD TO END OF RM03 UCB'S FOR ADDITIONAL STATUS
0000    105 ;
0000    106 ;                   NICK HOWGATE    11-JUL-78
0000    107 ;         06        ADD RK07 SUPPORT IN DM TYPE UCB
0000    108 ;
0000    109 ;                   NICK HOWGATE    13-JUL-78
0000    110 ;         07        ADD LONGWORD TO CRB FOR UNIT INIT CODE
0000    111 ;                   CRB$L_INTD+VEC$L_UNITINIT
0000    112 ;
0000    113 ;                   NICK HOWGATE    24-AUG-78
0000    114 ;         08        INCLUDE NEW UCB ENTRIES (UCB$W_ERRCNT ,UCB$B_FEX, UCB$B_CEX)
```

```
0000  ;
0000  ;                Roger Riggs     30-SEP-78       VERSION 6 (ESSAA 5.00)
0000  ;        09      Major modifications to accomodate new QIO initialization
0000  ;                and loadable drivers.
0000  ;                                                                                ;G:40
0000  ;                Roger Riggs     1-OCT-1979
0000  ;        10      Added DMP11,DMR11,DV11,LA180,MS780
0000  ;
0000  ;        11      John Ciukaj     1-15-80
0000  ;                Added DL11,RX02,RX211,TU58,TS11
0000  ;
0000  ;        12      M. Baggett      18-MAR-80
0000  ;                Add TU78 magtape
0000  ;
0000  ;                Dave Butenhof, 15-dec-1980, V6.2
0000  ;        13      Add ML11 PTDESC.
0000  ;
0000  ;                Dave Butenhof, 19-mar-1981, version 6.3
0000  ;        14      Add DV11 PTdesc.
0000  ;        15      Add KA730 Ptdesc.
0000  ;
0000  ;        16      Dave butenhof, 4-sep-1981, version 6.5
0000  ;                add RB730 and R80 Ptable descriptors
0000  ;
0000  ;        17      Dave Butenhof, 26-Oct-1981, version 6.-
0000  ;                Add some missing devices
0000  ;
0000  ;        18      Dave Butenhof, 10-Nov-1981, version 6.-
0000  ;                Add the UDA50 and RA80 devices
0000  ;
0000  ;        19      Dave Butenhof, 15-Dec-1981, Version 6.6
0000  ;                Add PX780
0000  ;
0000  ;        20      Dave Butenhof, 23-Feb-1982, version 6.6
0000  ;                Add CI_NODE and CI750 Ptable descriptors
0000  ;
0000  ;        21      Dave Butenhof, 16-Apr-1982, verison 6.7
0000  ;                Add RA81 Ptable descriptor
0000  ;
0000  ;        22      Dave Butenhof, 29-Apr-1982, version 6.7
0000  ;                Add RA60 Ptable descriptor
0000  ;
0000  ;        23      Dave Butenhof, 05-May-1982, version 6.8
0000  ;                Activate the device name check feature which
0000  ;                was added (latent) in 6.7 macros.
0000  ;
0000  ;        24      Dave Butenhof, 11-May-1982, version 6.8
0000  ;                Add Lp07 and Lp26 Ptables for Dan Milleville
0000  ;
0000  ;        25      Dave Butenhof, 25-May-1982, version 6.8
0000  ;                Add "console" Ptable desc. to IOBASE.  This allows
0000  ;                new console Ptable to be displayed by SHOW
0000  ;                (DEVICE | SELECT) without the annoying "..." which
0000  ;                means it couldn't find the Ptable.  This device is
0000  ;                attached at DS startup by KERNEL, and as it has
0000  ;                a lower-case device type, CANNOT be attached manually!
0000  ;
```

```
                                0000   172 ;       26      M. Baggett, 20-Jul-1982 Ver 6.9
                                0000   173 ;               Added LP27 line printer.
                                0000   174 ;
                                0000   175 ;       27      Bob Bergazzi    1-Oct-82        Version 6.9
                                0000   176 ;               Added UNA11 (UNIBUS to NI Adapter)
                                0000   177 ;
                                0000   178 ;       28      M. Baggett      1-Dec-1982      Version 6.10
                                0000   179 ;               Added TU80 magtape and DZ32.
                                0000   180 ;
                                0000   181 ;       29      M. Baggett      8-Dec-1982      Version 6.10
                                0000   182 ;               Added TU81 magtape.
                                0000   183 ;
                                0000   184 ;       30      M. Baggett      10-Dec-1982     Version 6.10
                                0000   185 ;               Added VS100.
                                0000   186 ;
                                0000   187 ;       31      Bob Bergazzi    10-Jan-83       Version 6.11
                                0000   188 ;               Changed the comment before DS$GA_PTABLE.
                                0000   189 ;
                                0000   190 ;       32      M. Baggett      14-Jan-83       Version 6.11
                                0000   191 ;               Added RC25.
                                0000   192 ;
                                0000   193 ;       33      M. Baggett      15-Feb-83       Version 6.11
                                0000   194 ;               Added VS300 and RCF25.
                                0000   195 ;
                                0000   196 ;       34      M. Baggett      14-Mar-1983     Version 6.11
                                0000   197 ;               Added p-table for LESI controller.
                                0000   198 ;
                                0000   199 ;       35      M. Baggett      21-March-1983   Version 6.11
                                0000   200 ;               Added p-tables for LA12,LA100,LN01,VT220,VT240.
                                0000   201 ;
                                0000   202 ;       36      Bob Bergazzi    25-April-1983   Version 6.11
                                0000   203 ;               Added Ptable for KA785.
                                0000   204 ;
                                0000   205 ;       37      Richard Brown   2-May-1983      Version 6.11
                                0000   206 ;               Added SBIA and KAxxx.
                                0000   207 ;
                                0000   208 ;       38      Richard Brown   15-July-1983    Version 6-12
                                0000   209 ;               Added RC11.
                                0000   210 ;
                                0000   211 ;       39      M. Baggett      8-Aug-1983      Version 6.12
                                0000   212 ;               Added IEU11A
                                0000   213 ;
                                0000   214 ;       40      M. Baggett      7-Sep-1983      Version 6.13
                                0000   215 ;               Added VS125.
                                0000   216 ;
                                0000   217 ;       41      M. Baggett      21-Sep-1983     Version 6.13
                                0000   218 ;               Added DISK p-table.
                                0000   219 ;
                                0000   220 ;       42      M. Baggett      11-Oct-1983     Version 6.13
                                0000   221 ;               Added TAPE p-table.
                                0000   222 ;
                                0000   223 ;       43      M. Baggett      13-Oct-1983     Version 6.13
                                0000   224 ;               Added TS05 p-table.
                                0000   225 ;
                                0000   226 ;       44      Bob Bergazzi    Jan 23, 1984    Version 6.14
                                0000   227 ;               Added KAZZZ Ptable.
                                0000   228 ;
```

```
                        0000   229 ;       45      M. Baggett      Feb 6, 1984     Version 6.14
                        0000   230 ;               Added DMZ32 p-table.
                        0000   231 ;
                        0000   232 ;       46      Richard Brown   July 7, 1984    Version 7.0
                        0000   233 ;               Removed RC11, since it is really LESI.
                        0000   234 ;
                        0000   235 ;       47      Bob Bergazzi    Aug. 6, 1984    Version 7.1
                        0000   236 ;               Added DW000 ptable for BUA.
                        0000   237 ;
                        0000   238 ;       48      Bob Bergazzi    Sep. 4, 1984    Version 7.1
                        0000   239 ;               Changed BUA device name from DW000 to DWBUA.
                        0000   240 ;
                        0000   241 ;       49      Amy Iorio       Oct. 1, 1984    Version 7.1
                        0000   242 ;               Change KAXXX to KA86.
                        0000   243 ;
                        0000   244 ;       50      Amy Iorio       Oct. 11, 1984   Version 7.1
                        0000   245 ;               Added KAXXX.
                        0000   246 ;
                        0000   247 ;       51      M. Baggett      2-NOV-1984      Version 7.1
                        0000   248 ;               Added TK50.
                        0000   249 ;
                        0000   250 ;       52      Bob Bergazzi    Nov. 12, 1984   Version 7.1
                        0000   251 ;               Added BCI750.
                        0000   252 ;
                        0000   253 ;       53      M. Baggett      15-Nov-1984     Version 7.1
                        0000   254 ;               Added RX31,RX32,RX50,RX180,RUX50.
                        0000   255 ;
                        0000   256 ;       54      Bob Bergazzi    Nov. 29,1984    Version 8.0
                        0000   257 ;               Added KA810. Removed KAXXX.
                        0000   258 ;
                        0000   259 ;       55      M. Bagggett     18-Dec-1984     Version 8.0
                        0000   260 ;               Added VT101, VT102, VT125.
                        0000   261 ;
                        0000   262 ;       56      A. Iorio        18-Jan-1985     Version 8.1
                        0000   263 ;               Added DHU11, DR11C
                        0000   264 ;
                        0000   265 ;       57      A. Iorio        5-Mar-1985      Version 8.1
                        0000   266 ;               Added DX20, TU70, TU72
                        0000   267 ;
                        0000   268 ;       58      A. Iorio        12-Mar-1985     Version 8.1
                        0000   269 ;               Added DHV11
                        0000   270 ;
                        0000   271 ;       59      Bob Bergazzi    24-Apr-1985     Version 8.1
                        0000   272 ;               Added KA820, remove KA810 later.
                        0000   273 ;
                        0000   274 ;       60      Bob Bergazzi    16-May-1985     Version 8.2
                        0000   275 ;               Added BLA ptable.
                        0000   276 ;
                        0000   277 ;       61      Bob Bergazzi    23-May-1985     Version 8.2
                        0000   278 ;               Added KDB50 ptable (BDA).
                        0000   279 ;
                        0000   280 ;       62      Bob Bergazzi    30-May-1985     Version 8.3
                        0000   281 ;               Added SLU, NI ptables.
                        0000   282 ;
                        0000   283 ;       63      Bob Bergazzi    11-June-1985    Version 8.3
                        0000   284 ;               Changed BCI750 to CIBCI.
                        0000   285 ;
```

```
                        0000   286 ;       64      Bob Bergazzi    18-June-1985    Version 8.3
                        0000   287 ;               Removed KAZZZ ptable.
                        0000   288 ;
                        0000   289 ;       65      M. Bag          7-Aug-1985      Version 8.3
                        0000   290 ;               Added RDBX1 p-table.
                        0000   291 ;
                        0000   292 ;       66      Amy Iorio       12-Aug-1985     Version 8.3
                        0000   293 ;               Added LN03, LA210, VT61, VT71, VT131, and VT132 ptables.
                        0000   294 ;
                        0000   295 ;       67      M. Baggett      19-Aug-1985     Version 9.1
                        0000   296 ;               Added RD52,RD53.
                        0000   297 ;
                        0000   298 ;       68      M. Baggett      11-Sep-1985     VDS Ver 9.0
                        0000   299 ;               Added p-tables for DEBNT(AIE),KA800,KFBTA(AIO),LANCE
                        0000   300 ;
                        0000   301 ;       69      Bob Bergazzi    20-Sep-1985     Version 9.0
                        0000   302 ;               Removed KA810 ptable since KA820 supersedes it.
                        0000   303 ;
                        0000   304 ;       70      Jim Baranski    09-Oct-1985     Version 9.0
                        0000   305 ;               Added RA82 disk Ptable descriptor.
                        0000   306 ;
                        0000   307 ;       71      Jim Baranski    10-Oct-1985     Version 9.0
                        0000   308 ;               Added LUA11 Ptable descriptor.
                        0000   309 ;
                        0000   310 ;       72      M. Baggett      22-Oct-1985     Version 9.0
                        0000   311 ;               Removed RDBX1 p-table.
                        0000   312 ;                                                                       ;G:4
                        0000   313 ;       73      J. Baranski     11-Mar-1986     Version 10.0            ;G:4
                        0000   314 ;               Added RRD50 PTable descriptor.                          ;G:4
                        0000   315 ;                                                                       ;G:12
                        0000   316 ;       74      M. Baggett      5-May-1986      Version 10.0            ;G:12
                        0000   317 ;               Added RD54.                                             ;G:12
                        0000   318 ;                                                                       ;G:13
                        0000   319 ;       75      Richard E. Muse   19-may-1986   version 10.1            ;G:13
                        0000   320 ;               added CIBCA                                             ;G:13
                        0000   321 ;                                                                       ;G:14
                        0000   322 ;       76      M. Baggett      10-Jun-1986     Version 10.1            ;G:17
                        0000   323 ;               Added DRB32.                                            ;G:14
                        0000   324 ;                                                                       ;G:17
                        0000   325 ;       77      Bob Bergazzi    2-Sep-1986      Version 10.3            ;G:17
                        0000   326 ;               Arranged the table of ptable descriptors so that there is one ;G:17
                        0000   327 ;               device listed per line of code.  Added the symbol       ;G:17
                        0000   328 ;               PTABLE_DESCRIPTORS to identify the beginning of this list.  ;G:17
                        0000   329 ;               This was done so that the release group can more easily parse ;G:17
                        0000   330 ;               the source to obtain a list of supported devices.       ;G:17
                        0000   331 ;                                                                       ;G:17
                        0000   332 ;               ********************************************************** ;G:17
                        0000   333 ;                                       IMPORTANT!!                     ;G:17
                        0000   334 ;               ********************************************************** ;G:17
                        0000   335 ;               Please follow this convention when adding new devices   ;G:17
                        0000   336 ;               ********************************************************** ;G:17
                        0000   337 ;                                                                       ;G:17
                        0000   338 ;               Also removed $DS_RRD50_DEF call because it was moved into the ;G:17
                        0000   339 ;               $DS_RRD50 ptable descriptor in DIAG.MAR.                ;G:17
                        0000   340 ;                                                                       ;G:19
                        0000   341 ;       78      Stephen Meier   30-Oct-1986     Version 10.4            ;G:40
                        0000   342 ;               Added XBI .                                             ;G:19
```

```
0000   343 ;                                                                                  ;G:27
0000   344 ;        79      David Gearin    30-Jan-1987     Version 10.5                      ;G:28
0000   345 ;                Added p-table for LG01, LG02, LG03.                               ;G:27
0000   346 ;                                                                                  ;G:27
0000   347 ;        80      Ingrid Martin   4-Feb-1987      Version 10.5                      ;G:28
0000   348 ;                Added KA888.                                                      ;G:43A
0000   349 ;                                                                                  ;G:33
0000   350 ;        81      David Gearin    11-Feb-1987     Version 10.5                      ;G:33
0000   351 ;                Added P-table data for drb32.                                     ;G:33
0000   352 ;                                                                                  ;G:33
0000   353 ;        82      Bob Bergazzi    4-Mar-1987      Version 10.7                      ;G:33
0000   354 ;                Added RA70.                                                       ;G:33
0000   355 ;        83      Stephen Meier   18-Mar-1987     Version 10.7                      ;G:36
0000   356 ;                Added KA9CC     (LLL processor)                                   ;G:38
0000   357 ;                                                                                  ;G:40
0000   358 ;        84      M. Baggett      10-Apr-1987     Version 10.7                      ;G:39
0000   359 ;                Added p-table for TK70,DHB32.                                     ;G:39
0000   360 ;                                                                                  ;G:41
0000   361 ;                Bob Bergazzi    22-may-1987     Version 10.8                      ;G:41
0000   362 ;                Added DEBNA, DEBNK ptables.                                       ;G:41
0000   363 ;                                                                                  ; ;G:4
0000   364 ;                David Gearin    01-July-1987    Version 10.9                      ; ;G:4
0000   365 ;                Added RV20 ptable                                                 ;G:42
0000   366 ;                                                                                  ;G:43
0000   367 ;                Bob Bergazzi    12-aug-1987     Version 10.9                      ;G:45
0000   368 ;                Changed .LIB back to $DIAG.                                       ;G:45
0000   369 ;                                                                                  ;G:46
0000   370 ;        46      Stephen Meier   29-Sep-1987     Version 10.10                     ;G:46
0000   371 ;                Version 70 of Diag... support for > 4 BI's.                       ;G:46
0000   372 ;                                                                                  ;G:47
0000   373 ;                Ingrid Martin   6-oct-87        version 10.10                     ;G:47
0000   374 ;                Permently changed KA888 to KA884.                                 ;G:47
0000   375 ;                Added DS_PBIA to device list.                                     ;G:47
0000   376 ;                Removed call to $DS_DEVTYP for DS$GA_PTDESC because there         ;G:47
0000   377 ;                are now too many devices.                                         ;G:47
0000   378 ;                                                                                  ;G:54
0000   379 ;                David Gearin    20-Oct-1987     Version 10.10                     ;G:54
0000   380 ;                Added support for RA90 ptables.                                   ;G:54
0000   381 ;--
```

```
                     0000   383          .SBTTL  DECLARATIONS
                     0000   384 ;
                     0000   385 ; MACRO LIBRARY CALLS
                     0000   386 ;
                     0000   387                                                          ;G:40
                     0000   388 .library        /$diag/                                  ;G:52
                     0000   389
                     0000   390          $DS_HPODEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
           00000008  0000                .IIF    NB,.BLKQ,       .BLKQ   1
                     0008                .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
           0000000A  0008                .IIF    NB,.BLKW,       .BLKW   1
                     000A                .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
           0000000B  000A                .IIF    NB,.BLKB,       .BLKB   1
                     000B                .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
           0000000C  000B                .IIF    NB,.BLKB,       .BLKB   1
                     000C                .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
           00000018  000C                .IIF    NB,.BLKB,       .BLKB   12
                     0018                .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
           0000001C  0018                .IIF    NB,.BLKL,       .BLKL   1
                     001C                .IIF    NB,HP$A_DVA,    HP$A_DVA:
           00000020  001C                .IIF    NB,.BLKL,       .BLKL   1
                     0020                .IIF    NB,HP$A_LINK,   HP$A_LINK:
           00000024  0020                .IIF    NB,.BLKL,       .BLKL   1
                     0024                .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
           00000026  0024                .IIF    NB,.BLKW,       .BLKW   1
                     0026                .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
           00000032  0026                .IIF    NB,.BLKB,       .BLKB   12
                     0032                .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                     0000                .RESTORE
                     0000   391
```

```
                                                                                                                    ;G:40
                                 0000   393
                                 0000   394 ;
                                 0000   395 ; GLOBAL DATA
                                 0000   396 ;
                                 0000   397 ; SYSTEM BOOT UCB TABLES
                                 0000   398 ;
                                 0000   399
                             00000000   400          .PSECT  SEP,    SHR,    EXE,    WRT,    LONG
                                 0000   401 IOC$AL_SYSUCB::                          ; SYSTEM DEVICE UCB TABLE
                         00000000 0000   402          .LONG   0                      ; RP06 UCB TABLE
                         00000000 0004   403          .LONG   0
                                 0008   404
                                 0008   405 IOC$GL_DEVLIST::
                         00000000 0008   406          .LONG   0                      ; START OF DEVICE LIST
                                 000C   407 IOC$GL_ADPLIST::
                         00000000 000C   408          .LONG   0                      ; START OF ADAPTER CONTROL BLOCK LIST
                                 0010   409 IOC$GL_DPTLIST::                         ; QUEUE OF DRIVER PROLOGUE BLOCKS
                 00000010'00000010' 0010   410          .LONG   IOC$GL_DPTLIST,IOC$GL_DPTLIST
```

```
                            00000000   412          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                0000   413 ;+
                                0000   414 ; The following table contains a select bit corresonding to the
                                0000   415 ; entries in DS$GA_PTABLE.
                                0000   416 ;-
                            00000080    0000   417 MAXPT=128
                                0000   418 DS$GA_SELECTS::
                            00000014    0000   419          .BLKL   MAXPT/32+1
                                0014   420
                                0014   421 ;+
                                0014   422 ; The following table contains an allocate bit corresponding to the
                                0014   423 ; to entries in DS$GA_TABLE
                                0014   424 ;-
                                0014   425 DS$GA_ALLOCATES::                                          ;                [48]
                            00000028    0014   426          .BLKL   MAXPT/32+1                       ;                [48]
                                0028   427
                                0028   428 ;+
                                0028   429 ; This table contains the addresses of each P-table actually to be tested.
                                0028   430 ; It is built at program start time by matching the device types in the
                                0028   431 ; program DEVTYP list with the type field in each P-table.
                                0028   432 ; DSP$GEN_PTABLES in DEVICE actually performs this.
                                0028   433 ;-
                                0028   434 DS$GA_SELECTED::
                            00000080    0028   435          .LONG   MAXPT
00000000'00000000'00000000'00000000'    002C   436          .LONG   0[MAXPT]
00000000'00000000'00000000'00000000'    003C
00000000'00000000'00000000'00000000'    004C
00000000'00000000'00000000'00000000'    005C
00000000'00000000'00000000'00000000'    006C
00000000'00000000'00000000'00000000'    007C
00000000'00000000'00000000'00000000'    008C
00000000'00000000'00000000'00000000'    009C
00000000'00000000'00000000'00000000'    00AC
00000000'00000000'00000000'00000000'    00BC
00000000'00000000'00000000'00000000'    00CC
00000000'00000000'00000000'00000000'    00DC
00000000'00000000'00000000'00000000'    00EC
00000000'00000000'00000000'00000000'    00FC
00000000'00000000'00000000'00000000'    010C
00000000'00000000'00000000'00000000'    011C
00000000'00000000'00000000'00000000'    012C
00000000'00000000'00000000'00000000'    013C
00000000'00000000'00000000'00000000'    014C
00000000'00000000'00000000'00000000'    015C
00000000'00000000'00000000'00000000'    016C
00000000'00000000'00000000'00000000'    017C
00000000'00000000'00000000'00000000'    018C
00000000'00000000'00000000'00000000'    019C
00000000'00000000'00000000'00000000'    01AC
00000000'00000000'00000000'00000000'    01BC
00000000'00000000'00000000'00000000'    01CC
00000000'00000000'00000000'00000000'    01DC
00000000'00000000'00000000'00000000'    01EC
00000000'00000000'00000000'00000000'    01FC
00000000'00000000'00000000'00000000'    020C
00000000'00000000'00000000'00000000'    021C
                                022C   437 DS$GA_ALLOCATED::
```

```
                          00000080  022C   438          .LONG   MAXPT
00000000'00000000'00000000'00000000' 0230   439          .LONG   0[MAXPT]
00000000'00000000'00000000'00000000' 0240
00000000'00000000'00000000'00000000' 0250
00000000'00000000'00000000'00000000' 0260
00000000'00000000'00000000'00000000' 0270
00000000'00000000'00000000'00000000' 0280
00000000'00000000'00000000'00000000' 0290
00000000'00000000'00000000'00000000' 02A0
00000000'00000000'00000000'00000000' 02B0
00000000'00000000'00000000'00000000' 02C0
00000000'00000000'00000000'00000000' 02D0
00000000'00000000'00000000'00000000' 02E0
00000000'00000000'00000000'00000000' 02F0
00000000'00000000'00000000'00000000' 0300
00000000'00000000'00000000'00000000' 0310
00000000'00000000'00000000'00000000' 0320
00000000'00000000'00000000'00000000' 0330
00000000'00000000'00000000'00000000' 0340
00000000'00000000'00000000'00000000' 0350
00000000'00000000'00000000'00000000' 0360
00000000'00000000'00000000'00000000' 0370
00000000'00000000'00000000'00000000' 0380
00000000'00000000'00000000'00000000' 0390
00000000'00000000'00000000'00000000' 03A0
00000000'00000000'00000000'00000000' 03B0
00000000'00000000'00000000'00000000' 03C0
00000000'00000000'00000000'00000000' 03D0
00000000'00000000'00000000'00000000' 03E0
00000000'00000000'00000000'00000000' 03F0
00000000'00000000'00000000'00000000' 0400
00000000'00000000'00000000'00000000' 0410
00000000'00000000'00000000'00000000' 0420
                                     0430   440 ;+
                                     0430   441 ; This table contains the address of each known device.  It is ordered.
                                     0430   442 ; The most recently attached device has the lowest index in the table because ;G:17
                                     0430   443 ; entries are allocated from the bottom up. In addition, as a byproduct of    ;G:17
                                     0430   444 ; the SELECT command, the specified devices are moved to the end of the list. ;G:17
                                     0430   445 ; This results in the most recently selected device is the last (in a series)
                                     0430   446 ; to be tested.
                                     0430   447 ;-
                                     0430   448
                                     0430   449 DS$GA_PTABLE::
                          00000080   0430   450          .LONG   MAXPT
00000000'00000000'00000000'00000000' 0434   451          .LONG   0[MAXPT]
00000000'00000000'00000000'00000000' 0444
00000000'00000000'00000000'00000000' 0454
00000000'00000000'00000000'00000000' 0464
00000000'00000000'00000000'00000000' 0474
00000000'00000000'00000000'00000000' 0484
00000000'00000000'00000000'00000000' 0494
00000000'00000000'00000000'00000000' 04A4
00000000'00000000'00000000'00000000' 04B4
00000000'00000000'00000000'00000000' 04C4
00000000'00000000'00000000'00000000' 04D4
00000000'00000000'00000000'00000000' 04E4
00000000'00000000'00000000'00000000' 04F4
```

```
00000000'00000000'00000000'00000000' 0504
00000000'00000000'00000000'00000000' 0514
00000000'00000000'00000000'00000000' 0524
00000000'00000000'00000000'00000000' 0534
00000000'00000000'00000000'00000000' 0544
00000000'00000000'00000000'00000000' 0554
00000000'00000000'00000000'00000000' 0564
00000000'00000000'00000000'00000000' 0574
00000000'00000000'00000000'00000000' 0584
00000000'00000000'00000000'00000000' 0594
00000000'00000000'00000000'00000000' 05A4
00000000'00000000'00000000'00000000' 05B4
00000000'00000000'00000000'00000000' 05C4
00000000'00000000'00000000'00000000' 05D4
00000000'00000000'00000000'00000000' 05E4
00000000'00000000'00000000'00000000' 05F4
00000000'00000000'00000000'00000000' 0604
00000000'00000000'00000000'00000000' 0614
00000000'00000000'00000000'00000000' 0624
```

```
                 0634          .SBTTL  Device description database
              00000000          .PSECT  DATA, SHR, NOWRT, NOEXE, BYTE
                 0000   ;
                 0000   ;       The following list contains the descriptor for each type
                 0000   ;       of device the supervisor knows about.                          ;G:47
                 0000   ;
                 0000
                 0000   DS$GA_PTDESC::                  ; Note:  always update the first longword; this ;G:4
                 0000                                   ; is the number of devices in the list.       ;G:49
        000000AC 0000          .LONG   172                                                          ;G:54
        000002B4' 0004         .LONG   AA11K                                                        ;G:47
        00000305' 0008         .LONG   AD11K                                                        ;G:47
        000003F2' 000C         .LONG   CIBCA                                                        ;G:47
        00000356' 0010         .LONG   CIBCI                                                        ;G:47
        0000050D' 0014         .LONG   CI780                                                        ;G:47
        0000048E' 0018         .LONG   CI750                                                        ;G:47
        00000594' 001C         .LONG   CI_NODE                                                      ;G:47
        000006B0' 0020         .LONG   Console                                                      ;G:47
        000006BE' 0024         .LONG   CR11                                                         ;G:47
        00000715' 0028         .LONG   DEBNA                                                        ;G:47
        00000780' 002C         .LONG   DEBNK                                                        ;G:47
        000007EB' 0030         .LONG   DEBNT                                                        ;G:47
        00000856' 0034         .LONG   DHB32                                                        ;G:47
        000008BD' 0038         .LONG   DHU11                                                        ;G:47
        0000090B' 003C         .LONG   DHV11                                                        ;G:47
        00000959' 0040         .LONG   DISK                                                         ;G:47
        0000096C' 0044         .LONG   DL11                                                         ;G:47
        000009B9' 0048         .LONG   DMB32                                                        ;G:47
        00000A20' 004C         .LONG   DMC11                                                        ;G:47
        00000A6E' 0050         .LONG   DMF32                                                        ;G:47
        00000AC1' 0054         .LONG   DMF32A                                                       ;G:47
        00000BE3' 0058         .LONG   DMF32P                                                       ;G:47
        00000C5D' 005C         .LONG   DMF32S                                                       ;G:47
        00000CD1' 0060         .LONG   DMP11                                                        ;G:47
        00000D9A' 0064         .LONG   DMR11                                                        ;G:47
        00000DE8' 0068         .LONG   DMZ32                                                        ;G:47
        00000F46' 006C         .LONG   DRB32                                                        ;G:47
        00000FAE' 0070         .LONG   DR11B                                                        ;G:47
        00000FFC' 0074         .LONG   DR11C                                                        ;G:47
        00001045' 0078         .LONG   DR11K                                                        ;G:47
        00001096' 007C         .LONG   DR11W                                                        ;G:47
        000010E4' 0080         .LONG   DR750                                                        ;G:47
        0000116F' 0084         .LONG   DR780                                                        ;G:47
        00001202' 0088         .LONG   DUP11                                                        ;G:47
        00001250' 008C         .LONG   DW730                                                        ;G:47
        0000127F' 0090         .LONG   DW750                                                        ;G:47
        000012C3' 0094         .LONG   DW780                                                        ;G:47
        00001337' 0098         .LONG   DWBLA                                                        ;G:47
        000013A7' 009C         .LONG   DWBUA                                                        ;G:47
        0000141E' 00A0         .LONG   DX20                                                         ;G:47
        0000144C' 00A4         .LONG   DZ11                                                         ;G:47
        000014B5' 00A8         .LONG   DZ32                                                         ;G:47
        000015A6' 00AC         .LONG   IEU11A                                                       ;G:47
        000015F1' 00B0         .LONG   KA730                                                        ;G:47
        000016CC' 00B4         .LONG   KA750                                                        ;G:47
        00001798' 00B8         .LONG   KA780                                                        ;G:47
        00001841' 00BC         .LONG   KA785                                                        ;G:47
```

```
000018EA' 00C0   510         .LONG   KA800                                   ;G:47
0000195C' 00C4   511         .LONG   KA820                                   ;G:47
000019E7' 00C8   512         .LONG   KA86                                    ;G:47
00001A8F' 00CC   513         .LONG   KA850                                   ;G:47
00001AC8' 00D0   514         .LONG   KA880                                   ;G:47
00001B05' 00D4   515         .LONG   KA884                                   ;G:47
00001B5A' 00D8   516         .LONG   KA9CC                                   ;G:47
00001C5C' 00DC   517         .LONG   KFBTA                                   ;G:47
00001BC9' 00E0   518         .LONG   KDB50                                   ;G:47
00001CC2' 00E4   519         .LONG   KMC11                                   ;G:47
00001D10' 00E8   520         .LONG   KW11K                                   ;G:47
00001D61' 00EC   521         .LONG   LA12                                    ;G:47
00001D7B' 00F0   522         .LONG   LA34                                    ;G:47
00001D95' 00F4   523         .LONG   LA36                                    ;G:47
00001DAF' 00F8   524         .LONG   LA38                                    ;G:47
00001DC9' 00FC   525         .LONG   LA100                                   ;G:47
00001DE4' 0100   526         .LONG   LA120                                   ;G:47
00001DFF' 0104   527         .LONG   LA180                                   ;G:47
00001E1A' 0108   528         .LONG   LA210                                   ;G:47
00001E35' 010C   529         .LONG   LANCE                                   ;G:47
00001E46' 0110   530         .LONG   LESI                                    ;G:47
00001E92' 0114   531         .LONG   LG01                                    ;G:47
00001EAC' 0118   532         .LONG   LG02                                    ;G:47
00001EC6' 011C   533         .LONG   LG03                                    ;G:47
00001EE0' 0120   534         .LONG   LN01                                    ;G:47
00001EFA' 0124   535         .LONG   LN03                                    ;G:47
00001F0F' 0128   536         .LONG   LP04                                    ;G:47
00001F29' 012C   537         .LONG   LP05                                    ;G:47
00001F43' 0130   538         .LONG   LP06                                    ;G:47
00001F5D' 0134   539         .LONG   LP07                                    ;G:47
00001F77' 0138   540         .LONG   LP11                                    ;G:47
00001FC4' 013C   541         .LONG   LP14                                    ;G:47
00001FDE' 0140   542         .LONG   LP25                                    ;G:47
00001FF8' 0144   543         .LONG   LP26                                    ;G:47
00002012' 0148   544         .LONG   LP27                                    ;G:47
0000202C' 014C   545         .LONG   LPA11K                                  ;G:47
00002085' 0150   546         .LONG   LUA11                                   ;G:47
000020D3' 0154   547         .LONG   MA780                                   ;G:47
0000214E' 0158   548         .LONG   MBE                                     ;G:47
00002162' 015C   549         .LONG   ML11                                    ;G:47
000021C4' 0160   550         .LONG   MS750                                   ;G:47
000021D5' 0164   551         .LONG   MS780                                   ;G:47
00002225' 0168   552         .LONG   MSAAA                                   ;G:47
00002236' 016C   553         .LONG   NBIA                                    ;G:47
000022A6' 0170   554         .LONG   NBIB                                    ;G:47
00002309' 0174   555         .LONG   NI                                      ;G:47
0000233F' 0178   556         .LONG   PBIA                                    ;G:47
000023BF' 017C   557         .LONG   PCL11                                   ;G:47
00002408' 0180   558         .LONG   PX780                                   ;G:47
00002471' 0184   559         .LONG   R80                                     ;G:47
0000248A' 0188   560         .LONG   RA60                                    ;G:47
000024A4' 018C   561         .LONG   RA70                                    ;G:47
000024BE' 0190   562         .LONG   RA80                                    ;G:47
000024D8' 0194   563         .LONG   RA81                                    ;G:47
0000250C' 0198   564         .LONG   RA90                                    ;G:54
000024F2' 019C   565         .LONG   RA82                                    ;G:47
00002529' 01A0   566         .LONG   RB730                                   ;G:47
```

```
00002558' 01A4   567         .LONG   RCF25                                   ;G:47
00002573' 01A8   568         .LONG   RC25                                    ;G:47
0000258D' 01AC   569         .LONG   RD52                                    ;G:47
000025A7' 01B0   570         .LONG   RD53                                    ;G:47
000025C1' 01B4   571         .LONG   RD54                                    ;G:47
000025DB' 01B8   572         .LONG   RH750                                   ;G:47
00002634' 01BC   573         .LONG   RH780                                   ;G:47
00002699' 01C0   574         .LONG   RL01                                    ;G:47
000026B3' 01C4   575         .LONG   RL02                                    ;G:47
000026CD' 01C8   576         .LONG   RL11                                    ;G:47
0000271A' 01CC   577         .LONG   RK06                                    ;G:47
00002734' 01D0   578         .LONG   RK07                                    ;G:47
0000274E' 01D4   579         .LONG   RK611                                   ;G:47
0000279C' 01D8   580         .LONG   RM03                                    ;G:47
000027C0' 01DC   581         .LONG   RM05                                    ;G:47
000027E4' 01E0   582         .LONG   RM80                                    ;G:47
00002898' 01E4   583         .LONG   RRD50                                   ;G:47
00002808' 01E8   584         .LONG   RP04                                    ;G:47
0000282C' 01EC   585         .LONG   RP05                                    ;G:47
00002850' 01F0   586         .LONG   RP06                                    ;G:47
00002874' 01F4   587         .LONG   RP07                                    ;G:47
000028F4' 01F8   588         .LONG   RV20                                    ;G:47
0000294F' 01FC   589         .LONG   RUX50                                   ;G:47
00002997' 0200   590         .LONG   RX02                                    ;G:47
000029B1' 0204   591         .LONG   RX31                                    ;G:47
000029C1' 0208   592         .LONG   RX32                                    ;G:47
000029D1' 020C   593         .LONG   RX50                                    ;G:47
000029E6' 0210   594         .LONG   RX180                                   ;G:47
000029F7' 0214   595         .LONG   RX211                                   ;G:47
00002A45' 0218   596         .LONG   SBIA                                    ;G:47
00002A87' 021C   597         .LONG   SLU                                     ;G:47
00002AFC' 0220   598         .LONG   TAPE                                    ;G:47
00002B0C' 0224   599         .LONG   TE16                                    ;G:47
00002B26' 0228   600         .LONG   TK50                                    ;G:47
00002B40' 022C   601         .LONG   TK70                                    ;G:47
00002B5A' 0230   602         .LONG   TM03                                    ;G:47
00002B88' 0234   603         .LONG   TM78                                    ;G:47
00002BB6' 0238   604         .LONG   TS04                                    ;G:47
00002C0D' 023C   605         .LONG   TS05                                    ;G:47
00002C77' 0240   606         .LONG   TS11                                    ;G:47
00002CCE' 0244   607         .LONG   TU45                                    ;G:47
00002CE8' 0248   608         .LONG   TU58                                    ;G:47
00002D02' 024C   609         .LONG   TU70                                    ;G:47
00002D1C' 0250   610         .LONG   TU72                                    ;G:47
00002D36' 0254   611         .LONG   TU77                                    ;G:47
00002D50' 0258   612         .LONG   TU78                                    ;G:47
00002D6A' 025C   613         .LONG   TU80                                    ;G:47
00002DD9' 0260   614         .LONG   TU81                                    ;G:47
00002E30' 0264   615         .LONG   UBE                                     ;G:47
00002E6B' 0268   616         .LONG   UDA50                                   ;G:47
00002ED4' 026C   617         .LONG   UNA11                                   ;G:47
00002F22' 0270   618         .LONG   VS100                                   ;G:47
00002F7A' 0274   619         .LONG   VS125                                   ;G:47
00002FD2' 0278   620         .LONG   VS300                                   ;G:47
0000302A' 027C   621         .LONG   VT50                                    ;G:47
00003044' 0280   622         .LONG   VT52                                    ;G:47
0000305E' 0284   623         .LONG   VT55                                    ;G:47
```

```
00003078' 0288   624        .LONG   VT61                                    ;G:47
00003092' 028C   625        .LONG   VT71                                    ;G:47
000030AC' 0290   626        .LONG   VT100                                   ;G:47
000030C7' 0294   627        .LONG   VT101                                   ;G:47
000030E2' 0298   628        .LONG   VT102                                   ;G:47
000030FD' 029C   629        .LONG   VT125                                   ;G:47
00003118' 02A0   630        .LONG   VT131                                   ;G:47
00003133' 02A4   631        .LONG   VT132                                   ;G:47
0000314E' 02A8   632        .LONG   VT220                                   ;G:47
00003169' 02AC   633        .LONG   VT240                                   ;G:47
00003184' 02B0   634        .LONG   XBI                                     ;G:47
          02B4   635
```

```
                    000002B4   637          .PSECT  DATA, SHR, NOWRT, NOEXE, BYTE
                         02B4   638          .NLIST  ME,MEB
                         02B4   639          .LIST   MC
                         02B4   640 ;                                                                          ;G:17
                         02B4   641 ; Device dependent field definitions which apply to more than one      [77]  ;G:17
                         02B4   642 ; ptable, and thus are not called from the devices' ptable descriptor  [77]  ;G:17
                         02B4   643 ; because multiply defined symbols would result.                       [77]  ;G:17
                         02B4   644 ;                                                                          ;G:17
                         02B4   645          $DS_CI_DEF                                  ;                 [20]  ;G:17
                         02B4               .SAVE   LOCAL_BLOCK
                         02B4               .PSECT  $ABS$,ABS
                         0032               .IIF    NB,HP$B_CI_TR,  HP$B_CI_TR:
                         0032               .IIF    NB,CI$B_BI_ID,  CI$B_BI_ID:
                         0032               .IIF    NB,CI$B_TR,     CI$B_TR:
                00000033 0032               .IIF    NB,,BLKB,       .BLKB   1
                         0033               .IIF    NB,HP$B_CI_BR,  HP$B_CI_BR:
                         0033               .IIF    NB,CI$B_BR,     CI$B_BR:
                00000034 0033               .IIF    NB,,BLKB,       .BLKB   1
                         0034               .IIF    NB,HP$B_CI_NODE,        HP$B_CI_NODE:
                         0034               .IIF    NB,CI$B_NODE,   CI$B_NODE:
                00000035 0034               .IIF    NB,,BLKB,       .BLKB   1
                         0035               .IIF    NB,HP$L_CI_FUNC,        HP$L_CI_FUNC:
                         0035               .IIF    NB,Ci$L_Func,   Ci$L_Func:
                00000039 0035               .IIF    NB,,BlkL,       .BlkL   1
                         0039               .IIF    NB,HP$L_CI_MAINT_ID,    HP$L_CI_MAINT_ID:
                         0039               .IIF    NB,Ci$L_Maint_Id,       Ci$L_Maint_Id:
                0000003D 0039               .IIF    NB,,BlkL,       .BlkL   1
                         003D               .IIF    NB,HP$L_CI_INDEX,       HP$L_CI_INDEX:
                         003D               .IIF    NB,Ci$L_Index,  Ci$L_Index:
                00000041 003D               .IIF    NB,,BlkL,       .BlkL   1
                         0041               .IIF    NB,HP$K_CI_LEN, HP$K_CI_LEN:
                         0041               .IIF    NB,CI$K_LEN,    CI$K_LEN:
                    000002B4               .RESTORE
                         02B4   646         $DS_DHUV_DEF
                         02B4               .SAVE   LOCAL_BLOCK
                         02B4               .PSECT  $ABS$,ABS
                00000032 0041               .=HP$A_DEPENDENT
                         0032               .IIF    NB,HP$L_DHUV_CSR,       HP$L_DHUV_CSR:
                00000036 0032               .IIF    NB,,BLKL,       .BLKL   1
                         0036               .IIF    NB,HP$W_DHUV_VEC,       HP$W_DHUV_VEC:
                00000038 0036               .IIF    NB,,BLKW,       .BLKW   1
                         0038               .IIF    NB,HP$B_DHUV_BR,        HP$B_DHUV_BR:
                00000039 0038               .IIF    NB,,BLKB,       .BLKB   1
                         0039               .IIF    NB,HP$K_DHUV_LEN,       HP$K_DHUV_LEN:
                    000002B4               .RESTORE
                         02B4   647         $DS_KA_DEF                                                        ;G:17
                         02B4               .SAVE   LOCAL_BLOCK
                         02B4               .PSECT  $ABS$,ABS
                00000032 0041               .=HP$A_Dependent
                         0032               .IIF    NB,KA$L_Flags,  KA$L_Flags::
                00000036 0032               .IIF    NB,,BlkI,       .BlkI   1
                         0036               .IIF    NB,KA$W_WCS,    KA$W_WCS::
                00000038 0036               .IIF    NB,,Blkw,       .Blkw   1
                         0038               .IIF    NB,KA$W_Reserved,       KA$W_Reserved::
                0000003A 0038               .IIF    NB,,Blkw,       .Blkw   1
                         003A               .IIF    NB,KA$L_SID,    KA$L_SID::
                0000003D 003A               .IIF    NB,,Blkb,       .Blkb   3
```

```
                    003D            .IIF    NB,KA$B_SID_Type,       KA$B_SID_Type::
          0000003E  003D            .IIF    NB,.Blkb,       .Blkb   1
                    003E            .IIF    NB,KA$W_Latency,        KA$W_Latency::
          00000040  003E            .IIF    NB,.Blkw,       .Blkw   1
                    0040            .IIF    NB,KA$W_Progress,       KA$W_Progress::
          00000042  0040            .IIF    NB,.Blkw,       .Blkw   1
                    0042            .IIF    NB,KA$B_Acc_Type,       KA$B_Acc_Type::
          00000043  0042            .IIF    NB,.Blkb,       .Blkb   1
                    0043            .IIF    NB,KA$B_Reserved,       KA$B_Reserved::
          00000044  0043            .IIF    NB,.Blkb,       .Blkb   1
                    0044            .IIF    NB,KA$W_Mem_Size,       KA$W_Mem_Size::
          00000046  0044            .IIF    NB,.Blkw,       .Blkw   1
                    0046            .IIF    NB,KA$B_Node_ID,        KA$B_Node_ID::
          00000047  0046            .IIF    NB,.Blkb,       .Blkb   1
                    0047            .IIF    NB,KA$L_Mem_Size,       KA$L_Mem_Size::
          0000004B  0047            .IIF    NB,.Blkl,       .Blkl   1
                    004B            .IIF    NB,HP$L_KAA_Slot,       HP$L_KAA_Slot::
          0000004F  004B            .IIF    NB,.Blkl,       .Blkl   1
                    004F            .IIF    NB,HP$B_Cpu_ID, HP$B_Cpu_ID::
          00000050  004F            .IIF    NB,.Blkb,       .Blkb   1
                    0050            .IIF    NB,KA$K_Len,    KA$K_Len::
                000002B4            .RESTORE
                    02B4  648                                                                       ;G:17
                    02B4  649 ;                                                                     ;G:17
                    02B4  650 ;     ************************************************************  [77]  ;G:17
                    02B4  651 ;                             IMPORTANT!!                           [77]  ;G:17
                    02B4  652 ;     ************************************************************  [77]  ;G:17
                    02B4  653 ;     When adding devices to this list, place each new              [77]  ;G:17
                    02B4  654 ;     device on one line with no spaces between lines.  There should [77]  ;G:17
                    02B4  655 ;     be no spaces between the colon and the symbol, and the symbol  [77]  ;G:17
                    02B4  656 ;     must begin in column 1. This is done so that the release group [77]  ;G:17
                    02B4  657 ;     can parse this source module at release time in order to obtain [77]  ;G:17
                    02B4  658 ;     a list of supported devices.  They assume that the list is    [77]  ;G:17
                    02B4  659 ;     delimited by the symbol PTABLE_DESCRIPTORS at the             [77]  ;G:17
                    02B4  660 ;     beginning, and .END at the end.                               [77]  ;G:17
                    02B4  661 ;     ************************************************************  [77]  ;G:17
                    02B4  662                                                                       ;G:17
                    02B4  663 PTABLE_DESCRIPTORS:                           ;                     [77]  ;G:17
                    02B4  664 AA11K:  $DS_AA11K                                                     ;G:40
                    02B4            .SAVE   LOCAL_BLOCK
                    02B4            .PSECT  $ABS$,ABS
          00000032  0050            .=HP$A_DEPENDENT
                    0032            .IIF    NB,AA11K$B_BR,  AA11K$B_BR:
          00000033  0032            .IIF    NB,.BLKB,       .BLKB   1
                    0033            .IIF    NB,AA11K$K_LEN, AA11K$K_LEN:
                000002B4            .RESTORE
    4B 31 31 41 41 00' 02B4         .ASCIC  "AA11K" ; AA11K name
                05  02B4
                33  02BA            .BYTE   AA11K$K_LEN             ; AA11K$K_LEN of resultant P-table
                04  02BB            .BYTE   4               ; Maximum unit number                   ;G:65
              0000  02BC            .WORD   ^A""    ; Two character mnemonic for QIO AA11K
                80  02BE            .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                8D  02BF            .Byte   Pd$_Name        ; Directive op-code
                03  02C0            .Byte   Ptd$M_Device            ; Enforced AA codes
          41 41 00' 02C1            .Ascic  "AA"    ; Enforced device name prefix
                02  02C1
                83  02C4            .BYTE   Pd$_Octal       ; Indicate scan octal
```

```
52 53 43 20 53 55 42 20 4F 2F 49 00' 02C5                .ASCIC  "I/O BUS CSR"    ; I/O BUS CSR string
                   0B  02C5
             0003FFFE 0003E000  02D1                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                   88  02D9                .BYTE   Pd$_Store        ; Indicate store operation
                 0018  02DA                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                   00  02DC                .BYTE   0                ; 0 HP$A_DEVICE to data
                   12  02DD                .BYTE   18               ; 18 of data field
                   83  02DE                .BYTE   Pd$_Octal        ; Indicate scan octal
          52 4F 54 43 45 56 00' 02DF                .ASCIC  "VECTOR"         ; VECTOR string
                   06  02DF
             000001FE 00000002  02E6                .LONG   ^O2,^O776        ; 2 , 776 limits on input
                   88  02EE                .BYTE   Pd$_Store        ; Indicate store operation
                 0024  02EF                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                   00  02F1                .BYTE   0                ; 0 HP$W_VECTOR to data
                   09  02F2                .BYTE   9                ; 9 of data field
                   82  02F3                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                52 42 00' 02F4                .ASCIC  "BR"     ; BR string
                   02  02F4
             00000007 00000004  02F7                .LONG   4,7      ; 4 , 7 limits on input
                   88  02FF                .BYTE   Pd$_Store        ; Indicate store operation
                 0032  0300                .WORD   AA11K$B_BR               ; Byte AA11K$B_BR to data
                   00  0302                .BYTE   0                ; 0 AA11K$B_BR to data
                   08  0303                .BYTE   8                ; 8 of data field
                   81  0304                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                       0305    665 AD11K:  $DS_AD11K                                                    ;G:40
                       0305                .SAVE   LOCAL_BLOCK
                       0305                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                       0032                .IIF    NB,AD11K$B_BR,  AD11K$B_BR:
                 00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                       0033                .IIF    NB,AD11K$K_LEN, AD11K$K_LEN:
                   00000305                .RESTORE
          4B 31 31 44 41 00' 0305                .ASCIC  "AD11K" ; AD11K name
                   05  0305
                   33  030B                .BYTE   AD11K$K_LEN              ; AD11K$K_LEN of resultant P-table
                   04  030C                .BYTE   4                ; Maximum unit number                   ;G:65
                 0000  030D                .WORD   ^A""     ; Two character mnemonic for QIO AD11K
                   80  030F                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                   8D  0310                .Byte   Pd$_Name         ; Directive op-code
                   03  0311                .Byte   Ptd$M_Device             ; Enforced AD codes
                44 41 00' 0312                .Ascic  "AD"     ; Enforced device name prefix
                   02  0312
                   83  0315                .BYTE   Pd$_Octal        ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 0316                .ASCIC  "I/O BUS CSR"    ; I/O BUS CSR string
                   0B  0316
             0003FFFE 0003E000  0322                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                   88  032A                .BYTE   Pd$_Store        ; Indicate store operation
                 0018  032B                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                   00  032D                .BYTE   0                ; 0 HP$A_DEVICE to data
                   12  032E                .BYTE   18               ; 18 of data field
                   83  032F                .BYTE   Pd$_Octal        ; Indicate scan octal
          52 4F 54 43 45 56 00' 0330                .ASCIC  "VECTOR"         ; VECTOR string
                   06  0330
             000001FE 00000002  0337                .LONG   ^O2,^O776        ; 2 , 776 limits on input
                   88  033F                .BYTE   Pd$_Store        ; Indicate store operation
                 0024  0340                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                   00  0342                .BYTE   0                ; 0 HP$W_VECTOR to data
```

```
                                09  0343                .BYTE   9                       ; 9 of data field
                                82  0344                .BYTE   Pd$_Decimal          ; Indicate scan decimal
                          52 42 00' 0345                .ASCIC  "BR"     ; BR string
                                02  0345
                  00000007 00000004  0348                .LONG   4,7      ; 4 , 7 limits on input
                                88  0350                .BYTE   Pd$_Store            ; Indicate store operation
                              0032  0351                .WORD   AD11K$B_BR                  ; Byte AD11K$B_BR to data
                                00  0353                .BYTE   0                    ; 0 AD11K$B_BR to data
                                08  0354                .BYTE   8                    ; 8 of data field
                                81  0355                .BYTE   Pd$_End              ; Indicate end of PT-descriptor
                                    0356   666 CIBCI:    $DS_CIBCI                                ;       [51][63]                ;G:17
                    49 43 42 49 43 00' 0356             .ASCIC  "CIBCI" ; CIBCI name
                                05  0356
                                41  035C                .BYTE   CI$K_LEN                    ; CI$K_LEN of resultant P-table
                                10  035D                .BYTE   16                   ; Maximum unit number                   ;G:65
                              0000  035E                .WORD   ^A""     ; Two character mnemonic for QIO CIBCI
                                80  0360                .BYTE   Pd$_Start            ; Constant to identify start of descriptor
                                8D  0361                .Byte   Pd$_Name             ; Directive op-code
                                03  0362                .Byte   Ptd$M_Device                ; Enforced PA codes
                          41 50 00' 0363                .Ascic  "PA"     ; Enforced device name prefix
                                02  0363
                                87  0366                .BYTE   Pd$_Fetch            ; Indicate fetch operation
                              0018  0367                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                                19  0369                .BYTE   25                   ; 25 HP$A_DEVICE to data
                                07  036A                .BYTE   7                    ; 7 of data field
                                8C  036B                .BYTE   Pd$_Case             ; Indicate case operation
                                01  036C                .BYTE   $$_$                 ; Count of pairs
                  00000030 00000000  036D                .LONG   0,^X30               ; Store the pair
                                88  0375                .BYTE   Pd$_Store            ; Indicate store operation
                              0018  0376                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                                19  0378                .BYTE   25                   ; 25 HP$A_DEVICE to data
                                07  0379                .BYTE   7                    ; 7 of data field
                                84  037A                .BYTE   Pd$_Hexadecimal      ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 037B               .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
        29 58 45 48 28 20 72 65 62   0387
                                14  037B
                  0000000F 00000000  0390                .LONG   ^X0,^XF ; 0 , F limits on input
                                88  0398                .BYTE   Pd$_Store            ; Indicate store operation
                              0032  0399                .WORD   CI$B_BI_ID                  ; Byte CI$B_BI_ID to data
                                00  039B                .BYTE   0                    ; 0 CI$B_BI_ID to data
                                08  039C                .BYTE   8                    ; 8 of data field
                                88  039D                .BYTE   Pd$_Store            ; Indicate store operation
                              0018  039E                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                                0D  03A0                .BYTE   13                   ; 13 HP$A_DEVICE to data
                                04  03A1                .BYTE   4                    ; 4 of data field
                                88  03A2                .BYTE   Pd$_Store            ; Indicate store operation
                              0024  03A3                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                                02  03A5                .BYTE   2                    ; 2 HP$W_VECTOR to data
                                04  03A6                .BYTE   4                    ; 4 of data field
                                87  03A7                .BYTE   Pd$_Fetch            ; Indicate fetch operation
                              0018  03A8                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                                00  03AA                .BYTE   0                    ; 0 HP$A_DEVICE to data
                                20  03AB                .BYTE   32                   ; 32 of data field
                                88  03AC                .BYTE   Pd$_Store            ; Indicate store operation
                              001C  03AD                .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                                00  03AF                .BYTE   0                    ; 0 HP$A_DVA to data
                                20  03B0                .BYTE   32                   ; 32 of data field
```

```
                      82  03B1                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 03B2                .ASCIC  "BR"     ; BR string
                      02  03B2
          00000007 00000004  03B5             .LONG   4,7     ; 4 , 7 limits on input
                      88  03BD                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  03BE                .WORD   CI$B_BR         ; Byte CI$B_BR to data
                      00  03C0                .BYTE   0               ; 0 CI$B_BR to data
                      08  03C1                .BYTE   8               ; 8 of data field
                      88  03C2                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  03C3                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      06  03C5                .BYTE   6               ; 6 HP$W_VECTOR to data
                      03  03C6                .BYTE   3               ; 3 of data field
                      82  03C7                .BYTE   Pd$_Decimal     ; Indicate scan decimal
       65 64 6F 4E 20 49 43 00' 03C8          .ASCIC  "CI Node"       ; CI Node string
                      07  03C8
          000000E0 00000000  03D0             .LONG   0,224   ; 0 , 224 limits on input
                      88  03D8                .BYTE   Pd$_Store       ; Indicate store operation
                    0034  03D9                .WORD   CI$B_NODE               ; Byte CI$B_NODE to data
                      00  03DB                .BYTE   0               ; 0 CI$B_NODE to data
                      08  03DC                .BYTE   8               ; 8 of data field
                      86  03DD                .BYTE   Pd$_Literal     ; Indicate literal
                FFFF0F00  03DE                .LONG   ^XFFFF0F00              ; Constant
                      88  03E2                .BYTE   Pd$_Store       ; Indicate store operation
                    0035  03E3                .WORD   Ci$L_Func               ; Byte Ci$L_Func to data
                      00  03E5                .BYTE   0               ; 0 Ci$L_Func to data
                      20  03E6                .BYTE   32              ; 32 of data field
                      86  03E7                .BYTE   Pd$_Literal     ; Indicate literal
                80000002  03E8                .LONG   ^X80000002              ; Constant
                      88  03EC                .BYTE   Pd$_Store       ; Indicate store operation
                    0039  03ED                .WORD   Ci$L_Maint_Id           ; Byte Ci$L_Maint_Id to data
                      00  03EF                .BYTE   0               ; 0 Ci$L_Maint_Id to data
                      20  03F0                .BYTE   32              ; 32 of data field
                      81  03F1                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          03F2  667 CIBCA:      $DS_CIBCA                       ;       [75]            ;G:17
       41 43 42 49 43 00' 03F2                .ASCIC  "CIBCA" ; CIBCA name
                      05  03F2
                      41  03F8                .BYTE   CI$K_LEN                ; CI$K_LEN of resultant P-table
                      10  03F9                .BYTE   16              ; Maximum unit number                   ;G:65
                    0000  03FA                .WORD   ^A""    ; Two character mnemonic for QIO CIBCA
                      80  03FC                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  03FD                .Byte   Pd$_Name        ; Directive op-code
                      03  03FE                .Byte   Ptd$M_Device            ; Enforced PA codes
                41 50 00' 03FF                .Ascic  "PA"    ; Enforced device name prefix
                      02  03FF
                      87  0402                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  0403                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      19  0405                .BYTE   25              ; 25 HP$A_DEVICE to data
                      07  0406                .BYTE   7               ; 7 of data field
                      8C  0407                .BYTE   Pd$_Case        ; Indicate case operation
                      01  0408                .BYTE   $$_$            ; Count of pairs
          00000030 00000000  0409             .LONG   0,^X30          ; Store the pair
                      88  0411                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0412                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      19  0414                .BYTE   25              ; 25 HP$A_DEVICE to data
                      07  0415                .BYTE   7               ; 7 of data field
                      84  0416                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 0417     .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
```

```
        29 58 45 48 28 20 72 65 62  0423
                                14  0417
                  0000000F 00000000  042C                .LONG   ^X0,^XF ; 0 , F limits on input
                                88  0434                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  0435                .WORD   CI$B_BI_ID              ; Byte CI$B_BI_ID to data
                                00  0437                .BYTE   0               ; 0 CI$B_BI_ID to data
                                08  0438                .BYTE   8               ; 8 of data field
                                88  0439                .BYTE   Pd$_Store       ; Indicate store operation
                              0018  043A                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                0D  043C                .BYTE   13              ; 13 HP$A_DEVICE to data
                                04  043D                .BYTE   4               ; 4 of data field
                                88  043E                .BYTE   Pd$_Store       ; Indicate store operation
                              0024  043F                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                02  0441                .BYTE   2               ; 2 HP$W_VECTOR to data
                                04  0442                .BYTE   4               ; 4 of data field
                                87  0443                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                              0018  0444                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                00  0446                .BYTE   0               ; 0 HP$A_DEVICE to data
                                20  0447                .BYTE   32              ; 32 of data field
                                88  0448                .BYTE   Pd$_Store       ; Indicate store operation
                              001C  0449                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                                00  044B                .BYTE   0               ; 0 HP$A_DVA to data
                                20  044C                .BYTE   32              ; 32 of data field
                                82  044D                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                        52 42 00'  044E                .ASCIC  "BR"    ; BR string
                                02  044E
                  00000007 00000004  0451                .LONG   4,7     ; 4 , 7 limits on input
                                88  0459                .BYTE   Pd$_Store       ; Indicate store operation
                              0033  045A                .WORD   CI$B_BR         ; Byte CI$B_BR to data
                                00  045C                .BYTE   0               ; 0 CI$B_BR to data
                                08  045D                .BYTE   8               ; 8 of data field
                                88  045E                .BYTE   Pd$_Store       ; Indicate store operation
                              0024  045F                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                06  0461                .BYTE   6               ; 6 HP$W_VECTOR to data
                                03  0462                .BYTE   3               ; 3 of data field
                                82  0463                .BYTE   Pd$_Decimal     ; Indicate scan decimal
        65 64 6F 4E 20 49 43 00'  0464                .ASCIC  "CI Node"       ; CI Node string
                                07  0464
                  000000E0 00000000  046C                .LONG   0,224   ; 0 , 224 limits on input
                                88  0474                .BYTE   Pd$_Store       ; Indicate store operation
                              0034  0475                .WORD   CI$B_NODE               ; Byte CI$B_NODE to data
                                00  0477                .BYTE   0               ; 0 CI$B_NODE to data
                                08  0478                .BYTE   8               ; 8 of data field
                                86  0479                .BYTE   Pd$_Literal     ; Indicate literal
                          FFFF0F00  047A                .LONG   ^XFFFF0F00              ; Constant
                                88  047E                .BYTE   Pd$_Store       ; Indicate store operation
                              0035  047F                .WORD   Ci$L_Func               ; Byte Ci$L_Func to data
                                00  0481                .BYTE   0               ; 0 Ci$L_Func to data
                                20  0482                .BYTE   32              ; 32 of data field
                                86  0483                .BYTE   Pd$_Literal     ; Indicate literal
                          8000000B  0484                .LONG   ^X8000000B              ; Constant
                                88  0488                .BYTE   Pd$_Store       ; Indicate store operation
                              0039  0489                .WORD   Ci$L_Maint_Id           ; Byte Ci$L_Maint_Id to data
                                00  048B                .BYTE   0               ; 0 Ci$L_Maint_Id to data
                                20  048C                .BYTE   32              ; 32 of data field
                                81  048D                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    048E   668 CI750:   $DS_CI750                       ;       [20]            ;G:17
```

```
30 35 37 49 43 00' 048E          .ASCIC  "CI750" ; CI750 name
                05  048E
                41  0494          .BYTE   CI$K_LEN                ; CI$K_LEN of resultant P-table
                08  0495          .BYTE   8                ; Maximum unit number                    ;G:65
              0000  0496          .WORD   ^A""     ; Two character mnemonic for QIO CI750
                80  0498          .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                8D  0499          .Byte   Pd$_Name         ; Directive op-code
                03  049A          .Byte   Ptd$M_Device            ; Enforced PA codes
          41 50 00' 049B          .Ascic  "PA"     ; Enforced device name prefix
                02  049B
                86  049E          .BYTE   Pd$_Literal      ; Indicate literal
          40F30000  049F          .LONG   ^X40F30000              ; Constant
                88  04A3          .BYTE   Pd$_Store        ; Indicate store operation
              0018  04A4          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                00  04A6          .BYTE   0                ; 0 HP$A_DEVICE to data
                20  04A7          .BYTE   32               ; 32 of data field
                82  04A8          .BYTE   Pd$_Decimal      ; Indicate scan decimal
     74 6F 6C 53 00' 04A9          .ASCIC  "Slot"   ; Slot string
                04  04A9
 0000000F 0000000A  04AE          .LONG   10,15    ; 10 , 15 limits on input
                88  04B6          .BYTE   Pd$_Store        ; Indicate store operation
              0032  04B7          .WORD   CI$B_TR          ; Byte CI$B_TR to data
                00  04B9          .BYTE   0                ; 0 CI$B_TR to data
                08  04BA          .BYTE   8                ; 8 of data field
                88  04BB          .BYTE   Pd$_Store        ; Indicate store operation
              0018  04BC          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                0D  04BE          .BYTE   13               ; 13 HP$A_DEVICE to data
                04  04BF          .BYTE   4                ; 4 of data field
                88  04C0          .BYTE   Pd$_Store        ; Indicate store operation
              0024  04C1          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                02  04C3          .BYTE   2                ; 2 HP$W_VECTOR to data
                04  04C4          .BYTE   4                ; 4 of data field
                87  04C5          .BYTE   Pd$_Fetch        ; Indicate fetch operation
              0018  04C6          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                00  04C8          .BYTE   0                ; 0 HP$A_DEVICE to data
                20  04C9          .BYTE   32               ; 32 of data field
                88  04CA          .BYTE   Pd$_Store        ; Indicate store operation
              001C  04CB          .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                00  04CD          .BYTE   0                ; 0 HP$A_DVA to data
                20  04CE          .BYTE   32               ; 32 of data field
                82  04CF          .BYTE   Pd$_Decimal      ; Indicate scan decimal
          52 42 00' 04D0          .ASCIC  "BR"     ; BR string
                02  04D0
 00000007 00000004  04D3          .LONG   4,7      ; 4 , 7 limits on input
                88  04DB          .BYTE   Pd$_Store        ; Indicate store operation
              0033  04DC          .WORD   CI$B_BR          ; Byte CI$B_BR to data
                00  04DE          .BYTE   0                ; 0 CI$B_BR to data
                08  04DF          .BYTE   8                ; 8 of data field
                88  04E0          .BYTE   Pd$_Store        ; Indicate store operation
              0024  04E1          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                06  04E3          .BYTE   6                ; 6 HP$W_VECTOR to data
                03  04E4          .BYTE   3                ; 3 of data field
                82  04E5          .BYTE   Pd$_Decimal      ; Indicate scan decimal
     65 64 6F 4E 00' 04E6          .ASCIC  "Node"   ; Node string
                04  04E6
 000000FF 00000000  04EB          .LONG   0,255    ; 0 , 255 limits on input
                88  04F3          .BYTE   Pd$_Store        ; Indicate store operation
```

```
                     0034  04F4                .WORD   CI$B_NODE               ; Byte CI$B_NODE to data
                       00  04F6                .BYTE   0                   ; 0 CI$B_NODE to data
                       08  04F7                .BYTE   8                   ; 8 of data field
                       86  04F8                .BYTE   Pd$_Literal          ; Indicate literal
                 FFFF0F00  04F9                .LONG   ^XFFFF0F00                  ; Constant
                       88  04FD                .BYTE   Pd$_Store           ; Indicate store operation
                     0035  04FE                .WORD   Ci$L_Func               ; Byte Ci$L_Func to data
                       00  0500                .BYTE   0                   ; 0 Ci$L_Func to data
                       20  0501                .BYTE   32                  ; 32 of data field
                       86  0502                .BYTE   Pd$_Literal          ; Indicate literal
                 80000002  0503                .LONG   ^X80000002                  ; Constant
                       88  0507                .BYTE   Pd$_Store           ; Indicate store operation
                     0039  0508                .WORD   Ci$L_Maint_Id           ; Byte Ci$L_Maint_Id to data
                       00  050A                .BYTE   0                   ; 0 Ci$L_Maint_Id to data
                       20  050B                .BYTE   32                  ; 32 of data field
                       81  050C                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           050D    669 CI780:  $DS_CI780                                               ;G:17
           30 38 37 49 43 00'  050D            .ASCIC  "CI780" ; CI780 name
                       05  050D
                       41  0513                .BYTE   CI$K_LEN                ; CI$K_LEN of resultant P-table
                       08  0514                .BYTE   8                   ; Maximum unit number               ;G:65
                     0000  0515                .WORD   ^A""    ; Two character mnemonic for QIO CI780
                       80  0517                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  0518                .Byte   Pd$_Name            ; Directive op-code
                       03  0519                .Byte   Ptd$M_Device            ; Enforced PA codes
                 41 50 00' 051A                .Ascic  "PA"    ; Enforced device name prefix
                       02  051A
                       82  051D                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                 52 54 00' 051E                .ASCIC  "TR"    ; TR string
                       02  051E
           0000000F 00000001  0521             .LONG   1,15    ; 1 , 15 limits on input
                       88  0529                .BYTE   Pd$_Store           ; Indicate store operation
                     0032  052A                .WORD   CI$B_TR             ; Byte CI$B_TR to data
                       00  052C                .BYTE   0                   ; 0 CI$B_TR to data
                       08  052D                .BYTE   8                   ; 8 of data field
                       88  052E                .BYTE   Pd$_Store           ; Indicate store operation
                     0018  052F                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                       0D  0531                .BYTE   13                  ; 13 HP$A_DEVICE to data
                       04  0532                .BYTE   4                   ; 4 of data field
                       88  0533                .BYTE   Pd$_Store           ; Indicate store operation
                     0024  0534                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                       02  0536                .BYTE   2                   ; 2 HP$W_VECTOR to data
                       04  0537                .BYTE   4                   ; 4 of data field
                       82  0538                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                 52 42 00' 0539                .ASCIC  "BR"    ; BR string
                       02  0539
           00000007 00000004  053C             .LONG   4,7     ; 4 , 7 limits on input
                       88  0544                .BYTE   Pd$_Store           ; Indicate store operation
                     0033  0545                .WORD   CI$B_BR             ; Byte CI$B_BR to data
                       00  0547                .BYTE   0                   ; 0 CI$B_BR to data
                       08  0548                .BYTE   8                   ; 8 of data field
                       88  0549                .BYTE   Pd$_Store           ; Indicate store operation
                     0024  054A                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                       06  054C                .BYTE   6                   ; 6 HP$W_VECTOR to data
                       02  054D                .BYTE   2                   ; 2 of data field
                       86  054E                .BYTE   Pd$_Literal          ; Indicate literal
                 00000006  054F                .LONG   6                   ; Constant
```

```
                      88  0553                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  0554                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      1C  0556                .BYTE   28              ; 28 HP$A_DEVICE to data
                      04  0557                .BYTE   4               ; 4 of data field
                      87  0558                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  0559                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  055B                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  055C                .BYTE   32              ; 32 of data field
                      88  055D                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  055E                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      00  0560                .BYTE   0               ; 0 HP$A_DVA to data
                      20  0561                .BYTE   32              ; 32 of data field
                      86  0562                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  0563                .LONG   1               ; Constant
                      88  0567                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  0568                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      08  056A                .BYTE   8               ; 8 HP$W_VECTOR to data
                      01  056B                .BYTE   1               ; 1 of data field
                      82  056C                .BYTE   Pd$_Decimal     ; Indicate scan decimal
          65 64 6F 4E 00' 056D                .ASCIC  "Node"  ; Node string
                      04  056D
            000000FF 00000000  0572           .LONG   0,255   ; 0 , 255 limits on input
                      88  057A                .BYTE   Pd$_Store       ; Indicate store operation
                    0034  057B                .WORD   CI$B_NODE               ; Byte CI$B_NODE to data
                      00  057D                .BYTE   0               ; 0 CI$B_NODE to data
                      08  057E                .BYTE   8               ; 8 of data field
                      86  057F                .BYTE   Pd$_Literal     ; Indicate literal
                FFFF0F00  0580                .LONG   ^XFFFF0F00              ; Constant
                      88  0584                .BYTE   Pd$_Store       ; Indicate store operation
                    0035  0585                .WORD   Ci$L_Func               ; Byte Ci$L_Func to data
                      00  0587                .BYTE   0               ; 0 Ci$L_Func to data
                      20  0588                .BYTE   32              ; 32 of data field
                      86  0589                .BYTE   Pd$_Literal     ; Indicate literal
                80000002  058A                .LONG   ^X80000002              ; Constant
                      88  058E                .BYTE   Pd$_Store       ; Indicate store operation
                    0039  058F                .WORD   Ci$L_Maint_Id           ; Byte Ci$L_Maint_Id to data
                      00  0591                .BYTE   0               ; 0 Ci$L_Maint_Id to data
                      20  0592                .BYTE   32              ; 32 of data field
                      81  0593                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          0594    670 CI_NODE:$DS_CI_NODE                         ;       [20]                    ;G:17
45 44 4F 4E 5F 49 43 00'  0594                .ASCIC  "CI_NODE"       ; CI_NODE name
                      07  0594
                      41  059C                .BYTE   CI$K_LEN                ; CI$K_LEN of resultant P-table
                      00  059D                .BYTE   0               ; Maximum unit number                   ;G:65
                    0000  059E                .WORD   ^A""    ; Two character mnemonic for QIO CI_NODE
                      80  05A0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  05A1                .Byte   Pd$_Name        ; Directive op-code
                      04  05A2                .Byte   Ptd$M_NAME              ; Enforced  codes
                      00' 05A3                .Ascic  ""      ; Enforced device name prefix
                      00  05A3
                      87  05A4                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  05A5                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  05A7                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  05A8                .BYTE   32              ; 32 of data field
                      88  05A9                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  05AA                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      00  05AC                .BYTE   0               ; 0 HP$A_DVA to data
```

```
                                20  05AD          .BYTE   32              ; 32 of data field
                                85  05AE          .BYTE   Pd$_String      ; Indicate scan and verify string
   65 70 79 74 5F 65 64 6F 4E 00' 05AF          .ASCIC  "Node_type"     ; Node_type string
                                09  05AF
                             00 01  05B9          .BYTE   1, 0
                             00 01  05BB          .BYTE   1, 0
            30 38 37 58 41 56 00' 05BD          .ASCIC  "VAX780"
                                06  05BD
            30 35 37 58 41 56 00' 05C4          .ASCIC  "VAX750"
                                06  05C4
               30 35 43 53 48 00' 05CB          .ASCIC  "HSC50"
                                05  05CB
                             00 01  05D1          .BYTE   1, 0
                  30 31 4C 4B 00' 05D3          .ASCIC  "KL10"
                                04  05D3
                  54 4E 49 43 00' 05D8          .ASCIC  "CINT"
                                04  05D8
            35 38 37 58 41 56 00' 05DD          .ASCIC  "VAX785"
                                06  05DD
         30 30 36 38 58 41 56 00' 05E4          .ASCIC  "VAX8600"
                                07  05E4
         30 30 32 38 58 41 56 00' 05EC          .ASCIC  "VAX8200"
                                07  05EC
         30 30 38 38 58 41 56 00' 05F4          .ASCIC  "VAX8800"
                                07  05F4
                                00  05FC          .BYTE   0               ; Null length is end of valid ,,
                                88  05FD          .BYTE   Pd$_Store       ; Indicate store operation
                              003D  05FE          .WORD   Ci$L_Index              ; Byte Ci$L_Index to data
                                00  0600          .BYTE   0               ; 0 Ci$L_Index to data
                                20  0601          .BYTE   32              ; 32 of data field
                                8C  0602          .BYTE   Pd$_Case        ; Indicate case operation
                                05  0603          .BYTE   $$_$            ; Count of pairs
              00000002 00000003  0604          .LONG   ^X3,                    ^X2             ; Store the pair
              00000002 00000008  060C          .LONG   ^X8,                    ^X2             ; Store the pair
              00000002 00000009  0614          .LONG   ^X9,                    ^X2             ; Store the pair
              00000002 0000000A  061C          .LONG   ^XA,                    ^X2             ; Store the pair
              00000002 0000000B  0624          .LONG   ^XB,                    ^X2             ; Store the pair
                                88  062C          .BYTE   Pd$_Store       ; Indicate store operation
                              0039  062D          .WORD   Ci$L_Maint_Id           ; Byte Ci$L_Maint_Id to data
                                00  062F          .BYTE   0               ; 0 Ci$L_Maint_Id to data
                                20  0630          .BYTE   32              ; 32 of data field
                                87  0631          .BYTE   Pd$_Fetch       ; Indicate fetch operation
                              003D  0632          .WORD   Ci$L_index              ; Byte Ci$L_index to data
                                00  0634          .BYTE   0               ; 0 Ci$L_index to data
                                20  0635          .BYTE   32              ; 32 of data field
                                8C  0636          .BYTE   Pd$_Case        ; Indicate case operation
                                09  0637          .BYTE   $$_$            ; Count of pairs
              FFFF0F00 00000002  0638          .LONG   ^X2,                    ^XFFFF0F00              ; Store the pair
              FFFF0F00 00000003  0640          .LONG   ^X3,                    ^XFFFF0F00              ; Store the pair
              4F710200 00000004  0648          .LONG   ^X4,                    ^X4F710200              ; Store the pair
              A3FFAD00 00000006  0650          .LONG   ^X6,                    ^XA3FFAD00              ; Store the pair
              FFFFFF00 00000007  0658          .LONG   ^X7,                    ^XFFFFFF00              ; Store the pair
              FFFF0F00 00000008  0660          .LONG   ^X8,                    ^XFFFF0F00              ; Store the pair
              FFFF0F00 00000009  0668          .LONG   ^X9,                    ^XFFFF0F00              ; Store the pair
              FFFF0F00 0000000A  0670          .LONG   ^XA,                    ^XFFFF0F00              ; Store the pair
              FFFF0F00 0000000B  0678          .LONG   ^XB,                    ^XFFFF0F00              ; Store the pair
                                88  0680          .BYTE   Pd$_Store       ; Indicate store operation
```

```
                          0035  0681                .WORD   Ci$L_Func               ; Byte Ci$L_Func to data
                            00  0683                .BYTE   0                   ; 0 Ci$L_Func to data
                            20  0684                .BYTE   32                  ; 32 of data field
                            82  0685                .BYTE   Pd$_Decimal         ; Indicate scan decimal
73 65 72 64 64 61 5F 65 64 6F 4E 00' 0686           .ASCIC  "Node_address"      ; Node_address string
                            73  0692
                            0C  0686
                  000000FF 00000000  0693           .LONG   0,255   ; 0 , 255 limits on input
                            88  069B                .BYTE   Pd$_Store           ; Indicate store operation
                          0034  069C                .WORD   CI$B_NODE                   ; Byte CI$B_NODE to data
                            00  069E                .BYTE   0                   ; 0 CI$B_NODE to data
                            08  069F                .BYTE   8                   ; 8 of data field
                            88  06A0                .BYTE   Pd$_Store           ; Indicate store operation
                          000B  06A1                .WORD   HP$B_DRIVE                  ; Byte HP$B_DRIVE to data
                            00  06A3                .BYTE   0                   ; 0 HP$B_DRIVE to data
                            08  06A4                .BYTE   8                   ; 8 of data field
                            86  06A5                .BYTE   Pd$_Literal         ; Indicate literal
                      80000000  06A6                .LONG   ^X80000000                  ; Constant
                            8A  06AA                .BYTE   Pd$_Add             ; Indicate add value to field operation
                          0039  06AB                .WORD   CI$L_MAINT_ID               ; Byte CI$L_MAINT_ID to data
                            00  06AD                .BYTE   0                   ; 0 CI$L_MAINT_ID to data
                            20  06AE                .BYTE   32                  ; 32 of data field
                            81  06AF                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                                06B0    671 CONSOLE:$DS_CONSOLE                                                 ;G:40
                                06B0                .SAVE   LOCAL_BLOCK
                                06B0                .PSECT  $ABS$,ABS
                      00000032  0050                .=Hp$A_Dependent
                                0032                .IIF    NB,Console$K_Len,       Console$K_Len:
                                000006B0            .RESTORE
          65 6C 6F 73 6E 6F 63 00' 06B0             .ASCIC  "console"           ; console name
                            07  06B0
                            32  06B8                .BYTE   Console$K_Len               ; Console$K_Len of resultant P-table
                            01  06B9                .BYTE   1                   ; Maximum unit number                   ;G:65
                          0000  06BA                .WORD   ^A""    ; Two character mnemonic for QIO console
                            80  06BC                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                            81  06BD                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                                06BE    672 CR11:   $DS_CR11                                                    ;G:17
                                06BE                .SAVE   LOCAL_BLOCK
                                06BE                .PSECT  $ABS$,ABS
                      00000032  0050                .=HP$A_DEPENDENT
                                0032                .IIF    NB,CR11$L_CSR,  CR11$L_CSR:
                      00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                0036                .IIF    NB,CR11$B_BR,   CR11$B_BR:
                      00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                0037                .IIF    NB,CR11$K_LEN,  CR11$K_LEN:
                                000006BE            .RESTORE
                    31 31 52 43 00' 06BE            .ASCIC  "CR11"  ; CR11 name
                            04  06BE
                            37  06C3                .BYTE   CR11$K_LEN                  ; CR11$K_LEN of resultant P-table
                            01  06C4                .BYTE   1                   ; Maximum unit number                   ;G:65
                          0000  06C5                .WORD   ^A""    ; Two character mnemonic for QIO CR11
                            80  06C7                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                            8D  06C8                .Byte   Pd$_Name            ; Directive op-code
                            03  06C9                .Byte   Ptd$M_Device                ; Enforced CR codes
                    52 43 00' 06CA                  .Ascic  "CR"    ; Enforced device name prefix
                            02  06CA
                            86  06CD                .BYTE   Pd$_Literal         ; Indicate literal
```

```
                00000001  06CE                .LONG   HP$M_ALLOC                      ; Constant
                      88  06D2                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  06D3                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  06D5                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  06D6                .BYTE   8                       ; 8 of data field
                      83  06D7                .BYTE   Pd$_Octal               ; Indicate scan octal
             52 53 43 00' 06D8                .ASCIC  "CSR"   ; CSR string
                      03  06D8
         0003FFFE 0003E000  06DC                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  06E4                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  06E5                .WORD   CR11$L_CSR                      ; Byte CR11$L_CSR to data
                      00  06E7                .BYTE   0                       ; 0 CR11$L_CSR to data
                      20  06E8                .BYTE   32                      ; 32 of data field
                      88  06E9                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  06EA                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  06EC                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  06ED                .BYTE   18                      ; 18 of data field
                      83  06EE                .BYTE   Pd$_Octal               ; Indicate scan octal
   52 4F 54 43 45 56 00' 06EF                .ASCIC  "VECTOR"                ; VECTOR string
                      06  06EF
         000001FE 00000002  06F6                .LONG   ^O2,^O776               ; 2 , 776 limits on input
                      88  06FE                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  06FF                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  0701                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  0702                .BYTE   9                       ; 9 of data field
                      82  0703                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                52 42 00' 0704                .ASCIC  "BR"    ; BR string
                      02  0704
         00000007 00000004  0707                .LONG   4,7     ; 4 , 7 limits on input
                      88  070F                .BYTE   Pd$_Store               ; Indicate store operation
                    0036  0710                .WORD   CR11$B_BR                       ; Byte CR11$B_BR to data
                      00  0712                .BYTE   0                       ; 0 CR11$B_BR to data
                      08  0713                .BYTE   8                       ; 8 of data field
                      81  0714                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          0715    673 DEBNA:  $DS_DEBNA                                                       ;G:41
                          0715                .SAVE   LOCAL_BLOCK
                          0715                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DEBNA$B_node_id,     DEBNA$B_node_id:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,DEBNA$K_LEN, DEBNA$K_LEN:
                00000715  0715                .RESTORE
   41 4E 42 45 4A 00' 0715                .ASCIC  "DEBNA" ; DEBNA name
                      05  0715
                      33  071B                .BYTE   DEBNA$K_LEN                     ; DEBNA$K_LEN of resultant P-table
                      0F  071C                .BYTE   15                      ; Maximum unit number                       ;G:65
                    0000  071D                .WORD   ^A""    ; Two character mnemonic for QIO DEBNA
                      80  071F                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  0720                .Byte   Pd$_Name                ; Directive op-code
                      02  0721                .Byte   PTD$M_CONTROLLER                ; Enforced ET codes
                54 45 00' 0722                .Ascic  "ET"    ; Enforced device name prefix
                      02  0722
                      87  0725                .BYTE   Pd$_Fetch               ; Indicate fetch operation
                    0018  0726                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      19  0728                .BYTE   25                      ; 25 HP$A_DEVICE to data
                      07  0729                .BYTE   7                       ; 7 of data field
                      8C  072A                .BYTE   Pd$_Case                ; Indicate case operation
```

```
                                  01  072B          .BYTE   $$_$              ; Count of pairs
                    00000030 00000000  072C          .LONG   0,^X30            ; Store the pair
                                  88  0734          .BYTE   Pd$_Store         ; Indicate store operation
                                0018  0735          .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                                  19  0737          .BYTE   25                ; 25 HP$A_DEVICE to data
                                  07  0738          .BYTE   7                 ; 7 of data field
                                  88  0739          .BYTE   Pd$_Store         ; Indicate store operation
                                001C  073A          .WORD   HP$A_DVA                  ; Byte HP$A_DVA to data
                                  19  073C          .BYTE   25                ; 25 HP$A_DVA to data
                                  07  073D          .BYTE   7                 ; 7 of data field
                                  86  073E          .BYTE   Pd$_Literal       ; Indicate literal
                            00000001  073F          .LONG   ^X1               ; Constant
                                  88  0743          .BYTE   Pd$_Store         ; Indicate store operation
                                001C  0744          .WORD   HP$A_DVA                  ; Byte HP$A_DVA to data
                                  16  0746          .BYTE   22                ; 22 HP$A_DVA to data
                                  01  0747          .BYTE   1                 ; 1 of data field
                                  84  0748          .BYTE   Pd$_Hexadecimal   ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'  0749          .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
        29 58 45 48 28 20 72 65 62    0755
                                  14  0749
                    0000000F 00000000  075E          .LONG   ^X0,^XF ; 0 , F limits on input
                                  88  0766          .BYTE   Pd$_Store         ; Indicate store operation
                                0032  0767          .WORD   DEBNA$B_node_id           ; Byte DEBNA$B_node_id to data
                                  00  0769          .BYTE   0                 ; 0 DEBNA$B_node_id to data
                                  08  076A          .BYTE   8                 ; 8 of data field
                                  88  076B          .BYTE   Pd$_Store         ; Indicate store operation
                                000B  076C          .WORD   HP$B_DRIVE                ; Byte HP$B_DRIVE to data
                                  00  076E          .BYTE   0                 ; 0 HP$B_DRIVE to data
                                  08  076F          .BYTE   8                 ; 8 of data field
                                  88  0770          .BYTE   Pd$_Store         ; Indicate store operation
                                0018  0771          .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                                  0D  0773          .BYTE   13                ; 13 HP$A_DEVICE to data
                                  04  0774          .BYTE   4                 ; 4 of data field
                                  88  0775          .BYTE   Pd$_Store         ; Indicate store operation
                                001C  0776          .WORD   HP$A_DVA                  ; Byte HP$A_DVA to data
                                  12  0778          .BYTE   18                ; 18 HP$A_DVA to data
                                  04  0779          .BYTE   4                 ; 4 of data field
                                  88  077A          .BYTE   Pd$_Store         ; Indicate store operation
                                0024  077B          .WORD   HP$W_VECTOR               ; Byte HP$W_VECTOR to data
                                  02  077D          .BYTE   2                 ; 2 HP$W_VECTOR to data
                                  04  077E          .BYTE   4                 ; 4 of data field
                                  81  077F          .BYTE   Pd$_End           ; Indicate end of PT-descriptor
                                      0780  674 DEBNK:  $DS_DEBNK                                                    ;G:41
                                      0780          .SAVE   LOCAL_BLOCK
                                      0780          .PSECT  $ABS$,ABS
                            00000032  0050          .=HP$A_DEPENDENT
                                      0032          .IIF    NB,DEBNK$B_node_id,     DEBNK$B_node_id:
                            00000033  0032          .IIF    NB,.BLKB,       .BLKB   1
                                      0033          .IIF    NB,DEBNK$K_LEN, DEBNK$K_LEN:
                                00000780          .RESTORE
                    4B 4E 42 45 44 00'  0780          .ASCIC  "DEBNK" ; DEBNK name
                                  05  0780
                                  33  0786          .BYTE   DEBNK$K_LEN               ; DEBNK$K_LEN of resultant P-table
                                  0F  0787          .BYTE   15                ; Maximum unit number                  ;G:65
                                0000  0788          .WORD   ^A""    ; Two character mnemonic for QIO DEBNK
                                  80  078A          .BYTE   Pd$_Start         ; Constant to identify start of descriptor
                                  8D  078B          .Byte   Pd$_Name          ; Directive op-code
```

```
                                 02  078C          .Byte   PTD$M_CONTROLLER                ; Enforced ET codes
                          54 45 00'  078D          .Ascic  "ET"     ; Enforced device name prefix
                                 02  078D
                                 87  0790          .BYTE   Pd$_Fetch        ; Indicate fetch operation
                               0018  0791          .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                                 19  0793          .BYTE   25               ; 25 HP$A_DEVICE to data
                                 07  0794          .BYTE   7                ; 7 of data field
                                 8C  0795          .BYTE   Pd$_Case         ; Indicate case operation
                                 01  0796          .BYTE   $$_$             ; Count of pairs
                   00000030 00000000  0797          .LONG   0,^X30           ; Store the pair
                                 88  079F          .BYTE   Pd$_Store        ; Indicate store operation
                               0018  07A0          .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                                 19  07A2          .BYTE   25               ; 25 HP$A_DEVICE to data
                                 07  07A3          .BYTE   7                ; 7 of data field
                                 88  07A4          .BYTE   Pd$_Store        ; Indicate store operation
                               001C  07A5          .WORD   HP$A_DVA                 ; Byte HP$A_DVA to data
                                 19  07A7          .BYTE   25               ; 25 HP$A_DVA to data
                                 07  07A8          .BYTE   7                ; 7 of data field
                                 86  07A9          .BYTE   Pd$_Literal      ; Indicate literal
                          00000001  07AA          .LONG   ^X1              ; Constant
                                 88  07AE          .BYTE   Pd$_Store        ; Indicate store operation
                               001C  07AF          .WORD   HP$A_DVA                 ; Byte HP$A_DVA to data
                                 16  07B1          .BYTE   22               ; 22 HP$A_DVA to data
                                 01  07B2          .BYTE   1                ; 1 of data field
                                 84  07B3          .BYTE   Pd$_Hexadecimal  ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'  07B4          .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
         29 58 45 48 28 20 72 65 62  07C0
                                 14  07B4
                   0000000F 00000000  07C9          .LONG   ^X0,^XF ; 0 , F limits on input
                                 88  07D1          .BYTE   Pd$_Store        ; Indicate store operation
                               0032  07D2          .WORD   DEBNK$B_node_id          ; Byte DEBNK$B_node_id to data
                                 00  07D4          .BYTE   0                ; 0 DEBNK$B_node_id to data
                                 08  07D5          .BYTE   8                ; 8 of data field
                                 88  07D6          .BYTE   Pd$_Store        ; Indicate store operation
                               000B  07D7          .WORD   HP$B_DRIVE               ; Byte HP$B_DRIVE to data
                                 00  07D9          .BYTE   0                ; 0 HP$B_DRIVE to data
                                 08  07DA          .BYTE   8                ; 8 of data field
                                 88  07DB          .BYTE   Pd$_Store        ; Indicate store operation
                               0018  07DC          .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                                 0D  07DE          .BYTE   13               ; 13 HP$A_DEVICE to data
                                 04  07DF          .BYTE   4                ; 4 of data field
                                 88  07E0          .BYTE   Pd$_Store        ; Indicate store operation
                               001C  07E1          .WORD   HP$A_DVA                 ; Byte HP$A_DVA to data
                                 12  07E3          .BYTE   18               ; 18 HP$A_DVA to data
                                 04  07E4          .BYTE   4                ; 4 of data field
                                 88  07E5          .BYTE   Pd$_Store        ; Indicate store operation
                               0024  07E6          .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                                 02  07E8          .BYTE   2                ; 2 HP$W_VECTOR to data
                                 04  07E9          .BYTE   4                ; 4 of data field
                                 81  07EA          .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                     07EB  675 DEBNT:  $DS_DEBNT                                ;       [68]                    ;G:17
                                     07EB          .SAVE   LOCAL_BLOCK
                                     07EB          .PSECT  $ABS$,ABS
                          00000032  0050          .=HP$A_DEPENDENT
                                     0032          .IIF    NB,DEBNT$B_node_id,     DEBNT$B_node_id:
                          00000033  0032          .IIF    NB,.BLKB,       .BLKB   1
                                     0033          .IIF    NB,DEBNT$K_LEN, DEBNT$K_LEN:
```

```
                    000007EB                .RESTORE
          54 4E 42 45 44 00' 07EB           .ASCIC  "DEBNT" ; DEBNT name
                       05  07EB
                       33  07F1           .BYTE   DEBNT$K_LEN             ; DEBNT$K_LEN of resultant P-table
                       0F  07F2           .BYTE   15                      ; Maximum unit number                   ;G:65
                     0000  07F3           .WORD   ^A""    ; Two character mnemonic for QIO DEBNT
                       80  07F5           .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  07F6           .Byte   Pd$_Name        ; Directive op-code
                       02  07F7           .Byte   PTD$M_CONTROLLER                ; Enforced ET codes
                  54 45 00' 07F8           .Ascic  "ET"    ; Enforced device name prefix
                       02  07F8
                       87  07FB           .BYTE   Pd$_Fetch       ; Indicate fetch operation
                     0018  07FC           .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       19  07FE           .BYTE   25                      ; 25 HP$A_DEVICE to data
                       07  07FF           .BYTE   7                       ; 7 of data field
                       8C  0800           .BYTE   Pd$_Case        ; Indicate case operation
                       01  0801           .BYTE   $$_$            ; Count of pairs
          00000030 00000000  0802           .LONG   0,^X30          ; Store the pair
                       88  080A           .BYTE   Pd$_Store       ; Indicate store operation
                     0018  080B           .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       19  080D           .BYTE   25                      ; 25 HP$A_DEVICE to data
                       07  080E           .BYTE   7                       ; 7 of data field
                       88  080F           .BYTE   Pd$_Store       ; Indicate store operation
                     001C  0810           .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       19  0812           .BYTE   25                      ; 25 HP$A_DVA to data
                       07  0813           .BYTE   7                       ; 7 of data field
                       86  0814           .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  0815           .LONG   ^X1             ; Constant
                       88  0819           .BYTE   Pd$_Store       ; Indicate store operation
                     001C  081A           .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       16  081C           .BYTE   22                      ; 22 HP$A_DVA to data
                       01  081D           .BYTE   1                       ; 1 of data field
                       84  081E           .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 081F  .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
         29 58 45 48 28 20 72 65 62  082B
                       14  081F
                 0000000F 00000000  0834           .LONG   ^X0,^XF ; 0 , F limits on input
                       88  083C           .BYTE   Pd$_Store       ; Indicate store operation
                     0032  083D           .WORD   DEBNT$B_node_id         ; Byte DEBNT$B_node_id to data
                       00  083F           .BYTE   0                       ; 0 DEBNT$B_node_id to data
                       08  0840           .BYTE   8                       ; 8 of data field
                       88  0841           .BYTE   Pd$_Store       ; Indicate store operation
                     000B  0842           .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                       00  0844           .BYTE   0                       ; 0 HP$B_DRIVE to data
                       08  0845           .BYTE   8                       ; 8 of data field
                       88  0846           .BYTE   Pd$_Store       ; Indicate store operation
                     0018  0847           .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       0D  0849           .BYTE   13                      ; 13 HP$A_DEVICE to data
                       04  084A           .BYTE   4                       ; 4 of data field
                       88  084B           .BYTE   Pd$_Store       ; Indicate store operation
                     001C  084C           .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                       12  084E           .BYTE   18                      ; 18 HP$A_DVA to data
                       04  084F           .BYTE   4                       ; 4 of data field
                       88  0850           .BYTE   Pd$_Store       ; Indicate store operation
                     0024  0851           .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       02  0853           .BYTE   2                       ; 2 HP$W_VECTOR to data
                       04  0854           .BYTE   4                       ; 4 of data field
```

```
                                  81  0855                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                      0856   676 DHB32:   $DS_DHB32                                  ;       [84]                  ;G:39
                                      0856                .SAVE   LOCAL_BLOCK
                                      0856                .PSECT  $ABS$,ABS
                            00000032  0050                .=HP$A_DEPENDENT
                                      0032                .IIF    NB,HP$B_DHB32_NODE_ID,  HP$B_DHB32_NODE_ID::
                            00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                                      0033                .IIF    NB,HP$K_DHB32_LEN,      HP$K_DHB32_LEN::
                                00000856                .RESTORE
                  32 33 42 48 44 00'  0856                .ASCIC  "DHB32" ; DHB32 name
                                  05  0856
                                  33  085C                .BYTE   HP$K_DHB32_LEN          ; HP$K_DHB32_LEN of resultant P-table
                                  00  085D                .BYTE   0                ; Maximum unit number                          ;G:65
                                0000  085E                .WORD   ^A""     ; Two character mnemonic for QIO DHB32
                                  80  0860                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                                  87  0861                .BYTE   Pd$_Fetch        ; Indicate fetch operation
                                0018  0862                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                                  19  0864                .BYTE   25               ; 25 HP$A_DEVICE to data
                                  07  0865                .BYTE   7                ; 7 of data field
                                  8C  0866                .BYTE   Pd$_Case         ; Indicate case operation
                                  01  0867                .BYTE   $$_$             ; Count of pairs
                  00000030 00000000  0868                .LONG   0,^X30           ; Store the pair
                                  88  0870                .BYTE   Pd$_Store        ; Indicate store operation
                                0018  0871                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                                  19  0873                .BYTE   25               ; 25 HP$A_DEVICE to data
                                  07  0874                .BYTE   7                ; 7 of data field
                                  88  0875                .BYTE   Pd$_Store        ; Indicate store operation
                                001C  0876                .WORD   HP$A_DVA                 ; Byte HP$A_DVA to data
                                  19  0878                .BYTE   25               ; 25 HP$A_DVA to data
                                  07  0879                .BYTE   7                ; 7 of data field
                                  86  087A                .BYTE   Pd$_Literal      ; Indicate literal
                            00000001  087B                .LONG   ^X1              ; Constant
                                  88  087F                .BYTE   Pd$_Store        ; Indicate store operation
                                001C  0880                .WORD   HP$A_DVA                 ; Byte HP$A_DVA to data
                                  16  0882                .BYTE   22               ; 22 HP$A_DVA to data
                                  01  0883                .BYTE   1                ; 1 of data field
                                  88  0884                .BYTE   Pd$_Store        ; Indicate store operation
                                0024  0885                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                                  08  0887                .BYTE   8                ; 8 HP$W_VECTOR to data
                                  01  0888                .BYTE   1                ; 1 of data field
                                  84  0889                .BYTE   Pd$_Hexadecimal  ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'  088A                .ASCIC  "BI Node Number (HEX) " ; BI Node Number (HEX)  string
      20 29 58 45 48 28 20 72 65 62  0896
                                  15  088A
                  0000000F 00000000  08A0                .LONG   ^X0,^XF ; 0 , F limits on input
                                  88  08A8                .BYTE   Pd$_Store        ; Indicate store operation
                                0032  08A9                .WORD   HP$B_DHB32_NODE_ID              ; Byte HP$B_DHB32_NODE_ID to data
                                  00  08AB                .BYTE   0                ; 0 HP$B_DHB32_NODE_ID to data
                                  08  08AC                .BYTE   8                ; 8 of data field
                                  88  08AD                .BYTE   Pd$_Store        ; Indicate store operation
                                0018  08AE                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                                  0D  08B0                .BYTE   13               ; 13 HP$A_DEVICE to data
                                  04  08B1                .BYTE   4                ; 4 of data field
                                  88  08B2                .BYTE   Pd$_Store        ; Indicate store operation
                                0024  08B3                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                                  02  08B5                .BYTE   2                ; 2 HP$W_VECTOR to data
                                  04  08B6                .BYTE   4                ; 4 of data field
```

```
                    88  08B7                .BYTE   Pd$_Store        ; Indicate store operation
                  001C  08B8                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                    12  08BA                .BYTE   18               ; 18 HP$A_DVA to data
                    04  08BB                .BYTE   4                ; 4 of data field
                    81  08BC                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                        08BD   677 DHU11:   $DS_DHU11                        ;       [56]                    ;G:17
     31 31 55 48 44 00' 08BD                .ASCIC  "DHU11" ; DHU11 name
                    05  08BD
                    39  08C3                .BYTE   HP$K_DHUV_LEN           ; HP$K_DHUV_LEN of resultant P-table
                    08  08C4                .BYTE   8                ; Maximum unit number                   ;G:65
                  5854  08C5                .WORD   ^A"TX"  ; Two character mnemonic for QIO DHU11 TX
                    80  08C7                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                    8D  08C8                .Byte   Pd$_Name         ; Directive op-code
                    02  08C9                .Byte   Ptd$M_Controller                ; Enforced TX codes
              58 54 00' 08CA                .Ascic  "TX"    ; Enforced device name prefix
                    02  08CA
                    83  08CD                .BYTE   Pd$_Octal        ; Indicate scan octal
           52 53 43 00' 08CE                .ASCIC  "CSR"   ; CSR string
                    03  08CE
     0003FFFE 0003E000  08D2                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                    88  08DA                .BYTE   Pd$_Store        ; Indicate store operation
                  0032  08DB                .WORD   HP$L_DHUV_CSR           ; Byte HP$L_DHUV_CSR to data
                    00  08DD                .BYTE   0                ; 0 HP$L_DHUV_CSR to data
                    20  08DE                .BYTE   32               ; 32 of data field
                    88  08DF                .BYTE   Pd$_Store        ; Indicate store operation
                  0018  08E0                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  08E2                .BYTE   0                ; 0 HP$A_DEVICE to data
                    12  08E3                .BYTE   18               ; 18 of data field
                    83  08E4                .BYTE   Pd$_Octal        ; Indicate scan octal
  52 4F 54 43 45 56 00' 08E5                .ASCIC  "VECTOR"         ; VECTOR string
                    06  08E5
     000001FE 00000000  08EC                .LONG   ^O0,^O776        ; 0 , 776 limits on input
                    88  08F4                .BYTE   Pd$_Store        ; Indicate store operation
                  0036  08F5                .WORD   HP$W_DHUV_VEC           ; Byte HP$W_DHUV_VEC to data
                    00  08F7                .BYTE   0                ; 0 HP$W_DHUV_VEC to data
                    10  08F8                .BYTE   16               ; 16 of data field
                    82  08F9                .BYTE   Pd$_Decimal      ; Indicate scan decimal
              52 42 00' 08FA                .ASCIC  "BR"    ; BR string
                    02  08FA
     00000007 00000004  08FD                .LONG   4,7     ; 4 , 7 limits on input
                    88  0905                .BYTE   Pd$_Store        ; Indicate store operation
                  0038  0906                .WORD   HP$B_DHUV_BR            ; Byte HP$B_DHUV_BR to data
                    00  0908                .BYTE   0                ; 0 HP$B_DHUV_BR to data
                    08  0909                .BYTE   8                ; 8 of data field
                    81  090A                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                        090B   678 DHV11:   $DS_DHV11                        ;       [58]
     31 31 56 48 44 00' 090B                .ASCIC  "DHV11" ; DHV11 name
                    05  090B
                    39  0911                .BYTE   HP$K_DHUV_LEN           ; HP$K_DHUV_LEN of resultant P-table
                    08  0912                .BYTE   8                ; Maximum unit number                   ;G:65
                  5854  0913                .WORD   ^A"TX"  ; Two character mnemonic for QIO DHV11 TX
                    80  0915                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                    8D  0916                .Byte   Pd$_Name         ; Directive op-code
                    02  0917                .Byte   Ptd$M_Controller                ; Enforced TX codes
              58 54 00' 0918                .Ascic  "TX"    ; Enforced device name prefix
                    02  0918
                    83  091B                .BYTE   Pd$_Octal        ; Indicate scan octal
```

```
                52 53 43 00' 091C                       .ASCIC  "CSR"   ; CSR string
                          03 091C
             0003FFFE 0003E000 0920                     .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88 0928                     .BYTE   Pd$_Store       ; Indicate store operation
                        0032 0929                     .WORD   HP$L_DHUV_CSR           ; Byte HP$L_DHUV_CSR to data
                          00 092B                     .BYTE   0               ; 0 HP$L_DHUV_CSR to data
                          20 092C                     .BYTE   32              ; 32 of data field
                          88 092D                     .BYTE   Pd$_Store       ; Indicate store operation
                        0018 092E                     .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00 0930                     .BYTE   0               ; 0 HP$A_DEVICE to data
                          12 0931                     .BYTE   18              ; 18 of data field
                          83 0932                     .BYTE   Pd$_Octal       ; Indicate scan octal
          52 4F 54 43 45 56 00' 0933                  .ASCIC  "VECTOR"        ; VECTOR string
                          06 0933
             000001FE 00000000 093A                     .LONG   ^O0,^O776       ; 0 , 776 limits on input
                          88 0942                     .BYTE   Pd$_Store       ; Indicate store operation
                        0036 0943                     .WORD   HP$W_DHUV_VEC           ; Byte HP$W_DHUV_VEC to data
                          00 0945                     .BYTE   0               ; 0 HP$W_DHUV_VEC to data
                          10 0946                     .BYTE   16              ; 16 of data field
                          82 0947                     .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 0948                      .ASCIC  "BR"    ; BR string
                          02 0948
             00000007 00000004 094B                     .LONG   4,7     ; 4 , 7 limits on input
                          88 0953                     .BYTE   Pd$_Store       ; Indicate store operation
                        0038 0954                     .WORD   HP$B_DHUV_BR            ; Byte HP$B_DHUV_BR to data
                          00 0956                     .BYTE   0               ; 0 HP$B_DHUV_BR to data
                          08 0957                     .BYTE   8               ; 8 of data field
                          81 0958                     .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                             0959     679 DISK:       $DS_DISK                                ;       [41]
                             0959                     .SAVE   LOCAL_BLOCK
                             0959                     .PSECT  $ABS$,ABS
                    00000032 0050                     .=HP$A_DEPENDENT
                             0032                     .IIF    NB,HP$K_DISK_LEN,       HP$K_DISK_LEN:
                             0032                     .IIF    NB,DISK$K_LEN,  DISK$K_LEN:
                    00000959 0959                     .RESTORE
             4B 53 49 44 00' 0959                     .ASCIC  "DISK"  ; DISK name
                          04 0959
                          32 095E                     .BYTE   DISK$K_LEN              ; DISK$K_LEN of resultant P-table
                          00 095F                     .BYTE   0                                                       ;G:77
                        0000 0960                     .WORD   ^A""    ; Two character mnemonic for QIO DISK
                          80 0962                     .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8E 0963                     .BYTE   PD$_EPB         ; EXTENDED P-TABLE BLOCK                ;G:77
                        0FFF 0964                     .WORD   4095                                                    ;G:77
                          8D 0966                     .Byte   Pd$_Name        ; Directive op-code
                          03 0967                     .Byte   Ptd$M_DEVICE            ; Enforced DU codes
                55 44 00' 0968                      .Ascic  "DU"    ; Enforced device name prefix
                          02 0968
                          81 096B                     .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                             096C     680 DL11:       $DS_DL11
                             096C                     .SAVE   LOCAL_BLOCK
                             096C                     .PSECT  $ABS$,ABS
                    00000032 0050                     .=HP$A_DEPENDENT
                             0032                     .IIF    NB,DL11$L_CSR,  DL11$L_CSR:
                    00000036 0032                     .IIF    NB,.BLKL,       .BLKL   1
                             0036                     .IIF    NB,DL11$B_BR,   DL11$B_BR:
                    00000037 0036                     .IIF    NB,.BLKB,       .BLKB   1
                             0037                     .IIF    NB,DL11$K_LEN,  DL11$K_LEN:
```

```
                    0000096C                .RESTORE
          31 31 4C 44 00' 096C              .ASCIC  "DL11"  ; DL11 name
                      04  096C
                      37  0971              .BYTE   DL11$K_LEN                  ; DL11$K_LEN of resultant P-table
                      00  0972              .BYTE   0                   ; Maximum unit number                       ;G:65
                    0000  0973              .WORD   ^A""    ; Two character mnemonic for QIO DL11
                      80  0975              .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  0976              .Byte   Pd$_Name            ; Directive op-code
                      02  0977              .Byte   Ptd$M_Controller                ; Enforced TT codes
                54 54 00' 0978              .Ascic  "TT"    ; Enforced device name prefix
                      02  0978
                      83  097B              .BYTE   Pd$_Octal           ; Indicate scan octal
             52 53 43 00' 097C              .ASCIC  "CSR"   ; CSR string
                      03  097C
          0003FFFE 0003E000  0980           .LONG   ^O760000,^O777776           ; 760000 , 777776 limits on input
                      88  0988              .BYTE   Pd$_Store           ; Indicate store operation
                    0032  0989              .WORD   DL11$L_CSR                  ; Byte DL11$L_CSR to data
                      00  098B              .BYTE   0                   ; 0 DL11$L_CSR to data
                      20  098C              .BYTE   32                  ; 32 of data field
                      88  098D              .BYTE   Pd$_Store           ; Indicate store operation
                    0018  098E              .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      00  0990              .BYTE   0                   ; 0 HP$A_DEVICE to data
                      12  0991              .BYTE   18                  ; 18 of data field
                      83  0992              .BYTE   Pd$_Octal           ; Indicate scan octal
       52 4F 54 43 45 56 00' 0993           .ASCIC  "VECTOR"            ; VECTOR string
                      06  0993
          000001FE 00000002  099A           .LONG   ^O2,^O776           ; 2 , 776 limits on input
                      88  09A2              .BYTE   Pd$_Store           ; Indicate store operation
                    0024  09A3              .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      00  09A5              .BYTE   0                   ; 0 HP$W_VECTOR to data
                      09  09A6              .BYTE   9                   ; 9 of data field
                      82  09A7              .BYTE   Pd$_Decimal         ; Indicate scan decimal
                52 42 00' 09A8              .ASCIC  "BR"    ; BR string
                      02  09A8
          00000007 00000004  09AB           .LONG   4,7     ; 4 , 7 limits on input
                      88  09B3              .BYTE   Pd$_Store           ; Indicate store operation
                    0036  09B4              .WORD   DL11$B_BR                   ; Byte DL11$B_BR to data
                      00  09B6              .BYTE   0                   ; 0 DL11$B_BR to data
                      08  09B7              .BYTE   8                   ; 8 of data field
                      81  09B8              .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          09B9  681 DMB32:  $DS_DMB32                                                           ;G:3
                          09B9              .SAVE   LOCAL_BLOCK
                          09B9              .PSECT  $ABS$,ABS
                00000032  0050              .=HP$A_DEPENDENT
                          0032              .IIF    NB,HP$B_DMB32_NODE_ID,  HP$B_DMB32_NODE_ID::
                00000033  0032              .IIF    NB,,BLKB,       .BLKB   1
                          0033              .IIF    NB,HP$K_DMB32_LEN,      HP$K_DMB32_LEN::
                    000009B9                .RESTORE
          32 33 42 4D 44 00' 09B9           .ASCIC  "DMB32" ; DMB32 name
                      05  09B9
                      33  09BF              .BYTE   HP$K_DMB32_LEN              ; HP$K_DMB32_LEN of resultant P-table
                      00  09C0              .BYTE   0                   ; Maximum unit number                       ;G:65
                    0000  09C1              .WORD   ^A""    ; Two character mnemonic for QIO DMB32
                      80  09C3              .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      87  09C4              .BYTE   Pd$_Fetch           ; Indicate fetch operation
                    0018  09C5              .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      19  09C7              .BYTE   25                  ; 25 HP$A_DEVICE to data
```

```
                                07  09C8                .BYTE   7                       ; 7 of data field
                                8C  09C9                .BYTE   Pd$_Case                ; Indicate case operation
                                01  09CA                .BYTE   $$_$                    ; Count of pairs
                  00000030 00000000  09CB                .LONG   0,^X30                  ; Store the pair
                                88  09D3                .BYTE   Pd$_Store               ; Indicate store operation
                              0018  09D4                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                                19  09D6                .BYTE   25                      ; 25 HP$A_DEVICE to data
                                07  09D7                .BYTE   7                       ; 7 of data field
                                88  09D8                .BYTE   Pd$_Store               ; Indicate store operation
                              001C  09D9                .WORD   HP$A_DVA                        ; Byte HP$A_DVA to data
                                19  09DB                .BYTE   25                      ; 25 HP$A_DVA to data
                                07  09DC                .BYTE   7                       ; 7 of data field
                                86  09DD                .BYTE   Pd$_Literal             ; Indicate literal
                          00000001  09DE                .LONG   ^X1                     ; Constant
                                88  09E2                .BYTE   Pd$_Store               ; Indicate store operation
                              001C  09E3                .WORD   HP$A_DVA                        ; Byte HP$A_DVA to data
                                16  09E5                .BYTE   22                      ; 22 HP$A_DVA to data
                                01  09E6                .BYTE   1                       ; 1 of data field
                                88  09E7                .BYTE   Pd$_Store               ; Indicate store operation
                              0024  09E8                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                                08  09EA                .BYTE   8                       ; 8 HP$W_VECTOR to data
                                01  09EB                .BYTE   1                       ; 1 of data field
                                84  09EC                .BYTE   Pd$_Hexadecimal         ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'  09ED                .ASCIC  "BI Node Number (HEX) " ; BI Node Number (HEX)  string
      20 29 58 45 48 28 20 72 65 62   09F9
                                15  09ED
                  0000000F 00000000  0A03                .LONG   ^X0,^XF ; 0 , F limits on input
                                88  0A0B                .BYTE   Pd$_Store               ; Indicate store operation
                              0032  0A0C                .WORD   HP$B_DMB32_NODE_ID                      ; Byte HP$B_DMB32_NODE_ID to data
                                00  0A0E                .BYTE   0                       ; 0 HP$B_DMB32_NODE_ID to data
                                08  0A0F                .BYTE   8                       ; 8 of data field
                                88  0A10                .BYTE   Pd$_Store               ; Indicate store operation
                              0018  0A11                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                                0D  0A13                .BYTE   13                      ; 13 HP$A_DEVICE to data
                                04  0A14                .BYTE   4                       ; 4 of data field
                                88  0A15                .BYTE   Pd$_Store               ; Indicate store operation
                              0024  0A16                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                                02  0A18                .BYTE   2                       ; 2 HP$W_VECTOR to data
                                04  0A19                .BYTE   4                       ; 4 of data field
                                88  0A1A                .BYTE   Pd$_Store               ; Indicate store operation
                              001C  0A1B                .WORD   HP$A_DVA                        ; Byte HP$A_DVA to data
                                12  0A1D                .BYTE   18                      ; 18 HP$A_DVA to data
                                04  0A1E                .BYTE   4                       ; 4 of data field
                                81  0A1F                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                                    0A20    682 DMC11:  $DS_DMC11
                                    0A20                .SAVE   LOCAL_BLOCK
                                    0A20                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,DMC11$L_CSR, DMC11$L_CSR:
                          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                    0036                .IIF    NB,DMC11$B_BR,  DMC11$B_BR:
                          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                    0037                .IIF    NB,DMC11$K_LEN, DMC11$K_LEN:
                                00000A20                .RESTORE
                  31 31 43 4D 44 00'  0A20              .ASCIC  "DMC11" ; DMC11 name
                                05  0A20
                                37  0A26                .BYTE   DMC11$K_LEN             ; DMC11$K_LEN of resultant P-table
```

```
                    00  0A27          .BYTE   0                   ; Maximum unit number                          ;G:65
                  4D58  0A28          .WORD   ^A"XM"  ; Two character mnemonic for QIO DMC11 XM
                    80  0A2A          .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                    8D  0A2B          .Byte   Pd$_Name            ; Directive op-code
                    03  0A2C          .Byte   Ptd$M_Device                ; Enforced XM codes
              4D 58 00' 0A2D          .Ascic  "XM"    ; Enforced device name prefix
                    02  0A2D
                    83  0A30          .BYTE   Pd$_Octal           ; Indicate scan octal
           52 53 43 00' 0A31          .ASCIC  "CSR"   ; CSR string
                    03  0A31
     0003FFFE 0003E000  0A35          .LONG   ^O760000,^O777776           ; 760000 , 777776 limits on input
                    88  0A3D          .BYTE   Pd$_Store           ; Indicate store operation
                  0032  0A3E          .WORD   DMC11$L_CSR                 ; Byte DMC11$L_CSR to data
                    00  0A40          .BYTE   0                   ; 0 DMC11$L_CSR to data
                    20  0A41          .BYTE   32                  ; 32 of data field
                    88  0A42          .BYTE   Pd$_Store           ; Indicate store operation
                  0018  0A43          .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                    00  0A45          .BYTE   0                   ; 0 HP$A_DEVICE to data
                    12  0A46          .BYTE   18                  ; 18 of data field
                    83  0A47          .BYTE   Pd$_Octal           ; Indicate scan octal
  52 4F 54 43 45 56 00' 0A48          .ASCIC  "VECTOR"            ; VECTOR string
                    06  0A48
     000001FE 00000002  0A4F          .LONG   ^O2,^O776           ; 2 , 776 limits on input
                    88  0A57          .BYTE   Pd$_Store           ; Indicate store operation
                  0024  0A58          .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                    00  0A5A          .BYTE   0                   ; 0 HP$W_VECTOR to data
                    09  0A5B          .BYTE   9                   ; 9 of data field
                    82  0A5C          .BYTE   Pd$_Decimal         ; Indicate scan decimal
              52 42 00' 0A5D          .ASCIC  "BR"    ; BR string
                    02  0A5D
     00000007 00000004  0A60          .LONG   4,7     ; 4 , 7 limits on input
                    88  0A68          .BYTE   Pd$_Store           ; Indicate store operation
                  0036  0A69          .WORD   DMC11$B_BR                  ; Byte DMC11$B_BR to data
                    00  0A6B          .BYTE   0                   ; 0 DMC11$B_BR to data
                    08  0A6C          .BYTE   8                   ; 8 of data field
                    81  0A6D          .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                        0A6E  683 DMF32:  $DS_DMF32
                        0A6E          .SAVE   LOCAL_BLOCK
                        0A6E          .PSECT  $ABS$,ABS
              00000032  0050          .=HP$A_DEPENDENT
                        0032          .IIF    NB,HP$B_DMF32_COMMON,   HP$B_DMF32_COMMON:
                        0032          .IIF    NB,DMF32$B_COMMON,      DMF32$B_COMMON:
              00000033  0032          .IIF    NB,.BLKB,       .BLKB   1
                        0033          .IIF    NB,HP$B_DMF32_FLAGS,    HP$B_DMF32_FLAGS:
                        0033          .IIF    NB,DMF32$B_FLAGS,       DMF32$B_FLAGS:
              00000034  0033          .IIF    NB,.BLKB,       .BLKB   1
                        0034          .IIF    NB,HP$B_DMF32_BR,       HP$B_DMF32_BR:
                        0034          .IIF    NB,DMF32$B_BR,  DMF32$B_BR:
              00000035  0034          .IIF    NB,.BLKB,       .BLKB   1
                        0035          .IIF    NB,HP$L_DMF32_CSR,      HP$L_DMF32_CSR:
                        0035          .IIF    NB,DMF32$L_CSR, DMF32$L_CSR:
              00000039  0035          .IIF    NB,.BLKL,       .BLKL   1
                        0039          .IIF    NB,HP$K_DMF32_LEN,      HP$K_DMF32_LEN:
                        0039          .IIF    NB,DMF32$K_LEN, DMF32$K_LEN:
                    00000A6E          .RESTORE
     32 33 46 4D 44 00' 0A6E          .ASCIC  "DMF32" ; DMF32 name
                    05  0A6E
```

```
                      39  0A74                .BYTE   DMF32$K_LEN             ; DMF32$K_LEN of resultant P-table
                      05  0A75                .BYTE   5                       ; Maximum unit number                   ;G:65
                    434C  0A76                .WORD   ^A"LC"  ; Two character mnemonic for QIO DMF32 LC
                      80  0A78                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      83  0A79                .BYTE   Pd$_Octal               ; Indicate scan octal
                52 53 43 00' 0A7A             .ASCIC  "CSR"   ; CSR string
                      03  0A7A
            0003FFFE 0003E000  0A7E           .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  0A86                .BYTE   Pd$_Store               ; Indicate store operation
                    0035  0A87                .WORD   DMF32$L_CSR                     ; Byte DMF32$L_CSR to data
                      00  0A89                .BYTE   0                       ; 0 DMF32$L_CSR to data
                      20  0A8A                .BYTE   32                      ; 32 of data field
                      88  0A8B                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  0A8C                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  0A8E                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  0A8F                .BYTE   18                      ; 18 of data field
                      83  0A90                .BYTE   Pd$_Octal               ; Indicate scan octal
         72 6F 74 63 65 56 00' 0A91           .ASCIC  "Vector"                ; Vector string
                      06  0A91
            000001FE 000000C0  0A98           .LONG   ^O300,^O776             ; 300 , 776 limits on input
                      88  0AA0                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  0AA1                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  0AA3                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  0AA4                .BYTE   9                       ; 9 of data field
                      82  0AA5                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                52 42 00' 0AA6                .ASCIC  "BR"    ; BR string
                      02  0AA6
            00000007 00000004  0AA9           .LONG   4,7     ; 4 , 7 limits on input
                      88  0AB1                .BYTE   Pd$_Store               ; Indicate store operation
                    0034  0AB2                .WORD   DMF32$B_BR                      ; Byte DMF32$B_BR to data
                      00  0AB4                .BYTE   0                       ; 0 DMF32$B_BR to data
                      08  0AB5                .BYTE   8                       ; 8 of data field
                      86  0AB6                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  0AB7                .LONG   1                       ; Constant
                      88  0ABB                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  0ABC                .WORD   DMF32$B_COMMON                  ; Byte DMF32$B_COMMON to data
                      00  0ABE                .BYTE   0                       ; 0 DMF32$B_COMMON to data
                      08  0ABF                .BYTE   8                       ; 8 of data field
                      81  0AC0                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          0AC1    684 DMF32A: $DS_DMF32A
                          0AC1                .SAVE   LOCAL_BLOCK
                          0AC1                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_DMF32A_BR,      HP$B_DMF32A_BR:
                          0032                .IIF    NB,DMF32A$B_BR, DMF32A$B_BR:
                00000033  0032                .IIF    NB,,BLKB,               .BLKB   1
                          0033                .IIF    NB,HP$B_DMF32A_ACTIV,   HP$B_DMF32A_ACTIV:
                          0033                .IIF    NB,DMF32A$B_ACTIV,      DMF32A$B_ACTIV:
                00000034  0033                .IIF    NB,,BLKB,               .BLKB   1
                          0034                .IIF    NB,HP$B_DMF32A_SPEED,   HP$B_DMF32A_SPEED:
                          0034                .IIF    NB,DMF32A$B_SPEED,      DMF32A$B_SPEED:
                00000035  0034                .IIF    NB,,BLKB,               .BLKB   1
                          0035                .IIF    NB,HP$W_DMF32A_FLAGS,   HP$W_DMF32A_FLAGS:
                          0035                .IIF    NB,DMF32A$W_FLAGS,      DMF32A$W_FLAGS:
                00000037  0035                .IIF    NB,,BLKW,               .BLKW   1
                          0037                .IIF    NB,HP$B_DMF32A_JUMP,    HP$B_DMF32A_JUMP:
                          0037                .IIF    NB,DMF32A$B_JUMP,       DMF32A$B_JUMP:
```

```
                    00000038  0037              .IIF    NB,.BLKB,       .BLKB   1
                              0038              .IIF    NB,HP$K_DMF32A_LEN,     HP$K_DMF32A_LEN:
                              0038              .IIF    NB,DMF32A$K_LEN,        DMF32A$K_LEN:
                          00000AC1              .RESTORE
                 41 32 33 46 4D 44 00' 0AC1     .ASCIC  "DMF32A"        ; DMF32A name
                              06  0AC1
                              38  0AC8          .BYTE   DMF32A$K_LEN            ; DMF32A$K_LEN of resultant P-table
                              00  0AC9          .BYTE   0                       ; Maximum unit number                   ;G:65
                            5854  0ACA          .WORD   ^A"TX"  ; Two character mnemonic for QIO DMF32A TX
                              80  0ACC          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              8D  0ACD          .Byte   Pd$_Name        ; Directive op-code
                              02  0ACE          .Byte   Ptd$M_Controller                ; Enforced TX codes
                       58 54 00' 0ACF          .Ascic  "TX"    ; Enforced device name prefix
                              02  0ACF
                              83  0AD2          .BYTE   Pd$_Octal       ; Indicate scan octal
                    52 53 43 00' 0AD3           .ASCIC  "CSR"   ; CSR string
                              03  0AD3
                0003FFFE 0003E000  0AD7         .LONG   ^O00760000,^O00777776     ; 0760000 , 0777776 limits on input
                              88  0ADF          .BYTE   Pd$_Store       ; Indicate store operation
                            0018  0AE0          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              00  0AE2          .BYTE   0                       ; 0 HP$A_DEVICE to data
                              12  0AE3          .BYTE   18                      ; 18 of data field
                              83  0AE4          .BYTE   Pd$_Octal       ; Indicate scan octal
             72 6F 74 63 65 56 00' 0AE5         .ASCIC  "Vector"        ; Vector string
                              06  0AE5
                000001FE 000000C0  0AEC         .LONG   ^O300,^O776     ; 300 , 776 limits on input
                              88  0AF4          .BYTE   Pd$_Store       ; Indicate store operation
                            0024  0AF5          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                              00  0AF7          .BYTE   0                       ; 0 HP$W_VECTOR to data
                              09  0AF8          .BYTE   9                       ; 9 of data field
                              82  0AF9          .BYTE   Pd$_Decimal     ; Indicate scan decimal
                          52 42 00' 0AFA        .ASCIC  "BR"    ; BR string
                              02  0AFA
                00000007 00000004  0AFD         .LONG   4,7     ; 4 , 7 limits on input
                              88  0B05          .BYTE   Pd$_Store       ; Indicate store operation
                            0032  0B06          .WORD   DMF32A$B_BR             ; Byte DMF32A$B_BR to data
                              00  0B08          .BYTE   0                       ; 0 DMF32A$B_BR to data
                              08  0B09          .BYTE   8                       ; 8 of data field
                              83  0B0A          .BYTE   Pd$_Octal       ; Indicate scan octal
65 6E 69 4C 20 65 76 69 74 63 41 00' 0B0B       .ASCIC  "Active Lines"  ; Active Lines string
                              73  0B17
                              0C  0B0B
                000000FF 00000001  0B18         .LONG   ^O1,^O377       ; 1 , 377 limits on input
                              88  0B20          .BYTE   Pd$_Store       ; Indicate store operation
                            0033  0B21          .WORD   DMF32A$B_ACTIV          ; Byte DMF32A$B_ACTIV to data
                              00  0B23          .BYTE   0                       ; 0 DMF32A$B_ACTIV to data
                              08  0B24          .BYTE   8                       ; 8 of data field
                              85  0B25          .BYTE   Pd$_String      ; Indicate scan and verify string
          65 74 61 52 20 64 75 61 42 00' 0B26   .ASCIC  "Baud Rate"     ; Baud Rate string
                              09  0B26
                          30 35 00' 0B30        .ASCIC  "50"
                              02  0B30
                          35 37 00' 0B33        .ASCIC  "75"
                              02  0B33
                       30 31 31 00' 0B36        .ASCIC  "110"
                              03  0B36
                    35 2E 34 33 31 00' 0B3A     .ASCIC  "134.5"
```

```
                        05  0B3A
                30 35 31 00' 0B40               .ASCIC  "150"
                        03  0B40
                30 30 33 00' 0B44               .ASCIC  "300"
                        03  0B44
                30 30 36 00' 0B48               .ASCIC  "600"
                        03  0B48
             30 30 32 31 00' 0B4C               .ASCIC  "1200"
                        04  0B4C
             30 30 38 31 00' 0B51               .ASCIC  "1800"
                        04  0B51
             30 30 30 32 00' 0B56               .ASCIC  "2000"
                        04  0B56
             30 30 34 32 00' 0B5B               .ASCIC  "2400"
                        04  0B5B
             30 30 36 33 00' 0B60               .ASCIC  "3600"
                        04  0B60
             30 30 38 34 00' 0B65               .ASCIC  "4800"
                        04  0B65
             30 30 32 37 00' 0B6A               .ASCIC  "7200"
                        04  0B6A
             30 30 36 39 00' 0B6F               .ASCIC  "9600"
                        04  0B6F
          30 30 32 39 31 00' 0B74               .ASCIC  "19200"
                        05  0B74
                        00  0B7A               .BYTE   0               ; Null length is end of valid 50,75,110,134.5,150,30
                        88  0B7B               .BYTE   Pd$_Store       ; Indicate store operation
                      0034  0B7C               .WORD   DMF32A$B_SPEED          ; Byte DMF32A$B_SPEED to data
                        00  0B7E               .BYTE   0               ; 0 DMF32A$B_SPEED to data
                        08  0B7F               .BYTE   8               ; 8 of data field
                        85  0B80               .BYTE   Pd$_String      ; Indicate scan and verify string
79 54 20 6B 63 61 62 70 6F 6F 4C 00' 0B81       .ASCIC  "Loopback Type" ; Loopback Type string
                     65 70  0B8D
                        0D  0B81
          4C 41 4E 52 45 54 4E 49 00' 0B8F       .ASCIC  "INTERNAL"
                        08  0B8F
                38 34 32 33 48 00' 0B98         .ASCIC  "H3248"
                        05  0B98
4D 45 44 4F 4D 5F 4C 41 43 4F 4C 00' 0B9E       .ASCIC  "LOCAL_MODEM"
                        0B  0B9E
4C 42 41 4D 4D 41 52 47 4F 52 50 00' 0BAA       .ASCIC  "PROGRAMMABLE_MODEM"
             4D 45 44 4F 4D 5F 45   0BB6
                        12  0BAA
                39 34 32 33 48 00' 0BBD         .ASCIC  "H3249"
                        05  0BBD
                        00  0BC3               .BYTE   0               ; Null length is end of valid INTERNAL,<H3248>,LOCAL
                        88  0BC4               .BYTE   Pd$_Store       ; Indicate store operation
                      0035  0BC5               .WORD   DMF32A$W_FLAGS          ; Byte DMF32A$W_FLAGS to data
                        00  0BC7               .BYTE   0               ; 0 DMF32A$W_FLAGS to data
                        05  0BC8               .BYTE   5               ; 5 of data field
                        8B  0BC9               .BYTE   Pd$_Logical     ; Indicate logical value operation
74 69 6E 49 20 73 75 62 69 6E 55 00' 0BCA       .ASCIC  \Unibus Init Jumper\    ; Unibus Init Jumper string
             72 65 70 6D 75 4A 20   0BD6
                        12  0BCA
                        88  0BDD               .BYTE   Pd$_Store       ; Indicate store operation
                      0037  0BDE               .WORD   DMF32A$B_JUMP           ; Byte DMF32A$B_JUMP to data
                        00  0BE0               .BYTE   0               ; 0 DMF32A$B_JUMP to data
```

```
                      01  0BE1                .BYTE   1                       ; 1 of data field
                      81  0BE2                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          0BE3    685 DMF32P: $DS_DMF32P
                          0BE3                .SAVE   LOCAL_BLOCK
                          0BE3                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_DMF32P_COMMON,  HP$B_DMF32P_COMMON:
                          0032                .IIF    NB,DMF32P$B_COMMON,     DMF32P$B_COMMON:
                00000033  0032                .IIF    NB,,.BLKB,        .BLKB 1
                          0033                .IIF    NB,HP$B_DMF32P_FLAGS,   HP$B_DMF32P_FLAGS:
                          0033                .IIF    NB,DMF32P$B_FLAGS,      DMF32P$B_FLAGS:
                00000034  0033                .IIF    NB,,.BLKB,        .BLKB 1
                          0034                .IIF    NB,HP$B_DMF32P_BR,      HP$B_DMF32P_BR:
                          0034                .IIF    NB,DMF32P$B_BR, DMF32P$B_BR:
                00000035  0034                .IIF    NB,,.BLKB,        .BLKB 1
                          0035                .IIF    NB,HP$L_DMF32P_CSR,     HP$L_DMF32P_CSR:
                          0035                .IIF    NB,DMF32P$L_CSR,        DMF32P$L_CSR:
                00000039  0035                .IIF    NB,,.BLKL,        .BLKL 1
                          0039                .IIF    NB,HP$K_DMF32P_LEN,     HP$K_DMF32P_LEN:
                          0039                .IIF    NB,DMF32P$K_LEN,        DMF32P$K_LEN:
                      00000BE3                .RESTORE
           50 32 33 46 4D 44 00' 0BE3         .ASCIC  "DMF32P"                ; DMF32P name
                      06  0BE3
                      39  0BEA                .BYTE   DMF32P$K_LEN                    ; DMF32P$K_LEN of resultant P-table
                      00  0BEB                .BYTE   0                       ; Maximum unit number                   ;G:65
                    434C  0BEC                .WORD   ^A"LC"  ; Two character mnemonic for QIO DMF32P LC
                      80  0BEE                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  0BEF                .Byte   Pd$_Name                ; Directive op-code
                      02  0BF0                .Byte   Ptd$M_Controller                ; Enforced LC codes
                 43 4C 00' 0BF1                .Ascic  "LC"    ; Enforced device name prefix
                      02  0BF1
                      83  0BF4                .BYTE   Pd$_Octal               ; Indicate scan octal
              52 53 43 00' 0BF5                .ASCIC  "CSR"   ; CSR string
                      03  0BF5
          0003FFFE 0003E000  0BF9                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  0C01                .BYTE   Pd$_Store               ; Indicate store operation
                    0035  0C02                .WORD   DMF32P$L_CSR                    ; Byte DMF32P$L_CSR to data
                      00  0C04                .BYTE   0                       ; 0 DMF32P$L_CSR to data
                      20  0C05                .BYTE   32                      ; 32 of data field
                      88  0C06                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  0C07                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  0C09                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  0C0A                .BYTE   18                      ; 18 of data field
                      83  0C0B                .BYTE   Pd$_Octal               ; Indicate scan octal
        72 6F 74 63 65 56 00' 0C0C            .ASCIC  "Vector"                ; Vector string
                      06  0C0C
          000001FE 000000C0  0C13                .LONG   ^O300,^O776     ; 300 , 776 limits on input
                      88  0C1B                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  0C1C                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  0C1E                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  0C1F                .BYTE   9                       ; 9 of data field
                      82  0C20                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                 52 42 00' 0C21                .ASCIC  "BR"    ; BR string
                      02  0C21
          00000007 00000004  0C24                .LONG   4,7     ; 4 , 7 limits on input
                      88  0C2C                .BYTE   Pd$_Store               ; Indicate store operation
                    0034  0C2D                .WORD   DMF32P$B_BR                     ; Byte DMF32P$B_BR to data
```

```
                                00  0C2F                .BYTE   0               ; 0 DMF32P$B_BR to data
                                08  0C30                .BYTE   8               ; 8 of data field
                                85  0C31                .BYTE   Pd$_String      ; Indicate scan and verify string
65 63 69 76 65 44 20 74 72 6F 50 00' 0C32               .ASCIC  "Port Device"   ; Port Device string
                                0B  0C32
                   52 45 48 54 4F 00' 0C3E              .ASCIC  "OTHER"
                                05  0C3E
                         50 4C 00' 0C44                 .ASCIC  "LP"
                                02  0C44
                      50 41 52 57 00' 0C47              .ASCIC  "WRAP"
                                04  0C47
                                00  0C4C                .BYTE   0               ; Null length is end of valid OTHER,LP,WRAP
                                88  0C4D                .BYTE   Pd$_Store       ; Indicate store operation
                              0033  0C4E                .WORD   DMF32P$B_FLAGS          ; Byte DMF32P$B_FLAGS to data
                                00  0C50                .BYTE   0               ; 0 DMF32P$B_FLAGS to data
                                08  0C51                .BYTE   8               ; 8 of data field
                                86  0C52                .BYTE   Pd$_Literal     ; Indicate literal
                          00000001  0C53                .LONG   1               ; Constant
                                88  0C57                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  0C58                .WORD   DMF32P$B_COMMON         ; Byte DMF32P$B_COMMON to data
                                00  0C5A                .BYTE   0               ; 0 DMF32P$B_COMMON to data
                                08  0C5B                .BYTE   8               ; 8 of data field
                                81  0C5C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    0C5D    686 DMF32S: $DS_DMF32S
                                    0C5D                .SAVE   LOCAL_BLOCK
                                    0C5D                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,HP$B_DMF32S_COMMON,  HP$B_DMF32S_COMMON:
                                    0032                .IIF    NB,DMF32S$B_COMMON,     DMF32S$B_COMMON:
                          00000033  0032                .IIF    NB,,.BLKB,      .BLKB   1
                                    0033                .IIF    NB,HP$B_DMF32S_FLAGS,   HP$B_DMF32S_FLAGS:
                                    0033                .IIF    NB,DMF32S$B_FLAGS,      DMF32S$B_FLAGS:
                          00000034  0033                .IIF    NB,,.BLKB,      .BLKB   1
                                    0034                .IIF    NB,HP$B_DMF32S_BR,      HP$B_DMF32S_BR:
                                    0034                .IIF    NB,DMF32S$B_BR, DMF32S$B_BR:
                          00000035  0034                .IIF    NB,,.BLKB,      .BLKB   1
                                    0035                .IIF    NB,HP$L_DMF32S_CSR,     HP$L_DMF32S_CSR:
                                    0035                .IIF    NB,DMF32S$L_CSR,        DMF32S$L_CSR:
                          00000039  0035                .IIF    NB,,.BLKL,      .BLKL   1
                                    0039                .IIF    NB,HP$K_DMF32S_LEN,     HP$K_DMF32S_LEN:
                                    0039                .IIF    NB,DMF32S$K_LEN,        DMF32S$K_LEN:
                                00000C5D                .RESTORE
                   53 32 33 46 4D 44 00' 0C5D           .ASCIC  "DMF32S"        ; DMF32S name
                                06  0C5D
                                39  0C64                .BYTE   DMF32$K_LEN             ; DMF32$K_LEN of resultant P-table
                                01  0C65                .BYTE   1               ; Maximum unit number                   ;G:65
                             4758  0C66                .WORD   ^A"XG"  ; Two character mnemonic for QIO DMF32S XG
                                80  0C68                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  0C69                .Byte   Pd$_Name        ; Directive op-code
                                03  0C6A                .Byte   Ptd$M_Device            ; Enforced XG codes
                         47 58 00' 0C6B                .Asic   "XG"    ; Enforced device name prefix
                                02  0C6B
                                83  0C6E                .BYTE   Pd$_Octal       ; Indicate scan octal
                      52 53 43 00' 0C6F                 .ASCIC  "CSR"   ; CSR string
                                03  0C6F
                    0003FFFE 0003E000  0C73             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                88  0C7B                .BYTE   Pd$_Store       ; Indicate store operation
```

```
                    0035  0C7C                .WORD   DMF32S$L_CSR            ; Byte DMF32S$L_CSR to data
                      00  0C7E                .BYTE   0                   ; 0 DMF32S$L_CSR to data
                      20  0C7F                .BYTE   32                  ; 32 of data field
                      88  0C80                .BYTE   Pd$_Store           ; Indicate store operation
                    0018  0C81                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      00  0C83                .BYTE   0                   ; 0 HP$A_DEVICE to data
                      12  0C84                .BYTE   18                  ; 18 of data field
                      83  0C85                .BYTE   Pd$_Octal           ; Indicate scan octal
          72 6F 74 63 65 56 00' 0C86          .ASCIC  "Vector"            ; Vector string
                      06  0C86
             000001FE 00000002  0C8D          .LONG   ^O2,^O776           ; 2 , 776 limits on input
                      88  0C95                .BYTE   Pd$_Store           ; Indicate store operation
                    0024  0C96                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      00  0C98                .BYTE   0                   ; 0 HP$W_VECTOR to data
                      09  0C99                .BYTE   9                   ; 9 of data field
                      82  0C9A                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                   52 42 00' 0C9B             .ASCIC  "BR"    ; BR string
                      02  0C9B
             00000007 00000004  0C9E          .LONG   4,7     ; 4 , 7 limits on input
                      88  0CA6                .BYTE   Pd$_Store           ; Indicate store operation
                    0034  0CA7                .WORD   DMF32S$B_BR                 ; Byte DMF32S$B_BR to data
                      00  0CA9                .BYTE   0                   ; 0 DMF32S$B_BR to data
                      08  0CAA                .BYTE   8                   ; 8 of data field
                      85  0CAB                .BYTE   Pd$_String          ; Indicate scan and verify string
72 57 20 6C 61 6E 72 65 74 78 45 00' 0CAC     .ASCIC  "External Wrap" ; External Wrap string
                   70 61  0CB8
                      0D  0CAC
                45 4E 4F 4E 00' 0CBA          .ASCIC  "NONE"
                      04  0CBA
             4D 45 44 4F 4D 00' 0CBF          .ASCIC  "MODEM"
                      05  0CBF
                50 41 52 57 00' 0CC5          .ASCIC  "WRAP"
                      04  0CC5
                      00  0CCA                .BYTE   0                   ; Null length is end of valid NONE,MODEM,WRAP
                      88  0CCB                .BYTE   Pd$_Store           ; Indicate store operation
                    0033  0CCC                .WORD   DMF32S$B_FLAGS              ; Byte DMF32S$B_FLAGS to data
                      00  0CCE                .BYTE   0                   ; 0 DMF32S$B_FLAGS to data
                      02  0CCF                .BYTE   2                   ; 2 of data field
                      81  0CD0                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          0CD1    687 DMP11:  $DS_DMP11
                          0CD1                .SAVE   LOCAL_BLOCK
                          0CD1                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DMP11$B_COMMON,      DMP11$B_COMMON:
                 00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,DMP11$B_FLAGS,       DMP11$B_FLAGS:
                 00000034 0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,DMP11$B_BR,  DMP11$B_BR:
                 00000035 0034                .IIF    NB,.BLKB,       .BLKB   1
                          0035                .IIF    NB,DMP11$L_CSR, DMP11$L_CSR:
                 00000039 0035                .IIF    NB,.BLKL,       .BLKL   1
                          0039                .IIF    NB,DMP11$K_LEN, DMP11$K_LEN:
                      00000CD1                .RESTORE
          31 31 50 4D 44 00' 0CD1             .ASCIC  "DMP11" ; DMP11 name
                      05  0CD1
                      39  0CD7                .BYTE   DMP11$K_LEN                 ; DMP11$K_LEN of resultant P-table
                      00  0CD8                .BYTE   0                   ; Maximum unit number                   ;G:65
```

```
                        4458  0CD9                .WORD   ^A"XD"   ; Two character mnemonic for QIO DMP11 XD
                          80  0CDB                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  0CDC                .Byte   Pd$_Name        ; Directive op-code
                          03  0CDD                .Byte   Ptd$M_Device            ; Enforced XD codes
                  44 58 00'   0CDE                .Ascic  "XD"    ; Enforced device name prefix
                          02  0CDE
                          83  0CE1                .BYTE   Pd$_Octal       ; Indicate scan octal
                52 53 43 00'  0CE2                .ASCIC  "CSR"   ; CSR string
                          03  0CE2
            0003FFFE 0003E000  0CE6                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  0CEE                .BYTE   Pd$_Store       ; Indicate store operation
                        0035  0CEF                .WORD   DMP11$L_CSR             ; Byte DMP11$L_CSR to data
                          00  0CF1                .BYTE   0               ; 0 DMP11$L_CSR to data
                          20  0CF2                .BYTE   32              ; 32 of data field
                          88  0CF3                .BYTE   Pd$_Store       ; Indicate store operation
                        0018  0CF4                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  0CF6                .BYTE   0               ; 0 HP$A_DEVICE to data
                          12  0CF7                .BYTE   18              ; 18 of data field
                          83  0CF8                .BYTE   Pd$_Octal       ; Indicate scan octal
          52 4F 54 43 45 56 00' 0CF9              .ASCIC  "VECTOR"        ; VECTOR string
                          06  0CF9
            000001FE 00000002  0D00                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  0D08                .BYTE   Pd$_Store       ; Indicate store operation
                        0024  0D09                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  0D0B                .BYTE   0               ; 0 HP$W_VECTOR to data
                          09  0D0C                .BYTE   9               ; 9 of data field
                          82  0D0D                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                  52 42 00'   0D0E                .ASCIC  "BR"    ; BR string
                          02  0D0E
                00000007 00000004  0D11           .LONG   4,7     ; 4 , 7 limits on input
                          88  0D19                .BYTE   Pd$_Store       ; Indicate store operation
                        0034  0D1A                .WORD   DMP11$B_BR              ; Byte DMP11$B_BR to data
                          00  0D1C                .BYTE   0               ; 0 DMP11$B_BR to data
                          08  0D1D                .BYTE   8               ; 8 of data field
                          85  0D1E                .BYTE   Pd$_String      ; Indicate scan and verify string
61 74 53 20 6C 6F 72 74 6E 6F 43 00' 0D1F         .ASCIC  "Control Station"       ; Control Station string
                   6E 6F 69 74  0D2B
                          0F  0D1F
                     4F 4E 00'  0D2F              .ASCIC  "NO"
                          02  0D2F
                  53 45 59 00'  0D32              .ASCIC  "YES"
                          03  0D32
                          00  0D36                .BYTE   0               ; Null length is end of valid NO,YES
                          88  0D37                .BYTE   Pd$_Store       ; Indicate store operation
                        0033  0D38                .WORD   DMP11$B_FLAGS           ; Byte DMP11$B_FLAGS to data
                          00  0D3A                .BYTE   0               ; 0 DMP11$B_FLAGS to data
                          01  0D3B                .BYTE   1               ; 1 of data field
                          85  0D3C                .BYTE   Pd$_String      ; Indicate scan and verify string
      79 72 61 74 75 62 69 72 54 00' 0D3D         .ASCIC  "Tributary"     ; Tributary string
                          09  0D3D
                     4F 4E 00'  0D47              .ASCIC  "NO"
                          02  0D47
                  53 45 59 00'  0D4A              .ASCIC  "YES"
                          03  0D4A
                          00  0D4E                .BYTE   0               ; Null length is end of valid NO,YES
                          88  0D4F                .BYTE   Pd$_Store       ; Indicate store operation
                        0033  0D50                .WORD   DMP11$B_FLAGS           ; Byte DMP11$B_FLAGS to data
```

```
                                  01  0D52                .BYTE   1                   ; 1 DMP11$B_FLAGS to data
                                  01  0D53                .BYTE   1                   ; 1 of data field
                                  85  0D54                .BYTE   Pd$_String          ; Indicate scan and verify string
69 6C 69 62 61 74 61 70 6D 6F 43 00' 0D55                .ASCIC  "Compatability Mode"    ; Compatability Mode string
              65 64 6F 4D 20 79 74  0D61
                                  12  0D55
                           4F 4E 00' 0D68                .ASCIC  "NO"
                                  02  0D68
                        53 45 59 00' 0D6B                .ASCIC  "YES"
                                  03  0D6B
                                  00  0D6F                .BYTE   0                   ; Null length is end of valid NO,YES
                                  88  0D70                .BYTE   Pd$_Store           ; Indicate store operation
                                0033  0D71                .WORD   DMP11$B_FLAGS               ; Byte DMP11$B_FLAGS to data
                                  02  0D73                .BYTE   2                   ; 2 DMP11$B_FLAGS to data
                                  01  0D74                .BYTE   1                   ; 1 of data field
                                  85  0D75                .BYTE   Pd$_String          ; Indicate scan and verify string
78 65 6C 70 75 44 20 66 6C 61 48 00' 0D76                .ASCIC  "Half Duplex"       ; Half Duplex string
                                  0B  0D76
                           4F 4E 00' 0D82                .ASCIC  "NO"
                                  02  0D82
                        53 45 59 00' 0D85                .ASCIC  "YES"
                                  03  0D85
                                  00  0D89                .BYTE   0                   ; Null length is end of valid NO,YES
                                  88  0D8A                .BYTE   Pd$_Store           ; Indicate store operation
                                0033  0D8B                .WORD   DMP11$B_FLAGS               ; Byte DMP11$B_FLAGS to data
                                  03  0D8D                .BYTE   3                   ; 3 DMP11$B_FLAGS to data
                                  01  0D8E                .BYTE   1                   ; 1 of data field
                                  86  0D8F                .BYTE   Pd$_Literal         ; Indicate literal
                          00000001  0D90                .LONG   1                   ; Constant
                                  88  0D94                .BYTE   Pd$_Store           ; Indicate store operation
                                0032  0D95                .WORD   DMP11$B_COMMON              ; Byte DMP11$B_COMMON to data
                                  00  0D97                .BYTE   0                   ; 0 DMP11$B_COMMON to data
                                  08  0D98                .BYTE   8                   ; 8 of data field
                                  81  0D99                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                                      0D9A   688 DMR11:   $DS_DMR11
                                      0D9A                .SAVE   LOCAL_BLOCK
                                      0D9A                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                      0032                .IIF    NB,DMR11$L_CSR, DMR11$L_CSR:
                          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                      0036                .IIF    NB,DMR11$B_BR,  DMR11$B_BR:
                          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                      0037                .IIF    NB,DMR11$K_LEN, DMR11$K_LEN:
                                00000D9A                .RESTORE
                31 31 52 4D 44 00' 0D9A                .ASCIC  "DMR11" ; DMR11 name
                                  05  0D9A
                                  37  0DA0                .BYTE   DMR11$K_LEN                 ; DMR11$K_LEN of resultant P-table
                                  00  0DA1                .BYTE   0                   ; Maximum unit number                    ;G:65
                                4D58  0DA2                .WORD   ^A"XM"  ; Two character mnemonic for QIO DMR11 XM
                                  80  0DA4                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                                  8D  0DA5                .Byte   Pd$_Name            ; Directive op-code
                                  03  0DA6                .Byte   Ptd$M_Device                ; Enforced XM codes
                           4D 58 00' 0DA7                .Ascic  "XM"    ; Enforced device name prefix
                                  02  0DA7
                                  83  0DAA                .BYTE   Pd$_Octal           ; Indicate scan octal
                        52 53 43 00' 0DAB                .ASCIC  "CSR"   ; CSR string
                                  03  0DAB
```

```
                0003FFFE 0003E000  0DAF                .LONG   ^O760000,^O777776      ; 760000 , 777776 limits on input
                              88   0DB7                .BYTE   Pd$_Store       ; Indicate store operation
                            0032   0DB8                .WORD   DMR11$L_CSR             ; Byte DMR11$L_CSR to data
                              00   0DBA                .BYTE   0               ; 0 DMR11$L_CSR to data
                              20   0DBB                .BYTE   32              ; 32 of data field
                              88   0DBC                .BYTE   Pd$_Store       ; Indicate store operation
                            0018   0DBD                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              00   0DBF                .BYTE   0               ; 0 HP$A_DEVICE to data
                              12   0DC0                .BYTE   18              ; 18 of data field
                              83   0DC1                .BYTE   Pd$_Octal       ; Indicate scan octal
             52 4F 54 43 45 56 00'  0DC2                .ASCIC  "VECTOR"        ; VECTOR string
                              06   0DC2
                000001FE 00000002  0DC9                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                              88   0DD1                .BYTE   Pd$_Store       ; Indicate store operation
                            0024   0DD2                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                              00   0DD4                .BYTE   0               ; 0 HP$W_VECTOR to data
                              09   0DD5                .BYTE   9               ; 9 of data field
                              82   0DD6                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                       52 42 00'  0DD7                .ASCIC  "BR"    ; BR string
                              02   0DD7
                00000007 00000004  0DDA                .LONG   4,7     ; 4 , 7 limits on input
                              88   0DE2                .BYTE   Pd$_Store       ; Indicate store operation
                            0036   0DE3                .WORD   DMR11$B_BR              ; Byte DMR11$B_BR to data
                              00   0DE5                .BYTE   0               ; 0 DMR11$B_BR to data
                              08   0DE6                .BYTE   8               ; 8 of data field
                              81   0DE7                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                   0DE8    689 DMZ32:   $DS_DMZ32                                      ;       [45]
                                   0DE8                .SAVE   LOCAL_BLOCK
                                   0DE8                .PSECT  $ABS$,ABS
                        00000032   0050                .=HP$A_DEPENDENT
                                   0032                .IIF    NB,HP$B_DMZ32_BR,       HP$B_DMZ32_BR:
                        00000033   0032                .IIF    NB,..BLKB,      .BLKB   1
                                   0033                .IIF    NB,HP$B_DMZ32_OCTET0,   HP$B_DMZ32_OCTET0:
                        00000034   0033                .IIF    NB,..BLKB,      .BLKB   1
                                   0034                .IIF    NB,HP$B_DMZ32_OCTET1,   HP$B_DMZ32_OCTET1:
                        00000035   0034                .IIF    NB,..BLKB,      .BLKB   1
                                   0035                .IIF    NB,HP$B_DMZ32_OCTET2,   HP$B_DMZ32_OCTET2:
                        00000036   0035                .IIF    NB,..BLKB,      .BLKB   1
                                   0036                .IIF    NB,HP$B_DMZ32_SPEED,    HP$B_DMZ32_SPEED:
                        00000037   0036                .IIF    NB,..BLKB,      .BLKB   1
                                   0037                .IIF    NB,HP$W_DMZ32_LOOP,     HP$W_DMZ32_LOOP:
                        00000039   0037                .IIF    NB,..BLKW,      .BLKW   1
                                   0039                .IIF    NB,HP$B_DMZ32_MODEM,    HP$B_DMZ32_MODEM:
                        0000003A   0039                .IIF    NB,..BLKB,      .BLKB   1
                                   003A                .IIF    NB,HP$K_DMZ32_LEN,      HP$K_DMZ32_LEN:
                                00000DE8               .RESTORE
          32 33 5A 4D 44 00'  0DE8                .ASCIC  "DMZ32" ; DMZ32 name
                              05   0DE8
                              3A   0DEE                .BYTE   HP$K_DMZ32_LEN          ; HP$K_DMZ32_LEN of resultant P-table
                              00   0DEF                .BYTE   0               ; Maximum unit number                   ;G:65
                            0000   0DF0                .WORD   ^A""    ; Two character mnemonic for QIO DMZ32
                              80   0DF2                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                              8D   0DF3                .Byte   Pd$_Name        ; Directive op-code
                              02   0DF4                .Byte   PTD$M_CONTROLLER                ; Enforced TX codes
                       58 54 00'  0DF5                .Ascic  "TX"    ; Enforced device name prefix
                              02   0DF5
                              83   0DF8                .BYTE   Pd$_Octal       ; Indicate scan octal
```

```
                        52 53 43 00'  0DF9                .ASCIC  "CSR"   ; CSR string
                              03      0DF9
                    0003FFFE 0003E000 0DFD                .LONG   ^O760000,^O00777776      ; 760000 , 0777776 limits on input
                              88      0E05                .BYTE   Pd$_Store       ; Indicate store operation
                            0018      0E06                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                              00      0E08                .BYTE   0               ; 0 HP$A_DEVICE to data
                              12      0E09                .BYTE   18              ; 18 of data field
                              83      0E0A                .BYTE   Pd$_Octal       ; Indicate scan octal
                  72 6F 74 63 65 56 00' 0E0B              .ASCIC  "Vector"        ; Vector string
                              06      0E0B
                    000001FE 000000C0 0E12                .LONG   ^O300,^O776     ; 300 , 776 limits on input
                              88      0E1A                .BYTE   Pd$_Store       ; Indicate store operation
                            0024      0E1B                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                              00      0E1D                .BYTE   0               ; 0 HP$W_VECTOR to data
                              09      0E1E                .BYTE   9               ; 9 of data field
                              82      0E1F                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                        52 42 00'     0E20                .ASCIC  "BR"    ; BR string
                              02      0E20
                    00000006 00000005 0E23                .LONG   5,6     ; 5 , 6 limits on input
                              88      0E2B                .BYTE   Pd$_Store       ; Indicate store operation
                            0032      0E2C                .WORD   HP$B_DMZ32_BR           ; Byte HP$B_DMZ32_BR to data
                              00      0E2E                .BYTE   0               ; 0 HP$B_DMZ32_BR to data
                              08      0E2F                .BYTE   8               ; 8 of data field
                              83      0E30                .BYTE   Pd$_Octal       ; Indicate scan octal
20 53 45 4E 49 4C 20 54 53 45 54 00'  0E31                .ASCIC  "TEST LINES (OCTET0)"   ; TEST LINES (OCTET0) string
      29 30 54 45 54 43 4F 28         0E3D
                              13      0E31
                    000000FF 00000000 0E45                .LONG   ^O0,^O0377      ; 0 , 377 limits on input
                              88      0E4D                .BYTE   Pd$_Store       ; Indicate store operation
                            0033      0E4E                .WORD   HP$B_DMZ32_OCTET0               ; Byte HP$B_DMZ32_OCTET0 to data
                              00      0E50                .BYTE   0               ; 0 HP$B_DMZ32_OCTET0 to data
                              08      0E51                .BYTE   8               ; 8 of data field
                              83      0E52                .BYTE   Pd$_Octal       ; Indicate scan octal
4C 4E 4F 20 33 20 4C 45 56 45 4C 00'  0E53                .ASCIC  "LEVEL 3 ONLY - TEST LINES (OCTET1)"    ; LEVEL 3 ONLY - TEST LINES
4E 49 4C 20 54 53 45 54 20 2D 20 59   0E5F
   29 31 54 45 54 43 4F 28 20 53 45   0E6B
                              22      0E53
                    000000FF 00000000 0E76                .LONG   ^O0,^O0377      ; 0 , 377 limits on input
                              88      0E7E                .BYTE   Pd$_Store       ; Indicate store operation
                            0034      0E7F                .WORD   HP$B_DMZ32_OCTET1               ; Byte HP$B_DMZ32_OCTET1 to data
                              00      0E81                .BYTE   0               ; 0 HP$B_DMZ32_OCTET1 to data
                              08      0E82                .BYTE   8               ; 8 of data field
                              83      0E83                .BYTE   Pd$_Octal       ; Indicate scan octal
4C 4E 4F 20 33 20 4C 45 56 45 4C 00'  0E84                .ASCIC  "LEVEL 3 ONLY - TEST LINES (OCTET2)"    ; LEVEL 3 ONLY - TEST LINES
4E 49 4C 20 54 53 45 54 20 2D 20 59   0E90
   29 32 54 45 54 43 4F 28 20 53 45   0E9C
                              22      0E84
                    000000FF 00000000 0EA7                .LONG   ^O0,^O0377      ; 0 , 377 limits on input
                              88      0EAF                .BYTE   Pd$_Store       ; Indicate store operation
                            0035      0EB0                .WORD   HP$B_DMZ32_OCTET2               ; Byte HP$B_DMZ32_OCTET2 to data
                              00      0EB2                .BYTE   0               ; 0 HP$B_DMZ32_OCTET2 to data
                              08      0EB3                .BYTE   8               ; 8 of data field
                              85      0EB4                .BYTE   Pd$_String      ; Indicate scan and verify string
                  65 74 61 52 20 64 75 61 42 00' 0EB5     .ASCIC  "Baud Rate"     ; Baud Rate string
                              09      0EB5
                        30 35 00'     0EBF                .ASCIC  "50"
                              02      0EBF
```

```
                    35 37 00' 0EC2              .ASCIC  "75"
                          02  0EC2
                 30 31 31 00' 0EC5              .ASCIC  "110"
                          03  0EC5
              35 2E 34 33 31 00' 0EC9           .ASCIC  "134.5"
                          05  0EC9
                 30 35 31 00' 0ECF              .ASCIC  "150"
                          03  0ECF
                 30 30 33 00' 0ED3              .ASCIC  "300"
                          03  0ED3
                 30 30 36 00' 0ED7              .ASCIC  "600"
                          03  0ED7
              30 30 32 31 00' 0EDB              .ASCIC  "1200"
                          04  0EDB
              30 30 38 31 00' 0EE0              .ASCIC  "1800"
                          04  0EE0
              30 30 30 32 00' 0EE5              .ASCIC  "2000"
                          04  0EE5
              30 30 34 32 00' 0EEA              .ASCIC  "2400"
                          04  0EEA
              30 30 38 34 00' 0EEF              .ASCIC  "4800"
                          04  0EEF
              30 30 36 39 00' 0EF4              .ASCIC  "9600"
                          04  0EF4
           30 30 32 39 31 00' 0EF9              .ASCIC  "19200"
                          05  0EF9
                          00  0EFF              .BYTE   0               ; Null length is end of valid 50,75,110,134.5,150,30
                          88  0F00              .BYTE   Pd$_Store       ; Indicate store operation
                        0036  0F01              .WORD   HP$B_DMZ32_SPEED                ; Byte HP$B_DMZ32_SPEED to data
                          00  0F03              .BYTE   0               ; 0 HP$B_DMZ32_SPEED to data
                          08  0F04              .BYTE   8               ; 8 of data field
                          85  0F05              .BYTE   Pd$_String      ; Indicate scan and verify string
79 54 20 6B 63 61 62 70 6F 6F 4C 00' 0F06       .ASCIC  "Loopback Type" ; Loopback Type string
                       65 70  0F12
                          0D  0F06
                       30 00' 0F14              .ASCIC  "0"
                          01  0F14
                       31 00' 0F16              .ASCIC  "1"
                          01  0F16
                       32 00' 0F18              .ASCIC  "2"
                          01  0F18
                       33 00' 0F1A              .ASCIC  "3"
                          01  0F1A
                       34 00' 0F1C              .ASCIC  "4"
                          01  0F1C
                       35 00' 0F1E              .ASCIC  "5"
                          01  0F1E
                       36 00' 0F20              .ASCIC  "6"
                          01  0F20
                       37 00' 0F22              .ASCIC  "7"
                          01  0F22
                       38 00' 0F24              .ASCIC  "8"
                          01  0F24
                          00  0F26              .BYTE   0               ; Null length is end of valid 0,1,2,3,4,5,6,7,8
                          88  0F27              .BYTE   Pd$_Store       ; Indicate store operation
                        0037  0F28              .WORD   HP$W_DMZ32_LOOP         ; Byte HP$W_DMZ32_LOOP to data
                          00  0F2A              .BYTE   0               ; 0 HP$W_DMZ32_LOOP to data
```

```
                            05  0F2B                .BYTE   5                       ; 5 of data field
                            8B  0F2C                .BYTE   Pd$_Logical             ; Indicate logical value operation
72 74 6E 6F 43 20 6D 65 64 6F 4D 00' 0F2D           .ASCIC  \Modem Control\ ; Modem Control string
                         6C 6F  0F39
                            0D  0F2D
                            88  0F3B                .BYTE   Pd$_Store               ; Indicate store operation
                          0039  0F3C                .WORD   HP$B_DMZ32_MODEM                        ; Byte HP$B_DMZ32_MODEM to data
                            00  0F3E                .BYTE   0                       ; 0 HP$B_DMZ32_MODEM to data
                            02  0F3F                .BYTE   2                       ; 2 of data field
                            86  0F40                .BYTE   Pd$_Literal             ; Indicate literal
                      00000001  0F41                .LONG   1                       ; Constant
                            81  0F45                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                                0F46    690 DRB32:  $DS_DRB32                                       ;       [74][81]                ;G:33
                                0F46                .SAVE   LOCAL_BLOCK
                                0F46                .PSECT  $ABS$,ABS
                      00000032  0050                .=HP$A_DEPENDENT
                                0032                .IIF    NB,DRB32$B_node_id,     DRB32$B_node_id:
                      00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                                0033                .IIF    NB,DRB32$B_BR,  DRB32$B_BR:
                      00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                                0034                .IIF    NB,DRB32$K_LEN, DRB32$K_LEN:
                            00000F46                .RESTORE
                  32 33 42 52 44 00' 0F46           .ASCIC  "DRB32" ; DRB32 name
                            05  0F46
                            34  0F4C                .BYTE   DRB32$K_LEN                     ; DRB32$K_LEN of resultant P-table
                            10  0F4D                .BYTE   16                      ; Maximum unit number                   ;G:65
                          0000  0F4E                .WORD   ^A""    ; Two character mnemonic for QIO DRB32
                            80  0F50                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                            8D  0F51                .Byte   Pd$_Name                ; Directive op-code
                            01  0F52                .Byte   PTD$M_UNIT                      ; Enforced UQ codes
                       51 55 00' 0F53               .Ascic  "UQ"    ; Enforced device name prefix
                            02  0F53
                            87  0F56                .BYTE   Pd$_Fetch               ; Indicate fetch operation
                          0018  0F57                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                            19  0F59                .BYTE   25                      ; 25 HP$A_DEVICE to data
                            07  0F5A                .BYTE   7                       ; 7 of data field
                            8C  0F5B                .BYTE   Pd$_Case                ; Indicate case operation
                            01  0F5C                .BYTE   $$_$                    ; Count of pairs
                  00000030 00000000  0F5D           .LONG   0,^X30                  ; Store the pair
                            88  0F65                .BYTE   Pd$_Store               ; Indicate store operation
                          0018  0F66                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                            19  0F68                .BYTE   25                      ; 25 HP$A_DEVICE to data
                            07  0F69                .BYTE   7                       ; 7 of data field
                            84  0F6A                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 0F6B           .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
      29 58 45 48 28 20 72 65 62  0F77
                            14  0F6B
                  0000000F 00000000  0F80           .LONG   ^X0,^XF ; 0 , F limits on input
                            88  0F88                .BYTE   Pd$_Store               ; Indicate store operation
                          0032  0F89                .WORD   DRB32$B_NODE_ID                 ; Byte DRB32$B_NODE_ID to data
                            00  0F8B                .BYTE   0                       ; 0 DRB32$B_NODE_ID to data
                            08  0F8C                .BYTE   8                       ; 8 of data field
                            88  0F8D                .BYTE   Pd$_Store               ; Indicate store operation
                          0018  0F8E                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                            0D  0F90                .BYTE   13                      ; 13 HP$A_DEVICE to data
                            04  0F91                .BYTE   4                       ; 4 of data field
                            88  0F92                .BYTE   Pd$_Store               ; Indicate store operation
```

```
                    0024  0F93                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      02  0F95                .BYTE   2                ; 2 HP$W_VECTOR to data
                      04  0F96                .BYTE   4                ; 4 of data field
                      82  0F97                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                52 42 00' 0F98                .ASCIC  "BR"     ; BR string
                      02  0F98
          00000007 00000004  0F9B             .LONG   4,7      ; 4 , 7 limits on input
                      88  0FA3                .BYTE   Pd$_Store        ; Indicate store operation
                    0033  0FA4                .WORD   DRB32$B_BR              ; Byte DRB32$B_BR to data
                      00  0FA6                .BYTE   0                ; 0 DRB32$B_BR to data
                      08  0FA7                .BYTE   8                ; 8 of data field
                      88  0FA8                .BYTE   Pd$_Store        ; Indicate store operation
                    0024  0FA9                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      06  0FAB                .BYTE   6                ; 6 HP$W_VECTOR to data
                      03  0FAC                .BYTE   3                ; 3 of data field
                      81  0FAD                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          0FAE    691 DR11B:  $DS_DR11B
                          0FAE                .SAVE   LOCAL_BLOCK
                          0FAE                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DR11B$L_CSR, DR11B$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,DR11B$B_BR,  DR11B$B_BR:
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,DR11B$K_LEN, DR11B$K_LEN:
                      0000FAE                 .RESTORE
          42 31 31 52 44 00' 0FAE             .ASCIC  "DR11B" ; DR11B name
                      05  0FAE
                      37  0FB4                .BYTE   DR11B$K_LEN             ; DR11B$K_LEN of resultant P-table
                      00  0FB5                .BYTE   0                ; Maximum unit number                          ;G:65
                    0000  0FB6                .WORD   ^A""     ; Two character mnemonic for QIO DR11B
                      80  0FB8                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  0FB9                .Byte   Pd$_Name         ; Directive op-code
                      03  0FBA                .Byte   Ptd$M_Device             ; Enforced XB codes
                   42 58 00' 0FBB             .Ascic  "XB"     ; Enforced device name prefix
                      02  0FBB
                      83  0FBE                .BYTE   Pd$_Octal        ; Indicate scan octal
                52 53 43 00' 0FBF             .ASCIC  "CSR"    ; CSR string
                      03  0FBF
          0003FFFE 0003E000  0FC3             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  0FCB                .BYTE   Pd$_Store        ; Indicate store operation
                    0032  0FCC                .WORD   DR11B$L_CSR             ; Byte DR11B$L_CSR to data
                      00  0FCE                .BYTE   0                ; 0 DR11B$L_CSR to data
                      20  0FCF                .BYTE   32               ; 32 of data field
                      88  0FD0                .BYTE   Pd$_Store        ; Indicate store operation
                    0018  0FD1                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  0FD3                .BYTE   0                ; 0 HP$A_DEVICE to data
                      12  0FD4                .BYTE   18               ; 18 of data field
                      83  0FD5                .BYTE   Pd$_Octal        ; Indicate scan octal
       52 4F 54 43 45 56 00' 0FD6             .ASCIC  "VECTOR"         ; VECTOR string
                      06  0FD6
          000001FE 00000002  0FDD             .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  0FE5                .BYTE   Pd$_Store        ; Indicate store operation
                    0024  0FE6                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  0FE8                .BYTE   0                ; 0 HP$W_VECTOR to data
                      09  0FE9                .BYTE   9                ; 9 of data field
                      82  0FEA                .BYTE   Pd$_Decimal      ; Indicate scan decimal
```

```
                52 42 00' 0FEB                  .ASCIC  "BR"    ; BR string
                   02  0FEB
          00000007 00000004  0FEE                .LONG   4,7     ; 4 , 7 limits on input
                      88  0FF6                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  0FF7                .WORD   DR11B$B_BR              ; Byte DR11B$B_BR to data
                      00  0FF9                .BYTE   0               ; 0 DR11B$B_BR to data
                      08  0FFA                .BYTE   8               ; 8 of data field
                      81  0FFB                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          0FFC    692 DR11C:  $DS_DR11C                               ;       [56]
                          0FFC                .SAVE   LOCAL_BLOCK
                          0FFC                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_DR11C_BR,       HP$B_DR11C_BR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_DR11C_LEN,      HP$K_DR11C_LEN:
                      00000FFC                .RESTORE
             43 31 31 52 44 00' 0FFC          .ASCIC  "DR11C" ; DR11C name
                      05  0FFC
                      33  1002                .BYTE   HP$K_DR11C_LEN          ; HP$K_DR11C_LEN of resultant P-table
                      00  1003                .BYTE   0               ; Maximum unit number                   ;G:65
                    414F  1004                .WORD   ^A"OA"  ; Two character mnemonic for QIO DR11C OA
                      80  1006                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1007                .Byte   Pd$_Name        ; Directive op-code
                      02  1008                .Byte   Ptd$M_Controller                ; Enforced OA codes
                41 4F 00' 1009                .Ascic  "OA"    ; Enforced device name prefix
                      02  1009
                      83  100C                .BYTE   Pd$_Octal       ; Indicate scan octal
                52 53 43 00' 100D             .ASCIC  "CSR"   ; CSR string
                      03  100D
          0003FFFE 0003E000  1011             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  1019                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  101A                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  101C                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  101D                .BYTE   18              ; 18 of data field
                      83  101E                .BYTE   Pd$_Octal       ; Indicate scan octal
       52 4F 54 43 45 56 00' 101F             .ASCIC  "VECTOR"        ; VECTOR string
                      06  101F
          000001FE 00000000  1026             .LONG   ^O0,^O776       ; 0 , 776 limits on input
                      88  102E                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  102F                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  1031                .BYTE   0               ; 0 HP$W_VECTOR to data
                      10  1032                .BYTE   16              ; 16 of data field
                      82  1033                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 1034                .ASCIC  "BR"    ; BR string
                      02  1034
          00000007 00000004  1037             .LONG   4,7     ; 4 , 7 limits on input
                      88  103F                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1040                .WORD   HP$B_DR11C_BR           ; Byte HP$B_DR11C_BR to data
                      00  1042                .BYTE   0               ; 0 HP$B_DR11C_BR to data
                      08  1043                .BYTE   8               ; 8 of data field
                      81  1044                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1045    693 DR11K:  $DS_DR11K
                          1045                .SAVE   LOCAL_BLOCK
                          1045                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DR11K$B_BR,  DR11K$B_BR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
```

```
                          0033                  .IIF    NB,DR11K$K_LEN, DR11K$K_LEN:
                      00001045                  .RESTORE
                4B 31 31 52 44 00' 1045         .ASCIC  "DR11K" ; DR11K name
                          05  1045
                          33  104B              .BYTE   DR11K$K_LEN             ; DR11K$K_LEN of resultant P-table
                          06  104C              .BYTE   6                       ; Maximum unit number                       ;G:65
                        0000  104D              .WORD   ^A""    ; Two character mnemonic for QIO DR11K
                          80  104F              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  1050              .Byte   Pd$_Name        ; Directive op-code
                          03  1051              .Byte   Ptd$M_Device            ; Enforced XR codes
                       52 58 00' 1052           .Ascic  "XR"    ; Enforced device name prefix
                          02  1052
                          83  1055              .BYTE   Pd$_Octal       ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 1056       .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                          0B  1056
                0003FFFE 0003E000 1062          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  106A              .BYTE   Pd$_Store       ; Indicate store operation
                        0018  106B              .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  106D              .BYTE   0                       ; 0 HP$A_DEVICE to data
                          12  106E              .BYTE   18                      ; 18 of data field
                          83  106F              .BYTE   Pd$_Octal       ; Indicate scan octal
             52 4F 54 43 45 56 00' 1070         .ASCIC  "VECTOR"        ; VECTOR string
                          06  1070
                000001FE 00000002 1077          .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  107F              .BYTE   Pd$_Store       ; Indicate store operation
                        0024  1080              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  1082              .BYTE   0                       ; 0 HP$W_VECTOR to data
                          09  1083              .BYTE   9                       ; 9 of data field
                          82  1084              .BYTE   Pd$_Decimal     ; Indicate scan decimal
                       52 42 00' 1085           .ASCIC  "BR"    ; BR string
                          02  1085
            00000007 00000004 1088              .LONG   4,7     ; 4 , 7 limits on input
                          88  1090              .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1091              .WORD   DR11K$B_BR              ; Byte DR11K$B_BR to data
                          00  1093              .BYTE   0                       ; 0 DR11K$B_BR to data
                          08  1094              .BYTE   8                       ; 8 of data field
                          81  1095              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              1096  694 DR11W:   $DS_DR11W
                              1096              .SAVE   LOCAL_BLOCK
                              1096              .PSECT  $ABS$,ABS
                      00000032  0050            .=HP$A_DEPENDENT
                              0032              .IIF    NB,DR11W$L_CSR, DR11W$L_CSR:
                      00000036  0032            .IIF    NB,.BLKL,       .BLKL   1
                              0036              .IIF    NB,DR11W$B_BR,  DR11W$B_BR:
                      00000037  0036            .IIF    NB,.BLKB,       .BLKB   1
                              0037              .IIF    NB,DR11W$K_LEN, DR11W$K_LEN:
                      00001096                  .RESTORE
             57 31 31 52 44 00' 1096            .ASCIC  "DR11W" ; DR11W name
                          05  1096
                          37  109C              .BYTE   DR11W$K_LEN             ; DR11W$K_LEN of resultant P-table
                          00  109D              .BYTE   0                       ; Maximum unit number                       ;G:65
                        4158  109E              .WORD   ^A"XA"  ; Two character mnemonic for QIO DR11W XA
                          80  10A0              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  10A1              .Byte   Pd$_Name        ; Directive op-code
                          03  10A2              .Byte   Ptd$M_Device            ; Enforced XA codes
                       41 58 00' 10A3           .Ascic  "XA"    ; Enforced device name prefix
                          02  10A3
```

```
                    83  10A6                .BYTE   Pd$_Octal       ; Indicate scan octal
          52 53 43 00'  10A7                .ASCIC  "CSR"    ; CSR string
                    03  10A7
    0003FFFE 0003E000   10AB                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                    88  10B3                .BYTE   Pd$_Store       ; Indicate store operation
                  0032  10B4                .WORD   DR11W$L_CSR             ; Byte DR11W$L_CSR to data
                    00  10B6                .BYTE   0               ; 0 DR11W$L_CSR to data
                    20  10B7                .BYTE   32              ; 32 of data field
                    88  10B8                .BYTE   Pd$_Store       ; Indicate store operation
                  0018  10B9                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  10BB                .BYTE   0               ; 0 HP$A_DEVICE to data
                    12  10BC                .BYTE   18              ; 18 of data field
                    83  10BD                .BYTE   Pd$_Octal       ; Indicate scan octal
  52 4F 54 43 45 56 00'  10BE               .ASCIC  "VECTOR"        ; VECTOR string
                    06  10BE
    000001FE 00000002   10C5                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                    88  10CD                .BYTE   Pd$_Store       ; Indicate store operation
                  0024  10CE                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  10D0                .BYTE   0               ; 0 HP$W_VECTOR to data
                    09  10D1                .BYTE   9               ; 9 of data field
                    82  10D2                .BYTE   Pd$_Decimal     ; Indicate scan decimal
          52 42 00'  10D3                   .ASCIC  "BR"     ; BR string
                    02  10D3
    00000007 00000004   10D6                .LONG   4,7     ; 4 , 7 limits on input
                    88  10DE                .BYTE   Pd$_Store       ; Indicate store operation
                  0036  10DF                .WORD   DR11W$B_BR              ; Byte DR11W$B_BR to data
                    00  10E1                .BYTE   0               ; 0 DR11W$B_BR to data
                    08  10E2                .BYTE   8               ; 8 of data field
                    81  10E3                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        10E4    695 DR750:  $DS_DR750
                        10E4                .SAVE   LOCAL_BLOCK
                        10E4                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,DR750$B_SLOT,        DR750$B_SLOT:
              00000033  0032                .IIF    NB,,BLKB,       .BLKB   1
                        0033                .IIF    NB,DR750$B_BR,  DR750$B_BR:
              00000034  0033                .IIF    NB,,BLKB,       .BLKB   1
                        0034                .IIF    NB,DR750$B_SELF,        DR750$B_SELF:
              00000035  0034                .IIF    NB,,BLKB,       .BLKB   1
                        0035                .IIF    NB,DR750$B_CONFIG,      DR750$B_CONFIG:
              00000036  0035                .IIF    NB,,BLKB,       .BLKB   1
                        0036                .IIF    NB,DR750$B_UUT, DR750$B_UUT:
              00000037  0036                .IIF    NB,,BLKB,       .BLKB   1
                        0037                .IIF    NB,DR750$K_LEN, DR750$K_LEN:
                    000010E4                .RESTORE
  30 35 37 52 44 00'  10E4                  .ASCIC  "DR750" ; DR750 name
                    05  10E4
                    37  10EA                .BYTE   DR750$K_LEN             ; DR750$K_LEN of resultant P-table
                    FF  10EB                .BYTE   255             ; Maximum unit number                   ;G:65
                  4658  10EC                .WORD   ^A"XF"  ; Two character mnemonic for QIO DR750 XF
                    80  10EE                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  10EF                .Byte   Pd$_Name        ; Directive op-code
                    03  10F0                .Byte   Ptd$M_Device            ; Enforced XF codes
              46 58 00'  10F1               .Ascic  "XF"     ; Enforced device name prefix
                    02  10F1
                    86  10F4                .BYTE   Pd$_Literal     ; Indicate literal
              40F30000  10F5                .LONG   ^X40F30000              ; Constant
```

```
                      88  10F9                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  10FA                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                      00  10FC                .BYTE   0                ; 0 HP$A_DEVICE to data
                      20  10FD                .BYTE   32               ; 32 of data field
                      82  10FE                .BYTE   Pd$_Decimal      ; Indicate scan decimal
         54 4F 4C 53 00'  10FF                .ASCIC  "SLOT"  ; SLOT string
                      04  10FF
            0000000F 0000000A  1104           .LONG   10,15   ; 10 , 15 limits on input
                      88  110C                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  110D                .WORD   DR750$B_SLOT             ; Byte DR750$B_SLOT to data
                      00  110F                .BYTE   0                ; 0 DR750$B_SLOT to data
                      08  1110                .BYTE   8                ; 8 of data field
                      88  1111                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1112                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                      0D  1114                .BYTE   13               ; 13 HP$A_DEVICE to data
                      04  1115                .BYTE   4                ; 4 of data field
                      88  1116                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1117                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                      02  1119                .BYTE   2                ; 2 HP$W_VECTOR to data
                      04  111A                .BYTE   4                ; 4 of data field
                      82  111B                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                 52 42 00'  111C                .ASCIC  "BR"    ; BR string
                      02  111C
            00000007 00000004  111F           .LONG   4,7     ; 4 , 7 limits on input
                      88  1127                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  1128                .WORD   DR750$B_BR               ; Byte DR750$B_BR to data
                      00  112A                .BYTE   0                ; 0 DR750$B_BR to data
                      08  112B                .BYTE   8                ; 8 of data field
                      88  112C                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  112D                .WORD   HP$W_VECTOR              ; Byte HP$W_VECTOR to data
                      06  112F                .BYTE   6                ; 6 HP$W_VECTOR to data
                      03  1130                .BYTE   3                ; 3 of data field
                      8B  1131                .BYTE   Pd$_Logical      ; Indicate logical value operation
   54 53 45 54 20 46 4C 45 53 00'  1132        .ASCIC  \SELF TEST\      ; SELF TEST string
                      09  1132
                      88  113C                .BYTE   Pd$_Store       ; Indicate store operation
                    0034  113D                .WORD   DR750$B_SELF             ; Byte DR750$B_SELF to data
                      00  113F                .BYTE   0                ; 0 DR750$B_SELF to data
                      08  1140                .BYTE   8                ; 8 of data field
                      85  1141                .BYTE   Pd$_String       ; Indicate scan and verify string
47 49 46 4E 4F 43 20 54 53 45 54 00'  1142    .ASCIC  "TEST CONFIG"    ; TEST CONFIG string
                      0B  1142
                  55 50 43 31 00'  114E       .ASCIC  "1CPU"
                      04  114E
                  55 50 43 32 00'  1153       .ASCIC  "2CPU"
                      04  1153
                      00  1158                .BYTE   0                ; Null length is end of valid 1CPU,2CPU
                      88  1159                .BYTE   Pd$_Store       ; Indicate store operation
                    0035  115A                .WORD   DR750$B_CONFIG           ; Byte DR750$B_CONFIG to data
                      00  115C                .BYTE   0                ; 0 DR750$B_CONFIG to data
                      08  115D                .BYTE   8                ; 8 of data field
                      8B  115E                .BYTE   Pd$_Logical      ; Indicate logical value operation
   54 49 4E 55 20 54 53 45 54 00'  115F        .ASCIC  \TEST UNIT\      ; TEST UNIT string
                      09  115F
                      88  1169                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  116A                .WORD   DR750$B_UUT              ; Byte DR750$B_UUT to data
                      00  116C                .BYTE   0                ; 0 DR750$B_UUT to data
```

```
                      08  116D                .BYTE   8                       ; 8 of data field
                      81  116E                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          116F    696 DR780:   $DS_DR780
                          116F                .SAVE   LOCAL_BLOCK
                          116F                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DR780$B_TR,  DR780$B_TR:
                00000033  0032                .IIF    NB,,            .BLKB   1
                          0033                .IIF    NB,DR780$B_BR,  DR780$B_BR:
                00000034  0033                .IIF    NB,,            .BLKB   1
                          0034                .IIF    NB,DR780$B_SELF,        DR780$B_SELF:
                00000035  0034                .IIF    NB,,            .BLKB   1
                          0035                .IIF    NB,DR780$B_CONFIG,      DR780$B_CONFIG:
                00000036  0035                .IIF    NB,,            .BLKB   1
                          0036                .IIF    NB,DR780$B_UUT, DR780$B_UUT:
                00000037  0036                .IIF    NB,,            .BLKB   1
                          0037                .IIF    NB,DR780$K_LEN, DR780$K_LEN:
                 0000116F                     .RESTORE
          30 38 37 52 44 00' 116F             .ASCIC  "DR780" ; DR780 name
                      05  116F
                -     37  1175                .BYTE   DR780$K_LEN                     ; DR780$K_LEN of resultant P-table
                      FF  1176                .BYTE   255                     ; Maximum unit number                           ;G:65
                    4658  1177                .WORD   ^A"XF"  ; Two character mnemonic for QIO DR780 XF
                      80  1179                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  117A                .Byte   Pd$_Name        ; Directive op-code
                      03  117B                .Byte   Ptd$M_Device            ; Enforced XF codes
                46 58 00' 117C                .Ascic  "XF"    ; Enforced device name prefix
                      02  117C
                      82  117F                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 54 00' 1180                .ASCIC  "TR"    ; TR string
                      02  1180
            0000000F 00000001   1183          .LONG   1,15    ; 1 . 15 limits on input
                      88  118B                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  118C                .WORD   DR780$B_TR              ; Byte DR780$B_TR to data
                      00  118E                .BYTE   0               ; 0 DR780$B_TR to data
                      08  118F                .BYTE   8               ; 8 of data field
                      88  1190                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1191                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      0D  1193                .BYTE   13              ; 13 HP$A_DEVICE to data
                      04  1194                .BYTE   4               ; 4 of data field
                      88  1195                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1196                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      02  1198                .BYTE   2               ; 2 HP$W_VECTOR to data
                      04  1199                .BYTE   4               ; 4 of data field
                      82  119A                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 119B                .ASCIC  "BR"    ; BR string
                      02  119B
            00000007 00000004   119E          .LONG   4,7     ; 4 . 7 limits on input
                      88  11A6                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  11A7                .WORD   DR780$B_BR              ; Byte DR780$B_BR to data
                      00  11A9                .BYTE   0               ; 0 DR780$B_BR to data
                      08  11AA                .BYTE   8               ; 8 of data field
                      88  11AB                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  11AC                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      06  11AE                .BYTE   6               ; 6 HP$W_VECTOR to data
                      02  11AF                .BYTE   2               ; 2 of data field
                      86  11B0                .BYTE   Pd$_Literal     ; Indicate literal
```

```
                    00000006  11B1                .LONG   6               ; Constant
                          88  11B5                .BYTE   Pd$_Store       ; Indicate store operation
                        0018  11B6                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          1C  11B8                .BYTE   28              ; 28 HP$A_DEVICE to data
                          04  11B9                .BYTE   4               ; 4 of data field
                          86  11BA                .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  11BB                .LONG   1               ; Constant
                          88  11BF                .BYTE   Pd$_Store       ; Indicate store operation
                        0024  11C0                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          08  11C2                .BYTE   8               ; 8 HP$W_VECTOR to data
                          01  11C3                .BYTE   1               ; 1 of data field
                          8B  11C4                .BYTE   Pd$_Logical     ; Indicate logical value operation
      54 53 45 54 20 46 4C 45 53 00' 11C5                .ASCIC  \SELF TEST\     ; SELF TEST string
                          09  11C5
                          88  11CF                .BYTE   Pd$_Store       ; Indicate store operation
                        0034  11D0                .WORD   DR780$B_SELF            ; Byte DR780$B_SELF to data
                          00  11D2                .BYTE   0               ; 0 DR780$B_SELF to data
                          08  11D3                .BYTE   8               ; 8 of data field
                          85  11D4                .BYTE   Pd$_String      ; Indicate scan and verify string
47 49 46 4E 4F 43 20 54 53 45 54 00' 11D5                .ASCIC  "TEST CONFIG"   ; TEST CONFIG string
                          0B  11D5
                  55 50 43 31 00' 11E1                .ASCIC  "1CPU"
                          04  11E1
                  55 50 43 32 00' 11E6                .ASCIC  "2CPU"
                          04  11E6
                          00  11EB                .BYTE   0               ; Null length is end of valid 1CPU,2CPU
                          88  11EC                .BYTE   Pd$_Store       ; Indicate store operation
                        0035  11ED                .WORD   DR780$B_CONFIG           ; Byte DR780$B_CONFIG to data
                          00  11EF                .BYTE   0               ; 0 DR780$B_CONFIG to data
                          08  11F0                .BYTE   8               ; 8 of data field
                          8B  11F1                .BYTE   Pd$_Logical     ; Indicate logical value operation
   54 49 4E 55 20 54 53 45 54 00' 11F2                .ASCIC  \TEST UNIT\     ; TEST UNIT string
                          09  11F2
                          88  11FC                .BYTE   Pd$_Store       ; Indicate store operation
                        0036  11FD                .WORD   DR780$B_UUT             ; Byte DR780$B_UUT to data
                          00  11FF                .BYTE   0               ; 0 DR780$B_UUT to data
                          08  1200                .BYTE   8               ; 8 of data field
                          81  1201                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              1202    697 DUP11:  $DS_DUP11
                              1202                .SAVE   LOCAL_BLOCK
                              1202                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,DUP11$L_CSR, DUP11$L_CSR:
                    00000036  0032                .IIF    NB,..BLKL,      .BLKL   1
                              0036                .IIF    NB,DUP11$B_BR,  DUP11$B_BR:
                    00000037  0036                .IIF    NB,..BLKB,      .BLKB   1
                              0037                .IIF    NB,DUP11$K_LEN, DUP11$K_LEN:
                          0000120A                .RESTORE
            31 31 50 55 44 00' 1202                .ASCIC  "DUP11" ; DUP11 name
                          05  1202
                          37  1208                .BYTE   DUP11$K_LEN             ; DUP11$K_LEN of resultant P-table
                          00  1209                .BYTE   0               ; Maximum unit number                           ;G:65
                        0000  120A                .WORD   ^A""    ; Two character mnemonic for QIO DUP11
                          80  120C                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  120D                .Byte   Pd$_Name        ; Directive op-code
                          03  120E                .Byte   Ptd$M_Device            ; Enforced XW codes
                  57 58 00' 120F                .Ascic  "XW"    ; Enforced device name prefix
```

```
                      02  120F
                      83  1212                .BYTE   Pd$_Octal       ; Indicate scan octal
                52 53 43 00' 1213                .ASCIC  "CSR"   ; CSR string
                      03  1213
            0003FFFE 0003E000  1217                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  121F                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1220                .WORD   DUP11$L_CSR             ; Byte DUP11$L_CSR to data
                      00  1222                .BYTE   0               ; 0 DUP11$L_CSR to data
                      20  1223                .BYTE   32              ; 32 of data field
                      88  1224                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1225                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  1227                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  1228                .BYTE   18              ; 18 of data field
                      83  1229                .BYTE   Pd$_Octal       ; Indicate scan octal
          52 4F 54 43 45 56 00' 122A                .ASCIC  "VECTOR"        ; VECTOR string
                      06  122A
            000001FE 00000002  1231                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  1239                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  123A                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  123C                .BYTE   0               ; 0 HP$W_VECTOR to data
                      09  123D                .BYTE   9               ; 9 of data field
                      82  123E                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 123F                .ASCIC  "BR"    ; BR string
                      02  123F
            00000007 00000004  1242                .LONG   4,7     ; 4 , 7 limits on input
                      88  124A                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  124B                .WORD   DUP11$B_BR              ; Byte DUP11$B_BR to data
                      00  124D                .BYTE   0               ; 0 DUP11$B_BR to data
                      08  124E                .BYTE   8               ; 8 of data field
                      81  124F                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        1250    698 DW730:      $DS_DW730                       ;       [16]
                        1250                    .SAVE   LOCAL_BLOCK
                        1250                    .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                        0032                    .IIF    NB,DW730$K_LEN, DW730$K_LEN:
                    00001250                    .RESTORE
          30 33 37 57 44 00' 1250                .ASCIC  "DW730" ; DW730 name
                      05  1250
                      32  1256                .BYTE   DW730$K_LEN             ; DW730$K_LEN of resultant P-table
                      01  1257                .BYTE   1               ; Maximum unit number                   ;G:65
                    0000  1258                .WORD   ^A""    ; Two character mnemonic for QIO DW730
                      80  125A                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  125B                .Byte   Pd$_Name        ; Directive op-code
                      01  125C                .Byte   Ptd$M_Unit              ; Enforced DW codes
                57 44 00' 125D                .Ascic  "DW"    ; Enforced device name prefix
                      02  125D
                      86  1260                .BYTE   Pd$_Literal     ; Indicate literal
                40F26000  1261                .LONG   ^X40F26000              ; Constant
                      88  1265                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1266                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  1268                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  1269                .BYTE   32              ; 32 of data field
                      86  126A                .BYTE   Pd$_Literal     ; Indicate literal
                40FC0000  126B                .LONG   ^X40FC0000              ; Constant
                      88  126F                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  1270                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      00  1272                .BYTE   0               ; 0 HP$A_DVA to data
```

```
                      20  1273                .BYTE   32                  ; 32 of data field
                      86  1274                .BYTE   Pd$_Literal        ; Indicate literal
                00000200  1275                .LONG   ^X200             ; Constant
                      88  1279                .BYTE   Pd$_Store         ; Indicate store operation
                    0024  127A                .WORD   HP$W_VECTOR               ; Byte HP$W_VECTOR to data
                      00  127C                .BYTE   0                 ; 0 HP$W_VECTOR to data
                      10  127D                .BYTE   16                ; 16 of data field
                      81  127E                .BYTE   Pd$_End           ; Indicate end of PT-descriptor
                          127F    699 DW750:    $DS_DW750
                          127F                .SAVE   LOCAL_BLOCK
                          127F                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DW750$K_LEN, DW750$K_LEN:
                          0000127F            .RESTORE
          30 35 37 57 44 00' 127F             .ASCIC  "DW750" ; DW750 name
                      05  127F
                      32  1285                .BYTE   DW750$K_LEN               ; DW750$K_LEN of resultant P-table
                      04  1286                .BYTE   4                 ; Maximum unit number                          ;G:65
                    0000  1287                .WORD   ^A""      ; Two character mnemonic for QIO DW750
                      80  1289                .BYTE   Pd$_Start         ; Constant to identify start of descriptor
                      8D  128A                .Byte   Pd$_Name          ; Directive op-code
                      01  128B                .Byte   Ptd$M_Unit                ; Enforced DW codes
                   57 44 00' 128C             .Ascic  "DW"      ; Enforced device name prefix
                      02  128C
                      86  128F                .BYTE   Pd$_Literal        ; Indicate literal
                40F30000  1290                .LONG   ^X40F30000                ; Constant
                      88  1294                .BYTE   Pd$_Store         ; Indicate store operation
                    0018  1295                .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                      00  1297                .BYTE   0                 ; 0 HP$A_DEVICE to data
                      20  1298                .BYTE   32                ; 32 of data field
                      86  1299                .BYTE   Pd$_Literal        ; Indicate literal
                40F80000  129A                .LONG   ^X40F80000                ; Constant
                      88  129E                .BYTE   Pd$_Store         ; Indicate store operation
                    001C  129F                .WORD   HP$A_DVA                  ; Byte HP$A_DVA to data
                      00  12A1                .BYTE   0                 ; 0 HP$A_DVA to data
                      20  12A2                .BYTE   32                ; 32 of data field
                      86  12A3                .BYTE   Pd$_Literal        ; Indicate literal
                00000200  12A4                .LONG   ^X200             ; Constant
                      88  12A8                .BYTE   Pd$_Store         ; Indicate store operation
                    0024  12A9                .WORD   HP$W_VECTOR               ; Byte HP$W_VECTOR to data
                      00  12AB                .BYTE   0                 ; 0 HP$W_VECTOR to data
                      10  12AC                .BYTE   16                ; 16 of data field
                      87  12AD                .BYTE   Pd$_Fetch         ; Indicate fetch operation
                    000B  12AE                .WORD   HP$B_DRIVE                ; Byte HP$B_DRIVE to data
                      00  12B0                .BYTE   0                 ; 0 HP$B_DRIVE to data
                      08  12B1                .BYTE   8                 ; 8 of data field
                      88  12B2                .BYTE   Pd$_Store         ; Indicate store operation
                    0018  12B3                .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                      0D  12B5                .BYTE   13                ; 13 HP$A_DEVICE to data
                      01  12B6                .BYTE   1                 ; 1 of data field
                      8A  12B7                .BYTE   Pd$_Add           ; Indicate add value to field operation
                    0024  12B8                .WORD   HP$W_VECTOR               ; Byte HP$W_VECTOR to data
                      09  12BA                .BYTE   9                 ; 9 HP$W_VECTOR to data
                      02  12BB                .BYTE   2                 ; 2 of data field
                      89  12BC                .BYTE   Pd$_Complement    ; Indicate complement value operation
                      88  12BD                .BYTE   Pd$_Store         ; Indicate store operation
                    001C  12BE                .WORD   HP$A_DVA                  ; Byte HP$A_DVA to data
```

```
                      12  12C0                .BYTE   18              ; 18 HP$A_DVA to data
                      01  12C1                .BYTE   1               ; 1 of data field
                      81  12C2                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          12C3    700 DW780:   $DS_DW780
                          12C3                .SAVE   LOCAL_BLOCK
                          12C3                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DW780$B_TR,  DW780$B_TR:
                 00000033 0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,DW780$B_BR,  DW780$B_BR:
                 00000034 0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,DW780$K_LEN, DW780$K_LEN:
                      000012C3                .RESTORE
           30 38 37 57 44 00' 12C3            .ASCIC  "DW780" ; DW780 name
                      05  12C3
                      34  12C9                .BYTE   DW780$K_LEN             ; DW780$K_LEN of resultant P-table
                      08  12CA                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  12CB                .WORD   ^A""    ; Two character mnemonic for QIO DW780
                      80  12CD                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  12CE                .Byte   Pd$_Name        ; Directive op-code
                      01  12CF                .Byte   Ptd$M_Unit              ; Enforced DW codes
                 57 44 00' 12D0             .Ascic  "DW"    ; Enforced device name prefix
                      02  12D0
                      82  12D3                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 54 00' 12D4             .ASCIC  "TR"    ; TR string
                      02  12D4
            0000000F 00000001  12D7          .LONG   1,15    ; 1 , 15 limits on input
                      88  12DF               .BYTE   Pd$_Store       ; Indicate store operation
                    0032  12E0               .WORD   DW780$B_TR              ; Byte DW780$B_TR to data
                      00  12E2               .BYTE   0               ; 0 DW780$B_TR to data
                      08  12E3               .BYTE   8               ; 8 of data field
                      88  12E4               .BYTE   Pd$_Store       ; Indicate store operation
                    0018  12E5               .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      0D  12E7               .BYTE   13              ; 13 HP$A_DEVICE to data
                      04  12E8               .BYTE   4               ; 4 of data field
                      88  12E9               .BYTE   Pd$_Store       ; Indicate store operation
                    0024  12EA               .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      02  12EC               .BYTE   2               ; 2 HP$W_VECTOR to data
                      04  12ED               .BYTE   4               ; 4 of data field
                      82  12EE               .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 12EF             .ASCIC  "BR"    ; BR string
                      02  12EF
            00000007 00000004  12F2          .LONG   4,7     ; 4 , 7 limits on input
                      88  12FA               .BYTE   Pd$_Store       ; Indicate store operation
                    0033  12FB               .WORD   DW780$B_BR              ; Byte DW780$B_BR to data
                      00  12FD               .BYTE   0               ; 0 DW780$B_BR to data
                      08  12FE               .BYTE   8               ; 8 of data field
                      88  12FF               .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1300               .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      06  1302               .BYTE   6               ; 6 HP$W_VECTOR to data
                      02  1303               .BYTE   2               ; 2 of data field
                      87  1304               .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    000B  1305               .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                      00  1307               .BYTE   0               ; 0 HP$B_DRIVE to data
                      08  1308               .BYTE   8               ; 8 of data field
                      88  1309               .BYTE   Pd$_Store       ; Indicate store operation
                    001C  130A               .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
```

```
                      12  130C                .BYTE   18              ; 18 HP$A_DVA to data
                      02  130D                .BYTE   2               ; 2 of data field
                      86  130E                .BYTE   Pd$_Literal     ; Indicate literal
                00000006  130F                .LONG   6               ; Constant
                      88  1313                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1314                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      1C  1316                .BYTE   28              ; 28 HP$A_DEVICE to data
                      04  1317                .BYTE   4               ; 4 of data field
                      88  1318                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  1319                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      1C  131B                .BYTE   28              ; 28 HP$A_DVA to data
                      04  131C                .BYTE   4               ; 4 of data field
                      87  131D                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  131E                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      19  1320                .BYTE   25              ; 25 HP$A_DEVICE to data
                      02  1321                .BYTE   2               ; 2 of data field
                      88  1322                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  1323                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      19  1325                .BYTE   25              ; 25 HP$A_DVA to data
                      02  1326                .BYTE   2               ; 2 of data field
                      86  1327                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1328                .LONG   1               ; Constant
                      88  132C                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  132D                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      14  132F                .BYTE   20              ; 20 HP$A_DVA to data
                      01  1330                .BYTE   1               ; 1 of data field
                      88  1331                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1332                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      08  1334                .BYTE   8               ; 8 HP$W_VECTOR to data
                      01  1335                .BYTE   1               ; 1 of data field
                      81  1336                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1337  701 DWBLA:     $DS_DWBLA                       ;       [60]                    ;G:40
                          1337                .SAVE   LOCAL_BLOCK
                          1337                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,DWBLA$B_NODE_ID,     DWBLA$B_NODE_ID:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,DWBLA$B_BR,  DWBLA$B_BR:
                00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,DWBLA$W_VECTOR,      DWBLA$W_VECTOR:
                00000036  0034                .IIF    NB,.BLKW,       .BLKW   1
                          0036                .IIF    NB,DWBLA$K_LEN, DWBLA$K_LEN:
                    00001337                .RESTORE
          41 4C 42 57 44 00' 1337             .ASCIC  "DWBLA" ; DWBLA name
                      05  1337
                      36  133D                .BYTE   DWBLA$K_LEN             ; DWBLA$K_LEN of resultant P-table
                      10  133E                .BYTE   16              ; Maximum unit number                   ;G:65
                    0000  133F                .WORD   ^A""    ; Two character mnemonic for QIO DWBLA
                      80  1341                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      87  1342                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    0018  1343                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      19  1345                .BYTE   25              ; 25 HP$A_DEVICE to data
                      07  1346                .BYTE   7               ; 7 of data field
                      8C  1347                .BYTE   Pd$_Case        ; Indicate case operation
                      01  1348                .BYTE   $$_$            ; Count of pairs
          00000030 00000000  1349             .LONG   0,^X30          ; Store the pair
                      88  1351                .BYTE   Pd$_Store       ; Indicate store operation
```

```
                     0018  1352                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       19  1354                .BYTE   25                  ; 25 HP$A_DEVICE to data
                       07  1355                .BYTE   7                   ; 7 of data field
                       88  1356                .BYTE   Pd$_Store           ; Indicate store operation
                     001C  1357                .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                       19  1359                .BYTE   25                  ; 25 HP$A_DVA to data
                       07  135A                .BYTE   7                   ; 7 of data field
                       86  135B                .BYTE   Pd$_Literal         ; Indicate literal
                 00000001  135C                .LONG   ^X1                 ; Constant
                       88  1360                .BYTE   Pd$_Store           ; Indicate store operation
                     001C  1361                .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                       16  1363                .BYTE   22                  ; 22 HP$A_DVA to data
                       01  1364                .BYTE   1                   ; 1 of data field
                       84  1365                .BYTE   Pd$_Hexadecimal     ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 1366      .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
         29 58 45 48 28 20 72 65 62  1372
                       14  1366
                 0000000F 00000000  137B                .LONG   ^X0,^XF ; 0 , F limits on input
                       88  1383                .BYTE   Pd$_Store           ; Indicate store operation
                     0032  1384                .WORD   DWBLA$B_NODE_ID             ; Byte DWBLA$B_NODE_ID to data
                       00  1386                .BYTE   0                   ; 0 DWBLA$B_NODE_ID to data
                       08  1387                .BYTE   8                   ; 8 of data field
                       88  1388                .BYTE   Pd$_Store           ; Indicate store operation
                     0018  1389                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                       0D  138B                .BYTE   13                  ; 13 HP$A_DEVICE to data
                       04  138C                .BYTE   4                   ; 4 of data field
                       88  138D                .BYTE   Pd$_Store           ; Indicate store operation
                     001C  138E                .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                       12  1390                .BYTE   18                  ; 18 HP$A_DVA to data
                       04  1391                .BYTE   4                   ; 4 of data field
                       88  1392                .BYTE   Pd$_Store           ; Indicate store operation
                     0024  1393                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                       02  1395                .BYTE   2                   ; 2 HP$W_VECTOR to data
                       04  1396                .BYTE   4                   ; 4 of data field
                       86  1397                .BYTE   Pd$_Literal         ; Indicate literal
                 00000005  1398                .LONG   ^X5                 ; Constant
                       88  139C                .BYTE   Pd$_Store           ; Indicate store operation
                     0024  139D                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                       06  139F                .BYTE   6                   ; 6 HP$W_VECTOR to data
                       03  13A0                .BYTE   3                   ; 3 of data field
                       88  13A1                .BYTE   Pd$_Store           ; Indicate store operation
                     0033  13A2                .WORD   DWBLA$B_BR                  ; Byte DWBLA$B_BR to data
                       00  13A4                .BYTE   0                   ; 0 DWBLA$B_BR to data
                       08  13A5                .BYTE   8                   ; 8 of data field
                       81  13A6                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           13A7    702 DWBUA:   $DS_DWBUA                                  ;       [48]
                           13A7                .SAVE   LOCAL_BLOCK
                           13A7                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,DWBUA$B_NODE_ID,     DWBUA$B_NODE_ID:
                 00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                           0033                .IIF    NB,DWBUA$B_BR,  DWBUA$B_BR:
                 00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                           0034                .IIF    NB,DWBUA$W_VECTOR,      DWBUA$W_VECTOR:
                 00000036  0034                .IIF    NB,.BLKW,       .BLKW   1
                           0036                .IIF    NB,DWBUA$K_LEN, DWBUA$K_LEN:
                       000013A7                .RESTORE
```

```
                    41 55 42 57 44 00'  13A7          .ASCIC  "DWBUA" ; DWBUA name
                                 05     13A7
                                 36     13AD          .BYTE   DWBUA$K_LEN             ; DWBUA$K_LEN of resultant P-table
                                 10     13AE          .BYTE   16                      ; Maximum unit number                    ;G:65
                               0000     13AF          .WORD   ^A""    ; Two character mnemonic for QIO DWBUA
                                 80     13B1          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                 87     13B2          .BYTE   Pd$_Fetch       ; Indicate fetch operation
                               0018     13B3          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                 19     13B5          .BYTE   25              ; 25 HP$A_DEVICE to data
                                 07     13B6          .BYTE   7               ; 7 of data field
                                 8C     13B7          .BYTE   Pd$_Case        ; Indicate case operation
                                 01     13B8          .BYTE   $$_$            ; Count of pairs
                    00000030 00000000   13B9          .LONG   0,^X30          ; Store the pair
                                 88     13C1          .BYTE   Pd$_Store       ; Indicate store operation
                               0018     13C2          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                 19     13C4          .BYTE   25              ; 25 HP$A_DEVICE to data
                                 07     13C5          .BYTE   7               ; 7 of data field
                                 88     13C6          .BYTE   Pd$_Store       ; Indicate store operation
                               001C     13C7          .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                                 19     13C9          .BYTE   25              ; 25 HP$A_DVA to data
                                 07     13CA          .BYTE   7               ; 7 of data field
                                 86     13CB          .BYTE   Pd$_Literal     ; Indicate literal
                           00000001     13CC          .LONG   ^X1             ; Constant
                                 88     13D0          .BYTE   Pd$_Store       ; Indicate store operation
                               001C     13D1          .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                                 16     13D3          .BYTE   22              ; 22 HP$A_DVA to data
                                 01     13D4          .BYTE   1               ; 1 of data field
                                 84     13D5          .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'    13D6          .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
      29 58 45 48 28 20 72 65 62        13E2
                                 14     13D6
                    0000000F 00000000   13EB          .LONG   ^X0,^XF ; 0 , F limits on input
                                 88     13F3          .BYTE   Pd$_Store       ; Indicate store operation
                               0032     13F4          .WORD   DWBUA$B_NODE_ID         ; Byte DWBUA$B_NODE_ID to data
                                 00     13F6          .BYTE   0               ; 0 DWBUA$B_NODE_ID to data
                                 08     13F7          .BYTE   8               ; 8 of data field
                                 88     13F8          .BYTE   Pd$_Store       ; Indicate store operation
                               0018     13F9          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                 0D     13FB          .BYTE   13              ; 13 HP$A_DEVICE to data
                                 04     13FC          .BYTE   4               ; 4 of data field
                                 88     13FD          .BYTE   Pd$_Store       ; Indicate store operation
                               001C     13FE          .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                                 12     1400          .BYTE   18              ; 18 HP$A_DVA to data
                                 04     1401          .BYTE   4               ; 4 of data field
                                 88     1402          .BYTE   Pd$_Store       ; Indicate store operation
                               0024     1403          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                 02     1405          .BYTE   2               ; 2 HP$W_VECTOR to data
                                 04     1406          .BYTE   4               ; 4 of data field
                                 82     1407          .BYTE   Pd$_Decimal     ; Indicate scan decimal
                          52 42 00'     1408          .ASCIC  "BR"    ; BR string
                                 02     1408
                    00000007 00000004   140B          .LONG   4,7     ; 4 , 7 limits on input
                                 88     1413          .BYTE   Pd$_Store       ; Indicate store operation
                               0024     1414          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                 06     1416          .BYTE   6               ; 6 HP$W_VECTOR to data
                                 03     1417          .BYTE   3               ; 3 of data field
                                 88     1418          .BYTE   Pd$_Store       ; Indicate store operation
```

```
                     0033  1419                .WORD   DWBUA$B_BR              ; Byte DWBUA$B_BR to data
                       00  141B                .BYTE   0                 ; 0 DWBUA$B_BR to data
                       08  141C                .BYTE   8                 ; 8 of data field
                       81  141D                .BYTE   Pd$_End           ; Indicate end of PT-descriptor
                           141E    703 DX20:    $DS_DX20                                ;       [57]
                           141E                .SAVE   LOCAL_BLOCK
                           141E                .PSECT  $ABS$,ABS
                   00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$B_DX20_DRIVE,     HP$B_DX20_DRIVE::
                           0032                .IIF    NB,DX20$B_DRIVE,        DX20$B_DRIVE::
                   00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                           0033                .IIF    NB,HP$K_DX20_LEN,       HP$K_DX20_LEN::
                           0033                .IIF    NB,DX20$K_LEN,  DX20$K_LEN::
                       0000141E                .RESTORE
                30 32 58 44 00' 141E            .ASCIC  "DX20"  ; DX20 name
                       04  141E
                       33  1423                .BYTE   DX20$K_LEN              ; DX20$K_LEN of resultant P-table
                       08  1424                .BYTE   8                 ; Maximum unit number                  ;G:65
                     0000  1425                .WORD   ^A""    ; Two character mnemonic for QIO DX20
                       80  1427                .BYTE   Pd$_Start         ; Constant to identify start of descriptor
                       8D  1428                .Byte   Pd$_Name          ; Directive op-code
                       02  1429                .Byte   PTD$M_CONTROLLER                ; Enforced MR codes
                   52 4D 00' 142A              .Ascic  "MR"    ; Enforced device name prefix
                       02  142A
                       82  142D                .BYTE   Pd$_Decimal       ; Indicate scan decimal
             45 56 49 52 44 00' 142E            .ASCIC  "DRIVE" ; DRIVE string
                       05  142E
           00000007 00000000  1434              .LONG   0,7     ; 0 , 7 limits on input
                       88  143C                .BYTE   Pd$_Store         ; Indicate store operation
                     0032  143D                .WORD   DX20$B_DRIVE            ; Byte DX20$B_DRIVE to data
                       00  143F                .BYTE   0                 ; 0 DX20$B_DRIVE to data
                       08  1440                .BYTE   8                 ; 8 of data field
                       88  1441                .BYTE   Pd$_Store         ; Indicate store operation
                     000B  1442                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                       00  1444                .BYTE   0                 ; 0 HP$B_DRIVE to data
                       08  1445                .BYTE   8                 ; 8 of data field
                       88  1446                .BYTE   Pd$_Store         ; Indicate store operation
                     0018  1447                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       07  1449                .BYTE   7                 ; 7 HP$A_DEVICE to data
                       03  144A                .BYTE   3                 ; 3 of data field
                       81  144B                .BYTE   Pd$_End           ; Indicate end of PT-descriptor
                           144C    704 DZ11:    $DS_DZ11
                           144C                .SAVE   LOCAL_BLOCK
                           144C                .PSECT  $ABS$,ABS
                   00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,DZ11$L_CSR,  DZ11$L_CSR:
                   00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                           0036                .IIF    NB,DZ11$B_BR,   DZ11$B_BR:
                   00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                           0037                .IIF    NB,DZ11$B_MTYPE,        DZ11$B_MTYPE:
                   00000038  0037                .IIF    NB,.BLKB,       .BLKB   1
                           0038                .IIF    NB,DZ11$K_LEN,  DZ11$K_LEN:
                       0000144C                .RESTORE
                31 31 5A 44 00' 144C            .ASCIC  "DZ11"  ; DZ11 name
                       04  144C
                       38  1451                .BYTE   DZ11$K_LEN              ; DZ11$K_LEN of resultant P-table
                       00  1452                .BYTE   0                 ; Maximum unit number                  ;G:65
```

```
                      5454  1453                .WORD   ^A"TT"  ; Two character mnemonic for QIO DZ11 TT
                        80  1455                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  1456                .Byte   Pd$_Name        ; Directive op-code
                        02  1457                .Byte   Ptd$M_Controller                ; Enforced TT codes
                   54 54 00' 1458                .Asclc  "TT"    ; Enforced device name prefix
                        02  1458
                        83  145B                .BYTE   Pd$_Octal       ; Indicate scan octal
                 52 53 43 00' 145C                .ASCIC  "CSR"   ; CSR string
                        03  145C
              0003FFFE 0003E000  1460                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  1468                .BYTE   Pd$_Store       ; Indicate store operation
                      0032  1469                .WORD   DZ11$L_CSR              ; Byte DZ11$L_CSR to data
                        00  146B                .BYTE   0               ; 0 DZ11$L_CSR to data
                        20  146C                .BYTE   32              ; 32 of data field
                        88  146D                .BYTE   Pd$_Store       ; Indicate store operation
                      0018  146E                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        00  1470                .BYTE   0               ; 0 HP$A_DEVICE to data
                        12  1471                .BYTE   18              ; 18 of data field
                        83  1472                .BYTE   Pd$_Octal       ; Indicate scan octal
           52 4F 54 43 45 56 00' 1473                .ASCIC  "VECTOR"        ; VECTOR string
                        06  1473
              000001FE 00000002  147A                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                        88  1482                .BYTE   Pd$_Store       ; Indicate store operation
                      0024  1483                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                        00  1485                .BYTE   0               ; 0 HP$W_VECTOR to data
                        09  1486                .BYTE   9               ; 9 of data field
                        82  1487                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                   52 42 00' 1488                .ASCIC  "BR"    ; BR string
                        02  1488
              00000007 00000004  148B                .LONG   4,7     ; 4 , 7 limits on input
                        88  1493                .BYTE   Pd$_Store       ; Indicate store operation
                      0036  1494                .WORD   DZ11$B_BR               ; Byte DZ11$B_BR to data
                        00  1496                .BYTE   0               ; 0 DZ11$B_BR to data
                        08  1497                .BYTE   8               ; 8 of data field
                        85  1498                .BYTE   Pd$_String      ; Indicate scan and verify string
45 50 59 54 20 45 4C 55 44 4F 4D 00' 1499                .ASCIC  "MODULE TYPE"   ; MODULE TYPE string
                        0B  1499
                  41 49 45 00' 14A5                .ASCIC  "EIA"
                        03  14A5
               41 4D 30 32 00' 14A9                .ASCIC  "20MA"
                        04  14A9
                        00  14AE                .BYTE   0               ; Null length is end of valid EIA,20MA
                        88  14AF                .BYTE   Pd$_Store       ; Indicate store operation
                      0037  14B0                .WORD   DZ11$B_MTYPE            ; Byte DZ11$B_MTYPE to data
                        00  14B2                .BYTE   0               ; 0 DZ11$B_MTYPE to data
                        08  14B3                .BYTE   8               ; 8 of data field
                        81  14B4                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            14B5    705 DZ32:   $DS_DZ32                                ;       [28]
                            14B5                .SAVE   LOCAL_BLOCK
                            14B5                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$B_DZ32_BRLVL,     HP$B_DZ32_BRLVL:
                            0032                .IIF    NB,DZ32$B_BRLVL,        DZ32$B_BRLVL:
                  00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                            0033                .IIF    NB,HP$B_DZ32_ACTIV,     HP$B_DZ32_ACTIV:
                            0033                .IIF    NB,DZ32$B_ACTIV,        DZ32$B_ACTIV:
                  00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
```

```
                        0034                .IIF    NB,HP$B_DZ32_SPEED,     HP$B_DZ32_SPEED:
                        0034                .IIF    NB,DZ32$B_SPEED,        DZ32$B_SPEED:
                    00000035  0034            .IIF    NB,,BLKB,       .BLKB   1
                        0035                .IIF    NB,HP$W_DZ32_FLAGS,     HP$W_DZ32_FLAGS:
                        0035                .IIF    NB,DZ32$W_FLAGS,        DZ32$W_FLAGS:
                    00000037  0035            .IIF    NB,,BLKW,       .BLKW   1
                        0037                .IIF    NB,HP$K_LEN,    HP$K_LEN:
                        0037                .IIF    NB,DZ32$_LEN,   DZ32$_LEN:
                   000014B5                 .RESTORE
               32 33 5A 44 00'  14B5        .ASCIC  "DZ32"  ; DZ32 name
                          04    14B5
                          37    14BA        .BYTE   DZ32$_LEN                   ; DZ32$_LEN of resultant P-table
                          00    14BB        .BYTE   0                   ; Maximum unit number                       ;G:65
                        0000    14BC        .WORD   ^A""    ; Two character mnemonic for QIO DZ32
                          80    14BE        .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                          8D    14BF        .Byte   Pd$_Name            ; Directive op-code
                          02    14C0        .Byte   Ptd$M_Controller                    ; Enforced TT codes
                     54 54 00'  14C1        .Ascic  "TT"    ; Enforced device name prefix
                          02    14C1
                          83    14C4        .BYTE   Pd$_Octal           ; Indicate scan octal
                   52 53 43 00'  14C5        .ASCIC  "CSR"   ; CSR string
                          03    14C5
                  0003FFFE 0003E000  14C9    .LONG   ^O760000,^O777776           ; 760000 , 777776 limits on input
                          88    14D1        .BYTE   Pd$_Store           ; Indicate store operation
                        0018    14D2        .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                          00    14D4        .BYTE   0                   ; 0 HP$A_DEVICE to data
                          12    14D5        .BYTE   18                  ; 18 of data field
                          83    14D6        .BYTE   Pd$_Octal           ; Indicate scan octal
            52 4F 54 43 45 56 00'  14D7      .ASCIC  "VECTOR"            ; VECTOR string
                          06    14D7
                  000001FE 00000002  14DE    .LONG   ^O2,^O776           ; 2 , 776 limits on input
                          88    14E6        .BYTE   Pd$_Store           ; Indicate store operation
                        0024    14E7        .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                          00    14E9        .BYTE   0                   ; 0 HP$W_VECTOR to data
                          10    14EA        .BYTE   16                  ; 16 of data field
                          83    14EB        .BYTE   Pd$_Octal           ; Indicate scan octal
                     52 42 00'  14EC        .ASCIC  "BR"    ; BR string
                          02    14EC
                  00000007 00000004  14EF    .LONG   ^O4,^O7 ; 4 , 7 limits on input
                          88    14F7        .BYTE   Pd$_Store           ; Indicate store operation
                        0032    14F8        .WORD   DZ32$B_BRLVL                ; Byte DZ32$B_BRLVL to data
                          00    14FA        .BYTE   0                   ; 0 DZ32$B_BRLVL to data
                          08    14FB        .BYTE   8                   ; 8 of data field
                          83    14FC        .BYTE   Pd$_Octal           ; Indicate scan octal
45 4E 49 4C 20 45 56 49 54 43 41 00'  14FD  .ASCIC  "ACTIVE LINES"      ; ACTIVE LINES string
                          53    1509
                          0C    14FD
                  000000FF 00000000  150A    .LONG   ^O0,^O377           ; 0 , 377 limits on input
                          88    1512        .BYTE   Pd$_Store           ; Indicate store operation
                        0033    1513        .WORD   DZ32$B_ACTIV                ; Byte DZ32$B_ACTIV to data
                          00    1515        .BYTE   0                   ; 0 DZ32$B_ACTIV to data
                          08    1516        .BYTE   8                   ; 8 of data field
                          83    1517        .BYTE   Pd$_Octal           ; Indicate scan octal
      45 54 41 52 20 44 55 41 42 00'  1518  .ASCIC  "BAUD RATE"        ; BAUD RATE string
                          09    1518
                  0000000E 00000000  1522    .LONG   ^O0,^O16            ; 0 , 16 limits on input
                          88    152A        .BYTE   Pd$_Store           ; Indicate store operation
```

```
                        0034  152B                .WORD   DZ32$B_SPEED            ; Byte DZ32$B_SPEED to data
                          00  152D                .BYTE   0                ; 0 DZ32$B_SPEED to data
                          08  152E                .BYTE   8                ; 8 of data field
                          83  152F                .BYTE   Pd$_Octal        ; Indicate scan octal
59 54 20 4B 43 41 42 50 4F 4F 4C 00' 1530        .ASCIC  "LOOPBACK TYPE" ; LOOPBACK TYPE string
                       45 50  153C
                          0D  1530
                00000004 00000000  153E        .LONG   ^O0,^O4 ; 0 , 4 limits on input
                          88  1546             .BYTE   Pd$_Store        ; Indicate store operation
                        0035  1547             .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                          00  1549             .BYTE   0                ; 0 DZ32$W_FLAGS to data
                          03  154A             .BYTE   3                ; 3 of data field
                          83  154B             .BYTE   Pd$_Octal        ; Indicate scan octal
53 20 54 45 53 45 52 20 48 4E 49 00' 154C        .ASCIC  "INH RESET SWITCH"       ; INH RESET SWITCH string
                   48 43 54 49 57  1558
                          10  154C
                00000001 00000000  155D        .LONG   ^O0,^O1 ; 0 , 1 limits on input
                          88  1565             .BYTE   Pd$_Store        ; Indicate store operation
                        0035  1566             .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                          03  1568             .BYTE   3                ; 3 DZ32$W_FLAGS to data
                          01  1569             .BYTE   1                ; 1 of data field
                          83  156A             .BYTE   Pd$_Octal        ; Indicate scan octal
48 43 54 49 57 53 20 45 44 4F 4D 00' 156B        .ASCIC  "MODE SWITCH"    ; MODE SWITCH string
                          0B  156B
                00000001 00000000  1577        .LONG   ^O0,^O1 ; 0 , 1 limits on input
                          88  157F             .BYTE   Pd$_Store        ; Indicate store operation
                        0035  1580             .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                          04  1582             .BYTE   4                ; 4 DZ32$W_FLAGS to data
                          01  1583             .BYTE   1                ; 1 of data field
                          83  1584             .BYTE   Pd$_Octal        ; Indicate scan octal
44 45 45 50 53 20 54 49 4C 50 53 00' 1585        .ASCIC  "SPLIT SPEED JUMPER"     ; SPLIT SPEED JUMPER string
                   52 45 50 4D 55 4A  1591
                          12  1585
                00000002 00000000  1598        .LONG   ^O0,^O2 ; 0 , 2 limits on input
                          88  15A0             .BYTE   Pd$_Store        ; Indicate store operation
                        0035  15A1             .WORD   DZ32$W_FLAGS             ; Byte DZ32$W_FLAGS to data
                          05  15A3             .BYTE   5                ; 5 DZ32$W_FLAGS to data
                          02  15A4             .BYTE   2                ; 2 of data field
                          81  15A5             .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                              15A6  706 IEU11A: $DS_IEU11A                              ;       [39]
                              15A6             .SAVE   LOCAL_BLOCK
                              15A6             .PSECT  $ABS$,ABS
                    00000032  0050             .=HP$A_DEPENDENT
                              0032             .IIF    NB,HP$B_IEU11A_BR,      HP$B_IEU11A_BR:
                              0032             .IIF    NB,IEU11A$B_BR, IEU11A$B_BR:
                    00000033  0032             .IIF    NB,,BLKB,       .BLKB   1
                              0033             .IIF    NB,HP$K_IEU11A_LEN,     HP$K_IEU11A_LEN:
                              0033             .IIF    NB,IEU11A$K_LEN,        IEU11A$K_LEN:
                              15A6             .RESTORE
                41 31 31 55 45 49 00' 15A6     .ASCIC  "IEU11A"         ; IEU11A name
                          06  15A6
                          33  15AD             .BYTE   IEU11A$K_LEN             ; IEU11A$K_LEN of resultant P-table
                          01  15AE             .BYTE   1                ; Maximum unit number                          ;G:65
                        0000  15AF             .WORD   ^A""     ; Two character mnemonic for QIO IEU11A
                          80  15B1             .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                          8D  15B2             .Byte   Pd$_Name         ; Directive op-code
                          02  15B3             .Byte   PTD$M_CONTROLLER                 ; Enforced IX codes
```

```
                    58 49 00' 15B4                  .Ascic  "IX"    ; Enforced device name prefix
                       02  15B4
                       83  15B7                    .BYTE   Pd$_Octal       ; Indicate scan octal
                 65 73 61 42 00' 15B8               .ASCIC  "Base"  ; Base string
                       04  15B8
                0003FFF0 0003E000  15BD             .LONG   ^O760000,^O777760       ; 760000 , 777760 limits on input
                       88  15C5                     .BYTE   Pd$_Store       ; Indicate store operation
                     0018  15C6                     .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  15C8                     .BYTE   0               ; 0 HP$A_DEVICE to data
                       12  15C9                     .BYTE   18              ; 18 of data field
                       83  15CA                     .BYTE   Pd$_Octal       ; Indicate scan octal
              72 6F 74 63 65 56 00' 15CB            .ASCIC  "Vector"        ; Vector string
                       06  15CB
                000001FC 00000000  15D2             .LONG   ^O0,^O774       ; 0 , 774 limits on input
                       88  15DA                     .BYTE   Pd$_Store       ; Indicate store operation
                     0024  15DB                     .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       00  15DD                     .BYTE   0               ; 0 HP$W_VECTOR to data
                       10  15DE                     .BYTE   16              ; 16 of data field
                       82  15DF                     .BYTE   Pd$_Decimal     ; Indicate scan decimal
                    52 42 00' 15E0                  .ASCIC  "BR"    ; BR string
                       02  15E0
                00000007 00000004  15E3             .LONG   4,7     ; 4 , 7 limits on input
                       88  15EB                     .BYTE   Pd$_Store       ; Indicate store operation
                     0032  15EC                     .WORD   IEU11A$B_BR             ; Byte IEU11A$B_BR to data
                       00  15EE                     .BYTE   0               ; 0 IEU11A$B_BR to data
                       08  15EF                     .BYTE   8               ; 8 of data field
                       81  15F0                     .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                           15F1     707 KA730:      $DS_KA730                               ;       [15]
              30 33 37 41 4B 00' 15F1               .ASCIC  "KA730" ; KA730 name
                       05  15F1
                       50  15F7                     .BYTE   KA$K_LEN                ; KA$K_LEN of resultant P-table
                       04  15F8                     .BYTE   4               ; Maximum unit number                   ;G:65
                     0000  15F9                     .WORD   ^A""    ; Two character mnemonic for QIO KA730
                       80  15FB                     .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  15FC                     .Byte   Pd$_Name        ; Directive op-code
                       01  15FD                     .Byte   Ptd$M_Unit              ; Enforced KA codes
                    41 4B 00' 15FE                  .Ascic  "KA"    ; Enforced device name prefix
                       02  15FE
                       86  1601                     .BYTE   Pd$_Literal     ; Indicate literal
                 40F30000  1602                     .LONG   ^X40F30000              ; Constant
                       88  1606                     .BYTE   Pd$_Store       ; Indicate store operation
                     0018  1607                     .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  1609                     .BYTE   0               ; 0 HP$A_DEVICE to data
                       20  160A                     .BYTE   32              ; 32 of data field
                       86  160B                     .BYTE   Pd$_Literal     ; Indicate literal
                 000001FF  160C                     .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                       88  1610                     .BYTE   Pd$_Store       ; Indicate store operation
                     0032  1611                     .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                       00  1613                     .BYTE   0               ; 0 KA$L_FLAGS to data
                       20  1614                     .BYTE   32              ; 32 of data field
                       8B  1615                     .BYTE   Pd$_Logical     ; Indicate logical value operation
61 65 79 2D 66 6F 2D 65 6D 69 54 00' 1616           .ASCIC  \Time-of-year clock\    ; Time-of-year clock string
           6B 63 6F 6C 63 20 72  1622
                       12  1616
                       88  1629                     .BYTE   Pd$_Store       ; Indicate store operation
                     0032  162A                     .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                       10  162C                     .BYTE   KA$V_TODR               ; KA$V_TODR KA$L_FLAGS to data
```

```
                                  01  162D          .BYTE   1                  ; 1 of data field
                                  84  162E          .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'  162F          .ASCIC  "WCS last address"       ; WCS last address string
                     73 73 65 72 64  163B
                                  10  162F
                  0000FFFF 00000000  1640          .LONG   ^X0,^X1@16-1     ; 0 , 1@16-1 limits on input
                                  88  1648          .BYTE   Pd$_Store        ; Indicate store operation
                                0036  1649          .WORD   KA$W_WCS                 ; Byte KA$W_WCS to data
                                  00  164B          .BYTE   0                ; 0 KA$W_WCS to data
                                  10  164C          .BYTE   16               ; 16 of data field
                                  86  164D          .BYTE   Pd$_Literal      ; Indicate literal
                            00002710  164E          .LONG   10000            ; Constant
                                  88  1652          .BYTE   Pd$_Store        ; Indicate store operation
                                003E  1653          .WORD   KA$W_LATENCY             ; Byte KA$W_LATENCY to data
                                  00  1655          .BYTE   0                ; 0 KA$W_LATENCY to data
                                  20  1656          .BYTE   32               ; 32 of data field
                                  86  1657          .BYTE   Pd$_Literal      ; Indicate literal
                            000003E8  1658          .LONG   1000             ; Constant
                                  88  165C          .BYTE   Pd$_Store        ; Indicate store operation
                                0040  165D          .WORD   KA$W_PROGRESS            ; Byte KA$W_PROGRESS to data
                                  00  165F          .BYTE   0                ; 0 KA$W_PROGRESS to data
                                  20  1660          .BYTE   32               ; 32 of data field
                                  82  1661          .BYTE   Pd$_Decimal      ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00'  1662          .ASCIC  "Accelerator type"       ; Accelerator type string
                     65 70 79 74 20  166E
                                  10  1662
                  000000FF 00000000  1673          .LONG   0,255    ; 0 , 255 limits on input
                                  88  167B          .BYTE   Pd$_Store        ; Indicate store operation
                                0042  167C          .WORD   KA$B_ACC_TYPE            ; Byte KA$B_ACC_TYPE to data
                                  00  167E          .BYTE   0                ; 0 KA$B_ACC_TYPE to data
                                  08  167F          .BYTE   8                ; 8 of data field
                                  82  1680          .BYTE   Pd$_Decimal      ; Indicate scan decimal
20 66 6F 20 73 65 74 79 62 2D 4B 00'  1681          .ASCIC  "K-bytes of Main Memory"         ; K-bytes of Main Memory string
   79 72 6F 6D 65 4D 20 6E 69 61 4D  168D
                                  16  1681
                  00001400 00000000  1698          .LONG   0,5120   ; 0 , 5120 limits on input
                                  88  16A0          .BYTE   Pd$_Store        ; Indicate store operation
                                0044  16A1          .WORD   KA$W_MEM_SIZE            ; Byte KA$W_MEM_SIZE to data
                                  00  16A3          .BYTE   0                ; 0 KA$W_MEM_SIZE to data
                                  10  16A4          .BYTE   16               ; 16 of data field
                                  8B  16A5          .BYTE   Pd$_Logical      ; Indicate logical value operation
6F 6C 20 53 43 57 20 72 65 73 55 00'  16A6          .ASCIC  \User WCS loaded\        ; User WCS loaded string
                        64 65 64 61  16B2
                                  0F  16A6
                                  88  16B6          .BYTE   Pd$_Store        ; Indicate store operation
                                0032  16B7          .WORD   KA$L_FLAGS               ; Byte KA$L_FLAGS to data
                                  18  16B9          .BYTE   KA$V_WCS_LD              ; KA$V_WCS_LD KA$L_FLAGS to data
                                  01  16BA          .BYTE   1                ; 1 of data field
                                  8B  16BB          .BYTE   Pd$_Logical      ; Indicate logical value operation
      73 72 6F 72 72 65 20 42 53 00'  16BC          .ASCIC  \SB errors\      ; SB errors string
                                  09  16BC
                                  88  16C6          .BYTE   Pd$_Store        ; Indicate store operation
                                0032  16C7          .WORD   KA$L_FLAGS               ; Byte KA$L_FLAGS to data
                                  19  16C9          .BYTE   KA$V_SB_ERR              ; KA$V_SB_ERR KA$L_FLAGS to data
                                  01  16CA          .BYTE   1                ; 1 of data field
                                  81  16CB          .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                      16CC   708 KA750:  $DS_KA750
```

```
                30 35 37 41 4B 00'  16CC                .ASCIC  "KA750" ; KA750 name
                                05  16CC
                                50  16D2                .BYTE   KA$K_LEN                    ; KA$K_LEN of resultant P-table
                                04  16D3                .BYTE   4                   ; Maximum unit number                       ;G:65
                              0000  16D4                .WORD   ^A""    ; Two character mnemonic for QIO KA750
                                80  16D6                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  16D7                .Byte   Pd$_Name        ; Directive op-code
                                01  16D8                .Byte   Ptd$M_Unit              ; Enforced KA codes
                           41 4B 00'  16D9              .Ascic  "KA"    ; Enforced device name prefix
                                02  16D9
                                86  16DC                .BYTE   Pd$_Literal     ; Indicate literal
                          40F30000  16DD                .LONG   ^X40F30000                  ; Constant
                                88  16E1                .BYTE   Pd$_Store       ; Indicate store operation
                              0018  16E2                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                                00  16E4                .BYTE   0                   ; 0 HP$A_DEVICE to data
                                20  16E5                .BYTE   32                  ; 32 of data field
                                86  16E6                .BYTE   Pd$_Literal     ; Indicate literal
                          000001F3  16E7                .LONG   KA$M_F_FLOAT ! KA$M_D_FLOAT !                   KA$M_CRC !
                                88  16EB                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  16EC                .WORD   KA$L_FLAGS                  ; Byte KA$L_FLAGS to data
                                00  16EE                .BYTE   0                   ; 0 KA$L_FLAGS to data
                                20  16EF                .BYTE   32                  ; 32 of data field
                                8B  16F0                .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 47 00'  16F1              .ASCIC  \G-floating instructions\       ; G-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69   16FD
                                17  16F1
                                88  1709                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  170A                .WORD   KA$L_FLAGS                  ; Byte KA$L_FLAGS to data
                                02  170C                .BYTE   KA$V_G_FLOAT                ; KA$V_G_FLOAT KA$L_FLAGS to data
                                01  170D                .BYTE   1                   ; 1 of data field
                                8B  170E                .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 48 00'  170F              .ASCIC  \H-floating instructions\       ; H-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69   171B
                                17  170F
                                88  1727                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  1728                .WORD   KA$L_FLAGS                  ; Byte KA$L_FLAGS to data
                                03  172A                .BYTE   KA$V_H_FLOAT                ; KA$V_H_FLOAT KA$L_FLAGS to data
                                01  172B                .BYTE   1                   ; 1 of data field
                                8B  172C                .BYTE   Pd$_Logical     ; Indicate logical value operation
61 65 79 2D 66 6F 2D 65 6D 69 54 00'  172D              .ASCIC  \Time-of-year clock\    ; Time-of-year clock string
                   6B 63 6F 6C 63 20 72  1739
                                12  172D
                                88  1740                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  1741                .WORD   KA$L_FLAGS                  ; Byte KA$L_FLAGS to data
                                10  1743                .BYTE   KA$V_TODR                   ; KA$V_TODR KA$L_FLAGS to data
                                01  1744                .BYTE   1                   ; 1 of data field
                                84  1745                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'  1746              .ASCIC  "WCS last address"      ; WCS last address string
                        73 73 65 72 64  1752
                                10  1746
                   0000FFFF 00000000  1757              .LONG   ^X0,^X1a16-1    ; 0 , 1a16-1 limits on input
                                88  175F                .BYTE   Pd$_Store       ; Indicate store operation
                              0036  1760                .WORD   KA$W_WCS                    ; Byte KA$W_WCS to data
                                00  1762                .BYTE   0                   ; 0 KA$W_WCS to data
                                10  1763                .BYTE   16                  ; 16 of data field
                                86  1764                .BYTE   Pd$_Literal     ; Indicate literal
                          00002710  1765                .LONG   10000               ; Constant
```

```
                                  88  1769                .BYTE   Pd$_Store        ; Indicate store operation
                                003E  176A                .WORD   KA$W_LATENCY             ; Byte KA$W_LATENCY to data
                                  00  176C                .BYTE   0                ; 0 KA$W_LATENCY to data
                                  20  176D                .BYTE   32               ; 32 of data field
                                  86  176E                .BYTE   Pd$_Literal      ; Indicate literal
                            000003E8  176F                .LONG   1000             ; Constant
                                  88  1773                .BYTE   Pd$_Store        ; Indicate store operation
                                0040  1774                .WORD   KA$W_PROGRESS            ; Byte KA$W_PROGRESS to data
                                  00  1776                .BYTE   0                ; 0 KA$W_PROGRESS to data
                                  20  1777                .BYTE   32               ; 32 of data field
                                  82  1778                .BYTE   Pd$_Decimal      ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00'  1779                .ASCIC  "Accelerator type"       ; Accelerator type string
                     65 70 79 74 20  1785
                                  10  1779
                  000000FF 00000000  178A                .LONG   0,255    ; 0 , 255 limits on input
                                  88  1792                .BYTE   Pd$_Store        ; Indicate store operation
                                0042  1793                .WORD   KA$B_ACC_TYPE            ; Byte KA$B_ACC_TYPE to data
                                  00  1795                .BYTE   0                ; 0 KA$B_ACC_TYPE to data
                                  08  1796                .BYTE   8                ; 8 of data field
                                  81  1797                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                      1798   709 KA780:   $DS_KA780
                  30 38 37 41 4B 00'  1798                .ASCIC  "KA780" ; KA780 name
                                  05  1798
                                  50  179E                .BYTE   KA$K_LEN                 ; KA$K_LEN of resultant P-table
                                  04  179F                .BYTE   4                ; Maximum unit number                    ;G:65
                                0000  17A0                .WORD   ^A""     ; Two character mnemonic for QIO KA780
                                  80  17A2                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                                  8D  17A3                .Byte   Pd$_Name         ; Directive op-code
                                  01  17A4                .Byte   Ptd$M_Unit               ; Enforced KA codes
                            41 4B 00'  17A5                .Ascic  "KA"     ; Enforced device name prefix
                                  02  17A5
                                  86  17A8                .BYTE   Pd$_Literal      ; Indicate literal
                            000101F3  17A9                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                    KA$M_CRC +
                                  88  17AD                .BYTE   Pd$_Store        ; Indicate store operation
                                0032  17AE                .WORD   KA$L_FLAGS               ; Byte KA$L_FLAGS to data
                                  00  17B0                .BYTE   0                ; 0 KA$L_FLAGS to data
                                  20  17B1                .BYTE   32               ; 32 of data field
                                  8B  17B2                .BYTE   Pd$_Logical      ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 47 00'  17B3                .ASCIC  \G-floating instructions\        ; G-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  17BF
                                  17  17B3
                                  88  17CB                .BYTE   Pd$_Store        ; Indicate store operation
                                0032  17CC                .WORD   KA$L_FLAGS               ; Byte KA$L_FLAGS to data
                                  02  17CE                .BYTE   KA$V_G_FLOAT             ; KA$V_G_FLOAT KA$L_FLAGS to data
                                  01  17CF                .BYTE   1                ; 1 of data field
                                  8B  17D0                .BYTE   Pd$_Logical      ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 48 00'  17D1                .ASCIC  \H-floating instructions\        ; H-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  17DD
                                  17  17D1
                                  88  17E9                .BYTE   Pd$_Store        ; Indicate store operation
                                0032  17EA                .WORD   KA$L_FLAGS               ; Byte KA$L_FLAGS to data
                                  03  17EC                .BYTE   KA$V_H_FLOAT             ; KA$V_H_FLOAT KA$L_FLAGS to data
                                  01  17ED                .BYTE   1                ; 1 of data field
                                  84  17EE                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'  17EF                .ASCIC  "WCS last address"       ; WCS last address string
                     73 73 65 72 64  17FB
                                  10  17EF
```

```
                  0000FFFF 00000000  1800                .LONG   ^X0,^X1a16-1    ; 0 , 1a16-1 limits on input
                                88  1808                .BYTE   Pd$_Store       ; Indicate store operation
                              0036  1809                .WORD   KA$W_WCS                ; Byte KA$W_WCS to data
                                00  180B                .BYTE   0               ; 0 KA$W_WCS to data
                                10  180C                .BYTE   16              ; 16 of data field
                                86  180D                .BYTE   Pd$_Literal     ; Indicate literal
                          00002710  180E                .LONG   10000           ; Constant
                                88  1812                .BYTE   Pd$_Store       ; Indicate store operation
                              003E  1813                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                                00  1815                .BYTE   0               ; 0 KA$W_LATENCY to data
                                20  1816                .BYTE   32              ; 32 of data field
                                86  1817                .BYTE   Pd$_Literal     ; Indicate literal
                          000003E8  1818                .LONG   1000            ; Constant
                                88  181C                .BYTE   Pd$_Store       ; Indicate store operation
                              0040  181D                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                                00  181F                .BYTE   0               ; 0 KA$W_PROGRESS to data
                                20  1820                .BYTE   32              ; 32 of data field
                                82  1821                .BYTE   Pd$_Decimal     ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00*  1822                .ASCIC  "Accelerator type"      ; Accelerator type string
                   65 70 79 74 20  182E
                                10  1822
                  000000FF 00000000  1833                .LONG   0,255   ; 0 , 255 limits on input
                                88  183B                .BYTE   Pd$_Store       ; Indicate store operation
                              0042  183C                .WORD   KA$B_ACC_TYPE           ; Byte KA$B_ACC_TYPE to data
                                00  183E                .BYTE   0               ; 0 KA$B_ACC_TYPE to data
                                08  183F                .BYTE   8               ; 8 of data field
                                81  1840                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    1841    710 KA785:  $DS_KA785                               ;       [36]
                  35 38 37 41 4B 00*  1841                .ASCIC  "KA785" ; KA785 name
                                05  1841
                                50  1847                .BYTE   KA$K_LEN                ; KA$K_LEN of resultant P-table
                                04  1848                .BYTE   4               ; Maximum unit number                   ;G:65
                              0000  1849                .WORD   ^A""    ; Two character mnemonic for QIO KA785
                                80  184B                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  184C                .Byte   Pd$_Name        ; Directive op-code
                                01  184D                .Byte   Ptd$M_Unit              ; Enforced KA codes
                           41 4B 00*  184E                .Ascic  "KA"    ; Enforced device name prefix
                                02  184E
                                86  1851                .BYTE   Pd$_Literal     ; Indicate literal
                          000101F3  1852                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                                88  1856                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  1857                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                00  1859                .BYTE   0               ; 0 KA$L_FLAGS to data
                                20  185A                .BYTE   32              ; 32 of data field
                                8B  185B                .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 47 00*  185C                .ASCIC  \G-floating instructions\       ; G-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  1868
                                17  185C
                                88  1874                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  1875                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                02  1877                .BYTE   KA$V_G_FLOAT            ; KA$V_G_FLOAT KA$L_FLAGS to data
                                01  1878                .BYTE   1               ; 1 of data field
                                8B  1879                .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 48 00*  187A                .ASCIC  \H-floating instructions\       ; H-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  1886
                                17  187A
                                88  1892                .BYTE   Pd$_Store       ; Indicate store operation
```

```
                          0032  1893                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                            03  1895                .BYTE   KA$V_H_FLOAT            ; KA$V_H_FLOAT KA$L_FLAGS to data
                            01  1896                .BYTE   1                   ; 1 of data field
                            84  1897                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 4A 00' 1898           .ASCIC  "JCS last address"      ; JCS last address string
                  73 73 65 72 64  18A4
                            10  1898
                  0000FFFF 00000000  18A9           .LONG   ^X0,^X1a16-1    ; 0 , 1a16-1 limits on input
                            88  18B1                .BYTE   Pd$_Store       ; Indicate store operation
                          0036  18B2                .WORD   KA$W_WCS                ; Byte KA$W_WCS to data
                            00  18B4                .BYTE   0               ; 0 KA$W_WCS to data
                            10  18B5                .BYTE   16              ; 16 of data field
                            86  18B6                .BYTE   Pd$_Literal     ; Indicate literal
                      00002710  18B7                .LONG   10000           ; Constant
                            88  18BB                .BYTE   Pd$_Store       ; Indicate store operation
                          003E  18BC                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                            00  18BE                .BYTE   0               ; 0 KA$W_LATENCY to data
                            20  18BF                .BYTE   32              ; 32 of data field
                            86  18C0                .BYTE   Pd$_Literal     ; Indicate literal
                      000003E8  18C1                .LONG   1000            ; Constant
                            88  18C5                .BYTE   Pd$_Store       ; Indicate store operation
                          0040  18C6                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                            00  18C8                .BYTE   0               ; 0 KA$W_PROGRESS to data
                            20  18C9                .BYTE   32              ; 32 of data field
                            82  18CA                .BYTE   Pd$_Decimal     ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00' 18CB           .ASCIC  "Accelerator type"      ; Accelerator type string
                  65 70 79 74 20  18D7
                            10  18CB
                  000000FF 00000000  18DC           .LONG   0,255   ; 0 , 255 limits on input
                            88  18E4                .BYTE   Pd$_Store       ; Indicate store operation
                          0042  18E5                .WORD   KA$B_ACC_TYPE           ; Byte KA$B_ACC_TYPE to data
                            00  18E7                .BYTE   0               ; 0 KA$B_ACC_TYPE to data
                            08  18E8                .BYTE   8               ; 8 of data field
                            81  18E9                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                18EA    711 KA800:  $DS_KA800                       ;       [68]
                                18EA                .SAVE   LOCAL_BLOCK
                                18EA                .PSECT  $ABS$,ABS
                      00000032  0050                .=HP$A_DEPENDENT
                                0032                .IIF    NB,KA800$B_Float,       KA800$B_Float:
                      00000033  0032                .IIF    NB,..blkb,      .blkb   1
                                0033                .IIF    NB,KA800$B_MEM, KA800$B_MEM:
                      00000034  0033                .IIF    NB,..BLKB,      .BLKB   1
                                0034                .IIF    NB,KA800$B_ID,  KA800$B_ID:
                      00000035  0034                .IIF    NB,..BLKB,      .BLKB   1
                                0035                .IIF    NB,KA800$K_LEN, KA800$K_LEN:
                          000018EA                  .RESTORE
                  30 30 38 41 4B 00' 18EA           .ASCIC  "KA800" ; KA800 name
                            05  18EA
                            35  18F0                .BYTE   KA800$K_LEN             ; KA800$K_LEN of resultant P-table
                            06  18F1                .BYTE   6               ; Maximum unit number                   ;G:65
                          0000  18F2                .WORD   ^A""    ; Two character mnemonic for QIO KA800
                            80  18F4                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                            8D  18F5                .Byte   Pd$_Name        ; Directive op-code
                            03  18F6                .Byte   PTD$M_DEVICE            ; Enforced K codes
                        4B 00'  18F7                .Ascic  "K"     ; Enforced device name prefix
                            01  18F7
                            86  18F9                .BYTE   Pd$_Literal     ; Indicate literal
```

```
                    00000001  18FA                .LONG   1                       ; Constant
                          88  18FE                .BYTE   Pd$_Store               ; Indicate store operation
                        0032  18FF                .WORD   KA800$B_Float                   ; Byte KA800$B_Float to data
                          00  1901                .BYTE   0                       ; 0 KA800$B_Float to data
                          08  1902                .BYTE   8                       ; 8 of data field
                          82  1903                .BYTE   Pd$_Decimal             ; Indicate scan decimal
72 6F 6D 65 4D 20 6C 61 63 6F 4C 00' 1904         .ASCIC  "Local Memory Size in MB"       ; Local Memory Size in MB string
42 4D 20 6E 69 20 65 7A 69 53 20 79  1910
                          17  1904
                00000015 00000001  191C                .LONG   1,21    ; 1 , 21 limits on input
                          88  1924                .BYTE   Pd$_Store               ; Indicate store operation
                        0033  1925                .WORD   KA800$B_MEM                     ; Byte KA800$B_MEM to data
                          00  1927                .BYTE   0                       ; 0 KA800$B_MEM to data
                          08  1928                .BYTE   8                       ; 8 of data field
                          86  1929                .BYTE   Pd$_Literal             ; Indicate literal
                    00000006  192A                .LONG   ^X6                     ; Constant
                          88  192E                .BYTE   Pd$_Store               ; Indicate store operation
                        0018  192F                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                          1C  1931                .BYTE   28                      ; 28 HP$A_DEVICE to data
                          04  1932                .BYTE   4                       ; 4 of data field
                          84  1933                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 1934         .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
        29 58 45 48 28 20 72 65 62   1940
                          14  1934
                0000000F 00000000  1949                .LONG   ^X0,^XF ; 0 , F limits on input
                          88  1951                .BYTE   Pd$_Store               ; Indicate store operation
                        0034  1952                .WORD   KA800$B_ID                      ; Byte KA800$B_ID to data
                          00  1954                .BYTE   0                       ; 0 KA800$B_ID to data
                          08  1955                .BYTE   8                       ; 8 of data field
                          88  1956                .BYTE   Pd$_Store               ; Indicate store operation
                        0018  1957                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                          0D  1959                .BYTE   13                      ; 13 HP$A_DEVICE to data
                          04  195A                .BYTE   4                       ; 4 of data field
                          81  195B                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                              195C    712 KA820:  $DS_KA820                               ;       [59]
30 32 38 41 4B 00' 195C                   .ASCIC  "KA820" ; KA820 name
                          05  195C
                          50  1962                .BYTE   KA$K_Len                        ; KA$K_Len of resultant P-table
                          0F  1963                .BYTE   15                      ; Maximum unit number                   ;G:65
                        0000  1964                .WORD   ^A""    ; Two character mnemonic for QIO KA820
                          80  1966                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                          8D  1967                .Byte   Pd$_Name                ; Directive op-code
                          04  1968                .Byte   Ptd$M_Name                      ; Enforced KA codes
                    41 4B 00' 1969                .Ascic  "KA"    ; Enforced device name prefix
                          02  1969
                          86  196C                .BYTE   Pd$_Literal             ; Indicate literal
                    000101FF  196D                .LONG   KA$M_F_Float!KA$M_D_Float!KA$M_CRC!KA$M_Packed!                 KA$
                          88  1971                .BYTE   Pd$_Store               ; Indicate store operation
                        0032  1972                .WORD   KA$L_Flags                      ; Byte KA$L_Flags to data
                          00  1974                .BYTE   0                       ; 0 KA$L_Flags to data
                          20  1975                .BYTE   32                      ; 32 of data field
                          86  1976                .BYTE   Pd$_Literal             ; Indicate literal
                    00000000  1977                .LONG   0                       ; Constant
                          88  197B                .BYTE   Pd$_Store               ; Indicate store operation
                        0036  197C                .WORD   KA$W_Wcs                        ; Byte KA$W_Wcs to data
                          00  197E                .BYTE   0                       ; 0 KA$W_Wcs to data
                          10  197F                .BYTE   16                      ; 16 of data field
```

```
                                    86  1980                .BYTE   Pd$_Literal     ; Indicate literal
                            00002710  1981                .LONG   10000           ; Constant
                                    88  1985                .BYTE   Pd$_Store       ; Indicate store operation
                                003E  1986                .WORD   KA$W_Latency            ; Byte KA$W_Latency to data
                                    00  1988                .BYTE   0               ; 0 KA$W_Latency to data
                                    10  1989                .BYTE   16              ; 16 of data field
                                    86  198A                .BYTE   Pd$_Literal     ; Indicate literal
                            000003E8  198B                .LONG   1000            ; Constant
                                    88  198F                .BYTE   Pd$_Store       ; Indicate store operation
                                0040  1990                .WORD   KA$W_Progress           ; Byte KA$W_Progress to data
                                    00  1992                .BYTE   0               ; 0 KA$W_Progress to data
                                    10  1993                .BYTE   16              ; 16 of data field
                                    86  1994                .BYTE   Pd$_Literal     ; Indicate literal
                            00000001  1995                .LONG   1               ; Constant
                                    88  1999                .BYTE   Pd$_Store       ; Indicate store operation
                                0042  199A                .WORD   KA$B_Acc_Type           ; Byte KA$B_Acc_Type to data
                                    00  199C                .BYTE   0               ; 0 KA$B_Acc_Type to data
                                    08  199D                .BYTE   8               ; 8 of data field
                                    82  199E                .BYTE   Pd$_Decimal     ; Indicate scan decimal
20 66 6F 20 73 65 74 79 62 2D 4B 00' 199F                .ASCIC  "K-bytes of Main Memory"        ; K-bytes of Main Memory string
   79 72 6F 6D 65 4D 20 6E 69 61 4D  19AB
                                    16  199F
                 00080000 00000000  19B6                .LONG   0,524288        ; 0 , 524288 limits on input
                                    88  19BE                .BYTE   Pd$_Store       ; Indicate store operation
                                0047  19BF                .WORD   KA$L_Mem_Size           ; Byte KA$L_Mem_Size to data
                                    00  19C1                .BYTE   0               ; 0 KA$L_Mem_Size to data
                                    20  19C2                .BYTE   32              ; 32 of data field
                                    84  19C3                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 19C4                .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
        29 58 45 48 28 20 72 65 62  19D0
                                    14  19C4
                 0000000F 00000000  19D9                .LONG   ^X0,^XF ; 0 , F limits on input
                                    88  19E1                .BYTE   Pd$_Store       ; Indicate store operation
                                0046  19E2                .WORD   KA$B_Node_ID            ; Byte KA$B_Node_ID to data
                                    00  19E4                .BYTE   0               ; 0 KA$B_Node_ID to data
                                    08  19E5                .BYTE   8               ; 8 of data field
                                    81  19E6                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                        19E7  713 KA86:     $DS_KA86                                ;       [49]
                     36 38 41 4B 00' 19E7                .ASCIC  "KA86"  ; KA86 name
                                    04  19E7
                                    50  19EC                .BYTE   KA$K_LEN                ; KA$K_LEN of resultant P-table
                                    04  19ED                .BYTE   4               ; Maximum unit number                   ;G:65
                                0000  19EE                .WORD   ^A""    ; Two character mnemonic for QIO KA86
                                    80  19F0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                    8D  19F1                .Byte   Pd$_Name        ; Directive op-code
                                    01  19F2                .Byte   Ptd$M_Unit              ; Enforced KA codes
                            41 4B 00' 19F3                .Ascic  "KA"    ; Enforced device name prefix
                                    02  19F3
                                    86  19F6                .BYTE   Pd$_Literal     ; Indicate literal
                            000101F3  19F7                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                                    88  19FB                .BYTE   Pd$_Store       ; Indicate store operation
                                0032  19FC                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                    00  19FE                .BYTE   0               ; 0 KA$L_FLAGS to data
                                    20  19FF                .BYTE   32              ; 32 of data field
                                    8B  1A00                .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 47 00' 1A01                .ASCIC  \G-floating instructions\       ; G-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69  1A0D
```

```
                                  17  1A01
                                  88  1A19                .BYTE   Pd$_Store       ; Indicate store operation
                                0032  1A1A                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  02  1A1C                .BYTE   KA$V_G_FLOAT            ; KA$V_G_FLOAT KA$L_FLAGS to data
                                  01  1A1D                .BYTE   1               ; 1 of data field
                                  8B  1A1E                .BYTE   Pd$_Logical     ; Indicate logical value operation
20 67 6E 69 74 61 6F 6C 66 2D 48 00'  1A1F                .ASCIC  \H-floating instructions\       ; H-floating instructions string
73 6E 6F 69 74 63 75 72 74 73 6E 69   1A2B
                                  17  1A1F
                                  88  1A37                .BYTE   Pd$_Store       ; Indicate store operation
                                0032  1A38                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                                  03  1A3A                .BYTE   KA$V_H_FLOAT            ; KA$V_H_FLOAT KA$L_FLAGS to data
                                  01  1A3B                .BYTE   1               ; 1 of data field
                                  84  1A3C                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
64 61 20 74 73 61 6C 20 53 43 57 00'  1A3D                .ASCIC  "WCS last address"      ; WCS last address string
                     73 73 65 72 64   1A49
                                  10  1A3D
                  0000FFFF 00000000   1A4E                .LONG   ^X0,^X1@16-1    ; 0 , 1@16-1 limits on input
                                  88  1A56                .BYTE   Pd$_Store       ; Indicate store operation
                                0036  1A57                .WORD   KA$W_WCS                ; Byte KA$W_WCS to data
                                  00  1A59                .BYTE   0               ; 0 KA$W_WCS to data
                                  10  1A5A                .BYTE   16              ; 16 of data field
                                  86  1A5B                .BYTE   Pd$_Literal     ; Indicate literal
                            00002710  1A5C                .LONG   10000           ; Constant
                                  88  1A60                .BYTE   Pd$_Store       ; Indicate store operation
                                003E  1A61                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                                  00  1A63                .BYTE   0               ; 0 KA$W_LATENCY to data
                                  20  1A64                .BYTE   32              ; 32 of data field
                                  86  1A65                .BYTE   Pd$_Literal     ; Indicate literal
                            000003E8  1A66                .LONG   1000            ; Constant
                                  88  1A6A                .BYTE   Pd$_Store       ; Indicate store operation
                                0040  1A6B                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                                  00  1A6D                .BYTE   0               ; 0 KA$W_PROGRESS to data
                                  20  1A6E                .BYTE   32              ; 32 of data field
                                  82  1A6F                .BYTE   Pd$_Decimal     ; Indicate scan decimal
72 6F 74 61 72 65 6C 65 63 63 41 00'  1A70                .ASCIC  "Accelerator type"      ; Accelerator type string
                     65 70 79 74 20   1A7C
                                  10  1A70
                  000000FF 00000000   1A81                .LONG   0,255   ; 0 , 255 limits on input
                                  88  1A89                .BYTE   Pd$_Store       ; Indicate store operation
                                0042  1A8A                .WORD   KA$B_ACC_TYPE           ; Byte KA$B_ACC_TYPE to data
                                  00  1A8C                .BYTE   0               ; 0 KA$B_ACC_TYPE to data
                                  08  1A8D                .BYTE   8               ; 8 of data field
                                  81  1A8E                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                      1A8F   714 KA850:   $DS_KA850                       ;                                   ;G:9
                  30 35 38 41 4B 00'  1A8F                .ASCIC  "KA850" ; KA850 name
                                  05  1A8F
                                  37  1A95                .BYTE   HP$K_LEN                ; HP$K_LEN of resultant P-table
                                  02  1A96                .BYTE   2               ; Maximum unit number                   ;G:65
                                0000  1A97                .WORD   ^A""    ; Two character mnemonic for QIO KA850
                                  80  1A99                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                  8D  1A9A                .Byte   Pd$_Name        ; Directive op-code
                                  01  1A9B                .Byte   PTD$M_UNIT              ; Enforced KA codes
                            41 4B 00'  1A9C                .Ascic  "KA"    ; Enforced device name prefix
                                  02  1A9C
                                  86  1A9F                .BYTE   Pd$_Literal     ; Indicate literal
                            00000001  1AA0                .LONG   1               ; Constant
```

```
                      88  1AA4                .BYTE   Pd$_Store       ; Indicate store operation
                    004B  1AA5                .WORD   HP$L_KAA_SLOT           ; Byte HP$L_KAA_SLOT to data
                      00  1AA7                .BYTE   0               ; 0 HP$L_KAA_SLOT to data
                      20  1AA8                .BYTE   32              ; 32 of data field
                      86  1AA9                .BYTE   Pd$_Literal     ; Indicate literal
                000001FF  1AAA                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                      88  1AAE                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1AAF                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                      00  1AB1                .BYTE   0               ; 0 KA$L_FLAGS to data
                      20  1AB2                .BYTE   32              ; 32 of data field
                      86  1AB3                .BYTE   Pd$_Literal     ; Indicate literal
                00002710  1AB4                .LONG   10000           ; Constant
                      88  1AB8                .BYTE   Pd$_Store       ; Indicate store operation
                    003E  1AB9                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                      00  1ABB                .BYTE   0               ; 0 KA$W_LATENCY to data
                      20  1ABC                .BYTE   32              ; 32 of data field
                      86  1ABD                .BYTE   Pd$_Literal     ; Indicate literal
                000003E8  1ABE                .LONG   1000            ; Constant
                      88  1AC2                .BYTE   Pd$_Store       ; Indicate store operation
                    0040  1AC3                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                      00  1AC5                .BYTE   0               ; 0 KA$W_PROGRESS to data
                      20  1AC6                .BYTE   32              ; 32 of data field
                      81  1AC7                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1AC8    715 KA880:  $DS_KA880                                   ;
      30 38 38 41 4B 00'  1AC8                .ASCIC  "KA880" ; KA880 name
                      05  1AC8
                      37  1ACE                .BYTE   HP$K_LEN                ; HP$K_LEN of resultant P-table
                      02  1ACF                .BYTE   2               ; Maximum unit number                   ;G:65
                    0000  1AD0                .WORD   ^A""    ; Two character mnemonic for QIO KA880
                      80  1AD2                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1AD3                .Byte   Pd$_Name        ; Directive op-code
                      01  1AD4                .Byte   PTD$M_UNIT              ; Enforced KA codes
                   41 4B  1AD5                .Ascic  "KA"    ; Enforced device name prefix
                      02  1AD5
                      8B  1AD8                .BYTE   Pd$_Logical     ; Indicate logical value operation
79 72 61 6D 69 72 50 00'  1AD9                .ASCIC  \Primary\       ; Primary string
                      07  1AD9
                      88  1AE1                .BYTE   Pd$_Store       ; Indicate store operation
                    004B  1AE2                .WORD   HP$L_KAA_SLOT           ; Byte HP$L_KAA_SLOT to data
                      00  1AE4                .BYTE   0               ; 0 HP$L_KAA_SLOT to data
                      20  1AE5                .BYTE   32              ; 32 of data field
                      86  1AE6                .BYTE   Pd$_Literal     ; Indicate literal
                000001FF  1AE7                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                      88  1AEB                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1AEC                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                      00  1AEE                .BYTE   0               ; 0 KA$L_FLAGS to data
                      20  1AEF                .BYTE   32              ; 32 of data field
                      86  1AF0                .BYTE   Pd$_Literal     ; Indicate literal
                00002710  1AF1                .LONG   10000           ; Constant
                      88  1AF5                .BYTE   Pd$_Store       ; Indicate store operation
                    003E  1AF6                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                      00  1AF8                .BYTE   0               ; 0 KA$W_LATENCY to data
                      20  1AF9                .BYTE   32              ; 32 of data field
                      86  1AFA                .BYTE   Pd$_Literal     ; Indicate literal
                000003E8  1AFB                .LONG   1000            ; Constant
                      88  1AFF                .BYTE   Pd$_Store       ; Indicate store operation
                    0040  1B00                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
```

```
                          00  1B02                .BYTE   0               ; 0 KA$W_PROGRESS to data
                          20  1B03                .BYTE   32              ; 32 of data field
                          81  1B04                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              1B05    716 KA884:  $DS_KA884                       ;       [80]                    ;G:43A
                34 38 38 41 4B 00' 1B05           .ASCIC  "KA884" ; KA884 name
                          05  1B05
                          37  1B0B                .BYTE   HP$K_LEN                ; HP$K_LEN of resultant P-table
                          04  1B0C                .BYTE   4               ; Maximum unit number                   ;G:65
                        0000  1B0D                .WORD   ^A""    ; Two character mnemonic for QIO KA884
                          80  1B0F                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  1B10                .Byte   Pd$_Name        ; Directive op-code
                          01  1B11                .Byte   PTD$M_UNIT              ; Enforced KA codes
                       41 4B 00' 1B12             .Ascic  "KA"    ; Enforced device name prefix
                          02  1B12
                          8B  1B15                .BYTE   Pd$_Logical     ; Indicate logical value operation
             79 72 61 6D 69 72 50 00' 1B16        .ASCIC  \Primary\       ; Primary string
                          07  1B16
                          88  1B1E                .BYTE   Pd$_Store       ; Indicate store operation
                        004B  1B1F                .WORD   HP$L_KAA_SLOT           ; Byte HP$L_KAA_SLOT to data
                          00  1B21                .BYTE   0               ; 0 HP$L_KAA_SLOT to data
                          20  1B22                .BYTE   32              ; 32 of data field
                          82  1B23                .BYTE   Pd$_Decimal     ; Indicate scan decimal
          20 23 20 44 49 20 55 50 43 00' 1B24     .ASCIC  "CPU ID # "     ; CPU ID #  string
                          09  1B24
                    00000003  1B2E                .LONG   0,3     ; 0 , 3 limits on input
                          88  1B36                .BYTE   Pd$_Store       ; Indicate store operation
                        004F  1B37                .WORD   HP$B_CPU_ID             ; Byte HP$B_CPU_ID to data
                          00  1B39                .BYTE   0               ; 0 HP$B_CPU_ID to data
                          08  1B3A                .BYTE   8               ; 8 of data field
                          86  1B3B                .BYTE   Pd$_Literal     ; Indicate literal
                    000001FF  1B3C                .LONG   KA$M_F_FLOAT + KA$M_D_FLOAT +                   KA$M_CRC +
                          88  1B40                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1B41                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                          00  1B43                .BYTE   0               ; 0 KA$L_FLAGS to data
                          20  1B44                .BYTE   32              ; 32 of data field
                          86  1B45                .BYTE   Pd$_Literal     ; Indicate literal
                    00002710  1B46                .LONG   10000           ; Constant
                          88  1B4A                .BYTE   Pd$_Store       ; Indicate store operation
                        003E  1B4B                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                          00  1B4D                .BYTE   0               ; 0 KA$W_LATENCY to data
                          20  1B4E                .BYTE   32              ; 32 of data field
                          86  1B4F                .BYTE   Pd$_Literal     ; Indicate literal
                    000003E8  1B50                .LONG   1000            ; Constant
                          88  1B54                .BYTE   Pd$_Store       ; Indicate store operation
                        0040  1B55                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                          00  1B57                .BYTE   0               ; 0 KA$W_PROGRESS to data
                          20  1B58                .BYTE   32              ; 32 of data field
                          81  1B59                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              1B5A    717 KA9CC:  $DS_KA9CC                       ;       [83]                    ;G:37
                43 43 39 41 4B 00' 1B5A           .ASCIC  "KA9CC" ; KA9CC name
                          05  1B5A
                          50  1B60                .BYTE   KA$K_LEN                ; KA$K_LEN of resultant P-table
                          04  1B61                .BYTE   4               ; Maximum unit number                   ;G:65
                        0000  1B62                .WORD   ^A""    ; Two character mnemonic for QIO KA9CC
                          80  1B64                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          84  1B65                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
44 49 20 65 64 6F 4E 20 49 4D 58 00' 1B66         .ASCIC  "XMI Node ID"   ; XMI Node ID string
```

```
                          0B  1B66
                0000000F 00000000  1B72                .LONG   ^X0,^XF ; 0 , F limits on input
                          88  1B7A                .BYTE   Pd$_Store       ; Indicate store operation
                        0046  1B7B                .WORD   KA$B_NODE_ID            ; Byte KA$B_NODE_ID to data
                          00  1B7D                .BYTE   0               ; 0 KA$B_NODE_ID to data
                          08  1B7E                .BYTE   8               ; 8 of data field
                          88  1B7F                .BYTE   Pd$_Store       ; Indicate store operation
                        0018  1B80                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          13  1B82                .BYTE   19              ; 19 HP$A_DEVICE to data
                          04  1B83                .BYTE   4               ; 4 of data field
                          86  1B84                .BYTE   Pd$_Literal     ; Indicate literal
                    61800000  1B85                .LONG   ^X61800000              ; Constant
                          8A  1B89                .BYTE   Pd$_Add         ; Indicate add value to field operation
                        0018  1B8A                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  1B8C                .BYTE   0               ; 0 HP$A_DEVICE to data
                          20  1B8D                .BYTE   32              ; 32 of data field
                          86  1B8E                .BYTE   Pd$_Literal     ; Indicate literal
                    00010000  1B8F                .LONG   KA$M_TODR               ; Constant
                          88  1B93                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1B94                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                          00  1B96                .BYTE   0               ; 0 KA$L_FLAGS to data
                          20  1B97                .BYTE   32              ; 32 of data field
                          8B  1B98                .BYTE   Pd$_Logical     ; Indicate logical value operation
74 6E 65 73 65 72 50 20 55 50 46 00' 1B99                .ASCIC  \FPU Present\   ; FPU Present string
                          0B  1B99
                          88  1BA5                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1BA6                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                          00  1BA8                .BYTE   KA$V_F_FLOAT            ; KA$V_F_FLOAT KA$L_FLAGS to data
                          01  1BA9                .BYTE   1               ; 1 of data field
                          88  1BAA                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1BAB                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                          01  1BAD                .BYTE   KA$V_D_FLOAT            ; KA$V_D_FLOAT KA$L_FLAGS to data
                          01  1BAE                .BYTE   1               ; 1 of data field
                          88  1BAF                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  1BB0                .WORD   KA$L_FLAGS              ; Byte KA$L_FLAGS to data
                          02  1BB2                .BYTE   KA$V_G_FLOAT            ; KA$V_G_FLOAT KA$L_FLAGS to data
                          01  1BB3                .BYTE   1               ; 1 of data field
                          86  1BB4                .BYTE   Pd$_Literal     ; Indicate literal
                    00002710  1BB5                .LONG   10000           ; Constant
                          88  1BB9                .BYTE   Pd$_Store       ; Indicate store operation
                        003E  1BBA                .WORD   KA$W_LATENCY            ; Byte KA$W_LATENCY to data
                          00  1BBC                .BYTE   0               ; 0 KA$W_LATENCY to data
                          20  1BBD                .BYTE   32              ; 32 of data field
                          86  1BBE                .BYTE   Pd$_Literal     ; Indicate literal
                    000003E8  1BBF                .LONG   1000            ; Constant
                          88  1BC3                .BYTE   Pd$_Store       ; Indicate store operation
                        0040  1BC4                .WORD   KA$W_PROGRESS           ; Byte KA$W_PROGRESS to data
                          00  1BC6                .BYTE   0               ; 0 KA$W_PROGRESS to data
                          20  1BC7                .BYTE   32              ; 32 of data field
                          81  1BC8                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              1BC9    718 KDB50:  $DS_KDB50                                   ; BDA   [61]
                              1BC9                .SAVE   LOCAL_BLOCK
                              1BC9                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$L_KDB50_IP,       HP$L_KDB50_IP:
                              0032                .IIF    NB,KDB50$L_IP,  KDB50$L_IP:
                    00000036  0032                .IIF    NB,,BLKL,       .BLKL   1
```

```
                        0036                .IIF    NB,HP$B_KDB50_BR,       HP$B_KDB50_BR:
                        0036                .IIF    NB,KDB50$B_BR,  KDB50$B_BR:
                00000037  0036                .IIF    NB,,BLKB,       .BLKB   1
                        0037                .IIF    NB,HP$B_KDB50_NODE_ID,  HP$B_KDB50_NODE_ID:
                        0037                .IIF    NB,KDB50$B_NODE_ID,     KDB50$B_NODE_ID:
                00000038  0037                .IIF    NB,,BLKB,       .BLKB   1
                        0038                .IIF    NB,HP$B_KDB50_BURST,    HP$B_KDB50_BURST:
                        0038                .IIF    NB,KDB50$B_BURST,       KDB50$B_BURST:
                00000039  0038                .IIF    NB,,BLKB,       .BLKB   1
                        0039                .IIF    NB,HP$K_KDB50_LEN,      HP$K_KDB50_LEN:
                        0039                .IIF    NB,KDB50$K_LEN, KDB50$K_LEN:
                    00001BC9                .RESTORE
          30 35 42 44 4B 00'  1BC9          .ASCIC  "KDB50" ; KDB50 name
                        05  1BC9
                        39  1BCF          .BYTE   HP$K_KDB50_LEN          ; HP$K_KDB50_LEN of resultant P-table
                        00  1BD0          .BYTE   0               ; Maximum unit number                           ;G:65
                      0000  1BD1          .WORD   ^A""    ; Two character mnemonic for QIO KDB50
                        80  1BD3          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  1BD4          .Byte   Pd$_Name        ; Directive op-code
                        02  1BD5          .Byte   Ptd$M_Controller                ; Enforced DU codes
                55 44 00'  1BD6          .Ascic  "DU"    ; Enforced device name prefix
                        02  1BD6
                        87  1BD9          .BYTE   Pd$_Fetch       ; Indicate fetch operation
                      0018  1BDA          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        19  1BDC          .BYTE   25              ; 25 HP$A_DEVICE to data
                        07  1BDD          .BYTE   7               ; 7 of data field
                        8C  1BDE          .BYTE   Pd$_Case        ; Indicate case operation
                        01  1BDF          .BYTE   $$_$            ; Count of pairs
            00000030 00000000  1BE0          .LONG   0,^X30          ; Store the pair
                        88  1BE8          .BYTE   Pd$_Store       ; Indicate store operation
                      0018  1BE9          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        19  1BEB          .BYTE   25              ; 25 HP$A_DEVICE to data
                        07  1BEC          .BYTE   7               ; 7 of data field
                        88  1BED          .BYTE   Pd$_Store       ; Indicate store operation
                      001C  1BEE          .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                        19  1BF0          .BYTE   25              ; 25 HP$A_DVA to data
                        07  1BF1          .BYTE   7               ; 7 of data field
                        86  1BF2          .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1BF3          .LONG   ^X1             ; Constant
                        88  1BF7          .BYTE   Pd$_Store       ; Indicate store operation
                      001C  1BF8          .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                        16  1BFA          .BYTE   22              ; 22 HP$A_DVA to data
                        01  1BFB          .BYTE   1               ; 1 of data field
                        84  1BFC          .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'  1BFD          .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
      29 58 45 48 28 20 72 65 62  1C09
                        14  1BFD
            0000000F 00000000  1C12          .LONG   ^X0,^XF ; 0 . F limits on input
                        88  1C1A          .BYTE   Pd$_Store       ; Indicate store operation
                      0037  1C1B          .WORD   KDB50$B_NODE_ID         ; Byte KDB50$B_NODE_ID to data
                        00  1C1D          .BYTE   0               ; 0 KDB50$B_NODE_ID to data
                        08  1C1E          .BYTE   8               ; 8 of data field
                        88  1C1F          .BYTE   Pd$_Store       ; Indicate store operation
                      0018  1C20          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        0D  1C22          .BYTE   13              ; 13 HP$A_DEVICE to data
                        04  1C23          .BYTE   4               ; 4 of data field
                        88  1C24          .BYTE   Pd$_Store       ; Indicate store operation
```

```
                001C  1C25                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                  12  1C27                .BYTE   18                  ; 18 HP$A_DVA to data
                  04  1C28                .BYTE   4                   ; 4 of data field
                  88  1C29                .BYTE   Pd$_Store           ; Indicate store operation
                0024  1C2A                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                  02  1C2C                .BYTE   2                   ; 2 HP$W_VECTOR to data
                  04  1C2D                .BYTE   4                   ; 4 of data field
                  86  1C2E                .BYTE   Pd$_Literal         ; Indicate literal
            00000005  1C2F                .LONG   ^X5                 ; Constant
                  88  1C33                .BYTE   Pd$_Store           ; Indicate store operation
                0024  1C34                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                  06  1C36                .BYTE   6                   ; 6 HP$W_VECTOR to data
                  03  1C37                .BYTE   3                   ; 3 of data field
                  88  1C38                .BYTE   Pd$_Store           ; Indicate store operation
                0036  1C39                .WORD   KDB50$B_BR                  ; Byte KDB50$B_BR to data
                  00  1C3B                .BYTE   0                   ; 0 KDB50$B_BR to data
                  08  1C3C                .BYTE   8                   ; 8 of data field
                  87  1C3D                .BYTE   Pd$_Fetch           ; Indicate fetch operation
                0018  1C3E                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                  00  1C40                .BYTE   0                   ; 0 HP$A_DEVICE to data
                  20  1C41                .BYTE   32                  ; 32 of data field
                  88  1C42                .BYTE   Pd$_Store           ; Indicate store operation
                0032  1C43                .WORD   HP$L_KDB50_IP               ; Byte HP$L_KDB50_IP to data
                  00  1C45                .BYTE   0                   ; 0 HP$L_KDB50_IP to data
                  20  1C46                .BYTE   32                  ; 32 of data field
                  86  1C47                .BYTE   Pd$_Literal         ; Indicate literal
            000000F2  1C48                .LONG   ^XF2                ; Constant
                  88  1C4C                .BYTE   Pd$_Store           ; Indicate store operation
                0032  1C4D                .WORD   HP$L_KDB50_IP               ; Byte HP$L_KDB50_IP to data
                  00  1C4F                .BYTE   0                   ; 0 HP$L_KDB50_IP to data
                  08  1C50                .BYTE   8                   ; 8 of data field
                  86  1C51                .BYTE   Pd$_Literal         ; Indicate literal
            00000000  1C52                .LONG   ^X0                 ; Constant
                  88  1C56                .BYTE   Pd$_Store           ; Indicate store operation
                0038  1C57                .WORD   HP$B_KDB50_BURST                    ; Byte HP$B_KDB50_BURST to data
                  00  1C59                .BYTE   0                   ; 0 HP$B_KDB50_BURST to data
                  08  1C5A                .BYTE   8                   ; 8 of data field
                  81  1C5B                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                      1C5C    719 KFBTA:    $DS_KFBTA                                 ;       [68]
                      1C5C                .SAVE   LOCAL_BLOCK
                      1C5C                .PSECT  $ABS$,ABS
            00000032  0050                .=HP$A_DEPENDENT
                      0032                .IIF    NB,KFBTA$B_ID,  KFBTA$B_ID:
            00000033  0032                .IIF    NB,,BLKB,       .BLKB   1
                      0033                .IIF    NB,KFBTA$K_LEN, KFBTA$K_LEN:
                  0001C5C                 .RESTORE
   41 54 42 46 4B 00'  1C5C                .ASCIC  "KFBTA" ; KFBTA name
                  05  1C5C
                  33  1C62                .BYTE   KFBTA$K_LEN                 ; KFBTA$K_LEN of resultant P-table
                  00  1C63                .BYTE   0                   ; Maximum unit number                           ;G:65
                0000  1C64                .WORD   ^A""    ; Two character mnemonic for QIO KFBTA
                  80  1C66                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                  8D  1C67                .Byte   Pd$_Name            ; Directive op-code
                  02  1C68                .Byte   Ptd$M_Controller                ; Enforced DU codes
            55 44 00'  1C69               .Ascic  "DU"    ; Enforced device name prefix
                  02  1C69
                  87  1C6C                .BYTE   Pd$_Fetch           ; Indicate fetch operation
```

```
                          0018  1C6D          .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                            19  1C6F          .BYTE   25                  ; 25 HP$A_DEVICE to data
                            07  1C70          .BYTE   7                   ; 7 of data field
                            8C  1C71          .BYTE   Pd$_Case            ; Indicate case operation
                            01  1C72          .BYTE   $$_$                ; Count of pairs
                 00000030 00000000  1C73          .LONG   0,^X30              ; Store the pair
                            88  1C7B          .BYTE   Pd$_Store           ; Indicate store operation
                          0018  1C7C          .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                            19  1C7E          .BYTE   25                  ; 25 HP$A_DEVICE to data
                            07  1C7F          .BYTE   7                   ; 7 of data field
                            88  1C80          .BYTE   Pd$_Store           ; Indicate store operation
                          001C  1C81          .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                            19  1C83          .BYTE   25                  ; 25 HP$A_DVA to data
                            07  1C84          .BYTE   7                   ; 7 of data field
                            86  1C85          .BYTE   Pd$_Literal         ; Indicate literal
                      00000001  1C86          .LONG   ^X1                 ; Constant
                            88  1C8A          .BYTE   Pd$_Store           ; Indicate store operation
                          001C  1C8B          .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                            16  1C8D          .BYTE   22                  ; 22 HP$A_DVA to data
                            01  1C8E          .BYTE   1                   ; 1 of data field
                            84  1C8F          .BYTE   Pd$_Hexadecimal     ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'  1C90          .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
        29 58 45 48 28 20 72 65 62  1C9C
                            14  1C90
                 0000000F 00000000  1CA5          .LONG   ^X0,^XF ; 0 , F limits on input
                            88  1CAD          .BYTE   Pd$_Store           ; Indicate store operation
                          000B  1CAE          .WORD   HP$B_DRIVE                  ; Byte HP$B_DRIVE to data
                            00  1CB0          .BYTE   0                   ; 0 HP$B_DRIVE to data
                            08  1CB1          .BYTE   8                   ; 8 of data field
                            88  1CB2          .BYTE   Pd$_Store           ; Indicate store operation
                          0032  1CB3          .WORD   KFBTA$B_ID                  ; Byte KFBTA$B_ID to data
                            00  1CB5          .BYTE   0                   ; 0 KFBTA$B_ID to data
                            08  1CB6          .BYTE   8                   ; 8 of data field
                            88  1CB7          .BYTE   Pd$_Store           ; Indicate store operation
                          0018  1CB8          .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                            0D  1CBA          .BYTE   13                  ; 13 HP$A_DEVICE to data
                            04  1CBB          .BYTE   4                   ; 4 of data field
                            88  1CBC          .BYTE   Pd$_Store           ; Indicate store operation
                          001C  1CBD          .WORD   HP$A_DVA                    ; Byte HP$A_DVA to data
                            12  1CBF          .BYTE   18                  ; 18 HP$A_DVA to data
                            04  1CC0          .BYTE   4                   ; 4 of data field
                            81  1CC1          .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                                1CC2  720 KMC11:  $DS_KMC11
                                1CC2          .SAVE   LOCAL_BLOCK
                                1CC2          .PSECT  $ABS$,ABS
                      00000032  0050          .=HP$A_DEPENDENT
                                0032          .IIF    NB,KMC11$L_CSR, KMC11$L_CSR:
                      00000036  0032          .IIF    NB,.BLKL,       .BLKL   1
                                0036          .IIF    NB,KMC11$B_BR,  KMC11$B_BR:
                      00000037  0036          .IIF    NB,.BLKB,       .BLKB   1
                                0037          .IIF    NB,KMC11$K_LEN, KMC11$K_LEN:
                          00001CC2          .RESTORE
                 31 31 43 4D 4B 00'  1CC2          .ASCIC  "KMC11" ; KMC11 name
                            05  1CC2
                            37  1CC8          .BYTE   KMC11$K_LEN                 ; KMC11$K_LEN of resultant P-table
                            00  1CC9          .BYTE   0                   ; Maximum unit number                    ;G:65
                          0000  1CCA          .WORD   ^A""    ; Two character mnemonic for QIO KMC11
```

```
                        80  1CCC                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  1CCD                .Byte   Pd$_Name        ; Directive op-code
                        03  1CCE                .Byte   Ptd$M_Device            ; Enforced XK codes
                  4B 58 00' 1CCF                .Ascic  "XK"    ; Enforced device name prefix
                        02  1CCF
                        83  1CD2                .BYTE   Pd$_Octal       ; Indicate scan octal
                  52 53 43 00' 1CD3             .ASCIC  "CSR"   ; CSR string
                        03  1CD3
              0003FFFE 0003E000  1CD7           .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  1CDF                .BYTE   Pd$_Store       ; Indicate store operation
                      0032  1CE0                .WORD   KMC11$L_CSR             ; Byte KMC11$L_CSR to data
                        00  1CE2                .BYTE   0               ; 0 KMC11$L_CSR to data
                        20  1CE3                .BYTE   32              ; 32 of data field
                        88  1CE4                .BYTE   Pd$_Store       ; Indicate store operation
                      0018  1CE5                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                        00  1CE7                .BYTE   0               ; 0 HP$A_DEVICE to data
                        12  1CE8                .BYTE   18              ; 18 of data field
                        83  1CE9                .BYTE   Pd$_Octal       ; Indicate scan octal
           52 4F 54 43 45 56 00' 1CEA           .ASCIC  "VECTOR"        ; VECTOR string
                        06  1CEA
              000001FE 00000002  1CF1           .LONG   ^O2,^O776       ; 2 , 776 limits on input
                        88  1CF9                .BYTE   Pd$_Store       ; Indicate store operation
                      0024  1CFA                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                        00  1CFC                .BYTE   0               ; 0 HP$W_VECTOR to data
                        09  1CFD                .BYTE   9               ; 9 of data field
                        82  1CFE                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                  52 42 00' 1CFF                .ASCIC  "BR"    ; BR string
                        02  1CFF
              00000007 00000004  1D02           .LONG   4,7     ; 4 , 7 limits on input
                        88  1D0A                .BYTE   Pd$_Store       ; Indicate store operation
                      0036  1D0B                .WORD   KMC11$B_BR              ; Byte KMC11$B_BR to data
                        00  1D0D                .BYTE   0               ; 0 KMC11$B_BR to data
                        08  1D0E                .BYTE   8               ; 8 of data field
                        81  1D0F                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            1D10    721 KW11K:  $DS_KW11K
                            1D10                .SAVE   LOCAL_BLOCK
                            1D10                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,KW11K$B_BR,  KW11K$B_BR:
                  00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                            0033                .IIF    NB,KW11K$K_LEN, KW11K$K_LEN:
                        0001D10                 .RESTORE
              4B 31 31 57 4B 00' 1D10           .ASCIC  "KW11K" ; KW11K name
                        05  1D10
                        33  1D16                .BYTE   KW11K$K_LEN             ; KW11K$K_LEN of resultant P-table
                        02  1D17                .BYTE   2               ; Maximum unit number                   ;G:65
                      0000  1D18                .WORD   ^A""    ; Two character mnemonic for QIO KW11K
                        80  1D1A                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  1D1B                .Byte   Pd$_Name        ; Directive op-code
                        03  1D1C                .Byte   Ptd$M_Device            ; Enforced KW codes
                  57 4B 00' 1D1D                .Ascic  "KW"    ; Enforced device name prefix
                        02  1D1D
                        83  1D20                .BYTE   Pd$_Octal       ; Indicate scan octal
52 53 43 20 53 55 42 20 4F 2F 49 00' 1D21       .ASCIC  "I/O BUS CSR"   ; I/O BUS CSR string
                        0B  1D21
              0003FFFE 0003E000  1D2D           .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  1D35                .BYTE   Pd$_Store       ; Indicate store operation
```

```
                     0018  1D36                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  1D38                .BYTE   0                 ; 0 HP$A_DEVICE to data
                       12  1D39                .BYTE   18                ; 18 of data field
                       83  1D3A                .BYTE   Pd$_Octal         ; Indicate scan octal
          52 4F 54 43 45 56 00' 1D3B           .ASCIC  "VECTOR"          ; VECTOR string
                       06  1D3B
             000001FE 00000002  1D42           .LONG   ^O2,^O776         ; 2 , 776 limits on input
                       88  1D4A                .BYTE   Pd$_Store         ; Indicate store operation
                     0024  1D4B                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                       00  1D4D                .BYTE   0                 ; 0 HP$W_VECTOR to data
                       09  1D4E                .BYTE   9                 ; 9 of data field
                       82  1D4F                .BYTE   Pd$_Decimal       ; Indicate scan decimal
                    52 42 00' 1D50             .ASCIC  "BR"     ; BR string
                       02  1D50
             00000007 00000004  1D53           .LONG   4,7      ; 4 , 7 limits on input
                       88  1D5B                .BYTE   Pd$_Store         ; Indicate store operation
                     0032  1D5C                .WORD   KW11K$B_BR                  ; Byte KW11K$B_BR to data
                       00  1D5E                .BYTE   0                 ; 0 KW11K$B_BR to data
                       08  1D5F                .BYTE   8                 ; 8 of data field
                       81  1D60                .BYTE   Pd$_End           ; Indicate end of PT-descriptor
                           1D61    722 LA12:    $DS_LA12                                ;       [35]
                           1D61                .SAVE   LOCAL_BLOCK
                           1D61                .PSECT  $ABS$,ABS
                    00000032  0050             .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_LA12_LEN,       HP$K_LA12_LEN:
                           0032                .IIF    NB,LA12$K_LEN,  LA12$K_LEN:
                       00001D61                .RESTORE
                32 31 41 4C 00' 1D61           .ASCIC  "LA12"   ; LA12 name
                       04  1D61
                       32  1D66                .BYTE   LA12$K_LEN                  ; LA12$K_LEN of resultant P-table
                       FF  1D67                .BYTE   255               ; Maximum unit number                         ;G:65
                     0000  1D68                .WORD   ^A""      ; Two character mnemonic for QIO LA12
                       80  1D6A                .BYTE   Pd$_Start         ; Constant to identify start of descriptor
                       8D  1D6B                .Byte   Pd$_Name          ; Directive op-code
                       1B  1D6C                .Byte   Ptd$M_EndDevice           ; Enforced TT codes
                    54 54 00' 1D6D             .Ascic  "TT"     ; Enforced device name prefix
                       02  1D6D
                       86  1D70                .BYTE   Pd$_Literal       ; Indicate literal
                 00000001  1D71                .LONG   HP$M_ALLOC                  ; Constant
                       88  1D75                .BYTE   Pd$_Store         ; Indicate store operation
                     000A  1D76                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                       00  1D78                .BYTE   0                 ; 0 HP$B_FLAGS to data
                       08  1D79                .BYTE   8                 ; 8 of data field
                       81  1D7A                .BYTE   Pd$_End           ; Indicate end of PT-descriptor
                           1D7B    723 LA34:    $DS_LA34
                           1D7B                .SAVE   LOCAL_BLOCK
                           1D7B                .PSECT  $ABS$,ABS
                    00000032  0050             .=HP$A_DEPENDENT
                           0032                .IIF    NB,LA34$K_LEN,  LA34$K_LEN:
                       00001D7B                .RESTORE
                34 33 41 4C 00' 1D7B           .ASCIC  "LA34"   ; LA34 name
                       04  1D7B
                       32  1D80                .BYTE   LA34$K_LEN                  ; LA34$K_LEN of resultant P-table
                       FF  1D81                .BYTE   255               ; Maximum unit number                         ;G:65
                     0000  1D82                .WORD   ^A""      ; Two character mnemonic for QIO LA34
                       80  1D84                .BYTE   Pd$_Start         ; Constant to identify start of descriptor
                       8D  1D85                .Byte   Pd$_Name          ; Directive op-code
```

```
                      1B  1D86                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 1D87                .Ascic  "TT"    ; Enforced device name prefix
                      02  1D87
                      86  1D8A                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  1D8B                .LONG   HP$M_ALLOC               ; Constant
                      88  1D8F                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  1D90                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  1D92                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  1D93                .BYTE   8                ; 8 of data field
                      81  1D94                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          1D95    724 LA36:   $DS_LA36
                          1D95                .SAVE   LOCAL_BLOCK
                          1D95                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LA36$K_LEN,  LA36$K_LEN:
                    0001D95                   .RESTORE
             36 33 41 4C 00' 1D95             .ASCIC  "LA36"  ; LA36 name
                      04  1D95
                      32  1D9A                .BYTE   LA36$K_LEN               ; LA36$K_LEN of resultant P-table
                      FF  1D9B                .BYTE   255              ; Maximum unit number                          ;G:65
                    0000  1D9C                .WORD   ^A""     ; Two character mnemonic for QIO LA36
                      80  1D9E                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  1D9F                .Byte   Pd$_Name         ; Directive op-code
                      1B  1DA0                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 1DA1                .Ascic  "TT"    ; Enforced device name prefix
                      02  1DA1
                      86  1DA4                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  1DA5                .LONG   HP$M_ALLOC               ; Constant
                      88  1DA9                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  1DAA                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  1DAC                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  1DAD                .BYTE   8                ; 8 of data field
                      81  1DAE                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          1DAF    725 LA38:   $DS_LA38
                          1DAF                .SAVE   LOCAL_BLOCK
                          1DAF                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LA38$K_LEN,  LA38$K_LEN:
                    0001DAF                   .RESTORE
             38 33 41 4C 00' 1DAF             .ASCIC  "LA38"  ; LA38 name
                      04  1DAF
                      32  1DB4                .BYTE   LA38$K_LEN               ; LA38$K_LEN of resultant P-table
                      FF  1DB5                .BYTE   255              ; Maximum unit number                          ;G:65
                    0000  1DB6                .WORD   ^A""     ; Two character mnemonic for QIO LA38
                      80  1DB8                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  1DB9                .Byte   Pd$_Name         ; Directive op-code
                      1B  1DBA                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 1DBB                .Ascic  "TT"    ; Enforced device name prefix
                      02  1DBB
                      86  1DBE                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  1DBF                .LONG   HP$M_ALLOC               ; Constant
                      88  1DC3                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  1DC4                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  1DC6                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  1DC7                .BYTE   8                ; 8 of data field
                      81  1DC8                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          1DC9    726 LA100:  $DS_LA100                                ;       [35]
```

```
                          1DC9                  .SAVE   LOCAL_BLOCK
                          1DC9                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,HP$K_LA100_LEN,      HP$K_LA100_LEN:
                          0032                  .IIF    NB,LA100$K_LEN, LA100$K_LEN:
                      00001DC9                  .RESTORE
          30 30 31 41 4C 00' 1DC9               .ASCIC  "LA100" ; LA100 name
                          05  1DC9
                          32  1DCF              .BYTE   LA100$K_LEN             ; LA100$K_LEN of resultant P-table
                          FF  1DD0              .BYTE   255             ; Maximum unit number                   ;G:65
                        0000  1DD1              .WORD   ^A""    ; Two character mnemonic for QIO LA100
                          80  1DD3              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  1DD4              .Byte   Pd$_Name        ; Directive op-code
                          1B  1DD5              .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                54 54 00' 1DD6                .Ascic  "TT"    ; Enforced device name prefix
                          02  1DD6
                          86  1DD9              .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1DDA                .LONG   HP$M_ALLOC              ; Constant
                          88  1DDE              .BYTE   Pd$_Store       ; Indicate store operation
                        000A  1DDF              .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  1DE1              .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  1DE2              .BYTE   8               ; 8 of data field
                          81  1DE3              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1DE4    727 LA120:    $DS_LA120
                          1DE4                  .SAVE   LOCAL_BLOCK
                          1DE4                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,LA120$K_LEN, LA120$K_LEN:
                      00001DE4                  .RESTORE
          30 32 31 41 4C 00' 1DE4               .ASCIC  "LA120" ; LA120 name
                          05  1DE4
                          32  1DEA              .BYTE   LA120$K_LEN             ; LA120$K_LEN of resultant P-table
                          FF  1DEB              .BYTE   255             ; Maximum unit number                   ;G:65
                        0000  1DEC              .WORD   ^A""    ; Two character mnemonic for QIO LA120
                          80  1DEE              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  1DEF              .Byte   Pd$_Name        ; Directive op-code
                          1B  1DF0              .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                54 54 00' 1DF1                .Ascic  "TT"    ; Enforced device name prefix
                          02  1DF1
                          86  1DF4              .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1DF5                .LONG   HP$M_ALLOC              ; Constant
                          88  1DF9              .BYTE   Pd$_Store       ; Indicate store operation
                        000A  1DFA              .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  1DFC              .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  1DFD              .BYTE   8               ; 8 of data field
                          81  1DFE              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1DFF    728 LA180:    $DS_LA180
                          1DFF                  .SAVE   LOCAL_BLOCK
                          1DFF                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,LA180$K_LEN, LA180$K_LEN:
                      00001DFF                  .RESTORE
          30 38 31 41 4C 00' 1DFF               .ASCIC  "LA180" ; LA180 name
                          05  1DFF
                          32  1E05              .BYTE   LA180$K_LEN             ; LA180$K_LEN of resultant P-table
                          FF  1E06              .BYTE   255             ; Maximum unit number                   ;G:65
                        0000  1E07              .WORD   ^A""    ; Two character mnemonic for QIO LA180
```

```
                      80  1E09                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1E0A                .Byte   Pd$_Name        ; Directive op-code
                      1B  1E0B                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                50 4C 00' 1E0C                .Ascic  "LP"    ; Enforced device name prefix
                      02  1E0C
                      86  1E0F                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1E10                .LONG   HP$M_ALLOC              ; Constant
                      88  1E14                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1E15                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1E17                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1E18                .BYTE   8               ; 8 of data field
                      81  1E19                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1E1A    729 LA210:  $DS_LA210                               ;       [66]
                          1E1A                .SAVE   LOCAL_BLOCK
                          1E1A                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_LA210_LEN,      HP$K_LA210_LEN:
                          0032                .IIF    NB,LA210$K_LEN, LA210$K_LEN:
                      0000 1E1A               .RESTORE
          30 31 32 41 4C 00' 1E1A               .ASCIC  "LA210" ; LA210 name
                      05  1E1A
                      32  1E20                .BYTE   LA210$K_LEN             ; LA210$K_LEN of resultant P-table
                      08  1E21                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  1E22                .WORD   ^A""    ; Two character mnemonic for QIO LA210
                      80  1E24                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1E25                .Byte   Pd$_Name        ; Directive op-code
                      1B  1E26                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                54 54 00' 1E27                .Ascic  "TT"    ; Enforced device name prefix
                      02  1E27
                      86  1E2A                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1E2B                .LONG   HP$M_ALLOC              ; Constant
                      88  1E2F                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1E30                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1E32                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1E33                .BYTE   8               ; 8 of data field
                      81  1E34                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1E35    730 LANCE:  $DS_LANCE                               ;       [68]
                          1E35                .SAVE   LOCAL_BLOCK
                          1E35                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LANCE$K_LEN, LANCE$K_LEN:
                      0000 1E35               .RESTORE
       45 43 4E 41 4C 00' 1E35                .ASCIC  "LANCE" ; LANCE name
                      05  1E35
                      32  1E3B                .BYTE   LANCE$K_LEN             ; LANCE$K_LEN of resultant P-table
                      01  1E3C                .BYTE   1               ; Maximum unit number                   ;G:65
                    0000  1E3D                .WORD   ^A""    ; Two character mnemonic for QIO LANCE
                      80  1E3F                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1E40                .Byte   Pd$_Name        ; Directive op-code
                      03  1E41                .Byte   PTD$M_Device            ; Enforced ET codes
                54 45 00' 1E42                .Ascic  "ET"    ; Enforced device name prefix
                      02  1E42
                      81  1E45                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1E46    731 LESI:   $DS_LESI                                ;       [34]
                          1E46                .SAVE   LOCAL_BLOCK
                          1E46                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
```

```
                        0032                .IIF    NB,HP$L_LESI_IP,        HP$L_LESI_IP:
                        0032                .IIF    NB,LESI$L_IP,   LESI$L_IP:
                00000036  0032                .IIF    NB,,BLKL,       .BLKL   1
                        0036                .IIF    NB,HP$B_LESI_BR,        HP$B_LESI_BR:
                        0036                .IIF    NB,LESI$B_BR,   LESI$B_BR:
                00000037  0036                .IIF    NB,,BLKB,       .BLKB   1
                        0037                .IIF    NB,HP$K_LESI_LEN,       HP$K_LESI_LEN:
                        0037                .IIF    NB,LESI$K_LEN,  LESI$K_LEN:
                    00001E46                .RESTORE
             49 53 45 4C 00' 1E46           .ASCIC  "LESI"  ; LESI name
                      04  1E46
                      37  1E4B              .BYTE   HP$K_LESI_LEN           ; HP$K_LESI_LEN of resultant P-table
                      00  1E4C              .BYTE   0               ; Maximum unit number                   ;G:65
                    0000  1E4D              .WORD   ^A""    ; Two character mnemonic for QIO LESI
                      80  1E4F              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1E50              .Byte   Pd$_Name        ; Directive op-code
                      02  1E51              .Byte   Ptd$M_Controller                ; Enforced DA codes
                41 44 00' 1E52              .Ascic  "DA"    ; Enforced device name prefix
                      02  1E52
                      83  1E55              .BYTE   Pd$_Octal       ; Indicate scan octal
                50 49 00' 1E56              .ASCIC  "IP"    ; IP string
                      02  1E56
            0003FFFC 0003E000  1E59          .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                      88  1E61              .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1E62              .WORD   HP$L_LESI_IP            ; Byte HP$L_LESI_IP to data
                      00  1E64              .BYTE   0               ; 0 HP$L_LESI_IP to data
                      20  1E65              .BYTE   32              ; 32 of data field
                      88  1E66              .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1E67              .WORD   Hp$A_Device             ; Byte Hp$A_Device to data
                      00  1E69              .BYTE   0               ; 0 Hp$A_Device to data
                      12  1E6A              .BYTE   18              ; 18 of data field
                      83  1E6B              .BYTE   Pd$_Octal       ; Indicate scan octal
         72 6F 74 63 65 56 00' 1E6C         .ASCIC  "Vector"        ; Vector string
                      06  1E6C
            000001FC 00000004  1E73          .LONG   ^O4,^O774       ; 4 , 774 limits on input
                      88  1E7B              .BYTE   Pd$_Store       ; Indicate store operation
                    0024  1E7C              .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  1E7E              .BYTE   0               ; 0 HP$W_VECTOR to data
                      08  1E7F              .BYTE   8               ; 8 of data field
                      82  1E80              .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 1E81              .ASCIC  "BR"    ; BR string
                      02  1E81
            00000007 00000004  1E84          .LONG   4,7     ; 4 , 7 limits on input
                      88  1E8C              .BYTE   Pd$_Store       ; Indicate store operation
                    0036  1E8D              .WORD   HP$B_LESI_BR            ; Byte HP$B_LESI_BR to data
                      00  1E8F              .BYTE   0               ; 0 HP$B_LESI_BR to data
                      08  1E90              .BYTE   8               ; 8 of data field
                      81  1E91              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1E92  732 LG01:     $DS_LG01                                ;       [79]            ;G:27
                          1E92              .SAVE   LOCAL_BLOCK
                          1E92              .PSECT  $ABS$,ABS
                00000032  0050              .=HP$A_DEPENDENT
                          0032              .IIF    NB,LG01$K_LEN,  LG01$K_LEN:
                    00001E92                .RESTORE
             31 30 47 4C 00' 1E92           .ASCIC  "LG01"  ; LG01 name
                      04  1E92
                      32  1E97              .BYTE   LG01$K_LEN              ; LG01$K_LEN of resultant P-table
```

```
                01  1E98                .BYTE   1                   ; Maximum unit number                    ;G:65
              0000  1E99                .WORD   ^A""      ; Two character mnemonic for QIO LG01
                80  1E9B                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                8D  1E9C                .Byte   Pd$_Name            ; Directive op-code
                1B  1E9D                .Byte   Ptd$M_EndDevice             ; Enforced LG codes
          47 4C 00' 1E9E                .Ascic  "LG"      ; Enforced device name prefix
                02  1E9E
                86  1EA1                .BYTE   Pd$_Literal         ; Indicate literal
          00000001  1EA2                .LONG   HP$M_ALLOC                  ; Constant
                88  1EA6                .BYTE   Pd$_Store           ; Indicate store operation
              000A  1EA7                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                00  1EA9                .BYTE   0                   ; 0 HP$B_FLAGS to data
                08  1EAA                .BYTE   8                   ; 8 of data field
                81  1EAB                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                    1EAC    733 LG02:   $DS_LG02                                ;       [79]                ;G:27
                    1EAC                .SAVE   LOCAL_BLOCK
                    1EAC                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                    0032                .IIF    NB,LG02$K_LEN,  LG02$K_LEN:
                00001EAC                .RESTORE
    32 30 47 4C 00' 1EAC                .ASCIC  "LG02"  ; LG02 name
                04  1EAC
                32  1EB1                .BYTE   LG02$K_LEN                  ; LG02$K_LEN of resultant P-table
                01  1EB2                .BYTE   1                   ; Maximum unit number                    ;G:65
              0000  1EB3                .WORD   ^A""      ; Two character mnemonic for QIO LG02
                80  1EB5                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                8D  1EB6                .Byte   Pd$_Name            ; Directive op-code
                1B  1EB7                .Byte   Ptd$M_EndDevice             ; Enforced LG codes
          47 4C 00' 1EB8                .Ascic  "LG"      ; Enforced device name prefix
                02  1EB8
                86  1EBB                .BYTE   Pd$_Literal         ; Indicate literal
          00000001  1EBC                .LONG   HP$M_ALLOC                  ; Constant
                88  1EC0                .BYTE   Pd$_Store           ; Indicate store operation
              000A  1EC1                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                00  1EC3                .BYTE   0                   ; 0 HP$B_FLAGS to data
                08  1EC4                .BYTE   8                   ; 8 of data field
                81  1EC5                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                    1EC6    734 LG03:   $DS_LG03                                ;       [79]                ;G:27
                    1EC6                .SAVE   LOCAL_BLOCK
                    1EC6                .PSECT  $ABS$,ABS
          00000032  0050                .=HP$A_DEPENDENT
                    0032                .IIF    NB,LG03$K_LEN,  LG03$K_LEN:
                00001EC6                .RESTORE
    33 30 47 4C 00' 1EC6                .ASCIC  "LG03"  ; LG03 name
                04  1EC6
                32  1ECB                .BYTE   LG03$K_LEN                  ; LG03$K_LEN of resultant P-table
                01  1ECC                .BYTE   1                   ; Maximum unit number                    ;G:65
              0000  1ECD                .WORD   ^A""      ; Two character mnemonic for QIO LG03
                80  1ECF                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                8D  1ED0                .Byte   Pd$_Name            ; Directive op-code
                1B  1ED1                .Byte   Ptd$M_EndDevice             ; Enforced LG codes
          47 4C 00' 1ED2                .Ascic  "LG"      ; Enforced device name prefix
                02  1ED2
                86  1ED5                .BYTE   Pd$_Literal         ; Indicate literal
          00000001  1ED6                .LONG   HP$M_ALLOC                  ; Constant
                88  1EDA                .BYTE   Pd$_Store           ; Indicate store operation
              000A  1EDB                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
```

```
                        00  1EDD                .BYTE   0                       ; 0 HP$B_FLAGS to data
                        08  1EDE                .BYTE   8                       ; 8 of data field
                        81  1EDF                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                            1EE0    735 LN01:   $DS_LN01                                        ;       [35]
                            1EE0                .SAVE   LOCAL_BLOCK
                            1EE0                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,LN01$K_LEN,  LN01$K_LEN:
                        00001EE0                .RESTORE
                 31 30 4E 4C 00' 1EE0           .ASCIC  "LN01"  ; LN01 name
                        04  1EE0
                        32  1EE5                .BYTE   LN01$K_LEN              ; LN01$K_LEN of resultant P-table
                        FF  1EE6                .BYTE   255             ; Maximum unit number                   ;G:65
                      0000  1EE7                .WORD   ^A""    ; Two character mnemonic for QIO LN01
                        80  1EE9                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        8D  1EEA                .Byte   Pd$_Name        ; Directive op-code
                        1B  1EEB                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                     50 4C 00' 1EEC             .Ascic  "LP"    ; Enforced device name prefix
                        02  1EEC
                        86  1EEF                .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  1EF0              .LONG   HP$M_ALLOC              ; Constant
                        88  1EF4                .BYTE   Pd$_Store       ; Indicate store operation
                      000A  1EF5                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                        00  1EF7                .BYTE   0                       ; 0 HP$B_FLAGS to data
                        08  1EF8                .BYTE   8                       ; 8 of data field
                        81  1EF9                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                            1EFA    736 LN03:   $DS_LN03                                        ;       [66]
                            1EFA                .SAVE   LOCAL_BLOCK
                            1EFA                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,LN03$K_LEN,  LN03$K_LEN:
                        00001EFA                .RESTORE
                 33 30 4E 4C 00' 1EFA           .ASCIC  "LN03"  ; LN03 name
                        04  1EFA
                        32  1EFF                .BYTE   LN03$K_LEN              ; LN03$K_LEN of resultant P-table
                        FF  1F00                .BYTE   255             ; Maximum unit number                   ;G:65
                      0000  1F01                .WORD   ^A""    ; Two character mnemonic for QIO LN03
                        80  1F03                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                        86  1F04                .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  1F05              .LONG   HP$M_ALLOC              ; Constant
                        88  1F09                .BYTE   Pd$_Store       ; Indicate store operation
                      000A  1F0A                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                        00  1F0C                .BYTE   0                       ; 0 HP$B_FLAGS to data
                        08  1F0D                .BYTE   8                       ; 8 of data field
                        81  1F0E                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                            1F0F    737 LP04:   $DS_LP04
                            1F0F                .SAVE   LOCAL_BLOCK
                            1F0F                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,LP04$K_LEN,  LP04$K_LEN:
                        00001F0F                .RESTORE
                 34 30 50 4C 00' 1F0F           .ASCIC  "LP04"  ; LP04 name
                        04  1F0F
                        32  1F14                .BYTE   LP04$K_LEN              ; LP04$K_LEN of resultant P-table
                        01  1F15                .BYTE   1               ; Maximum unit number                   ;G:65
                      0000  1F16                .WORD   ^A""    ; Two character mnemonic for QIO LP04
                        80  1F18                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
```

```
                      8D  1F19                .Byte   Pd$_Name         ; Directive op-code
                      1B  1F1A                .Byte   Ptd$M_EndDevice          ; Enforced LP codes
                50 4C 00' 1F1B                .Ascic  "LP"    ; Enforced device name prefix
                      02  1F1B
                      86  1F1E                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  1F1F                .LONG   HP$M_ALLOC               ; Constant
                      88  1F23                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  1F24                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  1F26                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  1F27                .BYTE   8                ; 8 of data field
                      81  1F28                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          1F29    738 LP05:   $DS_LP05
                          1F29                .SAVE   LOCAL_BLOCK
                          1F29                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LP05$K_LEN,  LP05$K_LEN:
                      0000 1F29               .RESTORE
             35 30 50 4C 00' 1F29             .ASCIC  "LP05"  ; LP05 name
                      04  1F29
                      32  1F2E                .BYTE   LP05$K_LEN               ; LP05$K_LEN of resultant P-table
                      01  1F2F                .BYTE   1                ; Maximum unit number                          ;G:65
                    0000  1F30                .WORD   ^A""    ; Two character mnemonic for QIO LP05
                      80  1F32                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  1F33                .Byte   Pd$_Name         ; Directive op-code
                      1B  1F34                .Byte   Ptd$M_EndDevice          ; Enforced LP codes
                50 4C 00' 1F35                .Ascic  "LP"    ; Enforced device name prefix
                      02  1F35
                      86  1F38                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  1F39                .LONG   HP$M_ALLOC               ; Constant
                      88  1F3D                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  1F3E                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  1F40                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  1F41                .BYTE   8                ; 8 of data field
                      81  1F42                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          1F43    739 LP06:   $DS_LP06
                          1F43                .SAVE   LOCAL_BLOCK
                          1F43                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LP06$K_LEN,  LP06$K_LEN:
                      0000 1F43               .RESTORE
             36 30 50 4C 00' 1F43             .ASCIC  "LP06"  ; LP06 name
                      04  1F43
                      32  1F48                .BYTE   LP06$K_LEN               ; LP06$K_LEN of resultant P-table
                      01  1F49                .BYTE   1                ; Maximum unit number                          ;G:65
                    0000  1F4A                .WORD   ^A""    ; Two character mnemonic for QIO LP06
                      80  1F4C                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  1F4D                .Byte   Pd$_Name         ; Directive op-code
                      1B  1F4E                .Byte   Ptd$M_EndDevice          ; Enforced LP codes
                50 4C 00' 1F4F                .Ascic  "LP"    ; Enforced device name prefix
                      02  1F4F
                      86  1F52                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  1F53                .LONG   HP$M_ALLOC               ; Constant
                      88  1F57                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  1F58                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  1F5A                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  1F5B                .BYTE   8                ; 8 of data field
                      81  1F5C                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
```

```
                          1F5D   740 LP07:   $DS_LP07
                          1F5D               .SAVE   LOCAL_BLOCK
                          1F5D               .PSECT  $ABS$,ABS
                00000032  0050               .=HP$A_DEPENDENT
                          0032               .IIF    NB,LP07$K_LEN,  LP07$K_LEN:
                      00001F5D               .RESTORE
          37 30 50 4C 00' 1F5D               .ASCIC  "LP07"  ; LP07 name
                      04  1F5D
                      32  1F62               .BYTE   LP07$K_LEN              ; LP07$K_LEN of resultant P-table
                      01  1F63               .BYTE   1               ; Maximum unit number                    ;G:65
                    0000  1F64               .WORD   ^A""    ; Two character mnemonic for QIO LP07
                      80  1F66               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1F67               .Byte   Pd$_Name        ; Directive op-code
                      1B  1F68               .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                50 4C 00' 1F69               .Ascic  "LP"    ; Enforced device name prefix
                      02  1F69
                      86  1F6C               .BYTE   Pd$_Literal     ; Indicate literal
                00000001  1F6D               .LONG   HP$M_ALLOC              ; Constant
                      88  1F71               .BYTE   Pd$_Store       ; Indicate store operation
                    000A  1F72               .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1F74               .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  1F75               .BYTE   8               ; 8 of data field
                      81  1F76               .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          1F77   741 LP11:   $DS_LP11
                          1F77               .SAVE   LOCAL_BLOCK
                          1F77               .PSECT  $ABS$,ABS
                00000032  0050               .=HP$A_DEPENDENT
                          0032               .IIF    NB,LP11$L_CSR,  LP11$L_CSR:
                00000036  0032               .IIF    NB,.BLKL,       .BLKL   1
                          0036               .IIF    NB,LP11$B_BR,   LP11$B_BR:
                00000037  0036               .IIF    NB,.BLKB,       .BLKB   1
                          0037               .IIF    NB,LP11$K_LEN,  LP11$K_LEN:
                      00001F77               .RESTORE
          31 31 50 4C 00' 1F77               .ASCIC  "LP11"  ; LP11 name
                      04  1F77
                      37  1F7C               .BYTE   LP11$K_LEN              ; LP11$K_LEN of resultant P-table
                      00  1F7D               .BYTE   0               ; Maximum unit number                    ;G:65
                    504C  1F7E               .WORD   ^A"LP"  ; Two character mnemonic for QIO LP11 LP
                      80  1F80               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  1F81               .Byte   Pd$_Name        ; Directive op-code
                      02  1F82               .Byte   Ptd$M_Controller                ; Enforced LP codes
                50 4C 00' 1F83               .Ascic  "LP"    ; Enforced device name prefix
                      02  1F83
                      83  1F86               .BYTE   Pd$_Octal       ; Indicate scan octal
             52 53 43 00' 1F87               .ASCIC  "CSR"   ; CSR string
                      03  1F87
        0003FFFE 0003E000  1F8B               .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  1F93               .BYTE   Pd$_Store       ; Indicate store operation
                    0032  1F94               .WORD   LP11$L_CSR              ; Byte LP11$L_CSR to data
                      00  1F96               .BYTE   0               ; 0 LP11$L_CSR to data
                      20  1F97               .BYTE   32              ; 32 of data field
                      88  1F98               .BYTE   Pd$_Store       ; Indicate store operation
                    0018  1F99               .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  1F9B               .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  1F9C               .BYTE   18              ; 18 of data field
                      83  1F9D               .BYTE   Pd$_Octal       ; Indicate scan octal
      52 4F 54 43 45 56 00' 1F9E               .ASCIC  "VECTOR"        ; VECTOR string
```

```
                    06  1F9E
000001FE 00000002  1FA5                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                    88  1FAD                .BYTE   Pd$_Store       ; Indicate store operation
                  0024  1FAE                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  1FB0                .BYTE   0               ; 0 HP$W_VECTOR to data
                    09  1FB1                .BYTE   9               ; 9 of data field
                    82  1FB2                .BYTE   Pd$_Decimal     ; Indicate scan decimal
             52 42 00'  1FB3                .ASCIC  "BR"    ; BR string
                    02  1FB3
00000007 00000004  1FB6                .LONG   4,7     ; 4 , 7 limits on input
                    88  1FBE                .BYTE   Pd$_Store       ; Indicate store operation
                  0036  1FBF                .WORD   LP11$B_BR               ; Byte LP11$B_BR to data
                    00  1FC1                .BYTE   0               ; 0 LP11$B_BR to data
                    08  1FC2                .BYTE   8               ; 8 of data field
                    81  1FC3                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        1FC4   742 LP14:    $DS_LP14
                        1FC4                .SAVE   LOCAL_BLOCK
                        1FC4                .PSECT  $ABS$,ABS
             00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,LP14$K_LEN,  LP14$K_LEN:
                  00001FC4                .RESTORE
       34 31 50 4C 00'  1FC4                .ASCIC  "LP14"  ; LP14 name
                    04  1FC4
                    32  1FC9                .BYTE   LP14$K_LEN              ; LP14$K_LEN of resultant P-table
                    01  1FCA                .BYTE   1               ; Maximum unit number                   ;G:65
                  0000  1FCB                .WORD   ^A""    ; Two character mnemonic for QIO LP14
                    80  1FCD                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  1FCE                .Byte   Pd$_Name        ; Directive op-code
                    1B  1FCF                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
             50 4C 00'  1FD0                .Ascic  "LP"    ; Enforced device name prefix
                    02  1FD0
                    86  1FD3                .BYTE   Pd$_Literal     ; Indicate literal
             00000001  1FD4                .LONG   HP$M_ALLOC              ; Constant
                    88  1FD8                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  1FD9                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  1FDB                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  1FDC                .BYTE   8               ; 8 of data field
                    81  1FDD                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        1FDE   743 LP25:    $DS_LP25
                        1FDE                .SAVE   LOCAL_BLOCK
                        1FDE                .PSECT  $ABS$,ABS
             00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,LP25$K_LEN,  LP25$K_LEN:
                  00001FDE                .RESTORE
       35 32 50 4C 00'  1FDE                .ASCIC  "LP25"  ; LP25 name
                    04  1FDE
                    32  1FE3                .BYTE   LP25$K_LEN              ; LP25$K_LEN of resultant P-table
                    01  1FE4                .BYTE   1               ; Maximum unit number                   ;G:65
                  0000  1FE5                .WORD   ^A""    ; Two character mnemonic for QIO LP25
                    80  1FE7                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  1FE8                .Byte   Pd$_Name        ; Directive op-code
                    1B  1FE9                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
             50 4C 00'  1FEA                .Ascic  "LP"    ; Enforced device name prefix
                    02  1FEA
                    86  1FED                .BYTE   Pd$_Literal     ; Indicate literal
             00000001  1FEE                .LONG   HP$M_ALLOC              ; Constant
                    88  1FF2                .BYTE   Pd$_Store       ; Indicate store operation
```

```
                    000A  1FF3                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  1FF5                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  1FF6                .BYTE   8                   ; 8 of data field
                      81  1FF7                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          1FF8    744 LP26:    $DS_LP26
                          1FF8                .SAVE   LOCAL_BLOCK
                          1FF8                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LP26$K_LEN,  LP26$K_LEN:
                      00001FF8                .RESTORE
               36 32 50 4C 00' 1FF8                .ASCIC  "LP26"  ; LP26 name
                      04  1FF8
                      32  1FFD                .BYTE   LP26$K_LEN              ; LP26$K_LEN of resultant P-table
                      01  1FFE                .BYTE   1                   ; Maximum unit number                   ;G:65
                    0000  1FFF                .WORD   ^A""    ; Two character mnemonic for QIO LP26
                      80  2001                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  2002                .Byte   Pd$_Name            ; Directive op-code
                      1B  2003                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                  50 4C 00' 2004                .Ascic  "LP"    ; Enforced device name prefix
                      02  2004
                      86  2007                .BYTE   Pd$_Literal     ; Indicate literal
                  00000001  2008                .LONG   HP$M_ALLOC              ; Constant
                      88  200C                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  200D                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  200F                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  2010                .BYTE   8                   ; 8 of data field
                      81  2011                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          2012    745 LP27:    $DS_LP27
                          2012                .SAVE   LOCAL_BLOCK
                          2012                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LP27$K_LEN,  LP27$K_LEN:
                      00002012                .RESTORE
               37 32 50 4C 00' 2012                .ASCIC  "LP27"  ; LP27 name
                      04  2012
                      32  2017                .BYTE   LP27$K_LEN              ; LP27$K_LEN of resultant P-table
                      01  2018                .BYTE   1                   ; Maximum unit number                   ;G:65
                    0000  2019                .WORD   ^A""    ; Two character mnemonic for QIO LP27
                      80  201B                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  201C                .Byte   Pd$_Name            ; Directive op-code
                      1B  201D                .Byte   Ptd$M_EndDevice         ; Enforced LP codes
                  50 4C 00' 201E                .Ascic  "LP"    ; Enforced device name prefix
                      02  201E
                      86  2021                .BYTE   Pd$_Literal     ; Indicate literal
                  00000001  2022                .LONG   HP$M_ALLOC              ; Constant
                      88  2026                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  2027                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  2029                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  202A                .BYTE   8                   ; 8 of data field
                      81  202B                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          202C    746 LPA11K:  $DS_LPA11K
                          202C                .SAVE   LOCAL_BLOCK
                          202C                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LPA11K$L_CSR,        LPA11K$L_CSR:
                  00000036  0032                .IIF    NB,,BLKL,       .BLKL   1
                          0036                .IIF    NB,LPA11K$B_BR, LPA11K$B_BR:
```

```
                00000037  0036                .IIF    NB,..BLKB,          .BLKB   1
                          0037                .IIF    NB,LPA11K$K_LEN,    LPA11K$K_LEN:
                 0000202C                     .RESTORE
4B 31 31 41 50 4C 00'  202C                  .ASCIC  "LPA11K"            ; LPA11K name
                      06  202C
                      37  2033                .BYTE   LPA11K$K_LEN                ; LPA11K$K_LEN of resultant P-table
                      01  2034                .BYTE   1                   ; Maximum unit number                       ;G:65
                    414C  2035                .WORD   ^A"LA"  ; Two character mnemonic for QIO LPA11K LA
                      80  2037                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  2038                .Byte   Pd$_Name            ; Directive op-code
                      03  2039                .Byte   Ptd$M_Device                ; Enforced LA codes
                 41 4C 00'  203A              .Ascic  "LA"    ; Enforced device name prefix
                      02  203A
                      86  203D                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  203E                .LONG   HP$M_ALLOC                  ; Constant
                      88  2042                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  2043                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  2045                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  2046                .BYTE   8                   ; 8 of data field
                      83  2047                .BYTE   Pd$_Octal           ; Indicate scan octal
              52 53 43 00'  2048              .ASCIC  "CSR"   ; CSR string
                      03  2048
          0003FFFE 0003E000  204C             .LONG   ^O760000,^O777776           ; 760000 , 777776 limits on input
                      88  2054                .BYTE   Pd$_Store           ; Indicate store operation
                    0018  2055                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      00  2057                .BYTE   0                   ; 0 HP$A_DEVICE to data
                      12  2058                .BYTE   18                  ; 18 of data field
                      88  2059                .BYTE   Pd$_Store           ; Indicate store operation
                    0032  205A                .WORD   LPA11K$L_CSR                ; Byte LPA11K$L_CSR to data
                      00  205C                .BYTE   0                   ; 0 LPA11K$L_CSR to data
                      20  205D                .BYTE   32                  ; 32 of data field
                      83  205E                .BYTE   Pd$_Octal           ; Indicate scan octal
52 4F 54 43 45 56 00'  205F                  .ASCIC  "VECTOR"            ; VECTOR string
                      06  205F
    000001FE 00000002  2066                 .LONG   ^O2,^O776           ; 2 , 776 limits on input
                      88  206E                .BYTE   Pd$_Store           ; Indicate store operation
                    0024  206F                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      00  2071                .BYTE   0                   ; 0 HP$W_VECTOR to data
                      09  2072                .BYTE   9                   ; 9 of data field
                      82  2073                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                 52 42 00'  2074              .ASCIC  "BR"    ; BR string
                      02  2074
    00000007 00000004  2077                 .LONG   4,7     ; 4 , 7 limits on input
                      88  207F                .BYTE   Pd$_Store           ; Indicate store operation
                    0036  2080                .WORD   LPA11K$B_BR                 ; Byte LPA11K$B_BR to data
                      00  2082                .BYTE   0                   ; 0 LPA11K$B_BR to data
                      08  2083                .BYTE   8                   ; 8 of data field
                      81  2084                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          2085    747 LUA11:  $DS_LUA11                                   ;       [71]
                          2085                .SAVE   LOCAL_BLOCK
                          2085                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,LUA11$B_BR,  LUA11$B_BR:
                00000033  0032                .IIF    NB,..BLKB,          .BLKB   1
                          0033                .IIF    NB,LUA11$L_CSR, LUA11$L_CSR:
                00000037  0033                .IIF    NB,..BLKL,          .BLKL   1
                          0037                .IIF    NB,LUA11$K_LEN, LUA11$K_LEN:
```

```
                    00002085                .RESTORE
          31 31 41 55 4C 00' 2085           .ASCIC  "LUA11" ; LUA11 name
                       05  2085
                       37  208B            .BYTE   LUA11$K_LEN             ; LUA11$K_LEN of resultant P-table
                       08  208C            .BYTE   8                       ; Maximum unit number                   ;G:65
                     0000  208D            .WORD   ^A""    ; Two character mnemonic for QIO LUA11
                       80  208F            .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                       8D  2090            .Byte   Pd$_Name        ; Directive op-code
                       03  2091            .Byte   PTD$M_DEVICE            ; Enforced XE codes
                    45 58 00' 2092         .Ascic  "XE"    ; Enforced device name prefix
                       02  2092
                       83  2095            .BYTE   Pd$_Octal       ; Indicate scan octal
                 52 53 43 00' 2096         .ASCIC  "CSR"   ; CSR string
                       03  2096
          0003FFFE 0003E000  209A          .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                       88  20A2            .BYTE   Pd$_Store       ; Indicate store operation
                     0033  20A3            .WORD   LUA11$L_CSR             ; Byte LUA11$L_CSR to data
                       00  20A5            .BYTE   0                       ; 0 LUA11$L_CSR to data
                       20  20A6            .BYTE   32                      ; 32 of data field
                       88  20A7            .BYTE   Pd$_Store       ; Indicate store operation
                     0018  20A8            .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  20AA            .BYTE   0                       ; 0 HP$A_DEVICE to data
                       12  20AB            .BYTE   18                      ; 18 of data field
                       83  20AC            .BYTE   Pd$_Octal       ; Indicate scan octal
       52 4F 54 43 45 56 00' 20AD          .ASCIC  "VECTOR"        ; VECTOR string
                       06  20AD
          000001FE 00000002  20B4          .LONG   ^O2,^O776       ; 2 , 776 limits on input
                       88  20BC            .BYTE   Pd$_Store       ; Indicate store operation
                     0024  20BD            .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                       00  20BF            .BYTE   0                       ; 0 HP$W_VECTOR to data
                       09  20C0            .BYTE   9                       ; 9 of data field
                       82  20C1            .BYTE   Pd$_Decimal     ; Indicate scan decimal
                    52 42 00' 20C2         .ASCIC  "BR"    ; BR string
                       02  20C2
          00000007 00000004  20C5          .LONG   4,7     ; 4 , 7 limits on input
                       88  20CD            .BYTE   Pd$_Store       ; Indicate store operation
                     0032  20CE            .WORD   LUA11$B_BR              ; Byte LUA11$B_BR to data
                       00  20D0            .BYTE   0                       ; 0 LUA11$B_BR to data
                       08  20D1            .BYTE   8                       ; 8 of data field
                       81  20D2            .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                           20D3    748 MA780:  $DS_MA780
                           20D3            .SAVE   LOCAL_BLOCK
                           20D3            .PSECT  $ABS$,ABS
                    00000032   0050        .=HP$A_DEPENDENT
                           0032            .IIF    NB,MA780$B_TR,  MA780$B_TR:
                    00000033   0032        .IIF    NB,..BLKB,      .BLKB   1
                           0033            .IIF    NB,MA780$B_BR,  MA780$B_BR:
                    00000034   0033        .IIF    NB,..BLKB,      .BLKB   1
                           0034            .IIF    NB,MA780$B_MPM, MA780$B_MPM:
                    00000035   0034        .IIF    NB,..BLKB,      .BLKB   1
                           0035            .IIF    NB,MA780$B_PORT,        MA780$B_PORT:
                    00000036   0035        .IIF    NB,..BLKB,      .BLKB   1
                           0036            .IIF    NB,MA780$K_LEN, MA780$K_LEN:
                      000020D3             .RESTORE
          30 38 37 41 4D 00' 20D3          .ASCIC  "MA780" ; MA780 name
                       05  20D3
                       36  20D9            .BYTE   MA780$K_LEN             ; MA780$K_LEN of resultant P-table
```

```
                  08  20DA          .BYTE   8                   ; Maximum unit number                    ;G:65
                0000  20DB          .WORD   ^A""      ; Two character mnemonic for QIO MA780
                  80  20DD          .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                  8D  20DE          .Byte   Pd$_Name            ; Directive op-code
                  01  20DF          .Byte   Ptd$M_Unit                  ; Enforced MA codes
           41 4D 00'  20E0          .Ascic  "MA"      ; Enforced device name prefix
                  02  20E0
                  82  20E3          .BYTE   Pd$_Decimal         ; Indicate scan decimal
           52 54 00'  20E4          .ASCIC  "TR"      ; TR string
                  02  20E4
    0000000F 00000001  20E7          .LONG   1,15      ; 1 , 15 limits on input
                  88  20EF          .BYTE   Pd$_Store           ; Indicate store operation
                0032  20F0          .WORD   MA780$B_TR                  ; Byte MA780$B_TR to data
                  00  20F2          .BYTE   0                   ; 0 MA780$B_TR to data
                  08  20F3          .BYTE   8                   ; 8 of data field
                  88  20F4          .BYTE   Pd$_Store           ; Indicate store operation
                0018  20F5          .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                  0D  20F7          .BYTE   13                  ; 13 HP$A_DEVICE to data
                  04  20F8          .BYTE   4                   ; 4 of data field
                  88  20F9          .BYTE   Pd$_Store           ; Indicate store operation
                0024  20FA          .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                  02  20FC          .BYTE   2                   ; 2 HP$W_VECTOR to data
                  04  20FD          .BYTE   4                   ; 4 of data field
                  82  20FE          .BYTE   Pd$_Decimal         ; Indicate scan decimal
           52 42 00'  20FF          .ASCIC  "BR"      ; BR string
                  02  20FF
    00000007 00000004  2102          .LONG   4,7       ; 4 , 7 limits on input
                  88  210A          .BYTE   Pd$_Store           ; Indicate store operation
                0033  210B          .WORD   MA780$B_BR                  ; Byte MA780$B_BR to data
                  00  210D          .BYTE   0                   ; 0 MA780$B_BR to data
                  08  210E          .BYTE   8                   ; 8 of data field
                  88  210F          .BYTE   Pd$_Store           ; Indicate store operation
                0024  2110          .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                  06  2112          .BYTE   6                   ; 6 HP$W_VECTOR to data
                  02  2113          .BYTE   2                   ; 2 of data field
                  86  2114          .BYTE   Pd$_Literal         ; Indicate literal
           00000001  2115          .LONG   1                   ; Constant
                  88  2119          .BYTE   Pd$_Store           ; Indicate store operation
                0024  211A          .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                  08  211C          .BYTE   8                   ; 8 HP$W_VECTOR to data
                  01  211D          .BYTE   1                   ; 1 of data field
                  86  211E          .BYTE   Pd$_Literal         ; Indicate literal
           00000006  211F          .LONG   6                   ; Constant
                  88  2123          .BYTE   Pd$_Store           ; Indicate store operation
                0018  2124          .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                  1C  2126          .BYTE   28                  ; 28 HP$A_DEVICE to data
                  04  2127          .BYTE   4                   ; 4 of data field
                  82  2128          .BYTE   Pd$_Decimal         ; Indicate scan decimal
        4D 50 4D 00'  2129          .ASCIC  "MPM"     ; MPM string
                  03  2129
    00000003 00000000  212D          .LONG   0,3       ; 0 , 3 limits on input
                  88  2135          .BYTE   Pd$_Store           ; Indicate store operation
                0034  2136          .WORD   MA780$B_MPM                 ; Byte MA780$B_MPM to data
                  00  2138          .BYTE   0                   ; 0 MA780$B_MPM to data
                  08  2139          .BYTE   8                   ; 8 of data field
                  82  213A          .BYTE   Pd$_Decimal         ; Indicate scan decimal
     54 52 4F 50 00'  213B          .ASCIC  "PORT"    ; PORT string
```

```
                    04  213B
          00000003 00000000  2140                .LONG   0,3     ; 0 , 3 limits on input
                    88  2148                .BYTE   Pd$_Store       ; Indicate store operation
                  0035  2149                .WORD   MA780$B_PORT             ; Byte MA780$B_PORT to data
                    00  214B                .BYTE   0               ; 0 MA780$B_PORT to data
                    08  214C                .BYTE   8               ; 8 of data field
                    81  214D                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        214E    749 MBE:    $DS_MBE
                        214E                .SAVE   LOCAL_BLOCK
                        214E                .PSECT  $ABS$,ABS
                00000032  0050              .=HP$A_DEPENDENT
                        0032                .IIF    NB,MBE$K_LEN,   MBE$K_LEN:
                    0000214E                .RESTORE
                45 42 4D 00' 214E           .ASCIC  "MBE"   ; MBE name
                    03  214E
                    32  2152                .BYTE   MBE$K_LEN               ; MBE$K_LEN of resultant P-table
                    08  2153                .BYTE   8               ; Maximum unit number                   ;G:65
                  0000  2154                .WORD   ^A""    ; Two character mnemonic for QIO MBE
                    80  2156                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    87  2157                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                  000B  2158                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                    00  215A                .BYTE   0               ; 0 HP$B_DRIVE to data
                    08  215B                .BYTE   8               ; 8 of data field
                    88  215C                .BYTE   Pd$_Store       ; Indicate store operation
                  0018  215D                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    07  215F                .BYTE   7               ; 7 HP$A_DEVICE to data
                    03  2160                .BYTE   3               ; 3 of data field
                    81  2161                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        2162    750 ML11:   $DS_ML11
                        2162                .SAVE   LOCAL_BLOCK
                        2162                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,ML11$B_ARRAY,        ML11$B_ARRAY:
              00000033  0032                .IIF    NB,,BLKB,       .BLKB   1
                        0033                .IIF    NB,ML11$B_CHIP, ML11$B_CHIP:
              00000034  0033                .IIF    NB,,BLKB,       .BLKB   1
                        0034                .IIF    NB,ML11$K_LEN,  ML11$K_LEN:
                    00002162                .RESTORE
           31 31 4C 4D 00' 2162             .ASCIC  "ML11"  ; ML11 name
                    04  2162
                    34  2167                .BYTE   ML11$K_LEN              ; ML11$K_LEN of resultant P-table
                    08  2168                .BYTE   8               ; Maximum unit number                   ;G:65
                  4D45  2169                .WORD   ^A"EM"  ; Two character mnemonic for QIO ML11 EM
                    80  216B                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  216C                .Byte   Pd$_Name        ; Directive op-code
                    03  216D                .Byte   Ptd$M_Device            ; Enforced EM codes
              4D 45 00' 216E                .Ascic  "EM"    ; Enforced device name prefix
                    02  216E
                    86  2171                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  2172                .LONG   HP$M_ALLOC              ; Constant
                    88  2176                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  2177                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  2179                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  217A                .BYTE   8               ; 8 of data field
                    87  217B                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                  000B  217C                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                    00  217E                .BYTE   0               ; 0 HP$B_DRIVE to data
```

```
                                08  217F                .BYTE   8                       ; 8 of data field
                                88  2180                .BYTE   Pd$_Store            ; Indicate store operation
                              0018  2181                .WORD   HP$A_DEVICE                  ; Byte HP$A_DEVICE to data
                                07  2183                .BYTE   7                       ; 7 HP$A_DEVICE to data
                                03  2184                .BYTE   3                       ; 3 of data field
                                82  2185                .BYTE   Pd$_Decimal          ; Indicate scan decimal
61 20 66 6F 20 72 65 62 6D 75 4E 00' 2186                .ASCIC  "Number of array boards"     ; Number of array boards string
   73 64 72 61 6F 62 20 79 61 72 72  2192
                                16  2186
                  00000010 00000001  219D                .LONG   1,16    ; 1 , 16 limits on input
                                88  21A5                .BYTE   Pd$_Store            ; Indicate store operation
                              0032  21A6                .WORD   ML11$B_ARRAY                 ; Byte ML11$B_ARRAY to data
                                00  21A8                .BYTE   0                       ; 0 ML11$B_ARRAY to data
                                08  21A9                .BYTE   8                       ; 8 of data field
                                85  21AA                .BYTE   Pd$_String           ; Indicate scan and verify string
      65 7A 69 73 20 70 69 68 43 00' 21AB                .ASCIC  "Chip size"          ; Chip size string
                                09  21AB
                        4B 36 31 00' 21B5                .ASCIC  "16K"
                                03  21B5
                        4B 34 36 00' 21B9                .ASCIC  "64K"
                                03  21B9
                                00  21BD                .BYTE   0                       ; Null length is end of valid 16K,64K
                                88  21BE                .BYTE   Pd$_Store            ; Indicate store operation
                              0033  21BF                .WORD   ML11$B_CHIP                  ; Byte ML11$B_CHIP to data
                                00  21C1                .BYTE   0                       ; 0 ML11$B_CHIP to data
                                08  21C2                .BYTE   8                       ; 8 of data field
                                81  21C3                .BYTE   Pd$_End              ; Indicate end of PT-descriptor
                                    21C4    751 MS750:  $DS_MS750
                                    21C4                .SAVE   LOCAL_BLOCK
                                    21C4                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,MS750$K_LEN, MS750$K_LEN:
                                000021C4                .RESTORE
30 35 37 53 4D 00' 21C4                .ASCIC  "MS750" ; MS750 name
                                05  21C4
                                32  21CA                .BYTE   MS750$K_LEN                  ; MS750$K_LEN of resultant P-table
                                08  21CB                .BYTE   8                       ; Maximum unit number                                ;G:65
                              0000  21CC                .WORD   ^A""    ; Two character mnemonic for QIO MS750
                                80  21CE                .BYTE   Pd$_Start            ; Constant to identify start of descriptor
                                8D  21CF                .Byte   Pd$_Name             ; Directive op-code
                                01  21D0                .Byte   Ptd$M_Unit                   ; Enforced MS codes
                          53 4D 00' 21D1                .Ascic  "MS"    ; Enforced device name prefix
                                02  21D1
                                81  21D4                .BYTE   Pd$_End              ; Indicate end of PT-descriptor
                                    21D5    752 MS780:  $DS_MS780
                                    21D5                .SAVE   LOCAL_BLOCK
                                    21D5                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,MS780$B_TR,  MS780$B_TR:
                          00000033  0032                .IIF    NB,..BLKB,      .BLKB   1
                                    0033                .IIF    NB,ms780$b_arrays,      ms780$b_arrays:
                          00000034  0033                .IIF    NB,..blkb,      .blkb   1
                                    0034                .IIF    NB,MS780$K_LEN, MS780$K_LEN:
                                000021D5                .RESTORE
30 38 37 53 4D 00' 21D5                .ASCIC  "MS780" ; MS780 name
                                05  21D5
                                34  21DB                .BYTE   MS780$K_LEN                  ; MS780$K_LEN of resultant P-table
```

```
                          08  21DC                .BYTE   8                   ; Maximum unit number                          ;G:65
                        0000  21DD                .WORD   ^A""     ; Two character mnemonic for QIO MS780
                          80  21DF                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                          8D  21E0                .Byte   Pd$_Name            ; Directive op-code
                          01  21E1                .Byte   Ptd$M_Unit                  ; Enforced MS codes
                    53 4D 00' 21E2                .Ascic  "MS"     ; Enforced device name prefix
                          02  21E2
                          82  21E5                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                    52 54 00' 21E6                .ASCIC  "TR"     ; TR string
                          02  21E6
              0000000F 00000001  21E9                .LONG   1,15     ; 1 , 15 limits on input
                          88  21F1                .BYTE   Pd$_Store           ; Indicate store operation
                        0032  21F2                .WORD   MS780$B_TR                  ; Byte MS780$B_TR to data
                          00  21F4                .BYTE   0                   ; 0 MS780$B_TR to data
                          08  21F5                .BYTE   8                   ; 8 of data field
                          88  21F6                .BYTE   Pd$_Store           ; Indicate store operation
                        0018  21F7                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                          0D  21F9                .BYTE   13                  ; 13 HP$A_DEVICE to data
                          04  21FA                .BYTE   4                   ; 4 of data field
                          86  21FB                .BYTE   Pd$_Literal         ; Indicate literal
                    00000006  21FC                .LONG   6                   ; Constant
                          88  2200                .BYTE   Pd$_Store           ; Indicate store operation
                        0018  2201                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                          1C  2203                .BYTE   28                  ; 28 HP$A_DEVICE to data
                          04  2204                .BYTE   4                   ; 4 of data field
                          82  2205                .BYTE   Pd$_Decimal         ; Indicate scan decimal
61 20 66 6F 20 72 65 62 6D 75 4E 00' 2206                .ASCIC  "Number of arrays"          ; Number of arrays string
                 73 79 61 72 72  2212
                          10  2206
              00000010 00000000  2217                .LONG   0,16     ; 0 , 16 limits on input
                          88  221F                .BYTE   Pd$_Store           ; Indicate store operation
                        0033  2220                .WORD   ms780$b_arrays              ; Byte ms780$b_arrays to data
                          00  2222                .BYTE   0                   ; 0 ms780$b_arrays to data
                          08  2223                .BYTE   8                   ; 8 of data field
                          81  2224                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                              2225   753 MSAAA:   $DS_MSAAA
                              2225                .SAVE   LOCAL_BLOCK
                              2225                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$K_MSA_LEN,        HP$K_MSA_LEN:
                        00002225                .RESTORE
              41 41 41 53 4D 00' 2225                .ASCIC  "MSAAA" ; MSAAA name
                          05  2225
                          32  222B                .BYTE   HP$K_MSA_LEN                ; HP$K_MSA_LEN of resultant P-table
                          02  222C                .BYTE   2                   ; Maximum unit number                          ;G:65
                        0000  222D                .WORD   ^A""     ; Two character mnemonic for QIO MSAAA
                          80  222F                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                          8D  2230                .Byte   Pd$_Name            ; Directive op-code
                          01  2231                .Byte   PTD$M_UNIT                  ; Enforced MS codes
                    53 4D 00' 2232                .Ascic  "MS"     ; Enforced device name prefix
                          02  2232
                          81  2235                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                              2236   754 NBIA:    $DS_NBIA
                              2236                .SAVE   LOCAL_BLOCK
                              2236                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$B_NBIA_COMMON,    HP$B_NBIA_COMMON:
```

```
                    00000033  0032                .IIF    NB,..BLKB,        .BLKB   1
                              0033                .IIF    NB,HP$L_NBIA_CSR,         HP$L_NBIA_CSR:
                    00000037  0033                .IIF    NB,..BLKL,        .BLKL   1
                              0037                .IIF    NB,HP$B_BR,     HP$B_BR:
                    00000038  0037                .IIF    NB,..BLKB,        .BLKB   1
                              0038                .IIF    NB,HP$K_NBIA_LEN,         HP$K_NBIA_LEN:
                        00002236                  .RESTORE
                 41 49 42 4E 00'  2236            .ASCIC  "NBIA"  ; NBIA name
                           04  2236
                           38  223B                .BYTE   HP$K_NBIA_LEN           ; HP$K_NBIA_LEN of resultant P-table
                           02  223C                .BYTE   2                       ; Maximum unit number                   ;G:65
                         0000  223D                .WORD   ^A""    ; Two character mnemonic for QIO NBIA
                           80  223F                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                           8D  2240                .Byte   Pd$_Name        ; Directive op-code
                           01  2241                .Byte   PTD$M_UNIT              ; Enforced NBIA codes
                 41 49 42 4E 00'  2242            .Ascic  "NBIA"  ; Enforced device name prefix
                           04  2242
                           86  2247                .BYTE   Pd$_Literal     ; Indicate literal
                     60080000  2248                .LONG   ^X60080000              ; Constant
                           88  224C                .BYTE   Pd$_Store       ; Indicate store operation
                         0018  224D                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                           00  224F                .BYTE   0               ; 0 HP$A_DEVICE to data
                           20  2250                .BYTE   32              ; 32 of data field
                           82  2251                .BYTE   Pd$_Decimal     ; Indicate scan decimal
41 44 41 20 4C 41 47 49 47 4F 4C 00'  2252        .ASCIC  "LOGIGAL ADAPTER # "    ; LOGIGAL ADAPTER #  string
                 20 23 20 52 45 54 50  225E
                           12  2252
                 00000001 00000000  2265                .LONG   0,1     ; 0 , 1 limits on input
                           88  226D                .BYTE   Pd$_Store       ; Indicate store operation
                         0018  226E                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                           1A  2270                .BYTE   26              ; 26 HP$A_DEVICE to data
                           01  2271                .BYTE   1               ; 1 of data field
                           8C  2272                .BYTE   Pd$_Case        ; Indicate case operation
                           02  2273                .BYTE   $$_$            ; Count of pairs
                 00000120 00000000  2274           .LONG   0,^X120         ; Store the pair
                 00000130 00000001  227C           .LONG   1,^X130         ; Store the pair
                           88  2284                .BYTE   Pd$_Store       ; Indicate store operation
                         0024  2285                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                           00  2287                .BYTE   0               ; 0 HP$W_VECTOR to data
                           10  2288                .BYTE   16              ; 16 of data field
                           87  2289                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                         0018  228A                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                           1A  228C                .BYTE   26              ; 26 HP$A_DEVICE to data
                           01  228D                .BYTE   1               ; 1 of data field
                           8C  228E                .BYTE   Pd$_Case        ; Indicate case operation
                           02  228F                .BYTE   $$_$            ; Count of pairs
                 60000000 00000000  2290           .LONG   0,^X60000000            ; Store the pair
                 64000000 00000001  2298           .LONG   1,^X64000000            ; Store the pair
                           88  22A0                .BYTE   Pd$_Store       ; Indicate store operation
                         001C  22A1                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                           00  22A3                .BYTE   0               ; 0 HP$A_DVA to data
                           20  22A4                .BYTE   32              ; 32 of data field
                           81  22A5                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                               22A6  755 NBIB:      $DS_NBIB
                               22A6                .SAVE   LOCAL_BLOCK
                               22A6                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
```

```
                        0032                .IIF    NB,HP$B_NBIB_COMMON,    HP$B_NBIB_COMMON:
                00000033  0032                .IIF    NB,..BLKB,        .BLKB   1
                          0033                .IIF    NB,HP$B_BI_NODE,        HP$B_BI_NODE:
                00000034  0033                .IIF    NB,..BLKB,        .BLKB   1
                          0034                .IIF    NB,HP$L_NBIB_CSR,       HP$L_NBIB_CSR:
                00000038  0034                .IIF    NB,..BLKL,        .BLKL   1
                          0038                .IIF    NB,HP$B_BI_NUM, HP$B_BI_NUM:
                00000039  0038                .IIF    NB,..BLKB,        .BLKB   1
                          0039                .IIF    NB,HP$K_NBIB_LEN,       HP$K_NBIB_LEN:
                   000022A6                 .RESTORE
          42 49 42 4E 00'  22A6                .ASCIC  "NBIB"  ; NBIB name
                     04  22A6
                     39  22AB                .BYTE   HP$K_NBIB_LEN           ; HP$K_NBIB_LEN of resultant P-table
                     06  22AC                .BYTE   6               ; Maximum unit number                           ;G:65
                   0000  22AD                .WORD   ^A""    ; Two character mnemonic for QIO NBIB
                     80  22AF                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                     8D  22B0                .Byte   Pd$_Name        ; Directive op-code
                     01  22B1                .Byte   PTD$M_UNIT              ; Enforced NBIB codes
          42 49 42 4E 00'  22B2                .Ascic  "NBIB"  ; Enforced device name prefix
                     04  22B2
                     82  22B7                .BYTE   Pd$_Decimal     ; Indicate scan decimal
       20 23 20 49 42 00'  22B8                .ASCIC  "BI # " ; BI #  string
                     05  22B8
          00000001 00000000  22BE                .LONG   0,1     ; 0 , 1 limits on input
                     88  22C6                .BYTE   Pd$_Store       ; Indicate store operation
                   0038  22C7                .WORD   HP$B_BI_NUM             ; Byte HP$B_BI_NUM to data
                     00  22C9                .BYTE   0               ; 0 HP$B_BI_NUM to data
                     02  22CA                .BYTE   2               ; 2 of data field
                     88  22CB                .BYTE   Pd$_Store       ; Indicate store operation
                   0018  22CC                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                     19  22CE                .BYTE   25              ; 25 HP$A_DEVICE to data
                     01  22CF                .BYTE   1               ; 1 of data field
                     87  22D0                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                   0018  22D1                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                     00  22D3                .BYTE   0               ; 0 HP$A_DEVICE to data
                     20  22D4                .BYTE   32              ; 32 of data field
                     88  22D5                .BYTE   Pd$_Store       ; Indicate store operation
                   001C  22D6                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                     00  22D8                .BYTE   0               ; 0 HP$A_DVA to data
                     20  22D9                .BYTE   32              ; 32 of data field
                     84  22DA                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00'  22DB                .ASCIC  "BI Node Number (HEX) " ; BI Node Number (HEX)  string
      20 29 58 45 48 28 20 72 65 62   22E7
                     15  22DB
                0000000F 00000000  22F1                .LONG   ^X0,^XF ; 0 , F limits on input
                     88  22F9                .BYTE   Pd$_Store       ; Indicate store operation
                   0033  22FA                .WORD   HP$B_BI_NODE             ; Byte HP$B_BI_NODE to data
                     00  22FC                .BYTE   0               ; 0 HP$B_BI_NODE to data
                     08  22FD                .BYTE   8               ; 8 of data field
                     87  22FE                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                   0033  22FF                .WORD   HP$B_BI_NODE             ; Byte HP$B_BI_NODE to data
                     00  2301                .BYTE   0               ; 0 HP$B_BI_NODE to data
                     08  2302                .BYTE   8               ; 8 of data field
                     88  2303                .BYTE   Pd$_Store       ; Indicate store operation
                   0018  2304                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                     0D  2306                .BYTE   13              ; 13 HP$A_DEVICE to data
                     04  2307                .BYTE   4               ; 4 of data field
```

```
                                81  2308                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                                    2309   756 NI:      $DS_NI                                  ;       [62]
                                    2309                .SAVE   LOCAL_BLOCK
                                    2309                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_Dependent
                                    0032                .IIF    NB,NI$B_Loopback,       NI$B_Loopback::
                          00000033  0032                .IIF    NB,.Blkb,       .Blkb   1
                                    0033                .IIF    NB,NI$K_Len,    NI$K_Len::
                              00002309                .RESTORE
                          49 4E 00' 2309                .ASCIC  "NI"    ; NI name
                                02  2309
                                33  230C                .BYTE   NI$K_Len                ; NI$K_Len of resultant P-table
                                00  230D                .BYTE   0               ; Maximum unit number                   ;G:65
                              0000  230E                .WORD   ^A""    ; Two character mnemonic for QIO NI
                                80  2310                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  2311                .Byte   Pd$_Name        ; Directive op-code
                                04  2312                .Byte   Ptd$M_Name              ; Enforced XEA0 codes
                    30 41 45 58 00' 2313                .Ascic  "XEA0"  ; Enforced device name prefix
                                04  2313
                                8B  2318                .BYTE   Pd$_Logical     ; Indicate logical value operation
6B 63 61 62 70 6F 6F 4C 20 49 4E 00' 2319               .ASCIC  \NI Loopback Connector Installed\       ; NI Loopback Connector Inst
49 20 72 6F 74 63 65 6E 6E 6F 43 20  2325
            64 65 6C 6C 61 74 73 6E  2331
                                1F  2319
                                88  2339                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  233A                .WORD   NI$B_Loopback           ; Byte NI$B_Loopback to data
                                00  233C                .BYTE   0               ; 0 NI$B_Loopback to data
                                01  233D                .BYTE   1               ; 1 of data field
                                81  233E                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    233F   757 PBIA:    $DS_PBIA                                ;                       ;G:48
                                    233F                .SAVE   LOCAL_BLOCK
                                    233F                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,HP$B_PBIA_COMMON,    HP$B_PBIA_COMMON:
                          00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                                    0033                .IIF    NB,HP$L_PBIA_CSR,       HP$L_PBIA_CSR:
                          00000037  0033                .IIF    NB,.BLKL,       .BLKL   1
                                    0037                .IIF    NB,HP$B_PBR,    HP$B_PBR:
                          00000038  0037                .IIF    NB,.BLKB,       .BLKB   1
                                    0038                .IIF    NB,HP$K_PBIA_LEN,       HP$K_PBIA_LEN:
                              0000233F                .RESTORE
                    41 49 42 50 00' 233F                .ASCIC  "PBIA"  ; PBIA name
                                04  233F
                                38  2344                .BYTE   HP$K_PBIA_LEN           ; HP$K_PBIA_LEN of resultant P-table
                                03  2345                .BYTE   3               ; Maximum unit number                   ;G:65
                              0000  2346                .WORD   ^A""    ; Two character mnemonic for QIO PBIA
                                80  2348                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  2349                .Byte   Pd$_Name        ; Directive op-code
                                01  234A                .Byte   PTD$M_UNIT              ; Enforced PBIA codes
                    41 49 42 50 00' 234B                .Ascic  "PBIA"  ; Enforced device name prefix
                                04  234B
                                86  2350                .BYTE   Pd$_Literal     ; Indicate literal
                          60080000  2351                .LONG   ^X60080000              ; Constant
                                88  2355                .BYTE   Pd$_Store       ; Indicate store operation
                              0018  2356                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                00  2358                .BYTE   0               ; 0 HP$A_DEVICE to data
                                20  2359                .BYTE   32              ; 32 of data field
```

```
                                82  235A                .BYTE   Pd$_Decimal     ; Indicate scan decimal
41 44 41 20 4C 41 47 49 47 4F 4C 00' 235B                .ASCIC  "LOGIGAL ADAPTER # "     ; LOGIGAL ADAPTER #  string
                20 23 20 52 45 54 50  2367
                                12  235B
                    00000002 00000000  236E                .LONG   0,2     ; 0 , 2 limits on input
                                88  2376                .BYTE   Pd$_Store       ; Indicate store operation
                              0018  2377                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                1A  2379                .BYTE   26              ; 26 HP$A_DEVICE to data
                                02  237A                .BYTE   2               ; 2 of data field
                                8C  237B                .BYTE   Pd$_Case        ; Indicate case operation
                                03  237C                .BYTE   $$_$            ; Count of pairs
                    00000120 00000000  237D                .LONG   0,^X120         ; Store the pair
                    00000130 00000001  2385                .LONG   1,^X130         ; Store the pair
                    00000140 00000002  238D                .LONG   2,^X140         ; Store the pair
                                88  2395                .BYTE   Pd$_Store       ; Indicate store operation
                              0024  2396                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                                00  2398                .BYTE   0               ; 0 HP$W_VECTOR to data
                                10  2399                .BYTE   16              ; 16 of data field
                                87  239A                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                              0018  239B                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                                1A  239D                .BYTE   26              ; 26 HP$A_DEVICE to data
                                02  239E                .BYTE   2               ; 2 of data field
                                8C  239F                .BYTE   Pd$_Case        ; Indicate case operation
                                03  23A0                .BYTE   $$_$            ; Count of pairs
                    60000000 00000000  23A1                .LONG   0,^X60000000            ; Store the pair
                    64000000 00000001  23A9                .LONG   1,^X64000000            ; Store the pair
                    68000000 00000002  23B1                .LONG   2,^X68000000            ; Store the pair
                                88  23B9                .BYTE   Pd$_Store       ; Indicate store operation
                              001C  23BA                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                                00  23BC                .BYTE   0               ; 0 HP$A_DVA to data
                                20  23BD                .BYTE   32              ; 32 of data field
                                81  23BE                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    23BF    758 PCL11:  $DS_PCL11
                                    23BF                .SAVE   LOCAL_BLOCK
                                    23BF                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,PCL11$L_CSR, PCL11$L_CSR:
                          00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                                    0036                .IIF    NB,PCL11$B_BR,  PCL11$B_BR:
                          00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                                    0037                .IIF    NB,PCL11$K_LEN, PCL11$K_LEN:
                             000023BF                   .RESTORE
                  31 31 4C 43 50 00' 23BF               .ASCIC  "PCL11" ; PCL11 name
                                05  23BF
                                37  23C5                .BYTE   PCL11$K_LEN             ; PCL11$K_LEN of resultant P-table
                                20  23C6                .BYTE   32              ; Maximum unit number                           ;G:65
                              0000  23C7                .WORD   ^A""    ; Two character mnemonic for QIO PCL11
                                80  23C9                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                83  23CA                .BYTE   Pd$_Octal       ; Indicate scan octal
                        52 53 43 00' 23CB               .ASCIC  "CSR"   ; CSR string
                                03  23CB
                  0003FFFE 0003E000  23CF                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                                88  23D7                .BYTE   Pd$_Store       ; Indicate store operation
                              0032  23D8                .WORD   PCL11$L_CSR             ; Byte PCL11$L_CSR to data
                                00  23DA                .BYTE   0               ; 0 PCL11$L_CSR to data
                                20  23DB                .BYTE   32              ; 32 of data field
                                88  23DC                .BYTE   Pd$_Store       ; Indicate store operation
```

```
                     0018  23DD                .WORD   HP$A_DEVICE                ; Byte HP$A_DEVICE to data
                       00  23DF                .BYTE   0                   ; 0 HP$A_DEVICE to data
                       12  23E0                .BYTE   18                  ; 18 of data field
                       83  23E1                .BYTE   Pd$_Octal           ; Indicate scan octal
           52 4F 54 43 45 56 00' 23E2           .ASCIC  "VECTOR"            ; VECTOR string
                       06  23E2
              000001FE 00000002  23E9           .LONG   ^O2,^O776           ; 2 , 776 limits on input
                       88  23F1                .BYTE   Pd$_Store           ; Indicate store operation
                     0024  23F2                .WORD   HP$W_VECTOR                ; Byte HP$W_VECTOR to data
                       00  23F4                .BYTE   0                   ; 0 HP$W_VECTOR to data
                       09  23F5                .BYTE   9                   ; 9 of data field
                       82  23F6                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                  52 42 00' 23F7               .ASCIC  "BR"    ; BR string
                       02  23F7
              00000007 00000004  23FA           .LONG   4,7     ; 4 , 7 limits on input
                       88  2402                .BYTE   Pd$_Store           ; Indicate store operation
                     0036  2403                .WORD   PCL11$B_BR                 ; Byte PCL11$B_BR to data
                       00  2405                .BYTE   0                   ; 0 PCL11$B_BR to data
                       08  2406                .BYTE   8                   ; 8 of data field
                       81  2407                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           2408   759 PX780:    $DS_PX780                                   ;       [19]
                           2408                .SAVE   LOCAL_BLOCK
                           2408                .PSECT  $ABS$,ABS
                  00000032 0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,PX780$B_TR,  PX780$B_TR:
                  00000033 0032                .IIF    NB,,BLKB,       .BLKB   1
                           0033                .IIF    NB,PX780$B_BR,  PX780$B_BR:
                  00000034 0033                .IIF    NB,,BLKB,       .BLKB   1
                           0034                .IIF    NB,PX780$B_NODE,        PX780$B_NODE:
                  00000035 0034                .IIF    NB,,BLKB,       .BLKB   1
                           0035                .IIF    NB,PX780$K_LEN, PX780$K_LEN:
                       00002408                .RESTORE
           30 38 37 58 50 00' 2408             .ASCIC  "PX780" ; PX780 name
                       05  2408
                       35  240E                .BYTE   PX780$K_LEN                ; PX780$K_LEN of resultant P-table
                       01  240F                .BYTE   1                   ; Maximum unit number                       ;G:65
                     5850  2410                .WORD   ^A"PX"  ; Two character mnemonic for QIO PX780 PX
                       80  2412                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  2413                .Byte   Pd$_Name            ; Directive op-code
                       03  2414                .Byte   Ptd$M_Device                ; Enforced PX codes
                  58 50 00' 2415               .Ascic  "PX"    ; Enforced device name prefix
                       02  2415
                       82  2418                .BYTE   Pd$_Decimal         ; Indicate scan decimal
                  52 54 00' 2419               .ASCIC  "TR"    ; TR string
                       02  2419
              0000000F 00000001  241C           .LONG   1,15    ; 1 , 15 limits on input
                       88  2424                .BYTE   Pd$_Store           ; Indicate store operation
                     0032  2425                .WORD   PX780$B_TR                 ; Byte PX780$B_TR to data
                       00  2427                .BYTE   0                   ; 0 PX780$B_TR to data
                       08  2428                .BYTE   8                   ; 8 of data field
                       88  2429                .BYTE   Pd$_Store           ; Indicate store operation
                     0018  242A                .WORD   HP$A_DEVICE                ; Byte HP$A_DEVICE to data
                       0D  242C                .BYTE   13                  ; 13 HP$A_DEVICE to data
                       04  242D                .BYTE   4                   ; 4 of data field
                       88  242E                .BYTE   Pd$_Store           ; Indicate store operation
                     0024  242F                .WORD   HP$W_VECTOR                ; Byte HP$W_VECTOR to data
                       02  2431                .BYTE   2                   ; 2 HP$W_VECTOR to data
```

```
                      04  2432                .BYTE   4                       ; 4 of data field
                      82  2433                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                52 42 00' 2434                .ASCIC  "BR"     ; BR string
                      02  2434
          00000007 00000004  2437             .LONG   4,7      ; 4 . 7 limits on input
                      88  243F                .BYTE   Pd$_Store        ; Indicate store operation
                    0033  2440                .WORD   PX780$B_BR                  ; Byte PX780$B_BR to data
                      00  2442                .BYTE   0                ; 0 PX780$B_BR to data
                      08  2443                .BYTE   8                ; 8 of data field
                      88  2444                .BYTE   Pd$_Store        ; Indicate store operation
                    0024  2445                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      06  2447                .BYTE   6                ; 6 HP$W_VECTOR to data
                      02  2448                .BYTE   2                ; 2 of data field
                      82  2449                .BYTE   Pd$_Decimal      ; Indicate scan decimal
             65 64 6F 4E 00' 244A             .ASCIC  "Node"   ; Node string
                      04  244A
          000000FF 00000000  244F             .LONG   0,255    ; 0 . 255 limits on input
                      88  2457                .BYTE   Pd$_Store        ; Indicate store operation
                    0034  2458                .WORD   PX780$B_NODE                ; Byte PX780$B_NODE to data
                      00  245A                .BYTE   0                ; 0 PX780$B_NODE to data
                      08  245B                .BYTE   8                ; 8 of data field
                      86  245C                .BYTE   Pd$_Literal      ; Indicate literal
                00000006  245D                .LONG   6                ; Constant
                      88  2461                .BYTE   Pd$_Store        ; Indicate store operation
                    0018  2462                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                      1C  2464                .BYTE   28               ; 28 HP$A_DEVICE to data
                      04  2465                .BYTE   4                ; 4 of data field
                      86  2466                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  2467                .LONG   1                ; Constant
                      88  246B                .BYTE   Pd$_Store        ; Indicate store operation
                    0024  246C                .WORD   HP$W_VECTOR                 ; Byte HP$W_VECTOR to data
                      08  246E                .BYTE   8                ; 8 HP$W_VECTOR to data
                      01  246F                .BYTE   1                ; 1 of data field
                      81  2470                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2471    760 R80:    $DS_R80                                 ;       [16]
                          2471                .SAVE   LOCAL_BLOCK
                          2471                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,R80$K_LEN,   R80$K_LEN:
                    0000247 1                 .RESTORE
             30 38 52 00' 2471                .ASCIC  "R80"    ; R80 name
                      03  2471
                      32  2475                .BYTE   R80$K_LEN                   ; R80$K_LEN of resultant P-table
                      08  2476                .BYTE   8                ; Maximum unit number                          ;G:65
                    0000  2477                .WORD   ^A""     ; Two character mnemonic for QIO R80
                      80  2479                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  247A                .Byte   Pd$_Name         ; Directive op-code
                      1B  247B                .Byte   Ptd$M_EndDevice          ; Enforced DQ codes
                51 44 00' 247C                .Ascic  "DQ"     ; Enforced device name prefix
                      02  247C
                      86  247F                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  2480                .LONG   HP$M_ALLOC                  ; Constant
                      88  2484                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2485                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  2487                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  2488                .BYTE   8                ; 8 of data field
                      81  2489                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
```

```
                        248A   761 RA60:   $DS_RA60
                        248A               .SAVE   LOCAL_BLOCK
                        248A               .PSECT  $ABS$,ABS
                    00000032  0050         .=HP$A_DEPENDENT
                              0032         .IIF    NB,HP$K_RA60_LEN,       HP$K_RA60_LEN:
                              0032         .IIF    NB,RA60$K_LEN,  RA60$K_LEN:
                        0000248A           .RESTORE
                30 36 41 52 00' 248A       .ASCIC  "RA60"  ; RA60 name
                          04    248A
                          32    248F       .BYTE   RA60$K_LEN              ; RA60$K_LEN of resultant P-table
                          FF    2490       .BYTE   255             ; Maximum unit number                   ;G:65
                        0000    2491       .WORD   ^A""    ; Two character mnemonic for QIO RA60
                          80    2493       .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D    2494       .Byte   Pd$_Name        ; Directive op-code
                          13    2495       .Byte   Ptd$M_UNIT!PTD$M_CONTROLLER!PTD$M_INHERIT_CON           ; Enforced D
                    4A 44 00'   2496       .Asic   "DJ"    ; Enforced device name prefix
                          02    2496
                          86    2499       .BYTE   Pd$_Literal     ; Indicate literal
                    00000001    249A       .LONG   HP$M_ALLOC              ; Constant
                          88    249E       .BYTE   Pd$_Store       ; Indicate store operation
                        000A    249F       .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00    24A1       .BYTE   0               ; 0 HP$B_FLAGS to data
                          08    24A2       .BYTE   8               ; 8 of data field
                          81    24A3       .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                24A4   762 RA70:   $DS_RA70                        ;       [82]            ;G:33
                                24A4               .SAVE   LOCAL_BLOCK
                                24A4               .PSECT  $ABS$,ABS
                    00000032    0050               .=HP$A_DEPENDENT
                                0032               .IIF    NB,HP$K_RA70_LEN,       HP$K_RA70_LEN:
                                0032               .IIF    NB,RA70$K_LEN,  RA70$K_LEN:
                          000024A4                 .RESTORE
                30 37 41 52 00' 24A4               .ASCIC  "RA70"  ; RA70 name
                          04    24A4
                          32    24A9               .BYTE   RA70$K_LEN              ; RA70$K_LEN of resultant P-table
                          FF    24AA               .BYTE   255             ; Maximum unit number                   ;G:65
                        0000    24AB               .WORD   ^A""    ; Two character mnemonic for QIO RA70
                          80    24AD               .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D    24AE               .Byte   Pd$_Name        ; Directive op-code
                          1B    24AF               .Byte   PTD$M_ENDDEVICE         ; Enforced DU codes
                    55 44 00'   24B0               .Asic   "DU"    ; Enforced device name prefix
                          02    24B0
                          86    24B3               .BYTE   Pd$_Literal     ; Indicate literal
                    00000001    24B4               .LONG   HP$M_ALLOC              ; Constant
                          88    24B8               .BYTE   Pd$_Store       ; Indicate store operation
                        000A    24B9               .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00    24BB               .BYTE   0               ; 0 HP$B_FLAGS to data
                          08    24BC               .BYTE   8               ; 8 of data field
                          81    24BD               .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                24BE   763 RA80:   $DS_RA80                        ;       [17]
                                24BE               .SAVE   LOCAL_BLOCK
                                24BE               .PSECT  $ABS$,ABS
                    00000032    0050               .=HP$A_DEPENDENT
                                0032               .IIF    NB,HP$K_RA80_LEN,       HP$K_RA80_LEN:
                                0032               .IIF    NB,RA80$K_LEN,  RA80$K_LEN:
                          000024BE                 .RESTORE
                30 38 41 52 00' 24BE               .ASCIC  "RA80"  ; RA80 name
                          04    24BE
```

```
                      32  24C3                .BYTE   RA80$K_LEN                ; RA80$K_LEN of resultant P-table
                      FF  24C4                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  24C5                .WORD   ^A""    ; Two character mnemonic for QIO RA80
                      80  24C7                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  24C8                .Byte   Pd$_Name            ; Directive op-code
                      1B  24C9                .Byte   PTD$M_ENDDEVICE             ; Enforced DU codes
                55 44 00' 24CA                .Ascic  "DU"    ; Enforced device name prefix
                      02  24CA
                      86  24CD                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  24CE                .LONG   HP$M_ALLOC                  ; Constant
                      88  24D2                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  24D3                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  24D5                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  24D6                .BYTE   8                   ; 8 of data field
                      81  24D7                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          24D8    764 RA81:   $DS_RA81                                ;       [21]
                          24D8                .SAVE   LOCAL_BLOCK
                          24D8                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RA81_LEN,       HP$K_RA81_LEN:
                          0032                .IIF    NB,RA81$K_LEN,  RA81$K_LEN:
                      000024D8                .RESTORE
             31 38 41 52 00' 24D8             .ASCIC  "RA81"  ; RA81 name
                      04  24D8
                      32  24DD                .BYTE   RA81$K_LEN                  ; RA81$K_LEN of resultant P-table
                      FF  24DE                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  24DF                .WORD   ^A""    ; Two character mnemonic for QIO RA81
                      80  24E1                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  24E2                .Byte   Pd$_Name            ; Directive op-code
                      1B  24E3                .Byte   PTD$M_ENDDEVICE             ; Enforced DU codes
                55 44 00' 24E4                .Ascic  "DU"    ; Enforced device name prefix
                      02  24E4
                      86  24E7                .BYTE   Pd$_Literal         ; Indicate literal
                00000001  24E8                .LONG   HP$M_ALLOC                  ; Constant
                      88  24EC                .BYTE   Pd$_Store           ; Indicate store operation
                    000A  24ED                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                      00  24EF                .BYTE   0                   ; 0 HP$B_FLAGS to data
                      08  24F0                .BYTE   8                   ; 8 of data field
                      81  24F1                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                          24F2    765 RA82:   $DS_RA82                                ;       [70]
                          24F2                .SAVE   LOCAL_BLOCK
                          24F2                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RA82_LEN,       HP$K_RA82_LEN:
                          0032                .IIF    NB,RA82$K_LEN,  RA82$K_LEN:
                      000024F2                .RESTORE
             32 38 41 52 00' 24F2             .ASCIC  "RA82"  ; RA82 name
                      04  24F2
                      32  24F7                .BYTE   RA82$K_LEN                  ; RA82$K_LEN of resultant P-table
                      FF  24F8                .BYTE   255                 ; Maximum unit number                       ;G:65
                    0000  24F9                .WORD   ^A""    ; Two character mnemonic for QIO RA82
                      80  24FB                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                      8D  24FC                .Byte   Pd$_Name            ; Directive op-code
                      1B  24FD                .Byte   PTD$M_ENDDEVICE             ; Enforced DU codes
                55 44 00' 24FE                .Ascic  "DU"    ; Enforced device name prefix
                      02  24FE
                      86  2501                .BYTE   Pd$_Literal         ; Indicate literal
```

```
                    00000001  2502                .LONG   HP$M_ALLOC                      ; Constant
                          88  2506                .BYTE   Pd$_Store       ; Indicate store operation
                        000A  2507                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                          00  2509                .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  250A                .BYTE   8               ; 8 of data field
                          81  250B                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              250C    766 RA90:   $DS_RA90                                                        ;G:54
                              250C                .SAVE   LOCAL_BLOCK
                              250C                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$K_RA90_LEN,       HP$K_RA90_LEN:
                              0032                .IIF    NB,RA90$K_LEN,  RA90$K_LEN:
                          0000250C                .RESTORE
                 30 39 41 52 00' 250C             .ASCIC  "RA90"  ; RA90 name
                          04  250C
                          32  2511                .BYTE   RA90$K_LEN              ; RA90$K_LEN of resultant P-table
                          00  2512                .BYTE   0                                                   ;G:77
                        0000  2513                .WORD   ^A""    ; Two character mnemonic for QIO RA90
                          80  2515                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8E  2516                .BYTE   PD$_EPB         ; EXTENDED P-TABLE BLOCK                ;G:77
                        0FFF  2517                .WORD   4095                                                ;G:77
                          8D  2519                .Byte   Pd$_Name        ; Directive op-code
                          1B  251A                .Byte   PTD$M_ENDDEVICE         ; Enforced DU codes
                    55 44 00' 251B                .Ascic  "DU"    ; Enforced device name prefix
                          02  251B
                          86  251E                .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  251F                .LONG   HP$M_ALLOC                      ; Constant
                          88  2523                .BYTE   Pd$_Store       ; Indicate store operation
                        000A  2524                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                          00  2526                .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  2527                .BYTE   8               ; 8 of data field
                          81  2528                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              2529    767 RB730:  $DS_RB730                                       ;       [16]
                              2529                .SAVE   LOCAL_BLOCK
                              2529                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$L_RB730_CSR,      HP$L_RB730_CSR:
                              0032                .IIF    NB,RB730$L_CSR, RB730$L_CSR:
                    00000036  0032                .IIF    NB,,BLKL,       .BLKL   1
                              0036                .IIF    NB,HP$B_RB730_BR,       HP$B_RB730_BR:
                              0036                .IIF    NB,RB730$B_BR,  RB730$B_BR:
                    00000037  0036                .IIF    NB,,BLKB,       .BLKB   1
                              0037                .IIF    NB,HP$K_RB730_LEN,      HP$K_RB730_LEN:
                              0037                .IIF    NB,RB730$K_LEN, RB730$K_LEN:
                          00002529                .RESTORE
              30 33 37 42 52 00' 2529             .ASCIC  "RB730" ; RB730 name
                          05  2529
                          37  252F                .BYTE   RB730$K_LEN             ; RB730$K_LEN of resultant P-table
                          00  2530                .BYTE   0               ; Maximum unit number                ;G:65
                        5144  2531                .WORD   ^A"DQ"  ; Two character mnemonic for QIO RB730 DQ
                          80  2533                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  2534                .Byte   Pd$_Name        ; Directive op-code
                          02  2535                .Byte   Ptd$M_Controller                ; Enforced DQ codes
                    51 44 00' 2536                .Ascic  "DQ"    ; Enforced device name prefix
                          02  2536
                          86  2539                .BYTE   Pd$_Literal     ; Indicate literal
                    00F26200  253A                .LONG   ^xF26200                        ; Constant
```

```
                      88  253E                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  253F                .WORD   hp$a_device             ; Byte hp$a_device to data
                      00  2541                .BYTE   0               ; 0 hp$a_device to data
                      1D  2542                .BYTE   29              ; 29 of data field
                      86  2543                .BYTE   Pd$_Literal      ; Indicate literal
                000000A8  2544                .LONG   ^o250           ; Constant
                      88  2548                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  2549                .WORD   hp$w_vector             ; Byte hp$w_vector to data
                      00  254B                .BYTE   0               ; 0 hp$w_vector to data
                      09  254C                .BYTE   9               ; 9 of data field
                      86  254D                .BYTE   Pd$_Literal      ; Indicate literal
                00000005  254E                .LONG   5               ; Constant
                      88  2552                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  2553                .WORD   rb730$b_br              ; Byte rb730$b_br to data
                      00  2555                .BYTE   0               ; 0 rb730$b_br to data
                      08  2556                .BYTE   8               ; 8 of data field
                      81  2557                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2558    768 RCF25:  $DS_RCF25                               ;       [33]
                          2558                .SAVE   LOCAL_BLOCK
                          2558                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RCF25_LEN,      HP$K_RCF25_LEN:
                          0032                .IIF    NB,RCF25$K_LEN, RCF25$K_LEN:
                          0000255B            .RESTORE
             35 32 46 43 52 00' 255B          .ASCIC  "RCF25" ; RCF25 name
                      05  255B
                      32  2563                .BYTE   RCF25$K_LEN             ; RCF25$K_LEN of resultant P-table
                      FF  2564                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  2565                .WORD   ^A""    ; Two character mnemonic for QIO RCF25
                      80  2567                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2568                .Byte   Pd$_Name        ; Directive op-code
                      1B  2569                .Byte   Ptd$M_EndDevice         ; Enforced DA codes
                41 44 00' 256A                .Ascic  "DA"    ; Enforced device name prefix
                      02  256A
                      86  256D                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  256E                .LONG   HP$M_ALLOC              ; Constant
                      88  2572                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  2573                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  2575                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  2576                .BYTE   8               ; 8 of data field
                      81  2577                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2578    769 RC25:   $DS_RC25                                ;       [32]
                          2578                .SAVE   LOCAL_BLOCK
                          2578                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RC25_LEN,       HP$K_RC25_LEN:
                          0032                .IIF    NB,RC25$K_LEN,  RC25$K_LEN:
                          0000257D            .RESTORE
             35 32 43 52 00' 257D             .ASCIC  "RC25"  ; RC25 name
                      04  257D
                      32  2582                .BYTE   RC25$K_LEN              ; RC25$K_LEN of resultant P-table
                      FF  2583                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  2584                .WORD   ^A""    ; Two character mnemonic for QIO RC25
                      80  2586                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2587                .Byte   Pd$_Name        ; Directive op-code
                      1B  2588                .Byte   Ptd$M_EndDevice         ; Enforced DA codes
                41 44 00' 2589                .Ascic  "DA"    ; Enforced device name prefix
```

```
                    02  257F
                    86  2582                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  2583                .LONG   HP$M_ALLOC              ; Constant
                    88  2587                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  2588                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  258A                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  258B                .BYTE   8               ; 8 of data field
                    81  258C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        258D    770 RD52:   $DS_RD52                                ;       [67]
                        258D                .SAVE   LOCAL_BLOCK
                        258D                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,RD52$K_LEN,  RD52$K_LEN:
                    0000258D                .RESTORE
           32 35 44 52 00' 258D             .ASCIC  "RD52"  ; RD52 name
                    04  258D
                    32  2592                .BYTE   RD52$K_LEN              ; RD52$K_LEN of resultant P-table
                    FF  2593                .BYTE   255             ; Maximum unit number                   ;G:65
                  0000  2594                .WORD   ^A""    ; Two character mnemonic for QIO RD52
                    80  2596                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  2597                .Byte   Pd$_Name        ; Directive op-code
                    1B  2598                .Byte   Ptd$M_EndDevice         ; Enforced DU codes
              55 44 00' 2599                .Ascic  "DU"    ; Enforced device name prefix
                    02  2599
                    86  259C                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  259D                .LONG   HP$M_ALLOC              ; Constant
                    88  25A1                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  25A2                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  25A4                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  25A5                .BYTE   8               ; 8 of data field
                    81  25A6                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        25A7    771 RD53:   $DS_RD53                                ;       [67]
                        25A7                .SAVE   LOCAL_BLOCK
                        25A7                .PSECT  $ABS$,ABS
              00000032  0050                .=HP$A_DEPENDENT
                        0032                .IIF    NB,RD53$K_LEN,  RD53$K_LEN:
                    000025A7                .RESTORE
           33 35 44 52 00' 25A7             .ASCIC  "RD53"  ; RD53 name
                    04  25A7
                    32  25AC                .BYTE   RD53$K_LEN              ; RD53$K_LEN of resultant P-table
                    FF  25AD                .BYTE   255             ; Maximum unit number                   ;G:65
                  0000  25AE                .WORD   ^A""    ; Two character mnemonic for QIO RD53
                    80  25B0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  25B1                .Byte   Pd$_Name        ; Directive op-code
                    1B  25B2                .Byte   Ptd$M_EndDevice         ; Enforced DU codes
              55 44 00' 25B3                .Ascic  "DU"    ; Enforced device name prefix
                    02  25B3
                    86  25B6                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  25B7                .LONG   HP$M_ALLOC              ; Constant
                    88  25BB                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  25BC                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  25BE                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  25BF                .BYTE   8               ; 8 of data field
                    81  25C0                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        25C1    772 RD54:   $DS_RD54                                ;       [74]            ;G:12
                        25C1                .SAVE   LOCAL_BLOCK
                        25C1                .PSECT  $ABS$,ABS
```

```
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RD54$K_LEN,  RD54$K_LEN:
                          000025C1            .RESTORE
                34 35 44 52 00' 25C1          .ASCIC  "RD54"  ; RD54 name
                          04  25C1
                          32  25C6            .BYTE   RD54$K_LEN              ; RD54$K_LEN of resultant P-table
                          FF  25C7            .BYTE   255             ; Maximum unit number                   ;G:65
                        0000  25C8            .WORD   ^A""    ; Two character mnemonic for QIO RD54
                          80  25CA            .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  25CB            .Byte   Pd$_Name        ; Directive op-code
                          1B  25CC            .Byte   Ptd$M_EndDevice         ; Enforced DU codes
                55 44 00' 25CD                .Ascic  "DU"    ; Enforced device name prefix
                          02  25CD
                          86  25D0            .BYTE   Pd$_Literal     ; Indicate literal
                00000001  25D1                .LONG   HP$M_ALLOC              ; Constant
                          88  25D5            .BYTE   Pd$_Store       ; Indicate store operation
                        000A  25D6            .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  25D8            .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  25D9            .BYTE   8               ; 8 of data field
                          81  25DA            .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              25DB    773 RH750:  $DS_RH750
                              25DB            .SAVE   LOCAL_BLOCK
                              25DB            .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_RH750_BR,       HP$B_RH750_BR:
                          0032                .IIF    NB,RH750$B_BR,  RH750$B_BR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_RH750_LEN,      HP$K_RH750_LEN:
                          0033                .IIF    NB,RH750$K_LEN, RH750$K_LEN:
                          000025DB            .RESTORE
             30 35 37 48 52 00' 25DB          .ASCIC  "RH750" ; RH750 name
                          05  25DB
                          33  25E1            .BYTE   RH750$K_LEN             ; RH750$K_LEN of resultant P-table
                          08  25E2            .BYTE   8               ; Maximum unit number                   ;G:65
                        0000  25E3            .WORD   ^A""    ; Two character mnemonic for QIO RH750
                          80  25E5            .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  25E6            .Byte   Pd$_Name        ; Directive op-code
                          01  25E7            .Byte   Ptd$M_Unit              ; Enforced RH codes
                48 52 00' 25E8                .Ascic  "RH"    ; Enforced device name prefix
                          02  25E8
                          86  25EB            .BYTE   Pd$_Literal     ; Indicate literal
                40F28000  25EC                .LONG   ^X40F28000              ; Constant
                          88  25F0            .BYTE   Pd$_Store       ; Indicate store operation
                        0018  25F1            .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  25F3            .BYTE   0               ; 0 HP$A_DEVICE to data
                          20  25F4            .BYTE   32              ; 32 of data field
                          87  25F5            .BYTE   Pd$_Fetch       ; Indicate fetch operation
                        000B  25F6            .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                          00  25F8            .BYTE   0               ; 0 HP$B_DRIVE to data
                          08  25F9            .BYTE   8               ; 8 of data field
                          88  25FA            .BYTE   Pd$_Store       ; Indicate store operation
                        0018  25FB            .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          0D  25FD            .BYTE   13              ; 13 HP$A_DEVICE to data
                          02  25FE            .BYTE   2               ; 2 of data field
                          87  25FF            .BYTE   Pd$_Fetch       ; Indicate fetch operation
                        0018  2600            .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          0D  2602            .BYTE   13              ; 13 HP$A_DEVICE to data
```

```
                      04  2603                .BYTE   4                       ; 4 of data field
                      88  2604                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  2605                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      02  2607                .BYTE   2                       ; 2 HP$W_VECTOR to data
                      04  2608                .BYTE   4                       ; 4 of data field
                      82  2609                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                 52 42 00' 260A                .ASCIC  "BR"    ; BR string
                      02  260A
            00000007 00000004  260D                .LONG   4,7     ; 4 , 7 limits on input
                      88  2615                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  2616                .WORD   RH750$B_BR                      ; Byte RH750$B_BR to data
                      00  2618                .BYTE   0                       ; 0 RH750$B_BR to data
                      08  2619                .BYTE   8                       ; 8 of data field
                      88  261A                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  261B                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      06  261D                .BYTE   6                       ; 6 HP$W_VECTOR to data
                      03  261E                .BYTE   3                       ; 3 of data field
                      87  261F                .BYTE   Pd$_Fetch               ; Indicate fetch operation
                    0018  2620                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  2622                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      20  2623                .BYTE   32                      ; 32 of data field
                      88  2624                .BYTE   Pd$_Store               ; Indicate store operation
                    001C  2625                .WORD   HP$A_DVA                        ; Byte HP$A_DVA to data
                      00  2627                .BYTE   0                       ; 0 HP$A_DVA to data
                      20  2628                .BYTE   32                      ; 32 of data field
                      86  2629                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  262A                .LONG   1                       ; Constant
                      88  262E                .BYTE   Pd$_Store               ; Indicate store operation
                    001C  262F                .WORD   HP$A_DVA                        ; Byte HP$A_DVA to data
                      0A  2631                .BYTE   10                      ; 10 HP$A_DVA to data
                      01  2632                .BYTE   1                       ; 1 of data field
                      81  2633                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          2634    774 RH780:  $DS_RH780
                          2634                .SAVE   LOCAL_BLOCK
                          2634                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_RH780_TR,       HP$B_RH780_TR:
                          0032                .IIF    NB,RH780$B_TR,  RH780$B_TR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$B_RH780_BR,       HP$B_RH780_BR:
                          0033                .IIF    NB,RH780$B_BR,  RH780$B_BR:
                00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,HP$K_RH780_LEN,      HP$K_RH780_LEN:
                          0034                .IIF    NB,RH780$K_LEN, RH780$K_LEN:
                      00002634                .RESTORE
          30 38 37 48 52 00' 2634                .ASCIC  "RH780" ; RH780 name
                      05  2634
                      34  263A                .BYTE   RH780$K_LEN                     ; RH780$K_LEN of resultant P-table
                      08  263B                .BYTE   8                       ; Maximum unit number                           ;G:65
                    0000  263C                .WORD   ^A""    ; Two character mnemonic for QIO RH780
                      80  263E                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  263F                .Byte   Pd$_Name                ; Directive op-code
                      01  2640                .Byte   Ptd$M_Unit                      ; Enforced RH codes
                 48 52 00' 2641                .Ascic  "RH"    ; Enforced device name prefix
                      02  2641
                      82  2644                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                 52 54 00' 2645                .ASCIC  "TR"    ; TR string
```

```
                     02  2645
0000000F 00000001    2648                .LONG   1,15    ; 1 , 15 limits on input
                     88  2650                .BYTE   Pd$_Store       ; Indicate store operation
                   0032  2651                .WORD   RH780$B_TR              ; Byte RH780$B_TR to data
                     00  2653                .BYTE   0               ; 0 RH780$B_TR to data
                     08  2654                .BYTE   8               ; 8 of data field
                     88  2655                .BYTE   Pd$_Store       ; Indicate store operation
                   0018  2656                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                     0D  2658                .BYTE   13              ; 13 HP$A_DEVICE to data
                     04  2659                .BYTE   4               ; 4 of data field
                     88  265A                .BYTE   Pd$_Store       ; Indicate store operation
                   0024  265B                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                     02  265D                .BYTE   2               ; 2 HP$W_VECTOR to data
                     04  265E                .BYTE   4               ; 4 of data field
                     82  265F                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 2660             .ASCIC  "BR"    ; BR string
                     02  2660
00000007 00000004    2663                .LONG   4,7     ; 4 , 7 limits on input
                     88  266B                .BYTE   Pd$_Store       ; Indicate store operation
                   0033  266C                .WORD   RH780$B_BR              ; Byte RH780$B_BR to data
                     00  266E                .BYTE   0               ; 0 RH780$B_BR to data
                     08  266F                .BYTE   8               ; 8 of data field
                     88  2670                .BYTE   Pd$_Store       ; Indicate store operation
                   0024  2671                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                     06  2673                .BYTE   6               ; 6 HP$W_VECTOR to data
                     02  2674                .BYTE   2               ; 2 of data field
                     86  2675                .BYTE   Pd$_Literal     ; Indicate literal
                00000006 2676                .LONG   6               ; Constant
                     88  267A                .BYTE   Pd$_Store       ; Indicate store operation
                   0018  267B                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                     1C  267D                .BYTE   28              ; 28 HP$A_DEVICE to data
                     04  267E                .BYTE   4               ; 4 of data field
                     87  267F                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                   0018  2680                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                     00  2682                .BYTE   0               ; 0 HP$A_DEVICE to data
                     20  2683                .BYTE   32              ; 32 of data field
                     88  2684                .BYTE   Pd$_Store       ; Indicate store operation
                   001C  2685                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                     00  2687                .BYTE   0               ; 0 HP$A_DVA to data
                     20  2688                .BYTE   32              ; 32 of data field
                     86  2689                .BYTE   Pd$_Literal     ; Indicate literal
                00000001 268A                .LONG   1               ; Constant
                     88  268E                .BYTE   Pd$_Store       ; Indicate store operation
                   001C  268F                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                     0A  2691                .BYTE   10              ; 10 HP$A_DVA to data
                     01  2692                .BYTE   1               ; 1 of data field
                     88  2693                .BYTE   Pd$_Store       ; Indicate store operation
                   0024  2694                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                     08  2696                .BYTE   8               ; 8 HP$W_VECTOR to data
                     01  2697                .BYTE   1               ; 1 of data field
                     81  2698                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                         2699    775 RL01:   $DS_RL01
                         2699                .SAVE   LOCAL_BLOCK
                         2699                .PSECT  $ABS$,ABS
                00000032 0050                .=HP$A_DEPENDENT
                         0032                .IIF    NB,HP$K_RL01_LEN,       HP$K_RL01_LEN:
                         0032                .IIF    NB,RL01$K_LEN,  RL01$K_LEN:
```

```
                    00002699                    .RESTORE
          31 30 4C 52 00' 2699                  .ASCIC  "RL01"  ; RL01 name
                      04  2699
                      32  269E                  .BYTE   RL01$K_LEN              ; RL01$K_LEN of resultant P-table
                      08  269F                  .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  26A0                  .WORD   ^A""    ; Two character mnemonic for QIO RL01
                      80  26A2                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  26A3                  .Byte   Pd$_Name        ; Directive op-code
                      1B  26A4                  .Byte   Ptd$M_EndDevice         ; Enforced DL codes
                 4C 44 00' 26A5                 .Ascic  "DL"    ; Enforced device name prefix
                      02  26A5
                      86  26A8                  .BYTE   Pd$_Literal     ; Indicate literal
                00000001  26A9                  .LONG   HP$M_ALLOC              ; Constant
                      88  26AD                  .BYTE   Pd$_Store       ; Indicate store operation
                    000A  26AE                  .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  26B0                  .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  26B1                  .BYTE   8               ; 8 of data field
                      81  26B2                  .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          26B3    776 RL02:     $DS_RL02
                          26B3                  .SAVE   LOCAL_BLOCK
                          26B3                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,HP$K_RL02_LEN,       HP$K_RL02_LEN:
                          0032                  .IIF    NB,RL02$K_LEN,  RL02$K_LEN:
                    000026B3                    .RESTORE
          32 30 4C 52 00' 26B3                  .ASCIC  "RL02"  ; RL02 name
                      04  26B3
                      32  26B8                  .BYTE   RL02$K_LEN              ; RL02$K_LEN of resultant P-table
                      08  26B9                  .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  26BA                  .WORD   ^A""    ; Two character mnemonic for QIO RL02
                      80  26BC                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  26BD                  .Byte   Pd$_Name        ; Directive op-code
                      1B  26BE                  .Byte   Ptd$M_EndDevice         ; Enforced DL codes
                 4C 44 00' 26BF                 .Ascic  "DL"    ; Enforced device name prefix
                      02  26BF
                      86  26C2                  .BYTE   Pd$_Literal     ; Indicate literal
                00000001  26C3                  .LONG   HP$M_ALLOC              ; Constant
                      88  26C7                  .BYTE   Pd$_Store       ; Indicate store operation
                    000A  26C8                  .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  26CA                  .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  26CB                  .BYTE   8               ; 8 of data field
                      81  26CC                  .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          26CD    777 RL11:     $DS_RL11
                          26CD                  .SAVE   LOCAL_BLOCK
                          26CD                  .PSECT  $ABS$,ABS
                00000032  0050                  .=HP$A_DEPENDENT
                          0032                  .IIF    NB,HP$L_RL11_CSR,       HP$L_RL11_CSR:
                          0032                  .IIF    NB,RL11$L_CSR,  RL11$L_CSR:
                00000036  0032                  .IIF    NB,.BLKL,       .BLKL   1
                          0036                  .IIF    NB,HP$B_RL11_BR,        HP$B_RL11_BR:
                          0036                  .IIF    NB,RL11$B_BR,   RL11$B_BR:
                00000037  0036                  .IIF    NB,.BLKB,       .BLKB   1
                          0037                  .IIF    NB,HP$K_RL11_LEN,       HP$K_RL11_LEN:
                          0037                  .IIF    NB,RL11$K_LEN,  RL11$K_LEN:
                    000026CD                    .RESTORE
          31 31 4C 52 00' 26CD                  .ASCIC  "RL11"  ; RL11 name
                      04  26CD
```

```
                        37  26D2                .BYTE   RL11$K_LEN                ; RL11$K_LEN of resultant P-table
                        00  26D3                .BYTE   0                       ; Maximum unit number                           ;G:65
                      4C44  26D4                .WORD   ^A"DL"  ; Two character mnemonic for QIO RL11 DL
                        80  26D6                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                        8D  26D7                .Byte   Pd$_Name                ; Directive op-code
                        02  26D8                .Byte   Ptd$M_Controller                ; Enforced DL codes
                  4C 44 00' 26D9                .Ascic  "DL"    ; Enforced device name prefix
                        02  26D9
                        83  26DC                .BYTE   Pd$_Octal               ; Indicate scan octal
                52 53 43 00' 26DD                .ASCIC  "CSR"   ; CSR string
                        03  26DD
            0003FFFE 0003E000  26E1                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                        88  26E9                .BYTE   Pd$_Store               ; Indicate store operation
                      0032  26EA                .WORD   RL11$L_CSR                      ; Byte RL11$L_CSR to data
                        00  26EC                .BYTE   0                       ; 0 RL11$L_CSR to data
                        20  26ED                .BYTE   32                      ; 32 of data field
                        88  26EE                .BYTE   Pd$_Store               ; Indicate store operation
                      0018  26EF                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                        00  26F1                .BYTE   0                       ; 0 HP$A_DEVICE to data
                        12  26F2                .BYTE   18                      ; 18 of data field
                        83  26F3                .BYTE   Pd$_Octal               ; Indicate scan octal
          52 4F 54 43 45 56 00' 26F4                .ASCIC  "VECTOR"                ; VECTOR string
                        06  26F4
            000001FE 00000002  26FB                .LONG   ^O2,^O776               ; 2 , 776 limits on input
                        88  2703                .BYTE   Pd$_Store               ; Indicate store operation
                      0024  2704                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                        00  2706                .BYTE   0                       ; 0 HP$W_VECTOR to data
                        09  2707                .BYTE   9                       ; 9 of data field
                        82  2708                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                52 42 00' 2709                .ASCIC  "BR"    ; BR string
                        02  2709
            00000007 00000004  270C                .LONG   4,7     ; 4 , 7 limits on input
                        88  2714                .BYTE   Pd$_Store               ; Indicate store operation
                      0036  2715                .WORD   RL11$B_BR                       ; Byte RL11$B_BR to data
                        00  2717                .BYTE   0                       ; 0 RL11$B_BR to data
                        08  2718                .BYTE   8                       ; 8 of data field
                        81  2719                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                            271A    778 RK06:   $DS_RK06
                            271A                .SAVE   LOCAL_BLOCK
                            271A                .PSECT  $ABS$,ABS
                00000032    0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$K_RK06_LEN,       HP$K_RK06_LEN:
                            0032                .IIF    NB,RK06$K_LEN,  RK06$K_LEN:
                        0000271A                .RESTORE
             36 30 4B 52 00' 271A                .ASCIC  "RK06"  ; RK06 name
                        04  271A
                        32  271F                .BYTE   RK06$K_LEN              ; RK06$K_LEN of resultant P-table
                        08  2720                .BYTE   8                       ; Maximum unit number                           ;G:65
                      0000  2721                .WORD   ^A""    ; Two character mnemonic for QIO RK06
                        80  2723                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                        8D  2724                .Byte   Pd$_Name                ; Directive op-code
                        1B  2725                .Byte   Ptd$M_EndDevice                 ; Enforced DM codes
                  4D 44 00' 2726                .Ascic  "DM"    ; Enforced device name prefix
                        02  2726
                        86  2729                .BYTE   Pd$_Literal             ; Indicate literal
                  00000001  272A                .LONG   HP$M_ALLOC                      ; Constant
                        88  272E                .BYTE   Pd$_Store               ; Indicate store operation
```

```
                     000A  272F                .WORD   HP$B_FLAGS                ; Byte HP$B_FLAGS to data
                       00  2731                .BYTE   0                  ; 0 HP$B_FLAGS to data
                       08  2732                .BYTE   8                  ; 8 of data field
                       81  2733                .BYTE   Pd$_End            ; Indicate end of PT-descriptor
                           2734    779 RK07:   $DS_RK07
                           2734                .SAVE   LOCAL_BLOCK
                           2734                .PSECT  $ABS$,ABS
                   00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_RK07_LEN,       HP$K_RK07_LEN:
                           0032                .IIF    NB,RK07$K_LEN,  RK07$K_LEN:
                       00002734                .RESTORE
               37 30 4B 52 00' 2734                .ASCIC  "RK07"  ; RK07 name
                       04  2734
                       32  2739                .BYTE   RK07$K_LEN              ; RK07$K_LEN of resultant P-table
                       08  273A                .BYTE   8                  ; Maximum unit number                ;G:65
                     0000  273B                .WORD   ^A""    ; Two character mnemonic for QIO RK07
                       80  273D                .BYTE   Pd$_Start          ; Constant to identify start of descriptor
                       8D  273E                .Byte   Pd$_Name           ; Directive op-code
                       1B  273F                .Byte   Ptd$M_EndDevice            ; Enforced DM codes
                   4D 44 00' 2740                .Ascic  "DM"    ; Enforced device name prefix
                       02  2740
                       86  2743                .BYTE   Pd$_Literal        ; Indicate literal
                   00000001  2744                .LONG   HP$M_ALLOC                 ; Constant
                       88  2748                .BYTE   Pd$_Store          ; Indicate store operation
                     000A  2749                .WORD   HP$B_FLAGS                 ; Byte HP$B_FLAGS to data
                       00  274B                .BYTE   0                  ; 0 HP$B_FLAGS to data
                       08  274C                .BYTE   8                  ; 8 of data field
                       81  274D                .BYTE   Pd$_End            ; Indicate end of PT-descriptor
                           274E    780 RK611:  $DS_RK611
                           274E                .SAVE   LOCAL_BLOCK
                           274E                .PSECT  $ABS$,ABS
                   00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$L_RK611_CSR,      HP$L_RK611_CSR:
                           0032                .IIF    NB,RK611$L_CSR, RK611$L_CSR:
                   00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                           0036                .IIF    NB,HP$B_RK611_BR,       HP$B_RK611_BR:
                           0036                .IIF    NB,RK611$B_BR,  RK611$B_BR:
                   00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                           0037                .IIF    NB,HP$K_RK611_LEN,      HP$K_RK611_LEN:
                           0037                .IIF    NB,RK611$K_LEN, RK611$K_LEN:
                       0000274E                .RESTORE
           31 31 36 4B 52 00' 274E                .ASCIC  "RK611" ; RK611 name
                       05  274E
                       37  2754                .BYTE   RK611$K_LEN             ; RK611$K_LEN of resultant P-table
                       00  2755                .BYTE   0                  ; Maximum unit number                ;G:65
                     4D44  2756                .WORD   ^A"DM"  ; Two character mnemonic for QIO RK611 DM
                       80  2758                .BYTE   Pd$_Start          ; Constant to identify start of descriptor
                       8D  2759                .Byte   Pd$_Name           ; Directive op-code
                       02  275A                .Byte   Ptd$M_Controller                ; Enforced DM codes
                   4D 44 00' 275B                .Ascic  "DM"    ; Enforced device name prefix
                       02  275B
                       83  275E                .BYTE   Pd$_Octal          ; Indicate scan octal
                 52 53 43 00' 275F                .ASCIC  "CSR"   ; CSR string
                       03  275F
           0003FFFE 0003E000  2763                .LONG   ^O760000,^O777776          ; 760000 , 777776 limits on input
                       88  276B                .BYTE   Pd$_Store          ; Indicate store operation
                     0032  276C                .WORD   RK611$L_CSR                ; Byte RK611$L_CSR to data
```

```
                      00  276E                .BYTE   0                       ; 0 RK611$L_CSR to data
                      20  276F                .BYTE   32                      ; 32 of data field
                      88  2770                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  2771                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      00  2773                .BYTE   0                       ; 0 HP$A_DEVICE to data
                      12  2774                .BYTE   18                      ; 18 of data field
                      83  2775                .BYTE   Pd$_Octal               ; Indicate scan octal
          52 4F 54 43 45 56 00' 2776          .ASCIC  "VECTOR"                ; VECTOR string
                      06  2776
             000001FE 00000002  277D          .LONG   ^O2,^O776               ; 2 , 776 limits on input
                      88  2785                .BYTE   Pd$_Store               ; Indicate store operation
                    0024  2786                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                      00  2788                .BYTE   0                       ; 0 HP$W_VECTOR to data
                      09  2789                .BYTE   9                       ; 9 of data field
                      82  278A                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                 52 42 00' 278B               .ASCIC  "BR"    ; BR string
                      02  278B
             00000007 00000004  278E          .LONG   4,7     ; 4 , 7 limits on input
                      88  2796                .BYTE   Pd$_Store               ; Indicate store operation
                    0036  2797                .WORD   RK611$B_BR                      ; Byte RK611$B_BR to data
                      00  2799                .BYTE   0                       ; 0 RK611$B_BR to data
                      08  279A                .BYTE   8                       ; 8 of data field
                      81  279B                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          279C    781 RM03:   $DS_RM03
                          279C                .SAVE   LOCAL_BLOCK
                          279C                .PSECT  $ABS$,ABS
                 00000032  0050               .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RM03_LEN,       HP$K_RM03_LEN:
                          0032                .IIF    NB,RM03$K_LEN,  RM03$K_LEN:
                          279C                .RESTORE
          33 30 4D 52 00' 279C                .ASCIC  "RM03"  ; RM03 name
                      04  279C
                      32  27A1                .BYTE   RM03$K_LEN                      ; RM03$K_LEN of resultant P-table
                      08  27A2                .BYTE   8                       ; Maximum unit number                   ;G:65
                    5244  27A3                .WORD   ^A"DR"  ; Two character mnemonic for QIO RM03 DR
                      80  27A5                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  27A6                .Byte   Pd$_Name                ; Directive op-code
                      03  27A7                .Byte   Ptd$M_Device                    ; Enforced DR codes
                 52 44 00' 27A8               .Ascic  "DR"    ; Enforced device name prefix
                      02  27A8
                      86  27AB                .BYTE   Pd$_Literal             ; Indicate literal
                 00000001  27AC               .LONG   HP$M_ALLOC                      ; Constant
                      88  27B0                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  27B1                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  27B3                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  27B4                .BYTE   8                       ; 8 of data field
                      87  27B5                .BYTE   Pd$_Fetch               ; Indicate fetch operation
                    000B  27B6                .WORD   HP$B_DRIVE                      ; Byte HP$B_DRIVE to data
                      00  27B8                .BYTE   0                       ; 0 HP$B_DRIVE to data
                      08  27B9                .BYTE   8                       ; 8 of data field
                      88  27BA                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  27BB                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      07  27BD                .BYTE   7                       ; 7 HP$A_DEVICE to data
                      03  27BE                .BYTE   3                       ; 3 of data field
                      81  27BF                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          27C0    782 RM05:   $DS_RM05
                          27C0                .SAVE   LOCAL_BLOCK
```

```
                    27C0                .PSECT  $ABS$,ABS
         00000032   0050                .=HP$A_DEPENDENT
                    0032                .IIF    NB,HP$K_RM05_LEN,       HP$K_RM05_LEN:
                    0032                .IIF    NB,RM05$K_LEN,  RM05$K_LEN:
                000027C0                .RESTORE
   35 30 4D 52 00'  27C0                .ASCIC  "RM05"  ; RM05 name
                04  27C0
                32  27C5                .BYTE   RM05$K_LEN              ; RM05$K_LEN of resultant P-table
                08  27C6                .BYTE   8               ; Maximum unit number                   ;G:65
              5244  27C7                .WORD   ^A"DR"  ; Two character mnemonic for QIO RM05 DR
                80  27C9                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                8D  27CA                .Byte   Pd$_Name        ; Directive op-code
                03  27CB                .Byte   Ptd$M_Device            ; Enforced DR codes
         52 44 00'  27CC                .Ascic  "DR"    ; Enforced device name prefix
                02  27CC
                86  27CF                .BYTE   Pd$_Literal     ; Indicate literal
         00000001   27D0                .LONG   HP$M_ALLOC              ; Constant
                88  27D4                .BYTE   Pd$_Store       ; Indicate store operation
              000A  27D5                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                00  27D7                .BYTE   0               ; 0 HP$B_FLAGS to data
                08  27D8                .BYTE   8               ; 8 of data field
                87  27D9                .BYTE   Pd$_Fetch       ; Indicate fetch operation
              000B  27DA                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                00  27DC                .BYTE   0               ; 0 HP$B_DRIVE to data
                08  27DD                .BYTE   8               ; 8 of data field
                88  27DE                .BYTE   Pd$_Store       ; Indicate store operation
              0018  27DF                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                07  27E1                .BYTE   7               ; 7 HP$A_DEVICE to data
                03  27E2                .BYTE   3               ; 3 of data field
                81  27E3                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                    27E4    783 RM80:   $DS_RM80
                    27E4                .SAVE   LOCAL_BLOCK
                    27E4                .PSECT  $ABS$,ABS
         00000032   0050                .=HP$A_DEPENDENT
                    0032                .IIF    NB,HP$K_RM80_LEN,       HP$K_RM80_LEN:
                    0032                .IIF    NB,RM80$K_LEN,  RM80$K_LEN:
                000027E4                .RESTORE
   30 38 4D 52 00'  27E4                .ASCIC  "RM80"  ; RM80 name
                04  27E4
                32  27E9                .BYTE   RM80$K_LEN              ; RM80$K_LEN of resultant P-table
                08  27EA                .BYTE   8               ; Maximum unit number                   ;G:65
              5244  27EB                .WORD   ^A"DR"  ; Two character mnemonic for QIO RM80 DR
                80  27ED                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                8D  27EE                .Byte   Pd$_Name        ; Directive op-code
                03  27EF                .Byte   Ptd$M_Device            ; Enforced DR codes
         52 44 00'  27F0                .Ascic  "DR"    ; Enforced device name prefix
                02  27F0
                86  27F3                .BYTE   Pd$_Literal     ; Indicate literal
         00000001   27F4                .LONG   HP$M_ALLOC              ; Constant
                88  27F8                .BYTE   Pd$_Store       ; Indicate store operation
              000A  27F9                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                00  27FB                .BYTE   0               ; 0 HP$B_FLAGS to data
                08  27FC                .BYTE   8               ; 8 of data field
                87  27FD                .BYTE   Pd$_Fetch       ; Indicate fetch operation
              000B  27FE                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                00  2800                .BYTE   0               ; 0 HP$B_DRIVE to data
                08  2801                .BYTE   8               ; 8 of data field
```

```
                      88  2802                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2803                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                      07  2805                .BYTE   7                ; 7 HP$A_DEVICE to data
                      03  2806                .BYTE   3                ; 3 of data field
                      81  2807                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2808    784 RP04:   $DS_RP04
                          2808                .SAVE   LOCAL_BLOCK
                          2808                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RP04_LEN,       HP$K_RP04_LEN:
                          0032                .IIF    NB,RP04$K_LEN,  RP04$K_LEN:
                 00002808                     .RESTORE
              34 30 50 52 00' 2808            .ASCIC  "RP04"  ; RP04 name
                      04  2808
                      32  280D                .BYTE   RP04$K_LEN               ; RP04$K_LEN of resultant P-table
                      08  280E                .BYTE   8                ; Maximum unit number                  ;G:65
                    4244  280F                .WORD   ^A"DB"  ; Two character mnemonic for QIO RP04 DB
                      80  2811                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  2812                .Byte   Pd$_Name         ; Directive op-code
                      03  2813                .Byte   Ptd$M_Device             ; Enforced DB codes
                 42 44 00' 2814               .Ascic  "DB"    ; Enforced device name prefix
                      02  2814
                      86  2817                .BYTE   Pd$_Literal      ; Indicate literal
                 00000001 2818                .LONG   HP$M_ALLOC               ; Constant
                      88  281C                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  281D                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  281F                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  2820                .BYTE   8                ; 8 of data field
                      87  2821                .BYTE   Pd$_Fetch        ; Indicate fetch operation
                    000B  2822                .WORD   HP$B_DRIVE               ; Byte HP$B_DRIVE to data
                      00  2824                .BYTE   0                ; 0 HP$B_DRIVE to data
                      08  2825                .BYTE   8                ; 8 of data field
                      88  2826                .BYTE   Pd$_Store        ; Indicate store operation
                    0018  2827                .WORD   HP$A_DEVICE              ; Byte HP$A_DEVICE to data
                      07  2829                .BYTE   7                ; 7 HP$A_DEVICE to data
                      03  282A                .BYTE   3                ; 3 of data field
                      81  282B                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          282C    785 RP05:   $DS_RP05                                               ;G:43
                          282C                .SAVE   LOCAL_BLOCK
                          282C                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RP05_LEN,       HP$K_RP05_LEN:
                          0032                .IIF    NB,RP05$K_LEN,  RP05$K_LEN:
                 0000282C                     .RESTORE
              35 30 50 52 00' 282C            .ASCIC  "RP05"  ; RP05 name
                      04  282C
                      32  2831                .BYTE   RP05$K_LEN               ; RP05$K_LEN of resultant P-table
                      08  2832                .BYTE   8                ; Maximum unit number                  ;G:65
                    4244  2833                .WORD   ^A"DB"  ; Two character mnemonic for QIO RP05 DB
                      80  2835                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  2836                .Byte   Pd$_Name         ; Directive op-code
                      03  2837                .Byte   Ptd$M_Device             ; Enforced DB codes
                 42 44 00' 2838               .Ascic  "DB"    ; Enforced device name prefix
                      02  2838
                      86  283B                .BYTE   Pd$_Literal      ; Indicate literal
                 00000001 283C                .LONG   HP$M_ALLOC               ; Constant
                      88  2840                .BYTE   Pd$_Store        ; Indicate store operation
```

```
                     000A  2841                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                       00  2843                .BYTE   0                   ; 0 HP$B_FLAGS to data
                       08  2844                .BYTE   8                   ; 8 of data field
                       87  2845                .BYTE   Pd$_Fetch           ; Indicate fetch operation
                     000B  2846                .WORD   HP$B_DRIVE                  ; Byte HP$B_DRIVE to data
                       00  2848                .BYTE   0                   ; 0 HP$B_DRIVE to data
                       08  2849                .BYTE   8                   ; 8 of data field
                       88  284A                .BYTE   Pd$_Store           ; Indicate store operation
                     0018  284B                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                       07  284D                .BYTE   7                   ; 7 HP$A_DEVICE to data
                       03  284E                .BYTE   3                   ; 3 of data field
                       81  284F                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           2850    786 RP06:   $DS_RP06
                           2850                .SAVE   LOCAL_BLOCK
                           2850                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_RP06_LEN,       HP$K_RP06_LEN:
                           0032                .IIF    NB,RP06$K_LEN,  RP06$K_LEN:
                       0000285O                .RESTORE
              36 30 50 52 00' 2850             .ASCIC  "RP06"  ; RP06 name
                       04  2850
                       32  2855                .BYTE   RP06$K_LEN              ; RP06$K_LEN of resultant P-table
                       08  2856                .BYTE   8                   ; Maximum unit number                   ;G:65
                     4244  2857                .WORD   ^A"DB"  ; Two character mnemonic for QIO RP06 DB
                       80  2859                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  285A                .Byte   Pd$_Name            ; Directive op-code
                       03  285B                .Byte   Ptd$M_Device                ; Enforced DB codes
                 42 44 00' 285C                .Ascic  "DB"    ; Enforced device name prefix
                       02  285C
                       86  285F                .BYTE   Pd$_Literal         ; Indicate literal
                 00000001  2860                .LONG   HP$M_ALLOC                  ; Constant
                       88  2864                .BYTE   Pd$_Store           ; Indicate store operation
                     000A  2865                .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                       00  2867                .BYTE   0                   ; 0 HP$B_FLAGS to data
                       08  2868                .BYTE   8                   ; 8 of data field
                       87  2869                .BYTE   Pd$_Fetch           ; Indicate fetch operation
                     000B  286A                .WORD   HP$B_DRIVE                  ; Byte HP$B_DRIVE to data
                       00  286C                .BYTE   0                   ; 0 HP$B_DRIVE to data
                       08  286D                .BYTE   8                   ; 8 of data field
                       88  286E                .BYTE   Pd$_Store           ; Indicate store operation
                     0018  286F                .WORD   HP$A_DEVICE                 ; Byte HP$A_DEVICE to data
                       07  2871                .BYTE   7                   ; 7 HP$A_DEVICE to data
                       03  2872                .BYTE   3                   ; 3 of data field
                       81  2873                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           2874    787 RP07:   $DS_RP07
                           2874                .SAVE   LOCAL_BLOCK
                           2874                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_RP07_LEN,       HP$K_RP07_LEN:
                           0032                .IIF    NB,RP07$K_LEN,  RP07$K_LEN:
                       0000287O                .RESTORE
              37 30 50 52 00' 2874             .ASCIC  "RP07"  ; RP07 name
                       04  2874
                       32  2879                .BYTE   RP07$K_LEN              ; RP07$K_LEN of resultant P-table
                       08  287A                .BYTE   8                   ; Maximum unit number                   ;G:65
                     5244  287B                .WORD   ^A"DR"  ; Two character mnemonic for QIO RP07 DR
                       80  287D                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
```

```
                      8D  287E                .Byte   Pd$_Name        ; Directive op-code
                      03  287F                .Byte   Ptd$M_Device            ; Enforced DR codes
                52 44 00' 2880                .Ascic  "DR"    ; Enforced device name prefix
                      02  2880
                      86  2883                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  2884                .LONG   HP$M_ALLOC              ; Constant
                      88  2888                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  2889                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  288B                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  288C                .BYTE   8               ; 8 of data field
                      87  288D                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    000B  288E                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                      00  2890                .BYTE   0               ; 0 HP$B_DRIVE to data
                      08  2891                .BYTE   8               ; 8 of data field
                      88  2892                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2893                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      07  2895                .BYTE   7               ; 7 HP$A_DEVICE to data
                      03  2896                .BYTE   3               ; 3 of data field
                      81  2897                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2898  788 RRD50:     $DS_RRD50                       ;       [73]            ;G:4
                          2898                .SAVE   LOCAL_BLOCK
                          2898                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RRD50$L_CSR, RRD50$L_CSR::
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,RRD50$B_BR,  RRD50$B_BR::
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,RRD50$K_LEN, RRD50$K_LEN::
                      00002898                .RESTORE
          30 35 44 52 52 00' 2898             .ASCIC  "RRD50" ; RRD50 name
                      05  2898
                      37  289E                .BYTE   RRD50$K_LEN             ; RRD50$K_LEN of resultant P-table
                      04  289F                .BYTE   4               ; Maximum unit number           ;G:65
                    0000  28A0                .WORD   ^A""    ; Two character mnemonic for QIO RRD50
                      80  28A2                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  28A3                .Byte   Pd$_Name        ; Directive op-code
                      03  28A4                .Byte   PTD$M_DEVICE            ; Enforced DU codes
                55 44 00' 28A5                .Ascic  "DU"    ; Enforced device name prefix
                      02  28A5
                      86  28A8                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  28A9                .LONG   HP$M_ALLOC              ; Constant
                      88  28AD                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  28AE                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  28B0                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  28B1                .BYTE   8               ; 8 of data field
                      83  28B2                .BYTE   Pd$_Octal       ; Indicate scan octal
    53 53 45 52 44 44 41 00' 28B3             .ASCIC  "ADDRESS"       ; ADDRESS string
                      07  28B3
          0003FFFE 0003E008  28BB             .LONG   ^O760010,^O777776       ; 760010 , 777776 limits on input
                      88  28C3                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  28C4                .WORD   RRD50$L_CSR             ; Byte RRD50$L_CSR to data
                      00  28C6                .BYTE   0               ; 0 RRD50$L_CSR to data
                      20  28C7                .BYTE   32              ; 32 of data field
                      88  28C8                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  28C9                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  28CB                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  28CC                .BYTE   18              ; 18 of data field
```

```
                      83  28CD                .BYTE   Pd$_Octal       ; Indicate scan octal
          52 4F 54 43 45 56 00' 28CE           .ASCIC  "VECTOR"        ; VECTOR string
                      06  28CE
       000001FC 00000040  28D5                .LONG   ^O100,^O774     ; 100 , 774 limits on input
                      88  28DD                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  28DE                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  28E0                .BYTE   0               ; 0 HP$W_VECTOR to data
                      10  28E1                .BYTE   16              ; 16 of data field
                      82  28E2                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 28E3                .ASCIC  "BR"    ; BR string
                      02  28E3
       00000007 00000004  28E6                .LONG   4,7     ; 4 , 7 limits on input
                      88  28EE                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  28EF                .WORD   RRD50$B_BR              ; Byte RRD50$B_BR to data
                      00  28F1                .BYTE   0               ; 0 RRD50$B_BR to data
                      08  28F2                .BYTE   8               ; 8 of data field
                      81  28F3                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          28F4    789 RV20:    $DS_RV20                                                ;G:42
                          28F4                .SAVE   LOCAL_BLOCK
                          28F4                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RV20$L_CSR,  RV20$L_CSR:
                 00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,RV20$B_BR,   RV20$B_BR:
                 00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,RV20$K_LEN,  RV20$K_LEN:
                      28F4                .RESTORE
             30 32 56 52 00' 28F4             .ASCIC  "RV20"  ; RV20 name
                      04  28F4
                      37  28F9                .BYTE   RV20$K_LEN              ; RV20$K_LEN of resultant P-table
                      04  28FA                .BYTE   4               ; Maximum unit number                   ;G:65
                    0000  28FB                .WORD   ^A""    ; Two character mnemonic for QIO RV20
                      80  28FD                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  28FE                .Byte   Pd$_Name        ; Directive op-code
                      03  28FF                .Byte   PTD$M_DEVICE            ; Enforced MU codes
                 55 4D 00' 2900                .Ascic  "MU"    ; Enforced device name prefix
                      02  2900
                      86  2903                .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  2904                .LONG   HP$M_ALLOC              ; Constant
                      88  2908                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  2909                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  290B                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  290C                .BYTE   8               ; 8 of data field
                      83  290D                .BYTE   Pd$_Octal       ; Indicate scan octal
    53 53 45 52 44 44 41 00' 290E             .ASCIC  "ADDRESS"       ; ADDRESS string
                      07  290E
       0003FFFE 0003E008  2916                .LONG   ^O760010,^O777776       ; 760010 , 777776 limits on input
                      88  291E                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  291F                .WORD   RV20$L_CSR              ; Byte RV20$L_CSR to data
                      00  2921                .BYTE   0               ; 0 RV20$L_CSR to data
                      20  2922                .BYTE   32              ; 32 of data field
                      88  2923                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2924                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  2926                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  2927                .BYTE   18              ; 18 of data field
                      83  2928                .BYTE   Pd$_Octal       ; Indicate scan octal
          52 4F 54 43 45 56 00' 2929           .ASCIC  "VECTOR"        ; VECTOR string
```

```
                    06  2929
000001FC 00000040   2930                .LONG   ^O100,^O774     ; 100 , 774 limits on input
                    88  2938                .BYTE   Pd$_Store       ; Indicate store operation
                  0024  2939                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  293B                .BYTE   0               ; 0 HP$W_VECTOR to data
                    10  293C                .BYTE   16              ; 16 of data field
                    82  293D                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 293E              .ASCIC  "BR"    ; BR string
                    02  293E
00000007 00000004   2941                .LONG   4,7     ; 4 , 7 limits on input
                    88  2949                .BYTE   Pd$_Store       ; Indicate store operation
                  0036  294A                .WORD   RV20$B_BR               ; Byte RV20$B_BR to data
                    00  294C                .BYTE   0               ; 0 RV20$B_BR to data
                    08  294D                .BYTE   8               ; 8 of data field
                    81  294E                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        294F    790 RUX50:  $DS_RUX50                               ;       [53]
                        294F                .SAVE   LOCAL_BLOCK
                        294F                .PSECT  $ABS$,ABS
                00000032  0050              .=HP$A_DEPENDENT
                          0032              .IIF    NB,HP$B_RUX50_BR,       HP$B_RUX50_BR:
                00000033  0032              .IIF    NB,.BLKB,       .BLKB   1
                          0033              .IIF    NB,HP$K_RUX50_LEN,      HP$K_RUX50_LEN:
                      0000294F              .RESTORE
             30 35 58 55 52 00' 294F        .ASCIC  "RUX50" ; RUX50 name
                    05  294F
                    33  2955                .BYTE   HP$K_RUX50_LEN          ; HP$K_RUX50_LEN of resultant P-table
                    04  2956                .BYTE   4               ; Maximum unit number                           ;G:65
                  0000  2957                .WORD   ^A""    ; Two character mnemonic for QIO RUX50
                    80  2959                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  295A                .Byte   Pd$_Name        ; Directive op-code
                    02  295B                .Byte   PTD$M_CONTROLLER                ; Enforced DU codes
                55 44 00' 295C              .Ascic  "DU"    ; Enforced device name prefix
                    02  295C
                    83  295F                .BYTE   Pd$_Octal       ; Indicate scan octal
                50 49 00' 2960              .ASCIC  "IP"    ; IP string
                    02  2960
0003FFFC 0003E000   2963                .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                    88  296B                .BYTE   Pd$_Store       ; Indicate store operation
                  0018  296C                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  296E                .BYTE   0               ; 0 HP$A_DEVICE to data
                    12  296F                .BYTE   18              ; 18 of data field
                    83  2970                .BYTE   Pd$_Octal       ; Indicate scan octal
52 4F 54 43 45 56 00' 2971                  .ASCIC  "VECTOR"        ; VECTOR string
                    06  2971
000001FC 00000004   2978                .LONG   ^O4,^O774       ; 4 , 774 limits on input
                    88  2980                .BYTE   Pd$_Store       ; Indicate store operation
                  0024  2981                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  2983                .BYTE   0               ; 0 HP$W_VECTOR to data
                    09  2984                .BYTE   9               ; 9 of data field
                    82  2985                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 2986              .ASCIC  "BR"    ; BR string
                    02  2986
00000007 00000004   2989                .LONG   4,7     ; 4 , 7 limits on input
                    88  2991                .BYTE   Pd$_Store       ; Indicate store operation
                  0032  2992                .WORD   HP$B_RUX50_BR           ; Byte HP$B_RUX50_BR to data
                    00  2994                .BYTE   0               ; 0 HP$B_RUX50_BR to data
                    08  2995                .BYTE   8               ; 8 of data field
```

```
                      81  2996                        .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2997    791 RX02:   $DS_RX02
                          2997                .SAVE   LOCAL_BLOCK
                          2997                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_RX02_LEN,       HP$K_RX02_LEN:
                          0032                .IIF    NB,RX02$K_LEN,  RX02$K_LEN:
                      0000299700              .RESTORE
              32 30 58 52 00' 2997            .ASCIC  "RX02"  ; RX02 name
                      04  2997
                      32  299C                .BYTE   RX02$K_LEN              ; RX02$K_LEN of resultant P-table
                      02  299D                .BYTE   2                       ; Maximum unit number                   ;G:65
                    0000  299E                .WORD   ^A""    ; Two character mnemonic for QIO RX02
                      80  29A0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  29A1                .Byte   Pd$_Name        ; Directive op-code
                      1B  29A2                .Byte   Ptd$M_EndDevice         ; Enforced DY codes
                  59 44 00' 29A3              .Ascic  "DY"    ; Enforced device name prefix
                      02  29A3
                      86  29A6                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  29A7                .LONG   HP$M_ALLOC              ; Constant
                      88  29AB                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  29AC                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  29AE                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  29AF                .BYTE   8                       ; 8 of data field
                      81  29B0                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          29B1    792 RX31:   $DS_RX31                                ;       [53]
                          29B1                .SAVE   LOCAL_BLOCK
                          29B1                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RX31$K_LEN,  RX31$K_LEN:
                      000029B100              .RESTORE
              31 33 58 52 00' 29B1            .ASCIC  "RX31"  ; RX31 name
                      04  29B1
                      32  29B6                .BYTE   RX31$K_LEN              ; RX31$K_LEN of resultant P-table
                      FF  29B7                .BYTE   255                     ; Maximum unit number                   ;G:65
                    0000  29B8                .WORD   ^A""    ; Two character mnemonic for QIO RX31
                      80  29BA                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  29BB                .Byte   Pd$_Name        ; Directive op-code
                      1B  29BC                .Byte   Ptd$M_EndDevice         ; Enforced DU codes
                  55 44 00' 29BD              .Ascic  "DU"    ; Enforced device name prefix
                      02  29BD
                      81  29C0                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          29C1    793 RX32:   $DS_RX32                                ;       [53]
                          29C1                .SAVE   LOCAL_BLOCK
                          29C1                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,RX32$K_LEN,  RX32$K_LEN:
                      000029C100              .RESTORE
              32 33 58 52 00' 29C1            .ASCIC  "RX32"  ; RX32 name
                      04  29C1
                      32  29C6                .BYTE   RX32$K_LEN              ; RX32$K_LEN of resultant P-table
                      FF  29C7                .BYTE   255                     ; Maximum unit number                   ;G:65
                    0000  29C8                .WORD   ^A""    ; Two character mnemonic for QIO RX32
                      80  29CA                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  29CB                .Byte   Pd$_Name        ; Directive op-code
                      1B  29CC                .Byte   Ptd$M_EndDevice         ; Enforced DU codes
                  55 44 00' 29CD              .Ascic  "DU"    ; Enforced device name prefix
```

```
                      02  29CD
                      81  29D0                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          29D1    794 RX50:   $DS_RX50                                ;       [53]
                          29D1                .SAVE   LOCAL_BLOCK
                          29D1                .PSECT  $ABS$,ABS
                 00000032  0050               .=HP$A_DEPENDENT
                           0032               .IIF    NB,RX50$K_LEN,  RX50$K_LEN:
                      000029D1                .RESTORE
              30 35 58 52 00' 29D1            .ASCIC  "RX50"  ; RX50 name
                      04  29D1
                      32  29D6                .BYTE   RX50$K_LEN              ; RX50$K_LEN of resultant P-table
                      FF  29D7                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  29D8                .WORD   ^A""    ; Two character mnemonic for QIO RX50
                      80  29DA                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      86  29DB                .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  29DC                .LONG   HP$M_ALLOC              ; Constant
                      88  29E0                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  29E1                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  29E3                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  29E4                .BYTE   8               ; 8 of data field
                      81  29E5                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          29E6    795 RX180:  $DS_RX180                               ;       [53]
                          29E6                .SAVE   LOCAL_BLOCK
                          29E6                .PSECT  $ABS$,ABS
                 00000032  0050               .=HP$A_DEPENDENT
                           0032               .IIF    NB,RX180$K_LEN, RX180$K_LEN:
                      000029E6                .RESTORE
              30 38 31 58 52 00' 29E6         .ASCIC  "RX180" ; RX180 name
                      05  29E6
                      32  29EC                .BYTE   RX180$K_LEN             ; RX180$K_LEN of resultant P-table
                      FF  29ED                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  29EE                .WORD   ^A""    ; Two character mnemonic for QIO RX180
                      80  29F0                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  29F1                .Byte   Pd$_Name        ; Directive op-code
                      1B  29F2                .Byte   Ptd$M_EndDevice         ; Enforced DU codes
                 55 44 00' 29F3               .Ascic  "DU"    ; Enforced device name prefix
                      02  29F3
                      81  29F6                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          29F7    796 RX211:  $DS_RX211
                          29F7                .SAVE   LOCAL_BLOCK
                          29F7                .PSECT  $ABS$,ABS
                 00000032  0050               .=HP$A_DEPENDENT
                           0032               .IIF    NB,HP$L_RX211_CSR,      HP$L_RX211_CSR:
                           0032               .IIF    NB,RX211$L_CSR, RX211$L_CSR:
                 00000036  0032               .IIF    NB,.BLKL,       .BLKL   1
                           0036               .IIF    NB,HP$B_RX211_BR,       HP$B_RX211_BR:
                           0036               .IIF    NB,RX211$B_BR,  RX211$B_BR:
                 00000037  0036               .IIF    NB,.BLKB,       .BLKB   1
                           0037               .IIF    NB,HP$K_RX211_LEN,      HP$K_RX211_LEN:
                           0037               .IIF    NB,RX211$K_LEN, RX211$K_LEN:
                      000029F7                .RESTORE
              31 31 32 58 52 00' 29F7         .ASCIC  "RX211" ; RX211 name
                      05  29F7
                      37  29FD                .BYTE   RX211$K_LEN             ; RX211$K_LEN of resultant P-table
                      00  29FE                .BYTE   0               ; Maximum unit number                   ;G:65
                    5944  29FF                .WORD   ^A"DY"  ; Two character mnemonic for QIO RX211 DY
                      80  2A01                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
```

```
                      8D  2A02                .Byte   Pd$_Name        ; Directive op-code
                      02  2A03                .Byte   Ptd$M_Controller                ; Enforced DY codes
                59 44 00' 2A04                .Ascic  "DY"    ; Enforced device name prefix
                      02  2A04
                      83  2A07                .BYTE   Pd$_Octal       ; Indicate scan octal
              52 53 43 00' 2A08                .ASCIC  "CSR"   ; CSR string
                      03  2A08
          0003FFFE 0003E000  2A0C                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  2A14                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  2A15                .WORD   RX211$L_CSR             ; Byte RX211$L_CSR to data
                      00  2A17                .BYTE   0               ; 0 RX211$L_CSR to data
                      20  2A18                .BYTE   32              ; 32 of data field
                      88  2A19                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2A1A                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  2A1C                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  2A1D                .BYTE   18              ; 18 of data field
                      83  2A1E                .BYTE   Pd$_Octal       ; Indicate scan octal
        52 4F 54 43 45 56 00' 2A1F            .ASCIC  "VECTOR"        ; VECTOR string
                      06  2A1F
          000001FE 00000002  2A26                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  2A2E                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  2A2F                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  2A31                .BYTE   0               ; 0 HP$W_VECTOR to data
                      09  2A32                .BYTE   9               ; 9 of data field
                      82  2A33                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 2A34                .ASCIC  "BR"    ; BR string
                      02  2A34
          00000007 00000004  2A37                .LONG   4,7     ; 4 , 7 limits on input
                      88  2A3F                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  2A40                .WORD   RX211$B_BR              ; Byte RX211$B_BR to data
                      00  2A42                .BYTE   0               ; 0 RX211$B_BR to data
                      08  2A43                .BYTE   8               ; 8 of data field
                      81  2A44                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2A45    797 SBIA:   $DS_SBIA                                ;       [37]
                          2A45                .SAVE   LOCAL_BLOCK
                          2A45                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_SBIA_LEN,       HP$K_SBIA_LEN:
                          0032                .IIF    NB,SBIA$K_LEN,  SBIA$K_LEN:
                      0000 2A45                .RESTORE
           41 49 42 53 00' 2A45                .ASCIC  "SBIA"  ; SBIA name
                      04  2A45
                      32  2A4A                .BYTE   SBIA$K_LEN              ; SBIA$K_LEN of resultant P-table
                      02  2A4B                .BYTE   2               ; Maximum unit number                   ;G:65
                    0000  2A4C                .WORD   ^A""    ; Two character mnemonic for QIO SBIA
                      80  2A4E                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2A4F                .Byte   Pd$_Name        ; Directive op-code
                      01  2A50                .Byte   Ptd$M_Unit              ; Enforced SI codes
                49 53 00' 2A51                .Ascic  "SI"    ; Enforced device name prefix
                      02  2A51
                      86  2A54                .BYTE   Pd$_Literal     ; Indicate literal
                60000000  2A55                .LONG   ^X60000000              ; Constant
                      88  2A59                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  2A5A                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      00  2A5C                .BYTE   0               ; 0 HP$A_DVA to data
                      20  2A5D                .BYTE   32              ; 32 of data field
                      86  2A5E                .BYTE   Pd$_Literal     ; Indicate literal
```

```
                60080000  2A5F                .LONG   ^X60080000              ; Constant
                      88  2A63                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2A64                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  2A66                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  2A67                .BYTE   32              ; 32 of data field
                      86  2A68                .BYTE   Pd$_Literal      ; Indicate literal
                00000000  2A69                .LONG   ^X000           ; Constant
                      88  2A6D                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  2A6E                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  2A70                .BYTE   0               ; 0 HP$W_VECTOR to data
                      10  2A71                .BYTE   16              ; 16 of data field
                      87  2A72                .BYTE   Pd$_Fetch       ; Indicate fetch operation
                    000B  2A73                .WORD   HP$B_DRIVE              ; Byte HP$B_DRIVE to data
                      00  2A75                .BYTE   0               ; 0 HP$B_DRIVE to data
                      08  2A76                .BYTE   8               ; 8 of data field
                      88  2A77                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2A78                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      19  2A7A                .BYTE   25              ; 25 HP$A_DEVICE to data
                      03  2A7B                .BYTE   3               ; 3 of data field
                      88  2A7C                .BYTE   Pd$_Store       ; Indicate store operation
                    001C  2A7D                .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
                      19  2A7F                .BYTE   25              ; 25 HP$A_DVA to data
                      03  2A80                .BYTE   3               ; 3 of data field
                      8A  2A81                .BYTE   Pd$_Add         ; Indicate add value to field operation
                    0024  2A82                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      09  2A84                .BYTE   9               ; 9 HP$W_VECTOR to data
                      07  2A85                .BYTE   7               ; 7 of data field
                      81  2A86                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2A87    798 SLU:    $DS_SLU                                 ;       [62]
                          2A87                .SAVE   LOCAL_BLOCK
                          2A87                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,SLU$B_EXT,   SLU$B_EXT::
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,SLU$B_ACTIV, SLU$B_ACTIV::
                00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,SLU$B_SPEED, SLU$B_SPEED::
                00000035  0034                .IIF    NB,.BLKB,       .BLKB   1
                          0035                .IIF    NB,SLU$K_LEN,   SLU$K_LEN::
                      00002A87                .RESTORE
             55 4C 53 00' 2A87                .ASCIC  "SLU"   ; SLU name
                      03  2A87
                      35  2A8B                .BYTE   SLU$K_LEN               ; SLU$K_LEN of resultant P-table
                      01  2A8C                .BYTE   1               ; Maximum unit number                   ;G:65
                    0000  2A8D                .WORD   ^A""    ; Two character mnemonic for QIO SLU
                      80  2A8F                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2A90                .Byte   Pd$_Name        ; Directive op-code
                      03  2A91                .Byte   Ptd$M_Device            ; Enforced TC codes
             43 54 00' 2A92                .Ascic  "TC"    ; Enforced device name prefix
                      02  2A92
                      8B  2A95                .BYTE   Pd$_Logical     ; Indicate logical value operation
65 6E 69 4C 20 6C 61 69 72 65 53 00' 2A96    .ASCIC  \Serial Line Externally Wrapped\        ; Serial Line Externally Wra
20 79 6C 6C 61 6E 72 65 74 78 45 20  2AA2
            64 65 70 70 61 72 57     2AAE
                      1E  2A96
                      88  2AB5                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  2AB6                .WORD   SLU$B_EXT               ; Byte SLU$B_EXT to data
```

```
                                00  2AB8                .BYTE   0               ; 0 SLU$B_EXT to data
                                08  2AB9                .BYTE   8               ; 8 of data field
                                86  2ABA                .BYTE   Pd$_Literal     ; Indicate literal
                          00000001  2ABB                .LONG   1               ; Constant
                                88  2ABF                .BYTE   Pd$_Store       ; Indicate store operation
                              0033  2AC0                .WORD   SLU$B_ACTIV             ; Byte SLU$B_ACTIV to data
                                00  2AC2                .BYTE   0               ; 0 SLU$B_ACTIV to data
                                08  2AC3                .BYTE   8               ; 8 of data field
                                85  2AC4                .BYTE   Pd$_String      ; Indicate scan and verify string
          65 74 61 52 20 64 75 61 42 00' 2AC5           .ASCIC  "Baud Rate"     ; Baud Rate string
                                09  2AC5
                   30 35 31 00' 2ACF                    .ASCIC  "150"
                                03  2ACF
                   30 30 33 00' 2AD3                    .ASCIC  "300"
                                03  2AD3
                   30 30 36 00' 2AD7                    .ASCIC  "600"
                                03  2AD7
                30 30 32 31 00' 2ADB                    .ASCIC  "1200"
                                04  2ADB
                30 30 34 32 00' 2AE0                    .ASCIC  "2400"
                                04  2AE0
                30 30 38 34 00' 2AE5                    .ASCIC  "4800"
                                04  2AE5
                30 30 36 39 00' 2AEA                    .ASCIC  "9600"
                                04  2AEA
             30 30 32 39 31 00' 2AEF                    .ASCIC  "19200"
                                05  2AEF
                                00  2AF5                .BYTE   0               ; Null length is end of valid 150,300,600,1200,2400,
                                88  2AF6                .BYTE   Pd$_Store       ; Indicate store operation
                              0034  2AF7                .WORD   SLU$B_SPEED             ; Byte SLU$B_SPEED to data
                                00  2AF9                .BYTE   0               ; 0 SLU$B_SPEED to data
                                08  2AFA                .BYTE   8               ; 8 of data field
                                81  2AFB                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    2AFC    799 TAPE:   $DS_TAPE                        ;       [42]
                                    2AFC                .SAVE   LOCAL_BLOCK
                                    2AFC                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,HP$K_TAPE_LEN,       HP$K_TAPE_LEN:
                                    0032                .IIF    NB,TAPE$K_LEN,  TAPE$K_LEN:
                                00002AFC                .RESTORE
                       45 50 41 54 00' 2AFC             .ASCIC  "TAPE"  ; TAPE name
                                04  2AFC
                                32  2B01                .BYTE   TAPE$K_LEN              ; TAPE$K_LEN of resultant P-table
                                FF  2B02                .BYTE   255             ; Maximum unit number                   ;G:65
                              0000  2B03                .WORD   ^A""    ; Two character mnemonic for QIO TAPE
                                80  2B05                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                                8D  2B06                .Byte   Pd$_Name        ; Directive op-code
                                03  2B07                .Byte   Ptd$M_DEVICE            ; Enforced MU codes
                          55 4D 00' 2B08                .Ascic  "MU"    ; Enforced device name prefix
                                02  2B08
                                81  2B0B                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                                    2B0C    800 TE16:   $DS_TE16
                                    2B0C                .SAVE   LOCAL_BLOCK
                                    2B0C                .PSECT  $ABS$,ABS
                          00000032  0050                .=HP$A_DEPENDENT
                                    0032                .IIF    NB,HP$K_TE16_LEN,       HP$K_TE16_LEN:
                                    0032                .IIF    NB,TE16$K_LEN,  TE16$K_LEN:
```

```
                00002B0C                        .RESTORE
      36 31 45 54 00' 2B0C                      .ASCIC  "TE16"  ; TE16 name
                04  2B0C
                32  2B11                       .BYTE   TE16$K_LEN                  ; TE16$K_LEN of resultant P-table
                08  2B12                       .BYTE   8                   ; Maximum unit number                       ;G:65
              0000  2B13                       .WORD   ^A""    ; Two character mnemonic for QIO TE16
                80  2B15                       .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                8D  2B16                       .Byte   Pd$_Name            ; Directive op-code
                1B  2B17                       .Byte   Ptd$M_EndDevice             ; Enforced MT codes
            54 4D 00' 2B18                     .Ascic  "MT"    ; Enforced device name prefix
                02  2B18
                86  2B1B                       .BYTE   Pd$_Literal         ; Indicate literal
          00000001  2B1C                       .LONG   HP$M_ALLOC                  ; Constant
                88  2B20                       .BYTE   Pd$_Store           ; Indicate store operation
              000A  2B21                       .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                00  2B23                       .BYTE   0                   ; 0 HP$B_FLAGS to data
                08  2B24                       .BYTE   8                   ; 8 of data field
                81  2B25                       .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                    2B26    801 TK50:           $DS_TK50                                ;       [50]
                    2B26                       .SAVE   LOCAL_BLOCK
                    2B26                       .PSECT  $ABS$,ABS
          00000032  0050                       .=HP$A_DEPENDENT
                    0032                       .IIF    NB,HP$K_TK50_LEN,       HP$K_TK50_LEN:
                00002B26                       .RESTORE
      30 35 4B 54 00' 2B26                     .ASCIC  "TK50"  ; TK50 name
                04  2B26
                32  2B2B                       .BYTE   HP$K_TK50_LEN               ; HP$K_TK50_LEN of resultant P-table
                10  2B2C                       .BYTE   16                  ; Maximum unit number                       ;G:65
              554D  2B2D                       .WORD   ^A"MU"  ; Two character mnemonic for QIO TK50 MU
                80  2B2F                       .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                8D  2B30                       .Byte   Pd$_Name            ; Directive op-code
                03  2B31                       .Byte   Ptd$M_Device                ; Enforced MU codes
            55 4D 00' 2B32                     .Ascic  "MU"    ; Enforced device name prefix
                02  2B32
                86  2B35                       .BYTE   Pd$_Literal         ; Indicate literal
          00000001  2B36                       .LONG   HP$M_ALLOC                  ; Constant
                88  2B3A                       .BYTE   Pd$_Store           ; Indicate store operation
              000A  2B3B                       .WORD   HP$B_FLAGS                  ; Byte HP$B_FLAGS to data
                00  2B3D                       .BYTE   0                   ; 0 HP$B_FLAGS to data
                08  2B3E                       .BYTE   8                   ; 8 of data field
                81  2B3F                       .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                    2B40    802 TK70:           $DS_TK70                                ;       [84]                ;G:39
                    2B40                       .SAVE   LOCAL_BLOCK
                    2B40                       .PSECT  $ABS$,ABS
          00000032  0050                       .=HP$A_DEPENDENT
                    0032                       .IIF    NB,HP$K_TK70_LEN,       HP$K_TK70_LEN:
                00002B40                       .RESTORE
      30 37 4B 54 00' 2B40                     .ASCIC  "TK70"  ; TK70 name
                04  2B40
                32  2B45                       .BYTE   HP$K_TK70_LEN               ; HP$K_TK70_LEN of resultant P-table
                10  2B46                       .BYTE   16                  ; Maximum unit number                       ;G:65
              554D  2B47                       .WORD   ^A"MU"  ; Two character mnemonic for QIO TK70 MU
                80  2B49                       .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                8D  2B4A                       .Byte   Pd$_Name            ; Directive op-code
                03  2B4B                       .Byte   Ptd$M_Device                ; Enforced MU codes
            55 4D 00' 2B4C                     .Ascic  "MU"    ; Enforced device name prefix
                02  2B4C
```

```
                      86  2B4F                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  2B50                .LONG   HP$M_ALLOC                ; Constant
                      88  2B54                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2B55                .WORD   HP$B_FLAGS                ; Byte HP$B_FLAGS to data
                      00  2B57                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  2B58                .BYTE   8                ; 8 of data field
                      81  2B59                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2B5A    803 TM03:     $DS_TM03
                          2B5A                .SAVE   LOCAL_BLOCK
                          2B5A                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_TM03_DRIVE,     HP$B_TM03_DRIVE:
                          0032                .IIF    NB,TM03$B_DRIVE,        TM03$B_DRIVE:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_TM03_LEN,       HP$K_TM03_LEN:
                          0033                .IIF    NB,TM03$K_LEN,  TM03$K_LEN:
                      0002B5A                 .RESTORE
           33 30 4D 54 00' 2B5A                .ASCIC  "TM03"  ; TM03 name
                      04  2B5A
                      33  2B5F                .BYTE   TM03$K_LEN               ; TM03$K_LEN of resultant P-table
                      00  2B60                .BYTE   0                ; Maximum unit number                          ;G:65
                    4D54  2B61                .WORD   ^A"TM"  ; Two character mnemonic for QIO TM03 TM
                      80  2B63                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  2B64                .Byte   Pd$_Name         ; Directive op-code
                      02  2B65                .Byte   Ptd$M_Controller                 ; Enforced MT codes
                 54 4D 00' 2B66                .Ascic  "MT"    ; Enforced device name prefix
                      02  2B66
                      82  2B69                .BYTE   Pd$_Decimal      ; Indicate scan decimal
        45 56 49 52 44 00' 2B6A                .ASCIC  "DRIVE" ; DRIVE string
                      05  2B6A
        00000007 00000000  2B70                .LONG   0,7     ; 0 , 7 limits on input
                      88  2B78                .BYTE   Pd$_Store        ; Indicate store operation
                    0032  2B79                .WORD   TM03$B_DRIVE              ; Byte TM03$B_DRIVE to data
                      00  2B7B                .BYTE   0                ; 0 TM03$B_DRIVE to data
                      08  2B7C                .BYTE   8                ; 8 of data field
                      88  2B7D                .BYTE   Pd$_Store        ; Indicate store operation
                    000B  2B7E                .WORD   HP$B_DRIVE                ; Byte HP$B_DRIVE to data
                      00  2B80                .BYTE   0                ; 0 HP$B_DRIVE to data
                      08  2B81                .BYTE   8                ; 8 of data field
                      88  2B82                .BYTE   Pd$_Store        ; Indicate store operation
                    0018  2B83                .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                      07  2B85                .BYTE   7                ; 7 HP$A_DEVICE to data
                      03  2B86                .BYTE   3                ; 3 of data field
                      81  2B87                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2B88    804 TM78:     $DS_TM78
                          2B88                .SAVE   LOCAL_BLOCK
                          2B88                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_TM78_DRIVE,     HP$B_TM78_DRIVE:
                          0032                .IIF    NB,TM78$B_DRIVE,        TM78$B_DRIVE:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_TM78_LEN,       HP$K_TM78_LEN:
                          0033                .IIF    NB,TM78$K_LEN,  TM78$K_LEN:
                      0002B88                 .RESTORE
           38 37 4D 54 00' 2B88                .ASCIC  "TM78"  ; TM78 name
                      04  2B88
                      33  2B8D                .BYTE   TM78$K_LEN               ; TM78$K_LEN of resultant P-table
```

```
                      00  2B8E                .BYTE   0                       ; Maximum unit number                                   ;G:65
                    4654  2B8F                .WORD   ^A"TF"  ; Two character mnemonic for QIO TM78 TF
                      80  2B91                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  2B92                .Byte   Pd$_Name                ; Directive op-code
                      02  2B93                .Byte   Ptd$M_Controller                ; Enforced MF codes
                   46 4D 00' 2B94             .Ascic  "MF"    ; Enforced device name prefix
                      02  2B94
                      82  2B97                .BYTE   Pd$_Decimal             ; Indicate scan decimal
               45 56 49 52 44 00' 2B98        .ASCIC  "DRIVE" ; DRIVE string
                      05  2B98
               00000007 00000000  2B9E        .LONG   0,7     ; 0 . 7 limits on input
                      88  2BA6                .BYTE   Pd$_Store               ; Indicate store operation
                    0032  2BA7                .WORD   TM78$B_DRIVE                    ; Byte TM78$B_DRIVE to data
                      00  2BA9                .BYTE   0                       ; 0 TM78$B_DRIVE to data
                      08  2BAA                .BYTE   8                       ; 8 of data field
                      88  2BAB                .BYTE   Pd$_Store               ; Indicate store operation
                    000B  2BAC                .WORD   HP$B_DRIVE                      ; Byte HP$B_DRIVE to data
                      00  2BAE                .BYTE   0                       ; 0 HP$B_DRIVE to data
                      08  2BAF                .BYTE   8                       ; 8 of data field
                      88  2BB0                .BYTE   Pd$_Store               ; Indicate store operation
                    0018  2BB1                .WORD   HP$A_DEVICE                     ; Byte HP$A_DEVICE to data
                      07  2BB3                .BYTE   7                       ; 7 HP$A_DEVICE to data
                      03  2BB4                .BYTE   3                       ; 3 of data field
                      81  2BB5                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          2BB6    805 TS04:   $DS_TS04
                          2BB6                .SAVE   LOCAL_BLOCK
                          2BB6                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_TS04_CSR,       HP$L_TS04_CSR:
                          0032                .IIF    NB,TS04$L_CSR,  TS04$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_TS04_BR,        HP$B_TS04_BR:
                          0036                .IIF    NB,TS04$B_BR,   TS04$B_BR:
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_TS04_LEN,       HP$K_TS04_LEN:
                          0037                .IIF    NB,TS04$K_LEN,  TS04$K_LEN:
                      0000  2BB6                .RESTORE
               34 30 53 54 00' 2BB6           .ASCIC  "TS04"  ; TS04 name
                      04  2BB6
                      37  2BBB                .BYTE   TS04$K_LEN              ; TS04$K_LEN of resultant P-table
                      01  2BBC                .BYTE   1                       ; Maximum unit number                                   ;G:65
                    5354  2BBD                .WORD   ^A"TS"  ; Two character mnemonic for QIO TS04 TS
                      80  2BBF                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  2BC0                .Byte   Pd$_Name                ; Directive op-code
                      03  2BC1                .Byte   Ptd$M_Device                    ; Enforced MS codes
                   53 4D 00' 2BC2             .Ascic  "MS"    ; Enforced device name prefix
                      02  2BC2
                      86  2BC5                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  2BC6                .LONG   HP$M_ALLOC                      ; Constant
                      88  2BCA                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  2BCB                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  2BCD                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2BCE                .BYTE   8                       ; 8 of data field
                      83  2BCF                .BYTE   Pd$_Octal               ; Indicate scan octal
                52 53 43 00' 2BD0             .ASCIC  "CSR"   ; CSR string
                      03  2BD0
            0003FFFE 0003E000  2BD4           .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
```

```
                      88  2BDC                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  2BDD                .WORD   TS04$L_CSR              ; Byte TS04$L_CSR to data
                      00  2BDF                .BYTE   0               ; 0 TS04$L_CSR to data
                      20  2BE0                .BYTE   32              ; 32 of data field
                      88  2BE1                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2BE2                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  2BE4                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  2BE5                .BYTE   18              ; 18 of data field
                      83  2BE6                .BYTE   Pd$_Octal       ; Indicate scan octal
        52 4F 54 43 45 56 00' 2BE7                .ASCIC  "VECTOR"        ; VECTOR string
                      06  2BE7
    000001FE 00000002  2BEE                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  2BF6                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  2BF7                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  2BF9                .BYTE   0               ; 0 HP$W_VECTOR to data
                      09  2BFA                .BYTE   9               ; 9 of data field
                      82  2BFB                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 2BFC                .ASCIC  "BR"    ; BR string
                      02  2BFC
    00000007 00000004  2BFF                .LONG   4,7     ; 4 , 7 limits on input
                      88  2C07                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  2C08                .WORD   TS04$B_BR               ; Byte TS04$B_BR to data
                      00  2C0A                .BYTE   0               ; 0 TS04$B_BR to data
                      08  2C0B                .BYTE   8               ; 8 of data field
                      81  2C0C                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2C0D    806 TS05:   $DS_TS05                                ;       [43]
                          2C0D                .SAVE   LOCAL_BLOCK
                          2C0D                .PSECT  $ABS$,ABS
                 00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,TS05$B_BR,   TS05$B_BR:
                 00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$K_TS05_LEN,       HP$K_TS05_LEN:
                          0033                .IIF    NB,TS05$K_LEN,  TS05$K_LEN:
                      0000  2C0D                .RESTORE
              35 30 53 54 00' 2C0D                .ASCIC  "TS05"  ; TS05 name
                      04  2C0D
                      33  2C12                .BYTE   TS05$K_LEN              ; TS05$K_LEN of resultant P-table
                      08  2C13                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  2C14                .WORD   ^A""    ; Two character mnemonic for QIO TS05
                      80  2C16                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2C17                .Byte   Pd$_Name        ; Directive op-code
                      03  2C18                .Byte   Ptd$M_Device            ; Enforced MS codes
                 53 4D 00' 2C19                .Ascic  "MS"    ; Enforced device name prefix
                      02  2C19
                      86  2C1C                .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  2C1D                .LONG   HP$M_ALLOC              ; Constant
                      88  2C21                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  2C22                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  2C24                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  2C25                .BYTE   8               ; 8 of data field
                      83  2C26                .BYTE   Pd$_Octal       ; Indicate scan octal
44 41 5F 52 53 43 5F 35 30 53 54 00' 2C27                .ASCIC  "TS05_CSR_ADDRESS"      ; TS05_CSR_ADDRESS string
                 53 53 45 52 44  2C33
                      10  2C27
                0003FFFE 0003E000  2C38                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  2C40                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2C41                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
```

```
                        00  2C43                .BYTE   0                       ; 0 HP$A_DEVICE to data
                        12  2C44                .BYTE   18                      ; 18 of data field
                        83  2C45                .BYTE   Pd$_Octal               ; Indicate scan octal
52 4F 54 43 45 56 5F 35 30 53 54 00' 2C46       .ASCIC  "TS05_VECTOR"           ; TS05_VECTOR string
                        0B  2C46
               000001FE 00000002  2C52          .LONG   ^O2,^O776               ; 2 , 776 limits on input
                        88  2C5A                .BYTE   Pd$_Store               ; Indicate store operation
                      0024  2C5B                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                        00  2C5D                .BYTE   0                       ; 0 HP$W_VECTOR to data
                        10  2C5E                .BYTE   16                      ; 16 of data field
                        82  2C5F                .BYTE   Pd$_Decimal             ; Indicate scan decimal
         4C 45 56 45 4C 5F 52 42 00' 2C60       .ASCIC  "BR_LEVEL"              ; BR_LEVEL string
                        08  2C60
               00000007 00000004  2C69          .LONG   4,7     ; 4 , 7 limits on input
                        88  2C71                .BYTE   Pd$_Store               ; Indicate store operation
                      0032  2C72                .WORD   TS05$B_BR                       ; Byte TS05$B_BR to data
                        00  2C74                .BYTE   0                       ; 0 TS05$B_BR to data
                        08  2C75                .BYTE   8                       ; 8 of data field
                        81  2C76                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                            2C77    807 TS11:   $DS_TS11
                            2C77                .SAVE   LOCAL_BLOCK
                            2C77                .PSECT  $ABS$,ABS
                00000032    0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$L_TS11_CSR,       HP$L_TS11_CSR:
                            0032                .IIF    NB,TS11$L_CSR,  TS11$L_CSR:
                00000036    0032                .IIF    NB,,BLKL,       .BLKL   1
                            0036                .IIF    NB,HP$B_TS11_BR,        HP$B_TS11_BR:
                            0036                .IIF    NB,TS11$B_BR,   TS11$B_BR:
                00000037    0036                .IIF    NB,,BLKB,       .BLKB   1
                            0037                .IIF    NB,HP$K_TS11_LEN,       HP$K_TS11_LEN:
                            0037                .IIF    NB,TS11$K_LEN,  TS11$K_LEN:
                        0002C77                 .RESTORE
            31 31 53 54 00' 2C77                .ASCIC  "TS11"  ; TS11 name
                        04  2C77
                        37  2C7C                .BYTE   TS11$K_LEN                      ; TS11$K_LEN of resultant P-table
                        01  2C7D                .BYTE   1                       ; Maximum unit number                   ;G:65
                      5354  2C7E                .WORD   ^A"TS"  ; Two character mnemonic for QIO TS11 TS
                        80  2C80                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                        8D  2C81                .Byte   Pd$_Name                ; Directive op-code
                        03  2C82                .Byte   Ptd$M_Device                    ; Enforced MS codes
                     53 4D 00' 2C83             .Ascic  "MS"    ; Enforced device name prefix
                        02  2C83
                        86  2C86                .BYTE   Pd$_Literal             ; Indicate literal
                00000001    2C87                .LONG   HP$M_ALLOC                      ; Constant
                        88  2C8B                .BYTE   Pd$_Store               ; Indicate store operation
                      000A  2C8C                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                        00  2C8E                .BYTE   0                       ; 0 HP$B_FLAGS to data
                        08  2C8F                .BYTE   8                       ; 8 of data field
                        83  2C90                .BYTE   Pd$_Octal               ; Indicate scan octal
                  52 53 43 00' 2C91             .ASCIC  "CSR"   ; CSR string
                        03  2C91
            0003FFFE 0003E000  2C95             .LONG   ^O760000,^O777776               ; 760000 , 777776 limits on input
                        88  2C9D                .BYTE   Pd$_Store               ; Indicate store operation
                      0032  2C9E                .WORD   TS11$L_CSR                      ; Byte TS11$L_CSR to data
                        00  2CA0                .BYTE   0                       ; 0 TS11$L_CSR to data
                        20  2CA1                .BYTE   32                      ; 32 of data field
                        88  2CA2                .BYTE   Pd$_Store               ; Indicate store operation
```

```
                     0018  2CA3                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                       00  2CA5                .BYTE   0                       ; 0 HP$A_DEVICE to data
                       12  2CA6                .BYTE   18                      ; 18 of data field
                       83  2CA7                .BYTE   Pd$_Octal               ; Indicate scan octal
           52 4F 54 43 45 56 00' 2CA8          .ASCIC  "VECTOR"                ; VECTOR string
                       06  2CA8
              000001FE 00000002  2CAF          .LONG   ^O2,^O776               ; 2 , 776 limits on input
                       88  2CB7                .BYTE   Pd$_Store               ; Indicate store operation
                     0024  2CB8                .WORD   HP$W_VECTOR                     ; Byte HP$W_VECTOR to data
                       00  2CBA                .BYTE   0                       ; 0 HP$W_VECTOR to data
                       09  2CBB                .BYTE   9                       ; 9 of data field
                       82  2CBC                .BYTE   Pd$_Decimal             ; Indicate scan decimal
                  52 42 00' 2CBD               .ASCIC  "BR"    ; BR string
                       02  2CBD
              00000007 00000004  2CC0          .LONG   4,7     ; 4 , 7 limits on input
                       88  2CC8                .BYTE   Pd$_Store               ; Indicate store operation
                     0036  2CC9                .WORD   TS11$B_BR                       ; Byte TS11$B_BR to data
                       00  2CCB                .BYTE   0                       ; 0 TS11$B_BR to data
                       08  2CCC                .BYTE   8                       ; 8 of data field
                       81  2CCD                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                           2CCE    808 TU45:    $DS_TU45
                           2CCE                .SAVE   LOCAL_BLOCK
                           2CCE                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_TU45_LEN,       HP$K_TU45_LEN:
                           0032                .IIF    NB,TU45$K_LEN,  TU45$K_LEN:
                       0002CCE                 .RESTORE
                  35 34 55 54 00' 2CCE         .ASCIC  "TU45"  ; TU45 name
                       04  2CCE
                       32  2CD3                .BYTE   TU45$K_LEN                      ; TU45$K_LEN of resultant P-table
                       08  2CD4                .BYTE   8                       ; Maximum unit number                   ;G:65
                     0000  2CD5                .WORD   ^A""    ; Two character mnemonic for QIO TU45
                       80  2CD7                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                       8D  2CD8                .Byte   Pd$_Name                ; Directive op-code
                       1B  2CD9                .Byte   Ptd$M_EndDevice                 ; Enforced MT codes
                  54 4D 00' 2CDA               .Ascic  "MT"    ; Enforced device name prefix
                       02  2CDA
                       86  2CDD                .BYTE   Pd$_Literal             ; Indicate literal
                  00000001  2CDE                .LONG   HP$M_ALLOC                      ; Constant
                       88  2CE2                .BYTE   Pd$_Store               ; Indicate store operation
                     000A  2CE3                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                       00  2CE5                .BYTE   0                       ; 0 HP$B_FLAGS to data
                       08  2CE6                .BYTE   8                       ; 8 of data field
                       81  2CE7                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                           2CE8    809 TU58:    $DS_TU58
                           2CE8                .SAVE   LOCAL_BLOCK
                           2CE8                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_TU58_LEN,       HP$K_TU58_LEN:
                           0032                .IIF    NB,TU58$K_LEN,  TU58$K_LEN:
                       0002CE8                 .RESTORE
               38 35 55 54 00' 2CE8            .ASCIC  "TU58"  ; TU58 name
                       04  2CE8
                       32  2CED                .BYTE   TU58$K_LEN                      ; TU58$K_LEN of resultant P-table
                       02  2CEE                .BYTE   2                       ; Maximum unit number                   ;G:65
                     4444  2CEF                .WORD   ^A"DD"  ; Two character mnemonic for QIO TU58 DD
                       80  2CF1                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
```

```
                      8D  2CF2                .Byte   Pd$_Name         ; Directive op-code
                      03  2CF3                .Byte   Ptd$M_Device              ; Enforced DD codes
                44 44 00' 2CF4                .Ascic  "DD"    ; Enforced device name prefix
                      02  2CF4
                      86  2CF7                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  2CF8                .LONG   HP$M_ALLOC               ; Constant
                      88  2CFC                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2CFD                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  2CFF                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  2D00                .BYTE   8                ; 8 of data field
                      81  2D01                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2D02    810 TU70:    $DS_TU70                                 ;      [57]
                          2D02                .SAVE   LOCAL_BLOCK
                          2D02                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_TU70_LEN,       HP$K_TU70_LEN::
                          0032                .IIF    NB,TU70$K_LEN,  TU70$K_LEN::
                      0000  2D02                .RESTORE
             30 37 55 54 00' 2D02                .ASCIC  "TU70"  ; TU70 name
                      04  2D02
                      32  2D07                .BYTE   TU70$K_LEN               ; TU70$K_LEN of resultant P-table
                      10  2D08                .BYTE   16               ; Maximum unit number                          ;G:65
                    0000  2D09                .WORD   ^A""     ; Two character mnemonic for QIO TU70
                      80  2D0B                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  2D0C                .Byte   Pd$_Name         ; Directive op-code
                      1B  2D0D                .Byte   PTD$M_ENDDEVICE          ; Enforced MR codes
                52 4D 00' 2D0E                .Ascic  "MR"    ; Enforced device name prefix
                      02  2D0E
                      86  2D11                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  2D12                .LONG   HP$M_ALLOC               ; Constant
                      88  2D16                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2D17                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  2D19                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  2D1A                .BYTE   8                ; 8 of data field
                      81  2D1B                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2D1C    811 TU72:    $DS_TU72                                 ;      [57]
                          2D1C                .SAVE   LOCAL_BLOCK
                          2D1C                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_TU72_LEN,       HP$K_TU72_LEN::
                          0032                .IIF    NB,TU72$K_LEN,  TU72$K_LEN::
                      0000  2D1C                .RESTORE
             32 37 55 54 00' 2D1C                .ASCIC  "TU72"  ; TU72 name
                      04  2D1C
                      32  2D21                .BYTE   TU72$K_LEN               ; TU72$K_LEN of resultant P-table
                      10  2D22                .BYTE   16               ; Maximum unit number                          ;G:65
                    0000  2D23                .WORD   ^A""     ; Two character mnemonic for QIO TU72
                      80  2D25                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  2D26                .Byte   Pd$_Name         ; Directive op-code
                      1B  2D27                .Byte   PTD$M_ENDDEVICE          ; Enforced MR codes
                52 4D 00' 2D28                .Ascic  "MR"    ; Enforced device name prefix
                      02  2D28
                      86  2D2B                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  2D2C                .LONG   HP$M_ALLOC               ; Constant
                      88  2D30                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2D31                .WORD   HP$B_FLAGS               ; Byte HP$B_FLAGS to data
                      00  2D33                .BYTE   0                ; 0 HP$B_FLAGS to data
```

```
                      08  2D34                .BYTE   8                       ; 8 of data field
                      81  2D35                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          2D36    812 TU77:    $DS_TU77
                          2D36                .SAVE   LOCAL_BLOCK
                          2D36                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_TU77_LEN,       HP$K_TU77_LEN:
                          0032                .IIF    NB,TU77$K_LEN,  TU77$K_LEN:
                      0000 2D36               .RESTORE
             37 37 55 54 00' 2D36             .ASCIC  "TU77"  ; TU77 name
                      04  2D36
                      32  2D3B                .BYTE   TU77$K_LEN                      ; TU77$K_LEN of resultant P-table
                      08  2D3C                .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  2D3D                .WORD   ^A""    ; Two character mnemonic for QIO TU77
                      80  2D3F                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  2D40                .Byte   Pd$_Name                ; Directive op-code
                      1B  2D41                .Byte   Ptd$M_EndDevice                 ; Enforced MT codes
                54 4D 00' 2D42                .Ascic  "MT"    ; Enforced device name prefix
                      02  2D42
                      86  2D45                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  2D46                .LONG   HP$M_ALLOC                      ; Constant
                      88  2D4A                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  2D4B                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  2D4D                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2D4E                .BYTE   8                       ; 8 of data field
                      81  2D4F                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          2D50    813 TU78:    $DS_TU78
                          2D50                .SAVE   LOCAL_BLOCK
                          2D50                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_TU78_LEN,       HP$K_TU78_LEN:
                          0032                .IIF    NB,TU78$K_LEN,  TU78$K_LEN:
                      0000 2D50               .RESTORE
             38 37 55 54 00' 2D50             .ASCIC  "TU78"  ; TU78 name
                      04  2D50
                      32  2D55                .BYTE   TU78$K_LEN                      ; TU78$K_LEN of resultant P-table
                      08  2D56                .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  2D57                .WORD   ^A""    ; Two character mnemonic for QIO TU78
                      80  2D59                .BYTE   Pd$_Start               ; Constant to identify start of descriptor
                      8D  2D5A                .Byte   Pd$_Name                ; Directive op-code
                      1B  2D5B                .Byte   Ptd$M_EndDevice                 ; Enforced MF codes
                46 4D 00' 2D5C                .Ascic  "MF"    ; Enforced device name prefix
                      02  2D5C
                      86  2D5F                .BYTE   Pd$_Literal             ; Indicate literal
                00000001  2D60                .LONG   HP$M_ALLOC                      ; Constant
                      88  2D64                .BYTE   Pd$_Store               ; Indicate store operation
                    000A  2D65                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  2D67                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  2D68                .BYTE   8                       ; 8 of data field
                      81  2D69                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          2D6A    814 TU80:    $DS_TU80                                ;       [28]
                          2D6A                .SAVE   LOCAL_BLOCK
                          2D6A                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_TU80_CSR,       HP$L_TU80_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,TU80$B_BR,   TU80$B_BR:
```

```
                            0036                .IIF    NB,HP$B_TU80_BR,        HP$B_TU80_BR:
                00000037    0036                .IIF    NB,.BLKB,       .BLKB   1
                            0037                .IIF    NB,HP$K_TU80_LEN,       HP$K_TU80_LEN:
                            0037                .IIF    NB,TU80$K_LEN,  TU80$K_LEN:
                  00002D6A                      .RESTORE
          30 38 55 54 00'   2D6A                .ASCIC  "TU80"  ; TU80 name
                      04    2D6A
                      37    2D6F                .BYTE   TU80$K_LEN              ; TU80$K_LEN of resultant P-table
                      08    2D70                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000    2D71                .WORD   ^A""    ; Two character mnemonic for QIO TU80
                      80    2D73                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D    2D74                .Byte   Pd$_Name        ; Directive op-code
                      03    2D75                .Byte   Ptd$M_Device            ; Enforced MS codes
                53 4D 00'   2D76                .Ascic  "MS"    ; Enforced device name prefix
                      02    2D76
                      86    2D79                .BYTE   Pd$_Literal     ; Indicate literal
                00000001    2D7A                .LONG   HP$M_ALLOC              ; Constant
                      88    2D7E                .BYTE   Pd$_Store       ; Indicate store operation
                    000A    2D7F                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00    2D81                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08    2D82                .BYTE   8               ; 8 of data field
                      83    2D83                .BYTE   Pd$_Octal       ; Indicate scan octal
44 41 5F 52 53 43 5F 30 38 55 54 00'   2D84      .ASCIC  "TU80_CSR_ADDRESS"      ; TU80_CSR_ADDRESS string
          53 53 45 52 44      2D90
                      10    2D84
            0003FFFE 0003E000   2D95             .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88    2D9D                .BYTE   Pd$_Store       ; Indicate store operation
                    0032    2D9E                .WORD   HP$L_TU80_CSR           ; Byte HP$L_TU80_CSR to data
                      00    2DA0                .BYTE   0               ; 0 HP$L_TU80_CSR to data
                      20    2DA1                .BYTE   32              ; 32 of data field
                      88    2DA2                .BYTE   Pd$_Store       ; Indicate store operation
                    0018    2DA3                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00    2DA5                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12    2DA6                .BYTE   18              ; 18 of data field
                      83    2DA7                .BYTE   Pd$_Octal       ; Indicate scan octal
52 4F 54 43 45 56 5F 30 38 55 54 00'   2DA8      .ASCIC  "TU80_VECTOR"   ; TU80_VECTOR string
                      0B    2DA8
            000001FE 00000002   2DB4             .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88    2DBC                .BYTE   Pd$_Store       ; Indicate store operation
                    0024    2DBD                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00    2DBF                .BYTE   0               ; 0 HP$W_VECTOR to data
                      10    2DC0                .BYTE   16              ; 16 of data field
                      82    2DC1                .BYTE   Pd$_Decimal     ; Indicate scan decimal
        4C 45 56 45 4C 5F 52 42 00'   2DC2      .ASCIC  "BR_LEVEL"      ; BR_LEVEL string
                      08    2DC2
            00000007 00000004   2DCB             .LONG   4,7     ; 4 , 7 limits on input
                      88    2DD3                .BYTE   Pd$_Store       ; Indicate store operation
                    0036    2DD4                .WORD   HP$B_TU80_BR             ; Byte HP$B_TU80_BR to data
                      00    2DD6                .BYTE   0               ; 0 HP$B_TU80_BR to data
                      08    2DD7                .BYTE   8               ; 8 of data field
                      81    2DD8                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                            2DD9    815 TU81:   $DS_TU81                                ;       [29]
                            2DD9                .SAVE   LOCAL_BLOCK
                            2DD9                .PSECT  $ABS$,ABS
                00000032    0050                .=HP$A_DEPENDENT
                            0032                .IIF    NB,HP$L_TU81_CSR,       HP$L_TU81_CSR:
                            0032                .IIF    NB,TU81$L_CSR,  TU81$L_CSR:
```

```
                    00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                              0036                .IIF    NB,HP$B_TU81_BR,        HP$B_TU81_BR:
                              0036                .IIF    NB,TU81$B_BR,   TU81$B_BR:
                    00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                              0037                .IIF    NB,HP$K_TU81_LEN,       HP$K_TU81_LEN:
                              0037                .IIF    NB,TU81$K_LEN,  TU81$K_LEN:
                        00002DD9                  .RESTORE
                31 38 55 54 00' 2DD9              .ASCIC  "TU81"  ; TU81 name
                          04  2DD9
                          37  2DDE                .BYTE   TU81$K_LEN              ; TU81$K_LEN of resultant P-table
                          04  2DDF                .BYTE   4                       ; Maximum unit number                   ;G:65
                        554D  2DE0                .WORD   ^A"MU"  ; Two character mnemonic for QIO TU81 MU
                          80  2DE2                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  2DE3                .Byte   Pd$_Name        ; Directive op-code
                          03  2DE4                .Byte   Ptd$M_Device            ; Enforced MU codes
                    55 4D 00' 2DE5                .Ascic  "MU"    ; Enforced device name prefix
                          02  2DE5
                          86  2DE8                .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  2DE9                .LONG   HP$M_ALLOC              ; Constant
                          88  2DED                .BYTE   Pd$_Store       ; Indicate store operation
                        000A  2DEE                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  2DF0                .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  2DF1                .BYTE   8               ; 8 of data field
                          83  2DF2                .BYTE   Pd$_Octal       ; Indicate scan octal
                 52 53 43 00' 2DF3                .ASCIC  "CSR"   ; CSR string
                          03  2DF3
                0003FFFE 0003E000  2DF7           .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  2DFF                .BYTE   Pd$_Store       ; Indicate store operation
                        0032  2E00                .WORD   TU81$L_CSR              ; Byte TU81$L_CSR to data
                          00  2E02                .BYTE   0               ; 0 TU81$L_CSR to data
                          20  2E03                .BYTE   32              ; 32 of data field
                          88  2E04                .BYTE   Pd$_Store       ; Indicate store operation
                        0018  2E05                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  2E07                .BYTE   0               ; 0 HP$A_DEVICE to data
                          12  2E08                .BYTE   18              ; 18 of data field
                          83  2E09                .BYTE   Pd$_Octal       ; Indicate scan octal
          52 4F 54 43 45 56 00' 2E0A              .ASCIC  "VECTOR"        ; VECTOR string
                          06  2E0A
                000001FE 00000002  2E11           .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  2E19                .BYTE   Pd$_Store       ; Indicate store operation
                        0024  2E1A                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  2E1C                .BYTE   0               ; 0 HP$W_VECTOR to data
                          09  2E1D                .BYTE   9               ; 9 of data field
                          82  2E1E                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                    52 42 00' 2E1F                .ASCIC  "BR"    ; BR string
                          02  2E1F
                00000007 00000004  2E22           .LONG   4,7     ; 4 , 7 limits on input
                          88  2E2A                .BYTE   Pd$_Store       ; Indicate store operation
                        0036  2E2B                .WORD   TU81$B_BR               ; Byte TU81$B_BR to data
                          00  2E2D                .BYTE   0               ; 0 TU81$B_BR to data
                          08  2E2E                .BYTE   8               ; 8 of data field
                          81  2E2F                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              2E30    816 UBE:    $DS_UBE
                              2E30                .SAVE   LOCAL_BLOCK
                              2E30                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$L_UBE_CSR,        HP$L_UBE_CSR:
```

```
                        0032                .IIF    NB,UBE$L_CSR,   UBE$L_CSR:
                    00000036  0032            .IIF    NB,.BLKL,       .BLKL   1
                        0036                .IIF    NB,HP$K_UBE_LEN,        HP$K_UBE_LEN:
                        0036                .IIF    NB,UBE$K_LEN,   UBE$K_LEN:
                    00002E30                .RESTORE
                45 42 55 00' 2E30           .ASCIC  "UBE"   ; UBE name
                          03  2E30
                          36  2E34          .BYTE   UBE$K_LEN               ; UBE$K_LEN of resultant P-table
                          08  2E35          .BYTE   8               ; Maximum unit number                   ;G:65
                        4555  2E36          .WORD   ^A"UE"  ; Two character mnemonic for QIO UBE UE
                          80  2E38          .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  2E39          .Byte   Pd$_Name        ; Directive op-code
                          03  2E3A          .Byte   Ptd$M_Device            ; Enforced UB codes
                    42 55 00' 2E3B          .Ascic  "UB"    ; Enforced device name prefix
                          02  2E3B
                          83  2E3E          .BYTE   Pd$_Octal       ; Indicate scan octal
                    52 53 43 00' 2E3F       .ASCIC  "CSR"   ; CSR string
                          03  2E3F
                0003FFFE 0003E000  2E43     .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  2E4B          .BYTE   Pd$_Store       ; Indicate store operation
                        0032  2E4C          .WORD   UBE$L_CSR               ; Byte UBE$L_CSR to data
                          00  2E4E          .BYTE   0               ; 0 UBE$L_CSR to data
                          20  2E4F          .BYTE   32              ; 32 of data field
                          88  2E50          .BYTE   Pd$_Store       ; Indicate store operation
                        0018  2E51          .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  2E53          .BYTE   0               ; 0 HP$A_DEVICE to data
                          12  2E54          .BYTE   18              ; 18 of data field
                          83  2E55          .BYTE   Pd$_Octal       ; Indicate scan octal
            52 4F 54 43 45 56 00' 2E56      .ASCIC  "VECTOR"        ; VECTOR string
                          06  2E56
                000001FE 00000002  2E5D     .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  2E65          .BYTE   Pd$_Store       ; Indicate store operation
                        0024  2E66          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  2E68          .BYTE   0               ; 0 HP$W_VECTOR to data
                          09  2E69          .BYTE   9               ; 9 of data field
                          81  2E6A          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              2E6B  817 UDA50:  $DS_UDA50                             ;       [17]
                              2E6B          .SAVE   LOCAL_BLOCK
                              2E6B          .PSECT  $ABS$,ABS
                    00000032  0050          .=HP$A_DEPENDENT
                              0032          .IIF    NB,HP$L_UDA50_UDAIP,    HP$L_UDA50_UDAIP:
                              0032          .IIF    NB,UDA50$L_UDAIP,       UDA50$L_UDAIP:
                    00000036  0032          .IIF    NB,.BLKL,       .BLKL   1
                              0036          .IIF    NB,HP$B_UDA50_BR,       HP$B_UDA50_BR:
                              0036          .IIF    NB,UDA50$B_BR,  UDA50$B_BR:
                    00000037  0036          .IIF    NB,.BLKB,       .BLKB   1
                              0037          .IIF    NB,HP$B_UDA50_BURST,    HP$B_UDA50_BURST:
                              0037          .IIF    NB,UDA50$B_BURST,       UDA50$B_BURST:
                    00000038  0037          .IIF    NB,.BLKB,       .BLKB   1
                              0038          .IIF    NB,HP$K_UDA50_LEN,      HP$K_UDA50_LEN:
                              0038          .IIF    NB,UDA50$K_LEN, UDA50$K_LEN:
                    00002E6B                .RESTORE
            30 35 41 44 55 00' 2E6B         .ASCIC  "UDA50" ; UDA50 name
                          05  2E6B
                          38  2E71          .BYTE   HP$K_UDA50_LEN          ; HP$K_UDA50_LEN of resultant P-table
                          00  2E72          .BYTE   0               ; Maximum unit number                   ;G:65
                        0000  2E73          .WORD   ^A""    ; Two character mnemonic for QIO UDA50
```

```
                      80  2E75                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2E76                .Byte   Pd$_Name        ; Directive op-code
                      02  2E77                .Byte   Ptd$M_Controller                ; Enforced DU codes
                55 44 00' 2E78                .Ascic  "DU"    ; Enforced device name prefix
                      02  2E78
                      83  2E7B                .BYTE   Pd$_Octal       ; Indicate scan octal
          50 49 41 44 55 00' 2E7C             .ASCIC  "UDAIP" ; UDAIP string
                      05  2E7C
          0003FFFC 0003E000  2E82             .LONG   ^O760000,^O777774       ; 760000 , 777774 limits on input
                      88  2E8A                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  2E8B                .WORD   HP$L_UDA50_UDAIP                ; Byte HP$L_UDA50_UDAIP to data
                      00  2E8D                .BYTE   0               ; 0 HP$L_UDA50_UDAIP to data
                      20  2E8E                .BYTE   32              ; 32 of data field
                      88  2E8F                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2E90                .WORD   Hp$A_Device             ; Byte Hp$A_Device to data
                      00  2E92                .BYTE   0               ; 0 Hp$A_Device to data
                      12  2E93                .BYTE   18              ; 18 of data field
                      83  2E94                .BYTE   Pd$_Octal       ; Indicate scan octal
       72 6F 74 63 65 56 00' 2E95             .ASCIC  "Vector"        ; Vector string
                      06  2E95
          000001FC 00000004  2E9C             .LONG   ^O4,^O774       ; 4 , 774 limits on input
                      88  2EA4                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  2EA5                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  2EA7                .BYTE   0               ; 0 HP$W_VECTOR to data
                      08  2EA8                .BYTE   8               ; 8 of data field
                      82  2EA9                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                52 42 00' 2EAA                .ASCIC  "BR"    ; BR string
                      02  2EAA
          00000007 00000004  2EAD             .LONG   4,7     ; 4 , 7 limits on input
                      88  2EB5                .BYTE   Pd$_Store       ; Indicate store operation
                    0036  2EB6                .WORD   HP$B_UDA50_BR           ; Byte HP$B_UDA50_BR to data
                      00  2EB8                .BYTE   0               ; 0 HP$B_UDA50_BR to data
                      08  2EB9                .BYTE   8               ; 8 of data field
                      82  2EBA                .BYTE   Pd$_Decimal     ; Indicate scan decimal
65 74 61 52 5F 74 73 72 75 42 00' 2EBB        .ASCIC  "Burst_Rate"    ; Burst_Rate string
                      0A  2EBB
          0000003F 00000000  2EC6             .LONG   0,63    ; 0 , 63 limits on input
                      88  2ECE                .BYTE   Pd$_Store       ; Indicate store operation
                    0037  2ECF                .WORD   HP$B_UDA50_BURST                ; Byte HP$B_UDA50_BURST to data
                      00  2ED1                .BYTE   0               ; 0 HP$B_UDA50_BURST to data
                      06  2ED2                .BYTE   6               ; 6 of data field
                      81  2ED3                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2ED4    818 UNA11:  $DS_UNA11                               ;       [27]
                          2ED4                .SAVE   LOCAL_BLOCK
                          2ED4                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,UNA11$B_BR,  UNA11$B_BR:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,UNA11$L_CSR, UNA11$L_CSR:
                00000037  0033                .IIF    NB,.BLKL,       .BLKL   1
                          0037                .IIF    NB,UNA11$K_LEN, UNA11$K_LEN:
                      0002ED4                 .RESTORE
          31 31 41 4E 55 00' 2ED4             .ASCIC  "UNA11" ; UNA11 name
                      05  2ED4
                      37  2EDA                .BYTE   UNA11$K_LEN             ; UNA11$K_LEN of resultant P-table
                      08  2EDB                .BYTE   8               ; Maximum unit number                           ;G:65
                    0000  2EDC                .WORD   ^A""    ; Two character mnemonic for QIO UNA11
```

```
                      80  2EDE                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2EDF                .Byte   Pd$_Name        ; Directive op-code
                      03  2EE0                .Byte   PTD$M_DEVICE            ; Enforced XE codes
                 45 58 00' 2EE1                .Ascic  "XE"    ; Enforced device name prefix
                      02  2EE1
                      83  2EE4                .BYTE   Pd$_Octal       ; Indicate scan octal
              52 53 43 00' 2EE5                .ASCIC  "CSR"   ; CSR string
                      03  2EE5
          0003FFFE 0003E000  2EE9                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                      88  2EF1                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  2EF2                .WORD   UNA11$L_CSR             ; Byte UNA11$L_CSR to data
                      00  2EF4                .BYTE   0               ; 0 UNA11$L_CSR to data
                      20  2EF5                .BYTE   32              ; 32 of data field
                      88  2EF6                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  2EF7                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  2EF9                .BYTE   0               ; 0 HP$A_DEVICE to data
                      12  2EFA                .BYTE   18              ; 18 of data field
                      83  2EFB                .BYTE   Pd$_Octal       ; Indicate scan octal
        52 4F 54 43 45 56 00' 2EFC                .ASCIC  "VECTOR"        ; VECTOR string
                      06  2EFC
          000001FE 00000002  2F03                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                      88  2F0B                .BYTE   Pd$_Store       ; Indicate store operation
                    0024  2F0C                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                      00  2F0E                .BYTE   0               ; 0 HP$W_VECTOR to data
                      09  2F0F                .BYTE   9               ; 9 of data field
                      82  2F10                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 2F11                .ASCIC  "BR"    ; BR string
                      02  2F11
          00000007 00000004  2F14                .LONG   4,7     ; 4 , 7 limits on input
                      88  2F1C                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  2F1D                .WORD   UNA11$B_BR              ; Byte UNA11$B_BR to data
                      00  2F1F                .BYTE   0               ; 0 UNA11$B_BR to data
                      08  2F20                .BYTE   8               ; 8 of data field
                      81  2F21                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          2F22    819 VS100:  $DS_VS100                               ;       [30]
                          2F22                .SAVE   LOCAL_BLOCK
                          2F22                .PSECT  $ABS$,ABS
                 00000032 0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_VS100_CSR,      HP$L_VS100_CSR:
                          0032                .IIF    NB,VS100$L_CSR, VS100$L_CSR:
                 00000036 0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_VS100_BR,       HP$B_VS100_BR:
                          0036                .IIF    NB,VS100$B_BR,  VS100$B_BR:
                 00000037 0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_VS100_LEN,      HP$K_VS100_LEN:
                          0037                .IIF    NB,VS100$K_LEN, VS100$K_LEN:
                      00002F22                .RESTORE
        30 30 31 53 56 00' 2F22                .ASCIC  "VS100" ; VS100 name
                      05  2F22
                      37  2F28                .BYTE   VS100$K_LEN             ; VS100$K_LEN of resultant P-table
                      01  2F29                .BYTE   1               ; Maximum unit number                           ;G:65
                    5356  2F2A                .WORD   ^A"VS"  ; Two character mnemonic for QIO VS100 VS
                      80  2F2C                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  2F2D                .Byte   Pd$_Name        ; Directive op-code
                      1B  2F2E                .Byte   Ptd$M_EndDevice         ; Enforced VB codes
                 42 56 00' 2F2F                .Ascic  "VB"    ; Enforced device name prefix
                      02  2F2F
```

```
                      86  2F32                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  2F33                .LONG   HP$M_ALLOC              ; Constant
                      88  2F37                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  2F38                .WORD   HP$B_FLAGS                ; Byte HP$B_FLAGS to data
                      00  2F3A                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  2F3B                .BYTE   8                ; 8 of data field
                      83  2F3C                .BYTE   Pd$_Octal        ; Indicate scan octal
              52 53 43 00' 2F3D                .ASCIC  "CSR"    ; CSR string
                      03  2F3D
          0003FFFE 0003E000  2F41                .LONG   ^O760000,^O777776        ; 760000 , 777776 limits on input
                      88  2F49                .BYTE   Pd$_Store        ; Indicate store operation
                    0032  2F4A                .WORD   VS100$L_CSR               ; Byte VS100$L_CSR to data
                      00  2F4C                .BYTE   0                ; 0 VS100$L_CSR to data
                      20  2F4D                .BYTE   32               ; 32 of data field
                      88  2F4E                .BYTE   Pd$_Store        ; Indicate store operation
                    0018  2F4F                .WORD   HP$A_DEVICE               ; Byte HP$A_DEVICE to data
                      00  2F51                .BYTE   0                ; 0 HP$A_DEVICE to data
                      12  2F52                .BYTE   18               ; 18 of data field
                      83  2F53                .BYTE   Pd$_Octal        ; Indicate scan octal
        52 4F 54 43 45 56 00' 2F54                .ASCIC  "VECTOR"         ; VECTOR string
                      06  2F54
          000001FE 00000002  2F5B                .LONG   ^O2,^O776        ; 2 , 776 limits on input
                      88  2F63                .BYTE   Pd$_Store        ; Indicate store operation
                    0024  2F64                .WORD   HP$W_VECTOR               ; Byte HP$W_VECTOR to data
                      00  2F66                .BYTE   0                ; 0 HP$W_VECTOR to data
                      09  2F67                .BYTE   9                ; 9 of data field
                      82  2F68                .BYTE   Pd$_Decimal      ; Indicate scan decimal
                52 42 00' 2F69                .ASCIC  "BR"     ; BR string
                      02  2F69
          00000007 00000004  2F6C                .LONG   4,7      ; 4 , 7 limits on input
                      88  2F74                .BYTE   Pd$_Store        ; Indicate store operation
                    0036  2F75                .WORD   VS100$B_BR                ; Byte VS100$B_BR to data
                      00  2F77                .BYTE   0                ; 0 VS100$B_BR to data
                      08  2F78                .BYTE   8                ; 8 of data field
                      81  2F79                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          2F7A    820 VS125:    $DS_VS125                                ;       [40]
                          2F7A                .SAVE   LOCAL_BLOCK
                          2F7A                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$L_VS125_CSR,      HP$L_VS125_CSR:
                          0032                .IIF    NB,VS125$L_CSR, VS125$L_CSR:
                00000036  0032                .IIF    NB,.BLKL,       .BLKL   1
                          0036                .IIF    NB,HP$B_VS125_BR,       HP$B_VS125_BR:
                          0036                .IIF    NB,VS125$B_BR,  VS125$B_BR:
                00000037  0036                .IIF    NB,.BLKB,       .BLKB   1
                          0037                .IIF    NB,HP$K_VS125_LEN,      HP$K_VS125_LEN:
                          0037                .IIF    NB,VS125$K_LEN, VS125$K_LEN:
                      0002F7A                 .RESTORE
        35 32 31 53 56 00' 2F7A                .ASCIC  "VS125" ; VS125 name
                      05  2F7A
                      37  2F80                .BYTE   VS125$K_LEN              ; VS125$K_LEN of resultant P-table
                      01  2F81                .BYTE   1                ; Maximum unit number                          ;G:65
                    5356  2F82                .WORD   ^A"VS"   ; Two character mnemonic for QIO VS125 VS
                      80  2F84                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  2F85                .Byte   Pd$_Name         ; Directive op-code
                      1B  2F86                .Byte   Ptd$M_EndDevice          ; Enforced VB codes
                42 56 00' 2F87                .Ascic  "VB"     ; Enforced device name prefix
```

```
                    02  2F87
                    86  2F8A                .BYTE   Pd$_Literal     ; Indicate literal
              00000001  2F8B                .LONG   HP$M_ALLOC              ; Constant
                    88  2F8F                .BYTE   Pd$_Store       ; Indicate store operation
                  000A  2F90                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                    00  2F92                .BYTE   0               ; 0 HP$B_FLAGS to data
                    08  2F93                .BYTE   8               ; 8 of data field
                    83  2F94                .BYTE   Pd$_Octal       ; Indicate scan octal
           52 53 43 00' 2F95                .ASCIC  "CSR"   ; CSR string
                    03  2F95
       0003FFFE 0003E000  2F99                .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                    88  2FA1                .BYTE   Pd$_Store       ; Indicate store operation
                  0032  2FA2                .WORD   VS125$L_CSR             ; Byte VS125$L_CSR to data
                    00  2FA4                .BYTE   0               ; 0 VS125$L_CSR to data
                    20  2FA5                .BYTE   32              ; 32 of data field
                    88  2FA6                .BYTE   Pd$_Store       ; Indicate store operation
                  0018  2FA7                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                    00  2FA9                .BYTE   0               ; 0 HP$A_DEVICE to data
                    12  2FAA                .BYTE   18              ; 18 of data field
                    83  2FAB                .BYTE   Pd$_Octal       ; Indicate scan octal
     52 4F 54 43 45 56 00' 2FAC             .ASCIC  "VECTOR"        ; VECTOR string
                    06  2FAC
       000001FE 00000002  2FB3                .LONG   ^O2,^O776       ; 2 , 776 limits on input
                    88  2FBB                .BYTE   Pd$_Store       ; Indicate store operation
                  0024  2FBC                .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                    00  2FBE                .BYTE   0               ; 0 HP$W_VECTOR to data
                    09  2FBF                .BYTE   9               ; 9 of data field
                    82  2FC0                .BYTE   Pd$_Decimal     ; Indicate scan decimal
                 52 42 00' 2FC1             .ASCIC  "BR"    ; BR string
                    02  2FC1
       00000007 00000004  2FC4                .LONG   4,7     ; 4 , 7 limits on input
                    88  2FCC                .BYTE   Pd$_Store       ; Indicate store operation
                  0036  2FCD                .WORD   VS125$B_BR              ; Byte VS125$B_BR to data
                    00  2FCF                .BYTE   0               ; 0 VS125$B_BR to data
                    08  2FD0                .BYTE   8               ; 8 of data field
                    81  2FD1                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                        2FD2    821 VS300:  $DS_VS300                               ;       [33]
                        2FD2                .SAVE   LOCAL_BLOCK
                        2FD2                .PSECT  $ABS$,ABS
                  00000032  0050            .=HP$A_DEPENDENT
                        0032                .IIF    NB,HP$L_VS300_CSR,      HP$L_VS300_CSR:
                        0032                .IIF    NB,VS300$L_CSR, VS300$L_CSR:
                  00000036  0032            .IIF    NB,.BLKL,       .BLKL   1
                        0036                .IIF    NB,HP$B_VS300_BR,       HP$B_VS300_BR:
                        0036                .IIF    NB,VS300$B_BR,  VS300$B_BR:
                  00000037  0036            .IIF    NB,.BLKB,       .BLKB   1
                        0037                .IIF    NB,HP$K_VS300_LEN,      HP$K_VS300_LEN:
                        0037                .IIF    NB,VS300$K_LEN, VS300$K_LEN:
                    00002FD2                .RESTORE
     30 30 33 53 56 00' 2FD2                .ASCIC  "VS300" ; VS300 name
                    05  2FD2
                    37  2FD8                .BYTE   VS300$K_LEN             ; VS300$K_LEN of resultant P-table
                    01  2FD9                .BYTE   1               ; Maximum unit number                   ;G:65
                  5356  2FDA                .WORD   ^A"VS"  ; Two character mnemonic for QIO VS300 VS
                    80  2FDC                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                    8D  2FDD                .Byte   Pd$_Name        ; Directive op-code
                    1B  2FDE                .Byte   Ptd$M_EndDevice         ; Enforced VB codes
```

```
                    42 56 00' 2FDF                  .Ascic  "VB"    ; Enforced device name prefix
                          02  2FDF
                          86  2FE2                  .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  2FE3                  .LONG   HP$M_ALLOC              ; Constant
                          88  2FE7                  .BYTE   Pd$_Store       ; Indicate store operation
                        000A  2FE8                  .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  2FEA                  .BYTE   0               ; 0 HP$B_FLAGS to data
                          08  2FEB                  .BYTE   8               ; 8 of data field
                          83  2FEC                  .BYTE   Pd$_Octal       ; Indicate scan octal
                 52 53 43 00' 2FED                  .ASCIC  "CSR"   ; CSR string
                          03  2FED
             0003FFFE 0003E000  2FF1                  .LONG   ^O760000,^O777776       ; 760000 , 777776 limits on input
                          88  2FF9                  .BYTE   Pd$_Store       ; Indicate store operation
                        0032  2FFA                  .WORD   VS300$L_CSR             ; Byte VS300$L_CSR to data
                          00  2FFC                  .BYTE   0               ; 0 VS300$L_CSR to data
                          20  2FFD                  .BYTE   32              ; 32 of data field
                          88  2FFE                  .BYTE   Pd$_Store       ; Indicate store operation
                        0018  2FFF                  .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                          00  3001                  .BYTE   0               ; 0 HP$A_DEVICE to data
                          12  3002                  .BYTE   18              ; 18 of data field
                          83  3003                  .BYTE   Pd$_Octal       ; Indicate scan octal
           52 4F 54 43 45 56 00' 3004               .ASCIC  "VECTOR"        ; VECTOR string
                          06  3004
             000001FE 00000002  300B                  .LONG   ^O2,^O776       ; 2 , 776 limits on input
                          88  3013                  .BYTE   Pd$_Store       ; Indicate store operation
                        0024  3014                  .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
                          00  3016                  .BYTE   0               ; 0 HP$W_VECTOR to data
                          09  3017                  .BYTE   9               ; 9 of data field
                          82  3018                  .BYTE   Pd$_Decimal     ; Indicate scan decimal
                    52 42 00' 3019                  .ASCIC  "BR"    ; BR string
                          02  3019
             00000007 00000004  301C                  .LONG   4,7     ; 4 , 7 limits on input
                          88  3024                  .BYTE   Pd$_Store       ; Indicate store operation
                        0036  3025                  .WORD   VS300$B_BR              ; Byte VS300$B_BR to data
                          00  3027                  .BYTE   0               ; 0 VS300$B_BR to data
                          08  3028                  .BYTE   8               ; 8 of data field
                          81  3029                  .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              302A  822 VT50:       $DS_VT50
                              302A                  .SAVE   LOCAL_BLOCK
                              302A                  .PSECT  $ABS$,ABS
                    00000032  0050                  .=HP$A_DEPENDENT
                              0032                  .IIF    NB,HP$K_VT50_LEN,       HP$K_VT50_LEN:
                              0032                  .IIF    NB,VT50$K_LEN,  VT50$K_LEN:
                          0000302A                  .RESTORE
              30 35 54 56 00' 302A                  .ASCIC  "VT50"  ; VT50 name
                          04  302A
                          32  302F                  .BYTE   VT50$K_LEN              ; VT50$K_LEN of resultant P-table
                          FF  3030                  .BYTE   255             ; Maximum unit number                   ;G:65
                        0000  3031                  .WORD   ^A""    ; Two character mnemonic for QIO VT50
                          80  3033                  .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  3034                  .Byte   Pd$_Name        ; Directive op-code
                          1B  3035                  .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                    54 54 00' 3036                  .Ascic  "TT"    ; Enforced device name prefix
                          02  3036
                          86  3039                  .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  303A                  .LONG   HP$M_ALLOC              ; Constant
                          88  303E                  .BYTE   Pd$_Store       ; Indicate store operation
```

```
                     000A  303F                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                       00  3041                .BYTE   0                   ; 0 HP$B_FLAGS to data
                       08  3042                .BYTE   8                   ; 8 of data field
                       81  3043                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           3044    823 VT52:   $DS_VT52
                           3044                .SAVE   LOCAL_BLOCK
                           3044                .PSECT  $ABS$,ABS
                   00000032  0050              .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_VT52_LEN,       HP$K_VT52_LEN:
                           0032                .IIF    NB,VT52$K_LEN,  VT52$K_LEN:
                       0000304C                .RESTORE
                32 35 54 56 00' 304C           .ASCIC  "VT52"  ; VT52 name
                       04  3052
                       32  3057                .BYTE   VT52$K_LEN              ; VT52$K_LEN of resultant P-table
                       FF  3058                .BYTE   255                 ; Maximum unit number                   ;G:65
                     0000  3059                .WORD   ^A""    ; Two character mnemonic for QIO VT52
                       80  305B                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  305C                .Byte   Pd$_Name            ; Directive op-code
                       1B  305D                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                54 54 00' 305E                .Ascic  "TT"    ; Enforced device name prefix
                       02  305E
                       86  3061                .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  3062                .LONG   HP$M_ALLOC              ; Constant
                       88  3066                .BYTE   Pd$_Store           ; Indicate store operation
                     000A  3067                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                       00  3069                .BYTE   0                   ; 0 HP$B_FLAGS to data
                       08  306A                .BYTE   8                   ; 8 of data field
                       81  306B                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           306C    824 VT55:   $DS_VT55
                           306C                .SAVE   LOCAL_BLOCK
                           306C                .PSECT  $ABS$,ABS
                   00000032  0050              .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_VT55_LEN,       HP$K_VT55_LEN:
                           0032                .IIF    NB,VT55$K_LEN,  VT55$K_LEN:
                       0000306C                .RESTORE
                35 35 54 56 00' 306C           .ASCIC  "VT55"  ; VT55 name
                       04  3072
                       32  3077                .BYTE   VT55$K_LEN              ; VT55$K_LEN of resultant P-table
                       FF  3078                .BYTE   255                 ; Maximum unit number                   ;G:65
                     0000  3079                .WORD   ^A""    ; Two character mnemonic for QIO VT55
                       80  307B                .BYTE   Pd$_Start           ; Constant to identify start of descriptor
                       8D  307C                .Byte   Pd$_Name            ; Directive op-code
                       1B  307D                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                54 54 00' 307E                .Ascic  "TT"    ; Enforced device name prefix
                       02  307E
                       86  3081                .BYTE   Pd$_Literal     ; Indicate literal
                 00000001  3082                .LONG   HP$M_ALLOC              ; Constant
                       88  3086                .BYTE   Pd$_Store           ; Indicate store operation
                     000A  3087                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                       00  3089                .BYTE   0                   ; 0 HP$B_FLAGS to data
                       08  308A                .BYTE   8                   ; 8 of data field
                       81  308B                .BYTE   Pd$_End             ; Indicate end of PT-descriptor
                           308C    825 VT61:   $DS_VT61                                ;       [66]
                           308C                .SAVE   LOCAL_BLOCK
                           308C                .PSECT  $ABS$,ABS
                   00000032  0050              .=HP$A_DEPENDENT
                           0032                .IIF    NB,HP$K_VT61_LEN,       HP$K_VT61_LEN:
```

```
                        0032                    .IIF    NB,VT61$K_LEN,  VT61$K_LEN:
                    00003078                    .RESTORE
              31 36 54 56 00' 3078              .ASCIC  "VT61"  ; VT61 name
                          04  3078
                          32  307D              .BYTE   VT61$K_LEN              ; VT61$K_LEN of resultant P-table
                          08  307E              .BYTE   8                       ; Maximum unit number                   ;G:65
                        0000  307F              .WORD   ^A""    ; Two character mnemonic for QIO VT61
                          80  3081              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  3082              .Byte   Pd$_Name        ; Directive op-code
                          1B  3083              .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                    54 54 00' 3084              .Ascic  "TT"    ; Enforced device name prefix
                          02  3084
                          86  3087              .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  3088              .LONG   HP$M_ALLOC              ; Constant
                          88  308C              .BYTE   Pd$_Store       ; Indicate store operation
                        000A  308D              .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  308F              .BYTE   0                       ; 0 HP$B_FLAGS to data
                          08  3090              .BYTE   8                       ; 8 of data field
                          81  3091              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              3092    826 VT71:   $DS_VT71                               ;       [66]
                              3092                .SAVE   LOCAL_BLOCK
                              3092                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$K_VT71_LEN,       HP$K_VT71_LEN:
                              0032                .IIF    NB,VT71$K_LEN,  VT71$K_LEN:
                    00003092                    .RESTORE
              31 37 54 56 00' 3092              .ASCIC  "VT71"  ; VT71 name
                          04  3092
                          32  3097              .BYTE   VT71$K_LEN              ; VT71$K_LEN of resultant P-table
                          08  3098              .BYTE   8                       ; Maximum unit number                   ;G:65
                        0000  3099              .WORD   ^A""    ; Two character mnemonic for QIO VT71
                          80  309B              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  309C              .Byte   Pd$_Name        ; Directive op-code
                          1B  309D              .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                    54 54 00' 309E              .Ascic  "TT"    ; Enforced device name prefix
                          02  309E
                          86  30A1              .BYTE   Pd$_Literal     ; Indicate literal
                    00000001  30A2              .LONG   HP$M_ALLOC              ; Constant
                          88  30A6              .BYTE   Pd$_Store       ; Indicate store operation
                        000A  30A7              .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                          00  30A9              .BYTE   0                       ; 0 HP$B_FLAGS to data
                          08  30AA              .BYTE   8                       ; 8 of data field
                          81  30AB              .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                              30AC    827 VT100:  $DS_VT100
                              30AC                .SAVE   LOCAL_BLOCK
                              30AC                .PSECT  $ABS$,ABS
                    00000032  0050                .=HP$A_DEPENDENT
                              0032                .IIF    NB,HP$K_VT100_LEN,      HP$K_VT100_LEN:
                              0032                .IIF    NB,VT100$K_LEN, VT100$K_LEN:
                    000030AC                    .RESTORE
          30 30 31 54 56 00' 30AC               .ASCIC  "VT100" ; VT100 name
                          05  30AC
                          32  30B2              .BYTE   VT100$K_LEN             ; VT100$K_LEN of resultant P-table
                          FF  30B3              .BYTE   255                     ; Maximum unit number                   ;G:65
                        0000  30B4              .WORD   ^A""    ; Two character mnemonic for QIO VT100
                          80  30B6              .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                          8D  30B7              .Byte   Pd$_Name        ; Directive op-code
```

```
                      1B  30B8                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 30B9                .Ascic  "TT"    ; Enforced device name prefix
                      02  30B9
                      86  30BC                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  30BD                .LONG   HP$M_ALLOC                      ; Constant
                      88  30C1                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  30C2                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  30C4                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  30C5                .BYTE   8                ; 8 of data field
                      81  30C6                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          30C7    828 VT101:  $DS_VT101                                       ;       [55]
                          30C7                .SAVE   LOCAL_BLOCK
                          30C7                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT101_LEN,      HP$K_VT101_LEN:
                          0032                .IIF    NB,VT101$K_LEN, VT101$K_LEN:
                          30C7                .RESTORE
          31 30 31 54 56 00' 30C7             .ASCIC  "VT101" ; VT101 name
                      05  30C7
                      32  30CD                .BYTE   VT101$K_LEN              ; VT101$K_LEN of resultant P-table
                      FF  30CE                .BYTE   255              ; Maximum unit number                          ;G:65
                    0000  30CF                .WORD   ^A""     ; Two character mnemonic for QIO VT101
                      80  30D1                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  30D2                .Byte   Pd$_Name         ; Directive op-code
                      1B  30D3                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 30D4                .Ascic  "TT"    ; Enforced device name prefix
                      02  30D4
                      86  30D7                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  30D8                .LONG   HP$M_ALLOC                      ; Constant
                      88  30DC                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  30DD                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  30DF                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  30E0                .BYTE   8                ; 8 of data field
                      81  30E1                .BYTE   Pd$_End          ; Indicate end of PT-descriptor
                          30E2    829 VT102:  $DS_VT102
                          30E2                .SAVE   LOCAL_BLOCK
                          30E2                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT102_LEN,      HP$K_VT102_LEN:
                          0032                .IIF    NB,VT102$K_LEN, VT102$K_LEN:
                          30E2                .RESTORE
          32 30 31 54 56 00' 30E2             .ASCIC  "VT102" ; VT102 name
                      05  30E2
                      32  30E8                .BYTE   VT102$K_LEN              ; VT102$K_LEN of resultant P-table
                      FF  30E9                .BYTE   255              ; Maximum unit number                          ;G:65
                    0000  30EA                .WORD   ^A""     ; Two character mnemonic for QIO VT102
                      80  30EC                .BYTE   Pd$_Start        ; Constant to identify start of descriptor
                      8D  30ED                .Byte   Pd$_Name         ; Directive op-code
                      1B  30EE                .Byte   Ptd$M_EndDevice          ; Enforced TT codes
                54 54 00' 30EF                .Ascic  "TT"    ; Enforced device name prefix
                      02  30EF
                      86  30F2                .BYTE   Pd$_Literal      ; Indicate literal
                00000001  30F3                .LONG   HP$M_ALLOC                      ; Constant
                      88  30F7                .BYTE   Pd$_Store        ; Indicate store operation
                    000A  30F8                .WORD   HP$B_FLAGS                      ; Byte HP$B_FLAGS to data
                      00  30FA                .BYTE   0                ; 0 HP$B_FLAGS to data
                      08  30FB                .BYTE   8                ; 8 of data field
```

```
                      81  30FC                        .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          30FD    830 VT125:  $DS_VT125
                          30FD                .SAVE   LOCAL_BLOCK
                          30FD                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT125_LEN,      HP$K_VT125_LEN:
                          0032                .IIF    NB,VT125$K_LEN, VT125$K_LEN:
                      000030FD                .RESTORE
            35 32 31 54 56 00' 30FD           .ASCIC  "VT125" ; VT125 name
                      05  30FD
                      32  3103                .BYTE   VT125$K_LEN             ; VT125$K_LEN of resultant P-table
                      FF  3104                .BYTE   255             ; Maximum unit number                   ;G:65
                    0000  3105                .WORD   ^A""    ; Two character mnemonic for QIO VT125
                      80  3107                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  3108                .Byte   Pd$_Name        ; Directive op-code
                      1B  3109                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                54 54 00' 310A                .Ascic  "TT"    ; Enforced device name prefix
                      02  310A
                      86  310D                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  310E                .LONG   HP$M_ALLOC              ; Constant
                      88  3112                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  3113                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  3115                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  3116                .BYTE   8               ; 8 of data field
                      81  3117                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          3118    831 VT131:  $DS_VT131                       ;       [66]
                          3118                .SAVE   LOCAL_BLOCK
                          3118                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT131_LEN,      HP$K_VT131_LEN:
                          0032                .IIF    NB,VT131$K_LEN, VT131$K_LEN:
                      00003118                .RESTORE
            31 33 31 54 56 00' 3118           .ASCIC  "VT131" ; VT131 name
                      05  3118
                      32  311E                .BYTE   VT131$K_LEN             ; VT131$K_LEN of resultant P-table
                      08  311F                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  3120                .WORD   ^A""    ; Two character mnemonic for QIO VT131
                      80  3122                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  3123                .Byte   Pd$_Name        ; Directive op-code
                      1B  3124                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                54 54 00' 3125                .Ascic  "TT"    ; Enforced device name prefix
                      02  3125
                      86  3128                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  3129                .LONG   HP$M_ALLOC              ; Constant
                      88  312D                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  312E                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  3130                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  3131                .BYTE   8               ; 8 of data field
                      81  3132                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          3133    832 VT132:  $DS_VT132                       ;       [66]
                          3133                .SAVE   LOCAL_BLOCK
                          3133                .PSECT  $ABS$,ABS
                  00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT132_LEN,      HP$K_VT132_LEN:
                          0032                .IIF    NB,VT132$K_LEN, VT132$K_LEN:
                      00003133                .RESTORE
            32 33 31 54 56 00' 3133           .ASCIC  "VT132" ; VT132 name
```

```
                      05  3133
                      32  3139                .BYTE   VT132$K_LEN             ; VT132$K_LEN of resultant P-table
                      08  313A                .BYTE   8                       ; Maximum unit number                   ;G:65
                    0000  313B                .WORD   ^A""     ; Two character mnemonic for QIO VT132
                      80  313D                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  313E                .Byte   Pd$_Name        ; Directive op-code
                      1B  313F                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                 54 54 00' 3140                .Ascic  "TT"    ; Enforced device name prefix
                      02  3140
                      86  3143                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  3144                .LONG   HP$M_ALLOC              ; Constant
                      88  3148                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  3149                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  314B                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  314C                .BYTE   8                       ; 8 of data field
                      81  314D                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          314E    833 VT220:  $DS_VT220                               ;       [35]
                          314E                .SAVE   LOCAL_BLOCK
                          314E                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT220_LEN,      HP$K_VT220_LEN:
                          0032                .IIF    NB,VT220$K_LEN, VT220$K_LEN:
                      0000314E                .RESTORE
              30 32 32 54 56 00' 314E                .ASCIC  "VT220" ; VT220 name
                      05  314E
                      32  3154                .BYTE   VT220$K_LEN             ; VT220$K_LEN of resultant P-table
                      FF  3155                .BYTE   255                     ; Maximum unit number                   ;G:65
                    0000  3156                .WORD   ^A""     ; Two character mnemonic for QIO VT220
                      80  3158                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  3159                .Byte   Pd$_Name        ; Directive op-code
                      1B  315A                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                 54 54 00' 315B                .Ascic  "TT"    ; Enforced device name prefix
                      02  315B
                      86  315E                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  315F                .LONG   HP$M_ALLOC              ; Constant
                      88  3163                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  3164                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  3166                .BYTE   0                       ; 0 HP$B_FLAGS to data
                      08  3167                .BYTE   8                       ; 8 of data field
                      81  3168                .BYTE   Pd$_End                 ; Indicate end of PT-descriptor
                          3169    834 VT240:  $DS_VT240                               ;       [35]
                          3169                .SAVE   LOCAL_BLOCK
                          3169                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$K_VT240_LEN,      HP$K_VT240_LEN:
                          0032                .IIF    NB,VT240$K_LEN, VT240$K_LEN:
                      00003169                .RESTORE
              30 34 32 54 56 00' 3169                .ASCIC  "VT240" ; VT240 name
                      05  3169
                      32  316F                .BYTE   VT240$K_LEN             ; VT240$K_LEN of resultant P-table
                      FF  3170                .BYTE   255                     ; Maximum unit number                   ;G:65
                    0000  3171                .WORD   ^A""     ; Two character mnemonic for QIO VT240
                      80  3173                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  3174                .Byte   Pd$_Name        ; Directive op-code
                      1B  3175                .Byte   Ptd$M_EndDevice         ; Enforced TT codes
                 54 54 00' 3176                .Ascic  "TT"    ; Enforced device name prefix
                      02  3176
```

```
                      86  3179                .BYTE   Pd$_Literal     ; Indicate literal
                00000001  317A                .LONG   HP$M_ALLOC              ; Constant
                      88  317E                .BYTE   Pd$_Store       ; Indicate store operation
                    000A  317F                .WORD   HP$B_FLAGS              ; Byte HP$B_FLAGS to data
                      00  3181                .BYTE   0               ; 0 HP$B_FLAGS to data
                      08  3182                .BYTE   8               ; 8 of data field
                      81  3183                .BYTE   Pd$_End         ; Indicate end of PT-descriptor
                          3184    835 XBI:    $DS_XBI                         ;       [78]                    ;G:19
                          3184                .SAVE   LOCAL_BLOCK
                          3184                .PSECT  $ABS$,ABS
                00000032  0050                .=HP$A_DEPENDENT
                          0032                .IIF    NB,HP$B_XBI_BI_NODE,    HP$B_XBI_BI_NODE:
                00000033  0032                .IIF    NB,.BLKB,       .BLKB   1
                          0033                .IIF    NB,HP$B_XMI_NODE,       HP$B_XMI_NODE:
                00000034  0033                .IIF    NB,.BLKB,       .BLKB   1
                          0034                .IIF    NB,HP$A_BIIC,   HP$A_BIIC:
                00000038  0034                .IIF    NB,.BLKL,       .BLKL   1
                          0038                .IIF    NB,HP$K_XBI_LEN,        HP$K_XBI_LEN:
                      00003184                .RESTORE
             49 42 58 00' 3184                .ASCIC  "XBI"   ; XBI name
                      03  3184
                      38  3188                .BYTE   HP$K_XBI_LEN            ; HP$K_XBI_LEN of resultant P-table
                      08  3189                .BYTE   8               ; Maximum unit number                   ;G:65
                    0000  318A                .WORD   ^A""    ; Two character mnemonic for QIO XBI
                      80  318C                .BYTE   Pd$_Start       ; Constant to identify start of descriptor
                      8D  318D                .Byte   Pd$_Name        ; Directive op-code
                      01  318E                .Byte   PTD$M_UNIT              ; Enforced XBI codes
             49 42 58 00' 318F                .Ascic  "XBI"   ; Enforced device name prefix
                      03  318F
                      84  3193                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
20 23 20 65 64 6F 6E 20 49 4D 58 00' 3194     .ASCIC  "XMI node # (1,2,3,4,B,C,D,E) " ; XMI node # (1,2,3,4,B,C,D,E)  stri
43 2C 42 2C 34 2C 33 2C 32 2C 31 28  31A0
                20 29 45 2C 44 2C    31AC
                      1D  3194
                0000000E 00000001    31B2                .LONG   ^X1,^XE ; 1 , E limits on input
                      88  31BA                .BYTE   Pd$_Store       ; Indicate store operation
                    0033  31BB                .WORD   HP$B_XMI_NODE           ; Byte HP$B_XMI_NODE to data
                      00  31BD                .BYTE   0               ; 0 HP$B_XMI_NODE to data
                      08  31BE                .BYTE   8               ; 8 of data field
                      88  31BF                .BYTE   Pd$_Store       ; Indicate store operation
                    0018  31C0                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      13  31C2                .BYTE   19              ; 19 HP$A_DEVICE to data
                      04  31C3                .BYTE   4               ; 4 of data field
                      86  31C4                .BYTE   Pd$_Literal     ; Indicate literal
                61800000  31C5                .LONG   ^X61800000              ; Constant
                      8A  31C9                .BYTE   Pd$_Add         ; Indicate add value to field operation
                    0018  31CA                .WORD   HP$A_DEVICE             ; Byte HP$A_DEVICE to data
                      00  31CC                .BYTE   0               ; 0 HP$A_DEVICE to data
                      20  31CD                .BYTE   32              ; 32 of data field
                      84  31CE                .BYTE   Pd$_Hexadecimal ; Indicate scan hexadecimal
6D 75 4E 20 65 64 6F 4E 20 49 42 00' 31CF     .ASCIC  "BI Node Number (HEX)"  ; BI Node Number (HEX) string
      29 58 45 48 28 20 72 65 62     31DB
                      14  31CF
                0000000F 00000000    31E4                .LONG   ^X0,^XF ; 0 , F limits on input
                      88  31EC                .BYTE   Pd$_Store       ; Indicate store operation
                    0032  31ED                .WORD   HP$B_XBI_BI_NODE                ; Byte HP$B_XBI_BI_NODE to data
                      00  31EF                .BYTE   0               ; 0 HP$B_XBI_BI_NODE to data
```

```
        08  31F0          .BYTE   8               ; 8 of data field
        88  31F1          .BYTE   Pd$_Store       ; Indicate store operation
      0034  31F2          .WORD   HP$A_BIIC               ; Byte HP$A_BIIC to data
        0D  31F4          .BYTE   13              ; 13 HP$A_BIIC to data
        04  31F5          .BYTE   4               ; 4 of data field
        87  31F6          .BYTE   Pd$_Fetch       ; Indicate fetch operation
      0033  31F7          .WORD   HP$B_XMI_NODE           ; Byte HP$B_XMI_NODE to data
        00  31F9          .BYTE   0               ; 0 HP$B_XMI_NODE to data
        08  31FA          .BYTE   8               ; 8 of data field
        8A  31FB          .BYTE   Pd$_Add         ; Indicate add value to field operation
      0034  31FC          .WORD   HP$A_BIIC               ; Byte HP$A_BIIC to data
        19  31FE          .BYTE   25              ; 25 HP$A_BIIC to data
        04  31FF          .BYTE   4               ; 4 of data field
        88  3200          .BYTE   Pd$_Store       ; Indicate store operation
      0024  3201          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
        04  3203          .BYTE   4               ; 4 HP$W_VECTOR to data
        04  3204          .BYTE   4               ; 4 of data field
        86  3205          .BYTE   Pd$_Literal     ; Indicate literal
  00000001  3206          .LONG   1               ; Constant
        88  320A          .BYTE   Pd$_Store       ; Indicate store operation
      0024  320B          .WORD   HP$W_VECTOR             ; Byte HP$W_VECTOR to data
        08  320D          .BYTE   8               ; 8 HP$W_VECTOR to data
        01  320E          .BYTE   1               ; 1 of data field
        86  320F          .BYTE   Pd$_Literal     ; Indicate literal
  60000000  3210          .LONG   ^X60000000              ; Constant
        8A  3214          .BYTE   Pd$_Add         ; Indicate add value to field operation
      0034  3215          .WORD   HP$A_BIIC               ; Byte HP$A_BIIC to data
        00  3217          .BYTE   0               ; 0 HP$A_BIIC to data
        20  3218          .BYTE   32              ; 32 of data field
        87  3219          .BYTE   Pd$_Fetch       ; Indicate fetch operation
      0034  321A          .WORD   HP$A_BIIC               ; Byte HP$A_BIIC to data
        19  321C          .BYTE   25              ; 25 HP$A_BIIC to data
        07  321D          .BYTE   7               ; 7 of data field
        88  321E          .BYTE   Pd$_Store       ; Indicate store operation
      001C  321F          .WORD   HP$A_DVA                ; Byte HP$A_DVA to data
        19  3221          .BYTE   25              ; 25 HP$A_DVA to data
        07  3222          .BYTE   7               ; 7 of data field
        81  3223          .BYTE   Pd$_End         ; Indicate end of PT-descriptor
            3224  836     .END
```

```
$$_$                    = 00000003     D
AA11K                     000002B4 R   D   04
AA11K$B_BR                00000032     D
AA11K$K_LEN               00000033     D
AD11K                     00000305 R   D   04
AD11K$B_BR                00000032     D
AD11K$K_LEN               00000033     D
BIT...                  = 00000005     D
CI$B_BI_ID                00000032     D
CI$B_BR                   00000033     D
CI$B_NODE                 00000034     D
CI$B_TR                   00000032     D
CI$K_LEN                  00000041     D
CI$L_FUNC                 00000035     D
CI$L_INDEX                0000003D     D
CI$L_MAINT_ID             00000039     D
CI750                     0000048E R   D   04
CI780                     0000050D R   D   04
CIBCA                     000003F2 R   D   04
CIBCI                     00000356 R   D   04
CI_NODE                   00000594 R   D   04
CONSOLE                   000006B0 R   D   04
CONSOLE$K_LEN             00000032     D
CR11                      000006BE R   D   04
CR11$B_BR                 00000036     D
CR11$K_LEN                00000037     D
CR11$L_CSR                00000032     D
DEBNA                     00000715 R   D   04
DEBNA$B_NODE_ID           00000032     D
DEBNA$K_LEN               00000033     D
DEBNK                     00000780 R   D   04
DEBNK$B_NODE_ID           00000032     D
DEBNK$K_LEN               00000033     D
DEBNT                     000007EB R   D   04
DEBNT$B_NODE_ID           00000032     D
DEBNT$K_LEN               00000033     D
DHB32                     00000856 R   D   04
DHU11                     000008BD R   D   04
DHV11                     0000090B R   D   04
DISK                      00000959 R   D   04
DISK$K_LEN                00000032     D
DL11                      0000096C R   D   04
DL11$B_BR                 00000036     D
DL11$K_LEN                00000037     D
DL11$L_CSR                00000032     D
DMB32                     000009B9 R   D   04
DMC11                     00000A20 R   D   04
DMC11$B_BR                00000036     D
DMC11$K_LEN               00000037     D
DMC11$L_CSR               00000032     D
DMF32                     00000A6E R   D   04
DMF32$B_BR                00000034     D
DMF32$B_COMMON            00000032     D
DMF32$B_FLAGS             00000033     D
DMF32$K_LEN               00000039     D
DMF32$L_CSR               00000035     D
DMF32A                    00000AC1 R   D   04
DMF32A$B_ACTIV            00000033     D
DMF32A$B_BR               00000032     D
DMF32A$B_JUMP             00000037     D
DMF32A$B_SPEED            00000034     D
DMF32A$K_LEN              00000038     D
DMF32A$W_FLAGS            00000035     D
DMF32P                    00000BE3 R   D   04
DMF32P$B_BR               00000034     D
DMF32P$B_COMMON           00000032     D
DMF32P$B_FLAGS            00000033     D
DMF32P$K_LEN              00000039     D
DMF32P$L_CSR              00000035     D
DMF32S                    00000C5D R   D   04
DMF32S$B_BR               00000034     D
DMF32S$B_COMMON           00000032     D
DMF32S$B_FLAGS            00000033     D
DMF32S$K_LEN              00000039     D
DMF32S$L_CSR              00000035     D
DMP11                     00000CD1 R   D   04
DMP11$B_BR                00000034     D
DMP11$B_COMMON            00000032     D
DMP11$B_FLAGS             00000033     D
DMP11$K_LEN               00000039     D
DMP11$L_CSR               00000035     D
DMR11                     00000D9A R   D   04
DMR11$B_BR                00000036     D
DMR11$K_LEN               00000037     D
DMR11$L_CSR               00000032     D
DMZ32                     00000DE8 R   D   04
DR11B                     00000FAE R   D   04
DR11B$B_BR                00000036     D
DR11B$K_LEN               00000037     D
DR11B$L_CSR               00000032     D
DR11C                     00000FFC R   D   04
DR11K                     00001045 R   D   04
DR11K$B_BR                00000032     D
DR11K$K_LEN               00000033     D
DR11W                     00001096 R   D   04
DR11W$B_BR                00000036     D
DR11W$K_LEN               00000037     D
DR11W$L_CSR               00000032     D
DR750                     000010E4 R   D   04
DR750$B_BR                00000033     D
DR750$B_CONFIG            00000035     D
DR750$B_SELF              00000034     D
DR750$B_SLOT              00000032     D
DR750$B_UUT               00000036     D
DR750$K_LEN               00000037     D
DR780                     0000116F R   D   04
DR780$B_BR                00000033     D
DR780$B_CONFIG            00000035     D
DR780$B_SELF              00000034     D
DR780$B_TR                00000032     D
DR780$B_UUT               00000036     D
DR780$K_LEN               00000037     D
DRB32                     00000F46 R   D   04
DRB32$B_BR                00000033     D
```

DRB32$B_NODE_ID	00000032		D		HP$B_DHUV_BR	00000038		D
DRB32$K_LEN	00000034		D		HP$B_DMB32_NODE_ID	00000032	G	D
DS$GA_ALLOCATED	0000022C	RG	D	03	HP$B_DMF32A_ACTIV	00000033		D
DS$GA_ALLOCATES	00000014	RG	D	03	HP$B_DMF32A_BR	00000032		D
DS$GA_PTABLE	00000430	RG	D	03	HP$B_DMF32A_JUMP	00000037		D
DS$GA_PTDESC	00000000	RG	D	04	HP$B_DMF32A_SPEED	00000034		D
DS$GA_SELECTED	00000028	RG	D	03	HP$B_DMF32P_BR	00000034		D
DS$GA_SELECTS	00000000	RG	D	03	HP$B_DMF32P_COMMON	00000032		D
DUP11	00001202	R	D	04	HP$B_DMF32P_FLAGS	00000033		D
DUP11$B_BR	00000036		D		HP$B_DMF32S_BR	00000034		D
DUP11$K_LEN	00000037		D		HP$B_DMF32S_COMMON	00000032		D
DUP11$L_CSR	00000032		D		HP$B_DMF32S_FLAGS	00000033		D
DW730	00001250	R	D	04	HP$B_DMF32_BR	00000034		D
DW730$K_LEN	00000032		D		HP$B_DMF32_COMMON	00000032		D
DW750	0000127F	R	D	04	HP$B_DMF32_FLAGS	00000033		D
DW750$K_LEN	00000032		D		HP$B_DMZ32_BR	00000032		D
DW780	000012C3	R	D	04	HP$B_DMZ32_MODEM	00000039		D
DW780$B_BR	00000033		D		HP$B_DMZ32_OCTET0	00000033		D
DW780$B_TR	00000032		D		HP$B_DMZ32_OCTET1	00000034		D
DW780$K_LEN	00000034		D		HP$B_DMZ32_OCTET2	00000035		D
DWBLA	00001337	R	D	04	HP$B_DMZ32_SPEED	00000036		D
DWBLA$B_BR	00000033		D		HP$B_DR11C_BR	00000032		D
DWBLA$B_NODE_ID	00000032		D		HP$B_DRIVE	0000000B		D
DWBLA$K_LEN	00000036		D		HP$B_DX20_DRIVE	00000032	G	D
DWBLA$W_VECTOR	00000034		D		HP$B_DZ32_ACTIV	00000033		D
DWBUA	000013A7	R	D	04	HP$B_DZ32_BRLVL	00000032		D
DWBUA$B_BR	00000033		D		HP$B_DZ32_SPEED	00000034		D
DWBUA$B_NODE_ID	00000032		D		HP$B_FLAGS	0000000A		D
DWBUA$K_LEN	00000036		D		HP$B_IEU11A_BR	00000032		D
DWBUA$W_VECTOR	00000034		D		HP$B_KDB50_BR	00000036		D
DX20	0000141E	R	D	04	HP$B_KDB50_BURST	00000038		D
DX20$B_DRIVE	00000032	G	D		HP$B_KDB50_NODE_ID	00000037		D
DX20$K_LEN	00000033	G	D		HP$B_LESI_BR	00000036		D
DZ11	0000144C	R	D	04	HP$B_NBIA_COMMON	00000032		D
DZ11$B_BR	00000036		D		HP$B_NBIB_COMMON	00000032		D
DZ11$B_MTYPE	00000037		D		HP$B_PBIA_COMMON	00000032		D
DZ11$K_LEN	00000038		D		HP$B_PBR	00000037		D
DZ11$L_CSR	00000032		D		HP$B_RB730_BR	00000036		D
DZ32	000014B5	R	D	04	HP$B_RH750_BR	00000032		D
DZ32$B_ACTIV	00000033		D		HP$B_RH780_BR	00000033		D
DZ32$B_BRLVL	00000032		D		HP$B_RH780_TR	00000032		D
DZ32$B_SPEED	00000034		D		HP$B_RK611_BR	00000036		D
DZ32$W_FLAGS	00000035		D		HP$B_RL11_BR	00000036		D
DZ32$_LEN	00000037		D		HP$B_RUX50_BR	00000032		D
HP$A_BIIC	00000034		D		HP$B_RX211_BR	00000036		D
HP$A_DEPENDENT	00000032		D		HP$B_TM03_DRIVE	00000032		D
HP$A_DEVICE	00000018		D		HP$B_TM78_DRIVE	00000032		D
HP$A_DVA	0000001C		D		HP$B_TS04_BR	00000036		D
HP$A_LINK	00000020		D		HP$B_TS11_BR	00000036		D
HP$B_BI_NODE	00000033		D		HP$B_TU80_BR	00000036		D
HP$B_BI_NUM	00000038		D		HP$B_TU81_BR	00000036		D
HP$B_BR	00000037		D		HP$B_UDA50_BR	00000036		D
HP$B_CI_BR	00000033		D		HP$B_UDA50_BURST	00000037		D
HP$B_CI_NODE	00000034		D		HP$B_VS100_BR	00000036		D
HP$B_CI_TR	00000032		D		HP$B_VS125_BR	00000036		D
HP$B_CPU_ID	0000004F	G	D		HP$B_VS300_BR	00000036		D
HP$B_DHB32_NODE_ID	00000032	G	D		HP$B_XBI_BI_NODE	00000032		D

HP$B_XMI_NODE	00000033		D	HP$K_TM78_LEN	00000033		D
HP$K_CI_LEN	00000041		D	HP$K_TS04_LEN	00000037		D
HP$K_DHB32_LEN	00000033	G	D	HP$K_TS05_LEN	00000033		D
HP$K_DHUV_LEN	00000039		D	HP$K_TS11_LEN	00000037		D
HP$K_DISK_LEN	00000032		D	HP$K_TU45_LEN	00000032		D
HP$K_DMB32_LEN	00000033	G	D	HP$K_TU58_LEN	00000032		D
HP$K_DMF32A_LEN	00000038		D	HP$K_TU70_LEN	00000032	G	D
HP$K_DMF32P_LEN	00000039		D	HP$K_TU72_LEN	00000032	G	D
HP$K_DMF32S_LEN	00000039		D	HP$K_TU77_LEN	00000032		D
HP$K_DMF32_LEN	00000039		D	HP$K_TU78_LEN	00000032		D
HP$K_DMZ32_LEN	0000003A		D	HP$K_TU80_LEN	00000037		D
HP$K_DR11C_LEN	00000033		D	HP$K_TU81_LEN	00000037		D
HP$K_DX20_LEN	00000033	G	D	HP$K_UBE_LEN	00000036		D
HP$K_IEU11A_LEN	00000033		D	HP$K_UDA50_LEN	00000038		D
HP$K_KDB50_LEN	00000039		D	HP$K_VS100_LEN	00000037		D
HP$K_LA100_LEN	00000032		D	HP$K_VS125_LEN	00000037		D
HP$K_LA12_LEN	00000032		D	HP$K_VS300_LEN	00000037		D
HP$K_LA210_LEN	00000032		D	HP$K_VT100_LEN	00000032		D
HP$K_LEN	00000037		D	HP$K_VT101_LEN	00000032		D
HP$K_LESI_LEN	00000037		D	HP$K_VT102_LEN	00000032		D
HP$K_MSA_LEN	00000032		D	HP$K_VT125_LEN	00000032		D
HP$K_NBIA_LEN	00000038		D	HP$K_VT131_LEN	00000032		D
HP$K_NBIB_LEN	00000039		D	HP$K_VT132_LEN	00000032		D
HP$K_PBIA_LEN	00000038		D	HP$K_VT220_LEN	00000032		D
HP$K_RA60_LEN	00000032		D	HP$K_VT240_LEN	00000032		D
HP$K_RA70_LEN	00000032		D	HP$K_VT50_LEN	00000032		D
HP$K_RA80_LEN	00000032		D	HP$K_VT52_LEN	00000032		D
HP$K_RA81_LEN	00000032		D	HP$K_VT55_LEN	00000032		D
HP$K_RA82_LEN	00000032		D	HP$K_VT61_LEN	00000032		D
HP$K_RA90_LEN	00000032		D	HP$K_VT71_LEN	00000032		D
HP$K_RB730_LEN	00000037		D	HP$K_XBI_LEN	00000038		D
HP$K_RC25_LEN	00000032		D	HP$L_CI_FUNC	00000035		D
HP$K_RCF25_LEN	00000032		D	HP$L_CI_INDEX	0000003D		D
HP$K_RH750_LEN	00000033		D	HP$L_CI_MAINT_ID	00000039		D
HP$K_RH780_LEN	00000034		D	HP$L_DHUV_CSR	00000032		D
HP$K_RK06_LEN	00000032		D	HP$L_DMF32P_CSR	00000035		D
HP$K_RK07_LEN	00000032		D	HP$L_DMF32S_CSR	00000035		D
HP$K_RK611_LEN	00000037		D	HP$L_DMF32_CSR	00000035		D
HP$K_RL01_LEN	00000032		D	HP$L_KAA_SLOT	0000004B	G	D
HP$K_RL02_LEN	00000032		D	HP$L_KDB50_IP	00000032		D
HP$K_RL11_LEN	00000037		D	HP$L_LESI_IP	00000032		D
HP$K_RM03_LEN	00000032		D	HP$L_NBIA_CSR	00000033		D
HP$K_RM05_LEN	00000032		D	HP$L_NBIB_CSR	00000034		D
HP$K_RM80_LEN	00000032		D	HP$L_PBIA_CSR	00000033		D
HP$K_RP04_LEN	00000032		D	HP$L_RB730_CSR	00000032		D
HP$K_RP05_LEN	00000032		D	HP$L_RK611_CSR	00000032		D
HP$K_RP06_LEN	00000032		D	HP$L_RL11_CSR	00000032		D
HP$K_RP07_LEN	00000032		D	HP$L_RX211_CSR	00000032		D
HP$K_RUX50_LEN	00000033		D	HP$L_TS04_CSR	00000032		D
HP$K_RX02_LEN	00000032		D	HP$L_TS11_CSR	00000032		D
HP$K_RX211_LEN	00000037		D	HP$L_TU80_CSR	00000032		D
HP$K_SBIA_LEN	00000032		D	HP$L_TU81_CSR	00000032		D
HP$K_TAPE_LEN	00000032		D	HP$L_UBE_CSR	00000032		D
HP$K_TE16_LEN	00000032		D	HP$L_UDA50_UDAIP	00000032		D
HP$K_TK50_LEN	00000032		D	HP$L_VS100_CSR	00000032		D
HP$K_TK70_LEN	00000032		D	HP$L_VS125_CSR	00000032		D
HP$K_TM03_LEN	00000033		D	HP$L_VS300_CSR	00000032		D

```
HP$M_ALLOC                      = 00000001    D
HP$Q_DEVICE                       00000000    D
HP$T_DEVICE                       0000000C    D
HP$T_TYPE                         00000026    D
HP$W_DHUV_VEC                     00000036    D
HP$W_DMF32A_FLAGS                 00000035    D
HP$W_DMZ32_LOOP                   00000037    D
HP$W_DZ32_FLAGS                   00000035    D
HP$W_SIZE                         00000008    D
HP$W_VECTOR                       00000024    D
IEU11A                            000015A6 R  D  04
IEU11A$B_BR                       00000032    D
IEU11A$K_LEN                      00000033    D
IOC$AL_SYSUCB                     00000000 RG D  02
IOC$GL_ADPLIST                    0000000C RG D  02
IOC$GL_DEVLIST                    00000008 RG D  02
IOC$GL_DPTLIST                    00000010 RG D  02
KA$B_ACC_TYPE                     00000042  G D
KA$B_NODE_ID                      00000046  G D
KA$B_RESERVED                     00000043  G D
KA$B_SID_TYPE                     0000003D  G D
KA$K_LEN                          00000050  G D
KA$L_FLAGS                        00000032  G D
KA$L_MEM_SIZE                     00000047  G D
KA$L_SID                          0000003A  G D
KA$M_CHAR                       = 00000100  G D
KA$M_CRC                        = 00000010  G D
KA$M_D_FLOAT                    = 00000002  G D
KA$M_EDIT                       = 00000080  G D
KA$M_F_FLOAT                    = 00000001  G D
KA$M_G_FLOAT                    = 00000004  G D
KA$M_H_FLOAT                    = 00000008  G D
KA$M_PACKED                     = 00000020  G D
KA$M_PACK_MUL                   = 00000040  G D
KA$M_SB_ERR                     = 02000000  G D
KA$M_TODR                       = 00010000  G D
KA$M_WCS_LD                     = 01000000  G D
KA$V_CHAR                       = 00000008  G D
KA$V_CRC                        = 00000004  G D
KA$V_D_FLOAT                    = 00000001  G D
KA$V_EDIT                       = 00000007  G D
KA$V_F_FLOAT                    = 00000000  G D
KA$V_G_FLOAT                    = 00000002  G D
KA$V_H_FLOAT                    = 00000003  G D
KA$V_PACKED                     = 00000005  G D
KA$V_PACK_MUL                   = 00000006  G D
KA$V_SB_ERR                     = 00000019  G D
KA$V_TODR                       = 00000010  G D
KA$V_WCS_LD                     = 00000018  G D
KA$W_LATENCY                      0000003E  G D
KA$W_MEM_SIZE                     00000044  G D
KA$W_PROGRESS                     00000040  G D
KA$W_RESERVED                     00000038  G D
KA$W_WCS                          00000036  G D
KA730                             000015F1 R  D  04
KA750                             000016CC R  D  04
KA780                             00001798 R  D  04
KA785                             00001841 R  D  04
KA800                             000018EA R  D  04
KA800$B_FLOAT                     00000032    D
KA800$B_ID                        00000034    D
KA800$B_MEM                       00000033    D
KA800$K_LEN                       00000035    D
KA820                             0000195C R  D  04
KA850                             00001A8F R  D  04
KA86                              000019E7 R  D  04
KA880                             00001AC8 R  D  04
KA884                             00001B05 R  D  04
KA9CC                             00001B5A R  D  04
KDB50                             00001BC9 R  D  04
KDB50$B_BR                        00000036    D
KDB50$B_BURST                     00000038    D
KDB50$B_NODE_ID                   00000037    D
KDB50$K_LEN                       00000039    D
KDB50$L_IP                        00000032    D
KFBTA                             00001C5C R  D  04
KFBTA$B_ID                        00000032    D
KFBTA$K_LEN                       00000033    D
KMC11                             00001CC2 R  D  04
KMC11$B_BR                        00000036    D
KMC11$K_LEN                       00000037    D
KMC11$L_CSR                       00000032    D
KW11K                             00001D10 R  D  04
KW11K$B_BR                        00000032    D
KW11K$K_LEN                       00000033    D
LA100                             00001DC9 R  D  04
LA100$K_LEN                       00000032    D
LA12                              00001D61 R  D  04
LA12$K_LEN                        00000032    D
LA120                             00001DE4 R  D  04
LA120$K_LEN                       00000032    D
LA180                             00001DFF R  D  04
LA180$K_LEN                       00000032    D
LA210                             00001E1A R  D  04
LA210$K_LEN                       00000032    D
LA34                              00001D7B R  D  04
LA34$K_LEN                        00000032    D
LA36                              00001D95 R  D  04
LA36$K_LEN                        00000032    D
LA38                              00001DAF R  D  04
LA38$K_LEN                        00000032    D
LANCE                             00001E35 R  D  04
LANCE$K_LEN                       00000032    D
LESI                              00001E46 R  D  04
LESI$B_BR                         00000036    D
LESI$K_LEN                        00000037    D
LESI$L_IP                         00000032    D
LG01                              00001E92 R  D  04
LG01$K_LEN                        00000032    D
LG02                              00001EAC R  D  04
LG02$K_LEN                        00000032    D
LG03                              00001EC6 R  D  04
LG03$K_LEN                        00000032    D
LN01                              00001EE0 R  D  04
```

```
LN01$K_LEN                      00000032    D
LN03                            00001EFA R  D  04
LN03$K_LEN                      00000032    D
LP04                            00001F0F R  D  04
LP04$K_LEN                      00000032    D
LP05                            00001F29 R  D  04
LP05$K_LEN                      00000032    D
LP06                            00001F43 R  D  04
LP06$K_LEN                      00000032    D
LP07                            00001F5D R  D  04
LP07$K_LEN                      00000032    D
LP11                            00001F77 R  D  04
LP11$B_BR                       00000036    D
LP11$K_LEN                      00000037    D
LP11$L_CSR                      00000032    D
LP14                            00001FC4 R  D  04
LP14$K_LEN                      00000032    D
LP25                            00001FDE R  D  04
LP25$K_LEN                      00000032    D
LP26                            00001FF8 R  D  04
LP26$K_LEN                      00000032    D
LP27                            00002012 R  D  04
LP27$K_LEN                      00000032    D
LPA11K                          0000202C R  D  04
LPA11K$B_BR                     00000036    D
LPA11K$K_LEN                    00000037    D
LPA11K$L_CSR                    00000032    D
LUA11                           00002085 R  D  04
LUA11$B_BR                      00000032    D
LUA11$K_LEN                     00000037    D
LUA11$L_CSR                     00000033    D
MA780                           000020D3 R  D  04
MA780$B_BR                      00000033    D
MA780$B_MPM                     00000034    D
MA780$B_PORT                    00000035    D
MA780$B_TR                      00000032    D
MA780$K_LEN                     00000036    D
MAXPT                         = 00000080    D
MBE                             0000214E R  D  04
MBE$K_LEN                       00000032    D
ML11                            00002162 R  D  04
ML11$B_ARRAY                    00000032    D
ML11$B_CHIP                     00000033    D
ML11$K_LEN                      00000034    D
MS750                           000021C4 R  D  04
MS750$K_LEN                     00000032    D
MS780                           000021D5 R  D  04
MS780$B_ARRAYS                  00000033    D
MS780$B_TR                      00000032    D
MS780$K_LEN                     00000034    D
MSAAA                           00002225 R  D  04
NBIA                            00002236 R  D  04
NBIB                            000022A6 R  D  04
NI                              00002309 R  D  04
NI$B_LOOPBACK                   00000032  G D
NI$K_LEN                        00000033  G D
PBIA                            0000233F R  D  04
PCL11                           000023BF R  D  04
PCL11$B_BR                      00000036    D
PCL11$K_LEN                     00000037    D
PCL11$L_CSR                     00000032    D
PD$_ADD                       = 0000008A    D
PD$_CASE                      = 0000008C    D
PD$_COMPLEMENT                = 00000089    D
PD$_DECIMAL                   = 00000082    D
PD$_END                       = 00000081    D
PD$_EPB                       = 0000008E    D
PD$_FETCH                     = 00000087    D
PD$_HEXADECIMAL               = 00000084    D
PD$_LITERAL                   = 00000086    D
PD$_LOGICAL                   = 0000008B    D
PD$_NAME                      = 0000008D    D
PD$_OCTAL                     = 00000083    D
PD$_START                     = 00000080    D
PD$_STORE                     = 00000088    D
PD$_STRING                    = 00000085    D
PTABLE_DESCRIPTORS              000002B4 R  D  04
PTD$M_CONTROLLER              = 00000002    D
PTD$M_DEVICE                  = 00000003    D
PTD$M_ENDDEVICE               = 0000001B    D
PTD$M_INHERIT                 = 00000018    D
PTD$M_INHERIT_CON             = 00000010    D
PTD$M_INHERIT_PRE             = 00000008    D
PTD$M_NAME                    = 00000004    D
PTD$M_UNIT                    = 00000001    D
PX780                           00002408 R  D  04
PX780$B_BR                      00000033    D
PX780$B_NODE                    00000034    D
PX780$B_TR                      00000032    D
PX780$K_LEN                     00000035    D
R80                             00002471 R  D  04
R80$K_LEN                       00000032    D
RA60                            0000248A R  D  04
RA60$K_LEN                      00000032    D
RA70                            000024A4 R  D  04
RA70$K_LEN                      00000032    D
RA80                            000024BE R  D  04
RA80$K_LEN                      00000032    D
RA81                            000024D8 R  D  04
RA81$K_LEN                      00000032    D
RA82                            000024F2 R  D  04
RA82$K_LEN                      00000032    D
RA90                            0000250C R  D  04
RA90$K_LEN                      00000032    D
RB730                           00002529 R  D  04
RB730$B_BR                      00000036    D
RB730$K_LEN                     00000037    D
RB730$L_CSR                     00000032    D
RC25                            00002573 R  D  04
RC25$K_LEN                      00000032    D
RCF25                           00002558 R  D  04
RCF25$K_LEN                     00000032    D
RD52                            0000258D R  D  04
RD52$K_LEN                      00000032    D
```

```
RD53                            000025A7 R  D  04
RD53$K_LEN                      00000032    D
RD54                            000025C1 R  D  04
RD54$K_LEN                      00000032    D
RH750                           000025DB R  D  04
RH750$B_BR                      00000032    D
RH750$K_LEN                     00000033    D
RH780                           00002634 R  D  04
RH780$B_BR                      00000033    D
RH780$B_TR                      00000032    D
RH780$K_LEN                     00000034    D
RK06                            0000271A R  D  04
RK06$K_LEN                      00000032    D
RK07                            00002734 R  D  04
RK07$K_LEN                      00000032    D
RK611                           0000274E R  D  04
RK611$B_BR                      00000036    D
RK611$K_LEN                     00000037    D
RK611$L_CSR                     00000032    D
RL01                            00002699 R  D  04
RL01$K_LEN                      00000032    D
RL02                            000026B3 R  D  04
RL02$K_LEN                      00000032    D
RL11                            000026CD R  D  04
RL11$B_BR                       00000036    D
RL11$K_LEN                      00000037    D
RL11$L_CSR                      00000032    D
RM03                            0000279C R  D  04
RM03$K_LEN                      00000032    D
RM05                            000027C0 R  D  04
RM05$K_LEN                      00000032    D
RM80                            000027E4 R  D  04
RM80$K_LEN                      00000032    D
RP04                            00002808 R  D  04
RP04$K_LEN                      00000032    D
RP05                            0000282C R  D  04
RP05$K_LEN                      00000032    D
RP06                            00002850 R  D  04
RP06$K_LEN                      00000032    D
RP07                            00002874 R  D  04
RP07$K_LEN                      00000032    D
RRD50                           00002898 R  D  04
RRD50$B_BR                      00000036  G D
RRD50$K_LEN                     00000037  G D
RRD50$L_CSR                     00000032  G D
RUX50                           0000294F R  D  04
RV20                            000028F4 R  D  04
RV20$B_BR                       00000036    D
RV20$K_LEN                      00000037    D
RV20$L_CSR                      00000032    D
RX02                            00002997 R  D  04
RX02$K_LEN                      00000032    D
RX180                           000029E6 R  D  04
RX180$K_LEN                     00000032    D
RX211                           000029F7 R  D  04
RX211$B_BR                      00000036    D
RX211$K_LEN                     00000037    D
RX211$L_CSR                     00000032    D
RX31                            000029B1 R  D  04
RX31$K_LEN                      00000032    D
RX32                            000029C1 R  D  04
RX32$K_LEN                      00000032    D
RX50                            000029D1 R  D  04
RX50$K_LEN                      00000032    D
SBIA                            00002A45 R  D  04
SBIA$K_LEN                      00000032    D
SIZ...                        = 00000001    D
SLU                             00002A87 R  D  04
SLU$B_ACTIV                     00000033  G D
SLU$B_EXT                       00000032  G D
SLU$B_SPEED                     00000034  G D
SLU$K_LEN                       00000035  G D
TAPE                            00002AFC R  D  04
TAPE$K_LEN                      00000032    D
TE16                            00002B0C R  D  04
TE16$K_LEN                      00000032    D
TK50                            00002B26 R  D  04
TK70                            00002B40 R  D  04
TM03                            00002B5A R  D  04
TM03$B_DRIVE                    00000032    D
TM03$K_LEN                      00000033    D
TM78                            00002B88 R  D  04
TM78$B_DRIVE                    00000032    D
TM78$K_LEN                      00000033    D
TS04                            00002BB6 R  D  04
TS04$B_BR                       00000036    D
TS04$K_LEN                      00000037    D
TS04$L_CSR                      00000032    D
TS05                            00002C0D R  D  04
TS05$B_BR                       00000032    D
TS05$K_LEN                      00000033    D
TS11                            00002C77 R  D  04
TS11$B_BR                       00000036    D
TS11$K_LEN                      00000037    D
TS11$L_CSR                      00000032    D
TU45                            00002CCE R  D  04
TU45$K_LEN                      00000032    D
TU58                            00002CE8 R  D  04
TU58$K_LEN                      00000032    D
TU70                            00002D02 R  D  04
TU70$K_LEN                      00000032  G D
TU72                            00002D1C R  D  04
TU72$K_LEN                      00000032  G D
TU77                            00002D36 R  D  04
TU77$K_LEN                      00000032    D
TU78                            00002D50 R  D  04
TU78$K_LEN                      00000032    D
TU80                            00002D6A R  D  04
TU80$B_BR                       00000036    D
TU80$K_LEN                      00000037    D
TU81                            00002DD9 R  D  04
TU81$B_BR                       00000036    D
TU81$K_LEN                      00000037    D
TU81$L_CSR                      00000032    D
```

```
UBE                             00002E30 R  D  04
UBE$K_LEN                       00000036    D
UBE$L_CSR                       00000032    D
UDA50                           00002E6B R  D  04
UDA50$B_BR                      00000036    D
UDA50$B_BURST                   00000037    D
UDA50$K_LEN                     00000038    D
UDA50$L_UDAIP                   00000032    D
UNA11                           00002ED4 R  D  04
UNA11$B_BR                      00000032    D
UNA11$K_LEN                     00000037    D
UNA11$L_CSR                     00000033    D
VS100                           00002F22 R  D  04
VS100$B_BR                      00000036    D
VS100$K_LEN                     00000037    D
VS100$L_CSR                     00000032    D
VS125                           00002F7A R  D  04
VS125$B_BR                      00000036    D
VS125$K_LEN                     00000037    D
VS125$L_CSR                     00000032    D
VS300                           00002FD2 R  D  04
VS300$B_BR                      00000036    D
VS300$K_LEN                     00000037    D
VS300$L_CSR                     00000032    D
VT100                           000030AC R  D  04
VT100$K_LEN                     00000032    D
VT101                           000030C7 R  D  04
VT101$K_LEN                     00000032    D
VT102                           000030E2 R  D  04
VT102$K_LEN                     00000032    D
VT125                           000030FD R  D  04
VT125$K_LEN                     00000032    D
VT131                           00003118 R  D  04
VT131$K_LEN                     00000032    D
VT132                           00003133 R  D  04
VT132$K_LEN                     00000032    D
VT220                           0000314E R  D  04
VT220$K_LEN                     00000032    D
VT240                           00003169 R  D  04
VT240$K_LEN                     00000032    D
VT50                            0000302A R  D  04
VT50$K_LEN                      00000032    D
VT52                            00003044 R  D  04
VT52$K_LEN                      00000032    D
VT55                            0000305E R  D  04
VT55$K_LEN                      00000032    D
VT61                            00003078 R  D  04
VT61$K_LEN                      00000032    D
VT71                            00003092 R  D  04
VT71$K_LEN                      00000032    D
XBI                             00003184 R  D  04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000050 (80.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
WORK	00000634 (1588.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00003224 (12836.)	04 (4.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                    VALUE       DEFINITION      REFERENCES...
------                    -----       ----------      -------------
$$_$                     =00000003    757    (6)      666    (6)      667    (6)      670    (6)      673    (6)
                                                      674    (6)      675    (6)      676    (6)      681    (6)
                                                      690    (6)      701    (6)      702    (6)      718    (6)
                                                      719    (6)      754    (6)      757    (6)
AA11K                     000002B4-R   664    (6)      463    (5)
AA11K$B_BR                00000032                     664    (6)
AA11K$K_LEN               00000033                     664    (6)
AD11K                     00000305-R   665    (6)      464    (5)
AD11K$B_BR                00000032                     665    (6)
AD11K$K_LEN               00000033                     665    (6)
BIT...                   =00000005    835    (6)      664    (6)      665    (6)      666    (6)      667    (6)
                                                      668    (6)      669    (6)      670    (6)      671    (6)
                                                      672    (6)      673    (6)      674    (6)      675    (6)
                                                      676    (6)      677    (6)      678    (6)      679    (6)
                                                      680    (6)      681    (6)      682    (6)      683    (6)
                                                      684    (6)      685    (6)      686    (6)      687    (6)
                                                      688    (6)      689    (6)      690    (6)      691    (6)
                                                      692    (6)      693    (6)      694    (6)      695    (6)
                                                      696    (6)      697    (6)      698    (6)      699    (6)
                                                      700    (6)      701    (6)      702    (6)      703    (6)
                                                      704    (6)      705    (6)      706    (6)      707    (6)
                                                      708    (6)      709    (6)      710    (6)      711    (6)
                                                      712    (6)      713    (6)      714    (6)      715    (6)
                                                      716    (6)      717    (6)      718    (6)      719    (6)
                                                      720    (6)      721    (6)      722    (6)      723    (6)
                                                      724    (6)      725    (6)      726    (6)      727    (6)
                                                      728    (6)      729    (6)      730    (6)      731    (6)
                                                      732    (6)      733    (6)      734    (6)      735    (6)
                                                      736    (6)      737    (6)      738    (6)      739    (6)
                                                      740    (6)      741    (6)      742    (6)      743    (6)
                                                      744    (6)      745    (6)      746    (6)      747    (6)
                                                      748    (6)      749    (6)      750    (6)      751    (6)
                                                      752    (6)      753    (6)      754    (6)      755    (6)
                                                      756    (6)      757    (6)      758    (6)      759    (6)
                                                      760    (6)      761    (6)      762    (6)      763    (6)
                                                      764    (6)      765    (6)      766    (6)      767    (6)
                                                      768    (6)      769    (6)      770    (6)      771    (6)
                                                      772    (6)      773    (6)      774    (6)      775    (6)
                                                      776    (6)      777    (6)      778    (6)      779    (6)
                                                      780    (6)      781    (6)      782    (6)      783    (6)
                                                      784    (6)      785    (6)      786    (6)      787    (6)
                                                      788    (6)      789    (6)      790    (6)      791    (6)
                                                      792    (6)      793    (6)      794    (6)      795    (6)
                                                      796    (6)      797    (6)      798    (6)      799    (6)
                                                      800    (6)      801    (6)      802    (6)      803    (6)
                                                      804    (6)      805    (6)      806    (6)      807    (6)
                                                      808    (6)      809    (6)      810    (6)      811    (6)
                                                      812    (6)      813    (6)      814    (6)      815    (6)
                                                      816    (6)      817    (6)      818    (6)      819    (6)
                                                      820    (6)      821    (6)      822    (6)      823    (6)
```

```
                                                824    (6)     825    (6)     826    (6)     827    (6)
                                                828    (6)     829    (6)     830    (6)     831    (6)
                                                832    (6)     833    (6)     834    (6)     835    (6)
CI$B_BI_ID                      00000032        666    (6)     667    (6)
CI$B_BR                         00000033        666    (6)     667    (6)     668    (6)     669    (6)
CI$B_NODE                       00000034        666    (6)     667    (6)     668    (6)     669    (6)
                                                670    (6)
CI$B_TR                         00000032        668    (6)     669    (6)
CI$K_LEN                        00000041        666    (6)     667    (6)     668    (6)     669    (6)
                                                670    (6)
CI$L_FUNC                       00000035        666    (6)     667    (6)     668    (6)     669    (6)
                                                670    (6)
CI$L_INDEX                      0000003D        670    (6)
CI$L_MAINT_ID                   00000039        666    (6)     667    (6)     668    (6)     669    (6)
                                                670    (6)
CI750                           0000048E-R  668 (6)  468    (5)
CI780                           0000050D-R  669 (6)  467    (5)
CIBCA                           000003F2-R  667 (6)  465    (5)
CIBCI                           00000356-R  666 (6)  466    (5)
CI_NODE                         00000594-R  670 (6)  469    (5)
CONSOLE                         000006B0-R  671 (6)  470    (5)
CONSOLE$K_LEN                   00000032            671    (6)
CR11                            000006BE-R  672 (6)  471    (5)
CR11$B_BR                       00000036            672    (6)
CR11$K_LEN                      00000037            672    (6)
CR11$L_CSR                      00000032            672    (6)
DEBNA                           00000715-R  673 (6)  472    (5)
DEBNA$B_NODE_ID                 00000032            673    (6)
DEBNA$K_LEN                     00000033            673    (6)
DEBNK                           00000780-R  674 (6)  473    (5)
DEBNK$B_NODE_ID                 00000032            674    (6)
DEBNK$K_LEN                     00000033            674    (6)
DEBNT                           000007EB-R  675 (6)  474    (5)
DEBNT$B_NODE_ID                 00000032            675    (6)
DEBNT$K_LEN                     00000033            675    (6)
DHB32                           00000856-R  676 (6)  475    (5)
DHU11                           000008BD-R  677 (6)  476    (5)
DHV11                           0000090B-R  678 (6)  477    (5)
DISK                            00000959-R  679 (6)  478    (5)
DISK$K_LEN                      00000032            679    (6)
DL11                            0000096C-R  680 (6)  479    (5)
DL11$B_BR                       00000036            680    (6)
DL11$K_LEN                      00000037            680    (6)
DL11$L_CSR                      00000032            680    (6)
DMB32                           000009B9-R  681 (6)  480    (5)
DMC11                           00000A20-R  682 (6)  481    (5)
DMC11$B_BR                      00000036            682    (6)
DMC11$K_LEN                     00000037            682    (6)
DMC11$L_CSR                     00000032            682    (6)
DMF32                           00000A6E-R  683 (6)  482    (5)
DMF32$B_BR                      00000034            683    (6)
DMF32$B_COMMON                  00000032            683    (6)
DMF32$K_LEN                     00000039            683    (6)     686    (6)
DMF32$L_CSR                     00000035            683    (6)
DMF32A                          00000AC1-R  684 (6)  483    (5)
DMF32A$B_ACTIV                  00000033            684    (6)
DMF32A$B_BR                     00000032            684    (6)
```

DMF32A$B_JUMP	00000037			684	(6)
DMF32A$B_SPEED	00000034			684	(6)
DMF32A$K_LEN	00000038			684	(6)
DMF32A$W_FLAGS	00000035			684	(6)
DMF32P	00000BE3-R	685	(6)	484	(5)
DMF32P$B_BR	00000034			685	(6)
DMF32P$B_COMMON	00000032			685	(6)
DMF32P$B_FLAGS	00000033			685	(6)
DMF32P$K_LEN	00000039			685	(6)
DMF32P$L_CSR	00000035			685	(6)
DMF32S	00000C5D-R	686	(6)	485	(5)
DMF32S$B_BR	00000034			686	(6)
DMF32S$B_FLAGS	00000033			686	(6)
DMF32S$L_CSR	00000035			686	(6)
DMP11	00000CD1-R	687	(6)	486	(5)
DMP11$B_BR	00000034			687	(6)
DMP11$B_COMMON	00000032			687	(6)
DMP11$B_FLAGS	00000033			687	(6)
DMP11$K_LEN	00000039			687	(6)
DMP11$L_CSR	00000035			687	(6)
DMR11	00000D9A-R	688	(6)	487	(5)
DMR11$B_BR	00000036			688	(6)
DMR11$K_LEN	00000037			688	(6)
DMR11$L_CSR	00000032			688	(6)
DMZ32	00000DE8-R	689	(6)	488	(5)
DR11B	00000FAE-R	691	(6)	490	(5)
DR11B$B_BR	00000036			691	(6)
DR11B$K_LEN	00000037			691	(6)
DR11B$L_CSR	00000032			691	(6)
DR11C	00000FFC-R	692	(6)	491	(5)
DR11K	00001045-R	693	(6)	492	(5)
DR11K$B_BR	00000032			693	(6)
DR11K$K_LEN	00000033			693	(6)
DR11W	00001096-R	694	(6)	493	(5)
DR11W$B_BR	00000036			694	(6)
DR11W$K_LEN	00000037			694	(6)
DR11W$L_CSR	00000032			694	(6)
DR750	000010E4-R	695	(6)	494	(5)
DR750$B_BR	00000033			695	(6)
DR750$B_CONFIG	00000035			695	(6)
DR750$B_SELF	00000034			695	(6)
DR750$B_SLOT	00000032			695	(6)
DR750$B_UUT	00000036			695	(6)
DR750$K_LEN	00000037			695	(6)
DR780	0000116F-R	696	(6)	495	(5)
DR780$B_BR	00000033			696	(6)
DR780$B_CONFIG	00000035			696	(6)
DR780$B_SELF	00000034			696	(6)
DR780$B_TR	00000032			696	(6)
DR780$B_UUT	00000036			696	(6)
DR780$K_LEN	00000037			696	(6)
DRB32	00000F46-R	690	(6)	489	(5)
DRB32$B_BR	00000033			690	(6)
DRB32$B_NODE_ID	00000032			690	(6)
DRB32$K_LEN	00000034			690	(6)
DS$GA_ALLOCATED	0000022C-R	437	(4)		
DS$GA_ALLOCATES	00000014-R	425	(4)		

```
DS$GA_PTABLE                    00000430-R   449    (4)
DS$GA_PTDESC                    00000000-R   460    (5)
DS$GA_SELECTED                  00000028-R   434    (4)
DS$GA_SELECTS                   00000000-R   418    (4)
DUP11                           00001202-R   697    (6)     496    (5)
DUP11$B_BR                      00000036                    697    (6)
DUP11$K_LEN                     00000037                    697    (6)
DUP11$L_CSR                     00000032                    697    (6)
DW730                           00001250-R   698    (6)     497    (5)
DW730$K_LEN                     00000032                    698    (6)
DW750                           0000127F-R   699    (6)     498    (5)
DW750$K_LEN                     00000032                    699    (6)
DW780                           000012C3-R   700    (6)     499    (5)
DW780$B_BR                      00000033                    700    (6)
DW780$B_TR                      00000032                    700    (6)
DW780$K_LEN                     00000034                    700    (6)
DWBLA                           00001337-R   701    (6)     500    (5)
DWBLA$B_BR                      00000033                    701    (6)
DWBLA$B_NODE_ID                 00000032                    701    (6)
DWBLA$K_LEN                     00000036                    701    (6)
DWBUA                           000013A7-R   702    (6)     501    (5)
DWBUA$B_BR                      00000033                    702    (6)
DWBUA$B_NODE_ID                 00000032                    702    (6)
DWBUA$K_LEN                     00000036                    702    (6)
DX20                            0000141E-R   703    (6)     502    (5)
DX20$B_DRIVE                    00000032                    703    (6)
DX20$K_LEN                      00000033                    703    (6)
DZ11                            0000144C-R   704    (6)     503    (5)
DZ11$B_BR                       00000036                    704    (6)
DZ11$B_MTYPE                    00000037                    704    (6)
DZ11$K_LEN                      00000038                    704    (6)
DZ11$L_CSR                      00000032                    704    (6)
DZ32                            000014B5-R   705    (6)     504    (5)
DZ32$B_ACTIV                    00000033                    705    (6)
DZ32$B_BRLVL                    00000032                    705    (6)
DZ32$B_SPEED                    00000034                    705    (6)
DZ32$W_FLAGS                    00000035                    705    (6)
DZ32$_LEN                       00000037                    705    (6)
HP$A_BIIC                       00000034                    835    (6)
HP$A_DEVICE                     00000018                    664    (6)     665    (6)     666    (6)     667    (6)
                                                            668    (6)     669    (6)     670    (6)     672    (6)
                                                            673    (6)     674    (6)     675    (6)     676    (6)
                                                            677    (6)     678    (6)     680    (6)     681    (6)
                                                            682    (6)     683    (6)     684    (6)     685    (6)
                                                            686    (6)     687    (6)     688    (6)     689    (6)
                                                            690    (6)     691    (6)     692    (6)     693    (6)
                                                            694    (6)     695    (6)     696    (6)     697    (6)
                                                            698    (6)     699    (6)     700    (6)     701    (6)
                                                            702    (6)     703    (6)     704    (6)     705    (6)
                                                            706    (6)     707    (6)     708    (6)     711    (6)
                                                            717    (6)     718    (6)     719    (6)     720    (6)
                                                            721    (6)     731    (6)     741    (6)     746    (6)
                                                            747    (6)     748    (6)     749    (6)     750    (6)
                                                            752    (6)     754    (6)     755    (6)     757    (6)
                                                            758    (6)     759    (6)     767    (6)     773    (6)
                                                            774    (6)     777    (6)     780    (6)     781    (6)
                                                            782    (6)     783    (6)     784    (6)     785    (6)
```

		786	(6)	787	(6)	788	(6)	789	(6)		
		790	(6)	796	(6)	797	(6)	803	(6)		
		804	(6)	805	(6)	806	(6)	807	(6)		
		814	(6)	815	(6)	816	(6)	817	(6)		
		818	(6)	819	(6)	820	(6)	821	(6)		
		835	(6)								
HP$A_DVA	0000001C	666	(6)	667	(6)	668	(6)	669	(6)		
		670	(6)	673	(6)	674	(6)	675	(6)		
		676	(6)	681	(6)	698	(6)	699	(6)		
		700	(6)	701	(6)	702	(6)	718	(6)		
		719	(6)	754	(6)	755	(6)	757	(6)		
		773	(6)	774	(6)	797	(6)	835	(6)		
HP$B_BI_NODE	00000033	755	(6)								
HP$B_BI_NUM	00000038	755	(6)								
HP$B_CPU_ID	0000004F	716	(6)								
HP$B_DHB32_NODE_ID	00000032	676	(6)								
HP$B_DHUV_BR	00000038	677	(6)	678	(6)						
HP$B_DMB32_NODE_ID	00000032	681	(6)								
HP$B_DMZ32_BR	00000032	689	(6)								
HP$B_DMZ32_MODEM	00000039	689	(6)								
HP$B_DMZ32_OCTET0	00000033	689	(6)								
HP$B_DMZ32_OCTET1	00000034	689	(6)								
HP$B_DMZ32_OCTET2	00000035	689	(6)								
HP$B_DMZ32_SPEED	00000036	689	(6)								
HP$B_DR11C_BR	00000032	692	(6)								
HP$B_DRIVE	0000000B	670	(6)	673	(6)	674	(6)	675	(6)		
		699	(6)	700	(6)	703	(6)	719	(6)		
		749	(6)	750	(6)	773	(6)	781	(6)		
		782	(6)	783	(6)	784	(6)	785	(6)		
		786	(6)	787	(6)	797	(6)	803	(6)		
		804	(6)								
HP$B_FLAGS	0000000A	672	(6)	722	(6)	723	(6)	724	(6)		
		725	(6)	726	(6)	727	(6)	728	(6)		
		729	(6)	732	(6)	733	(6)	734	(6)		
		735	(6)	736	(6)	737	(6)	738	(6)		
		739	(6)	740	(6)	742	(6)	743	(6)		
		744	(6)	745	(6)	746	(6)	750	(6)		
		760	(6)	761	(6)	762	(6)	763	(6)		
		764	(6)	765	(6)	766	(6)	768	(6)		
		769	(6)	770	(6)	771	(6)	772	(6)		
		775	(6)	776	(6)	778	(6)	779	(6)		
		781	(6)	782	(6)	783	(6)	784	(6)		
		785	(6)	786	(6)	787	(6)	788	(6)		
		789	(6)	791	(6)	794	(6)	800	(6)		
		801	(6)	802	(6)	805	(6)	806	(6)		
		807	(6)	808	(6)	809	(6)	810	(6)		
		811	(6)	812	(6)	813	(6)	814	(6)		
		815	(6)	819	(6)	820	(6)	821	(6)		
		822	(6)	823	(6)	824	(6)	825	(6)		
		826	(6)	827	(6)	828	(6)	829	(6)		
		830	(6)	831	(6)	832	(6)	833	(6)		
		834	(6)								
HP$B_KDB50_BURST	00000038	718	(6)								
HP$B_LESI_BR	00000036	731	(6)								
HP$B_RUX50_BR	00000032	790	(6)								
HP$B_TU80_BR	00000036	814	(6)								
HP$B_UDA50_BR	00000036	817	(6)								

```
HP$B_UDA50_BURST                00000037        817    (6)
HP$B_XBI_BI_NODE                00000032        835    (6)
HP$B_XMI_NODE                   00000033        835    (6)
HP$K_DHB32_LEN                  00000033        676    (6)
HP$K_DHUV_LEN                   00000039        677    (6)     678    (6)
HP$K_DMB32_LEN                  00000033        681    (6)
HP$K_DMZ32_LEN                  0000003A        689    (6)
HP$K_DR11C_LEN                  00000033        692    (6)
HP$K_KDB50_LEN                  00000039        718    (6)
HP$K_LEN                        00000037        714    (6)     715    (6)     716    (6)
HP$K_LESI_LEN                   00000037        731    (6)
HP$K_MSA_LEN                    00000032        753    (6)
HP$K_NBIA_LEN                   00000038        754    (6)
HP$K_NBIB_LEN                   00000039        755    (6)
HP$K_PBIA_LEN                   00000038        757    (6)
HP$K_RUX50_LEN                  00000033        790    (6)
HP$K_TK50_LEN                   00000032        801    (6)
HP$K_TK70_LEN                   00000032        802    (6)
HP$K_UDA50_LEN                  00000038        817    (6)
HP$K_XBI_LEN                    00000038        835    (6)
HP$L_DHUV_CSR                   00000032        677    (6)     678    (6)
HP$L_KAA_SLOT                   0000004B        714    (6)     715    (6)     716    (6)
HP$L_KDB50_IP                   00000032        718    (6)
HP$L_LESI_IP                    00000032        731    (6)
HP$L_TU80_CSR                   00000032        814    (6)
HP$L_UDA50_UDAIP                00000032        817    (6)
HP$M_ALLOC                     =00000001        672    (6)     722    (6)     723    (6)     724    (6)
                                                725    (6)     726    (6)     727    (6)     728    (6)
                                                729    (6)     732    (6)     733    (6)     734    (6)
                                                735    (6)     736    (6)     737    (6)     738    (6)
                                                739    (6)     740    (6)     742    (6)     743    (6)
                                                744    (6)     745    (6)     746    (6)     750    (6)
                                                760    (6)     761    (6)     762    (6)     763    (6)
                                                764    (6)     765    (6)     766    (6)     768    (6)
                                                769    (6)     770    (6)     771    (6)     772    (6)
                                                775    (6)     776    (6)     778    (6)     779    (6)
                                                781    (6)     782    (6)     783    (6)     784    (6)
                                                785    (6)     786    (6)     787    (6)     788    (6)
                                                789    (6)     791    (6)     794    (6)     800    (6)
                                                801    (6)     802    (6)     805    (6)     806    (6)
                                                807    (6)     808    (6)     809    (6)     810    (6)
                                                811    (6)     812    (6)     813    (6)     814    (6)
                                                815    (6)     819    (6)     820    (6)     821    (6)
                                                822    (6)     823    (6)     824    (6)     825    (6)
                                                826    (6)     827    (6)     828    (6)     829    (6)
                                                830    (6)     831    (6)     832    (6)     833    (6)
                                                834    (6)

HP$W_DHUV_VEC                   00000036        677    (6)     678    (6)
HP$W_DMZ32_LOOP                 00000037        689    (6)
HP$W_VECTOR                     00000024        664    (6)     665    (6)     666    (6)     667    (6)
                                                668    (6)     669    (6)     672    (6)     673    (6)
                                                674    (6)     675    (6)     676    (6)     680    (6)
                                                681    (6)     682    (6)     683    (6)     684    (6)
                                                685    (6)     686    (6)     687    (6)     688    (6)
                                                689    (6)     690    (6)     691    (6)     692    (6)
                                                693    (6)     694    (6)     695    (6)     696    (6)
                                                697    (6)     698    (6)     699    (6)     700    (6)
```

				701 (6) 702 (6) 704 (6) 705 (6)
				706 (6) 718 (6) 720 (6) 721 (6)
				731 (6) 741 (6) 746 (6) 747 (6)
				748 (6) 754 (6) 757 (6) 758 (6)
				759 (6) 767 (6) 773 (6) 774 (6)
				777 (6) 780 (6) 788 (6) 789 (6)
				790 (6) 796 (6) 797 (6) 805 (6)
				806 (6) 807 (6) 814 (6) 815 (6)
				816 (6) 817 (6) 818 (6) 819 (6)
				820 (6) 821 (6) 835 (6)
IEU11A	000015A6-R	706	(6)	505 (5)
IEU11A$B_BR	00000032			706 (6)
IEU11A$K_LEN	00000033			706 (6)
IOC$AL_SYSUCB	00000000-R	401	(3)	
IOC$GL_ADPLIST	0000000C-R	407	(3)	
IOC$GL_DEVLIST	00000008-R	405	(3)	
IOC$GL_DPTLIST	00000010-R	409	(3)	410 (3)
KA$B_ACC_TYPE	00000042			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6)
KA$B_NODE_ID	00000046			712 (6) 717 (6)
KA$K_LEN	00000050			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 717 (6)
KA$L_FLAGS	00000032			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6) 717 (6)
KA$L_MEM_SIZE	00000047			712 (6)
KA$M_CHAR	=00000100			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6)
KA$M_CRC	=00000010			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6)
KA$M_D_FLOAT	=00000002			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6)
KA$M_EDIT	=00000080			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6)
KA$M_F_FLOAT	=00000001			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6)
KA$M_G_FLOAT	=00000004			707 (6) 712 (6) 714 (6) 715 (6)
				716 (6)
KA$M_H_FLOAT	=00000008			707 (6) 712 (6) 714 (6) 715 (6)
				716 (6)
KA$M_PACKED	=00000020			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6)
KA$M_PACK_MUL	=00000040			707 (6) 708 (6) 709 (6) 710 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6)
KA$M_TODR	=00010000			709 (6) 710 (6) 712 (6) 713 (6)
				717 (6)
KA$V_D_FLOAT	=00000001			717 (6)
KA$V_F_FLOAT	=00000000			717 (6)
KA$V_G_FLOAT	=00000002			708 (6) 709 (6) 710 (6) 713 (6)
				717 (6)

```
KA$V_H_FLOAT                    =00000003                               708     (6)     709     (6)     710     (6)     713     (6)
KA$V_SBE_RR                     =00000019                               707     (6)
KA$V_TODR                       =00000010                               707     (6)     708     (6)
KA$V_WCS_LD                     =00000018                               707     (6)
KA$W_LATENCY                     0000003E                               707     (6)     708     (6)     709     (6)     710     (6)
                                                                        712     (6)     713     (6)     714     (6)     715     (6)
                                                                        716     (6)     717     (6)
KA$W_MEM_SIZE                    00000044                               707     (6)
KA$W_PROGRESS                    00000040                               707     (6)     708     (6)     709     (6)     710     (6)
                                                                        712     (6)     713     (6)     714     (6)     715     (6)
                                                                        716     (6)     717     (6)
KA$W_WCS                         00000036                               707     (6)     708     (6)     709     (6)     710     (6)
                                                                        712     (6)     713     (6)
KA730                            000015F1-R     707     (6)             506     (5)
KA750                            000016CC-R     708     (6)             507     (5)
KA780                            00001798-R     709     (6)             508     (5)
KA785                            00001841-R     710     (6)             509     (5)
KA800                            000018EA-R     711     (6)             510     (5)
KA800$B_FLOAT                    00000032                               711     (6)
KA800$B_ID                       00000034                               711     (6)
KA800$B_MEM                      00000033                               711     (6)
KA800$K_LEN                      00000035                               711     (6)
KA820                            0000195C-R     712     (6)             511     (5)
KA850                            00001A8F-R     714     (6)             513     (5)
KA86                             000019E7-R     713     (6)             512     (5)
KA880                            00001AC8-R     715     (6)             514     (5)
KA884                            00001B05-R     716     (6)             515     (5)
KA9CC                            00001B5A-R     717     (6)             516     (5)
KDB50                            00001BC9-R     718     (6)             518     (5)
KDB50$B_BR                       00000036                               718     (6)
KDB50$B_NODE_ID                  00000037                               718     (6)
KFBTA                            00001C5C-R     719     (6)             517     (5)
KFBTA$B_ID                       00000032                               719     (6)
KFBTA$K_LEN                      00000033                               719     (6)
KMC11                            00001CC2-R     720     (6)             519     (5)
KMC11$B_BR                       00000036                               720     (6)
KMC11$K_LEN                      00000037                               720     (6)
KMC11$L_CSR                      00000032                               720     (6)
KW11K                            00001D10-R     721     (6)             520     (5)
KW11K$B_BR                       00000032                               721     (6)
KW11K$K_LEN                      00000033                               721     (6)
LA100                            00001DC9-R     726     (6)             525     (5)
LA100$K_LEN                      00000032                               726     (6)
LA12                             00001D61-R     722     (6)             521     (5)
LA12$K_LEN                       00000032                               722     (6)
LA120                            00001DE4-R     727     (6)             526     (5)
LA120$K_LEN                      00000032                               727     (6)
LA180                            00001DFF-R     728     (6)             527     (5)
LA180$K_LEN                      00000032                               728     (6)
LA210                            00001E1A-R     729     (6)             528     (5)
LA210$K_LEN                      00000032                               729     (6)
LA34                             00001D7B-R     723     (6)             522     (5)
LA34$K_LEN                       00000032                               723     (6)
LA36                             00001D95-R     724     (6)             523     (5)
LA36$K_LEN                       00000032                               724     (6)
LA38                             00001DAF-R     725     (6)             524     (5)
LA38$K_LEN                       00000032                               725     (6)
```

```
LANCE                           00001E35-R   730     (6)      529     (5)
LANCE$K_LEN                     00000032                      730     (6)
LESI                            00001E46-R   731     (6)      530     (5)
LG01                            00001E92-R   732     (6)      531     (5)
LG01$K_LEN                      00000032                      732     (6)
LG02                            00001EAC-R   733     (6)      532     (5)
LG02$K_LEN                      00000032                      733     (6)
LG03                            00001EC6-R   734     (6)      533     (5)
LG03$K_LEN                      00000032                      734     (6)
LN01                            00001EE0-R   735     (6)      534     (5)
LN01$K_LEN                      00000032                      735     (6)
LN03                            00001EFA-R   736     (6)      535     (5)
LN03$K_LEN                      00000032                      736     (6)
LP04                            00001F0F-R   737     (6)      536     (5)
LP04$K_LEN                      00000032                      737     (6)
LP05                            00001F29-R   738     (6)      537     (5)
LP05$K_LEN                      00000032                      738     (6)
LP06                            00001F43-R   739     (6)      538     (5)
LP06$K_LEN                      00000032                      739     (6)
LP07                            00001F5D-R   740     (6)      539     (5)
LP07$K_LEN                      00000032                      740     (6)
LP11                            00001F77-R   741     (6)      540     (5)
LP11$B_BR                       00000036                      741     (6)
LP11$K_LEN                      00000037                      741     (6)
LP11$L_CSR                      00000032                      741     (6)
LP14                            00001FC4-R   742     (6)      541     (5)
LP14$K_LEN                      00000032                      742     (6)
LP25                            00001FDE-R   743     (6)      542     (5)
LP25$K_LEN                      00000032                      743     (6)
LP26                            00001FF8-R   744     (6)      543     (5)
LP26$K_LEN                      00000032                      744     (6)
LP27                            00002012-R   745     (6)      544     (5)
LP27$K_LEN                      00000032                      745     (6)
LPA11K                          0000202C-R   746     (6)      545     (5)
LPA11K$B_BR                     00000036                      746     (6)
LPA11K$K_LEN                    00000037                      746     (6)
LPA11K$L_CSR                    00000032                      746     (6)
LUA11                           00002085-R   747     (6)      546     (5)
LUA11$B_BR                      00000032                      747     (6)
LUA11$K_LEN                     00000037                      747     (6)
LUA11$L_CSR                     00000033                      747     (6)
MA780                           000020D3-R   748     (6)      547     (5)
MA780$B_BR                      00000033                      748     (6)
MA780$B_MPM                     00000034                      748     (6)
MA780$B_PORT                    00000035                      748     (6)
MA780$B_TR                      00000032                      748     (6)
MA780$K_LEN                     00000036                      748     (6)
MAXPT                          =00000080     417     (4)      419     (4)     426     (4)     435     (4)     436     (4)
                                                              438     (4)     439     (4)     450     (4)     451     (4)
MBE                             0000214E-R   749     (6)      548     (5)
MBE$K_LEN                       00000032                      749     (6)
ML11                            00002162-R   750     (6)      549     (5)
ML11$B_ARRAY                    00000032                      750     (6)
ML11$B_CHIP                     00000033                      750     (6)
ML11$K_LEN                      00000034                      750     (6)
MS750                           000021C4-R   751     (6)      550     (5)
MS750$K_LEN                     00000032                      751     (6)
```

```
MS780                           000021D5-R  752  (6)    551  (5)
MS780$B_ARRAYS                  00000033               752  (6)
MS780$B_TR                      00000032               752  (6)
MS780$K_LEN                     00000034               752  (6)
MSAAA                           00002225-R  753  (6)    552  (5)
NBIA                            00002236-R  754  (6)    553  (5)
NBIB                            000022A6-R  755  (6)    554  (5)
NI                              00002309-R  756  (6)    555  (5)
NI$B_LOOPBACK                   00000032               756  (6)
NI$K_LEN                        00000033               756  (6)
PBIA                            0000233F-R  757  (6)    556  (5)
PCL11                           000023BF-R  758  (6)    557  (5)
PCL11$B_BR                      00000036               758  (6)
PCL11$K_LEN                     00000037               758  (6)
PCL11$L_CSR                     00000032               758  (6)
PD$_ADD                        =0000008A    835  (6)    670  (6)   699  (6)   717  (6)   797  (6)
                                                       835  (6)
PD$_CASE                       =0000008C    835  (6)    666  (6)   667  (6)   670  (6)   673  (6)
                                                       674  (6)   675  (6)   676  (6)   681  (6)
                                                       690  (6)   701  (6)   702  (6)   718  (6)
                                                       719  (6)   754  (6)   757  (6)
PD$_COMPLEMENT                 =00000089    835  (6)    699  (6)
PD$_DECIMAL                    =00000082    835  (6)    664  (6)   665  (6)   666  (6)   667  (6)
                                                       668  (6)   669  (6)   670  (6)   672  (6)
                                                       677  (6)   678  (6)   680  (6)   682  (6)
                                                       683  (6)   684  (6)   685  (6)   686  (6)
                                                       687  (6)   688  (6)   689  (6)   690  (6)
                                                       691  (6)   692  (6)   693  (6)   694  (6)
                                                       695  (6)   696  (6)   697  (6)   700  (6)
                                                       702  (6)   703  (6)   704  (6)   706  (6)
                                                       707  (6)   708  (6)   709  (6)   710  (6)
                                                       711  (6)   712  (6)   713  (6)   716  (6)
                                                       720  (6)   721  (6)   731  (6)   741  (6)
                                                       746  (6)   747  (6)   748  (6)   750  (6)
                                                       752  (6)   754  (6)   755  (6)   757  (6)
                                                       758  (6)   759  (6)   773  (6)   774  (6)
                                                       777  (6)   780  (6)   788  (6)   789  (6)
                                                       790  (6)   796  (6)   803  (6)   804  (6)
                                                       805  (6)   806  (6)   807  (6)   814  (6)
                                                       815  (6)   817  (6)   818  (6)   819  (6)
                                                       820  (6)   821  (6)
PD$_END                        =00000081    835  (6)    664  (6)   665  (6)   666  (6)   667  (6)
                                                       668  (6)   669  (6)   670  (6)   671  (6)
                                                       672  (6)   673  (6)   674  (6)   675  (6)
                                                       676  (6)   677  (6)   678  (6)   679  (6)
                                                       680  (6)   681  (6)   682  (6)   683  (6)
                                                       684  (6)   685  (6)   686  (6)   687  (6)
                                                       688  (6)   689  (6)   690  (6)   691  (6)
                                                       692  (6)   693  (6)   694  (6)   695  (6)
                                                       696  (6)   697  (6)   698  (6)   699  (6)
                                                       700  (6)   701  (6)   702  (6)   703  (6)
                                                       704  (6)   705  (6)   706  (6)   707  (6)
                                                       708  (6)   709  (6)   710  (6)   711  (6)
                                                       712  (6)   713  (6)   714  (6)   715  (6)
                                                       716  (6)   717  (6)   718  (6)   719  (6)
                                                       720  (6)   721  (6)   722  (6)   723  (6)
                                                       724  (6)   725  (6)   726  (6)   727  (6)
```

```
                                                    728  (6)    729  (6)    730  (6)    731  (6)
                                                    732  (6)    733  (6)    734  (6)    735  (6)
                                                    736  (6)    737  (6)    738  (6)    739  (6)
                                                    740  (6)    741  (6)    742  (6)    743  (6)
                                                    744  (6)    745  (6)    746  (6)    747  (6)
                                                    748  (6)    749  (6)    750  (6)    751  (6)
                                                    752  (6)    753  (6)    754  (6)    755  (6)
                                                    756  (6)    757  (6)    758  (6)    759  (6)
                                                    760  (6)    761  (6)    762  (6)    763  (6)
                                                    764  (6)    765  (6)    766  (6)    767  (6)
                                                    768  (6)    769  (6)    770  (6)    771  (6)
                                                    772  (6)    773  (6)    774  (6)    775  (6)
                                                    776  (6)    777  (6)    778  (6)    779  (6)
                                                    780  (6)    781  (6)    782  (6)    783  (6)
                                                    784  (6)    785  (6)    786  (6)    787  (6)
                                                    788  (6)    789  (6)    790  (6)    791  (6)
                                                    792  (6)    793  (6)    794  (6)    795  (6)
                                                    796  (6)    797  (6)    798  (6)    799  (6)
                                                    800  (6)    801  (6)    802  (6)    803  (6)
                                                    804  (6)    805  (6)    806  (6)    807  (6)
                                                    808  (6)    809  (6)    810  (6)    811  (6)
                                                    812  (6)    813  (6)    814  (6)    815  (6)
                                                    816  (6)    817  (6)    818  (6)    819  (6)
                                                    820  (6)    821  (6)    822  (6)    823  (6)
                                                    824  (6)    825  (6)    826  (6)    827  (6)
                                                    828  (6)    829  (6)    830  (6)    831  (6)
                                                    832  (6)    833  (6)    834  (6)    835  (6)
PD$_EPB             =0000008E      835  (6)         679  (6)    766  (6)
PD$_FETCH           =00000087      835  (6)         666  (6)    667  (6)    668  (6)    669  (6)
                                                    670  (6)    673  (6)    674  (6)    675  (6)
                                                    676  (6)    681  (6)    690  (6)    699  (6)
                                                    700  (6)    701  (6)    702  (6)    718  (6)
                                                    719  (6)    749  (6)    750  (6)    754  (6)
                                                    755  (6)    757  (6)    773  (6)    774  (6)
                                                    781  (6)    782  (6)    783  (6)    784  (6)
                                                    785  (6)    786  (6)    787  (6)    797  (6)
                                                    835  (6)
PD$_HEXADECIMAL     =00000084      835  (6)         666  (6)    667  (6)    673  (6)    674  (6)
                                                    675  (6)    676  (6)    681  (6)    690  (6)
                                                    701  (6)    702  (6)    707  (6)    708  (6)
                                                    709  (6)    710  (6)    711  (6)    712  (6)
                                                    713  (6)    717  (6)    718  (6)    719  (6)
                                                    755  (6)    835  (6)
PD$_LITERAL         =00000086      835  (6)         666  (6)    667  (6)    668  (6)    669  (6)
                                                    670  (6)    672  (6)    673  (6)    674  (6)
                                                    675  (6)    676  (6)    681  (6)    683  (6)
                                                    685  (6)    687  (6)    689  (6)    695  (6)
                                                    696  (6)    698  (6)    699  (6)    700  (6)
                                                    701  (6)    702  (6)    707  (6)    708  (6)
                                                    709  (6)    710  (6)    711  (6)    712  (6)
                                                    713  (6)    714  (6)    715  (6)    716  (6)
                                                    717  (6)    718  (6)    719  (6)    722  (6)
                                                    723  (6)    724  (6)    725  (6)    726  (6)
                                                    727  (6)    728  (6)    729  (6)    732  (6)
                                                    733  (6)    734  (6)    735  (6)    736  (6)
                                                    737  (6)    738  (6)    739  (6)    740  (6)
                                                    742  (6)    743  (6)    744  (6)    745  (6)
```

				746 (6) 748 (6) 750 (6) 752 (6)
				754 (6) 757 (6) 759 (6) 760 (6)
				761 (6) 762 (6) 763 (6) 764 (6)
				765 (6) 766 (6) 767 (6) 768 (6)
				769 (6) 770 (6) 771 (6) 772 (6)
				773 (6) 774 (6) 775 (6) 776 (6)
				778 (6) 779 (6) 781 (6) 782 (6)
				783 (6) 784 (6) 785 (6) 786 (6)
				787 (6) 788 (6) 789 (6) 791 (6)
				794 (6) 797 (6) 798 (6) 800 (6)
				801 (6) 802 (6) 805 (6) 806 (6)
				807 (6) 808 (6) 809 (6) 810 (6)
				811 (6) 812 (6) 813 (6) 814 (6)
				815 (6) 819 (6) 820 (6) 821 (6)
				822 (6) 823 (6) 824 (6) 825 (6)
				826 (6) 827 (6) 828 (6) 829 (6)
				830 (6) 831 (6) 832 (6) 833 (6)
				834 (6) 835 (6)
PD$_LOGICAL	=0000008B	835	(6)	684 (6) 689 (6) 695 (6) 696 (6)
				707 (6) 708 (6) 709 (6) 710 (6)
				713 (6) 715 (6) 716 (6) 717 (6)
				756 (6) 798 (6)
PD$_NAME	=0000008D	835	(6)	664 (6) 665 (6) 666 (6) 667 (6)
				668 (6) 669 (6) 670 (6) 672 (6)
				673 (6) 674 (6) 675 (6) 677 (6)
				678 (6) 679 (6) 680 (6) 682 (6)
				684 (6) 685 (6) 686 (6) 687 (6)
				688 (6) 689 (6) 690 (6) 691 (6)
				692 (6) 693 (6) 694 (6) 695 (6)
				696 (6) 697 (6) 698 (6) 699 (6)
				700 (6) 703 (6) 704 (6) 705 (6)
				706 (6) 707 (6) 708 (6) 709 (6)
				710 (6) 711 (6) 712 (6) 713 (6)
				714 (6) 715 (6) 716 (6) 718 (6)
				719 (6) 720 (6) 721 (6) 722 (6)
				723 (6) 724 (6) 725 (6) 726 (6)
				727 (6) 728 (6) 729 (6) 730 (6)
				731 (6) 732 (6) 733 (6) 734 (6)
				735 (6) 737 (6) 738 (6) 739 (6)
				740 (6) 741 (6) 742 (6) 743 (6)
				744 (6) 745 (6) 746 (6) 747 (6)
				748 (6) 750 (6) 751 (6) 752 (6)
				753 (6) 754 (6) 755 (6) 756 (6)
				757 (6) 759 (6) 760 (6) 761 (6)
				762 (6) 763 (6) 764 (6) 765 (6)
				766 (6) 767 (6) 768 (6) 769 (6)
				770 (6) 771 (6) 772 (6) 773 (6)
				774 (6) 775 (6) 776 (6) 777 (6)
				778 (6) 779 (6) 780 (6) 781 (6)
				782 (6) 783 (6) 784 (6) 785 (6)
				786 (6) 787 (6) 788 (6) 789 (6)
				790 (6) 791 (6) 792 (6) 793 (6)
				795 (6) 796 (6) 797 (6) 798 (6)
				799 (6) 800 (6) 801 (6) 802 (6)
				803 (6) 804 (6) 805 (6) 806 (6)
				807 (6) 808 (6) 809 (6) 810 (6)
				811 (6) 812 (6) 813 (6) 814 (6)

				815 (6) 816 (6) 817 (6) 818 (6)
				819 (6) 820 (6) 821 (6) 822 (6)
				823 (6) 824 (6) 825 (6) 826 (6)
				827 (6) 828 (6) 829 (6) 830 (6)
				831 (6) 832 (6) 833 (6) 834 (6)
				835 (6)
PD$_OCTAL	=00000083	835	(6)	664 (6) 665 (6) 672 (6) 677 (6)
				678 (6) 680 (6) 682 (6) 683 (6)
				684 (6) 685 (6) 686 (6) 687 (6)
				688 (6) 689 (6) 691 (6) 692 (6)
				693 (6) 694 (6) 697 (6) 704 (6)
				705 (6) 706 (6) 720 (6) 721 (6)
				731 (6) 741 (6) 746 (6) 747 (6)
				758 (6) 777 (6) 780 (6) 788 (6)
				789 (6) 790 (6) 796 (6) 805 (6)
				806 (6) 807 (6) 814 (6) 815 (6)
				816 (6) 817 (6) 818 (6) 819 (6)
				820 (6) 821 (6)
PD$_START	=00000080	835	(6)	664 (6) 665 (6) 666 (6) 667 (6)
				668 (6) 669 (6) 670 (6) 671 (6)
				672 (6) 673 (6) 674 (6) 675 (6)
				676 (6) 677 (6) 678 (6) 679 (6)
				680 (6) 681 (6) 682 (6) 683 (6)
				684 (6) 685 (6) 686 (6) 687 (6)
				688 (6) 689 (6) 690 (6) 691 (6)
				692 (6) 693 (6) 694 (6) 695 (6)
				696 (6) 697 (6) 698 (6) 699 (6)
				700 (6) 701 (6) 702 (6) 703 (6)
				704 (6) 705 (6) 706 (6) 707 (6)
				708 (6) 709 (6) 710 (6) 711 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6) 717 (6) 718 (6) 719 (6)
				720 (6) 721 (6) 722 (6) 723 (6)
				724 (6) 725 (6) 726 (6) 727 (6)
				728 (6) 729 (6) 730 (6) 731 (6)
				732 (6) 733 (6) 734 (6) 735 (6)
				736 (6) 737 (6) 738 (6) 739 (6)
				740 (6) 741 (6) 742 (6) 743 (6)
				744 (6) 745 (6) 746 (6) 747 (6)
				748 (6) 749 (6) 750 (6) 751 (6)
				752 (6) 753 (6) 754 (6) 755 (6)
				756 (6) 757 (6) 758 (6) 759 (6)
				760 (6) 761 (6) 762 (6) 763 (6)
				764 (6) 765 (6) 766 (6) 767 (6)
				768 (6) 769 (6) 770 (6) 771 (6)
				772 (6) 773 (6) 774 (6) 775 (6)
				776 (6) 777 (6) 778 (6) 779 (6)
				780 (6) 781 (6) 782 (6) 783 (6)
				784 (6) 785 (6) 786 (6) 787 (6)
				788 (6) 789 (6) 790 (6) 791 (6)
				792 (6) 793 (6) 794 (6) 795 (6)
				796 (6) 797 (6) 798 (6) 799 (6)
				800 (6) 801 (6) 802 (6) 803 (6)
				804 (6) 805 (6) 806 (6) 807 (6)
				808 (6) 809 (6) 810 (6) 811 (6)
				812 (6) 813 (6) 814 (6) 815 (6)
				816 (6) 817 (6) 818 (6) 819 (6)

```
                                                                  820   (6)   821   (6)   822   (6)   823   (6)
                                                                  824   (6)   825   (6)   826   (6)   827   (6)
                                                                  828   (6)   829   (6)   830   (6)   831   (6)
                                                                  832   (6)   833   (6)   834   (6)   835   (6)
PD$_STORE                     =00000088     835    (6)            664   (6)   665   (6)   666   (6)   667   (6)
                                                                  668   (6)   669   (6)   670   (6)   672   (6)
                                                                  673   (6)   674   (6)   675   (6)   676   (6)
                                                                  677   (6)   678   (6)   680   (6)   681   (6)
                                                                  682   (6)   683   (6)   684   (6)   685   (6)
                                                                  686   (6)   687   (6)   688   (6)   689   (6)
                                                                  690   (6)   691   (6)   692   (6)   693   (6)
                                                                  694   (6)   695   (6)   696   (6)   697   (6)
                                                                  698   (6)   699   (6)   700   (6)   701   (6)
                                                                  702   (6)   703   (6)   704   (6)   705   (6)
                                                                  706   (6)   707   (6)   708   (6)   709   (6)
                                                                  710   (6)   711   (6)   712   (6)   713   (6)
                                                                  714   (6)   715   (6)   716   (6)   717   (6)
                                                                  718   (6)   719   (6)   720   (6)   721   (6)
                                                                  722   (6)   723   (6)   724   (6)   725   (6)
                                                                  726   (6)   727   (6)   728   (6)   729   (6)
                                                                  731   (6)   732   (6)   733   (6)   734   (6)
                                                                  735   (6)   736   (6)   737   (6)   738   (6)
                                                                  739   (6)   740   (6)   741   (6)   742   (6)
                                                                  743   (6)   744   (6)   745   (6)   746   (6)
                                                                  747   (6)   748   (6)   749   (6)   750   (6)
                                                                  752   (6)   754   (6)   755   (6)   756   (6)
                                                                  757   (6)   758   (6)   759   (6)   760   (6)
                                                                  761   (6)   762   (6)   763   (6)   764   (6)
                                                                  765   (6)   766   (6)   767   (6)   768   (6)
                                                                  769   (6)   770   (6)   771   (6)   772   (6)
                                                                  773   (6)   774   (6)   775   (6)   776   (6)
                                                                  777   (6)   778   (6)   779   (6)   780   (6)
                                                                  781   (6)   782   (6)   783   (6)   784   (6)
                                                                  785   (6)   786   (6)   787   (6)   788   (6)
                                                                  789   (6)   790   (6)   791   (6)   794   (6)
                                                                  796   (6)   797   (6)   798   (6)   800   (6)
                                                                  801   (6)   802   (6)   803   (6)   804   (6)
                                                                  805   (6)   806   (6)   807   (6)   808   (6)
                                                                  809   (6)   810   (6)   811   (6)   812   (6)
                                                                  813   (6)   814   (6)   815   (6)   816   (6)
                                                                  817   (6)   818   (6)   819   (6)   820   (6)
                                                                  821   (6)   822   (6)   823   (6)   824   (6)
                                                                  825   (6)   826   (6)   827   (6)   828   (6)
                                                                  829   (6)   830   (6)   831   (6)   832   (6)
                                                                  833   (6)   834   (6)   835   (6)
PD$_STRING                    =00000085     835    (6)            670   (6)   684   (6)   685   (6)   686   (6)
                                                                  687   (6)   689   (6)   695   (6)   696   (6)
                                                                  704   (6)   750   (6)   798   (6)
PTABLE_DESCRIPTORS             000002B4-R   663    (6)
PTD$M_CONTROLLER              =00000002     835    (6)            664   (6)   665   (6)   666   (6)   667   (6)
                                                                  668   (6)   669   (6)   670   (6)   671   (6)
                                                                  672   (6)   673   (6)   674   (6)   675   (6)
                                                                  676   (6)   677   (6)   678   (6)   679   (6)
                                                                  680   (6)   681   (6)   682   (6)   683   (6)
                                                                  684   (6)   685   (6)   686   (6)   687   (6)
                                                                  688   (6)   689   (6)   690   (6)   691   (6)
                                                                  692   (6)   693   (6)   694   (6)   695   (6)
```

				696 (6) 697 (6) 698 (6) 699 (6)
				700 (6) 701 (6) 702 (6) 703 (6)
				704 (6) 705 (6) 706 (6) 707 (6)
				708 (6) 709 (6) 710 (6) 711 (6)
				712 (6) 713 (6) 714 (6) 715 (6)
				716 (6) 717 (6) 718 (6) 719 (6)
				720 (6) 721 (6) 722 (6) 723 (6)
				724 (6) 725 (6) 726 (6) 727 (6)
				728 (6) 729 (6) 730 (6) 731 (6)
				732 (6) 733 (6) 734 (6) 735 (6)
				736 (6) 737 (6) 738 (6) 739 (6)
				740 (6) 741 (6) 742 (6) 743 (6)
				744 (6) 745 (6) 746 (6) 747 (6)
				748 (6) 749 (6) 750 (6) 751 (6)
				752 (6) 753 (6) 754 (6) 755 (6)
				756 (6) 757 (6) 758 (6) 759 (6)
				760 (6) 761 (6) 762 (6) 763 (6)
				764 (6) 765 (6) 766 (6) 767 (6)
				768 (6) 769 (6) 770 (6) 771 (6)
				772 (6) 773 (6) 774 (6) 775 (6)
				776 (6) 777 (6) 778 (6) 779 (6)
				780 (6) 781 (6) 782 (6) 783 (6)
				784 (6) 785 (6) 786 (6) 787 (6)
				788 (6) 789 (6) 790 (6) 791 (6)
				792 (6) 793 (6) 794 (6) 795 (6)
				796 (6) 797 (6) 798 (6) 799 (6)
				800 (6) 801 (6) 802 (6) 803 (6)
				804 (6) 805 (6) 806 (6) 807 (6)
				808 (6) 809 (6) 810 (6) 811 (6)
				812 (6) 813 (6) 814 (6) 815 (6)
				816 (6) 817 (6) 818 (6) 819 (6)
				820 (6) 821 (6) 822 (6) 823 (6)
				824 (6) 825 (6) 826 (6) 827 (6)
				828 (6) 829 (6) 830 (6) 831 (6)
				832 (6) 833 (6) 834 (6) 835 (6)
PTD$M_DEVICE	=00000003	835	(6)	664 (6) 665 (6) 666 (6) 667 (6)
				668 (6) 669 (6) 672 (6) 679 (6)
				682 (6) 686 (6) 687 (6) 688 (6)
				691 (6) 693 (6) 694 (6) 695 (6)
				696 (6) 697 (6) 711 (6) 720 (6)
				721 (6) 730 (6) 746 (6) 747 (6)
				750 (6) 759 (6) 781 (6) 782 (6)
				783 (6) 784 (6) 785 (6) 786 (6)
				787 (6) 788 (6) 789 (6) 798 (6)
				799 (6) 801 (6) 802 (6) 805 (6)
				806 (6) 807 (6) 809 (6) 814 (6)
				815 (6) 816 (6) 818 (6)
PTD$M_ENDDEVICE	=0000001B	835	(6)	722 (6) 723 (6) 724 (6) 725 (6)
				726 (6) 727 (6) 728 (6) 729 (6)
				732 (6) 733 (6) 734 (6) 735 (6)
				737 (6) 738 (6) 739 (6) 740 (6)
				742 (6) 743 (6) 744 (6) 745 (6)
				760 (6) 762 (6) 763 (6) 764 (6)
				765 (6) 766 (6) 768 (6) 769 (6)
				770 (6) 771 (6) 772 (6) 775 (6)
				776 (6) 778 (6) 779 (6) 791 (6)
				792 (6) 793 (6) 795 (6) 800 (6)

```
                                                        808    (6)    810    (6)    811    (6)    812    (6)
                                                        813    (6)    819    (6)    820    (6)    821    (6)
                                                        822    (6)    823    (6)    824    (6)    825    (6)
                                                        826    (6)    827    (6)    828    (6)    829    (6)
                                                        830    (6)    831    (6)    832    (6)    833    (6)
                                                        834    (6)
PTD$M_INHERIT                  =00000018     835    (6)  664    (6)    665    (6)    666    (6)    667    (6)
                                                        668    (6)    669    (6)    670    (6)    671    (6)
                                                        672    (6)    673    (6)    674    (6)    675    (6)
                                                        676    (6)    677    (6)    678    (6)    679    (6)
                                                        680    (6)    681    (6)    682    (6)    683    (6)
                                                        684    (6)    685    (6)    686    (6)    687    (6)
                                                        688    (6)    689    (6)    690    (6)    691    (6)
                                                        692    (6)    693    (6)    694    (6)    695    (6)
                                                        696    (6)    697    (6)    698    (6)    699    (6)
                                                        700    (6)    701    (6)    702    (6)    703    (6)
                                                        704    (6)    705    (6)    706    (6)    707    (6)
                                                        708    (6)    709    (6)    710    (6)    711    (6)
                                                        712    (6)    713    (6)    714    (6)    715    (6)
                                                        716    (6)    717    (6)    718    (6)    719    (6)
                                                        720    (6)    721    (6)    722    (6)    723    (6)
                                                        724    (6)    725    (6)    726    (6)    727    (6)
                                                        728    (6)    729    (6)    730    (6)    731    (6)
                                                        732    (6)    733    (6)    734    (6)    735    (6)
                                                        736    (6)    737    (6)    738    (6)    739    (6)
                                                        740    (6)    741    (6)    742    (6)    743    (6)
                                                        744    (6)    745    (6)    746    (6)    747    (6)
                                                        748    (6)    749    (6)    750    (6)    751    (6)
                                                        752    (6)    753    (6)    754    (6)    755    (6)
                                                        756    (6)    757    (6)    758    (6)    759    (6)
                                                        760    (6)    761    (6)    762    (6)    763    (6)
                                                        764    (6)    765    (6)    766    (6)    767    (6)
                                                        768    (6)    769    (6)    770    (6)    771    (6)
                                                        772    (6)    773    (6)    774    (6)    775    (6)
                                                        776    (6)    777    (6)    778    (6)    779    (6)
                                                        780    (6)    781    (6)    782    (6)    783    (6)
                                                        784    (6)    785    (6)    786    (6)    787    (6)
                                                        788    (6)    789    (6)    790    (6)    791    (6)
                                                        792    (6)    793    (6)    794    (6)    795    (6)
                                                        796    (6)    797    (6)    798    (6)    799    (6)
                                                        800    (6)    801    (6)    802    (6)    803    (6)
                                                        804    (6)    805    (6)    806    (6)    807    (6)
                                                        808    (6)    809    (6)    810    (6)    811    (6)
                                                        812    (6)    813    (6)    814    (6)    815    (6)
                                                        816    (6)    817    (6)    818    (6)    819    (6)
                                                        820    (6)    821    (6)    822    (6)    823    (6)
                                                        824    (6)    825    (6)    826    (6)    827    (6)
                                                        828    (6)    829    (6)    830    (6)    831    (6)
                                                        832    (6)    833    (6)    834    (6)    835    (6)
PTD$M_INHERITED                =00000008     835    (6)
PTD$M_INHERIT_CON              =00000010     835    (6)  664    (6)    665    (6)    666    (6)    667    (6)
                                                        668    (6)    669    (6)    670    (6)    671    (6)
                                                        672    (6)    673    (6)    674    (6)    675    (6)
                                                        676    (6)    677    (6)    678    (6)    679    (6)
                                                        680    (6)    681    (6)    682    (6)    683    (6)
                                                        684    (6)    685    (6)    686    (6)    687    (6)
                                                        688    (6)    689    (6)    690    (6)    691    (6)
```

					692	(6)	693	(6)	694	(6)	695	(6)			
					696	(6)	697	(6)	698	(6)	699	(6)			
					700	(6)	701	(6)	702	(6)	703	(6)			
					704	(6)	705	(6)	706	(6)	707	(6)			
					708	(6)	709	(6)	710	(6)	711	(6)			
					712	(6)	713	(6)	714	(6)	715	(6)			
					716	(6)	717	(6)	718	(6)	719	(6)			
					720	(6)	721	(6)	722	(6)	723	(6)			
					724	(6)	725	(6)	726	(6)	727	(6)			
					728	(6)	729	(6)	730	(6)	731	(6)			
					732	(6)	733	(6)	734	(6)	735	(6)			
					736	(6)	737	(6)	738	(6)	739	(6)			
					740	(6)	741	(6)	742	(6)	743	(6)			
					744	(6)	745	(6)	746	(6)	747	(6)			
					748	(6)	749	(6)	750	(6)	751	(6)			
					752	(6)	753	(6)	754	(6)	755	(6)			
					756	(6)	757	(6)	758	(6)	759	(6)			
					760	(6)	761	(6)	762	(6)	763	(6)			
					764	(6)	765	(6)	766	(6)	767	(6)			
					768	(6)	769	(6)	770	(6)	771	(6)			
					772	(6)	773	(6)	774	(6)	775	(6)			
					776	(6)	777	(6)	778	(6)	779	(6)			
					780	(6)	781	(6)	782	(6)	783	(6)			
					784	(6)	785	(6)	786	(6)	787	(6)			
					788	(6)	789	(6)	790	(6)	791	(6)			
					792	(6)	793	(6)	794	(6)	795	(6)			
					796	(6)	797	(6)	798	(6)	799	(6)			
					800	(6)	801	(6)	802	(6)	803	(6)			
					804	(6)	805	(6)	806	(6)	807	(6)			
					808	(6)	809	(6)	810	(6)	811	(6)			
					812	(6)	813	(6)	814	(6)	815	(6)			
					816	(6)	817	(6)	818	(6)	819	(6)			
					820	(6)	821	(6)	822	(6)	823	(6)			
					824	(6)	825	(6)	826	(6)	827	(6)			
					828	(6)	829	(6)	830	(6)	831	(6)			
					832	(6)	833	(6)	834	(6)	835	(6)			
PTD$M_INHERIT_PRE	=00000008	835	(6)		664	(6)	665	(6)	666	(6)	667	(6)			
					668	(6)	669	(6)	670	(6)	671	(6)			
					672	(6)	673	(6)	674	(6)	675	(6)			
					676	(6)	677	(6)	678	(6)	679	(6)			
					680	(6)	681	(6)	682	(6)	683	(6)			
					684	(6)	685	(6)	686	(6)	687	(6)			
					688	(6)	689	(6)	690	(6)	691	(6)			
					692	(6)	693	(6)	694	(6)	695	(6)			
					696	(6)	697	(6)	698	(6)	699	(6)			
					700	(6)	701	(6)	702	(6)	703	(6)			
					704	(6)	705	(6)	706	(6)	707	(6)			
					708	(6)	709	(6)	710	(6)	711	(6)			
					712	(6)	713	(6)	714	(6)	715	(6)			
					716	(6)	717	(6)	718	(6)	719	(6)			
					720	(6)	721	(6)	722	(6)	723	(6)			
					724	(6)	725	(6)	726	(6)	727	(6)			
					728	(6)	729	(6)	730	(6)	731	(6)			
					732	(6)	733	(6)	734	(6)	735	(6)			
					736	(6)	737	(6)	738	(6)	739	(6)			
					740	(6)	741	(6)	742	(6)	743	(6)			
					744	(6)	745	(6)	746	(6)	747	(6)			

```
                                                748 (6)  749 (6)  750 (6)  751 (6)
                                                752 (6)  753 (6)  754 (6)  755 (6)
                                                756 (6)  757 (6)  758 (6)  759 (6)
                                                760 (6)  761 (6)  762 (6)  763 (6)
                                                764 (6)  765 (6)  766 (6)  767 (6)
                                                768 (6)  769 (6)  770 (6)  771 (6)
                                                772 (6)  773 (6)  774 (6)  775 (6)
                                                776 (6)  777 (6)  778 (6)  779 (6)
                                                780 (6)  781 (6)  782 (6)  783 (6)
                                                784 (6)  785 (6)  786 (6)  787 (6)
                                                788 (6)  789 (6)  790 (6)  791 (6)
                                                792 (6)  793 (6)  794 (6)  795 (6)
                                                796 (6)  797 (6)  798 (6)  799 (6)
                                                800 (6)  801 (6)  802 (6)  803 (6)
                                                804 (6)  805 (6)  806 (6)  807 (6)
                                                808 (6)  809 (6)  810 (6)  811 (6)
                                                812 (6)  813 (6)  814 (6)  815 (6)
                                                816 (6)  817 (6)  818 (6)  819 (6)
                                                820 (6)  821 (6)  822 (6)  823 (6)
                                                824 (6)  825 (6)  826 (6)  827 (6)
                                                828 (6)  829 (6)  830 (6)  831 (6)
                                                832 (6)  833 (6)  834 (6)  835 (6)
PTD$M_NAME          =00000004   835 (6)         670 (6)  712 (6)  756 (6)
PTD$M_UNIT          =00000001   835 (6)         664 (6)  665 (6)  666 (6)  667 (6)
                                                668 (6)  669 (6)  670 (6)  671 (6)
                                                672 (6)  673 (6)  674 (6)  675 (6)
                                                676 (6)  677 (6)  678 (6)  679 (6)
                                                680 (6)  681 (6)  682 (6)  683 (6)
                                                684 (6)  685 (6)  686 (6)  687 (6)
                                                688 (6)  689 (6)  690 (6)  691 (6)
                                                692 (6)  693 (6)  694 (6)  695 (6)
                                                696 (6)  697 (6)  698 (6)  699 (6)
                                                700 (6)  701 (6)  702 (6)  703 (6)
                                                704 (6)  705 (6)  706 (6)  707 (6)
                                                708 (6)  709 (6)  710 (6)  711 (6)
                                                712 (6)  713 (6)  714 (6)  715 (6)
                                                716 (6)  717 (6)  718 (6)  719 (6)
                                                720 (6)  721 (6)  722 (6)  723 (6)
                                                724 (6)  725 (6)  726 (6)  727 (6)
                                                728 (6)  729 (6)  730 (6)  731 (6)
                                                732 (6)  733 (6)  734 (6)  735 (6)
                                                736 (6)  737 (6)  738 (6)  739 (6)
                                                740 (6)  741 (6)  742 (6)  743 (6)
                                                744 (6)  745 (6)  746 (6)  747 (6)
                                                748 (6)  749 (6)  750 (6)  751 (6)
                                                752 (6)  753 (6)  754 (6)  755 (6)
                                                756 (6)  757 (6)  758 (6)  759 (6)
                                                760 (6)  761 (6)  762 (6)  763 (6)
                                                764 (6)  765 (6)  766 (6)  767 (6)
                                                768 (6)  769 (6)  770 (6)  771 (6)
                                                772 (6)  773 (6)  774 (6)  775 (6)
                                                776 (6)  777 (6)  778 (6)  779 (6)
                                                780 (6)  781 (6)  782 (6)  783 (6)
                                                784 (6)  785 (6)  786 (6)  787 (6)
                                                788 (6)  789 (6)  790 (6)  791 (6)
                                                792 (6)  793 (6)  794 (6)  795 (6)
                                                796 (6)  797 (6)  798 (6)  799 (6)
```

```
                                                                  800    (6)    801    (6)    802    (6)    803    (6)
                                                                  804    (6)    805    (6)    806    (6)    807    (6)
                                                                  808    (6)    809    (6)    810    (6)    811    (6)
                                                                  812    (6)    813    (6)    814    (6)    815    (6)
                                                                  816    (6)    817    (6)    818    (6)    819    (6)
                                                                  820    (6)    821    (6)    822    (6)    823    (6)
                                                                  824    (6)    825    (6)    826    (6)    827    (6)
                                                                  828    (6)    829    (6)    830    (6)    831    (6)
                                                                  832    (6)    833    (6)    834    (6)    835    (6)
PTD$V_CONTROLLER                =00000001     835    (6)
PTD$V_INHERIT_CON               =00000004     835    (6)
PTD$V_INHERIT_PRE               =00000003     835    (6)
PTD$V_NAME                      =00000002     835    (6)
PTD$V_UNIT                      =00000000     835    (6)
PX780                            00002408-R   759    (6)    558    (5)
PX780$B_BR                       00000033                   759    (6)
PX780$B_NODE                     00000034                   759    (6)
PX780$B_TR                       00000032                   759    (6)
PX780$K_LEN                      00000035                   759    (6)
R80                              00002471-R   760    (6)    559    (5)
R80$K_LEN                        00000032                   760    (6)
RA60                             0000248A-R   761    (6)    560    (5)
RA60$K_LEN                       00000032                   761    (6)
RA70                             000024A4-R   762    (6)    561    (5)
RA70$K_LEN                       00000032                   762    (6)
RA80                             000024BE-R   763    (6)    562    (5)
RA80$K_LEN                       00000032                   763    (6)
RA81                             000024D8-R   764    (6)    563    (5)
RA81$K_LEN                       00000032                   764    (6)
RA82                             000024F2-R   765    (6)    565    (5)
RA82$K_LEN                       00000032                   765    (6)
RA90                             0000250C-R   766    (6)    564    (5)
RA90$K_LEN                       00000032                   766    (6)
RB730                            00002529-R   767    (6)    566    (5)
RB730$B_BR                       00000036                   767    (6)
RB730$K_LEN                      00000037                   767    (6)
RC25                             00002573-R   769    (6)    568    (5)
RC25$K_LEN                       00000032                   769    (6)
RCF25                            00002558-R   768    (6)    567    (5)
RCF25$K_LEN                      00000032                   768    (6)
RD52                             0000258D-R   770    (6)    569    (5)
RD52$K_LEN                       00000032                   770    (6)
RD53                             000025A7-R   771    (6)    570    (5)
RD53$K_LEN                       00000032                   771    (6)
RD54                             000025C1-R   772    (6)    571    (5)
RD54$K_LEN                       00000032                   772    (6)
RH750                            000025DB-R   773    (6)    572    (5)
RH750$B_BR                       00000032                   773    (6)
RH750$K_LEN                      00000033                   773    (6)
RH780                            00002634-R   774    (6)    573    (5)
RH780$B_BR                       00000033                   774    (6)
RH780$B_TR                       00000032                   774    (6)
RH780$K_LEN                      00000034                   774    (6)
RK06                             0000271A-R   778    (6)    577    (5)
RK06$K_LEN                       00000032                   778    (6)
RK07                             00002734-R   779    (6)    578    (5)
RK07$K_LEN                       00000032                   779    (6)
```

```
RK611                           0000274E-R  780  (6)    579  (5)
RK611$B_BR                      00000036             780  (6)
RK611$K_LEN                     00000037             780  (6)
RK611$L_CSR                     00000032             780  (6)
RL01                            00002699-R  775  (6)    574  (5)
RL01$K_LEN                      00000032             775  (6)
RL02                            000026B3-R  776  (6)    575  (5)
RL02$K_LEN                      00000032             776  (6)
RL11                            000026CD-R  777  (6)    576  (5)
RL11$B_BR                       00000036             777  (6)
RL11$K_LEN                      00000037             777  (6)
RL11$L_CSR                      00000032             777  (6)
RM03                            0000279C-R  781  (6)    580  (5)
RM03$K_LEN                      00000032             781  (6)
RM05                            000027C0-R  782  (6)    581  (5)
RM05$K_LEN                      00000032             782  (6)
RM80                            000027E4-R  783  (6)    582  (5)
RM80$K_LEN                      00000032             783  (6)
RP04                            00002808-R  784  (6)    584  (5)
RP04$K_LEN                      00000032             784  (6)
RP05                            0000282C-R  785  (6)    585  (5)
RP05$K_LEN                      00000032             785  (6)
RP06                            00002850-R  786  (6)    586  (5)
RP06$K_LEN                      00000032             786  (6)
RP07                            00002874-R  787  (6)    587  (5)
RP07$K_LEN                      00000032             787  (6)
RRD50                           00002898-R  788  (6)    583  (5)
RRD50$B_BR                      00000036             788  (6)
RRD50$K_LEN                     00000037             788  (6)
RRD50$L_CSR                     00000032             788  (6)
RUX50                           0000294F-R  790  (6)    589  (5)
RV20                            000028F4-R  789  (6)    588  (5)
RV20$B_BR                       00000036             789  (6)
RV20$K_LEN                      00000037             789  (6)
RV20$L_CSR                      00000032             789  (6)
RX02                            00002997-R  791  (6)    590  (5)
RX02$K_LEN                      00000032             791  (6)
RX180                           000029E6-R  795  (6)    594  (5)
RX180$K_LEN                     00000032             795  (6)
RX211                           000029F7-R  796  (6)    595  (5)
RX211$B_BR                      00000036             796  (6)
RX211$K_LEN                     00000037             796  (6)
RX211$L_CSR                     00000032             796  (6)
RX31                            000029B1-R  792  (6)    591  (5)
RX31$K_LEN                      00000032             792  (6)
RX32                            000029C1-R  793  (6)    592  (5)
RX32$K_LEN                      00000032             793  (6)
RX50                            000029D1-R  794  (6)    593  (5)
RX50$K_LEN                      00000032             794  (6)
SBIA                            00002A45-R  797  (6)    596  (5)
SBIA$K_LEN                      00000032             797  (6)
SIZ...                         =00000001    835  (6)    664  (6)    665  (6)    666  (6)    667  (6)
                                                        668  (6)    669  (6)    670  (6)    671  (6)
                                                        672  (6)    673  (6)    674  (6)    675  (6)
                                                        676  (6)    677  (6)    678  (6)    679  (6)
                                                        680  (6)    681  (6)    682  (6)    683  (6)
                                                        684  (6)    685  (6)    686  (6)    687  (6)
```

					688	(6)	689	(6)	690	(6)	691	(6)			
					692	(6)	693	(6)	694	(6)	695	(6)			
					696	(6)	697	(6)	698	(6)	699	(6)			
					700	(6)	701	(6)	702	(6)	703	(6)			
					704	(6)	705	(6)	706	(6)	707	(6)			
					708	(6)	709	(6)	710	(6)	711	(6)			
					712	(6)	713	(6)	714	(6)	715	(6)			
					716	(6)	717	(6)	718	(6)	719	(6)			
					720	(6)	721	(6)	722	(6)	723	(6)			
					724	(6)	725	(6)	726	(6)	727	(6)			
					728	(6)	729	(6)	730	(6)	731	(6)			
					732	(6)	733	(6)	734	(6)	735	(6)			
					736	(6)	737	(6)	738	(6)	739	(6)			
					740	(6)	741	(6)	742	(6)	743	(6)			
					744	(6)	745	(6)	746	(6)	747	(6)			
					748	(6)	749	(6)	750	(6)	751	(6)			
					752	(6)	753	(6)	754	(6)	755	(6)			
					756	(6)	757	(6)	758	(6)	759	(6)			
					760	(6)	761	(6)	762	(6)	763	(6)			
					764	(6)	765	(6)	766	(6)	767	(6)			
					768	(6)	769	(6)	770	(6)	771	(6)			
					772	(6)	773	(6)	774	(6)	775	(6)			
					776	(6)	777	(6)	778	(6)	779	(6)			
					780	(6)	781	(6)	782	(6)	783	(6)			
					784	(6)	785	(6)	786	(6)	787	(6)			
					788	(6)	789	(6)	790	(6)	791	(6)			
					792	(6)	793	(6)	794	(6)	795	(6)			
					796	(6)	797	(6)	798	(6)	799	(6)			
					800	(6)	801	(6)	802	(6)	803	(6)			
					804	(6)	805	(6)	806	(6)	807	(6)			
					808	(6)	809	(6)	810	(6)	811	(6)			
					812	(6)	813	(6)	814	(6)	815	(6)			
					816	(6)	817	(6)	818	(6)	819	(6)			
					820	(6)	821	(6)	822	(6)	823	(6)			
					824	(6)	825	(6)	826	(6)	827	(6)			
					828	(6)	829	(6)	830	(6)	831	(6)			
					832	(6)	833	(6)	834	(6)	835	(6)			
SLU		00002A87-R	798	(6)	597	(5)									
SLU$B_ACTIV		00000033			798	(6)									
SLU$B_EXT		00000032			798	(6)									
SLU$B_SPEED		00000034			798	(6)									
SLU$K_LEN		00000035			798	(6)									
TAPE		00002AFC-R	799	(6)	598	(5)									
TAPE$K_LEN		00000032			799	(6)									
TE16		00002B0C-R	800	(6)	599	(5)									
TE16$K_LEN		00000032			800	(6)									
TK50		00002B26-R	801	(6)	600	(5)									
TK70		00002B40-R	802	(6)	601	(5)									
TM03		00002B5A-R	803	(6)	602	(5)									
TM03$B_DRIVE		00000032			803	(6)									
TM03$K_LEN		00000033			803	(6)									
TM78		00002B88-R	804	(6)	603	(5)									
TM78$B_DRIVE		00000032			804	(6)									
TM78$K_LEN		00000033			804	(6)									
TS04		00002BB6-R	805	(6)	604	(5)									
TS04$B_BR		00000036			805	(6)									
TS04$K_LEN		00000037			805	(6)									

TS04$L_CSR	00000032			805	(6)
TS05	00002C0D-R	806	(6)	605	(5)
TS05$B_BR	00000032			806	(6)
TS05$K_LEN	00000033			806	(6)
TS11	00002C77-R	807	(6)	606	(5)
TS11$B_BR	00000036			807	(6)
TS11$K_LEN	00000037			807	(6)
TS11$L_CSR	00000032			807	(6)
TU45	00002CCE-R	808	(6)	607	(5)
TU45$K_LEN	00000032			808	(6)
TU58	00002CE8-R	809	(6)	608	(5)
TU58$K_LEN	00000032			809	(6)
TU70	00002D02-R	810	(6)	609	(5)
TU70$K_LEN	00000032			810	(6)
TU72	00002D1C-R	811	(6)	610	(5)
TU72$K_LEN	00000032			811	(6)
TU77	00002D36-R	812	(6)	611	(5)
TU77$K_LEN	00000032			812	(6)
TU78	00002D50-R	813	(6)	612	(5)
TU78$K_LEN	00000032			813	(6)
TU80	00002D6A-R	814	(6)	613	(5)
TU80$K_LEN	00000037			814	(6)
TU81	00002DD9-R	815	(6)	614	(5)
TU81$B_BR	00000036			815	(6)
TU81$K_LEN	00000037			815	(6)
TU81$L_CSR	00000032			815	(6)
UBE	00002E30-R	816	(6)	615	(5)
UBE$K_LEN	00000036			816	(6)
UBE$L_CSR	00000032			816	(6)
UDA50	00002E6B-R	817	(6)	616	(5)
UNA11	00002ED4-R	818	(6)	617	(5)
UNA11$B_BR	00000032			818	(6)
UNA11$K_LEN	00000037			818	(6)
UNA11$L_CSR	00000033			818	(6)
VS100	00002F22-R	819	(6)	618	(5)
VS100$B_BR	00000036			819	(6)
VS100$K_LEN	00000037			819	(6)
VS100$L_CSR	00000032			819	(6)
VS125	00002F7A-R	820	(6)	619	(5)
VS125$B_BR	00000036			820	(6)
VS125$K_LEN	00000037			820	(6)
VS125$L_CSR	00000032			820	(6)
VS300	00002FD2-R	821	(6)	620	(5)
VS300$B_BR	00000036			821	(6)
VS300$K_LEN	00000037			821	(6)
VS300$L_CSR	00000032			821	(6)
VT100	000030AC-R	827	(6)	626	(5)
VT100$K_LEN	00000032			827	(6)
VT101	000030C7-R	828	(6)	627	(5)
VT101$K_LEN	00000032			828	(6)
VT102	000030E2-R	829	(6)	628	(5)
VT102$K_LEN	00000032			829	(6)
VT125	000030FD-R	830	(6)	629	(5)
VT125$K_LEN	00000032			830	(6)
VT131	00003118-R	831	(6)	630	(5)
VT131$K_LEN	00000032			831	(6)
VT132	00003133-R	832	(6)	631	(5)

VT132$K_LEN	00000032			832	(6)
VT220	0000314E-R	833	(6)	632	(5)
VT220$K_LEN	00000032			833	(6)
VT240	00003169-R	834	(6)	633	(5)
VT240$K_LEN	00000032			834	(6)
VT50	0000302A-R	822	(6)	621	(5)
VT50$K_LEN	00000032			822	(6)
VT52	00003044-R	823	(6)	622	(5)
VT52$K_LEN	00000032			823	(6)
VT55	0000305E-R	824	(6)	623	(5)
VT55$K_LEN	00000032			824	(6)
VT61	00003078-R	825	(6)	624	(5)
VT61$K_LEN	00000032			825	(6)
VT71	00003092-R	826	(6)	625	(5)
VT71$K_LEN	00000032			826	(6)
XBI	00003184-R	835	(6)	634	(5)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEF	1	835 (6)	
$DEFINI	1	390 (2)	390 (2) 645 (6) 646 (6) 647 (6) 664 (6)
			665 (6) 671 (6) 672 (6) 673 (6) 674 (6)
			675 (6) 676 (6) 679 (6) 680 (6) 681 (6)
			682 (6) 683 (6) 684 (6) 685 (6) 686 (6)
			687 (6) 688 (6) 689 (6) 690 (6) 691 (6)
			692 (6) 693 (6) 694 (6) 695 (6) 696 (6)
			697 (6) 698 (6) 699 (6) 700 (6) 701 (6)
			702 (6) 703 (6) 704 (6) 705 (6) 706 (6)
			711 (6) 718 (6) 719 (6) 720 (6) 721 (6)
			722 (6) 723 (6) 724 (6) 725 (6) 726 (6)
			727 (6) 728 (6) 729 (6) 730 (6) 731 (6)
			732 (6) 733 (6) 734 (6) 735 (6) 736 (6)
			737 (6) 738 (6) 739 (6) 740 (6) 741 (6)
			742 (6) 743 (6) 744 (6) 745 (6) 746 (6)
			747 (6) 748 (6) 749 (6) 750 (6) 751 (6)
			752 (6) 753 (6) 754 (6) 755 (6) 756 (6)
			757 (6) 758 (6) 759 (6) 760 (6) 761 (6)
			762 (6) 763 (6) 764 (6) 765 (6) 766 (6)
			767 (6) 768 (6) 769 (6) 770 (6) 771 (6)
			772 (6) 773 (6) 774 (6) 775 (6) 776 (6)
			777 (6) 778 (6) 779 (6) 780 (6) 781 (6)
			782 (6) 783 (6) 784 (6) 785 (6) 786 (6)
			787 (6) 788 (6) 789 (6) 790 (6) 791 (6)
			792 (6) 793 (6) 794 (6) 795 (6) 796 (6)
			797 (6) 798 (6) 799 (6) 800 (6) 801 (6)
			802 (6) 803 (6) 804 (6) 805 (6) 806 (6)
			807 (6) 808 (6) 809 (6) 810 (6) 811 (6)
			812 (6) 813 (6) 814 (6) 815 (6) 816 (6)
			817 (6) 818 (6) 819 (6) 820 (6) 821 (6)
			822 (6) 823 (6) 824 (6) 825 (6) 826 (6)
			827 (6) 828 (6) 829 (6) 830 (6) 831 (6)
			832 (6) 833 (6) 834 (6) 835 (6)
$DS_$ADD	1	670 (6)	670 (6) 699 (6) 717 (6) 797 (6) 835 (6)
$DS_$CASE	1	666 (6)	666 (6) 667 (6) 670 (6) 673 (6) 674 (6)
			675 (6) 676 (6) 681 (6) 690 (6) 701 (6)
			702 (6) 718 (6) 719 (6) 754 (6) 757 (6)
$DS_$COMPLEMENT	1	699 (6)	699 (6)
$DS_$DECIMAL	1	664 (6)	664 (6) 665 (6) 666 (6) 667 (6) 668 (6)
			669 (6) 670 (6) 672 (6) 677 (6) 678 (6)
			680 (6) 682 (6) 683 (6) 684 (6) 685 (6)
			686 (6) 687 (6) 688 (6) 689 (6) 690 (6)
			691 (6) 692 (6) 693 (6) 694 (6) 695 (6)
			696 (6) 697 (6) 700 (6) 702 (6) 703 (6)
			704 (6) 706 (6) 707 (6) 708 (6) 709 (6)
			710 (6) 711 (6) 712 (6) 713 (6) 716 (6)
			720 (6) 721 (6) 731 (6) 741 (6) 746 (6)
			747 (6) 748 (6) 750 (6) 752 (6) 754 (6)
			755 (6) 757 (6) 758 (6) 759 (6) 773 (6)
			774 (6) 777 (6) 780 (6) 788 (6) 789 (6)

```
                                                  790   (6)   796   (6)   803   (6)   804   (6)   805   (6)
                                                  806   (6)   807   (6)   814   (6)   815   (6)   817   (6)
                                                  818   (6)   819   (6)   820   (6)   821   (6)
$DS_$END          1          664    (6)           664   (6)   665   (6)   666   (6)   667   (6)   668   (6)
                                                  669   (6)   670   (6)   671   (6)   672   (6)   673   (6)
                                                  674   (6)   675   (6)   676   (6)   677   (6)   678   (6)
                                                  679   (6)   680   (6)   681   (6)   682   (6)   683   (6)
                                                  684   (6)   685   (6)   686   (6)   687   (6)   688   (6)
                                                  689   (6)   690   (6)   691   (6)   692   (6)   693   (6)
                                                  694   (6)   695   (6)   696   (6)   697   (6)   698   (6)
                                                  699   (6)   700   (6)   701   (6)   702   (6)   703   (6)
                                                  704   (6)   705   (6)   706   (6)   707   (6)   708   (6)
                                                  709   (6)   710   (6)   711   (6)   712   (6)   713   (6)
                                                  714   (6)   715   (6)   716   (6)   717   (6)   718   (6)
                                                  719   (6)   720   (6)   721   (6)   722   (6)   723   (6)
                                                  724   (6)   725   (6)   726   (6)   727   (6)   728   (6)
                                                  729   (6)   730   (6)   731   (6)   732   (6)   733   (6)
                                                  734   (6)   735   (6)   736   (6)   737   (6)   738   (6)
                                                  739   (6)   740   (6)   741   (6)   742   (6)   743   (6)
                                                  744   (6)   745   (6)   746   (6)   747   (6)   748   (6)
                                                  749   (6)   750   (6)   751   (6)   752   (6)   753   (6)
                                                  754   (6)   755   (6)   756   (6)   757   (6)   758   (6)
                                                  759   (6)   760   (6)   761   (6)   762   (6)   763   (6)
                                                  764   (6)   765   (6)   766   (6)   767   (6)   768   (6)
                                                  769   (6)   770   (6)   771   (6)   772   (6)   773   (6)
                                                  774   (6)   775   (6)   776   (6)   777   (6)   778   (6)
                                                  779   (6)   780   (6)   781   (6)   782   (6)   783   (6)
                                                  784   (6)   785   (6)   786   (6)   787   (6)   788   (6)
                                                  789   (6)   790   (6)   791   (6)   792   (6)   793   (6)
                                                  794   (6)   795   (6)   796   (6)   797   (6)   798   (6)
                                                  799   (6)   800   (6)   801   (6)   802   (6)   803   (6)
                                                  804   (6)   805   (6)   806   (6)   807   (6)   808   (6)
                                                  809   (6)   810   (6)   811   (6)   812   (6)   813   (6)
                                                  814   (6)   815   (6)   816   (6)   817   (6)   818   (6)
                                                  819   (6)   820   (6)   821   (6)   822   (6)   823   (6)
                                                  824   (6)   825   (6)   826   (6)   827   (6)   828   (6)
                                                  829   (6)   830   (6)   831   (6)   832   (6)   833   (6)
                                                  834   (6)   835   (6)
$DS_$FETCH        1          666    (6)           666   (6)   667   (6)   668   (6)   669   (6)   670   (6)
                                                  673   (6)   674   (6)   675   (6)   676   (6)   681   (6)
                                                  690   (6)   699   (6)   700   (6)   701   (6)   702   (6)
                                                  718   (6)   719   (6)   749   (6)   750   (6)   754   (6)
                                                  755   (6)   757   (6)   773   (6)   774   (6)   781   (6)
                                                  782   (6)   783   (6)   784   (6)   785   (6)   786   (6)
                                                  787   (6)   797   (6)   835   (6)
$DS_$HEX          1          666    (6)           666   (6)   667   (6)   673   (6)   674   (6)   675   (6)
                                                  676   (6)   681   (6)   690   (6)   701   (6)   702   (6)
                                                  707   (6)   708   (6)   709   (6)   710   (6)   711   (6)
                                                  712   (6)   713   (6)   717   (6)   718   (6)   719   (6)
                                                  755   (6)   835   (6)
$DS_$INITIALIZE   2          664    (6)           664   (6)   665   (6)   666   (6)   667   (6)   668   (6)
                                                  669   (6)   670   (6)   671   (6)   672   (6)   673   (6)
                                                  674   (6)   675   (6)   676   (6)   677   (6)   678   (6)
                                                  679   (6)   680   (6)   681   (6)   682   (6)   683   (6)
                                                  684   (6)   685   (6)   686   (6)   687   (6)   688   (6)
                                                  689   (6)   690   (6)   691   (6)   692   (6)   693   (6)
                                                  694   (6)   695   (6)   696   (6)   697   (6)   698   (6)
```

```
                                699  (6)    700  (6)    701  (6)    702  (6)    703  (6)
                                704  (6)    705  (6)    706  (6)    707  (6)    708  (6)
                                709  (6)    710  (6)    711  (6)    712  (6)    713  (6)
                                714  (6)    715  (6)    716  (6)    717  (6)    718  (6)
                                719  (6)    720  (6)    721  (6)    722  (6)    723  (6)
                                724  (6)    725  (6)    726  (6)    727  (6)    728  (6)
                                729  (6)    730  (6)    731  (6)    732  (6)    733  (6)
                                734  (6)    735  (6)    736  (6)    737  (6)    738  (6)
                                739  (6)    740  (6)    741  (6)    742  (6)    743  (6)
                                744  (6)    745  (6)    746  (6)    747  (6)    748  (6)
                                749  (6)    750  (6)    751  (6)    752  (6)    753  (6)
                                754  (6)    755  (6)    756  (6)    757  (6)    758  (6)
                                759  (6)    760  (6)    761  (6)    762  (6)    763  (6)
                                764  (6)    765  (6)    766  (6)    767  (6)    768  (6)
                                769  (6)    770  (6)    771  (6)    772  (6)    773  (6)
                                774  (6)    775  (6)    776  (6)    777  (6)    778  (6)
                                779  (6)    780  (6)    781  (6)    782  (6)    783  (6)
                                784  (6)    785  (6)    786  (6)    787  (6)    788  (6)
                                789  (6)    790  (6)    791  (6)    792  (6)    793  (6)
                                794  (6)    795  (6)    796  (6)    797  (6)    798  (6)
                                799  (6)    800  (6)    801  (6)    802  (6)    803  (6)
                                804  (6)    805  (6)    806  (6)    807  (6)    808  (6)
                                809  (6)    810  (6)    811  (6)    812  (6)    813  (6)
                                814  (6)    815  (6)    816  (6)    817  (6)    818  (6)
                                819  (6)    820  (6)    821  (6)    822  (6)    823  (6)
                                824  (6)    825  (6)    826  (6)    827  (6)    828  (6)
                                829  (6)    830  (6)    831  (6)    832  (6)    833  (6)
                                834  (6)    835  (6)
$DS_$LITERAL        1    666    (6)  666  (6)    667  (6)    668  (6)    669  (6)    670  (6)
                                672  (6)    673  (6)    674  (6)    675  (6)    676  (6)
                                681  (6)    683  (6)    685  (6)    687  (6)    689  (6)
                                695  (6)    696  (6)    698  (6)    699  (6)    700  (6)
                                701  (6)    702  (6)    707  (6)    708  (6)    709  (6)
                                710  (6)    711  (6)    712  (6)    713  (6)    714  (6)
                                715  (6)    716  (6)    717  (6)    718  (6)    719  (6)
                                722  (6)    723  (6)    724  (6)    725  (6)    726  (6)
                                727  (6)    728  (6)    729  (6)    732  (6)    733  (6)
                                734  (6)    735  (6)    736  (6)    737  (6)    738  (6)
                                739  (6)    740  (6)    742  (6)    743  (6)    744  (6)
                                745  (6)    746  (6)    748  (6)    750  (6)    752  (6)
                                754  (6)    757  (6)    759  (6)    760  (6)    761  (6)
                                762  (6)    763  (6)    764  (6)    765  (6)    766  (6)
                                767  (6)    768  (6)    769  (6)    770  (6)    771  (6)
                                772  (6)    773  (6)    774  (6)    775  (6)    776  (6)
                                778  (6)    779  (6)    781  (6)    782  (6)    783  (6)
                                784  (6)    785  (6)    786  (6)    787  (6)    788  (6)
                                789  (6)    791  (6)    794  (6)    797  (6)    798  (6)
                                800  (6)    801  (6)    802  (6)    805  (6)    806  (6)
                                807  (6)    808  (6)    809  (6)    810  (6)    811  (6)
                                812  (6)    813  (6)    814  (6)    815  (6)    819  (6)
                                820  (6)    821  (6)    822  (6)    823  (6)    824  (6)
                                825  (6)    826  (6)    827  (6)    828  (6)    829  (6)
                                830  (6)    831  (6)    832  (6)    833  (6)    834  (6)
                                835  (6)
$DS_$LOGICAL        1    684    (6)  684  (6)    689  (6)    695  (6)    696  (6)    707  (6)
                                708  (6)    709  (6)    710  (6)    713  (6)    715  (6)
                                716  (6)    717  (6)    756  (6)    798  (6)
```

$DS_$NAME 1 664 (6) 664 (6) 665 (6) 666 (6) 667 (6) 668 (6)
669 (6) 670 (6) 672 (6) 673 (6) 674 (6)
675 (6) 677 (6) 678 (6) 679 (6) 680 (6)
682 (6) 684 (6) 685 (6) 686 (6) 687 (6)
688 (6) 689 (6) 690 (6) 691 (6) 692 (6)
693 (6) 694 (6) 695 (6) 696 (6) 697 (6)
698 (6) 699 (6) 700 (6) 703 (6) 704 (6)
705 (6) 706 (6) 707 (6) 708 (6) 709 (6)
710 (6) 711 (6) 712 (6) 713 (6) 714 (6)
715 (6) 716 (6) 718 (6) 719 (6) 720 (6)
721 (6) 722 (6) 723 (6) 724 (6) 725 (6)
726 (6) 727 (6) 728 (6) 729 (6) 730 (6)
731 (6) 732 (6) 733 (6) 734 (6) 735 (6)
737 (6) 738 (6) 739 (6) 740 (6) 741 (6)
742 (6) 743 (6) 744 (6) 745 (6) 746 (6)
747 (6) 748 (6) 750 (6) 751 (6) 752 (6)
753 (6) 754 (6) 755 (6) 756 (6) 757 (6)
759 (6) 760 (6) 761 (6) 762 (6) 763 (6)
764 (6) 765 (6) 766 (6) 767 (6) 768 (6)
769 (6) 770 (6) 771 (6) 772 (6) 773 (6)
774 (6) 775 (6) 776 (6) 777 (6) 778 (6)
779 (6) 780 (6) 781 (6) 782 (6) 783 (6)
784 (6) 785 (6) 786 (6) 787 (6) 788 (6)
789 (6) 790 (6) 791 (6) 792 (6) 793 (6)
795 (6) 796 (6) 797 (6) 798 (6) 799 (6)
800 (6) 801 (6) 802 (6) 803 (6) 804 (6)
805 (6) 806 (6) 807 (6) 808 (6) 809 (6)
810 (6) 811 (6) 812 (6) 813 (6) 814 (6)
815 (6) 816 (6) 817 (6) 818 (6) 819 (6)
820 (6) 821 (6) 822 (6) 823 (6) 824 (6)
825 (6) 826 (6) 827 (6) 828 (6) 829 (6)
830 (6) 831 (6) 832 (6) 833 (6) 834 (6)
835 (6)

$DS_$OCTAL 1 664 (6) 664 (6) 665 (6) 672 (6) 677 (6) 678 (6)
680 (6) 682 (6) 683 (6) 684 (6) 685 (6)
686 (6) 687 (6) 688 (6) 689 (6) 691 (6)
692 (6) 693 (6) 694 (6) 697 (6) 704 (6)
705 (6) 706 (6) 720 (6) 721 (6) 731 (6)
741 (6) 746 (6) 747 (6) 758 (6) 777 (6)
780 (6) 788 (6) 789 (6) 790 (6) 796 (6)
805 (6) 806 (6) 807 (6) 814 (6) 815 (6)
816 (6) 817 (6) 818 (6) 819 (6) 820 (6)
821 (6)

$DS_$STORE 1 664 (6) 664 (6) 665 (6) 666 (6) 667 (6) 668 (6)
669 (6) 670 (6) 672 (6) 673 (6) 674 (6)
675 (6) 676 (6) 677 (6) 678 (6) 680 (6)
681 (6) 682 (6) 683 (6) 684 (6) 685 (6)
686 (6) 687 (6) 688 (6) 689 (6) 690 (6)
691 (6) 692 (6) 693 (6) 694 (6) 695 (6)
696 (6) 697 (6) 698 (6) 699 (6) 700 (6)
701 (6) 702 (6) 703 (6) 704 (6) 705 (6)
706 (6) 707 (6) 708 (6) 709 (6) 710 (6)
711 (6) 712 (6) 713 (6) 714 (6) 715 (6)
716 (6) 717 (6) 718 (6) 719 (6) 720 (6)
721 (6) 722 (6) 723 (6) 724 (6) 725 (6)
726 (6) 727 (6) 728 (6) 729 (6) 731 (6)
732 (6) 733 (6) 734 (6) 735 (6) 736 (6)

```
                                                  737    (6)    738    (6)    739    (6)    740    (6)    741    (6)
                                                  742    (6)    743    (6)    744    (6)    745    (6)    746    (6)
                                                  747    (6)    748    (6)    749    (6)    750    (6)    752    (6)
                                                  754    (6)    755    (6)    756    (6)    757    (6)    758    (6)
                                                  759    (6)    760    (6)    761    (6)    762    (6)    763    (6)
                                                  764    (6)    765    (6)    766    (6)    767    (6)    768    (6)
                                                  769    (6)    770    (6)    771    (6)    772    (6)    773    (6)
                                                  774    (6)    775    (6)    776    (6)    777    (6)    778    (6)
                                                  779    (6)    780    (6)    781    (6)    782    (6)    783    (6)
                                                  784    (6)    785    (6)    786    (6)    787    (6)    788    (6)
                                                  789    (6)    790    (6)    791    (6)    794    (6)    796    (6)
                                                  797    (6)    798    (6)    800    (6)    801    (6)    802    (6)
                                                  803    (6)    804    (6)    805    (6)    806    (6)    807    (6)
                                                  808    (6)    809    (6)    810    (6)    811    (6)    812    (6)
                                                  813    (6)    814    (6)    815    (6)    816    (6)    817    (6)
                                                  818    (6)    819    (6)    820    (6)    821    (6)    822    (6)
                                                  823    (6)    824    (6)    825    (6)    826    (6)    827    (6)
                                                  828    (6)    829    (6)    830    (6)    831    (6)    832    (6)
                                                  833    (6)    834    (6)    835    (6)
$DS_$STRING        1          670    (6)          670    (6)    684    (6)    685    (6)    686    (6)    687    (6)
                                                  689    (6)    695    (6)    696    (6)    704    (6)    750    (6)
                                                  798    (6)
$DS_AA11K          1          664    (6)          664    (6)
$DS_AA11K_DEF      1          664    (6)          664    (6)
$DS_AD11K          1          665    (6)          665    (6)
$DS_AD11K_DEF      1          665    (6)          665    (6)
$DS_CI750          2          668    (6)          668    (6)
$DS_CI780          3          669    (6)          669    (6)
$DS_CIBCA          3          667    (6)          667    (6)
$DS_CIBCI          3          666    (6)          666    (6)
$DS_CI_DEF         2          645    (6)          645    (6)
$DS_CI_NODE        5          670    (6)          670    (6)
$DS_CONSOLE        1          671    (6)          671    (6)
$DS_CONSOLE_DEF    1          671    (6)          671    (6)
$DS_CR11           1          672    (6)          672    (6)
$DS_CR11_DEF       1          672    (6)          672    (6)
$DS_DEBNA          2          673    (6)          673    (6)
$DS_DEBNA_DEF      1          673    (6)          673    (6)
$DS_DEBNK          2          674    (6)          674    (6)
$DS_DEBNK_DEF      1          674    (6)          674    (6)
$DS_DEBNT          2          675    (6)          675    (6)
$DS_DEBNT_DEF      1          675    (6)          675    (6)
$DS_DHB32          3          676    (6)          676    (6)
$DS_DHB32_DEF      1          676    (6)          676    (6)
$DS_DHU11          1          677    (6)          677    (6)
$DS_DHUV_DEF       1          646    (6)          646    (6)
$DS_DHV11          1          678    (6)          678    (6)
$DS_DISK           1          679    (6)          679    (6)
$DS_DISK_DEF       1          679    (6)          679    (6)
$DS_DL11           1          680    (6)          680    (6)
$DS_DL11_DEF       1          680    (6)          680    (6)
$DS_DMB32          3          681    (6)          681    (6)
$DS_DMB32_DEF      1          681    (6)          681    (6)
$DS_DMC11          1          682    (6)          682    (6)
$DS_DMC11_DEF      1          682    (6)          682    (6)
$DS_DMF32          1          683    (6)          683    (6)
$DS_DMF32A         2          684    (6)          684    (6)
```

$DS_DMF32A_DEF	2	684	(6)	684	(6)
$DS_DMF32P	1	685	(6)	685	(6)
$DS_DMF32P_DEF	1	685	(6)	685	(6)
$DS_DMF32S	1	686	(6)	686	(6)
$DS_DMF32S_DEF	2	686	(6)	686	(6)
$DS_DMF32_DEF	1	683	(6)	683	(6)
$DS_DMP11	2	687	(6)	687	(6)
$DS_DMP11_DEF	1	687	(6)	687	(6)
$DS_DMR11	1	688	(6)	688	(6)
$DS_DMR11_DEF	1	688	(6)	688	(6)
$DS_DMZ32	3	689	(6)	689	(6)
$DS_DMZ32_DEF	2	689	(6)	689	(6)
$DS_DR11B	1	691	(6)	691	(6)
$DS_DR11B_DEF	1	691	(6)	691	(6)
$DS_DR11C	1	692	(6)	692	(6)
$DS_DR11C_DEF	1	692	(6)	692	(6)
$DS_DR11K	1	693	(6)	693	(6)
$DS_DR11K_DEF	1	693	(6)	693	(6)
$DS_DR11W	1	694	(6)	694	(6)
$DS_DR11W_DEF	1	694	(6)	694	(6)
$DS_DR750	2	695	(6)	695	(6)
$DS_DR750_DEF	1	695	(6)	695	(6)
$DS_DR780	2	696	(6)	696	(6)
$DS_DR780_DEF	1	696	(6)	696	(6)
$DS_DRB32	3	690	(6)	690	(6)
$DS_DRB32_DEF	1	690	(6)	690	(6)
$DS_DUP11	1	697	(6)	697	(6)
$DS_DUP11_DEF	1	697	(6)	697	(6)
$DS_DW730	1	698	(6)	698	(6)
$DS_DW730_DEF	1	698	(6)	698	(6)
$DS_DW750	1	699	(6)	699	(6)
$DS_DW750_DEF	1	699	(6)	699	(6)
$DS_DW780	2	700	(6)	700	(6)
$DS_DW780_DEF	1	700	(6)	700	(6)
$DS_DWBLA	2	701	(6)	701	(6)
$DS_DWBLA_DEF	1	701	(6)	701	(6)
$DS_DWBUA	2	702	(6)	702	(6)
$DS_DWBUA_DEF	1	702	(6)	702	(6)
$DS_DX20	1	703	(6)	703	(6)
$DS_DX20_DEF	1	703	(6)	703	(6)
$DS_DZ11	1	704	(6)	704	(6)
$DS_DZ11_DEF	1	704	(6)	704	(6)
$DS_DZ32	2	705	(6)	705	(6)
$DS_DZ32_DEF	1	705	(6)	705	(6)
$DS_HPODEF	2	390	(2)	390	(2)
$DS_IEU11A	1	706	(6)	706	(6)
$DS_IEU11A_DEF	1	706	(6)	706	(6)
$DS_KA730	3	707	(6)	707	(6)
$DS_KA750	3	708	(6)	708	(6)
$DS_KA780	2	709	(6)	709	(6)
$DS_KA785	2	710	(6)	710	(6)
$DS_KA800	2	711	(6)	711	(6)
$DS_KA800_DEF	1	711	(6)	711	(6)
$DS_KA820	2	712	(6)	712	(6)
$DS_KA850	2	714	(6)	714	(6)
$DS_KA86	2	713	(6)	713	(6)
$DS_KA880	2	715	(6)	715	(6)

$DS_KA884	3	716	(6)	716	(6)
$DS_KA9CC	2	717	(6)	717	(6)
$DS_KA_DEF	4	647	(6)	647	(6)
$DS_KDB50	2	718	(6)	718	(6)
$DS_KDB50_DEF	1	718	(6)	718	(6)
$DS_KFBTA	2	719	(6)	719	(6)
$DS_KFBTA_DEF	1	719	(6)	719	(6)
$DS_KMC11	1	720	(6)	720	(6)
$DS_KMC11_DEF	1	720	(6)	720	(6)
$DS_KW11K	1	721	(6)	721	(6)
$DS_KW11K_DEF	1	721	(6)	721	(6)
$DS_LA100	1	726	(6)	726	(6)
$DS_LA100_DEF	1	726	(6)	726	(6)
$DS_LA12	1	722	(6)	722	(6)
$DS_LA120	1	727	(6)	727	(6)
$DS_LA120_DEF	1	727	(6)	727	(6)
$DS_LA12_DEF	1	722	(6)	722	(6)
$DS_LA180	1	728	(6)	728	(6)
$DS_LA180_DEF	1	728	(6)	728	(6)
$DS_LA210	1	729	(6)	729	(6)
$DS_LA210_DEF	1	729	(6)	729	(6)
$DS_LA34	1	723	(6)	723	(6)
$DS_LA34_DEF	1	723	(6)	723	(6)
$DS_LA36	1	724	(6)	724	(6)
$DS_LA36_DEF	1	724	(6)	724	(6)
$DS_LA38	1	725	(6)	725	(6)
$DS_LA38_DEF	1	725	(6)	725	(6)
$DS_LANCE	1	730	(6)	730	(6)
$DS_LANCE_DEF	1	730	(6)	730	(6)
$DS_LESI	1	731	(6)	731	(6)
$DS_LESI_DEF	1	731	(6)	731	(6)
$DS_LG01	1	732	(6)	732	(6)
$DS_LG01_DEF	1	732	(6)	732	(6)
$DS_LG02	1	733	(6)	733	(6)
$DS_LG02_DEF	1	733	(6)	733	(6)
$DS_LG03	1	734	(6)	734	(6)
$DS_LG03_DEF	1	734	(6)	734	(6)
$DS_LN01	1	735	(6)	735	(6)
$DS_LN01_DEF	1	735	(6)	735	(6)
$DS_LN03	1	736	(6)	736	(6)
$DS_LN03_DEF	1	736	(6)	736	(6)
$DS_LP04	1	737	(6)	737	(6)
$DS_LP04_DEF	1	737	(6)	737	(6)
$DS_LP05	1	738	(6)	738	(6)
$DS_LP05_DEF	1	738	(6)	738	(6)
$DS_LP06	1	739	(6)	739	(6)
$DS_LP06_DEF	1	739	(6)	739	(6)
$DS_LP07	1	740	(6)	740	(6)
$DS_LP07_DEF	1	740	(6)	740	(6)
$DS_LP11	1	741	(6)	741	(6)
$DS_LP11_DEF	1	741	(6)	741	(6)
$DS_LP14	1	742	(6)	742	(6)
$DS_LP14_DEF	1	742	(6)	742	(6)
$DS_LP25	1	743	(6)	743	(6)
$DS_LP25_DEF	1	743	(6)	743	(6)
$DS_LP26	1	744	(6)	744	(6)
$DS_LP26_DEF	1	744	(6)	744	(6)

$DS_LP27	1	745	(6)	745	(6)										
$DS_LP27_DEF	1	745	(6)	745	(6)										
$DS_LPA11K	1	746	(6)	746	(6)										
$DS_LPA11K_DEF	1	746	(6)	746	(6)										
$DS_LUA11	2	747	(6)	747	(6)										
$DS_LUA11_DEF	1	747	(6)	747	(6)										
$DS_MA780	2	748	(6)	748	(6)										
$DS_MA780_DEF	1	748	(6)	748	(6)										
$DS_MBE	1	749	(6)	749	(6)										
$DS_MBE_DEF	1	749	(6)	749	(6)										
$DS_ML11	1	750	(6)	750	(6)										
$DS_ML11_DEF	1	750	(6)	750	(6)										
$DS_MS750	1	751	(6)	751	(6)										
$DS_MS750_DEF	1	751	(6)	751	(6)										
$DS_MS780	1	752	(6)	752	(6)										
$DS_MS780_DEF	1	752	(6)	752	(6)										
$DS_MSAAA	1	753	(6)	753	(6)										
$DS_MSAAA_DEF	1	753	(6)	753	(6)										
$DS_NBIA	2	754	(6)	754	(6)										
$DS_NBIA_DEF	1	754	(6)	754	(6)										
$DS_NBIB	1	755	(6)	755	(6)										
$DS_NBIB_DEF	1	755	(6)	755	(6)										
$DS_NI	1	756	(6)	756	(6)										
$DS_NI_DEF	1	756	(6)	756	(6)										
$DS_PBIA	2	757	(6)	757	(6)										
$DS_PBIA_DEF	1	757	(6)	757	(6)										
$DS_PCL11	1	758	(6)	758	(6)										
$DS_PCL11_DEF	1	758	(6)	758	(6)										
$DS_PDDEF	1	664	(6)	664	(6)	665	(6)	666	(6)	667	(6)	668	(6)		
				669	(6)	670	(6)	671	(6)	672	(6)	673	(6)		
				674	(6)	675	(6)	676	(6)	677	(6)	678	(6)		
				679	(6)	680	(6)	681	(6)	682	(6)	683	(6)		
				684	(6)	685	(6)	686	(6)	687	(6)	688	(6)		
				689	(6)	690	(6)	691	(6)	692	(6)	693	(6)		
				694	(6)	695	(6)	696	(6)	697	(6)	698	(6)		
				699	(6)	700	(6)	701	(6)	702	(6)	703	(6)		
				704	(6)	705	(6)	706	(6)	707	(6)	708	(6)		
				709	(6)	710	(6)	711	(6)	712	(6)	713	(6)		
				714	(6)	715	(6)	716	(6)	717	(6)	718	(6)		
				719	(6)	720	(6)	721	(6)	722	(6)	723	(6)		
				724	(6)	725	(6)	726	(6)	727	(6)	728	(6)		
				729	(6)	730	(6)	731	(6)	732	(6)	733	(6)		
				734	(6)	735	(6)	736	(6)	737	(6)	738	(6)		
				739	(6)	740	(6)	741	(6)	742	(6)	743	(6)		
				744	(6)	745	(6)	746	(6)	747	(6)	748	(6)		
				749	(6)	750	(6)	751	(6)	752	(6)	753	(6)		
				754	(6)	755	(6)	756	(6)	757	(6)	758	(6)		
				759	(6)	760	(6)	761	(6)	762	(6)	763	(6)		
				764	(6)	765	(6)	766	(6)	767	(6)	768	(6)		
				769	(6)	770	(6)	771	(6)	772	(6)	773	(6)		
				774	(6)	775	(6)	776	(6)	777	(6)	778	(6)		
				779	(6)	780	(6)	781	(6)	782	(6)	783	(6)		
				784	(6)	785	(6)	786	(6)	787	(6)	788	(6)		
				789	(6)	790	(6)	791	(6)	792	(6)	793	(6)		
				794	(6)	795	(6)	796	(6)	797	(6)	798	(6)		
				799	(6)	800	(6)	801	(6)	802	(6)	803	(6)		
				804	(6)	805	(6)	806	(6)	807	(6)	808	(6)		

				809	(6)	810	(6)	811	(6)	812	(6)	813	(6)
				814	(6)	815	(6)	816	(6)	817	(6)	818	(6)
				819	(6)	820	(6)	821	(6)	822	(6)	823	(6)
				824	(6)	825	(6)	826	(6)	827	(6)	828	(6)
				829	(6)	830	(6)	831	(6)	832	(6)	833	(6)
				834	(6)	835	(6)						
$DS_PTDDEF	1	664	(6)	664	(6)	665	(6)	666	(6)	667	(6)	668	(6)
				669	(6)	670	(6)	671	(6)	672	(6)	673	(6)
				674	(6)	675	(6)	676	(6)	677	(6)	678	(6)
				679	(6)	680	(6)	681	(6)	682	(6)	683	(6)
				684	(6)	685	(6)	686	(6)	687	(6)	688	(6)
				689	(6)	690	(6)	691	(6)	692	(6)	693	(6)
				694	(6)	695	(6)	696	(6)	697	(6)	698	(6)
				699	(6)	700	(6)	701	(6)	702	(6)	703	(6)
				704	(6)	705	(6)	706	(6)	707	(6)	708	(6)
				709	(6)	710	(6)	711	(6)	712	(6)	713	(6)
				714	(6)	715	(6)	716	(6)	717	(6)	718	(6)
				719	(6)	720	(6)	721	(6)	722	(6)	723	(6)
				724	(6)	725	(6)	726	(6)	727	(6)	728	(6)
				729	(6)	730	(6)	731	(6)	732	(6)	733	(6)
				734	(6)	735	(6)	736	(6)	737	(6)	738	(6)
				739	(6)	740	(6)	741	(6)	742	(6)	743	(6)
				744	(6)	745	(6)	746	(6)	747	(6)	748	(6)
				749	(6)	750	(6)	751	(6)	752	(6)	753	(6)
				754	(6)	755	(6)	756	(6)	757	(6)	758	(6)
				759	(6)	760	(6)	761	(6)	762	(6)	763	(6)
				764	(6)	765	(6)	766	(6)	767	(6)	768	(6)
				769	(6)	770	(6)	771	(6)	772	(6)	773	(6)
				774	(6)	775	(6)	776	(6)	777	(6)	778	(6)
				779	(6)	780	(6)	781	(6)	782	(6)	783	(6)
				784	(6)	785	(6)	786	(6)	787	(6)	788	(6)
				789	(6)	790	(6)	791	(6)	792	(6)	793	(6)
				794	(6)	795	(6)	796	(6)	797	(6)	798	(6)
				799	(6)	800	(6)	801	(6)	802	(6)	803	(6)
				804	(6)	805	(6)	806	(6)	807	(6)	808	(6)
				809	(6)	810	(6)	811	(6)	812	(6)	813	(6)
				814	(6)	815	(6)	816	(6)	817	(6)	818	(6)
				819	(6)	820	(6)	821	(6)	822	(6)	823	(6)
				824	(6)	825	(6)	826	(6)	827	(6)	828	(6)
				829	(6)	830	(6)	831	(6)	832	(6)	833	(6)
				834	(6)	835	(6)						
$DS_PX780	2	759	(6)	759	(6)								
$DS_PX780_DEF	1	759	(6)	759	(6)								
$DS_R80	1	760	(6)	760	(6)								
$DS_R80_DEF	1	760	(6)	760	(6)								
$DS_RA60	1	761	(6)	761	(6)								
$DS_RA60_DEF	1	761	(6)	761	(6)								
$DS_RA70	1	762	(6)	762	(6)								
$DS_RA70_DEF	1	762	(6)	762	(6)								
$DS_RA80	1	763	(6)	763	(6)								
$DS_RA80_DEF	1	763	(6)	763	(6)								
$DS_RA81	1	764	(6)	764	(6)								
$DS_RA81_DEF	1	764	(6)	764	(6)								
$DS_RA82	1	765	(6)	765	(6)								
$DS_RA82_DEF	1	765	(6)	765	(6)								
$DS_RA90	1	766	(6)	766	(6)								
$DS_RA90_DEF	1	766	(6)	766	(6)								

$DS_RB730	2	767	(6)	767	(6)
$DS_RB730_DEF	1	767	(6)	767	(6)
$DS_RC25	1	769	(6)	769	(6)
$DS_RC25_DEF	1	769	(6)	769	(6)
$DS_RCF25	1	768	(6)	768	(6)
$DS_RCF25_DEF	1	768	(6)	768	(6)
$DS_RD52	1	770	(6)	770	(6)
$DS_RD52_DEF	1	770	(6)	770	(6)
$DS_RD53	1	771	(6)	771	(6)
$DS_RD53_DEF	1	771	(6)	771	(6)
$DS_RD54	1	772	(6)	772	(6)
$DS_RD54_DEF	1	772	(6)	772	(6)
$DS_RH750	2	773	(6)	773	(6)
$DS_RH750_DEF	1	773	(6)	773	(6)
$DS_RH780	2	774	(6)	774	(6)
$DS_RH780_DEF	1	774	(6)	774	(6)
$DS_RK06	1	778	(6)	778	(6)
$DS_RK06_DEF	1	778	(6)	778	(6)
$DS_RK07	1	779	(6)	779	(6)
$DS_RK07_DEF	1	779	(6)	779	(6)
$DS_RK611	1	780	(6)	780	(6)
$DS_RK611_DEF	1	780	(6)	780	(6)
$DS_RL01	1	775	(6)	775	(6)
$DS_RL01_DEF	1	775	(6)	775	(6)
$DS_RL02	1	776	(6)	776	(6)
$DS_RL02_DEF	1	776	(6)	776	(6)
$DS_RL11	1	777	(6)	777	(6)
$DS_RL11_DEF	1	777	(6)	777	(6)
$DS_RM03	1	781	(6)	781	(6)
$DS_RM03_DEF	1	781	(6)	781	(6)
$DS_RM05	1	782	(6)	782	(6)
$DS_RM05_DEF	1	782	(6)	782	(6)
$DS_RM80	1	783	(6)	783	(6)
$DS_RM80_DEF	1	783	(6)	783	(6)
$DS_RP04	1	784	(6)	784	(6)
$DS_RP04_DEF	1	784	(6)	784	(6)
$DS_RP05	1	785	(6)	785	(6)
$DS_RP05_DEF	1	785	(6)	785	(6)
$DS_RP06	1	786	(6)	786	(6)
$DS_RP06_DEF	1	786	(6)	786	(6)
$DS_RP07	1	787	(6)	787	(6)
$DS_RP07_DEF	1	787	(6)	787	(6)
$DS_RRD50	3	788	(6)	788	(6)
$DS_RRD50_DEF	1	788	(6)	788	(6)
$DS_RUX50	1	790	(6)	790	(6)
$DS_RUX50_DEF	1	790	(6)	790	(6)
$DS_RV20	3	789	(6)	789	(6)
$DS_RV20_DEF	1	789	(6)	789	(6)
$DS_RX02	1	791	(6)	791	(6)
$DS_RX02_DEF	1	791	(6)	791	(6)
$DS_RX180	1	795	(6)	795	(6)
$DS_RX180_DEF	1	795	(6)	795	(6)
$DS_RX211	1	796	(6)	796	(6)
$DS_RX211_DEF	1	796	(6)	796	(6)
$DS_RX31	1	792	(6)	792	(6)
$DS_RX31_DEF	1	792	(6)	792	(6)
$DS_RX32	1	793	(6)	793	(6)

$DS_RX32_DEF	1	793	(6)	793	(6)
$DS_RX50	1	794	(6)	794	(6)
$DS_RX50_DEF	1	794	(6)	794	(6)
$DS_SBIA	2	797	(6)	797	(6)
$DS_SBIA_DEF	1	797	(6)	797	(6)
$DS_SLU	1	798	(6)	798	(6)
$DS_SLU_DEF	1	798	(6)	798	(6)
$DS_TAPE	1	799	(6)	799	(6)
$DS_TAPE_DEF	1	799	(6)	799	(6)
$DS_TE16	1	800	(6)	800	(6)
$DS_TE16_DEF	1	800	(6)	800	(6)
$DS_TK50	1	801	(6)	801	(6)
$DS_TK50_DEF	1	801	(6)	801	(6)
$DS_TK70	1	802	(6)	802	(6)
$DS_TK70_DEF	1	802	(6)	802	(6)
$DS_TM03	1	803	(6)	803	(6)
$DS_TM03_DEF	1	803	(6)	803	(6)
$DS_TM78	1	804	(6)	804	(6)
$DS_TM78_DEF	1	804	(6)	804	(6)
$DS_TS04	1	805	(6)	805	(6)
$DS_TS04_DEF	1	805	(6)	805	(6)
$DS_TS05	2	806	(6)	806	(6)
$DS_TS05_DEF	1	806	(6)	806	(6)
$DS_TS11	1	807	(6)	807	(6)
$DS_TS11_DEF	1	807	(6)	807	(6)
$DS_TU45	1	808	(6)	808	(6)
$DS_TU45_DEF	1	808	(6)	808	(6)
$DS_TU58	1	809	(6)	809	(6)
$DS_TU58_DEF	1	809	(6)	809	(6)
$DS_TU70	1	810	(6)	810	(6)
$DS_TU70_DEF	1	810	(6)	810	(6)
$DS_TU72	1	811	(6)	811	(6)
$DS_TU72_DEF	1	811	(6)	811	(6)
$DS_TU77	1	812	(6)	812	(6)
$DS_TU77_DEF	1	812	(6)	812	(6)
$DS_TU78	1	813	(6)	813	(6)
$DS_TU78_DEF	1	813	(6)	813	(6)
$DS_TU80	2	814	(6)	814	(6)
$DS_TU80_DEF	1	814	(6)	814	(6)
$DS_TU81	2	815	(6)	815	(6)
$DS_TU81_DEF	1	815	(6)	815	(6)
$DS_UBE	1	816	(6)	816	(6)
$DS_UBE_DEF	1	816	(6)	816	(6)
$DS_UDA50	1	817	(6)	817	(6)
$DS_UDA50_DEF	1	817	(6)	817	(6)
$DS_UNA11	2	818	(6)	818	(6)
$DS_UNA11_DEF	1	818	(6)	818	(6)
$DS_VS100	1	819	(6)	819	(6)
$DS_VS100_DEF	1	819	(6)	819	(6)
$DS_VS125	1	820	(6)	820	(6)
$DS_VS125_DEF	1	820	(6)	820	(6)
$DS_VS300	1	821	(6)	821	(6)
$DS_VS300_DEF	1	821	(6)	821	(6)
$DS_VT100	1	827	(6)	827	(6)
$DS_VT100_DEF	1	827	(6)	827	(6)
$DS_VT101	1	828	(6)	828	(6)
$DS_VT101_DEF	1	828	(6)	828	(6)

```
$DS_VT102           1       829     (6)     829     (6)
$DS_VT102_DEF       1       829     (6)     829     (6)
$DS_VT125           1       830     (6)     830     (6)
$DS_VT125_DEF       1       830     (6)     830     (6)
$DS_VT131           1       831     (6)     831     (6)
$DS_VT131_DEF       1       831     (6)     831     (6)
$DS_VT132           1       832     (6)     832     (6)
$DS_VT132_DEF       1       832     (6)     832     (6)
$DS_VT220           1       833     (6)     833     (6)
$DS_VT220_DEF       1       833     (6)     833     (6)
$DS_VT240           1       834     (6)     834     (6)
$DS_VT240_DEF       1       834     (6)     834     (6)
$DS_VT50            1       822     (6)     822     (6)
$DS_VT50_DEF        1       822     (6)     822     (6)
$DS_VT52            1       823     (6)     823     (6)
$DS_VT52_DEF        1       823     (6)     823     (6)
$DS_VT55            1       824     (6)     824     (6)
$DS_VT55_DEF        1       824     (6)     824     (6)
$DS_VT61            1       825     (6)     825     (6)
$DS_VT61_DEF        1       825     (6)     825     (6)
$DS_VT71            1       826     (6)     826     (6)
$DS_VT71_DEF        1       826     (6)     826     (6)
$DS_XBI             3       835     (6)     835     (6)
$DS_XBI_DEF         1       835     (6)     835     (6)
$EQU                1       835     (6)     664     (6)     665     (6)     666     (6)     667     (6)     668     (6)
                                            669     (6)     670     (6)     671     (6)     672     (6)     673     (6)
                                            674     (6)     675     (6)     676     (6)     677     (6)     678     (6)
                                            679     (6)     680     (6)     681     (6)     682     (6)     683     (6)
                                            684     (6)     685     (6)     686     (6)     687     (6)     688     (6)
                                            689     (6)     690     (6)     691     (6)     692     (6)     693     (6)
                                            694     (6)     695     (6)     696     (6)     697     (6)     698     (6)
                                            699     (6)     700     (6)     701     (6)     702     (6)     703     (6)
                                            704     (6)     705     (6)     706     (6)     707     (6)     708     (6)
                                            709     (6)     710     (6)     711     (6)     712     (6)     713     (6)
                                            714     (6)     715     (6)     716     (6)     717     (6)     718     (6)
                                            719     (6)     720     (6)     721     (6)     722     (6)     723     (6)
                                            724     (6)     725     (6)     726     (6)     727     (6)     728     (6)
                                            729     (6)     730     (6)     731     (6)     732     (6)     733     (6)
                                            734     (6)     735     (6)     736     (6)     737     (6)     738     (6)
                                            739     (6)     740     (6)     741     (6)     742     (6)     743     (6)
                                            744     (6)     745     (6)     746     (6)     747     (6)     748     (6)
                                            749     (6)     750     (6)     751     (6)     752     (6)     753     (6)
                                            754     (6)     755     (6)     756     (6)     757     (6)     758     (6)
                                            759     (6)     760     (6)     761     (6)     762     (6)     763     (6)
                                            764     (6)     765     (6)     766     (6)     767     (6)     768     (6)
                                            769     (6)     770     (6)     771     (6)     772     (6)     773     (6)
                                            774     (6)     775     (6)     776     (6)     777     (6)     778     (6)
                                            779     (6)     780     (6)     781     (6)     782     (6)     783     (6)
                                            784     (6)     785     (6)     786     (6)     787     (6)     788     (6)
                                            789     (6)     790     (6)     791     (6)     792     (6)     793     (6)
                                            794     (6)     795     (6)     796     (6)     797     (6)     798     (6)
                                            799     (6)     800     (6)     801     (6)     802     (6)     803     (6)
                                            804     (6)     805     (6)     806     (6)     807     (6)     808     (6)
                                            809     (6)     810     (6)     811     (6)     812     (6)     813     (6)
                                            814     (6)     815     (6)     816     (6)     817     (6)     818     (6)
                                            819     (6)     820     (6)     821     (6)     822     (6)     823     (6)
                                            824     (6)     825     (6)     826     (6)     827     (6)     828     (6)
```

				829	(6)	830	(6)	831	(6)	832	(6)	833	(6)		
				834	(6)	835	(6)								
$EQULS1	1	835	(6)	664	(6)	665	(6)	666	(6)	667	(6)	668	(6)		
				669	(6)	670	(6)	671	(6)	672	(6)	673	(6)		
				674	(6)	675	(6)	676	(6)	677	(6)	678	(6)		
				679	(6)	680	(6)	681	(6)	682	(6)	683	(6)		
				684	(6)	685	(6)	686	(6)	687	(6)	688	(6)		
				689	(6)	690	(6)	691	(6)	692	(6)	693	(6)		
				694	(6)	695	(6)	696	(6)	697	(6)	698	(6)		
				699	(6)	700	(6)	701	(6)	702	(6)	703	(6)		
				704	(6)	705	(6)	706	(6)	707	(6)	708	(6)		
				709	(6)	710	(6)	711	(6)	712	(6)	713	(6)		
				714	(6)	715	(6)	716	(6)	717	(6)	718	(6)		
				719	(6)	720	(6)	721	(6)	722	(6)	723	(6)		
				724	(6)	725	(6)	726	(6)	727	(6)	728	(6)		
				729	(6)	730	(6)	731	(6)	732	(6)	733	(6)		
				734	(6)	735	(6)	736	(6)	737	(6)	738	(6)		
				739	(6)	740	(6)	741	(6)	742	(6)	743	(6)		
				744	(6)	745	(6)	746	(6)	747	(6)	748	(6)		
				749	(6)	750	(6)	751	(6)	752	(6)	753	(6)		
				754	(6)	755	(6)	756	(6)	757	(6)	758	(6)		
				759	(6)	760	(6)	761	(6)	762	(6)	763	(6)		
				764	(6)	765	(6)	766	(6)	767	(6)	768	(6)		
				769	(6)	770	(6)	771	(6)	772	(6)	773	(6)		
				774	(6)	775	(6)	776	(6)	777	(6)	778	(6)		
				779	(6)	780	(6)	781	(6)	782	(6)	783	(6)		
				784	(6)	785	(6)	786	(6)	787	(6)	788	(6)		
				789	(6)	790	(6)	791	(6)	792	(6)	793	(6)		
				794	(6)	795	(6)	796	(6)	797	(6)	798	(6)		
				799	(6)	800	(6)	801	(6)	802	(6)	803	(6)		
				804	(6)	805	(6)	806	(6)	807	(6)	808	(6)		
				809	(6)	810	(6)	811	(6)	812	(6)	813	(6)		
				814	(6)	815	(6)	816	(6)	817	(6)	818	(6)		
				819	(6)	820	(6)	821	(6)	822	(6)	823	(6)		
				824	(6)	825	(6)	826	(6)	827	(6)	828	(6)		
				829	(6)	830	(6)	831	(6)	832	(6)	833	(6)		
				834	(6)	835	(6)								
$EQULST	1	664	(6)	664	(6)	665	(6)	666	(6)	667	(6)	668	(6)		
				669	(6)	670	(6)	671	(6)	672	(6)	673	(6)		
				674	(6)	675	(6)	676	(6)	677	(6)	678	(6)		
				679	(6)	680	(6)	681	(6)	682	(6)	683	(6)		
				684	(6)	685	(6)	686	(6)	687	(6)	688	(6)		
				689	(6)	690	(6)	691	(6)	692	(6)	693	(6)		
				694	(6)	695	(6)	696	(6)	697	(6)	698	(6)		
				699	(6)	700	(6)	701	(6)	702	(6)	703	(6)		
				704	(6)	705	(6)	706	(6)	707	(6)	708	(6)		
				709	(6)	710	(6)	711	(6)	712	(6)	713	(6)		
				714	(6)	715	(6)	716	(6)	717	(6)	718	(6)		
				719	(6)	720	(6)	721	(6)	722	(6)	723	(6)		
				724	(6)	725	(6)	726	(6)	727	(6)	728	(6)		
				729	(6)	730	(6)	731	(6)	732	(6)	733	(6)		
				734	(6)	735	(6)	736	(6)	737	(6)	738	(6)		
				739	(6)	740	(6)	741	(6)	742	(6)	743	(6)		
				744	(6)	745	(6)	746	(6)	747	(6)	748	(6)		
				749	(6)	750	(6)	751	(6)	752	(6)	753	(6)		
				754	(6)	755	(6)	756	(6)	757	(6)	758	(6)		
				759	(6)	760	(6)	761	(6)	762	(6)	763	(6)		

```
                                764  (6)   765  (6)   766  (6)   767  (6)   768  (6)
                                769  (6)   770  (6)   771  (6)   772  (6)   773  (6)
                                774  (6)   775  (6)   776  (6)   777  (6)   778  (6)
                                779  (6)   780  (6)   781  (6)   782  (6)   783  (6)
                                784  (6)   785  (6)   786  (6)   787  (6)   788  (6)
                                789  (6)   790  (6)   791  (6)   792  (6)   793  (6)
                                794  (6)   795  (6)   796  (6)   797  (6)   798  (6)
                                799  (6)   800  (6)   801  (6)   802  (6)   803  (6)
                                804  (6)   805  (6)   806  (6)   807  (6)   808  (6)
                                809  (6)   810  (6)   811  (6)   812  (6)   813  (6)
                                814  (6)   815  (6)   816  (6)   817  (6)   818  (6)
                                819  (6)   820  (6)   821  (6)   822  (6)   823  (6)
                                824  (6)   825  (6)   826  (6)   827  (6)   828  (6)
                                829  (6)   830  (6)   831  (6)   832  (6)   833  (6)
                                834  (6)   835  (6)
$GBLINI         2               664  (6)   665  (6)   666  (6)   667  (6)   668  (6)
                                669  (6)   670  (6)   671  (6)   672  (6)   673  (6)
                                674  (6)   675  (6)   676  (6)   677  (6)   678  (6)
                                679  (6)   680  (6)   681  (6)   682  (6)   683  (6)
                                684  (6)   685  (6)   686  (6)   687  (6)   688  (6)
                                689  (6)   690  (6)   691  (6)   692  (6)   693  (6)
                                694  (6)   695  (6)   696  (6)   697  (6)   698  (6)
                                699  (6)   700  (6)   701  (6)   702  (6)   703  (6)
                                704  (6)   705  (6)   706  (6)   707  (6)   708  (6)
                                709  (6)   710  (6)   711  (6)   712  (6)   713  (6)
                                714  (6)   715  (6)   716  (6)   717  (6)   718  (6)
                                719  (6)   720  (6)   721  (6)   722  (6)   723  (6)
                                724  (6)   725  (6)   726  (6)   727  (6)   728  (6)
                                729  (6)   730  (6)   731  (6)   732  (6)   733  (6)
                                734  (6)   735  (6)   736  (6)   737  (6)   738  (6)
                                739  (6)   740  (6)   741  (6)   742  (6)   743  (6)
                                744  (6)   745  (6)   746  (6)   747  (6)   748  (6)
                                749  (6)   750  (6)   751  (6)   752  (6)   753  (6)
                                754  (6)   755  (6)   756  (6)   757  (6)   758  (6)
                                759  (6)   760  (6)   761  (6)   762  (6)   763  (6)
                                764  (6)   765  (6)   766  (6)   767  (6)   768  (6)
                                769  (6)   770  (6)   771  (6)   772  (6)   773  (6)
                                774  (6)   775  (6)   776  (6)   777  (6)   778  (6)
                                779  (6)   780  (6)   781  (6)   782  (6)   783  (6)
                                784  (6)   785  (6)   786  (6)   787  (6)   788  (6)
                                789  (6)   790  (6)   791  (6)   792  (6)   793  (6)
                                794  (6)   795  (6)   796  (6)   797  (6)   798  (6)
                                799  (6)   800  (6)   801  (6)   802  (6)   803  (6)
                                804  (6)   805  (6)   806  (6)   807  (6)   808  (6)
                                809  (6)   810  (6)   811  (6)   812  (6)   813  (6)
                                814  (6)   815  (6)   816  (6)   817  (6)   818  (6)
                                819  (6)   820  (6)   821  (6)   822  (6)   823  (6)
                                824  (6)   825  (6)   826  (6)   827  (6)   828  (6)
                                829  (6)   830  (6)   831  (6)   832  (6)   833  (6)
                                834  (6)   835  (6)
$VIELD          1               664  (6)   665  (6)   666  (6)   667  (6)   668  (6)
                                669  (6)   670  (6)   671  (6)   672  (6)   673  (6)
                                674  (6)   675  (6)   676  (6)   677  (6)   678  (6)
                                679  (6)   680  (6)   681  (6)   682  (6)   683  (6)
                                684  (6)   685  (6)   686  (6)   687  (6)   688  (6)
                                689  (6)   690  (6)   691  (6)   692  (6)   693  (6)
                                694  (6)   695  (6)   696  (6)   697  (6)   698  (6)
```

				699	(6)	700	(6)	701	(6)	702	(6)	703	(6)
				704	(6)	705	(6)	706	(6)	707	(6)	708	(6)
				709	(6)	710	(6)	711	(6)	712	(6)	713	(6)
				714	(6)	715	(6)	716	(6)	717	(6)	718	(6)
				719	(6)	720	(6)	721	(6)	722	(6)	723	(6)
				724	(6)	725	(6)	726	(6)	727	(6)	728	(6)
				729	(6)	730	(6)	731	(6)	732	(6)	733	(6)
				734	(6)	735	(6)	736	(6)	737	(6)	738	(6)
				739	(6)	740	(6)	741	(6)	742	(6)	743	(6)
				744	(6)	745	(6)	746	(6)	747	(6)	748	(6)
				749	(6)	750	(6)	751	(6)	752	(6)	753	(6)
				754	(6)	755	(6)	756	(6)	757	(6)	758	(6)
				759	(6)	760	(6)	761	(6)	762	(6)	763	(6)
				764	(6)	765	(6)	766	(6)	767	(6)	768	(6)
				769	(6)	770	(6)	771	(6)	772	(6)	773	(6)
				774	(6)	775	(6)	776	(6)	777	(6)	778	(6)
				779	(6)	780	(6)	781	(6)	782	(6)	783	(6)
				784	(6)	785	(6)	786	(6)	787	(6)	788	(6)
				789	(6)	790	(6)	791	(6)	792	(6)	793	(6)
				794	(6)	795	(6)	796	(6)	797	(6)	798	(6)
				799	(6)	800	(6)	801	(6)	802	(6)	803	(6)
				804	(6)	805	(6)	806	(6)	807	(6)	808	(6)
				809	(6)	810	(6)	811	(6)	812	(6)	813	(6)
				814	(6)	815	(6)	816	(6)	817	(6)	818	(6)
				819	(6)	820	(6)	821	(6)	822	(6)	823	(6)
				824	(6)	825	(6)	826	(6)	827	(6)	828	(6)
				829	(6)	830	(6)	831	(6)	832	(6)	833	(6)
				834	(6)	835	(6)						
$VIELD1	1	835	(6)	664	(6)	665	(6)	666	(6)	667	(6)	668	(6)
				669	(6)	670	(6)	671	(6)	672	(6)	673	(6)
				674	(6)	675	(6)	676	(6)	677	(6)	678	(6)
				679	(6)	680	(6)	681	(6)	682	(6)	683	(6)
				684	(6)	685	(6)	686	(6)	687	(6)	688	(6)
				689	(6)	690	(6)	691	(6)	692	(6)	693	(6)
				694	(6)	695	(6)	696	(6)	697	(6)	698	(6)
				699	(6)	700	(6)	701	(6)	702	(6)	703	(6)
				704	(6)	705	(6)	706	(6)	707	(6)	708	(6)
				709	(6)	710	(6)	711	(6)	712	(6)	713	(6)
				714	(6)	715	(6)	716	(6)	717	(6)	718	(6)
				719	(6)	720	(6)	721	(6)	722	(6)	723	(6)
				724	(6)	725	(6)	726	(6)	727	(6)	728	(6)
				729	(6)	730	(6)	731	(6)	732	(6)	733	(6)
				734	(6)	735	(6)	736	(6)	737	(6)	738	(6)
				739	(6)	740	(6)	741	(6)	742	(6)	743	(6)
				744	(6)	745	(6)	746	(6)	747	(6)	748	(6)
				749	(6)	750	(6)	751	(6)	752	(6)	753	(6)
				754	(6)	755	(6)	756	(6)	757	(6)	758	(6)
				759	(6)	760	(6)	761	(6)	762	(6)	763	(6)
				764	(6)	765	(6)	766	(6)	767	(6)	768	(6)
				769	(6)	770	(6)	771	(6)	772	(6)	773	(6)
				774	(6)	775	(6)	776	(6)	777	(6)	778	(6)
				779	(6)	780	(6)	781	(6)	782	(6)	783	(6)
				784	(6)	785	(6)	786	(6)	787	(6)	788	(6)
				789	(6)	790	(6)	791	(6)	792	(6)	793	(6)
				794	(6)	795	(6)	796	(6)	797	(6)	798	(6)
				799	(6)	800	(6)	801	(6)	802	(6)	803	(6)
				804	(6)	805	(6)	806	(6)	807	(6)	808	(6)

809	(6)	810	(6)	811	(6)	812	(6)	813	(6)		
814	(6)	815	(6)	816	(6)	817	(6)	818	(6)		
819	(6)	820	(6)	821	(6)	822	(6)	823	(6)		
824	(6)	825	(6)	826	(6)	827	(6)	828	(6)		
829	(6)	830	(6)	831	(6)	832	(6)	833	(6)		
834	(6)	835	(6)								

Directives Cross Reference

DIRECTIVE	REFERENCES...
.ASCIC	664 (6) 665 (6) 666 (6) 667 (6) 668 (6) 669 (6) 670 (6) 671 (6)
	672 (6) 673 (6) 674 (6) 675 (6) 676 (6) 677 (6) 678 (6) 679 (6)
	680 (6) 681 (6) 682 (6) 683 (6) 684 (6) 685 (6) 686 (6) 687 (6)
	688 (6) 689 (6) 690 (6) 691 (6) 692 (6) 693 (6) 694 (6) 695 (6)
	696 (6) 697 (6) 698 (6) 699 (6) 700 (6) 701 (6) 702 (6) 703 (6)
	704 (6) 705 (6) 706 (6) 707 (6) 708 (6) 709 (6) 710 (6) 711 (6)
	712 (6) 713 (6) 714 (6) 715 (6) 716 (6) 717 (6) 718 (6) 719 (6)
	720 (6) 721 (6) 722 (6) 723 (6) 724 (6) 725 (6) 726 (6) 727 (6)
	728 (6) 729 (6) 730 (6) 731 (6) 732 (6) 733 (6) 734 (6) 735 (6)
	736 (6) 737 (6) 738 (6) 739 (6) 740 (6) 741 (6) 742 (6) 743 (6)
	744 (6) 745 (6) 746 (6) 747 (6) 748 (6) 749 (6) 750 (6) 751 (6)
	752 (6) 753 (6) 754 (6) 755 (6) 756 (6) 757 (6) 758 (6) 759 (6)
	760 (6) 761 (6) 762 (6) 763 (6) 764 (6) 765 (6) 766 (6) 767 (6)
	768 (6) 769 (6) 770 (6) 771 (6) 772 (6) 773 (6) 774 (6) 775 (6)
	776 (6) 777 (6) 778 (6) 779 (6) 780 (6) 781 (6) 782 (6) 783 (6)
	784 (6) 785 (6) 786 (6) 787 (6) 788 (6) 789 (6) 790 (6) 791 (6)
	792 (6) 793 (6) 794 (6) 795 (6) 796 (6) 797 (6) 798 (6) 799 (6)
	800 (6) 801 (6) 802 (6) 803 (6) 804 (6) 805 (6) 806 (6) 807 (6)
	808 (6) 809 (6) 810 (6) 811 (6) 812 (6) 813 (6) 814 (6) 815 (6)
	816 (6) 817 (6) 818 (6) 819 (6) 820 (6) 821 (6) 822 (6) 823 (6)
	824 (6) 825 (6) 826 (6) 827 (6) 828 (6) 829 (6) 830 (6) 831 (6)
	832 (6) 833 (6) 834 (6) 835 (6)
.BLKL	419 (4) 426 (4)
.BYTE	664 (6) 665 (6) 666 (6) 667 (6) 668 (6) 669 (6) 670 (6) 671 (6)
	672 (6) 673 (6) 674 (6) 675 (6) 676 (6) 677 (6) 678 (6) 679 (6)
	680 (6) 681 (6) 682 (6) 683 (6) 684 (6) 685 (6) 686 (6) 687 (6)
	688 (6) 689 (6) 690 (6) 691 (6) 692 (6) 693 (6) 694 (6) 695 (6)
	696 (6) 697 (6) 698 (6) 699 (6) 700 (6) 701 (6) 702 (6) 703 (6)
	704 (6) 705 (6) 706 (6) 707 (6) 708 (6) 709 (6) 710 (6) 711 (6)
	712 (6) 713 (6) 714 (6) 715 (6) 716 (6) 717 (6) 718 (6) 719 (6)
	720 (6) 721 (6) 722 (6) 723 (6) 724 (6) 725 (6) 726 (6) 727 (6)
	728 (6) 729 (6) 730 (6) 731 (6) 732 (6) 733 (6) 734 (6) 735 (6)
	736 (6) 737 (6) 738 (6) 739 (6) 740 (6) 741 (6) 742 (6) 743 (6)
	744 (6) 745 (6) 746 (6) 747 (6) 748 (6) 749 (6) 750 (6) 751 (6)
	752 (6) 753 (6) 754 (6) 755 (6) 756 (6) 757 (6) 758 (6) 759 (6)
	760 (6) 761 (6) 762 (6) 763 (6) 764 (6) 765 (6) 766 (6) 767 (6)
	768 (6) 769 (6) 770 (6) 771 (6) 772 (6) 773 (6) 774 (6) 775 (6)
	776 (6) 777 (6) 778 (6) 779 (6) 780 (6) 781 (6) 782 (6) 783 (6)
	784 (6) 785 (6) 786 (6) 787 (6) 788 (6) 789 (6) 790 (6) 791 (6)
	792 (6) 793 (6) 794 (6) 795 (6) 796 (6) 797 (6) 798 (6) 799 (6)
	800 (6) 801 (6) 802 (6) 803 (6) 804 (6) 805 (6) 806 (6) 807 (6)
	808 (6) 809 (6) 810 (6) 811 (6) 812 (6) 813 (6) 814 (6) 815 (6)
	816 (6) 817 (6) 818 (6) 819 (6) 820 (6) 821 (6) 822 (6) 823 (6)
	824 (6) 825 (6) 826 (6) 827 (6) 828 (6) 829 (6) 830 (6) 831 (6)
	832 (6) 833 (6) 834 (6) 835 (6)
.DSABL	63 (1)
.END	836 (6)
.ENDC	664 (6) 665 (6) 666 (6) 667 (6) 668 (6) 669 (6) 670 (6) 671 (6)
	672 (6) 673 (6) 674 (6) 675 (6) 676 (6) 677 (6) 678 (6) 679 (6)
	680 (6) 681 (6) 682 (6) 683 (6) 684 (6) 685 (6) 686 (6) 687 (6)

```
                688  (6)   689  (6)   690  (6)   691  (6)   692  (6)   693  (6)   694  (6)   695  (6)
                696  (6)   697  (6)   698  (6)   699  (6)   700  (6)   701  (6)   702  (6)   703  (6)
                704  (6)   705  (6)   706  (6)   707  (6)   708  (6)   709  (6)   710  (6)   711  (6)
                712  (6)   713  (6)   714  (6)   715  (6)   716  (6)   717  (6)   718  (6)   719  (6)
                720  (6)   721  (6)   722  (6)   723  (6)   724  (6)   725  (6)   726  (6)   727  (6)
                728  (6)   729  (6)   730  (6)   731  (6)   732  (6)   733  (6)   734  (6)   735  (6)
                736  (6)   737  (6)   738  (6)   739  (6)   740  (6)   741  (6)   742  (6)   743  (6)
                744  (6)   745  (6)   746  (6)   747  (6)   748  (6)   749  (6)   750  (6)   751  (6)
                752  (6)   753  (6)   754  (6)   755  (6)   756  (6)   757  (6)   758  (6)   759  (6)
                760  (6)   761  (6)   762  (6)   763  (6)   764  (6)   765  (6)   766  (6)   767  (6)
                768  (6)   769  (6)   770  (6)   771  (6)   772  (6)   773  (6)   774  (6)   775  (6)
                776  (6)   777  (6)   778  (6)   779  (6)   780  (6)   781  (6)   782  (6)   783  (6)
                784  (6)   785  (6)   786  (6)   787  (6)   788  (6)   789  (6)   790  (6)   791  (6)
                792  (6)   793  (6)   794  (6)   795  (6)   796  (6)   797  (6)   798  (6)   799  (6)
                800  (6)   801  (6)   802  (6)   803  (6)   804  (6)   805  (6)   806  (6)   807  (6)
                808  (6)   809  (6)   810  (6)   811  (6)   812  (6)   813  (6)   814  (6)   815  (6)
                816  (6)   817  (6)   818  (6)   819  (6)   820  (6)   821  (6)   822  (6)   823  (6)
                824  (6)   825  (6)   826  (6)   827  (6)   828  (6)   829  (6)   830  (6)   831  (6)
                832  (6)   833  (6)   834  (6)   835  (6)
.ENDM           390  (2)   645  (6)   646  (6)   647  (6)   664  (6)   665  (6)   666  (6)   667  (6)
                668  (6)   669  (6)   670  (6)   671  (6)   672  (6)   673  (6)   674  (6)   675  (6)
                676  (6)   677  (6)   678  (6)   679  (6)   680  (6)   681  (6)   682  (6)   683  (6)
                684  (6)   685  (6)   686  (6)   687  (6)   688  (6)   689  (6)   690  (6)   691  (6)
                692  (6)   693  (6)   694  (6)   695  (6)   696  (6)   697  (6)   698  (6)   699  (6)
                700  (6)   701  (6)   702  (6)   703  (6)   704  (6)   705  (6)   706  (6)   707  (6)
                708  (6)   709  (6)   710  (6)   711  (6)   712  (6)   713  (6)   714  (6)   715  (6)
                716  (6)   717  (6)   718  (6)   719  (6)   720  (6)   721  (6)   722  (6)   723  (6)
                724  (6)   725  (6)   726  (6)   727  (6)   728  (6)   729  (6)   730  (6)   731  (6)
                732  (6)   733  (6)   734  (6)   735  (6)   736  (6)   737  (6)   738  (6)   739  (6)
                740  (6)   741  (6)   742  (6)   743  (6)   744  (6)   745  (6)   746  (6)   747  (6)
                748  (6)   749  (6)   750  (6)   751  (6)   752  (6)   753  (6)   754  (6)   755  (6)
                756  (6)   757  (6)   758  (6)   759  (6)   760  (6)   761  (6)   762  (6)   763  (6)
                764  (6)   765  (6)   766  (6)   767  (6)   768  (6)   769  (6)   770  (6)   771  (6)
                772  (6)   773  (6)   774  (6)   775  (6)   776  (6)   777  (6)   778  (6)   779  (6)
                780  (6)   781  (6)   782  (6)   783  (6)   784  (6)   785  (6)   786  (6)   787  (6)
                788  (6)   789  (6)   790  (6)   791  (6)   792  (6)   793  (6)   794  (6)   795  (6)
                796  (6)   797  (6)   798  (6)   799  (6)   800  (6)   801  (6)   802  (6)   803  (6)
                804  (6)   805  (6)   806  (6)   807  (6)   808  (6)   809  (6)   810  (6)   811  (6)
                812  (6)   813  (6)   814  (6)   815  (6)   816  (6)   817  (6)   818  (6)   819  (6)
                820  (6)   821  (6)   822  (6)   823  (6)   824  (6)   825  (6)   826  (6)   827  (6)
                828  (6)   829  (6)   830  (6)   831  (6)   832  (6)   833  (6)   834  (6)   835  (6)
.ENDR           664  (6)   665  (6)   666  (6)   667  (6)   668  (6)   669  (6)   670  (6)   671  (6)
                672  (6)   673  (6)   674  (6)   675  (6)   676  (6)   677  (6)   678  (6)   679  (6)
                680  (6)   681  (6)   682  (6)   683  (6)   684  (6)   685  (6)   686  (6)   687  (6)
                688  (6)   689  (6)   690  (6)   691  (6)   692  (6)   693  (6)   694  (6)   695  (6)
                696  (6)   697  (6)   698  (6)   699  (6)   700  (6)   701  (6)   702  (6)   703  (6)
                704  (6)   705  (6)   706  (6)   707  (6)   708  (6)   709  (6)   710  (6)   711  (6)
                712  (6)   713  (6)   714  (6)   715  (6)   716  (6)   717  (6)   718  (6)   719  (6)
                720  (6)   721  (6)   722  (6)   723  (6)   724  (6)   725  (6)   726  (6)   727  (6)
                728  (6)   729  (6)   730  (6)   731  (6)   732  (6)   733  (6)   734  (6)   735  (6)
                736  (6)   737  (6)   738  (6)   739  (6)   740  (6)   741  (6)   742  (6)   743  (6)
                744  (6)   745  (6)   746  (6)   747  (6)   748  (6)   749  (6)   750  (6)   751  (6)
                752  (6)   753  (6)   754  (6)   755  (6)   756  (6)   757  (6)   758  (6)   759  (6)
                760  (6)   761  (6)   762  (6)   763  (6)   764  (6)   765  (6)   766  (6)   767  (6)
                768  (6)   769  (6)   770  (6)   771  (6)   772  (6)   773  (6)   774  (6)   775  (6)
                776  (6)   777  (6)   778  (6)   779  (6)   780  (6)   781  (6)   782  (6)   783  (6)
                784  (6)   785  (6)   786  (6)   787  (6)   788  (6)   789  (6)   790  (6)   791  (6)
```

```
                792  (6)   793  (6)   794  (6)   795  (6)   796  (6)   797  (6)   798  (6)   799  (6)
                800  (6)   801  (6)   802  (6)   803  (6)   804  (6)   805  (6)   806  (6)   807  (6)
                808  (6)   809  (6)   810  (6)   811  (6)   812  (6)   813  (6)   814  (6)   815  (6)
                816  (6)   817  (6)   818  (6)   819  (6)   820  (6)   821  (6)   822  (6)   823  (6)
                824  (6)   825  (6)   826  (6)   827  (6)   828  (6)   829  (6)   830  (6)   831  (6)
                832  (6)   833  (6)   834  (6)   835  (6)

.IDENT          61   (1)

.IF             664  (6)   665  (6)   666  (6)   667  (6)   668  (6)   669  (6)   670  (6)   671  (6)
                672  (6)   673  (6)   674  (6)   675  (6)   676  (6)   677  (6)   678  (6)   679  (6)
                680  (6)   681  (6)   682  (6)   683  (6)   684  (6)   685  (6)   686  (6)   687  (6)
                688  (6)   689  (6)   690  (6)   691  (6)   692  (6)   693  (6)   694  (6)   695  (6)
                696  (6)   697  (6)   698  (6)   699  (6)   700  (6)   701  (6)   702  (6)   703  (6)
                704  (6)   705  (6)   706  (6)   707  (6)   708  (6)   709  (6)   710  (6)   711  (6)
                712  (6)   713  (6)   714  (6)   715  (6)   716  (6)   717  (6)   718  (6)   719  (6)
                720  (6)   721  (6)   722  (6)   723  (6)   724  (6)   725  (6)   726  (6)   727  (6)
                728  (6)   729  (6)   730  (6)   731  (6)   732  (6)   733  (6)   734  (6)   735  (6)
                736  (6)   737  (6)   738  (6)   739  (6)   740  (6)   741  (6)   742  (6)   743  (6)
                744  (6)   745  (6)   746  (6)   747  (6)   748  (6)   749  (6)   750  (6)   751  (6)
                752  (6)   753  (6)   754  (6)   755  (6)   756  (6)   757  (6)   758  (6)   759  (6)
                760  (6)   761  (6)   762  (6)   763  (6)   764  (6)   765  (6)   766  (6)   767  (6)
                768  (6)   769  (6)   770  (6)   771  (6)   772  (6)   773  (6)   774  (6)   775  (6)
                776  (6)   777  (6)   778  (6)   779  (6)   780  (6)   781  (6)   782  (6)   783  (6)
                784  (6)   785  (6)   786  (6)   787  (6)   788  (6)   789  (6)   790  (6)   791  (6)
                792  (6)   793  (6)   794  (6)   795  (6)   796  (6)   797  (6)   798  (6)   799  (6)
                800  (6)   801  (6)   802  (6)   803  (6)   804  (6)   805  (6)   806  (6)   807  (6)
                808  (6)   809  (6)   810  (6)   811  (6)   812  (6)   813  (6)   814  (6)   815  (6)
                816  (6)   817  (6)   818  (6)   819  (6)   820  (6)   821  (6)   822  (6)   823  (6)
                824  (6)   825  (6)   826  (6)   827  (6)   828  (6)   829  (6)   830  (6)   831  (6)
                832  (6)   833  (6)   834  (6)   835  (6)

.IFF            664  (6)   665  (6)   666  (6)   667  (6)   668  (6)   669  (6)   670  (6)   671  (6)
                672  (6)   673  (6)   674  (6)   675  (6)   676  (6)   677  (6)   678  (6)   679  (6)
                680  (6)   681  (6)   682  (6)   683  (6)   684  (6)   685  (6)   686  (6)   687  (6)
                688  (6)   689  (6)   690  (6)   691  (6)   692  (6)   693  (6)   694  (6)   695  (6)
                696  (6)   697  (6)   698  (6)   699  (6)   700  (6)   701  (6)   702  (6)   703  (6)
                704  (6)   705  (6)   706  (6)   707  (6)   708  (6)   709  (6)   710  (6)   711  (6)
                712  (6)   713  (6)   714  (6)   715  (6)   716  (6)   717  (6)   718  (6)   719  (6)
                720  (6)   721  (6)   722  (6)   723  (6)   724  (6)   725  (6)   726  (6)   727  (6)
                728  (6)   729  (6)   730  (6)   731  (6)   732  (6)   733  (6)   734  (6)   735  (6)
                736  (6)   737  (6)   738  (6)   739  (6)   740  (6)   741  (6)   742  (6)   743  (6)
                744  (6)   745  (6)   746  (6)   747  (6)   748  (6)   749  (6)   750  (6)   751  (6)
                752  (6)   753  (6)   754  (6)   755  (6)   756  (6)   757  (6)   758  (6)   759  (6)
                760  (6)   761  (6)   762  (6)   763  (6)   764  (6)   765  (6)   766  (6)   767  (6)
                768  (6)   769  (6)   770  (6)   771  (6)   772  (6)   773  (6)   774  (6)   775  (6)
                776  (6)   777  (6)   778  (6)   779  (6)   780  (6)   781  (6)   782  (6)   783  (6)
                784  (6)   785  (6)   786  (6)   787  (6)   788  (6)   789  (6)   790  (6)   791  (6)
                792  (6)   793  (6)   794  (6)   795  (6)   796  (6)   797  (6)   798  (6)   799  (6)
                800  (6)   801  (6)   802  (6)   803  (6)   804  (6)   805  (6)   806  (6)   807  (6)
                808  (6)   809  (6)   810  (6)   811  (6)   812  (6)   813  (6)   814  (6)   815  (6)
                816  (6)   817  (6)   818  (6)   819  (6)   820  (6)   821  (6)   822  (6)   823  (6)
                824  (6)   825  (6)   826  (6)   827  (6)   828  (6)   829  (6)   830  (6)   831  (6)
                832  (6)   833  (6)   834  (6)   835  (6)

.IIF            664  (6)   665  (6)   666  (6)   667  (6)   668  (6)   669  (6)   670  (6)   671  (6)
                672  (6)   673  (6)   674  (6)   675  (6)   676  (6)   677  (6)   678  (6)   679  (6)
                680  (6)   681  (6)   682  (6)   683  (6)   684  (6)   685  (6)   686  (6)   687  (6)
                688  (6)   689  (6)   690  (6)   691  (6)   692  (6)   693  (6)   694  (6)   695  (6)
                696  (6)   697  (6)   698  (6)   699  (6)   700  (6)   701  (6)   702  (6)   703  (6)
                704  (6)   705  (6)   706  (6)   707  (6)   708  (6)   709  (6)   710  (6)   711  (6)
```

```
                712     (6)     713     (6)     714     (6)     715     (6)     716     (6)     717     (6)     718     (6)     719     (6)
                720     (6)     721     (6)     722     (6)     723     (6)     724     (6)     725     (6)     726     (6)     727     (6)
                728     (6)     729     (6)     730     (6)     731     (6)     732     (6)     733     (6)     734     (6)     735     (6)
                736     (6)     737     (6)     738     (6)     739     (6)     740     (6)     741     (6)     742     (6)     743     (6)
                744     (6)     745     (6)     746     (6)     747     (6)     748     (6)     749     (6)     750     (6)     751     (6)
                752     (6)     753     (6)     754     (6)     755     (6)     756     (6)     757     (6)     758     (6)     759     (6)
                760     (6)     761     (6)     762     (6)     763     (6)     764     (6)     765     (6)     766     (6)     767     (6)
                768     (6)     769     (6)     770     (6)     771     (6)     772     (6)     773     (6)     774     (6)     775     (6)
                776     (6)     777     (6)     778     (6)     779     (6)     780     (6)     781     (6)     782     (6)     783     (6)
                784     (6)     785     (6)     786     (6)     787     (6)     788     (6)     789     (6)     790     (6)     791     (6)
                792     (6)     793     (6)     794     (6)     795     (6)     796     (6)     797     (6)     798     (6)     799     (6)
                800     (6)     801     (6)     802     (6)     803     (6)     804     (6)     805     (6)     806     (6)     807     (6)
                808     (6)     809     (6)     810     (6)     811     (6)     812     (6)     813     (6)     814     (6)     815     (6)
                816     (6)     817     (6)     818     (6)     819     (6)     820     (6)     821     (6)     822     (6)     823     (6)
                824     (6)     825     (6)     826     (6)     827     (6)     828     (6)     829     (6)     830     (6)     831     (6)
                832     (6)     833     (6)     834     (6)     835     (6)
.IRP            664     (6)     665     (6)     666     (6)     667     (6)     668     (6)     669     (6)     670     (6)     671     (6)
                672     (6)     673     (6)     674     (6)     675     (6)     676     (6)     677     (6)     678     (6)     679     (6)
                680     (6)     681     (6)     682     (6)     683     (6)     684     (6)     685     (6)     686     (6)     687     (6)
                688     (6)     689     (6)     690     (6)     691     (6)     692     (6)     693     (6)     694     (6)     695     (6)
                696     (6)     697     (6)     698     (6)     699     (6)     700     (6)     701     (6)     702     (6)     703     (6)
                704     (6)     705     (6)     706     (6)     707     (6)     708     (6)     709     (6)     710     (6)     711     (6)
                712     (6)     713     (6)     714     (6)     715     (6)     716     (6)     717     (6)     718     (6)     719     (6)
                720     (6)     721     (6)     722     (6)     723     (6)     724     (6)     725     (6)     726     (6)     727     (6)
                728     (6)     729     (6)     730     (6)     731     (6)     732     (6)     733     (6)     734     (6)     735     (6)
                736     (6)     737     (6)     738     (6)     739     (6)     740     (6)     741     (6)     742     (6)     743     (6)
                744     (6)     745     (6)     746     (6)     747     (6)     748     (6)     749     (6)     750     (6)     751     (6)
                752     (6)     753     (6)     754     (6)     755     (6)     756     (6)     757     (6)     758     (6)     759     (6)
                760     (6)     761     (6)     762     (6)     763     (6)     764     (6)     765     (6)     766     (6)     767     (6)
                768     (6)     769     (6)     770     (6)     771     (6)     772     (6)     773     (6)     774     (6)     775     (6)
                776     (6)     777     (6)     778     (6)     779     (6)     780     (6)     781     (6)     782     (6)     783     (6)
                784     (6)     785     (6)     786     (6)     787     (6)     788     (6)     789     (6)     790     (6)     791     (6)
                792     (6)     793     (6)     794     (6)     795     (6)     796     (6)     797     (6)     798     (6)     799     (6)
                800     (6)     801     (6)     802     (6)     803     (6)     804     (6)     805     (6)     806     (6)     807     (6)
                808     (6)     809     (6)     810     (6)     811     (6)     812     (6)     813     (6)     814     (6)     815     (6)
                816     (6)     817     (6)     818     (6)     819     (6)     820     (6)     821     (6)     822     (6)     823     (6)
                824     (6)     825     (6)     826     (6)     827     (6)     828     (6)     829     (6)     830     (6)     831     (6)
                832     (6)     833     (6)     834     (6)     835     (6)
.LIBRARY        388     (2)
.LIST           62      (1)     639     (6)
.LONG           402     (3)     403     (3)     406     (3)     408     (3)     410     (3)     435     (4)     436     (4)     438     (4)
                439     (4)     450     (4)     451     (4)     462     (5)     463     (5)     464     (5)     465     (5)     466     (5)
                467     (5)     468     (5)     469     (5)     470     (5)     471     (5)     472     (5)     473     (5)     474     (5)
                475     (5)     476     (5)     477     (5)     478     (5)     479     (5)     480     (5)     481     (5)     482     (5)
                483     (5)     484     (5)     485     (5)     486     (5)     487     (5)     488     (5)     489     (5)     490     (5)
                491     (5)     492     (5)     493     (5)     494     (5)     495     (5)     496     (5)     497     (5)     498     (5)
                499     (5)     500     (5)     501     (5)     502     (5)     503     (5)     504     (5)     505     (5)     506     (5)
                507     (5)     508     (5)     509     (5)     510     (5)     511     (5)     512     (5)     513     (5)     514     (5)
                515     (5)     516     (5)     517     (5)     518     (5)     519     (5)     520     (5)     521     (5)     522     (5)
                523     (5)     524     (5)     525     (5)     526     (5)     527     (5)     528     (5)     529     (5)     530     (5)
                531     (5)     532     (5)     533     (5)     534     (5)     535     (5)     536     (5)     537     (5)     538     (5)
                539     (5)     540     (5)     541     (5)     542     (5)     543     (5)     544     (5)     545     (5)     546     (5)
                547     (5)     548     (5)     549     (5)     550     (5)     551     (5)     552     (5)     553     (5)     554     (5)
                555     (5)     556     (5)     557     (5)     558     (5)     559     (5)     560     (5)     561     (5)     562     (5)
                563     (5)     564     (5)     565     (5)     566     (5)     567     (5)     568     (5)     569     (5)     570     (5)
                571     (5)     572     (5)     573     (5)     574     (5)     575     (5)     576     (5)     577     (5)     578     (5)
                579     (5)     580     (5)     581     (5)     582     (5)     583     (5)     584     (5)     585     (5)     586     (5)
```

```
                587  (5)     588  (5)     589  (5)     590  (5)     591  (5)     592  (5)     593  (5)     594  (5)
                595  (5)     596  (5)     597  (5)     598  (5)     599  (5)     600  (5)     601  (5)     602  (5)
                603  (5)     604  (5)     605  (5)     606  (5)     607  (5)     608  (5)     609  (5)     610  (5)
                611  (5)     612  (5)     613  (5)     614  (5)     615  (5)     616  (5)     617  (5)     618  (5)
                619  (5)     620  (5)     621  (5)     622  (5)     623  (5)     624  (5)     625  (5)     626  (5)
                627  (5)     628  (5)     629  (5)     630  (5)     631  (5)     632  (5)     633  (5)     634  (5)
                664  (6)     665  (6)     666  (6)     667  (6)     668  (6)     669  (6)     670  (6)     672  (6)
                673  (6)     674  (6)     675  (6)     676  (6)     677  (6)     678  (6)     680  (6)     681  (6)
                682  (6)     683  (6)     684  (6)     685  (6)     686  (6)     687  (6)     688  (6)     689  (6)
                690  (6)     691  (6)     692  (6)     693  (6)     694  (6)     695  (6)     696  (6)     697  (6)
                698  (6)     699  (6)     700  (6)     701  (6)     702  (6)     703  (6)     704  (6)     705  (6)
                706  (6)     707  (6)     708  (6)     709  (6)     710  (6)     711  (6)     712  (6)     713  (6)
                714  (6)     715  (6)     716  (6)     717  (6)     718  (6)     719  (6)     720  (6)     721  (6)
                722  (6)     723  (6)     724  (6)     725  (6)     726  (6)     727  (6)     728  (6)     729  (6)
                731  (6)     732  (6)     733  (6)     734  (6)     735  (6)     736  (6)     737  (6)     738  (6)
                739  (6)     740  (6)     741  (6)     742  (6)     743  (6)     744  (6)     745  (6)     746  (6)
                747  (6)     748  (6)     750  (6)     752  (6)     754  (6)     755  (6)     757  (6)     758  (6)
                759  (6)     760  (6)     761  (6)     762  (6)     763  (6)     764  (6)     765  (6)     766  (6)
                767  (6)     768  (6)     769  (6)     770  (6)     771  (6)     772  (6)     773  (6)     774  (6)
                775  (6)     776  (6)     777  (6)     778  (6)     779  (6)     780  (6)     781  (6)     782  (6)
                783  (6)     784  (6)     785  (6)     786  (6)     787  (6)     788  (6)     789  (6)     790  (6)
                791  (6)     794  (6)     796  (6)     797  (6)     798  (6)     800  (6)     801  (6)     802  (6)
                803  (6)     804  (6)     805  (6)     806  (6)     807  (6)     808  (6)     809  (6)     810  (6)
                811  (6)     812  (6)     813  (6)     814  (6)     815  (6)     816  (6)     817  (6)     818  (6)
                819  (6)     820  (6)     821  (6)     822  (6)     823  (6)     824  (6)     825  (6)     826  (6)
                827  (6)     828  (6)     829  (6)     830  (6)     831  (6)     832  (6)     833  (6)     834  (6)
                835  (6)
.MACRO          664  (6)     665  (6)     666  (6)     667  (6)     668  (6)     669  (6)     670  (6)     671  (6)
                672  (6)     673  (6)     674  (6)     675  (6)     676  (6)     677  (6)     678  (6)     679  (6)
                680  (6)     681  (6)     682  (6)     683  (6)     684  (6)     685  (6)     686  (6)     687  (6)
                688  (6)     689  (6)     690  (6)     691  (6)     692  (6)     693  (6)     694  (6)     695  (6)
                696  (6)     697  (6)     698  (6)     699  (6)     700  (6)     701  (6)     702  (6)     703  (6)
                704  (6)     705  (6)     706  (6)     707  (6)     708  (6)     709  (6)     710  (6)     711  (6)
                712  (6)     713  (6)     714  (6)     715  (6)     716  (6)     717  (6)     718  (6)     719  (6)
                720  (6)     721  (6)     722  (6)     723  (6)     724  (6)     725  (6)     726  (6)     727  (6)
                728  (6)     729  (6)     730  (6)     731  (6)     732  (6)     733  (6)     734  (6)     735  (6)
                736  (6)     737  (6)     738  (6)     739  (6)     740  (6)     741  (6)     742  (6)     743  (6)
                744  (6)     745  (6)     746  (6)     747  (6)     748  (6)     749  (6)     750  (6)     751  (6)
                752  (6)     753  (6)     754  (6)     755  (6)     756  (6)     757  (6)     758  (6)     759  (6)
                760  (6)     761  (6)     762  (6)     763  (6)     764  (6)     765  (6)     766  (6)     767  (6)
                768  (6)     769  (6)     770  (6)     771  (6)     772  (6)     773  (6)     774  (6)     775  (6)
                776  (6)     777  (6)     778  (6)     779  (6)     780  (6)     781  (6)     782  (6)     783  (6)
                784  (6)     785  (6)     786  (6)     787  (6)     788  (6)     789  (6)     790  (6)     791  (6)
                792  (6)     793  (6)     794  (6)     795  (6)     796  (6)     797  (6)     798  (6)     799  (6)
                800  (6)     801  (6)     802  (6)     803  (6)     804  (6)     805  (6)     806  (6)     807  (6)
                808  (6)     809  (6)     810  (6)     811  (6)     812  (6)     813  (6)     814  (6)     815  (6)
                816  (6)     817  (6)     818  (6)     819  (6)     820  (6)     821  (6)     822  (6)     823  (6)
                824  (6)     825  (6)     826  (6)     827  (6)     828  (6)     829  (6)     830  (6)     831  (6)
                832  (6)     833  (6)     834  (6)     835  (6)
.NLIST          638  (6)     64   (1)
.NOCROSS        390  (2)     645  (6)     646  (6)     647  (6)     664  (6)     665  (6)     671  (6)     672  (6)
                673  (6)     674  (6)     675  (6)     676  (6)     679  (6)     680  (6)     681  (6)     682  (6)
                683  (6)     684  (6)     685  (6)     686  (6)     687  (6)     688  (6)     689  (6)     690  (6)
                691  (6)     692  (6)     693  (6)     694  (6)     695  (6)     696  (6)     697  (6)     698  (6)
                699  (6)     700  (6)     701  (6)     702  (6)     703  (6)     704  (6)     705  (6)     706  (6)
                711  (6)     718  (6)     719  (6)     720  (6)     721  (6)     722  (6)     723  (6)     724  (6)
                725  (6)     726  (6)     727  (6)     728  (6)     729  (6)     730  (6)     731  (6)     732  (6)
```

```
                733     (6)     734     (6)     735     (6)     736     (6)     737     (6)     738     (6)     739     (6)     740     (6)
                741     (6)     742     (6)     743     (6)     744     (6)     745     (6)     746     (6)     747     (6)     748     (6)
                749     (6)     750     (6)     751     (6)     752     (6)     753     (6)     754     (6)     755     (6)     756     (6)
                757     (6)     758     (6)     759     (6)     760     (6)     761     (6)     762     (6)     763     (6)     764     (6)
                765     (6)     766     (6)     767     (6)     768     (6)     769     (6)     770     (6)     771     (6)     772     (6)
                773     (6)     774     (6)     775     (6)     776     (6)     777     (6)     778     (6)     779     (6)     780     (6)
                781     (6)     782     (6)     783     (6)     784     (6)     785     (6)     786     (6)     787     (6)     788     (6)
                789     (6)     790     (6)     791     (6)     792     (6)     793     (6)     794     (6)     795     (6)     796     (6)
                797     (6)     798     (6)     799     (6)     800     (6)     801     (6)     802     (6)     803     (6)     804     (6)
                805     (6)     806     (6)     807     (6)     808     (6)     809     (6)     810     (6)     811     (6)     812     (6)
                813     (6)     814     (6)     815     (6)     816     (6)     817     (6)     818     (6)     819     (6)     820     (6)
                821     (6)     822     (6)     823     (6)     824     (6)     825     (6)     826     (6)     827     (6)     828     (6)
                829     (6)     830     (6)     831     (6)     832     (6)     833     (6)     834     (6)     835     (6)
.PSECT          400     (3)     412     (4)     454     (5)     637     (6)
.RESTORE        390     (2)     645     (6)     646     (6)     647     (6)     664     (6)     665     (6)     671     (6)     672     (6)
                673     (6)     674     (6)     675     (6)     676     (6)     679     (6)     680     (6)     681     (6)     682     (6)
                683     (6)     684     (6)     685     (6)     686     (6)     687     (6)     688     (6)     689     (6)     690     (6)
                691     (6)     692     (6)     693     (6)     694     (6)     695     (6)     696     (6)     697     (6)     698     (6)
                699     (6)     700     (6)     701     (6)     702     (6)     703     (6)     704     (6)     705     (6)     706     (6)
                711     (6)     718     (6)     719     (6)     720     (6)     721     (6)     722     (6)     723     (6)     724     (6)
                725     (6)     726     (6)     727     (6)     728     (6)     729     (6)     730     (6)     731     (6)     732     (6)
                733     (6)     734     (6)     735     (6)     736     (6)     737     (6)     738     (6)     739     (6)     740     (6)
                741     (6)     742     (6)     743     (6)     744     (6)     745     (6)     746     (6)     747     (6)     748     (6)
                749     (6)     750     (6)     751     (6)     752     (6)     753     (6)     754     (6)     755     (6)     756     (6)
                757     (6)     758     (6)     759     (6)     760     (6)     761     (6)     762     (6)     763     (6)     764     (6)
                765     (6)     766     (6)     767     (6)     768     (6)     769     (6)     770     (6)     771     (6)     772     (6)
                773     (6)     774     (6)     775     (6)     776     (6)     777     (6)     778     (6)     779     (6)     780     (6)
                781     (6)     782     (6)     783     (6)     784     (6)     785     (6)     786     (6)     787     (6)     788     (6)
                789     (6)     790     (6)     791     (6)     792     (6)     793     (6)     794     (6)     795     (6)     796     (6)
                797     (6)     798     (6)     799     (6)     800     (6)     801     (6)     802     (6)     803     (6)     804     (6)
                805     (6)     806     (6)     807     (6)     808     (6)     809     (6)     810     (6)     811     (6)     812     (6)
                813     (6)     814     (6)     815     (6)     816     (6)     817     (6)     818     (6)     819     (6)     820     (6)
                821     (6)     822     (6)     823     (6)     824     (6)     825     (6)     826     (6)     827     (6)     828     (6)
                829     (6)     830     (6)     831     (6)     832     (6)     833     (6)     834     (6)     835     (6)
.SAVE           390     (2)     645     (6)     646     (6)     647     (6)     664     (6)     665     (6)     671     (6)     672     (6)
                673     (6)     674     (6)     675     (6)     676     (6)     679     (6)     680     (6)     681     (6)     682     (6)
                683     (6)     684     (6)     685     (6)     686     (6)     687     (6)     688     (6)     689     (6)     690     (6)
                691     (6)     692     (6)     693     (6)     694     (6)     695     (6)     696     (6)     697     (6)     698     (6)
                699     (6)     700     (6)     701     (6)     702     (6)     703     (6)     704     (6)     705     (6)     706     (6)
                711     (6)     718     (6)     719     (6)     720     (6)     721     (6)     722     (6)     723     (6)     724     (6)
                725     (6)     726     (6)     727     (6)     728     (6)     729     (6)     730     (6)     731     (6)     732     (6)
                733     (6)     734     (6)     735     (6)     736     (6)     737     (6)     738     (6)     739     (6)     740     (6)
                741     (6)     742     (6)     743     (6)     744     (6)     745     (6)     746     (6)     747     (6)     748     (6)
                749     (6)     750     (6)     751     (6)     752     (6)     753     (6)     754     (6)     755     (6)     756     (6)
                757     (6)     758     (6)     759     (6)     760     (6)     761     (6)     762     (6)     763     (6)     764     (6)
                765     (6)     766     (6)     767     (6)     768     (6)     769     (6)     770     (6)     771     (6)     772     (6)
                773     (6)     774     (6)     775     (6)     776     (6)     777     (6)     778     (6)     779     (6)     780     (6)
                781     (6)     782     (6)     783     (6)     784     (6)     785     (6)     786     (6)     787     (6)     788     (6)
                789     (6)     790     (6)     791     (6)     792     (6)     793     (6)     794     (6)     795     (6)     796     (6)
                797     (6)     798     (6)     799     (6)     800     (6)     801     (6)     802     (6)     803     (6)     804     (6)
                805     (6)     806     (6)     807     (6)     808     (6)     809     (6)     810     (6)     811     (6)     812     (6)
                813     (6)     814     (6)     815     (6)     816     (6)     817     (6)     818     (6)     819     (6)     820     (6)
                821     (6)     822     (6)     823     (6)     824     (6)     825     (6)     826     (6)     827     (6)     828     (6)
                829     (6)     830     (6)     831     (6)     832     (6)     833     (6)     834     (6)     835     (6)
.SBTTL          383     (2)     453     (5)
.TITLE          60      (1)
.WORD           664     (6)     665     (6)     666     (6)     667     (6)     668     (6)     669     (6)     670     (6)     671     (6)
```

```
        672  (6)    673  (6)    674  (6)    675  (6)    676  (6)    677  (6)    678  (6)    679  (6)
        680  (6)    681  (6)    682  (6)    683  (6)    684  (6)    685  (6)    686  (6)    687  (6)
        688  (6)    689  (6)    690  (6)    691  (6)    692  (6)    693  (6)    694  (6)    695  (6)
        696  (6)    697  (6)    698  (6)    699  (6)    700  (6)    701  (6)    702  (6)    703  (6)
        704  (6)    705  (6)    706  (6)    707  (6)    708  (6)    709  (6)    710  (6)    711  (6)
        712  (6)    713  (6)    714  (6)    715  (6)    716  (6)    717  (6)    718  (6)    719  (6)
        720  (6)    721  (6)    722  (6)    723  (6)    724  (6)    725  (6)    726  (6)    727  (6)
        728  (6)    729  (6)    730  (6)    731  (6)    732  (6)    733  (6)    734  (6)    735  (6)
        736  (6)    737  (6)    738  (6)    739  (6)    740  (6)    741  (6)    742  (6)    743  (6)
        744  (6)    745  (6)    746  (6)    747  (6)    748  (6)    749  (6)    750  (6)    751  (6)
        752  (6)    753  (6)    754  (6)    755  (6)    756  (6)    757  (6)    758  (6)    759  (6)
        760  (6)    761  (6)    762  (6)    763  (6)    764  (6)    765  (6)    766  (6)    767  (6)
        768  (6)    769  (6)    770  (6)    771  (6)    772  (6)    773  (6)    774  (6)    775  (6)
        776  (6)    777  (6)    778  (6)    779  (6)    780  (6)    781  (6)    782  (6)    783  (6)
        784  (6)    785  (6)    786  (6)    787  (6)    788  (6)    789  (6)    790  (6)    791  (6)
        792  (6)    793  (6)    794  (6)    795  (6)    796  (6)    797  (6)    798  (6)    799  (6)
        800  (6)    801  (6)    802  (6)    803  (6)    804  (6)    805  (6)    806  (6)    807  (6)
        808  (6)    809  (6)    810  (6)    811  (6)    812  (6)    813  (6)    814  (6)    815  (6)
        816  (6)    817  (6)    818  (6)    819  (6)    820  (6)    821  (6)    822  (6)    823  (6)
        824  (6)    825  (6)    826  (6)    827  (6)    828  (6)    829  (6)    830  (6)    831  (6)
        832  (6)    833  (6)    834  (6)    835  (6)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.20
Command processing	877	00:00:00.23	00:00:01.10
Pass 1	9349	00:00:57.37	00:01:05.13
Symbol table sort	2	00:00:00.18	00:00:00.18
Pass 2	6019	00:00:07.77	00:00:14.72
Symbol table output	3	00:00:00.12	00:00:00.53
Psect synopsis output	3	00:00:00.00	00:00:00.00
Cross-reference output	899	00:00:03.49	00:00:05.66
Assembler run totals	17175	00:01:09.19	00:01:27.52

The working set limit was 4096 pages.
2609663 bytes (5097 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 766 non-local and 0 local symbols.
836 source lines were read in Pass 1, producing 110 object records in Pass 2.
2267 pages of virtual memory were used to define 515 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	347
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	352

```
3661 GETS were required to define 352 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:IOBASE.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1)	44	HISTORY	; DETAILED
(2)	100	DECLARATIONS	
(3)	139	I/O COMPLETION POSTING	
(4)	256	VIRTUAL I/O COMPLETION	
(5)	325	QUEUE NEXT SEGMENT	
(6)	439	BUFFERED READ COMPLETION AST ROUTINE	
(7)	523	DIRECT I/O COMPLETION AST ROUTINE	
(8)	600	MOVE DATA TO USER BUFFER	
(9)	623	UNLOCK AREAS IN IRPE'S	

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element IOPOST.MAR
0000     2 ; *1     6-FEB-1986 15:18:56 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element IOPOST.MAR
0000     4         .TITLE  IOPOST  *** IOPOST I/O completion posting
0000     5         .IDENT  /05-04/
0000     6
0000     7 ;
0000     8 ;*****************************************************************************
0000     9 ;*                                                                           *
0000    10 ;*  COPYRIGHT (c) 1978, 1979, 1980                                           *
0000    11 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    12 ;*                                                                           *
0000    13 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    14 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    15 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    16 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    17 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    18 ;*  TRANSFERRED.                                                             *
0000    19 ;*                                                                           *
0000    20 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    21 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    22 ;*  CORPORATION.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    25 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28
0000    29 ;++
0000    30 ; FACILITY: EXECUTIVE, I/O SYSTEM
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;       IOCIOPOST IMPLEMENTS THE DEVICE INDEPENDENT COMPLETION PROCESSING FOR
0000    34 ;       I/O PACKETS.  IT IS INVOKED BY QUEUEING THE PACKET ON THE I/O POST QUEUE
0000    35 ;       AND TRIGGERING THE IPL$_IOPOST SOFTWARE INTERRUPT.  SOME OF THE IOPOST
0000    36 ;       OPERATIONS SUCH AS SETTING EVENT FLAGS, UNLOCKING BUFFER PAGES,
0000    37 ;       RELEASING BUFFERS AND PAGING I/O COMPLETION ARE PERFORMED IN THE IOPOST
0000    38 ;       INTERRUPT SERVICE ROUTINE, WHILE OTHER OPERATIONS THAT REQUIRE ACCESS
0000    39 ;       TO PROCESS ADDRESS SPACE ARE PERFORMED BY SENDING A SPECIAL KERNEL AST.
0000    40 ;
0000    41 ; ENVIRONMENT:  MODE = KERNEL, RESIDENT
0000    42 ;
0000    43 ;--
0000    44         .SBTTL  HISTORY                 ; DETAILED
0000    45 ;
0000    46 ; AUTHOR: R. HUSTVEDT, CREATION DATE: 26-AUG-76
0000    47 ;
0000    48 ; MODIFIED BY:
0000    49 ;                   Dave Butenhof           12-may-1980, Version 5.4
0000    50 ;       01          Modify VMS source for supervisor
0000    51 ;                   Dave Butenhof           18-jun-1980, version 5.5
0000    52 ;       02          Restore buffered I/O support.
0000    53 ;
0000    54 ;                   Dave Butenhof   17-feb-1981, version 6.3
0000    55 ;       03          Change SEP psect to CODE, correct some reference truncation
0000    56 ;                   errors.
0000    57 ;       04          Bob Bergazzi    6-May,1983      Version 6.11
```

```
0000     58 ;                  Corrected truncation error.
0000     59 ;
0000     60 ;        V0211     KDM0103         KATHLEEN D. MORSE       13-MAR-1980
0000     61 ;                  REMOVE DUPLICATE MODIFICATION HISTORY.
0000     62 ;
0000     63 ;        V0210     RIH0059         RICHARD I. HUSTVEDT     25-FEB-1980
0000     64 ;                  CHANGE PRIORITY INCREMENT CLASS FOR PAGEFAULT TO GIVE
0000     65 ;                  SEPARATE AND LOWER BOOST.
0000     66 ;
0000     67 ;        V0209     KDM0094         KATHLEEN D. MORSE       22-JAN-1980     13:30
0000     68 ;                  ADD CHECK FOR DRIVER RETURNING TOTAL BYTE COUNT TRANSFERRED
0000     69 ;                  ON PAGE READ ERROR.  ASSUME THIS COUNT IS WRONG AND RESET
0000     70 ;                  IT TO ZERO BYTES TRANSFERRED.  THIS CAUSES THE FIRST PAGE OF
0000     71 ;                  THE TRANSFER TO BE THE PAGE WITH THE ERROR.
0000     72 ;
0000     73 ;        V0208     RIH0036         RICHARD I. HUSTVEDT     02-NOV-1979     09:36
0000     74 ;                  HONOR IRP$V_TERMIO AND IRP$V_FILACP FLAGS FOR PERFORMANCE
0000     75 ;                  IMPROVEMENT.
0000     76 ;
0000     77 ;        V0207     RIH0033         RICHARD I. HUSTVEDT     19-OCT-1979     15:32
0000     78 ;                  MOVE BYTCNT FROM PCB TO JIB.
0000     79 ;
0000     80 ;        V0206     RIH0032         RICHARD I. HUSTVEDT     10-SEP-1979     11:15
0000     81 ;                  ADD END ACTION DISPATCHING FOR INTERNAL PACKET COMPLETION.
0000     82 ;
0000     83 ;        V0205     KDM0054         KATHLEEN D. MORSE       31-AUG-1979     15:35
0000     84 ;                  FIX INTERFACE BUG IN CALLING PMS$START_RQ.
0000     85 ;
0000     86 ;        V0204     ACG23542        ANDREW C. GOLDSTEIN     04-MAY-1979     16:41
0000     87 ;                  CHECK LBN OF MAPPED VIRTUAL I/O AGAINST UCB$L_MAXBLOCK.
0000     88 ;
0000     89 ;        V0203     KDM0029         KATHLEEN D. MORSE       30-APR-1979     10:00
0000     90 ;                  ADD CODE TO MAKE $UPDSEC WORK FOR SHARED MEMORY.
0000     91 ;
0000     92 ;        V0202     SRB0001         STEVE BECKHARDT         29-MAR-1979     10:00
0000     93 ;                  ADDED CODE TO UNLOCK REGIONS SPECIFIED IN IRPE'S AND
0000     94 ;                  TO DEALLOCATE IRPE'S.
0000     95 ;
0000     96 ;        V0201     RIH21586        R. I. HUSTVEDT          12-JAN-1979     10:00
0000     97 ;                  CORRECTED BUFFER PROBING FOR BUFFERED I/O COMPLETION.
0000     98 ;
```

```
                        0000   100          .SBTTL  DECLARATIONS
                        0000   101 ;
                        0000   102 ; INCLUDE FILES:
                        0000   103 ;
                        0000   104          $ACBDEF                         ; AST CONTROL BLOCK DEFINITIONS
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .IIF    NB,ACB$L_ASTQFL,        ACB$L_ASTQFL:
                00000004 0000                      .BLKL   1
                        0004               .IIF    NB,ACB$L_ASTQBL,        ACB$L_ASTQBL:
                00000008 0004                      .BLKL   1
                        0008               .IIF    NB,ACB$W_SIZE,  ACB$W_SIZE:
                0000000A 0008                      .BLKW   1
                        000A               .IIF    NB,ACB$B_TYPE,  ACB$B_TYPE:
                0000000B 000A                      .BLKB   1
                        000B               .IIF    NB,ACB$B_RMOD,  ACB$B_RMOD:
                0000000C 000B                      .BLKB   1
                        000C               .IIF    NB,ACB$L_PID,   ACB$L_PID:
                00000010 000C                      .BLKL   1
                        0010               .IIF    NB,ACB$L_AST,   ACB$L_AST:
                00000014 0010                      .BLKL   1
                        0014               .IIF    NB,ACB$L_ASTPRM,        ACB$L_ASTPRM:
                00000018 0014                      .BLKL   1
                        0018               .IIF    NB,ACB$C_LENGTH,        ACB$C_LENGTH:
                        0018               .IIF    NB,ACB$K_LENGTH,        ACB$K_LENGTH:
                        0018               .IIF    NB,ACB$L_KAST,  ACB$L_KAST:
                0000001C 0018                      .BLKL   1
                        0000               .RESTORE
                        0000   105          $AQBDEF                         ; DEFINE AQB OFFSETS
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                00000000 001C              .=0
                        0000               .RESTORE
                        0000   106          $CADEF                          ; CONDITIONAL ASSEMBLY PARAMETERS
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                00000000 001C              .=0
                        0000               .RESTORE
                        0000   107          $CCBDEF                         ; CCB DEFINITIONS
                        0000               .SAVE   LOCAL_BLOCK
                        0000               .PSECT  $ABS$,ABS
                00000000 001C              .=0
                        0000               .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
                00000004 0000                      .BLKL   1
                        0004               .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
                00000008 0004                      .BLKL   1
                        0008               .IIF    NB,CCB$B_STS,   CCB$B_STS:
                00000009 0008                      .BLKB   1
                        0009               .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
                0000000A 0009                      .BLKB   1
                        000A               .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
                0000000C 000A                      .BLKW   1
                        000C               .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
                00000010 000C                      .BLKL   1
                        0010               .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                        0010               .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                        0000               .RESTORE
                        0000   108          $CXBDEF                         ; DEFINE CXB OFFSETS
```

```
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 001C               .=0
                        0000                .RESTORE
                        0000    109         $IPLDEF                             ; IPL DEFINITIONS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 001C               .=0
                        0000                .RESTORE
                        0000    110         $IRPDEF                             ; IRP DEFINITIONS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 001C               .=0
                        0000                .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
                00000004 0000                       .BLKL   1
                        0004                .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
                00000008 0004                       .BLKL   1
                        0008                .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
                0000000A 0008                       .BLKW   1
                        000A                .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
                0000000B 000A                       .BLKB   1
                        000B                .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
                0000000C 000B                       .BLKB   1
                        000C                .IIF    NB,IRP$L_PID,   IRP$L_PID:
                00000010 000C                       .BLKL   1
                        0010                .IIF    NB,IRP$L_AST,   IRP$L_AST:
                00000014 0010                       .BLKL   1
                        0014                .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
                00000018 0014                       .BLKL   1
                        0018                .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
                0000001C 0018                       .BLKL   1
                        001C                .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
                00000020 001C                       .BLKL   1
                        0020                .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
                00000022 0020                       .BLKW   1
                        0022                .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
                00000023 0022                       .BLKB   1
                        0023                .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
                00000024 0023                       .BLKB   1
                        0024                .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
                00000028 0024                       .BLKL   1
                        0028                .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
                0000002A 0028                       .BLKW   1
                        002A                .IIF    NB,IRP$W_STS,   IRP$W_STS:
                0000002C 002A                       .BLKW   1
                        002C                .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
                00000030 002C                       .BLKL   1
                        0030                .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
                00000032 0030                       .BLKW   1
                        0032                .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
                00000034 0032                       .BLKW   1
                        0034                .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
                        0034                .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
                00000038 0034                       .BLKL   1
                        0038                .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
                        0038                .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
                        0038                .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
```

```
00000039  0038                          .BLKB   1
0000003C  0039                          .BLKB   3
          003C              .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
          003C              .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
          003C              .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                          .BLKW   1
          003E              .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                          .BLKW
          0040              .IIF    NB,IRP$L_SEQVBN,        IRP$L_SEQVBN:
00000044  0040                          .BLKL   1
          0044              .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                          .BLKL   1
          0048              .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                          .BLKL   1
          004C              .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                          .BLKL   1
          0050              .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                          .BLKL   1
00000058  0054                          .BLKL   1
0000005C  0058                          .BLKL   1
          005C              .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
          005C              .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
          0000              .RESTORE
          0000    111       $IRPEDEF                        ; IRPE DEFINITIONS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  005C              .=0
00000004  0000                          .BLKL   1
00000008  0004                          .BLKL   1
          0008              .IIF    NB,IRPE$W_SIZE, IRPE$W_SIZE:
0000000A  0008                          .BLKW   1
          000A              .IIF    NB,IRPE$B_TYPE, IRPE$B_TYPE:
0000000B  000A                          .BLKB   1
0000000C  000B                          .BLKB   1
00000028  000C                          .BLKL   7
0000002A  0028                          .BLKW   1
          002A              .IIF    NB,IRPE$W_STS,  IRPE$W_STS:
0000002C  002A                          .BLKW   1
          002C              .IIF    NB,IRPE$L_SVAPTE1,      IRPE$L_SVAPTE1:
00000030  002C                          .BLKL   1
          0030              .IIF    NB,IRPE$W_BOFF1,        IRPE$W_BOFF1:
00000032  0030                          .BLKW   1
00000034  0032                          .BLKW   1
          0034              .IIF    NB,IRPE$L_BCNT1,        IRPE$L_BCNT1:
00000038  0034                          .BLKL   1
          0038              .IIF    NB,IRPE$L_SVAPTE2,      IRPE$L_SVAPTE2:
0000003C  0038                          .BLKL   1
          003C              .IIF    NB,IRPE$W_BOFF2,        IRPE$W_BOFF2:
0000003E  003C                          .BLKW   1
00000040  003E                          .BLKW   1
          0040              .IIF    NB,IRPE$L_BCNT2,        IRPE$L_BCNT2:
00000044  0040                          .BLKL   1
0000004C  0044                          .BLKL   2
          004C              .IIF    NB,IRPE$L_EXTEND,       IRPE$L_EXTEND:
00000050  004C                          .BLKL   1
          0050              .IIF    NB,IRPE$C_LENGTH,       IRPE$C_LENGTH:
          0050              .IIF    NB,IRPE$K_LENGTH,       IRPE$K_LENGTH:
```

```
                     0000                .RESTORE
                     0000    112         $JIBDEF                         ; JIB DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 005C                .=0
                     0000                .RESTORE
                     0000    113         $PCBDEF                         ; PCB DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
            00000000 005C                .=0
                     0000                .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
            00000004 0000                        .BLKL   1
                     0004                .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
            00000008 0004                        .BLKL   1
                     0008                .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
            0000000A 0008                        .BLKW   1
                     000A                .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
            0000000B 000A                        .BLKB   1
                     000B                .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
            0000000C 000B                        .BLKB   1
                     000C                .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
            0000000D 000C                        .BLKB   1
                     000D                .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
            0000000E 000D                        .BLKB   1
                     000E                .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
            00000010 000E                        .BLKW   1
                     0010                .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
            00000014 0010                        .BLKL   1
                     0014                .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
            00000018 0014                        .BLKL   1
                     0018                .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
            0000001C 0018                        .BLKL   1
                     001C                .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
            00000020 001C                        .BLKL   1
                     0020                .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
            00000024 0020                        .BLKL   1
                     0024                .IIF    NB,PCB$L_STS,   PCB$L_STS:
            00000028 0024                        .BLKL   1
                     0028                .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
            0000002C 0028                        .BLKL   1
                     002C                .IIF    NB,PCB$W_STATE, PCB$W_STATE:
            0000002E 002C                        .BLKW   1
                     002E                .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
            0000002F 002E                        .BLKB   1
                     002F                .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
            00000030 002F                        .BLKB   1
                     0030                .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
            00000032 0030                        .BLKW   1
                     0032                .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
            00000034 0032                        .BLKW   1
                     0034                .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
            00000036 0034                        .BLKW   1
                     0036                .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
            00000038 0036                        .BLKW   1
                     0038                .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
            0000003A 0038                        .BLKW   1
                     003A                .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
```

```
0000003C  003A                          .BLKW   1
          003C          .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                          .BLKW   1
          003E          .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                          .BLKW   1
          0040          .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                          .BLKW   1
          0042          .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                          .BLKW   1
          0044          .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                          .BLKB   8
          004C          .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
          004C          .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                          .BLKL   1
          0050          .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                          .BLKL   1
          0054          .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                          .BLKL   1
          0058          .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                          .BLKL   1
          005C          .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                          .BLKL   1
          0060          .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                          .BLKL   1
          0064          .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                          .BLKL   1
          0068          .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                          .BLKB   16
          0078          .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084  007C                          .BLKQ   1
          0084          .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
          0088          .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                          .BLKW   1
          008A          .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                          .BLKW   1
          008C          .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
          008C          .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
          0000          .RESTORE
          0000  114     $PFNDEF                         ; PFN DATA BASE DEFINITIONS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .RESTORE
          0000  115     $PHDDEF                         ; PROCESS HEADER DEFINITIONS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .RESTORE
          0000  116     $PRDEF                          ; PROCESSOR REGISTER DEFINITIONS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .RESTORE
```

```
                     0000   117          $PRIDEF                                 ; PRIORITY INCREMENT DEFS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
                00000000  009C           .=0
                     0000                .RESTORE
                     0000   118          $PTEDEF                                 ; PAGE TABLE ENTRY DEFINITIONS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
                00000000  008C           .=0
                     0000                .RESTORE
                     0000   119          $RSNDEF                                 ; DEFINE RESOURCE WAIT NUMBERS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
                00000000  008C           .=0
                     0000                .RESTORE
                     0000   120          $UCBDEF                                 ; DEFINE UCB OFFSETS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
                00000000  008C           .=0
                     0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                     0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                00000004  0000                                   .BLKL   1
                     0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                     0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                00000008  0004                                   .BLKL   1
                     0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                0000000A  0008                                   .BLKW   1
                     000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                0000000B  000A                                   .BLKB   1
                     000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                0000000C  000B                                   .BLKB   1
                     000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                     000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                     000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                0000000D  000C                                   .BLKB   1
                00000010  000D                                   .BLKB   3
                     0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                     0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
                00000014  0010                                   .BLKL   1
                     0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                     0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                     0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
                00000016  0014                                   .BLKW   1
                     0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
                00000018  0016                                   .BLKW   1
                     0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                     0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
                0000001A  0018                                   .BLKW   1
                     001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                     001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                0000001C  001A                                   .BLKW   1
                     001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                00000020  001C                                   .BLKL   1
                     0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                00000024  0020                                   .BLKL   1
                     0024                .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                00000028  0024                                   .BLKL   1
```

```
                    0028            .IIF    NB,UCB$L_PID,   UCB$L_PID:
0000002C  0028                              .BLKL   1
                    002C            .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030  002C                              .BLKL   1
                    0030            .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034  0030                              .BLKL   1
                    0034            .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038  0034                              .BLKL   1
                    0038            .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039  0038                              .BLKB   1
                    0039            .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A  0039                              .BLKB   1
                    003A            .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C  003A                              .BLKW   1
                    003C            .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C            .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C            .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D  003C                              .BLKB   1
                    003D            .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                    003D            .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E  003D                              .BLKB   1
                    003E            .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                    003E            .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F  003E                              .BLKB   1
                    003F            .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040  003F                              .BLKB   1
                    0040            .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                              .BLKL   1
                    0044            .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                              .BLKL   1
                    0048            .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                              .BLKW   1
                    004A            .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                    004A            .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                              .BLKB   1
                    004B            .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                              .BLKB   1
                    004C            .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                              .BLKL   1
                    0050            .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                              .BLKW   1
                    0052            .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                    0052            .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                              .BLKB   1
                    0053            .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                              .BLKB   1
                    0054            .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                              .BLKL   1
                    0058            .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                              .BLKW   1
                    005A            .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                              .BLKW   1
                    005C            .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                    005C            .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                              .BLKL   1
                    0060            .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                              .BLKL   1
```

```
                    0064            .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                    0064            .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068            0064                            .BLKL   1
                    0068            .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C            0068                            .BLKL   1
                    006C            .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E            006C                            .BLKW   1
                    006E            .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070            006E                            .BLKW   1
                    0070            .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071            0070                            .BLKB   1
                    0071            .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072            0071                            .BLKB   1
                    0072            .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074            0072                            .BLKW   1
                    0074            .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                    0074            .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000            0074            . = 0
                    0000            .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002            0000                            .BLKW   1
00000074            0002            . = 116
                    0074            .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078            0074                            .BLKL   1
                    0078            .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C            0078                            .BLKL   1
                    007C            .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080            007C                            .BLKL   1
                    0080            .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084            0080                            .BLKL   1
                    0084            .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088            0084                            .BLKL   1
                    0088            .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C            0088                            .BLKL   1
                    008C            .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090            008C                            .BLKL   1
                    0090            .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                    0090            .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074            0090            . = 116
                    0074            .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075            0074                            .BLKB   1
                    0075            .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076            0075                            .BLKB   1
                    0076            .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077            0076                            .BLKB   1
                    0077            .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078            0077                            .BLKB   1
                    0078            .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C            0078                            .BLKL   1
0000007E            007C                            .BLKW   1
                    007E            .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080            007E                            .BLKW   1
                    0080            .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084            0080                            .BLKL   1
00000080            0084            . = 128
00000084            0080                            .BLKL   1
                    0084            .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088            0084                            .BLKL   1
```

```
                        0088            .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                                  .BLKW   1
                        008A            .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                                  .BLKW   1
                        008C            .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                        008C            .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                  .BLKW   1
                        008E            .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                  .BLKW   1
                        0090            .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                  .BLKW   1
                        0092            .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                  .BLKW   1
                        0094            .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                  .BLKB   1
                        0095            .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                  .BLKB   1
                        0096            .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                  .BLKW   1
00000096  0098          . = 150
00000098  0096                                  .BLKW   1
                        0098            .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                  .BLKL   1
                        009C            .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                  .BLKL   1
                        00A0            .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                                  .BLKL   1
                        00A4            .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                                  .BLKW   1
                        00A6            .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                                  .BLKB   1
000000A8  00A7                                  .BLKB   1
00000074  00A8          . = 116
00000078  0074                                  .BLKL   1
0000007C  0078                                  .BLKL   1
00000080  007C                                  .BLKL   1
00000084  0080                                  .BLKL   1
00000088  0084                                  .BLKL   1
0000008C  0088                                  .BLKL   1
                        008C            .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                  .BLKL   1
00000094  0090                                  .BLKL   1
00000098  0094                                  .BLKL   1
0000009C  0098                                  .BLKL   1
                        009C            .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                  .BLKB   1
                        009D            .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                  .BLKB   1
                        009E            .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                  .BLKB   1
000000A0  009F                                  .BLKB   1
                        00A0            .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                  .BLKB   1
                        00A1            .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                  .BLKB   1
                        00A2            .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                  .BLKB   1
```

```
000000A4  00A3                          .BLKB   1
          00A4          .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                          .BLKB   1
          00A5          .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                          .BLKW   1
000000A8  00A7                          .BLKB   1
          00A8          .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                          .BLKL   1
000000B0  00AC                          .BLKL   1
000000B4  00B0                          .BLKL   1
000000B8  00B4                          .BLKL   1
          00B8          .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                          .BLKL   1
          00BC          .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC          .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC          . = 116
          0074          .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                          .BLKL   1
          0078          .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                          .BLKW   1
00000080  007E                          .BLKW   1
          0000          .RESTORE
          0000   121    $VADEF                          ; DEFINE VIRTUAL ADDRESS FIELDS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  00BC          .=0
          0000          .RESTORE
          0000   122    $VCBDEF                         ; DEFINE VCB OFFSETS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  00BC          .=0
          0000          .IIF    NB,VCB$L_FCBFL, VCB$L_FCBFL:
00000004  0000                  .BLKL   1
          0004          .IIF    NB,VCB$L_FCBBL, VCB$L_FCBBL:
00000008  0004                  .BLKL   1
          0008          .IIF    NB,VCB$W_SIZE,  VCB$W_SIZE:
0000000A  0008                  .BLKW   1
          000A          .IIF    NB,VCB$B_TYPE,  VCB$B_TYPE:
0000000B  000A                  .BLKB   1
          000B          .IIF    NB,VCB$C_MRKLEN,        VCB$C_MRKLEN:
          000B          .IIF    NB,VCB$K_MRKLEN,        VCB$K_MRKLEN:
          000B          .IIF    NB,VCB$B_STATUS,        VCB$B_STATUS:
0000000C  000B                  .BLKB   1
          000C          .IIF    NB,VCB$W_TRANS, VCB$W_TRANS:
0000000E  000C                  .BLKW   1
          000E          .IIF    NB,VCB$W_RVN,   VCB$W_RVN:
00000010  000E                  .BLKW   1
          0010          .IIF    NB,VCB$L_AQB,   VCB$L_AQB:
00000014  0010                  .BLKL   1
          0014          .IIF    NB,VCB$T_VOLNAME,       VCB$T_VOLNAME:
00000020  0014                  .BLKB   12
          0020          .IIF    NB,VCB$L_RVT,   VCB$L_RVT:
00000024  0020                  .BLKL   1
          0024          .IIF    NB,VCB$C_COMLEN,        VCB$C_COMLEN:
          0024          .IIF    NB,VCB$K_COMLEN,        VCB$K_COMLEN:
```

```
                        0024            .IIF    NB,VCB$L_HOMELBN,       VCB$L_HOMELBN:
00000028                0024                    .BLKL   1
                        0028            .IIF    NB,VCB$L_HOME2LBN,      VCB$L_HOME2LBN:
0000002C                0028                    .BLKL   1
                        002C            .IIF    NB,VCB$L_IXHDR2LBN,     VCB$L_IXHDR2LBN:
00000030                002C                    .BLKL   1
                        0030            .IIF    NB,VCB$L_IBMAPLBN,      VCB$L_IBMAPLBN:
00000034                0030                    .BLKL   1
                        0034            .IIF    NB,VCB$L_SBMAPLBN,      VCB$L_SBMAPLBN:
00000038                0034                    .BLKL   1
                        0038            .IIF    NB,VCB$B_IBMAPSIZE,     VCB$B_IBMAPSIZE:
00000039                0038                    .BLKB   1
                        0039            .IIF    NB,VCB$B_SBMAPSIZE,     VCB$B_SBMAPSIZE:
0000003A                0039                    .BLKB   1
                        003A            .IIF    NB,VCB$B_IBMAPVBN,      VCB$B_IBMAPVBN:
0000003B                003A                    .BLKB   1
                        003B            .IIF    NB,VCB$B_SBMAPVBN,      VCB$B_SBMAPVBN:
0000003C                003B                    .BLKB   1
                        003C            .IIF    NB,VCB$W_CLUSTER,       VCB$W_CLUSTER:
0000003E                003C                    .BLKW   1
                        003E            .IIF    NB,VCB$W_EXTEND,        VCB$W_EXTEND:
00000040                003E                    .BLKW   1
                        0040            .IIF    NB,VCB$L_FREE,  VCB$L_FREE:
00000044                0040                    .BLKL   1
                        0044            .IIF    NB,VCB$L_MAXFILES,      VCB$L_MAXFILES:
00000048                0044                    .BLKL   1
                        0048            .IIF    NB,VCB$B_WINDOW,        VCB$B_WINDOW:
00000049                0048                    .BLKB   1
                        0049            .IIF    NB,VCB$B_LRU_LIM,       VCB$B_LRU_LIM:
0000004A                0049                    .BLKB   1
                        004A            .IIF    NB,VCB$W_FILEPROT,      VCB$W_FILEPROT:
0000004C                004A                    .BLKW   1
                        004C            .IIF    NB,VCB$W_MCOUNT,        VCB$W_MCOUNT:
0000004E                004C                    .BLKW   1
                        004E            .IIF    NB,VCB$B_EOFDELTA,      VCB$B_EOFDELTA:
0000004F                004E                    .BLKB   1
                        004F            .IIF    NB,VCB$B_RESFILES,      VCB$B_RESFILES:
00000050                004F                    .BLKB   1
                        0050            .IIF    NB,VCB$W_RECORDSZ,      VCB$W_RECORDSZ:
00000052                0050                    .BLKW   1
                        0052            .IIF    NB,VCB$B_BLOCKFACT,     VCB$B_BLOCKFACT:
00000053                0052                    .BLKB   1
                        0053            .IIF    NB,VCB$B_STATUS2,       VCB$B_STATUS2:
00000054                0053                    .BLKB   1
                        0054            .IIF    NB,VCB$L_QUOTAFCB,      VCB$L_QUOTAFCB:
00000058                0054                    .BLKL   1
                        0058            .IIF    NB,VCB$L_CACHE, VCB$L_CACHE:
0000005C                0058                    .BLKL   1
                        005C            .IIF    NB,VCB$L_QUOCACHE,      VCB$L_QUOCACHE:
00000060                005C                    .BLKL   1
                        0060            .IIF    NB,VCB$W_QUOSIZE,       VCB$W_QUOSIZE:
00000062                0060                    .BLKW   1
                        0062            .IIF    NB,VCB$W_PENDERR,       VCB$W_PENDERR:
00000064                0062                    .BLKW   1
                        0064            .IIF    NB,VCB$C_LENGTH,        VCB$C_LENGTH:
                        0064            .IIF    NB,VCB$K_LENGTH,        VCB$K_LENGTH:
00000000                0064            . = 0
```

```
                    0000            .IIF    NB,VCB$L_BLOCKFL,       VCB$L_BLOCKFL:
00000004            0000                    .BLKL   1
                    0004            .IIF    NB,VCB$L_BLOCKBL,       VCB$L_BLOCKBL:
00000008            0004                    .BLKL   1
0000000B            0008            . = 11
00000024            000B            . = 36
                    0024            .IIF    NB,VCB$L_CUR_FID,       VCB$L_CUR_FID:
                    0024            .IIF    NB,VCB$W_CUR_NUM,       VCB$W_CUR_NUM:
00000026            0024                    .BLKW   1
                    0026            .IIF    NB,VCB$W_CUR_SEQ,       VCB$W_CUR_SEQ:
00000028            0026                    .BLKW   1
                    0028            .IIF    NB,VCB$L_START_FID,     VCB$L_START_FID:
                    0028            .IIF    NB,VCB$W_START_NUM,     VCB$W_START_NUM:
0000002A            0028                    .BLKW   1
                    002A            .IIF    NB,VCB$W_START_SEQ,     VCB$W_START_SEQ:
0000002C            002A                    .BLKW   1
                    002C            .IIF    NB,VCB$W_MODE,  VCB$W_MODE:
0000002E            002C                    .BLKW   1
                    002E            .IIF    NB,VCB$B_TM,    VCB$B_TM:
0000002F            002E                    .BLKB   1
                    002F            .IIF    NB,VCB$B_CUR_RVN,       VCB$B_CUR_RVN:
00000030            002F                    .BLKB   1
                    0030            .IIF    NB,VCB$L_ST_RECORD,     VCB$L_ST_RECORD:
00000034            0030                    .BLKL   1
                    0034            .IIF    NB,VCB$L_MVL,   VCB$L_MVL:
00000038            0034                    .BLKL   1
                    0038            .IIF    NB,VCB$L_WCB,   VCB$L_WCB:
0000003C            0038                    .BLKL   1
                    003C            .IIF    NB,VCB$L_VPFL,  VCB$L_VPFL:
00000040            003C                    .BLKL   1
                    0040            .IIF    NB,VCB$L_VPBL,  VCB$L_VPBL:
00000044            0040                    .BLKL   1
                    0044            .IIF    NB,VCB$L_USRLBLAST,     VCB$L_USRLBLAST:
00000048            0044                    .BLKL   1
0000000B            0048            . = 11
                    000B            .IIF    NB,VCB$B_QNAMECNT,      VCB$B_QNAMECNT:
0000000C            000B                    .BLKB   1
                    000C            .IIF    NB,VCB$T_QNAME, VCB$T_QNAME:
00000020            000C                    .BLKB   20
                    0000            .RESTORE
                    0000    123     $WCBDEF                         ; DEFINE WCB OFFSETS
                    0000            .SAVE   LOCAL_BLOCK
                    0000            .PSECT  $ABS$,ABS
00000000            00BC            .=0
                    0000            .IIF    NB,WCB$L_WLFL,  WCB$L_WLFL:
00000004            0000                    .BLKL   1
                    0004            .IIF    NB,WCB$L_WLBL,  WCB$L_WLBL:
00000008            0004                    .BLKL   1
                    0008            .IIF    NB,WCB$W_SIZE,  WCB$W_SIZE:
0000000A            0008                    .BLKW   1
                    000A            .IIF    NB,WCB$B_TYPE,  WCB$B_TYPE:
0000000B            000A                    .BLKB   1
                    000B            .IIF    NB,WCB$B_ACCESS,        WCB$B_ACCESS:
0000000C            000B                    .BLKB   1
                    000C            .IIF    NB,WCB$L_PID,   WCB$L_PID:
0000000E            000C                    .BLKB   2
                    000E            .IIF    NB,WCB$W_REFCNT,        WCB$W_REFCNT:
```

```
00000010  000E                        .BLKW   1
          0010              .IIF    NB,WCB$L_ORGUCB,        WCB$L_ORGUCB:
00000014  0010                        .BLKL   1
          0014              .IIF    NB,WCB$W_ACON,  WCB$W_ACON:
00000016  0014                        .BLKW   1
          0016              .IIF    NB,WCB$W_NMAP,  WCB$W_NMAP:
00000018  0016                        .BLKW   1
          0018              .IIF    NB,WCB$L_FCB,   WCB$L_FCB:
0000001C  0018                        .BLKL   1
          001C              .IIF    NB,WCB$L_RVT,   WCB$L_RVT:
00000020  001C                        .BLKL   1
          0020              .IIF    NB,WCB$L_STVBN, WCB$L_STVBN:
00000024  0020                        .BLKL   1
          0024              .IIF    NB,WCB$C_MAP,   WCB$C_MAP:
          0024              .IIF    NB,WCB$K_MAP,   WCB$K_MAP:
          0024              .IIF    NB,WCB$C_LENGTH,        WCB$C_LENGTH:
          0024              .IIF    NB,WCB$K_LENGTH,        WCB$K_LENGTH:
          0024              .IIF    NB,WCB$W_P1_COUNT,      WCB$W_P1_COUNT:
00000026  0024                        .BLKW   1
          0026              .IIF    NB,WCB$L_P1_LBN,        WCB$L_P1_LBN:
0000002A  0026                        .BLKL   1
          002A              .IIF    NB,WCB$W_P2_COUNT,      WCB$W_P2_COUNT:
0000002C  002A                        .BLKW   1
          002C              .IIF    NB,WCB$L_P2_LBN,        WCB$L_P2_LBN:
00000030  002C                        .BLKL   1
00000000  0030              . = 0
          0000              .IIF    NB,WCB$W_COUNT, WCB$W_COUNT:
00000002  0000                        .BLKW   1
          0002              .IIF    NB,WCB$L_LBN,   WCB$L_LBN:
00000006  0002                        .BLKL   1
00000000  0006              . = 0
FFFFFFFA  0000                        .BLKB   -6
          FFFA              .IIF    NB,WCB$W_PREVCOUNT,     WCB$W_PREVCOUNT:
FFFFFFFC  FFFA                        .BLKW   1
          FFFC              .IIF    NB,WCB$L_PREVLBN,       WCB$L_PREVLBN:
00000000  FFFC                        .BLKL   1
          0000              .RESTORE
          0000    124       $WQHDEF                         ; WAIT QUEUE HEADER DEFINITIONS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  FFFC              .=0
          0000              .IIF    NB,WQH$L_WQFL,  WQH$L_WQFL:
00000004  0000                        .BLKL   1
          0004              .IIF    NB,WQH$L_WQBL,  WQH$L_WQBL:
00000008  0004                        .BLKL   1
          0008              .IIF    NB,WQH$W_WQCNT, WQH$W_WQCNT:
0000000A  0008                        .BLKW   1
          000A              .IIF    NB,WQH$W_WQSTATE,       WQH$W_WQSTATE:
0000000C  000A                        .BLKW   1
          000C              .IIF    NB,WQH$C_LENGTH,        WQH$C_LENGTH:
          000C              .IIF    NB,WQH$K_LENGTH,        WQH$K_LENGTH:
          0000              .RESTORE
          0000    125
          0000    126 ;
          0000    127 ; OWN STORAGE:
          0000    128 ;
     00000000     129       .PSECT  DATA, SHR, NOEXE, NOWRT, LONG
```

```
                        0000   130          MODNAM  IOPOST
          54 53 4F 50 4F 49 00' 0000       $MODULE:        .ASCIC  \IOPOST\
                        06  0000
    52 4F 52 52 45 20 50 43 41 00' 0007   131 AACPERR:.ASCIC  .ACP ERROR.
                        09  0007
                        00000000   132          .PSECT  CODE, SHR, EXE, LONG, NOWRT
                            0000   133 PRITBL:                          ; TABLE OF PRIORITY INCR CLASSES
                        01  0000   134          .BYTE   PRI$_IOCOM      ; 0 => DIRECT WRITE
                        03  0001   135          .BYTE   PRI$_TOCOM      ; 1 => BUFFERED WRITE
                        01  0002   136          .BYTE   PRI$_IOCOM      ; 2 => DIRECT READ
                        04  0003   137          .BYTE   PRI$_TICOM      ; 3 => BUFFERED READ
```

```
                                0004          .SBTTL  I/O COMPLETION POSTING
                                0004    ;++
                                0004    ; FUNCTIONAL DESCRIPTION:
                                0004    ;
                                0004    ;       IOC$IOPOST IS INITIATED BY TRIGGERING AN IPL$_IOPOST SOFTWARE
                                0004    ;       INTERRUPT AFTER PLACING A COMPLETED I/O PACKET IN THE IOPOST
                                0004    ;       QUEUE.  IOC$IOPOST PERFORMS ALL APPROPRIATE COMPLETION ACTIVITY
                                0004    ;       REQUIRED FOR THE PACKET EITHER DIRECTLY OR BY QUEUEING KERNEL
                                0004    ;       ASTS TO CONCLUDE PROCESSING IN THE CONTEXT OF THE PROCESS
                                0004    ;       WHEN REQUIRED.
                                0004    ;
                                0004    ; CALLING SEQUENCE:
                                0004    ;
                                0004    ;       SOFTINT #IPL$_IOPOST
                                0004    ;
                                0004    ; INPUT PARAMETERS:
                                0004    ;
                                0004    ;       NONE
                                0004    ;
                                0004    ; IMPLICIT INPUTS:
                                0004    ;
                                0004    ;       IOC$GL_PSFL - IOPOSTING QUEUE
                                0004    ;
                                0004    ; OUTPUT PARAMETERS:
                                0004    ;
                                0004    ;       NONE
                                0004    ;
                                0004    ;--
                                0004
                                0004          .ENABL  LSB
                                0004    IOC$IOPOST::                                    ; I/O POSTING INTERRUPT
          7E    54    7D        0004          MOVQ    R4,-(SP)                        ; SAVE
          7E    52    7D        0007          MOVQ    R2,-(SP)                        ; NORMAL
          7E    50    7D        000A          MOVQ    R0,-(SP)                        ; REGISTERS
   55   00000000'FF    0F        000D    IOPOST: REMQUE  @IOC$GL_PSFL,R5                 ; GET HEAD OF POST QUEUE
                11    1C        0014          BVC     10$                             ; QUEUE NOT YET EMPTY
          50    8E    7D        0016          MOVQ    (SP)+,R0                        ; RESTORE
          52    8E    7D        0019          MOVQ    (SP)+,R2                        ; REGISTERS
          54    8E    7D        001C          MOVQ    (SP)+,R4                        ; AND EXIT
                      02        001F          REI                                     ; IF QUEUE EMPTY
                                0020
             00C5     31        0020    5$:     BRW     VIRTUAL                         ; PROCESS VIRTUAL I/O COMPLETION
                                0023
                61    16        0023    7$:     JSB     (R1)                            ; CALL END ACTION ROUTINE
                E6    11        0025          BRB     IOPOST                          ;
                                0027
       51    0C A5    D0        0027    10$:    MOVL    IRP$L_PID(R5),R1                ; GET PID/END ACTION ADDRESS
                F6    19        002B          BLSS    7$                              ; BR IF END ACTION ADDRESS
                                002D                                                  ; (SYSTEM SPACE ADDRESSES ARE NEGATIVE)
          51    51    3C        002D          MOVZWL  R1,R1                           ; GET PROCESS INDEX
 54   00000000'FF41    D0        0030          MOVL    @L^SCH$GL_PCBVEC[R1],R4         ; And translate to PCB address
      03 2A A5    00    E1        0038          BBC     #IRP$V_BUFIO,IRP$W_STS(R5),12$  ; IF CLEAR, DIRECT I/O
             0056     31        003D          BRW     BUFIO                           ; BUFFERED I/O
          3E A4       B6        0040    12$:    INCW    PCB$W_DIOCNT(R4)                ; UPDATE DIRECT I/O COUNT
       53    2C A5    D0        0043          MOVL    IRP$L_SVAPTE(R5),R3             ; GET ADDRESS OF FIRST PTE
                                0047
                                0047          ASSUME  IRP$V_PAGIO LE 7
```

```
                        0047   196          ASSUME  IRP$V_SWAPIO LE 7
          2A A5  44 8F   93    0047   197          BITB    #<IRP$M_PAGIO ! IRP$M_SWAPIO>,IRP$W_STS(R5) ; PAGIO OR SWAPIO?
                 37     12    004C   198          BNEQ    PAGIO_OR_SWAPIO
                              004E   199
                              004E   200 ;
                              004E   201 ; DIRECT I/O COMPLETION
                              004E   202 ;
                              004E   203
                 53     D5    004E   204 DIRIO:  TSTL    R3                      ; PTE ADDRESS VALID?
                 29     13    0050   205          BEQL    14$                     ; IF EQL NO PAGES TO UNLOCK
             51  32 A5  3C    0052   206          MOVZWL  IRP$W_BCNT(R5),R1       ; GET REQUESTED TRANSFER BYTE COUNT
             52  30 A5  3C    0056   207          MOVZWL  IRP$W_BOFF(R5),R2       ; GET BYTE OFFSET IN PAGE
       0B 2A A5     04    E1    005A   208          BBC     #IRP$V_VIRTUAL,IRP$W_STS(R5),UNLOCK ; BRANCH IF NOT VIRTUAL I/O
          BD 34 A5  E9    005F   209          BLBC    IRP$L_IOST1(R5),5$      ; BRANCH IF ERROR IN VIRTUAL REQUEST
       3E A5  36 A5  B1    0063   210          CMPW    IRP$L_IOST1+2(R5),IRP$W_OBCNT(R5) ; IF COMPLETED ORIGINAL BYTE COUNT
                              0068   211                                          ; THEN NO SPECIAL VIRTUAL PROCESSING
                 B6     12    0068   212          BNEQ    5$                      ; OTHERWISE DO THE SEGMENTED COMPLETION
        51   01FF C142  9E    006A   213 UNLOCK: MOVAB   511(R1)[R2],R1          ; COMBINE OFFSET AND COUNT AND ROUND
      51   51   F7 8F   78    0070   214          ASHL    #-VA$S_BYTE,R1,R1       ; CONVERT TO NUMBER OF PAGES
         00000000'EF     16    0075   215          JSB     L^MMG$UNLOCK            ; Unlock pages
       03 2A A5   0B     E1    007B   216 14$:    BBC     #IRP$V_EXTEND,IRP$W_STS(R5),15$ ; BRANCH IF NO IRPE'S ATTACHED
                02FE     30    0080   217          BSBW    UNLOCK_MORE             ; UNLOCK AREAS DESCRIBED IN IRPE'S
                              0083   218 15$:                                     ; REFERENCE LABEL
                 30     11    0083   219          BRB     30$                     ;
                              0085   220
                              0085   221 ;
                              0085   222 ;       PAGE I/O OR SWAP I/O COMPLETION
                              0085   223 ;
                              0085   224
                              0085   225 PAGIO_OR_SWAPIO:
                              0085   226          $ERRSUP_S
          00000000'EF   DF    0085                        PUSHAL  $MODULE
                   00   DD    008B                        PUSHL   #0
                   01   DD    008D                        PUSHL   #$ER
      00000000'EF      03   FB    008F                        CALLS   #3, DS_ERRSUP
                              0096   227 ;
                              0096   228 ; BUFFERED I/O COMPLETION
                              0096   229 ;
                              0096   230
             3A A4    B6    0096   231 BUFIO:  INCW    PCB$W_BIOCNT(R4)        ; UPDATE BUFFERED I/O COUNT
       03 2A A5   0C     E1    0099   232          BBC     #IRP$V_FILACP,IRP$W_STS(R5),NOTACP ; BR IF NOT ACP I/O
             3E A4    B6    009E   233          INCW    PCB$W_DIOCNT(R4)        ; RESTORE DIRECT I/O COUNT
                              00A1   234 NOTACP:                                  ;
          50   2C A5   D0    00A1   235          MOVL    IRP$L_SVAPTE(R5),R0     ; ANY BUFFER SPECIFIED?
                 0E     13    00A5   236          BEQL    30$                     ; IF EQL NO
       18 A5   01E8'CF   9E    00A7   237          MOVAB   W^BUFPOST,ACB$L_KAST(R5) ; ASSUME READ FUNCTION
         09 2A A5   01   E0    00AD   238          BBS     #IRP$V_FUNC,IRP$W_STS(R5),40$ ; IF SET, READ FUNCTION
                FF4B'    30    00B2   239          BSBW    EXE$DEANONPAGED         ; DEALLOCATE WRITE BUFFER
       18 A5   02BE'CF   9E    00B5   240 30$:    MOVAB   W^DIRPOST,ACB$L_KAST(R5) ; SET SPECIAL KERNEL AST ADDRESS
50     2A A5   02   00   EF    00BB   241 40$:    EXTZV   #IRP$V_BUFIO,#2,IRP$W_STS(R5),R0 ; GET PACKET TYPE
         03 2A A5   09   E0    00C1   242          BBS     #IRP$V_TERMIO,IRP$W_STS(R5),50$ ; BR IF TERMINAL I/O
                50   01   AA    00C6   243          BICW    #1,R0                   ; ELSE TREAT AS NORMAL I/O COMPLETION
                              00C9   244 50$:                                     ; FOR PRIORITY INCREMENT SELECTION
          51   0C A5   D0    00C9   245          MOVL    IRP$L_PID(R5),R1        ; PROCESS IDENTIFICATION
        52   FF2E CF40   9A    00CD   246          MOVZBL  PRITBL[R0],R2           ; SET PRIORITY INCREMENT CLASS
          53   22 A5   9A    00D3   247          MOVZBL  IRP$B_EFN(R5),R3        ; GET EVENT FLAG NUMBER
                FF26'    30    00D7   248          BSBW    SCH$POSTEF              ; AND POST IT
```

```
                        00DA   249 IOPOST_KAST:
0B A5   80 8F   88      00DA   250         BISB    #^X80,ACB$B_RMOD(R5)    ; SET INTERNAL AST FLAG
  00000000'EF   16      00DF   251         JSB     SCH$QAST                ; NOW QUEUE THE KERNEL AST          [04]
         FF25   31      00E5   252         BRW     IOPOST                  ; GET NEXT PACKET TO POST
                        00E8   253         .DSABL  LSB
                        00E8   254
```

```
                        00E8    256          .SBTTL  VIRTUAL I/O COMPLETION
                        00E8    257 ;
                        00E8    258 ; VIRTUAL I/O COMPLETION
                        00E8    259 ;
                        00E8    260 ; CALLING SEQUENCE:
                        00E8    261 ;
                        00E8    262 ;       BRW     VIRTUAL
                        00E8    263 ;
                        00E8    264 ; INPUTS:
                        00E8    265 ;
                        00E8    266 ;       R1 = REQUESTED BYTE COUNT, POSSIBLY DIFFERENT FROM TRANSFERRED
                        00E8    267 ;            BYTE COUNT FOR MAGTAPE
                        00E8    268 ;       R2 = IRP$W_BOFF CONTENTS
                        00E8    269 ;       R3 = SVAPTE OF START OF TRANSFER
                        00E8    270 ;
                        00E8    271 ; OUTPUTS:
                        00E8    272 ;
                        00E8    273 ;       BRANCHES TO UNLOCK, PRESERVING R1,R2,R3
                        00E8    274 ;       OR BRANCHES TO IOPOST
                        00E8    275 ;
                        00E8    276
                        00E8    277          .ENABL  LSB
                        00E8    278
                        00E8    279 VIRTUAL:                                    ; VIRTUAL I/O FUNCTION
          50    36 A5   3C    00E8    280          MOVZWL  IRP$L_IOST1+2(R5),R0    ; GET ACTUAL NUMBER OF BYTES TRANSFERED
          3C A5   50    A0    00EC    281          ADDW    R0,IRP$W_ABCNT(R5)      ; ACCUMULATE TOTAL BYTES TRANSFERED
    50    50    F7 8F   78    00F0    282          ASHL    #-VA$S_BYTE,R0,R0       ; CALCULATE NUMBER OF BLOCKS TRANSFERED
          40 A5   50    C0    00F5    283          ADDL    R0,IRP$L_SEGVBN(R5)     ; CALCULATE DISK ADDRESS OF NEXT SEGMENT
       36 A5   3C A5    B0    00F9    284          MOVW    IRP$W_ABCNT(R5),IRP$L_IOST1+2(R5) ; SET ACCUMULATED BYTES TRANSFERED
             33 34 A5   E9    00FE    285          BLBC    IRP$L_IOST1(R5),20$     ; IF LBC I/O ERROR
          50   1C A5    D0    0102    286          MOVL    IRP$L_UCB(R5),R0        ; GET ADDRESS OF DEVICE UCB
       1F 34 A0   00'    E0    0106    287          BBS     S^#DEV$V_SQD,UCB$L_DEVCHAR(R0),10$ ; IF SET, SEQUENTIAL DEVICE
       3E A5   3C A5    A3    010B    288          SUBW3   IRP$W_ABCNT(R5),IRP$W_OBCNT(R5),- ; CALCULATE BYTES REMAINING
                32 A5           0110    289                  IRP$W_BCNT(R5)     ;
                   16   13      0112    290          BEQL    10$                 ; IF EQL NONE
    51    51    F7 8F   78      0114    291          ASHL    #-VA$S_BYTE,R1,R1       ; CALCULATE NUMBER OF PAGES REQUESTED
                        0119    292 QNXTSEG:
          2C A5   6341    DE    0119    293          MOVAL   (R3)[R1],IRP$L_SVAPTE(R5) ; SET ADDRESS OF NEXT PTE ENTRY
             53   55      D0    011E    294          MOVL    R5,R3                   ; COPY I/O REQUEST PACKET ADDRESS
          55   1C A3    D0    0121    295          MOVL    IRP$L_UCB(R3),R5        ; GET ADDRESS OF DEVICE UCB
                  25    10    0125    296          BSBB    IOC$QNXTSEG             ; QUEUE THE NEXT VIRTUAL SEGMNET
                FEE3    31    0127    297 5$:      BRW     IOPOST
                        012A    298 ;
                        012A    299 ; ALL SEGMENTS OF THIS TRANSFER ARE COMPLETE
                        012A    300 ;
          51    3E A5   3C    012A    301 10$:     MOVZWL  IRP$W_OBCNT(R5),R1      ; GET ORIGINAL BYTE COUNT
          53    44 A5   D0    012E    302          MOVL    IRP$L_DIAGBUF(R5),R3    ; GET ORIGINAL PAGE TABLE ADDRESS
                FF35    31    0132    303          BRW     UNLOCK                  ;
                        0135    304
                        0135    305 ;
                        0135    306 ; I/O OPERATION ENDED WITH AN UNSUCCESSFUL STATUS
                        0135    307 ;
                        0135    308 ;               IF THE DEVICE IS A SEQUENTIAL DEVICE, THEN THE I/O PACKET IS
                        0135    309 ;               MERELY SENT TO THE ACP FOR NOTIFICATION OF THE ERROR.
                        0135    310 ;
                        0135    311 ;               IF THE DEVICE IS A RANDOM DEVICE, THEN THE VIRTUAL BLOCK NUMBER
                        0135    312 ;               STORED IN IRP$L_SEGVBN IS THE BLOCK THAT HAS AN ERROR.
```

```
                                0135   313 ;
                                0135   314
          53    55    D0        0135   315 20$:    MOVL    R5,R3                      ; COPY IRP ADDRESS
             3E A4    B7        0138   316         DECW    PCB$W_DIOCNT(R4)           ; ADJUST DIRECT I/O COUNT
       2A A3    10    AA        013B   317         BICW    #IRP$M_VIRTUAL,IRP$W_STS(R3) ; CLEAR VIRTUAL I/O FLAG
    2C A3    44 A3    D0        013F   318         MOVL    IRP$L_DIAGBUF(R3),IRP$L_SVAPTE(R3) ; RESET PAGE TABLE ADDRESS
       52    3E A3    3C        0144   319         MOVZWL  IRP$W_OBCNT(R3),R2         ; GET ORIGINAL BYTE COUNT
                5B    10        0148   320         BSBB    IOC$QTOACP                 ; QUEUE PACKET TO ACP
                DB    11        014A   321         BRB     5$
                                014C   322
                                014C   323         .DSABL  LSB
```

```
                                014C    325          .SBTTL  QUEUE NEXT SEGMENT
                                014C    326 ;
                                014C    327 ; FUNCTIONAL DESCRIPTION:
                                014C    328 ;
                                014C    329 ;       IOC$QNXTSEG PERFORMS THE FUNCTION OF QUEUEING THE NEXT
                                014C    330 ;       SEGMENT OF A VIRTUAL I/O REQUEST THAT DID NOT MAP TO A
                                014C    331 ;       SINGLE CONTIGUOUS I/O REQUEST.
                                014C    332 ;
                                014C    333 ; CALLING SEQUENCE:
                                014C    334 ;
                                014C    335 ;       BSBW    IOC$QNXTSEG
                                014C    336 ;
                                014C    337 ; INPUTS:
                                014C    338 ;
                                014C    339 ;       R3 = I/O REQUEST PACKET ADDRESS
                                014C    340 ;       R4 = PCB ADDRESS ASSOCIATED WITH THE PID IN THE PACKET
                                014C    341 ;       R5 = UCB ADDRESS OF THE ASSOCIATED DEVICE
                                014C    342 ;
                                014C    343 ; OUTPUTS:
                                014C    344 ;
                                014C    345 ;       R4 NOT PRESERVED
                                014C    346 ;
                                014C    347 IOC$QNXTSEG::
          52    18 A3    D0     014C    348          MOVL    IRP$L_WIND(R3),R2       ; GET ADDRESS OF MAPPING WINDOW
          51    32 A3    3C     0150    349          MOVZWL  IRP$W_BCNT(R3),R1       ; GET SIZE OF NEXT SEGMENT
          50    40 A3    D0     0154    350          MOVL    IRP$L_SEGVBN(R3),R0     ; GET STARTING VIRTUAL BLOCK NUMBER
                                0158    351 ;
                                0158    352 ; ALTERNATE ENTRY TO IOC$QNXTSEG:
                                0158    353 ;
                                0158    354 ;       BSBW    IOC$QNXTSEG1
                                0158    355 ;
                                0158    356 ; ADDITIONAL INPUTS:
                                0158    357 ;
                                0158    358 ;       R0 = VIRTUAL BLOCK NUMBER OF START OF NEXT SEGMENT
                                0158    359 ;       R1 = DESIRED BYTE COUNT OF NEXT SEGMENT
                                0158    360 ;       R2 = WINDOW ADDRESS
                                0158    361 ;
                                0158    362 IOC$QNXTSEG1::
             3E A4    B7     0158    363          DECW    PCB$W_DIOCNT(R4)        ; ADJUST THE DIRECT I/O COUNT
      00000000'EF    16     015B    364          JSB     L^IOC$MAPVBLK           ; Map virtual to logical block
32 A3    32 A3    52    A3     0161    365          SUBW3   R2,IRP$W_BCNT(R3),IRP$W_BCNT(R3) ; CALCULATE SIZE OF NEXT SEGMENT
                   3C    13     0167    366          BEQL    30$                     ; IF EQL TOTAL MAP FAILURE
             50    51    D0     0169    367          MOVL    R1,R0                   ; COPY STARTING LOGICAL BLOCK NUMBER
          52    32 A3    3C     016C    368          MOVZWL  IRP$W_BCNT(R3),R2       ; GET TRANSFER BYTE COUNT
                   52    D7     0170    369          DECL    R2                      ; ROUND DOWN AND...
    52    52    F7 8F    78     0172    370          ASHL    #-VA$S_BYTE,R2,R2       ; SHIFT DOWN FOR BLOCK COUNT - 1
             52    51    C0     0177    371          ADDL    R1,R2                   ; COMPUTE ENDING BLOCK NUMBER
                   13    1F     017A    372          BCS     25$                     ; BRANCH ON OVERFLOW
        0084 C5    52    D1     017C    373          CMPL    R2,UCB$L_MAXBLOCK(R5)   ; AND CHECK AGAINST DEVICE SIZE
                   0C    1E     0181    374          BGEQU   25$                     ; BRANCH IF NOT LEGAL
      00000000'EF    16     0183    375          JSB     L^IOC$CVTLOGPHY         ; Convert logical to physical block
      00000000'EF    17     0189    376          JMP     L^EXE$INSIOQ            ; Insert I/O packet in device queue
                                018F    377                                          ; AND RETURN
                                018F    378 ;
                                018F    379 ; TO HERE IF THE VIRTUAL BLOCKS MAP OFF THE END OF THE VOLUME. COMPLETE THE
                                018F    380 ; I/O WITH AN ERROR. WE QUEUE THE PACKET FOR PROCESSING, RATHER THAN WANDERING
                                018F    381 ; OFF INTO THE COMPLETION CODE BECAUSE THIS IS A GENERALLY CALLABLE ROUTINE.
```

```
                                  018F   ;
        34 A3   0000'8F    3C     018F   25$:    MOVZWL  #SS$_ILLBLKNUM,IRP$L_IOST1(R3) ; SET ILLEGAL BLOCK NUMBER STATUS
                   38 A3   D4     0195           CLRL    IRP$L_IOST2(R3)          ; ZERO 2ND I/O STATUS LONGWORD
     00000000'FF      63   0E     0198           INSQUE  (R3),@IOC$GL_PSBL        ; INSERT AT TAIL OF I/O POST QUEUE
                      03   12     019F           BNEQ    26$                      ; BRANCH IF NOT EMPTY
                                  01A1           SOFTINT #IPL$_IOPOST             ; WAKE UP I/O COMPLETION
                14    04   DA     01A1                   MTPR    #IPL$_IOPOST,S^#PR$_SIRR
                           05     01A4   26$:    RSB
                                  01A5
                                  01A5   30$:
                                  01A5   ;
                                  01A5   ; ALTERNATE ENTRY TO IOC$WAKACP:
                                  01A5   ;
                                  01A5   ;       BSBW    IOC$QTOACP
                                  01A5   ;
                                  01A5   ; INPUTS:
                                  01A5   ;
                                  01A5   ;       R2 = DESIRED BYTE COUNT
                                  01A5   ;       R3 = IRP ADDRESS
                                  01A5   ;       PCB$W_DIOCNT(R4) ALREADY DECREMENTED
                                  01A5   ;
                                  01A5   IOC$QTOACP:
           32 A3    52   B0     01A5           MOVW    R2,IRP$W_BCNT(R3)        ; SET REMAINING BYTES TO TRANSFER
           52    18 A3   D0     01A9           MOVL    IRP$L_WIND(R3),R2        ; GET WINDOW ADDRESS
           0B A2    02   E0     01AD           BBS     #WCB$V_NOTFCP,WCB$B_ACCESS(R2),- ; IF SET THEN
                    20           01B1                   NOTFCPWCB                ; NOT FCP WINDOW
           52    1C A3   D0     01B2           MOVL    IRP$L_UCB(R3),R2         ; GET ADDRESS OF DEVICE UCB
                                  01B6   ;
                                  01B6   ; FUNCTIONAL DESCRIPTION:
                                  01B6   ;
                                  01B6   ;       SUBROUTINE TO QUEUE AN I/O PACKET FOR AN ACP PROCESS AND WAKE
                                  01B6   ;       THE PROCESS IF ITS QUEUE WAS PREVIOUSLY EMPTY.
                                  01B6   ;
                                  01B6   ; CALLING SEQUENCE:
                                  01B6   ;
                                  01B6   ;       BSBW    IOC$WAKACP
                                  01B6   ;
                                  01B6   ; INPUTS:
                                  01B6   ;
                                  01B6   ;       R2 = DEVICE UCB ADDRESS
                                  01B6   ;       R3 = I/O REQUEST PACKET ADDRESS
                                  01B6   ;
                                  01B6   ; OUTPUTS:
                                  01B6   ;
                                  01B6   ;       R4 ALTERED
                                  01B6   ;
                                  01B6   IOC$WAKACP::                             ; QUEUE I/O PACKET AND WAKE ACP PROCESS
                                  01B6           DSBINT  #IPL$_SYNCH              ; SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
                7E    12   DB     01B6                   MFPR    S^#PR$_IPL,-(SP)
                12    07   DA     01B9                   MTPR    #IPL$_SYNCH,S^#PR$_IPL
                                  01BC           BUG_CHECK NONEXSTACP             ; NONEXISTENT ACP PROCESS
             00000000'9F    16     01BC           JSB     @#BUG$CHECK
2C 50 43 41 54 53 58 45 4E 4F 4E 00'  01C2           .ASCIC  "NONEXSTACP,"
                           0B     01C2
                                  01CE   10$:    ENBINT                           ; RESTORE SAVED IPL
                12    8E   DA     01CE                   MTPR    (SP)+,S^#PR$_IPL
                           05     01D1           RSB                              ;
```

```
                                    01D2   432 ;
                                    01D2   433 ; WINDOW IS NOT AN FCP WINDOW, ONLY USED FOR BOOT TIME INTIALIZED WINDOWS
                                    01D2   434 ; FOR CONTIGUOUS FILES.  IT IS NOT POSSIBLE TO NEED TO TURN SUCH A WINDOW.
                                    01D2   435 ;
                                    01D2   436 NOTFCPWCB:
                                    01D2   437      BUG_CHECK NOTFCPWCB,FATAL
                  00000000'9F   16  01D2           JSB     @#BUG$CHECK
46 2C 42 43 57 50 43 46 54 4F 4E 00' 01D8           .ASCIC  "NOTFCPWCB,FATAL"
                        4C 41 54 41  01E4
                                 0F  01D8
```

```
                                01E8   439          .SBTTL  BUFFERED READ COMPLETION AST ROUTINE
                                01E8   440 ;++
                                01E8   441 ; FUNCTIONAL DESCRIPTION:
                                01E8   442 ;
                                01E8   443 ;       BUFPOST PERFORMS ALL NECESSARY COMPLETION OPERATIONS REQUIRED
                                01E8   444 ;       FOR A BUFFERED READ OPERATION IN THE CONTEXT OF THE PROCESS
                                01E8   445 ;       ISSUING THE I/O REQUEST.
                                01E8   446 ;
                                01E8   447 ; CALLING SEQUENCE:
                                01E8   448 ;
                                01E8   449 ;       JSB     BUFPOST
                                01E8   450 ;
                                01E8   451 ; INPUT PARAMETERS:
                                01E8   452 ;
                                01E8   453 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                01E8   454 ;       R5 = IRP/AST CONTROL BLOCK.
                                01E8   455 ;
                                01E8   456 ; IMPLICIT INPUTS:
                                01E8   457 ;
                                01E8   458 ;       SCH$GL_CURPCB - POINTER TO PCB OF CURRENT PROCESS
                                01E8   459 ;--
                                01E8   460
                                01E8   461 BUFPOST:                                     ; BUFFERED READ COMPLETION
                    00E0 8F     BB  01E8   462          PUSHR   #^M<R5,R6,R7>           ; SAVE REGISTERS
                56    2C A5     D0  01EC   463          MOVL    IRP$L_SVAPTE(R5),R6     ; GET ADDRESS OF I/O BUFFER
                57    32 A5     3C  01F0   464          MOVZWL  IRP$W_BCNT(R5),R7       ; GET COUNT OF BYTES OR DESCRIPTORS
          4A 2A A5    03     E1  01F4   465          BBC     #IRP$V_COMPLX,IRP$W_STS(R5),40$ ;IF CLR, NOT COMPLEX BUFFER FORMAT
          4D 2A A5    05     E0  01F9   466          BBS     #IRP$V_CHAINED,IRP$W_STS(R5),50$ ;IF SET ,CHAINED BUFFERS
                56    66     D0  01FE   467          MOVL    (R6),R6                 ; GET ADDRESS OF FIRST BUFFER DESCRIPTOR
                50    02 A6  3C  0201   468 10$:     MOVZWL  2(R6),R0                ; GET COUNT OF BYTES TO TRANSFER
                      32     13  0205   469          BEQL    30$                     ; IF EQL NONE THIS DESCRIPTOR
                51    04 A6  D0  0207   470          MOVL    4(R6),R1                ; GET ADDRESS OF USER BUFFER
                   50    51  C0  020B   471          ADDL    R1,R0                   ; CALCULATE ENDING ADDRESS OF BUFFER
             51    01FF 8F     AA  020E   472          BICW    #VA$M_BYTE,R1           ; TRUNCATE ADDRESS TO PAGE BOUNDARY
                   50    51  C2  0213   473          SUBL    R1,R0                   ; COMPUTE NUMBER OF BYTES TO PROBE
                   54    66  3C  0216   474          MOVZWL  (R6),R4                 ; GET OFFSET TO DATA AREA
             53    FE00 8F     32  0219   475          CVTWL   #-^X200,R3              ; SET ADDITION CONSTANT
                               021E   476 20$:     IFNOWRT R0,(R1),35$,(R6)[R4]    ; CAN BUFFER BE WRITTEN?
          61    50    6644    0D  021E                   PROBEW  (R6)[R4],R0,(R1)
                            1C  13  0223                   BEQL    35$
                   51    53  C2  0225   477          SUBL    R3,R1                   ; UPDATE ADDRESS OF BUFFER
                   50  6043    3E  0228   478          MOVAW   (R0)[R3],R0             ; UPDATE REMAINING LENGTH
                            F0  14  022C   479          BGTR    20$                     ; IF GEQ MORE TO CHECK
04 B6    01 A644     02 A6     28  022E   480          MOVC    2(R6),1(R6)[R4],a4(R6)  ; MOVE DATA TO USER BUFFER
                   55    6E  D0  0236   481          MOVL    (SP),R5                 ; RESTORE ADDRESS OF I/O PACKET
                   56    08  C0  0239   482 30$:     ADDL    #8,R6                   ; ADVANCE TO NEXT BUFFER DESCRIPTOR
                      C2 57     F5  023C   483          SOBGTR  R7,10$                  ; ANY MORE DESCRIPTORS TO PROCESS?
                            72  11  023F   484          BRB     130$                    ;
                            6D  11  0241   485 35$:     BRB     120$                    ; CONTINUE
                         0103   30  0243   486 40$:     BSBW    MOVBUF                  ; MOVE BUFFER TO USER
                   55    6E  D0  0246   487          MOVL    (SP),R5                 ; RETRIEVE ADDRESS OF I/O PACKET
                            68  11  0249   488          BRB     130$                    ;
                   50    57  D0  024B   489 50$:     MOVL    R7,R0                   ; SET LENGTH OF USER BUFFER
                51    04 A6  D0  024E   490          MOVL    4(R6),R1                ; SET ADDRESS OF USER BUFFER
                   50    51  C0  0252   491          ADDL    R1,R0                   ; CALCULATE END OF USER BUFFER
             51    01FF 8F     AA  0255   492          BICW    #VA$M_BYTE,R1           ; TRUNCATE TO PAGE BOUNDARY
                   50    51  C2  025A   493          SUBL    R1,R0                   ; COMPUTE NUMBER OF BYTES TO PROBE
```

```
                52   0B A5   02    00   EF  025D   494          EXTZV   #0,#2,IRP$B_RMOD(R5),R2 ; GET REQUEST ACCESS MODE
                   53   FE00 8F    32   0263   495          CVTWL   #-^X200,R3              ; SET ADDITION CONSTANT
                                    0268   496 60$:     IFNOWRT R0,(R1),90$,R2          ; CAN BUFFER BE WRITTEN?
                   61   50   52    0D   0268                       PROBEW  R2,R0,(R1)
                             2D    13   026C                       BEQL    90$
                        51   53    C2   026E   497          SUBL    R3,R1                  ; UPDATE ADDRESS OF BUFFER
                     50    6043    3E   0271   498          MOVAW   (R0)[R3],R0            ; CALCULATE NEW LENGTH
                             F1    14   0275   499          BGTR    60$                    ; IF GEQ MORE TO PROBE
                   53    04 A6    D0   0277   500          MOVL    4(R6),R3               ; GET STARTING ADDRESS OF USER BUFFER
                   0C A6   57    B1   027B   501 70$:     CMPW    R7,CXB$W_LENGTH(R6)    ; REMAINING LENGTH LARGER THAN DATA AREA?
                             04    1E   027F   502          BGEQU   80$                    ; IF GEQU YES
                   0C A6   57    B0   0281   503          MOVW    R7,CXB$W_LENGTH(R6)    ; TRUNCATE LENGTH OF DATA AREA
                63   96    0C A6    28   0285   504 80$:     MOVC    CXB$W_LENGTH(R6),a(R6)+,(R3) ; MOVE DATA TO USER BUFFER
                      55    6E    D0   028A   505          MOVL    (SP),R5                ; RETRIEVE ADDRESS OF I/O PACKET
                   57    08 A6    A2   028D   506          SUBW    CXB$W_LENGTH-4(R6),R7  ; REDUCE REMAINING BYTES TO TRANSFER
                             0B    13   0291   507          BEQL    100$                   ; IF EQL DONE
                   56    0C A6    D0   0293   508          MOVL    CXB$L_LINK-4(R6),R6    ; GET ADDRESS OF NEXT BUFFER IN CHAIN
                             E2    12   0297   509          BNEQ    70$                    ; IF NEQ MORE TO GO
                             03    11   0299   510          BRB     100$                   ;
                          00DE    30   029B   511 90$:     BSBW    ACCVIO                 ; SET ACCESS VIOLATION
                   50    2C A5    D0   029E   512 100$:    MOVL    IRP$L_SVAPTE(R5),R0    ; GET ADDRESS OF FIRST BUFFER
                   56    10 A0    D0   02A2   513 110$:    MOVL    CXB$L_LINK(R0),R6      ; GET ADDRESS OF NEXT BUFFER
                             0F    13   02A6   514          BEQL    140$                   ; IF EQL NONE
                        FD55'    30   02A8   515          BSBW    EXE$DEANONPAGED        ; DEALLOCATE BUFFER
                      50    56    D0   02AB   516          MOVL    R6,R0                  ; SET ADDRESS OF NEXT BUFFER
                             F2    11   02AE   517          BRB     110$                   ;
                          00C9    30   02B0   518 120$:    BSBW    ACCVIO                 ; SET ACCESS VIOLATION STATUS
                   50    2C A5    D0   02B3   519 130$:    MOVL    IRP$L_SVAPTE(R5),R0    ; GET ADDRESS OF BUFFER TO RELEASE
                        FD46'    30   02B7   520 140$:    BSBW    EXE$DEANONPAGED        ; DEALLOCATE BUFFER
                      00E0 8F    BA   02BA   521          POPR    #^M<R5,R6,R7>          ; RESTORE REGISTERS
```

```
                                02BE   523          .SBTTL  DIRECT I/O COMPLETION AST ROUTINE
                                02BE   524 ;++
                                02BE   525 ; FUNCTIONAL DESCRIPTION:
                                02BE   526 ;
                                02BE   527 ;       DIRPOST PERFORMS ALL GENERAL I/O COMPLETION ACTIVITIES WHICH
                                02BE   528 ;       MUST BE DONE IN THE CONTEXT OF THE PROCESS. THESE INCLUDE
                                02BE   529 ;       I/O STATUS POSTING IF AN IOSB WAS SPECIFIED, CHANNEL CONTROL
                                02BE   530 ;       BLOCK ACTIVITY COUNT DECREMENTING, QUEUEING OF ANY REQUESTED
                                02BE   531 ;       AST OR RELEASE OF THE I/O REQUEST PACKET.
                                02BE   532 ;
                                02BE   533 ; CALLING SEQUENCE:
                                02BE   534 ;
                                02BE   535 ;       JSB     DIRPOST
                                02BE   536 ;
                                02BE   537 ; INPUT PARAMETERS:
                                02BE   538 ;
                                02BE   539 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                02BE   540 ;       R5 = IRP/AST CONTROL BLOCK ADDRESS.
                                02BE   541 ;
                                02BE   542 ; IMPLICIT INPUTS:
                                02BE   543 ;
                                02BE   544 ;       SCH$GL_CURPCB - POINTER TO CURRENT PCB
                                02BE   545 ;--
                                02BE   546
                                02BE   547 DIRPOST:                                     ; DIRECT I/O POSTING AST
1C 2A A5    07    E1            02BE   548          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R5),10$ ; IF CLR, NO DIAGNOSTIC BUFFER
     00E0 8F    BB              02C3   549          PUSHR   #^M<R5,R6,R7>           ; SAVE REGISTERS
   56    44 A5  D0              02C7   550          MOVL    IRP$L_DIAGBUF(R5),R6    ; GET ADDRESS OF DIAGNOSTIC BUFFER
   57    08 A6  3C              02CB   551          MOVZWL  IRP$W_SIZE(R6),R7       ; GET SIZE OF DIAGNOSTIC BUFFER
      57    0C  C2              02CF   552          SUBL    #12,R7                  ; REDUCE BY SIZE OF BUFFER HEADER
            75  10              02D2   553          BSBB    MOVBUF                  ; MOVE DIAGNOSTIC INFORMATION TO USER
     00E0 8F    BA              02D4   554          POPR    #^M<R5,R6,R7>           ; RESTORE REGISTERS
   50    44 A5  D0              02D8   555          MOVL    IRP$L_DIAGBUF(R5),R0    ; RETRIEVE ADDRESS OF DIAGNOSTIC BUFFER
         FD21'  30              02DC   556          BSBW    EXE$DEANONPAGED         ; DEALLOCATE DIAGNOSTIC BUFFER
   50    28 A5  32              02DF   557 10$:     CVTWL   IRP$W_CHAN(R5),R0       ; GET CHANNEL NUMBER (NEGATED)
51 00000000'FF40  9E            02E3   558          MOVAB   @CTL$GL_CCBBASE[R0],R1  ; SET CCB BASE ADDRESS
      0A A1  B7                 02EB   559          DECW    CCB$W_IOC(R1)           ; DECREMENT I/O COUNT FOR CHANNEL
         12  12                 02EE   560          BNEQ    30$                     ; NOT IDLE YET
   53    0C A1  D0              02F0   561          MOVL    CCB$L_DIRP(R1),R3       ; GET ADDRESS OF DEACCESS PACKET
         0C  13                 02F4   562          BEQL    30$                     ; IF EQL NONE
      0C A1  D4                 02F6   563          CLRL    CCB$L_DIRP(R1)          ; CLEAR ADDRESS OF DEACCESS PACKET
      0A A1  B6                 02F9   564          INCW    CCB$W_IOC(R1)           ; ACCOUNT FOR DEACCESS
   52    61  D0                 02FC   565          MOVL    CCB$L_UCB(R1),R2        ; GET ASSIGNED DEVICE UCB ADDRESS
         FEB4  30               02FF   566          BSBW    IOC$WAKACP              ; QUEUE I/O PACKET AND WAKE ACP
                                0302   567                                          ; R4 ALTERERED
                                0302   568 30$:
                                0302   569 ;
                                0302   570 ; R4 DOES NOT NECESSARILY HAVE CURRENT PCB ADDRESS IN IT AT THIS POINT
                                0302   571 ;
                                0302   572 IOC$DIRPOST1::
   50    24 A5  D0              0302   573          MOVL    IRP$L_IOSB(R5),R0       ; GET IOSB ADDRESS
         10  13                 0306   574          BEQL    35$                     ; IF EQL NONE SPECIFIED
51 0B A5  02  00  EF            0308   575          EXTZV   #0,#2,IRP$B_RMOD(R5),R1 ; GET REQUEST ACCESS MODE
                                030E   576          IFNOWRT #8,(R0),35$,R1          ; CAN I/O STATUS BE WRITTEN?
   60    08  51  0D             030E                PROBEW  R1,#8,(R0)
         04  13                 0312                BEQL    35$
   60    34 A5  7D              0314   577          MOVQ    IRP$L_IOST1(R5),(R0)    ; MOVE STATUS INTO IOSB
```

```
                13 2A A5    0B   E0  0318    578 35$:    BBS     #IRP$V_EXTEND,IRP$W_STS(R5),50$ ; BRANCH TO DEALLOCATE IRPE'S
                08 0B A5    06   E1  031D    579 37$:    BBC     #ACB$V_QUOTA,IRP$B_RMOD(R5),40$ ; IF CLR, NO AST SPECIFIED
                            52   D4  0322    580         CLRL    R2                              ; SET NULL PRIORITY INCREMENT
                  00000000'EF    17  0324    581         JMP     SCH$QAST                        ; QUEUE AST FOR REQUESTOR
                      50    55   D0  032A    582 40$:    MOVL    R5,R0                           ; SETUP ADDRESS FOR DEALLOCATE
                         FCD0'   31  032D    583         BRW     EXE$DEANONPAGED                 ; AND RELEASE I/O PACKET
                                     0330    584
                                     0330    585
                                     0330    586         ;
                                     0330    587         ; DEALLOCATE IRPE'S
                                     0330    588         ;
                                     0330    589
                   50    4C A5    D0  0330    590 50$:    MOVL    IRP$L_EXTEND(R5),R0             ; GET ADDRESS OF FIRST IRPE
                                     0334    591
                            54    D4  0334    592 60$:    CLRL    R4                              ; WILL HOLD ADDRESS OF NEXT IRPE
                04 2A A0    0B   E1  0336    593         BBC     #IRPE$V_EXTEND,IRPE$W_STS(R0),70$ ; BR. IF NO MORE IRPE'S
                   54    4C A0   D0  033B    594         MOVL    IRPE$L_EXTEND(R0),R4            ; SAVE ADDRESS OF NEXT IRPE
                         FCBE'   30  033F    595 70$:    BSBW    EXE$DEANONPAGED                 ; DEALLOCATE IRPE POINTED TO BY R0
                      50    54   D0  0342    596         MOVL    R4,R0                           ; PUT ADDRESS OF NEXT IRPE IN R0
                            ED   12  0345    597         BNEQ    60$                             ; BR. IF THERE IS ANOTHER IRPE
                            D4   11  0347    598         BRB     37$                             ; DONE DEALLOCATING IRPE'S
```

```
                                0349          .SBTTL  MOVE DATA TO USER BUFFER
                                0349  ;
                                0349  ; SUBROUTINE TO MOVE DATA FROM A SIMPLE BUFFERED I/O BUFFER TO A USER BUFFER
                                0349  ;
                                0349
                                0349  MOVBUF:                                          ; MOVE BUFFER
              50    57    D0    0349          MOVL    R7,R0                            ; SET LENGTH OF USER BUFFER
                    2D    13    034C          BEQL    15$                              ; BR IF NULL STRING
           51    04 A6    D0    034E          MOVL    4(R6),R1                         ; GET ADDRESS OF USER BUFFER
              50    51    C0    0352          ADDL    R1,R0                            ; CALCULATE ENDING BUFFER ADDRESS
        51    01FF 8F     AA    0355          BICW    #VA$M_BYTE,R1                    ; TRUNCATE ADDRESS TO START OF PAGE
              50    51    C2    035A          SUBL    R1,R0                            ; CALCULATE LENGTH OF BUFFER TO PROBE
  52    0B A5   02    00    EF    035D          EXTZV   #0,#2,IRP$B_RMOD(R5),R2          ; GET REQUEST ACCESS MODE
        53    FE00 8F     32    0363          CVTWL   #-^X200,R3                       ; SET ADDITION CONSTANT
                                0368  10$:    IFNOWRT R0,(R1),ACCVIO,R2                ; CAN BUFFER BE WRITTEN?
        61    50    52    0D    0368                  PROBEW  R2,R0,(R1)
                    0E    13    036C                  BEQL    ACCVIO
              51    53    C2    036E          SUBL    R3,R1                            ; UPDATE BUFFER ADDRESS
           50    6043     3E    0371          MOVAW   (R0)[R3],R0                      ; UPDATE REMAINING LENGTH OF BUFFER
                    F1    14    0375          BGTR    10$                              ; IF GEQ MORE TO CHECK
        96    96    57    28    0377          MOVC    R7,a(R6)+,a(R6)+                 ; MOVE DATA TO USER BUFFER
                          05    037B  15$:    RSB                                      ;
           34 A5    00'   B0    037C  ACCVIO: MOVW    S^#SS$_ACCVIO,IRP$L_IOST1(R5) ; SET FINAL TRANSFER STATUS
                          05    0380          RSB                                      ;
```

```
                                0381   623          .SBTTL  UNLOCK AREAS IN IRPE'S
                                0381   624 ;**
                                0381   625 ; FUNCTIONAL DESCRIPTION:
                                0381   626 ;
                                0381   627 ;       THIS ROUTINE UNLOCKS THE AREAS DESCRIBED BY FIELDS IN THE IRPE'S.  EACH
                                0381   628 ;       IRPE HAS SPACE TO HOLD TWO AREA DESCRIPTIONS.
                                0381   629 ;
                                0381   630 ; CALLING SEQUENCE:
                                0381   631 ;
                                0381   632 ;       BSBW    UNLOCK_MORE
                                0381   633 ;
                                0381   634 ; INPUT PARAMETERS:
                                0381   635 ;
                                0381   636 ;       R5 = I/O REQUEST PACKET ADDRESS
                                0381   637 ;
                                0381   638 ; SIDE EFFECTS:
                                0381   639 ;
                                0381   640 ;       R0 - R3 ARE NOT PRESERVED
                                0381   641 ;--
                                0381   642
                                0381   643          ASSUME  IRP$L_EXTEND EQ IRPE$L_EXTEND
                                0381   644
                                0381   645 UNLOCK_MORE:
                          55    DD    0381   646          PUSHL   R5                      ; SAVE IRP ADDRESS
                                0383   647
                                0383   648 10$:    ; UNLOCK AREAS SPECIFIED IN NEXT IRPE
                                0383   649
                    55    4C A5    D0  0383   650          MOVL    IRPE$L_EXTEND(R5),R5    ; GET ADDRESS OF NEXT IRPE
                    53    2C A5    D0  0387   651          MOVL    IRPE$L_SVAPTE1(R5),R3   ; GET SVAPTE OF FIRST AREA
                             0A    13  038B   652          BEQL    20$                     ; BR. IF NOTHING TO UNLOCK
                    52    30 A5    3C  038D   653          MOVZWL  IRPE$W_BOFF1(R5),R2     ; GET BYTE OFFSET IN PAGE
                    51    34 A5    D0  0391   654          MOVL    IRPE$L_BCNT1(R5),R1     ; GET SIZE OF AREA
                             19    10  0395   655          BSBB    UNLK                    ; UNLOCK FIRST AREA
                                0397   656
                    53    38 A5    D0  0397   657 20$:     MOVL    IRPE$L_SVAPTE2(R5),R3   ; GET SVAPTE OF SECOND AREA
                             0A    13  039B   658          BEQL    30$                     ; BR. IF NOTHING TO UNLOCK
                    52    3C A5    3C  039D   659          MOVZWL  IRPE$W_BOFF2(R5),R2     ; GET BYTE OFFSET IN PAGE
                    51    40 A5    D0  03A1   660          MOVL    IRPE$L_BCNT2(R5),R1     ; GET SIZE OF AREA
                             09    10  03A5   661          BSBB    UNLK                    ; UNLOCK SECOND AREA
                                03A7   662
              D7 2A A5       0B    E0  03A7   663 30$:     BBS     #IRPE$V_EXTEND,IRPE$W_STS(R5),10$ ; BR. IF THERE'S ANOTHER IRPE
                          55 8ED0    03AC   664          POPL    R5                      ; RESTORE R5
                                05    03AF   665          RSB
                                03B0   666
                                03B0   667          ;
                                03B0   668          ; LOCAL SUBROUTINE TO UNLOCK PAGES
                                03B0   669          ;
                                03B0   670          ;       R1 = BYTE COUNT (OR SIZE OF AREA)
                                03B0   671          ;       R2 = BYTE OFFSET IN PAGE
                                03B0   672          ;       R3 = SVAPTE OF START OF AREA
                                03B0   673          ;
                                03B0   674
           51    01FF C142    9E  03B0   675 UNLK:    MOVAB   511(R1)[R2],R1          ; COMBINE OFFSET AND SIZE AND ROUND
        51    51    F7 8F    78  03B6   676          ASHL    #-VA$S_BYTE,R1,R1       ; CONVERT TO NUMBER OF PAGES TO UNLOCK
            00000000'EF    16  03BB   677          JSB     L^MMG$UNLOCK            ; Unlock pages
                             05  03C1   678          RSB
                                03C2   679
```

```
                    03C2   680
                    03C2   681          .END
```

```
$ER                             = 00000002     D
$MODULE                           00000000 R   D   02
AACPERR                           00000007 R   D   02
ACB$B_RMOD                        0000000B     D
ACB$B_TYPE                        0000000A     D
ACB$C_LENGTH                      00000018     D
ACB$K_LENGTH                      00000018     D
ACB$L_AST                         00000010     D
ACB$L_ASTPRM                      00000014     D
ACB$L_ASTQBL                      00000004     D
ACB$L_ASTQFL                      00000000     D
ACB$L_KAST                        00000018     D
ACB$L_PID                         0000000C     D
ACB$V_QUOTA                     = 00000006     D
ACB$W_SIZE                        00000008     D
ACCVIO                            0000037C R   D   03
BUFIO                             00000096 R   D   03
BUFPOST                           000001E8 R   D   03
BUG$CHECK                         ********   X     03
CCB$B_AMOD                        00000009     D
CCB$B_STS                         00000008     D
CCB$C_LENGTH                      00000010     D
CCB$K_LENGTH                      00000010     D
CCB$L_DIRP                        0000000C     D
CCB$L_UCB                         00000000     D
CCB$L_WIND                        00000004     D
CCB$W_IOC                         0000000A     D
CTL$GL_CCBBASE                    ********   X     03
CXB$L_LINK                      = 00000010     D
CXB$W_LENGTH                    = 0000000C     D
DEV$V_SQD                         ********   X     03
DIRIO                             0000004E R   D   03
DIRPOST                           000002BE R   D   03
DS_ERRSUP                         ********   X     03
EXE$DEANONPAGED                   ********   X     03
EXE$INSIOQ                        ********   X     03
IOC$CVTLOGPHY                     ********   X     03
IOC$DIRPOST1                      00000302 RG  D   03
IOC$GL_PSBL                       ********   X     03
IOC$GL_PSFL                       ********   X     03
IOC$IOPOST                        00000004 RG  D   03
IOC$MAPVBLK                       ********   X     03
IOC$QNXTSEG                       0000014C RG  D   03
IOC$QNXTSEG1                      00000158 RG  D   03
IOC$QTOACP                        000001A5 R   D   03
IOC$WAKACP                        000001B6 RG  D   03
IOPOST                            0000000D R   D   03
IOPOST_KAST                       000000DA R   D   03
IPL$_IOPOST                     = 00000004     D
IPL$_SYNCH                      = 00000007     D
IRP$B_CARCON                      00000038     D
IRP$B_EFN                         00000022     D
IRP$B_PRI                         00000023     D
IRP$B_RMOD                        0000000B     D
IRP$B_TYPE                        0000000A     D
IRP$C_LENGTH                      0000005C     D
IRP$K_LENGTH                      0000005C     D
IRP$L_ARB                                      00000050     D
IRP$L_AST                                      00000010     D
IRP$L_ASTPRM                                   00000014     D
IRP$L_DIAGBUF                                  00000044     D
IRP$L_EXTEND                                   0000004C     D
IRP$L_IOQBL                                    00000004     D
IRP$L_IOQFL                                    00000000     D
IRP$L_IOSB                                     00000024     D
IRP$L_IOST1                                    00000034     D
IRP$L_IOST2                                    00000038     D
IRP$L_MEDIA                                    00000034     D
IRP$L_PID                                      0000000C     D
IRP$L_SEGVBN                                   00000040     D
IRP$L_SEQNUM                                   00000048     D
IRP$L_SVAPTE                                   0000002C     D
IRP$L_TT_TERM                                  00000038     D
IRP$L_UCB                                      0000001C     D
IRP$L_WIND                                     00000018     D
IRP$M_PAGIO                                  = 00000004     D
IRP$M_SWAPIO                                 = 00000040     D
IRP$M_VIRTUAL                                = 00000010     D
IRP$Q_NT_PRVMSK                                00000038     D
IRP$V_BUFIO                                  = 00000000     D
IRP$V_CHAINED                                = 00000005     D
IRP$V_COMPLX                                 = 00000003     D
IRP$V_DIAGBUF                                = 00000007     D
IRP$V_EXTEND                                 = 0000000B     D
IRP$V_FILACP                                 = 0000000C     D
IRP$V_FUNC                                   = 00000001     D
IRP$V_PAGIO                                  = 00000002     D
IRP$V_SWAPIO                                 = 00000006     D
IRP$V_TERMIO                                 = 00000009     D
IRP$V_VIRTUAL                                = 00000004     D
IRP$W_ABCNT                                    0000003C     D
IRP$W_BCNT                                     00000032     D
IRP$W_BOFF                                     00000030     D
IRP$W_CHAN                                     00000028     D
IRP$W_FUNC                                     00000020     D
IRP$W_OBCNT                                    0000003E     D
IRP$W_SIZE                                     00000008     D
IRP$W_STS                                      0000002A     D
IRP$W_TT_PRMPT                                 0000003C     D
IRPE$B_TYPE                                    0000000A     D
IRPE$C_LENGTH                                  00000050     D
IRPE$K_LENGTH                                  00000050     D
IRPE$L_BCNT1                                   00000034     D
IRPE$L_BCNT2                                   00000040     D
IRPE$L_EXTEND                                  0000004C     D
IRPE$L_SVAPTE1                                 0000002C     D
IRPE$L_SVAPTE2                                 00000038     D
IRPE$V_EXTEND                                = 0000000B     D
IRPE$W_BOFF1                                   00000030     D
IRPE$W_BOFF2                                   0000003C     D
IRPE$W_SIZE                                    00000008     D
IRPE$W_STS                                     0000002A     D
MMG$UNLOCK                                     ********   X     03
MOVBUF                                         00000349 R   D   03
```

NOTACP		000000A1	R	D	03	SCH$POSTEF		********	X	03	
NOTFCPWCB		000001D2	R	D	03	SCH$QAST		********	X	03	
PAGIO_OR_SWAPIO		00000085	R	D	03	SIZ...	=	00000001	D		
PCB$B_ASTACT		0000000C		D		SS$_ACCVIO		********	X	03	
PCB$B_ASTEN		0000000D		D		SS$_ILLBLKNUM		********	X	03	
PCB$B_PRI		0000000B		D		UCB$B_AMOD		00000053	D		
PCB$B_PRIB		0000002F		D		UCB$B_CEX		00000077	D		
PCB$B_TYPE		0000000A		D		UCB$B_CM1		0000004A	D		
PCB$B_WEFC		0000002E		D		UCB$B_CM2		0000004B	D		
PCB$C_LENGTH		0000008C		D		UCB$B_DEVCLASS		00000038	D		
PCB$K_LENGTH		0000008C		D		UCB$B_DEVTYPE		00000039	D		
PCB$L_ARB		00000084		D		UCB$B_DIPL		00000052	D		
PCB$L_ASTQBL		00000014		D		UCB$B_DX_SCTCNT		000000A6	D		
PCB$L_ASTQFL		00000010		D		UCB$B_ERTCNT		00000070	D		
PCB$L_EFC2P		00000058		D		UCB$B_ERTMAX		00000071	D		
PCB$L_EFC3P		0000005C		D		UCB$B_FEX		00000076	D		
PCB$L_EFCS		00000050		D		UCB$B_FIPL		0000000B	D		
PCB$L_EFCU		00000054		D		UCB$B_LOCSRV		0000003C	D		
PCB$L_EFWM		0000004C		D		UCB$B_OFFNDX		00000094	D		
PCB$L_JIB		00000078		D		UCB$B_OFFRTC		00000095	D		
PCB$L_OWNER		0000001C		D		UCB$B_REMSRV		0000003D	D		
PCB$L_PHD		00000064		D		UCB$B_SECTORS		0000003C	D		
PCB$L_PHYPCB		00000018		D		UCB$B_SLAVE		00000074	D		
PCB$L_PID		00000060		D		UCB$B_SPR		00000075	D		
PCB$L_PQB		0000004C		D		UCB$B_STATE		00000052	D		
PCB$L_SQBL		00000004		D		UCB$B_TRACKS		0000003D	D		
PCB$L_SQFL		00000000		D		UCB$B_TT_CRFILL		0000009D	D		
PCB$L_STS		00000024		D		UCB$B_TT_DECRF		000000A1	D		
PCB$L_UIC		00000088		D		UCB$B_TT_DELFF		000000A2	D		
PCB$L_WSSWP		00000020		D		UCB$B_TT_DESPEE		000000A0	D		
PCB$L_WTIME		00000028		D		UCB$B_TT_DETYPE		000000A4	D		
PCB$Q_PRIV		0000007C		D		UCB$B_TT_LFFILL		0000009E	D		
PCB$T_LNAME		00000068		D		UCB$B_TT_SPEED		0000009C	D		
PCB$T_TERMINAL		00000044		D		UCB$B_TYPE		0000000A	D		
PCB$W_APTCNT		00000030		D		UCB$B_VERTSZ		0000003F	D		
PCB$W_ASTCNT		00000038		D		UCB$C_LENGTH		00000074	D		
PCB$W_BIOCNT		0000003A		D		UCB$C_MB_LENGTH		00000090	D		
PCB$W_BIOLM		0000003C		D		UCB$C_TT_LENGTH		000000BC	D		
PCB$W_DIOCNT		0000003E		D		UCB$K_LENGTH		00000074	D		
PCB$W_DIOLM		00000040		D		UCB$K_MB_LENGTH		00000090	D		
PCB$W_GPGCNT		00000034		D		UCB$K_TT_LENGTH		000000BC	D		
PCB$W_GRP		0000008A		D		UCB$L_AMB		00000054	D		
PCB$W_MEM		00000088		D		UCB$L_ASTQBL		00000010	D		
PCB$W_MTXCNT		0000000E		D		UCB$L_ASTQFL		0000000C	D		
PCB$W_PPGCNT		00000036		D		UCB$L_CPID		0000005C	D		
PCB$W_PRCCNT		00000042		D		UCB$L_CRB		00000020	D		
PCB$W_SIZE		00000008		D		UCB$L_DDB		00000024	D		
PCB$W_STATE		0000002C		D		UCB$L_DEVCHAR		00000034	D		
PCB$W_TMBU		00000032		D		UCB$L_DEVDEPEND		0000003C	D		
PR$_IPL	=	00000012		D		UCB$L_DPC		00000080	D		
PR$_SIRR	=	00000014		D		UCB$L_DUETIM		0000005C	D		
PRI$_IOCOM	=	00000001		D		UCB$L_DX_BFPNT		0000009C	D		
PRI$_TICOM	=	00000004		D		UCB$L_DX_BUF		00000098	D		
PRI$_TOCOM	=	00000003		D		UCB$L_DX_RXDB		000000A0	D		
PRITBL		00000000	R	D	03	UCB$L_EMB		00000078	D		
QNXTSEG		00000119	R	D	03	UCB$L_FIRST		00000014	D		
SCH$GL_PCBVEC		********		X	03	UCB$L_FPC		0000000C	D		

UCB$L_FQBL	00000004	D	UCB$W_SRCADDR	0000001A	D					
UCB$L_FQFL	00000000	D	UCB$W_STS	00000058	D					
UCB$L_FR3	00000010	D	UCB$W_TT_DESIZE	000000A5	D					
UCB$L_FR4	00000014	D	UCB$W_UNIT	00000048	D					
UCB$L_IOQBL	00000044	D	UCB$W_VPROT	0000001A	D					
UCB$L_IOQFL	00000040	D	UNLK	000003B0	R	D	03			
UCB$L_IRP	0000004C	D	UNLOCK	0000006A	R	D	03			
UCB$L_LINK	0000002C	D	UNLOCK_MORE	00000381	R	D	03			
UCB$L_LOGADR	00000064	D	VA$M_BYTE	= 000001FF		D				
UCB$L_MAXBLOCK	00000084	D	VA$S_BYTE	= 00000009		D				
UCB$L_MB_MBX	0000007C	D	VCB$B_BLOCKFACT	00000052		D				
UCB$L_MB_PORT	0000008C	D	VCB$B_CUR_RVN	0000002F		D				
UCB$L_MB_RAST	00000078	D	VCB$B_EOFDELTA	0000004E		D				
UCB$L_MB_SHB	00000080	D	VCB$B_IBMAPSIZE	00000038		D				
UCB$L_MB_WAST	00000074	D	VCB$B_IBMAPVBN	0000003A		D				
UCB$L_MB_WIOQBL	00000088	D	VCB$B_LRU_LIM	00000049		D				
UCB$L_MB_WIOQFL	00000084	D	VCB$B_QNAMECNT	0000000B		D				
UCB$L_MEDIA	0000008C	D	VCB$B_RESFILES	0000004F		D				
UCB$L_NT_DATSSB	00000074	D	VCB$B_SBMAPSIZE	00000039		D				
UCB$L_NT_INTSSB	00000078	D	VCB$B_SBMAPVBN	0000003B		D				
UCB$L_OPCNT	00000060	D	VCB$B_STATUS	0000000B		D				
UCB$L_OWNUIC	0000001C	D	VCB$B_STATUS2	00000053		D				
UCB$L_PID	00000028	D	VCB$B_TM	0000002E		D				
UCB$L_RQBL	00000004	D	VCB$B_TYPE	0000000A		D				
UCB$L_RQFL	00000000	D	VCB$B_WINDOW	00000048		D				
UCB$L_SVAPTE	00000068	D	VCB$C_COMLEN	00000024		D				
UCB$L_SVPN	00000064	D	VCB$C_LENGTH	00000064		D				
UCB$L_TT_DECHAR	000000A8	D	VCB$C_MRKLEN	0000000B		D				
UCB$L_TT_RDUE	0000008C	D	VCB$K_COMLEN	00000024		D				
UCB$L_TT_RTIMOU	000000B8	D	VCB$K_LENGTH	00000064		D				
UCB$L_VCB	00000030	D	VCB$K_MRKLEN	0000000B		D				
UCB$T_PARTNER	0000000C	D	VCB$L_AQB	00000010		D				
UCB$W_BCNT	0000006E	D	VCB$L_BLOCKBL	00000004		D				
UCB$W_BCR	00000096	D	VCB$L_BLOCKFL	00000000		D				
UCB$W_BOFF	0000006C	D	VCB$L_CACHE	00000058		D				
UCB$W_BUFQUO	00000018	D	VCB$L_CUR_FID	00000024		D				
UCB$W_BYTESTOGO	0000003E	D	VCB$L_FCBBL	00000004		D				
UCB$W_CHARGE	0000004A	D	VCB$L_FCBFL	00000000		D				
UCB$W_CYLINDERS	0000003E	D	VCB$L_FREE	00000040		D				
UCB$W_DA	0000008C	D	VCB$L_HOME2LBN	00000028		D				
UCB$W_DC	0000008E	D	VCB$L_HOMELBN	00000024		D				
UCB$W_DEVBUFSIZ	0000003A	D	VCB$L_IBMAPLBN	00000030		D				
UCB$W_DEVSTS	0000005A	D	VCB$L_IXHDR2LBN	0000002C		D				
UCB$W_DIRSEQ	00000088	D	VCB$L_MAXFILES	00000044		D				
UCB$W_DSTADDR	00000018	D	VCB$L_MVL	00000034		D				
UCB$W_DX_BCR	000000A4	D	VCB$L_QUOCACHE	0000005C		D				
UCB$W_EC1	00000090	D	VCB$L_QUOTAFCB	00000054		D				
UCB$W_EC2	00000092	D	VCB$L_RVT	00000020		D				
UCB$W_ERRCNT	00000072	D	VCB$L_SBMAPLBN	00000034		D				
UCB$W_FUNC	0000007E	D	VCB$L_START_FID	00000028		D				
UCB$W_MB_SEED	00000000	D	VCB$L_ST_RECORD	00000030		D				
UCB$W_MSGCNT	00000016	D	VCB$L_USRLBLAST	00000044		D				
UCB$W_MSGMAX	00000014	D	VCB$L_VPBL	00000040		D				
UCB$W_NT_CHAN	0000007C	D	VCB$L_VPFL	0000003C		D				
UCB$W_OFFSET	0000008A	D	VCB$L_WCB	00000038		D				
UCB$W_REFC	00000050	D	VCB$T_QNAME	0000000C		D				
UCB$W_SIZE	00000008	D	VCB$T_VOLNAME	00000014		D				

```
VCB$W_CLUSTER                   0000003C    D
VCB$W_CUR_NUM                   00000024    D
VCB$W_CUR_SEQ                   00000026    D
VCB$W_EXTEND                    0000003E    D
VCB$W_FILEPROT                  0000004A    D
VCB$W_MCOUNT                    0000004C    D
VCB$W_MODE                      0000002C    D
VCB$W_PENDERR                   00000062    D
VCB$W_QUOSIZE                   00000060    D
VCB$W_RECORDSZ                  00000050    D
VCB$W_RVN                       0000000E    D
VCB$W_SIZE                      00000008    D
VCB$W_START_NUM                 00000028    D
VCB$W_START_SEQ                 0000002A    D
VCB$W_TRANS                     0000000C    D
VIRTUAL                         000000E8 R  D  03
WCB$B_ACCESS                    0000000B    D
WCB$B_TYPE                      0000000A    D
WCB$C_LENGTH                    00000024    D
WCB$C_MAP                       00000024    D
WCB$K_LENGTH                    00000024    D
WCB$K_MAP                       00000024    D
WCB$L_FCB                       00000018    D
WCB$L_LBN                       00000002    D
WCB$L_ORGUCB                    00000010    D
WCB$L_P1_LBN                    00000026    D
WCB$L_P2_LBN                    0000002C    D
WCB$L_PID                       0000000C    D
WCB$L_PREVLBN                   FFFFFFFC    D
WCB$L_RVT                       0000001C    D
WCB$L_STVBN                     00000020    D
WCB$L_WLBL                      00000004    D
WCB$L_WLFL                      00000000    D
WCB$V_NOTFCP                  = 00000002    D
WCB$W_ACON                      00000014    D
WCB$W_COUNT                     00000000    D
WCB$W_NMAP                      00000016    D
WCB$W_P1_COUNT                  00000024    D
WCB$W_P2_COUNT                  0000002A    D
WCB$W_PREVCOUNT                 FFFFFFFA    D
WCB$W_REFCNT                    0000000E    D
WCB$W_SIZE                      00000008    D
WQH$C_LENGTH                    0000000C    D
WQH$K_LENGTH                    0000000C    D
WQH$L_WQBL                      00000004    D
WQH$L_WQFL                      00000000    D
WQH$W_WQCNT                     00000008    D
WQH$W_WQSTATE                   0000000A    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000011 (17.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	000003C2 (962.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000002	226 (3)	#-226 (3)
$MODULE	00000000-R	130 (2)	226 (3)
AACPERR	00000007-R	131 (2)	
ACB$B_RMOD	0000000B		#-250 (3)
ACB$L_KAST	00000018		#-237 (3) #-240 (3)
ACB$V_QUOTA	=00000006		#-579 (7)
ACCVIO	0000037C-R	620 (8)	#-511 (6) #-518 (6) #-614 (8)
BUFIO	00000096-R	231 (3)	#-191 (3)
BUFPOST	000001E8-R	461 (6)	237 (3)
BUG$CHECK	00000000-XR		429 (5) 437 (5)
CCB$L_DIRP	0000000C		#-561 (7) #-563 (7)
CCB$L_UCB	00000000		#-565 (7)
CCB$W_IOC	0000000A		#-559 (7) #-564 (7)
CTL$GL_CCBBASE	00000000-XR		558 (7)
CXB$L_LINK	=00000010		#-508 (6) #-513 (6)
CXB$W_LENGTH	=0000000C		#-501 (6) #-503 (6) #-504 (6) #-506 (6)
DEV$V_SQD	00000000-XR		#-287 (4)
DIRIO	0000004E-R	204 (3)	
DIRPOST	000002BE-R	547 (7)	240 (3)
DS_ERRSUP	00000000-XR		226 (3)
EXE$DEANONPAGED	00000000-XR		#-239 (3) #-515 (6) #-520 (6) #-556 (7)
			#-583 (7) #-595 (7)
EXE$INSIOQ	00000000-XR		376 (5)
IOC$CVTLOGPHY	00000000-XR		375 (5)
IOC$DIRPOST1	00000302-R	572 (7)	
IOC$GL_PSBL	00000000-XR		385 (5)
IOC$GL_PSFL	00000000-XR		173 (3)
IOC$IOPOST	00000004-R	169 (3)	
IOC$MAPVBLK	00000000-XR		364 (5)
IOC$QNXTSEG	0000014C-R	347 (5)	#-296 (4)
IOC$QNXTSEG1	00000158-R	362 (5)	
IOC$QTOACP	000001A5-R	402 (5)	#-320 (4)
IOC$WAKACP	000001B6-R	427 (5)	#-566 (7)
IOPOST	0000000D-R	173 (3)	#-183 (3) #-252 (3) #-297 (4)
IOPOST_KAST	000000DA-R	249 (3)	
IPL$_IOPOST	=00000004		#-387 (5)
IPL$_SYNCH	=00000007		#-428 (5)
IRP$B_EFN	00000022		#-247 (3)
IRP$B_RMOD	0000000B		494 (6) 575 (7) 579 (7) 612 (8)
IRP$L_DIAGBUF	00000044		#-302 (4) #-318 (4) #-550 (7) #-555 (7)
IRP$L_EXTEND	0000004C		#-590 (7) 643 (9)
IRP$L_IOSB	00000024		#-573 (7)
IRP$L_IOST1	00000034		#-209 (3) #-210 (3) #-280 (4) #-284 (4)
			#-285 (4) #-383 (5) #-577 (7) #-620 (8)
IRP$L_IOST2	00000038		#-384 (5)
IRP$L_PID	000000C		#-185 (3) #-245 (3)
IRP$L_SEGVBN	00000040		#-283 (4) #-350 (5)
IRP$L_SVAPTE	0000002C		#-193 (3) #-235 (3) #-293 (4) #-318 (4)
			#-463 (6) #-512 (6) #-519 (6)
IRP$L_UCB	0000001C		#-286 (4) #-295 (4) #-407 (5)

```
IRP$L_WIND                  00000018                  #-348    (5)    #-404    (5)
IRP$M_PAGIO                =00000004                  #-197    (3)
IRP$M_SWAPIO               =00000040                  #-197    (3)
IRP$M_VIRTUAL              =00000010                  #-317    (4)
IRP$V_BUFIO                =00000000                  #-190    (3)    #-241    (3)
IRP$V_CHAINED              =00000005                  #-466    (6)
IRP$V_COMPLX               =00000003                  #-465    (6)
IRP$V_DIAGBUF              =00000007                  #-548    (7)
IRP$V_EXTEND               =0000000B                  #-216    (3)    #-578    (7)
IRP$V_FILACP               =0000000C                  #-232    (3)
IRP$V_FUNC                 =00000001                  #-238    (3)
IRP$V_PAGIO                =00000002                    195    (3)
IRP$V_SWAPIO               =00000006                    196    (3)
IRP$V_TERMIO               =00000009                  #-242    (3)
IRP$V_VIRTUAL              =00000004                  #-208    (3)
IRP$W_ABCNT                 0000003C                  #-281    (4)    #-284    (4)    #-288    (4)
IRP$W_BCNT                  00000032                  #-206    (3)    #-289    (4)    #-349    (5)    #-365    (5)
                                                      #-368    (5)    #-403    (5)    #-464    (6)
IRP$W_BOFF                  00000030                  #-207    (3)
IRP$W_CHAN                  00000028                  #-557    (7)
IRP$W_OBCNT                 0000003E                  #-210    (3)    #-288    (4)    #-301    (4)    #-319    (4)
IRP$W_SIZE                  00000008                  #-551    (7)
IRP$W_STS                   0000002A                    190    (3)    #-197    (3)      208    (3)      216    (3)
                                                        232    (3)      238    (3)      241    (3)      242    (3)
                                                      #-317    (4)      465    (6)      466    (6)      548    (7)
                                                        578    (7)
IRPE$L_BCNT1                00000034                  #-654    (9)
IRPE$L_BCNT2                00000040                  #-660    (9)
IRPE$L_EXTEND               0000004C                  #-594    (7)      643    (9)    #-650    (9)
IRPE$L_SVAPTE1              0000002C                  #-651    (9)
IRPE$L_SVAPTE2              00000038                  #-657    (9)
IRPE$V_EXTEND              =0000000B                  #-593    (7)    #-663    (9)
IRPE$W_BOFF1                00000030                  #-653    (9)
IRPE$W_BOFF2                0000003C                  #-659    (9)
IRPE$W_STS                  0000002A                    593    (7)      663    (9)
MMG$UNLOCK                  00000000-XR                 215    (3)      677    (9)
MOVBUF                      00000349-R   605    (8)   #-486    (6)    #-553    (7)
NOTACP                      000000A1-R   234    (3)   #-232    (3)
NOTFCPWCB                   000001D2-R   436    (5)   #-406    (5)
PAGIO_OR_SWAPIO             00000085-R   225    (3)   #-198    (3)
PCB$W_BIOCNT                0000003A                  #-231    (3)
PCB$W_DIOCNT                0000003E                  #-192    (3)    #-233    (3)    #-316    (4)    #-363    (5)
PR$_IPL                    =00000012                  #-428    (5)    #-430    (5)
PR$_SIRR                   =00000014                  #-387    (5)
PRI$_IOCOM                 =00000001                    134    (2)      136    (2)
PRI$_TICOM                 =00000004                    137    (2)
PRI$_TOCOM                 =00000003                    135    (2)
PRITBL                      00000000-R   133    (2)   #-246    (3)
QNXTSEG                     00000119-R   292    (4)
SCH$GL_PCBVEC               00000000-XR               #-189    (3)
SCH$POSTEF                  00000000-XR               #-248    (3)
SCH$QAST                    00000000-XR                 251    (3)      581    (7)
SS$_ACCVIO                  00000000-XR               #-620    (8)
SS$_ILLBLKNUM               00000000-XR               #-383    (5)
UCB$L_DEVCHAR               00000034                    287    (4)
UCB$L_MAXBLOCK              00000084                  #-373    (5)
UNLK                        000003B0-R   675    (9)   #-655    (9)    #-661    (9)
```

```
UNLOCK                          0000006A-R  213   (3)    #-208    (3)    #-303    (4)
UNLOCK_MORE                     00000381-R  645   (9)    #-217    (3)
VA$M_BYTE                      =000001FF                 #-472    (6)    #-492    (6)    #-610    (8)
VA$S_BYTE                      =00000009                 #-214    (3)    #-282    (4)    #-291    (4)    #-370    (5)
                                                         #-676    (9)
VIRTUAL                         000000E8-R  279   (4)    #-180    (3)
WCB$B_ACCESS                    0000000B                   405    (5)
WCB$V_NOTFCP                   =00000002                 #-405    (5)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	104 (2)	104 (2)
$AQBDEF	2	105 (2)	105 (2)
$CADEF	1	106 (2)	106 (2)
$CCBDEF	1	107 (2)	107 (2)
$CXBDEF	2	108 (2)	108 (2)
$DEFINI	1	104 (2)	104 (2) 105 (2) 106 (2) 107 (2) 108 (2)
			109 (2) 110 (2) 111 (2) 112 (2) 113 (2)
			114 (2) 115 (2) 116 (2) 117 (2) 118 (2)
			119 (2) 120 (2) 121 (2) 122 (2) 123 (2)
			124 (2)
$IPLDEF	1	109 (2)	109 (2)
$IRPDEF	4	110 (2)	110 (2)
$IRPEDEF	2	111 (2)	111 (2)
$JIBDEF	3	112 (2)	112 (2)
$PCBDEF	4	113 (2)	113 (2)
$PFNDEF	2	114 (2)	114 (2)
$PHDDEF	6	115 (2)	115 (2)
$PRDEF	5	116 (2)	116 (2)
$PRIDEF	1	117 (2)	117 (2)
$PTEDEF	3	118 (2)	118 (2)
$PUSHADR	1	226 (3)	226 (3)
$RSNDEF	1	119 (2)	119 (2)
$UCBDEF	10	120 (2)	120 (2)
$VADEF	1	121 (2)	121 (2)
$VCBDEF	6	122 (2)	122 (2)
$WCBDEF	3	123 (2)	123 (2)
$WQHDEF	1	124 (2)	124 (2)
ASSUME	1	195 (3)	195 (3) 196 (3) 643 (9)
BUG_CHECK	1	429 (5)	429 (5) 437 (5)
DSBINT	1	428 (5)	428 (5)
ENBINT	1	430 (5)	430 (5)
ERRSUP_S	1	226 (3)	226 (3)
IFNOWRT	1	476 (6)	476 (6) 496 (6) 576 (7) 614 (8)
MODNAM	1	130 (2)	130 (2)
SOFTINT	1	387 (5)	387 (5)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	283 (4) 371 (5) 471 (6) 482 (6) 491 (6) 609 (8)
ADDW	00A0	281 (4)
ASHL	0078	214 (3) 282 (4) 291 (4) 370 (5) 676 (9)
BBC	00E1	190 (3) 208 (3) 216 (3) 232 (3) 465 (6) 548 (7) 579 (7) 593 (7)
BBS	00E0	238 (3) 242 (3) 287 (4) 405 (5) 466 (6) 578 (7) 663 (9)
BCS	001F	372 (5)
BEQL	0013	205 (3) 236 (3) 290 (4) 366 (5) 469 (6) 476 (6) 496 (6) 507 (6) 514 (6) 562 (7) 574 (7) 576 (7) 607 (8) 614 (8) 652 (9) 658 (9)
BGEQU	001E	374 (5) 502 (6)
BGTR	0014	479 (6) 499 (6) 617 (8)
BICW	00AA	243 (3) 317 (4) 472 (6) 492 (6) 610 (8)
BISB	0088	250 (3)
BITB	0093	197 (3)
BLBC	00E9	209 (3) 285 (4)
BLSS	0019	186 (3)
BNEQ	0012	198 (3) 212 (3) 386 (5) 509 (6) 560 (7) 597 (7)
BRB	0011	183 (3) 219 (3) 321 (4) 484 (6) 485 (6) 488 (6) 510 (6) 517 (6) 598 (7)
BRW	0031	180 (3) 191 (3) 252 (3) 297 (4) 303 (4) 583 (7)
BSBB	0010	296 (4) 320 (4) 553 (7) 655 (9) 661 (9)
BSBW	0030	217 (3) 239 (3) 248 (3) 486 (6) 511 (6) 515 (6) 518 (6) 520 (6) 556 (7) 566 (7) 595 (7)
BVC	001C	174 (3)
CALLS	00FB	226 (3)
CLRL	00D4	384 (5) 563 (7) 580 (7) 592 (7)
CMPL	00D1	373 (5)
CMPW	00B1	210 (3) 501 (6)
CVTWL	0032	475 (6) 495 (6) 557 (7) 613 (8)
DECL	00D7	369 (5)
DECW	00B7	316 (4) 363 (5) 559 (7)
EXTZV	00EF	241 (3) 494 (6) 575 (7) 612 (8)
INCW	00B6	192 (3) 231 (3) 233 (3) 564 (7)
INSQUE	000E	385 (5)
JMP	0017	376 (5) 581 (7)
JSB	0016	182 (3) 215 (3) 251 (3) 364 (5) 375 (5) 429 (5) 437 (5) 677 (9)
MFPR	00DB	428 (5)
MOVAB	009E	213 (3) 237 (3) 240 (3) 558 (7) 675 (9)
MOVAL	00DE	293 (4)
MOVAW	003E	478 (6) 498 (6) 616 (8)
MOVC	0028	480 (6) 504 (6) 618 (8)
MOVL	00D0	185 (3) 189 (3) 193 (3) 235 (3) 245 (3) 286 (4) 294 (4) 295 (4) 302 (4) 315 (4) 318 (4) 348 (5) 350 (5) 367 (5) 404 (5) 407 (5) 463 (6) 467 (6) 470 (6) 481 (6) 487 (6) 489 (6) 490 (6) 500 (6) 505 (6) 508 (6) 512 (6) 513 (6) 516 (6) 519 (6) 550 (7) 555 (7) 561 (7) 565 (7) 573 (7) 582 (7) 590 (7) 594 (7) 596 (7) 606 (8) 608 (8) 650 (9) 651 (9) 654 (9) 657 (9) 660 (9)

MOVQ	007D	170	(3)	171	(3)	172	(3)	175	(3)	176	(3)	177	(3)	577	(7)
MOVW	00B0	284	(4)	403	(5)	503	(6)	620	(8)						
MOVZBL	009A	246	(3)	247	(3)										
MOVZWL	003C	188	(3)	206	(3)	207	(3)	280	(4)	301	(4)	319	(4)	349	(5)
		368	(5)	383	(5)	464	(6)	468	(6)	474	(6)	551	(7)	653	(9)
		659	(9)												
MTPR	00DA	387	(5)	428	(5)	430	(5)								
POPL	8ED0	664	(9)												
POPR	00BA	521	(6)	554	(7)										
PROBEW	000D	476	(6)	496	(6)	576	(7)	614	(8)						
PUSHAL	00DF	226	(3)												
PUSHL	00DD	226	(3)	646	(9)										
PUSHR	00BB	462	(6)	549	(7)										
REI	0002	178	(3)												
REMQUE	000F	173	(3)												
RSB	0005	388	(5)	431	(5)	619	(8)	621	(8)	665	(9)	678	(9)		
SOBGTR	00F5	483	(6)												
SUBL	00C2	473	(6)	477	(6)	493	(6)	497	(6)	552	(7)	611	(8)	615	(8)
SUBW	00A2	506	(6)												
SUBW3	00A3	288	(4)	365	(5)										
TSTL	00D5	204	(3)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	130	(2)	131	(2)	429	(5)	437	(5)								
.BYTE	134	(2)	135	(2)	136	(2)	137	(2)								
.DSABL	253	(3)	323	(4)												
.ENABL	168	(3)	277	(4)												
.END	681	(9)														
.ENDC	195	(3)	196	(3)	226	(3)	428	(5)	430	(5)	643	(9)				
.ENDM	104	(2)	105	(2)	106	(2)	107	(2)	108	(2)	109	(2)	110	(2)	111	(2)
	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)	117	(2)	118	(2)	119	(2)
	120	(2)	121	(2)	122	(2)	123	(2)	124	(2)	130	(2)	195	(3)	226	(3)
	387	(5)	428	(5)	429	(5)	430	(5)	476	(6)						
.IDENT	5	(1)														
.IF	195	(3)	196	(3)	226	(3)	428	(5)	430	(5)	643	(9)				
.IFF	195	(3)	196	(3)	226	(3)	428	(5)	430	(5)	643	(9)				
.IIF	226	(3)														
.NOCROSS	104	(2)	105	(2)	106	(2)	107	(2)	108	(2)	109	(2)	110	(2)	111	(2)
	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)	117	(2)	118	(2)	119	(2)
	120	(2)	121	(2)	122	(2)	123	(2)	124	(2)						
.PSECT	129	(2)	132	(2)												
.RESTORE	104	(2)	105	(2)	106	(2)	107	(2)	108	(2)	109	(2)	110	(2)	111	(2)
	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)	117	(2)	118	(2)	119	(2)
	120	(2)	121	(2)	122	(2)	123	(2)	124	(2)						
.SAVE	104	(2)	105	(2)	106	(2)	107	(2)	108	(2)	109	(2)	110	(2)	111	(2)
	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)	117	(2)	118	(2)	119	(2)
	120	(2)	121	(2)	122	(2)	123	(2)	124	(2)						
.SBTTL	100	(2)	139	(3)	256	(4)	325	(5)	439	(6)	44	(1)	523	(7)	600	(8)
	623	(9)														
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	48	#-172	(3)	#-175	(3)	#-235	(3)	#-241	(3)	#-243	(3)	#-246	(3)
		#-280	(4)	#-281	(4)	#-282	(4)	#-283	(4)	#-286	(4)	287	(4)
		#-350	(5)	#-367	(5)	#-468	(6)	#-471	(6)	#-473	(6)	#-476	(6)
		478	(6)	#-489	(6)	#-491	(6)	#-493	(6)	#-496	(6)	498	(6)
		#-512	(6)	#-513	(6)	#-516	(6)	#-519	(6)	#-555	(7)	#-557	(7)
		558	(7)	#-573	(7)	576	(7)	#-577	(7)	#-582	(7)	#-590	(7)
		593	(7)	#-594	(7)	#-596	(7)	#-606	(8)	#-609	(8)	#-611	(8)
		#-614	(8)	616	(8)								
R1	50	182	(3)	#-185	(3)	#-188	(3)	#-189	(3)	#-206	(3)	213	(3)
		#-214	(3)	#-245	(3)	#-291	(4)	293	(4)	#-301	(4)	#-349	(5)
		#-367	(5)	#-371	(5)	#-470	(6)	#-471	(6)	#-472	(6)	#-473	(6)
		476	(6)	#-477	(6)	#-490	(6)	#-491	(6)	#-492	(6)	#-493	(6)
		496	(6)	#-497	(6)	#-558	(7)	#-559	(7)	#-561	(7)	#-563	(7)
		#-564	(7)	#-565	(7)	#-575	(7)	#-576	(7)	#-608	(8)	#-609	(8)
		#-610	(8)	#-611	(8)	614	(8)	#-615	(8)	#-654	(9)	#-660	(9)
		675	(9)	#-676	(9)								
R2	27	#-171	(3)	#-176	(3)	#-207	(3)	213	(3)	#-246	(3)	#-319	(4)
		#-348	(5)	#-365	(5)	#-368	(5)	#-369	(5)	#-370	(5)	#-371	(5)
		#-373	(5)	#-403	(5)	#-404	(5)	405	(5)	#-407	(5)	#-494	(6)
		#-496	(6)	#-565	(7)	#-580	(7)	#-612	(8)	#-614	(8)	#-653	(9)
		#-659	(9)	675	(9)								
R3	38	#-193	(3)	#-204	(3)	#-247	(3)	293	(4)	#-294	(4)	#-295	(4)
		#-302	(4)	#-315	(4)	#-317	(4)	#-318	(4)	#-319	(4)	#-348	(5)
		#-349	(5)	#-350	(5)	#-365	(5)	#-368	(5)	#-383	(5)	#-384	(5)
		385	(5)	#-403	(5)	#-404	(5)	#-407	(5)	#-475	(6)	#-477	(6)
		478	(6)	#-495	(6)	#-497	(6)	498	(6)	#-500	(6)	504	(6)
		#-561	(7)	#-613	(8)	#-615	(8)	616	(8)	#-651	(9)	#-657	(9)
R4	14	#-170	(3)	#-177	(3)	#-189	(3)	#-192	(3)	#-231	(3)	#-233	(3)
		#-316	(4)	#-363	(5)	#-474	(6)	#-476	(6)	480	(6)	#-592	(7)
		#-594	(7)	#-596	(7)								
R5	77	#-173	(3)	#-185	(3)	190	(3)	#-193	(3)	#-197	(3)	#-206	(3)
		#-207	(3)	208	(3)	#-209	(3)	#-210	(3)	216	(3)	232	(3)
		#-235	(3)	#-237	(3)	238	(3)	#-240	(3)	241	(3)	242	(3)
		#-245	(3)	#-247	(3)	#-250	(3)	#-280	(4)	#-281	(4)	#-283	(4)
		#-284	(4)	#-285	(4)	#-286	(4)	#-288	(4)	#-289	(4)	#-293	(4)
		#-294	(4)	#-295	(4)	#-301	(4)	#-302	(4)	#-315	(4)	#-373	(5)
		#-462	(6)	#-463	(6)	#-464	(6)	465	(6)	466	(6)	#-481	(6)
		#-487	(6)	494	(6)	#-505	(6)	#-512	(6)	#-519	(6)	#-521	(6)
		548	(7)	#-549	(7)	#-550	(7)	#-554	(7)	#-555	(7)	#-557	(7)
		#-573	(7)	575	(7)	#-577	(7)	578	(7)	579	(7)	#-582	(7)
		#-590	(7)	612	(8)	#-620	(8)	#-646	(9)	#-650	(9)	#-651	(9)
		#-653	(9)	#-654	(9)	#-657	(9)	#-659	(9)	#-660	(9)	663	(9)
		#-664	(9)										
R6	31	#-462	(6)	#-463	(6)	#-467	(6)	#-468	(6)	#-470	(6)	#-474	(6)
		#-476	(6)	#-480	(6)	#-482	(6)	#-490	(6)	#-500	(6)	#-501	(6)
		#-503	(6)	#-504	(6)	#-506	(6)	#-508	(6)	#-513	(6)	#-516	(6)
		#-521	(6)	#-549	(7)	#-550	(7)	#-551	(7)	#-554	(7)	#-608	(8)
		618	(8)										
R7	14	#-462	(6)	#-464	(6)	#-483	(6)	#-489	(6)	#-501	(6)	#-503	(6)
		#-506	(6)	#-521	(6)	#-549	(7)	#-551	(7)	#-552	(7)	#-554	(7)

```
                  #-606    (8)    #-618    (8)
SP          11    #-170    (3)    #-171    (3)    #-172    (3)    #-175    (3)    #-176    (3)    #-177    (3)
                  #-428    (5)    #-430    (5)    #-481    (6)    #-487    (6)    #-505    (6)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.01	00:00:00.26
Command processing	898	00:00:00.26	00:00:00.78
Pass 1	909	00:00:03.57	00:00:05.27
Symbol table sort	0	00:00:00.28	00:00:00.27
Pass 2	76	00:00:01.01	00:00:01.84
Symbol table output	2	00:00:00.09	00:00:00.19
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	5	00:00:00.24	00:00:00.40
Assembler run totals	1930	00:00:05.47	00:00:09.02

```
The working set limit was 2900 pages.
186284 bytes (364 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1018 non-local and 44 local symbols.
681 source lines were read in Pass 1, producing 69 object records in Pass 2.
124 pages of virtual memory were used to define 39 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	10
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	3
SYS$COMMON:[SYSLIB]LIB.MLB;2	14
SYS$COMMON:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	35

```
1512 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:IOPOST.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1)	91	CANCEL I/O ON CHANNEL
(1)	125	FILL DIAGNOSTIC BUFFER
(1)	160	RELEASE I/O CHANNEL
(1)	215	REQUEST I/O CHANNEL
(1)	273	I/O REQUEST COMPLETION PROCESSING
(1)	317	INITIATE I/O FUNCTION ON DEVICE
(1)	354	RELEASE BUFFERED DATAPATH
(1)	402	REQUEST BUFFERED DATAPATH
(1)	456	RELEASE UNIBUS MAP REGISTERS
(1)	509	REQUEST UNIBUS MAP REGISTERS
(1)	544	ALTER UBA MAP REGISTER BITMAP
(1)	577	SEARCH MAP REGISTER BITMAP AND ALLOCATE MAP REGISTERS
(1)	630	RETURN TO CALLER
(1)	649	WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
(1)	683	WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
(1)	719	ALLOCATE SYSTEM PAGE TABLE

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element IOSNPG.MAR
0000     2 ; *2    26-OCT-1987 09:35:04 SALEM "DONE"
0000     3 ; *1     6-FEB-1986 15:18:58 DS ""
0000     4 ; DEC/CMS REPLACEMENT HISTORY, Element IOSNPG.MAR
0000     5         .TITLE  IOSNPG  *** IOSNPG services for QIO
0000     6         .IDENT  /06-04/                                                    ;G:2
0000     7 ;*****************************************************************************
0000     8 ;*                                                                           *
0000     9 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
0000    10 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    11 ;*                                                                           *
0000    12 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    13 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    14 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    15 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    16 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    17 ;*  TRANSFERRED.                                                             *
0000    18 ;*                                                                           *
0000    19 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    20 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    21 ;*  CORPORATION.                                                             *
0000    22 ;*                                                                           *
0000    23 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    24 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ; D. N. CUTLER 13-JUN-76
0000    29 ;
0000    30 ;
0000    31 ; NONPAGED I/O RELATED SUBROUTINES
0000    32 ;
0000    33 ; MODIFICATION HISTORY:
0000    34 ;
0000    35 ;
0000    36 ;       CHANGE PARAMETERS TO WAIT FOR INTERRUPTS.
0000    37 ;
0000    38 ;               Dave Butenhof   13-may-1980, Version 5.4
0000    39 ;       01      Modify VMS V2.0 sources for Supervisor environment
0000    40 ;               Roger Riggs, 23-july-1980, Version 5.5
0000    41 ;       02      Changed word offsets to longword
0000    42 ;               Dave Butenhof   24-nov-1980, Version 6.2
0000    43 ;       03      Restore entry point IOC$ALLOSPT to satisfy device drivers
0000    44 ;               using it.  It remains essentially "dummied out," since
0000    45 ;               Supervisor I/O drivers do not require system space
0000    46 ;                                                                          ;G:2
0000    47 ;       04      TED SALEM       14-JAN-87       VERSION 10.X              ;G:2
0000    48 ;               Changed BSBW to JSB for linking with /system=10000         ;G:2
0000    49 ;                - NI systems link.                                        ;G:2
0000    50 ;                                                                          ;G:2
0000    51 ;       0206    KDM0086         KATHLEEN D. MORSE       07-JAN-1980
0000    52 ;               CHANGE UCB$W_DEVSTS TO UCB$W_STS, AND EMB$W_DV_DEVSTS TO
0000    53 ;               EMB$W_DV_STS.
0000    54 ;
0000    55 ;       0205    RIH0033         R. HUSTVEDT             16-OCT-1979
0000    56 ;               CHANGE PCB$W_BYTLM TO JIB$L_BYTLM.
0000    57 ;
```

```
                        0000     58 ;        0204    NPK0002         N. KRONENBERG           21-SEP-1979
                        0000     59 ;                MODIFIED IOC$RELDATAP TO ZERO THE DATAPATH
                        0000     60 ;                NUMBER IN THE CALLERS CRB.
                        0000     61 ;
                        0000     62 ;        0203    NPKCOMET        N. KRONENBERG           11-FEB-1979
                        0000     63 ;                MODIFIED IOC$REQDATAP AND IOC$RELDATAP TO HANDLE
                        0000     64 ;                6 BIT DATAPATH NUMBERS.
                        0000     65 ;
                        0000     66 ;        0202    LMK0001         LEN KAWELL              07-FEB-1979
                        0000     67 ;                ADDED IOC$ALLOSPT ROUTINE THAT ALLOCATES SYSTEM
                        0000     68 ;                PAGE TABLE ENTRIES.
```

```
                        0000     70 ;
                        0000     71 ;
                        0000     72 ; MACRO LIBRARY CALLS
                        0000     73 ;
                        0000     74
                        0000     75          $ADPDEF                             ;DEFINE ADP OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,ADP$L_CSR,   ADP$L_CSR:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,ADP$L_LINK,  ADP$L_LINK:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,ADP$W_SIZE,  ADP$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,ADP$B_TYPE,  ADP$B_TYPE:
                0000000B 000A                        .BLKB   1
                        000B                 .IIF    NB,ADP$B_NUMBER,        ADP$B_NUMBER:
                0000000C 000B                        .BLKB   1
                        000C                 .IIF    NB,ADP$W_TR,    ADP$W_TR:
                0000000E 000C                        .BLKW   1
                        000E                 .IIF    NB,ADP$W_ADPTYPE,       ADP$W_ADPTYPE:
                00000010 000E                        .BLKW   1
                        0010                 .IIF    NB,ADP$L_VECTOR,        ADP$L_VECTOR:
                        0010                 .IIF    NB,ADP$L_CRB,   ADP$L_CRB:
                00000014 0010                        .BLKL   1
                        0014                 .IIF    NB,ADP$C_MBAADPLEN,     ADP$C_MBAADPLEN:
                        0014                 .IIF    NB,ADP$K_MBAADPLEN,     ADP$K_MBAADPLEN:
                        0014                 .IIF    NB,ADP$C_DRADPLEN,      ADP$C_DRADPLEN:
                        0014                 .IIF    NB,ADP$K_DRADPLEN,      ADP$K_DRADPLEN:
                        0014                 .IIF    NB,ADP$L_DPQFL, ADP$L_DPQFL:
                        0014                 .IIF    NB,ADP$L_PRQQFL,        ADP$L_PRQQFL:
                00000018 0014                        .BLKL   1
                        0018                 .IIF    NB,ADP$L_DPQBL, ADP$L_DPQBL:
                        0018                 .IIF    NB,ADP$L_PRQQBL,        ADP$L_PRQQBL:
                0000001C 0018                        .BLKL   1
                        001C                 .IIF    NB,ADP$L_MRQFL, ADP$L_MRQFL:
                        001C                 .IIF    NB,ADP$L_SHB,   ADP$L_SHB:
                00000020 001C                        .BLKL   1
                        0020                 .IIF    NB,ADP$L_MRQBL, ADP$L_MRQBL:
                        0020                 .IIF    NB,ADP$B_PORT,  ADP$B_PORT:
                00000021 0020                        .BLKB   1
                00000024 0021                        .BLKB   3
                        0024                 .IIF    NB,ADP$W_DPBITMAP,      ADP$W_DPBITMAP:
                00000026 0024                        .BLKW   1
                        0026                 .IIF    NB,ADP$W_MRBITMAP,      ADP$W_MRBITMAP:
                00000064 0026                        .BLKW   31
                        0064                 .IIF    NB,ADP$L_INTD,  ADP$L_INTD:
                00000070 0064                        .BLKL   3
                        0070                 .IIF    NB,ADP$C_MPMADPLEN,     ADP$C_MPMADPLEN:
                        0070                 .IIF    NB,ADP$K_MPMADPLEN,     ADP$K_MPMADPLEN:
                        0070                 .IIF    NB,ADP$C_UBAADPLEN,     ADP$C_UBAADPLEN:
                        0070                 .IIF    NB,ADP$K_UBAADPLEN,     ADP$K_UBAADPLEN:
                        0000                 .RESTORE
                        0000     76          $CADEF                              ;DEFINE CONDITIONAL ASSEMBLY PARAMETERS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 0070                .=0
                        0000                 .RESTORE
```

```
                    0000     77          $CRBDEF                                 ;DEFINE CRB OFFSETS
                    0000                 .SAVE   LOCAL_BLOCK
                    0000                 .PSECT  $ABS$,ABS
          00000000  0070                 .=0
                    0000                 .IIF    NB,CRB$L_WQFL,  CRB$L_WQFL:
          00000004  0000                         .BLKL   1
                    0004                 .IIF    NB,CRB$L_WQBL,  CRB$L_WQBL:
          00000008  0004                         .BLKL   1
                    0008                 .IIF    NB,CRB$W_SIZE,  CRB$W_SIZE:
          0000000A  0008                         .BLKW   1
                    000A                 .IIF    NB,CRB$B_TYPE,  CRB$B_TYPE:
          0000000B  000A                         .BLKB   1
          0000000C  000B                         .BLKB   1
                    000C                 .IIF    NB,CRB$W_REFC,  CRB$W_REFC:
          0000000E  000C                         .BLKW   1
                    000E                 .IIF    NB,CRB$B_MASK,  CRB$B_MASK:
          0000000F  000E                         .BLKB   1
          00000010  000F                         .BLKB   1
                    0010                 .IIF    NB,CRB$L_LINK,  CRB$L_LINK:
          00000014  0010                         .BLKL   1
                    0014                 .IIF    NB,CRB$L_INTD,  CRB$L_INTD:
          00000038  0014                         .BLKL   9
                    0038                 .IIF    NB,CRB$C_LENGTH,        CRB$C_LENGTH:
                    0038                 .IIF    NB,CRB$K_LENGTH,        CRB$K_LENGTH:
                    0038                 .IIF    NB,CRB$L_INTD2, CRB$L_INTD2:
          0000005C  0038                         .BLKL   9
                    0000                 .RESTORE
                    0000     78          $DDBDEF                                 ;DEFINE DDB OFFSETS
                    0000                 .SAVE   LOCAL_BLOCK
                    0000                 .PSECT  $ABS$,ABS
          00000000  0070                 .=0
                    0000                 .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
          00000004  0000                         .BLKL   1
                    0004                 .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
          00000008  0004                         .BLKL   1
                    0008                 .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
          0000000A  0008                         .BLKW   1
                    000A                 .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
          0000000B  000A                         .BLKB   1
          0000000C  000B                         .BLKB   1
                    000C                 .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
          00000010  000C                         .BLKL   1
                    0010                 .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
          00000013  0010                         .BLKB   3
                    0013                 .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
          00000014  0013                         .BLKB   1
                    0014                 .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
          00000015  0014                         .BLKB   1
          00000024  0015                         .BLKB   15
                    0024                 .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
          00000025  0024                         .BLKB   1
          00000034  0025                         .BLKB   15
                    0034                 .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                    0034                 .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                    0000                 .RESTORE
                    0000     79          $DDTDEF                                 ;DEFINE DDT OFFSETS
                    0000                 .SAVE   LOCAL_BLOCK
```

```
                 0000                .PSECT  $ABS$,ABS
        00000000 0070                .=0
                 0000                .IIF    NB,DDT$L_START, DDT$L_START:
        00000004 0000                                .BLKL   1
                 0004                .IIF    NB,DDT$L_UNSOLINT,      DDT$L_UNSOLINT:
        00000008 0004                                .BLKL   1
                 0008                .IIF    NB,DDT$L_FDT,   DDT$L_FDT:
        0000000C 0008                                .BLKL   1
                 000C                .IIF    NB,DDT$L_CANCEL,        DDT$L_CANCEL:
        00000010 000C                                .BLKL   1
                 0010                .IIF    NB,DDT$L_REGDUMP,       DDT$L_REGDUMP:
        00000014 0010                                .BLKL   1
                 0014                .IIF    NB,DDT$W_DIAGBUF,       DDT$W_DIAGBUF:
        00000016 0014                                .BLKW   1
                 0016                .IIF    NB,DDT$W_ERRORBUF,      DDT$W_ERRORBUF:
        00000018 0016                                .BLKW   1
                 0018                .IIF    NB,DDT$L_UNITINIT,      DDT$L_UNITINIT:
        0000001C 0018                                .BLKL   1
                 001C                .IIF    NB,DDT$L_ALTSTART,      DDT$L_ALTSTART:
        00000020 001C                                .BLKL   1
                 0000                .RESTORE
                 0000     80         $EMBDEF                         ;DEFINE EMB OFFSETS
                 0000                .SAVE   LOCAL_BLOCK
                 0000                .PSECT  $ABS$,ABS
        00000000 0070                .=0
                 0000                .RESTORE
                 0000                .SAVE   LOCAL_BLOCK
                 0000                .PSECT  $ABS$,ABS
        00000000 0070                .=0
                 0000                .RESTORE
                 0000                .SAVE   LOCAL_BLOCK
                 0000                .PSECT  $ABS$,ABS
        00000000 0070                .=0
                 0000                .RESTORE
                 0000                .SAVE   LOCAL_BLOCK
                 0000                .PSECT  $ABS$,ABS
        00000000 0070                .=0
                 0000                .RESTORE
                 0000     81         $IDBDEF                         ;DEFINE IDB OFFSETS
                 0000                .SAVE   LOCAL_BLOCK
                 0000                .PSECT  $ABS$,ABS
        00000000 0070                .=0
                 0000                .IIF    NB,IDB$L_CSR,   IDB$L_CSR:
        00000004 0000                                .BLKL   1
                 0004                .IIF    NB,IDB$L_OWNER, IDB$L_OWNER:
        00000008 0004                                .BLKL   1
                 0008                .IIF    NB,IDB$W_SIZE,  IDB$W_SIZE:
        0000000A 0008                                .BLKW   1
                 000A                .IIF    NB,IDB$B_TYPE,  IDB$B_TYPE:
        0000000B 000A                                .BLKB   1
        0000000C 000B                                .BLKB   1
                 000C                .IIF    NB,IDB$W_UNITS, IDB$W_UNITS:
        0000000E 000C                                .BLKW   1
        00000010 000E                                .BLKW   1
                 0010                .IIF    NB,IDB$L_ADP,   IDB$L_ADP:
        00000014 0010                                .BLKL   1
                 0014                .IIF    NB,IDB$L_UCBLST,        IDB$L_UCBLST:
```

```
00000034  0014                        .BLKL   8
          0034              .IIF    NB,IDB$C_LENGTH,        IDB$C_LENGTH:
          0034              .IIF    NB,IDB$K_LENGTH,        IDB$K_LENGTH:
          0000              .RESTORE
          0000    82        $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .RESTORE
          0000    83        $IRPDEF                         ;DEFINE IRP OFFSETS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
00000004  0000                      .BLKL   1
          0004              .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
00000008  0004                      .BLKL   1
          0008              .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
0000000A  0008                      .BLKW   1
          000A              .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
0000000B  000A                      .BLKB   1
          000B              .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
0000000C  000B                      .BLKB   1
          000C              .IIF    NB,IRP$L_PID,   IRP$L_PID:
00000010  000C                      .BLKL   1
          0010              .IIF    NB,IRP$L_AST,   IRP$L_AST:
00000014  0010                      .BLKL   1
          0014              .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
00000018  0014                      .BLKL   1
          0018              .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
0000001C  0018                      .BLKL   1
          001C              .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
00000020  001C                      .BLKL   1
          0020              .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
00000022  0020                      .BLKW   1
          0022              .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
00000023  0022                      .BLKB   1
          0023              .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
00000024  0023                      .BLKB   1
          0024              .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
00000028  0024                      .BLKL   1
          0028              .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
0000002A  0028                      .BLKW   1
          002A              .IIF    NB,IRP$W_STS,   IRP$W_STS:
0000002C  002A                      .BLKW   1
          002C              .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
00000030  002C                      .BLKL   1
          0030              .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
00000032  0030                      .BLKW   1
          0032              .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
00000034  0032                      .BLKW   1
          0034              .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
          0034              .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
00000038  0034                      .BLKL   1
          0038              .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
          0038              .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
          0038              .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
```

```
00000039  0038                          .BLKB   1
0000003C  0039                          .BLKB   3
          003C              .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
          003C              .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
          003C              .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                          .BLKW   1
          003E              .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                          .BLKW   1
          0040              .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                          .BLKL   1
          0044              .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                          .BLKL   1
          0048              .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                          .BLKL   1
          004C              .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                          .BLKL   1
          0050              .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                          .BLKL   1
00000058  0054                          .BLKL   1
0000005C  0058                          .BLKL   1
          005C              .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
          005C              .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
          0000              .RESTORE
          0000    84        $JIBDEF                         ;DEFINE JIB OFFSETS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .RESTORE
          0000    85        $LOGDEF                         ;DEFINE LOGICAL NAME BLOCK OFFSETS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .RESTORE
          0000    86        $PCBDEF                         ;DEFINE PCB OFFSETS
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004  0000                      .BLKL   1
          0004              .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008  0004                      .BLKL   1
          0008              .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A  0008                      .BLKW   1
          000A              .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B  000A                      .BLKB   1
          000B              .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C  000B                      .BLKB   1
          000C              .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D  000C                      .BLKB   1
          000D              .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E  000D                      .BLKB   1
          000E              .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010  000E                      .BLKW   1
          0010              .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014  0010                      .BLKL   1
          0014              .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018  0014                      .BLKL   1
```

```
                    0018            .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C  0018                              .BLKL   1
                    001C            .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020  001C                              .BLKL   1
                    0020            .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024  0020                              .BLKL   1
                    0024            .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028  0024                              .BLKL   1
                    0028            .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C  0028                              .BLKL   1
                    002C            .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E  002C                              .BLKW   1
                    002E            .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                              .BLKB   1
                    002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                              .BLKB   1
                    0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                              .BLKW   1
                    0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                              .BLKW   1
                    0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                              .BLKW   1
                    0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                              .BLKW   1
                    0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                              .BLKW   1
                    003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                              .BLKW   1
                    003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                              .BLKW   1
                    003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                              .BLKW   1
                    0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                              .BLKW   1
                    0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                              .BLKW   1
                    0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                              .BLKB   8
                    004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                    004C            .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                              .BLKL   1
                    0050            .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                              .BLKL   1
                    0054            .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                              .BLKL   1
                    0058            .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                              .BLKL   1
                    005C            .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                              .BLKL   1
                    0060            .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                              .BLKL   1
                    0064            .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                              .BLKL   1
                    0068            .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                              .BLKB   16
                    0078            .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                              .BLKL   1
```

```
                    007C                .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
          00000084  007C                                .BLKQ   1
                    0084                .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
          00000088  0084                                .BLKL   1
                    0088                .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                    0088                .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
          0000008A  0088                                .BLKW   1
                    008A                .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
          0000008C  008A                                .BLKW   1
                    008C                .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                    008C                .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                    0000                .RESTORE
                    0000      87        $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  008C                .=0
                    0000                .RESTORE
                    0000      88        $UCBDEF                         ;DEFINE UCB OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  008C                .=0
                    0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                    0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
          00000004  0000                                .BLKL   1
                    0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                    0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
          00000008  0004                                .BLKL   1
                    0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
          0000000A  0008                                .BLKW   1
                    000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
          0000000B  000A                                .BLKB   1
                    000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
          0000000C  000B                                .BLKB   1
                    000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
          0000000D  000C                                .BLKB   1
          00000010  000D                                .BLKB   3
                    0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
          00000014  0010                                .BLKL   1
                    0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
          00000016  0014                                .BLKW   1
                    0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
          00000018  0016                                .BLKW   1
                    0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
          0000001A  0018                                .BLKW   1
                    001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
          0000001C  001A                                .BLKW   1
                    001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
          00000020  001C                                .BLKL   1
                    0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
          00000024  0020                                .BLKL   1
```

```
                    0024          .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
        00000028    0024                          .BLKL   1
                    0028          .IIF    NB,UCB$L_PID,   UCB$L_PID:
        0000002C    0028                          .BLKL   1
                    002C          .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
        00000030    002C                          .BLKL   1
                    0030          .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
        00000034    0030                          .BLKL   1
                    0034          .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
        00000038    0034                          .BLKL   1
                    0038          .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
        00000039    0038                          .BLKB   1
                    0039          .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
        0000003A    0039                          .BLKB   1
                    003A          .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
        0000003C    003A                          .BLKW   1
                    003C          .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C          .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C          .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
        0000003D    003C                          .BLKB   1
                    003D          .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                    003D          .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
        0000003E    003D                          .BLKB   1
                    003E          .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                    003E          .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
        0000003F    003E                          .BLKB   1
                    003F          .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
        00000040    003F                          .BLKB   1
                    0040          .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
        00000044    0040                          .BLKL   1
                    0044          .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
        00000048    0044                          .BLKL   1
                    0048          .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
        0000004A    0048                          .BLKW   1
                    004A          .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                    004A          .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
        0000004B    004A                          .BLKB   1
                    004B          .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
        0000004C    004B                          .BLKB   1
                    004C          .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
        00000050    004C                          .BLKL   1
                    0050          .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
        00000052    0050                          .BLKW   1
                    0052          .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                    0052          .IIF    NB,UCB$B_STATE, UCB$B_STATE:
        00000053    0052                          .BLKB   1
                    0053          .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
        00000054    0053                          .BLKB   1
                    0054          .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
        00000058    0054                          .BLKL   1
                    0058          .IIF    NB,UCB$W_STS,   UCB$W_STS:
        0000005A    0058                          .BLKW   1
                    005A          .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
        0000005C    005A                          .BLKW   1
                    005C          .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                    005C          .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
        00000060    005C                          .BLKL   1
```

```
                    0060          .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
        00000064    0060                          .BLKL   1
                    0064          .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                    0064          .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
        00000068    0064                          .BLKL   1
                    0068          .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
        0000006C    0068                          .BLKL   1
                    006C          .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
        0000006E    006C                          .BLKW   1
                    006E          .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
        00000070    006E                          .BLKW   1
                    0070          .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
        00000071    0070                          .BLKB   1
                    0071          .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
        00000072    0071                          .BLKB   1
                    0072          .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
        00000074    0072                          .BLKW   1
                    0074          .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                    0074          .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
        00000000    0074          . = 0
                    0000          .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
        00000002    0000                          .BLKW   1
        00000074    0002          . = 116
                    0074          .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
        00000078    0074                          .BLKL   1
                    0078          .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
        0000007C    0078                          .BLKL   1
                    007C          .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
        00000080    007C                          .BLKL   1
                    0080          .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
        00000084    0080                          .BLKL   1
                    0084          .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
        00000088    0084                          .BLKL   1
                    0088          .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
        0000008C    0088                          .BLKL   1
                    008C          .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
        00000090    008C                          .BLKL   1
                    0090          .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                    0090          .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
        00000074    0090          . = 116
                    0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
        00000075    0074                          .BLKB   1
                    0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
        00000076    0075                          .BLKB   1
                    0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
        00000077    0076                          .BLKB   1
                    0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
        00000078    0077                          .BLKB   1
                    0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
        0000007C    0078                          .BLKL   1
        0000007E    007C                          .BLKW   1
                    007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
        00000080    007E                          .BLKW   1
                    0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
        00000084    0080                          .BLKL   1
        00000080    0084          . = 128
        00000084    0080                          .BLKL   1
```

```
                 0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
        00000088 0084                          .BLKL   1
                 0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
        0000008A 0088                          .BLKW   1
                 008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
        0000008C 008A                          .BLKW   1
                 008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                 008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
        0000008E 008C                          .BLKW   1
                 008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
        00000090 008E                          .BLKW   1
                 0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
        00000092 0090                          .BLKW   1
                 0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
        00000094 0092                          .BLKW   1
                 0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
        00000095 0094                          .BLKB   1
                 0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
        00000096 0095                          .BLKB   1
                 0096          .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
        00000098 0096                          .BLKW   1
        00000096 0098          . = 150
        00000098 0096                          .BLKW   1
                 0098          .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
        0000009C 0098                          .BLKL   1
                 009C          .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
        000000A0 009C                          .BLKL   1
                 00A0          .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
        000000A4 00A0                          .BLKL   1
                 00A4          .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
        000000A6 00A4                          .BLKW   1
                 00A6          .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
        000000A7 00A6                          .BLKB   1
        000000A8 00A7                          .BLKB   1
        00000074 00A8          . = 116
        00000078 0074                          .BLKL   1
        0000007C 0078                          .BLKL   1
        00000080 007C                          .BLKL   1
        00000084 0080                          .BLKL   1
        00000088 0084                          .BLKL   1
        0000008C 0088                          .BLKL   1
                 008C          .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
        00000090 008C                          .BLKL   1
        00000094 0090                          .BLKL   1
        00000098 0094                          .BLKL   1
        0000009C 0098                          .BLKL   1
                 009C          .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
        0000009D 009C                          .BLKB   1
                 009D          .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
        0000009E 009D                          .BLKB   1
                 009E          .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
        0000009F 009E                          .BLKB   1
        000000A0 009F                          .BLKB   1
                 00A0          .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
        000000A1 00A0                          .BLKB   1
                 00A1          .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
        000000A2 00A1                          .BLKB   1
```

```
                     00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                           .BLKB   1
000000A4  00A3                                           .BLKB   1
                     00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                           .BLKB   1
                     00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                           .BLKW   1
000000A8  00A7                                           .BLKB   1
                     00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                           .BLKL   1
000000B0  00AC                                           .BLKL   1
000000B4  00B0                                           .BLKL   1
000000B8  00B4                                           .BLKL   1
                     00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                           .BLKL   1
                     00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                     00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC                           . = 116
                     0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                                           .BLKL   1
                     0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                           .BLKL   1
                     007C                .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                           .BLKW   1
00000080  007E                                           .BLKW   1
                     0000                .RESTORE
                     0000    89          $VECDEF                         ;DEFINE CRB VECTOR OFFSETS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000  00BC                           .=0
                     0000                .IIF    NB,VEC$Q_DISPATCH,      VEC$Q_DISPATCH:
00000008  0000                                           .BLKQ   1
                     0008                .IIF    NB,VEC$L_IDB,   VEC$L_IDB:
0000000C  0008                                           .BLKL   1
                     000C                .IIF    NB,VEC$L_INITIAL,       VEC$L_INITIAL:
00000010  000C                                           .BLKL   1
                     0010                .IIF    NB,VEC$W_MAPREG,        VEC$W_MAPREG:
00000012  0010                                           .BLKW   1
                     0012                .IIF    NB,VEC$B_NUMREG,        VEC$B_NUMREG:
00000013  0012                                           .BLKB   1
                     0013                .IIF    NB,VEC$B_DATAPATH,      VEC$B_DATAPATH:
00000014  0013                                           .BLKB   1
                     0014                .IIF    NB,VEC$L_ADP,   VEC$L_ADP:
00000018  0014                                           .BLKL   1
                     0018                .IIF    NB,VEC$L_UNITINIT,      VEC$L_UNITINIT:
0000001C  0018                                           .BLKL   1
                     001C                .IIF    NB,VEC$L_START, VEC$L_START:
00000020  001C                                           .BLKL   1
                     0020                .IIF    NB,VEC$L_UNITDISC,      VEC$L_UNITDISC:
00000024  0020                                           .BLKL   1
                     0024                .IIF    NB,VEC$C_LENGTH,        VEC$C_LENGTH:
                     0024                .IIF    NB,VEC$K_LENGTH,        VEC$K_LENGTH:
                     0000                .RESTORE
```

```
                                0000     91          .SBTTL  CANCEL I/O ON CHANNEL
                                0000     92 ;+
                                0000     93 ; IOC$CANCELIO - CANCEL I/O ON CHANNEL
                                0000     94 ;
                                0000     95 ; THIS ROUTINE IS A DEVICE INDEPENDENT CANCEL I/O ROUTINE THAT CONDITIONALLY
                                0000     96 ; MARKS THE UCB SUCH THAT THE CURRENT I/O REQUEST WILL BE CANCELED IF CONDITIONS
                                0000     97 ; WARRANT SUCH A ACTION.
                                0000     98 ;
                                0000     99 ; INPUTS:
                                0000    100 ;
                                0000    101 ;       R2 = NEGATIVE OF THE CHANNEL NUMBER.
                                0000    102 ;       R3 = CURRENT IO PACKET.
                                0000    103 ;       R4 = PCB ADDRESS.
                                0000    104 ;       R5 = UCB ADDRESS.
                                0000    105 ;
                                0000    106 ; OUTPUTS:
                                0000    107 ;
                                0000    108 ;       IF THE DEVICE IS BUSY, THE REQUEST IS FOR THE CURRENT PROCESS, AND
                                0000    109 ;       THE I/O WAS ISSUED FROM THE DESIGNATED CHANNEL, THEN THE CANCEL I/O
                                0000    110 ;       BIT IS SET IN THE CORRESPONDING UCB.
                                0000    111 ;
                                0000    112 ;       R2, R3, R4, AND R5 ARE PRESERVED ACROSS CALL.
                                0000    113 ;-
                                0000    114
                           00000000     115          .PSECT  SEP, SHR, EXE, WRT, LONG        ; **** SUPERVISOR ****
                                0000    116 IOC$CANCELIO::                              ;CANCEL I/O ON CHANNEL
          11 58 A5    08    E1  0000    117          BBC     #UCB$V_BSY,UCB$W_STS(R5),10$ ;IF CLR, DEVICE NOT BUSY
          60 A4    0C A3    D1  0005    118          CMPL    IRP$L_PID(R3),PCB$L_PID(R4) ;PROCESS ID MATCH?
                      0A    12  000A    119          BNEQ    10$                         ;IF NEQ NO
             28 A3    52    B1  000C    120          CMPW    R2,IRP$W_CHAN(R3)           ;CHANNEL NUMBER MATCH
                      04    12  0010    121          BNEQ    10$                         ;IF NEQ NO
             58 A5    08    A8  0012    122          BISW    #UCB$M_CANCEL,UCB$W_STS(R5) ;SET CANCEL PENDING
                            05  0016    123 10$:     RSB                                 ;
```

```
                                0017   125          .SBTTL  FILL DIAGNOSTIC BUFFER
                                0017   126 ;+
                                0017   127 ; IOC$DIAGBUFILL - FILL DIAGNOSTIC BUFFER
                                0017   128 ;
                                0017   129 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION, BUT BEFORE RELEASING
                                0017   130 ; THE I/O CHANNEL, TO FILL THE FINAL DEVICE PARAMETERS INTO AN INTERNAL DIAG-
                                0017   131 ; NOSTIC BUFFER IF ONE IS SPECIFIED.
                                0017   132 ;
                                0017   133 ; INPUTS:
                                0017   134 ;
                                0017   135 ;       R4 = ADDRESS OF DEVICE CSR REGISTER.
                                0017   136 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0017   137 ;
                                0017   138 ; OUTPUTS:
                                0017   139 ;
                                0017   140 ;       IF A DIAGNOSTIC BUFFER WAS SPECIFIED IN THE ORIGINAL REQUEST, THEN
                                0017   141 ;       THE COMPLETION TIME, FINAL ERROR COUNTERS, AND DEVICE REGISTERS ARE
                                0017   142 ;       FILLED INTO THE DIAGNOSTIC BUFFER.
                                0017   143 ;-
                                0017   144
                                0017   145 IOC$DIAGBUFILL::                           ;FILL DIAGNOSTIC BUFFER
          53    4C A5    D0   0017   146          MOVL    UCB$L_IRP(R5),R3           ;GET ADDRESS OF I/O PACKET
       27 2A A3    07    E1   001B   147          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),10$ ;IF CLR, NO DIAGNOSTIC BUFFER
          50    44 B3    D0   0020   148          MOVL    @IRP$L_DIAGBUF(R3),R0      ;GET ADDRESS OF INTERNAL BUFFER DATA AREA
             50    08    C0   0024   149          ADDL    #8,R0                      ;POINT PAST START TIME
    80 00000000'EF    7D   0027   150          MOVQ    EXE$GQ_SYSTIME,(R0)+       ;INSERT COMPLETION TIME
       80    70 A5    3C   002E   151          MOVZWL  UCB$B_ERTCNT(R5),(R0)+     ;INSERT FINAL ERROR COUNTERS
       52    24 A5    D0   0032   152          MOVL    UCB$L_DDB(R5),R2           ;GET ADDRESS OF DDB
       52    0C A2    D0   0036   153          MOVL    DDB$L_DDT(R2),R2           ;GET ADDRESS OF DDT
       51    10 A2    D0   003A   154          MOVL    DDT$L_REGDUMP(R2),R1       ;FETCH ADDRESS OF DUMP ROUTINE
       03 51    10    E0   003E   155          BBS     #16,R1,5$                  ;+++ BRANCH IF SUPERVISOR ABSOLUTE
          51    52    C0   0042   156          ADDL    R2,R1                      ;+++ INCLUDE BASE ADDRESS
                61    16   0045   157 5$:      JSB     (R1)                       ;CALL DEVICE SPECIFIC REGISTER DUMP ROUTINE
                      05   0047   158 10$:     RSB                                ;
```

```
                     0048   160          .SBTTL  RELEASE I/O CHANNEL
                     0048   161 ;+
                     0048   162 ; IOC$RELCHAN - RELEASE ALL I/O CHANNELS
                     0048   163 ; IOC$RELSCHAN - RELEASE SECONDARY I/O CHANNEL
                     0048   164 ;
                     0048   165 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION TO RELEASE ALL
                     0048   166 ; CHANNELS THE I/O WAS BEING PERFORMED ON.
                     0048   167 ;
                     0048   168 ; INPUTS:
                     0048   169 ;
                     0048   170 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     0048   171 ;
                     0048   172 ; OUTPUTS:
                     0048   173 ;
                     0048   174 ;       THE CHANNELS ARE RELEASED AND AN ATTEMPT IS MADE TO REMOVE THE NEXT
                     0048   175 ;       WAITING DRIVER PROCESS FROM EACH CHANNEL QUEUE. IF A DRIVER PROCESS
                     0048   176 ;       IS WAITING, THEN THE CHANNEL IS ASSIGNED TO THAT DRIVER PROCESS AND
                     0048   177 ;       IT IS CALLED VIA A JSB TO ITS CHANNEL WAIT RETURN ADDRESS. WHEN THE
                     0048   178 ;       CALLED DRIVER PROCESS RETURNS, A RETURN IS MADE TO THE DRIVER PROCESS
                     0048   179 ;       THAT RELEASED THE CHANNEL. IF THERE IS NO DRIVER PROCESS WAITING FOR
                     0048   180 ;       THE CHANNEL, THEN THE CHANNEL STATUS IS SET TO IDLE.
                     0048   181 ;
                     0048   182 ;       R3 AND R4 ARE PRESERVED ACROSS CALL.
                     0048   183 ;-
                     0048   184
                     0048   185          .ENABL  LSB
                     0048   186 IOC$RELSCHAN::                          ;RELEASE SECONDARY I/O CHANNEL
     50   20 A5   D0  0048   187          MOVL    UCB$L_CRB(R5),R0        ;GET ADDRESS OF PRIMARY CRB
     50   10 A0   D0  004C   188          MOVL    CRB$L_LINK(R0),R0       ;GET ADDRESS OF SECONARY CRB
          10   11     0050   189          BRB     20$                     ;
                      0052   190 IOC$RELCHAN::                           ;RELEASE I/O CHANNEL
     50   20 A5   D0  0052   191          MOVL    UCB$L_CRB(R5),R0        ;GET ADDRESS OF PRIMARY CRB
     50   10 A0   D0  0056   192          MOVL    CRB$L_LINK(R0),R0       ;GET ADDRESS OF SECONDARY CRB
          02   13     005A   193          BEQL    10$                     ;IF EQL NONE
          04   10     005C   194          BSBB    20$                     ;RELEASE SECONDARY CHANNEL
     50   20 A5   D0  005E   195 10$:     MOVL    UCB$L_CRB(R5),R0        ;GET ADDRESS OF PRIMARY CRB
25 0E A0    00   E1  0062   196 20$:     BBC     #CRB$V_BSY,CRB$B_MASK(R0),30$ ;IF CLR, THEN CHANNEL NOT BUSY
     51   1C A0   D0  0067   197          MOVL    CRB$L_INTD+VEC$L_IDB(R0),R1 ;GET ADDRESS OF IDB
     04 A1    55   D1  006B   198          CMPL    R5,IDB$L_OWNER(R1)      ;DRIVER PROCESS OWN CHANNEL?
          1B   12     006F   199          BNEQ    30$                     ;IF NEQ NO
     52   00 B0   0F  0071   200          REMQUE  @CRB$L_WQFL(R0),R2      ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
          16   1D     0075   201          BVS     40$                     ;IF VS NO DRIVER PROCESS WAITING
          38   BB     0077   202          PUSHR   #^M<R3,R4,R5>           ;SAVE CONTEXT OF CURRENT DRIVER PROCESS
     55   52   D0     0079   203          MOVL    R2,R5                   ;COPY ADDRESS OF DRIVER PROCESS FORK BLOCK
     53   10 A5   D0  007C   204          MOVL    UCB$L_FR3(R5),R3        ;LOAD WAITING DRIVER PROCESS CONTEXT
     54   61   D0     0080   205          MOVL    IDB$L_CSR(R1),R4        ;SET ASSIGNED CHANNEL CSR ADDRESS
     04 A1    55   D0  0083   206          MOVL    R5,IDB$L_OWNER(R1)      ;SET ADDRESS OF OWNER PROCESS UCB
          0C B5   16     0087   207          JSB     @UCB$L_FPC(R5)          ;CALL DRIVER AT CHANNEL WAIT RETURN ADDRESS
          38   BA     008A   208          POPR    #^M<R3,R4,R5>           ;RESTORE PREVIOUS DRIVER PROCESS CONTEXT
          05   008C   209 30$:     RSB                                     ;
          04 A1    D4  008D   210 40$:     CLRL    IDB$L_OWNER(R1)         ;CLEAR OWNER UNIT UCB ADDRESS
     0E A0    01   8A  0090   211          BICB    #CRB$M_BSY,CRB$B_MASK(R0) ;CLEAR CHANNEL BUSY
          05   0094   212          RSB                                     ;
                      0095   213          .DSABL  LSB
```

```
                     0095   215          .SBTTL  REQUEST I/O CHANNEL
                     0095   216 ;+
                     0095   217 ; IOC$REQPCHANH - REQUEST PRIMARY I/O CHANNEL HIGH PRIORITY
                     0095   218 ; IOC$REQSCHANH - REQUEST SECONDARY I/O CHANNEL HIGH PRIORITY
                     0095   219 ; IOC$REQPCHANL - REQUEST PRIMARY I/O CHANNEL LOW PRIORITY
                     0095   220 ; IOC$REQSCHANL - REQUEST SECONDARY I/O CHANNEL LOW PRIORITY
                     0095   221 ;
                     0095   222 ; THESE ROUTINES ARE CALLED TO REQUEST AN I/O CHANNEL TO PERFORM AN I/O
                     0095   223 ; OPERATION ON.
                     0095   224 ;
                     0095   225 ; INPUTS:
                     0095   226 ;
                     0095   227 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     0095   228 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                     0095   229 ;
                     0095   230 ; OUTPUTS:
                     0095   231 ;
                     0095   232 ;       IF THE SPECIFIED I/O CHANNEL IS IDLE, THEN IT IS IMMEDIATELY
                     0095   233 ;       ASSIGNED TO THE CURRENT DRIVER PROCESS. ELSE THE DRIVER PROCESS
                     0095   234 ;       CONTEXT IS SAVED IN ITS FORK BLOCK, THE FORK BLOCK IS INSERTED
                     0095   235 ;       IN THE CHANNEL WAIT QUEUE, AND A RETURN TO THE DRIVER PROCESS'
                     0095   236 ;       CALLER IS EXECUTED.
                     0095   237 ;
                     0095   238 ;       WHEN THE CHANNEL IS ASSIGNED, THE CSR ADDRESS OF THE ASSIGNED
                     0095   239 ;       CONTROLLER IS RETURNED TO THE CALLER IN REGISTER R4.
                     0095   240 ;
                     0095   241 ;       R3 IS PRESERVED ACROSS CALL.
                     0095   242 ;-
                     0095   243
                     0095   244          .ENABL  LSB
                     0095   245 IOC$REQSCHANH::                          ;REQUEST SECONDARY I/O CHANNEL HIGH PRIORITY
 50   20 A5   D0     0095   246          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
 50   10 A0   D0     0099   247          MOVL    CRB$L_LINK(R0),R0        ;GET ADDRESS OF SECONDARY CRB
        0E   11     009D   248          BRB     10$                      ;
                     009F   249 IOC$REQSCHANL::                          ;REQUEST SECONDARY I/O CHANNEL LOW PRIORITY
 50   20 A5   D0     009F   250          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
 50   10 A0   D0     00A3   251          MOVL    CRB$L_LINK(R0),R0        ;GET ADDRESS OF SECONDARY CRB
        0D   11     00A7   252          BRB     20$                      ;
                     00A9   253 IOC$REQPCHANH::                          ;REQUEST PRIMARY I/O CHANNEL HIGH PRIORITY
 50   20 A5   D0     00A9   254          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
    52   50   D0     00AD   255 10$:     MOVL    R0,R2                    ;SET ADDDRESS OF WAIT QUEUE LISTHEAD
        08   11     00B0   256          BRB     30$                      ;
                     00B2   257 IOC$REQPCHANL::                          ;REQUEST PRIMARY I/O CHANNEL LOW PRIORITY
 50   20 A5   D0     00B2   258          MOVL    UCB$L_CRB(R5),R0         ;GET ADDRESS OF PRIMARY CRB
 52   04 A0   D0     00B6   259 20$:     MOVL    CRB$L_WQBL(R0),R2        ;GET ADDRESS OF LAST ENTRY IN QUEUE
 51   1C A0   D0     00BA   260 30$:     MOVL    CRB$L_INTD+VEC$L_IDB(R0),R1 ;GET ADDRESS OF IDB
08 0E A0     00   E2 00BE   261          BBSS    #CRB$V_BSY,CRB$B_MASK(R0),40$ ;IF SET, THEN CHANNEL BUSY
    54   61   D0     00C3   262          MOVL    IDB$L_CSR(R1),R4         ;SET ASSIGNED CHANNEL CSR ADDRESS
 04 A1   55   D0     00C6   263          MOVL    R5,IDB$L_OWNER(R1)       ;SET OWNER UCB ADDRESS
        05   00CA   264          RSB                              ;
 10 A5   53   D0     00CB   265 40$:     MOVL    R3,UCB$L_FR3(R5)         ;SAVE R3 IN FORK BLOCK
      0C A5 8ED0     00CF   266          POPL    UCB$L_FPC(R5)            ;SAVE CHANNEL WAIT RETURN ADDRESS
    62   65   0E   00D3   267          INSQUE  UCB$L_FQFL(R5),CRB$L_WQFL(R2) ;INSERT DRIVER PROCESS IN CHANNEL WAIT
 04 A1   55   D1     00D6   268          CMPL    R5,IDB$L_OWNER(R1)       ;CURRENT DRIVER PROCESS OWNER?
        41   13     00DA   269          BEQL    RELEASE                  ;IF EQL YES - RELEASE CHANNELS
             05     00DC   270          RSB                              ;
                     00DD   271          .DSABL  LSB
```

```
                                00DD   273          .SBTTL  I/O REQUEST COMPLETION PROCESSING
                                00DD   274 ;+
                                00DD   275 ; IOC$REQCOM - I/O REQUEST COMPLETE
                                00DD   276 ;
                                00DD   277 ; THIS ROUTINE IS ENTERED WHEN AN I/O OPERATION IS COMPLETED ON A
                                00DD   278 ; DEVICE UNIT. THE FINAL I/O STATUS IS STORED IN THE ASSOCIATED I/O
                                00DD   279 ; PACKET AND THE PACKET IS INSERTED IN THE I/O FINISH QUEUE FOR
                                00DD   280 ; I/O POST PROCESSING. DEVICE UNIT BUSY IS CLEARED AND AN ATTEMPT
                                00DD   281 ; IS MADE TO START ANOTHER I/O REQUEST ON THE DEVICE UNIT.
                                00DD   282 ;
                                00DD   283 ; INPUTS:
                                00DD   284 ;
                                00DD   285 ;       R0 = FIRST LONGWORD OF I/O STATUS.
                                00DD   286 ;       R1 = SECOND LONGWORD OF I/O STATUS.
                                00DD   287 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                00DD   288 ;
                                00DD   289 ; OUTPUTS:
                                00DD   290 ;
                                00DD   291 ;       THE I/O PACKET IS INSERTED IN THE I/O POST PROCESSING QUEUE
                                00DD   292 ;       AND DEVICE UNIT BUSY IS CLEARED. A SOFTWARE INTERRUPT IS
                                00DD   293 ;       REQUESTED TO INITIATE I/O POST PROCESSING.
                                00DD   294 ;-
                                00DD   295
                                00DD   296 IOC$REQCOM::                                 ;I/O DONE PROCESSING
          53    4C A5    D0     00DD   297          MOVL    UCB$L_IRP(R5),R3           ;GET ADDRESS OF I/O PACKET
          34 A3    50    7D     00E1   298          MOVQ    R0,IRP$L_MEDIA(R3)         ;STORE FINAL I/O STATUS
                60 A5    D6     00E5   299          INCL    UCB$L_OPCNT(R5)            ;INCREMENT OPERATIONS COMPLETED
                                00E8   300
                                00E8   301
00000000'FF       63    0E     00E8   302          INSQUE  (R3),@L^IOC$GL_PSBL        ;INSERT PACKET IN POST PROCESS QUEUE
                  03    12     00EF   303          BNEQ    10$                        ;IF NEQ NOT FIRST ENTRY IN QUEUE
                                00F1   304          SOFTINT #IPL$_IOPOST               ;INITIATE SOFTWARE INTERRUPT
                14    04     DA     00F1                   MTPR    #IPL$_IOPOST,S^#PR$_SIRR
          18 58 A5    02     E5     00F4   305 10$:       BBCC    #UCB$V_ERLOGIP,UCB$W_STS(R5),20$ ;IF CLR, ERROR LOG NOT IN PROGRESS
             52    78 A5     D0     00F9   306          MOVL    UCB$L_EMB(R5),R2           ;GET ADDRESS OF ERROR MESSAGE BUFFER
          1A A2    58 A5     B0     00FD   307          MOVW    UCB$W_STS(R5),EMB$W_DV_STS(R2) ;INSERT FINAL DEVICE STATUS
          10 A2    70 A5     B0     0102   308          MOVW    UCB$B_ERTCNT(R5),EMB$B_DV_ERTCNT(R2) ;INSERT FINAL ERROR COUNTERS
             12 A2    50     7D     0107   309          MOVQ    R0,EMB$Q_DV_IOSB(R2)       ;INSERT FINAL I/O STATUS
               00000000'EF    16     010B   310          JSB     ERL$RELEASEMB              ;RELEASE ERROR MESSAGE BUFFER   [04]  ;G:2
             53    40 B5     0F     0111   311 20$:       REMQUE  @UCB$L_IOQFL(R5),R3        ;REMOVE I/O PACKET FROM DEVICE UNIT QUEUE
                   09    1C     0115   312          BVC     IOC$INITIATE               ;IF VC INITIATE NEXT FUNCTION
       58 A5    0100    8F     AA     0117   313          BICW    #UCB$M_BSY,UCB$W_STS(R5)   ;CLEAR UNIT BUSY
                                011D   314 RELEASE:                                    ;RELEASE ALL CHANNELS
                   FF32    31     011D   315          BRW     IOC$RELCHAN                ;
```

```
                                  0120          .SBTTL  INITIATE I/O FUNCTION ON DEVICE
                                  0120  ;+
                                  0120  ; IOC$INITIATE - INITIATE NEXT FUNCTION ON DEVICE
                                  0120  ;
                                  0120  ; THIS ROUTINE IS CALLED TO INITIATE THE NEXT FUNCTION ON A DEVICE BY CLEARING
                                  0120  ; STATUS BITS, SETTING THE OPERATION START TIME IF A DIAGNOSTIC BUFFER IS
                                  0120  ; SPECIFIED, AND CALLING THE DRIVER AT ITS START I/O ENTRY POINT.
                                  0120  ;
                                  0120  ; INPUTS:
                                  0120  ;
                                  0120  ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                  0120  ;       R5 = DEVICE UNIT UCB ADDRESS.
                                  0120  ;
                                  0120  ; OUTPUTS:
                                  0120  ;
                                  0120  ;       CANCEL I/O, POWERFAIL, AND TIME OUT STATUS BITS ARE CLEARED, THE
                                  0120  ;       CURRENT SYSTEM TIME IS FILLED INTO THE INTERNAL DIAGNOSTIC BUFFER
                                  0120  ;       IF ONE IS SPECIFIED, AND THE DRIVER IS CALLED AT ITS START I/O ENTRY
                                  0120  ;       POINT.
                                  0120  ;-
                                  0120
                                  0120  IOC$INITIATE::                          ;INITIATE I/O FUNCTION
          4C A5    53    D0       0120          MOVL    R3,UCB$L_IRP(R5)        ;SAVE I/O PACKET ADDRESS
                                  0124
                                  0124
       68 A5    2C A3    7D       0124          MOVQ    IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5) ;COPY TRANSFER PARAMETERS
    58 A5    0048 8F    AA       0129          BICW    #UCB$M_CANCEL!UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR CANCEL AND TIME OUT
       0B 2A A3    07    E1       012F          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),10$ ;IF CLR, NO DIAGNOSTIC BUFFER
          50    44 B3    D0       0134          MOVL    @IRP$L_DIAGBUF(R3),R0   ;GET ADDRESS OF DIAGNOSTIC BUFFER DATA AREA
   60    00000000'EF    7D       0138          MOVQ    EXE$GQ_SYSTIME,(R0)     ;INSERT I/O OPERATION START TIME
          50    24 A5    D0       013F  10$:    MOVL    UCB$L_DDB(R5),R0        ;GET ADDRESS OF DEVICE DATA BLOCK
          50    0C A0    D0       0143          MOVL    DDB$L_DDT(R0),R0        ;GET ADDRESS OF DRIVER DISPATCH TABLE
             51    60    D0       0147          MOVL    DDT$L_START(R0),R1      ;FETCH ADDRESS OF START ENTRY POINT
          03 51    10    E0       014A          BBS     #16,R1,20$              ;+++ BRANCH IF SUPERVISOR ABSOLUTE
             51    50    C0       014E          ADDL    R0,R1                   ;+++ CHANGE OFFSET TO ABSOLUTE
                   61    17       0151  20$:    JMP     (R1)                    ;START I/O OPERATION
```

```
                                0153   354          .SBTTL  RELEASE BUFFERED DATAPATH
                                0153   355 ;+
                                0153   356 ; IOC$RELDATAP - RELEASE BUFFERED DATAPATH
                                0153   357 ;
                                0153   358 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION TO RELEASE A UNIBUS
                                0153   359 ; ADAPTER BUFFERED DATAPATH.
                                0153   360 ;
                                0153   361 ; INPUTS:
                                0153   362 ;
                                0153   363 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0153   364 ;
                                0153   365 ; OUTPUTS:
                                0153   366 ;
                                0153   367 ;       IF THE BUFFERED DATAPATH IS PERMANENTLY ASSIGNED TO THE ASSOCIATED
                                0153   368 ;       I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                                0153   369 ;       ELSE THE BUFFERED DATAPATH IS RELEASED AND AN ATTEMPT IS MADE TO
                                0153   370 ;       REMOVE THE NEXT WAITING DRIVER PROCESS FROM THE DATAPATH WAIT QUEUE.
                                0153   371 ;       IF A DRIVER PROCESS IS WAITING, THEN THE BUFFERED DATAPATH IS
                                0153   372 ;       ASSIGNED TO THE ASSOCIATED I/O CHANNEL AND THE DRIVER IS CALLED
                                0153   373 ;       VIA A JSB TO ITS BUFFERED DATAPATH WAIT RETURN ADDRESS.
                                0153   374 ;-
                                0153   375
                                0153   376 IOC$RELDATAP::                              ;RELEASE BUFFERED DATAPATH
          50    20 A5    D0     0153   377          MOVL    UCB$L_CRB(R5),R0          ;GET ADDRESS OF CRB
          51    28 A0    D0     0157   378          MOVL    CRB$L_INTD+VEC$L_ADP(R0),R1 ;GET ADDRESS OF ADP
          52    27 A0    98     015B   379          CVTBL   CRB$L_INTD+VEC$B_DATAPATH(R0),R2 ;GET DATAPATH DESIGNATOR
                   29    19     015F   380          BLSS    10$                       ;IF LSS PERMANENT ASSIGNMENT
                00 00    F0     0161   381          INSV    #0,#VEC$V_DATAPATH,-      ;ZERO DATAPATH NUMBER
                   05           0164   382                  #VEC$S_DATAPATH,-         ;
                   27 A0        0165   383                  CRB$L_INTD+VEC$B_DATAPATH(R0)
                00 EF           0167   384          EXTZV   #VEC$V_DATAPATH,-         ;EXTRACT DATAPATH NUMBER
             52 52 05           0169   385                  #VEC$S_DATAPATH,R2,R2     ;
          50    14 B1    0F     016C   386          REMQUE  @ADP$L_DPQFL(R1),R0       ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
                   19    1D     0170   387          BVS     20$                       ;IF VS NO DRIVER PROCESS WAITING
                   38    BB     0172   388          PUSHR   #^M<R3,R4,R5>             ;SAVE CONTEXT OF CURRENT DRIVER PROCESS
          55    50    D0     0174   389          MOVL    R0,R5                     ;COPY ADDRESS OF DRIVER PROCESS FORK BLOCK
          51    20 A5    D0     0177   390          MOVL    UCB$L_CRB(R5),R1          ;GET ADDRESS OF CRB
                00 52    F0     017B   391          INSV    R2,#VEC$V_DATAPATH,-      ;STORE ASSIGNED
                   05           017E   392                  #VEC$S_DATAPATH,-         ; DATAPATH NUMBER
                   27 A1        017F   393                  CRB$L_INTD+VEC$B_DATAPATH(R1)
          53    10 A5    7D     0181   394          MOVQ    UCB$L_FR3(R5),R3          ;RESTORE DRIVER PROCESS CONTEXT
                0C B5    16     0185   395          JSB     @UCB$L_FPC(R5)            ;CALL DRIVER AT DATAPATH WAIT RETURN ADDRESS
                   38    BA     0188   396          POPR    #^M<R3,R4,R5>             ;RESTORE PREVIOUS DRIVER PROCESS CONTEXT
                   05           018A   397 10$:     RSB                               ;
       FA 24 A1    52    E3     018B   398 20$:     BBCS    R2,ADP$W_DPBITMAP(R1),10$ ;SET DATAPATH BIT AND EXIT
                                0190   399          BUG_CHECK INCONSTATE              ;INCONSISTENT STATE
          00000000'9F    16     0190                JSB     @#BUG$CHECK
2C 45 54 41 54 53 4E 4F 43 4E 49 00' 0196          .ASCIC  "INCONSTATE,"
                   0B           0196
                   05           01A2   400          RSB                               ;
```

```
                                01A3    402          .SBTTL  REQUEST BUFFERED DATAPATH
                                01A3    403 ;+
                                01A3    404 ; IOC$REQDATAP - REQUEST BUFFERED DATAPATH (WAIT)
                                01A3    405 ; IOC$REQDATAPNW - REQUEST BUFFERED DATAPATH (NO WAIT)
                                01A3    406 ;
                                01A3    407 ; THIS ROUTINE IS CALLED TO REQUEST A UNIBUS ADAPTER BUFFERED DATAPATH TO
                                01A3    408 ; PERFORM AN I/O OPERATION ON.
                                01A3    409 ;
                                01A3    410 ; INPUTS:
                                01A3    411 ;
                                01A3    412 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                01A3    413 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                01A3    414 ;
                                01A3    415 ;       IT IS ASSUMED THAT THE CALLER OWNS THE I/O CHANNEL ON WHICH THE
                                01A3    416 ;       TRANSFER IS TO OCCUR.
                                01A3    417 ;
                                01A3    418 ; OUTPUTS:
                                01A3    419 ;
                                01A3    420 ;       IF A BUFFERED DATAPATH HAS BEEN PERMANENTLY ASSIGNED TO THE ASSOCIATED
                                01A3    421 ;       I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                                01A3    422 ;       ELSE AN ATTEMPT IS MADE TO FIND A FREE BUFFERED DATAPATH. IF A FREE
                                01A3    423 ;       BUFFERED DATAPATH IS FOUND, THEN ITS NUMBER IS STORED IN THE ASSO-
                                01A3    424 ;       CIATED CHANNEL REQUEST BLOCK AND CONTROL IS RETURNED TO THE CALLER.
                                01A3    425 ;       ELSE IF NO WAIT IS SPECIFIED, THEN A FAILURE INDICATION IS RETURNED
                                01A3    426 ;       TO THE CALLER. ELSE THE DRIVER PROCESS CONTEXT IS SAVED IN ITS FORK
                                01A3    427 ;       BLOCK, THE FORK BLOCK IS INSERTED IN THE BUFFERED DATAPATH WAIT QUEUE,
                                01A3    428 ;       AND A RETURN TO THE DRIVER PROCESS' CALLER IS EXECUTED.
                                01A3    429 ;-
                                01A3    430
                                01A3    431          .ENABL  LSB
                                01A3    432 IOC$REQDATAPNW::                                ;REQUEST BUFFERED DATAPATH NO WAIT
                      00    DD  01A3    433          PUSHL   #0                             ;SET NO WAIT INDICATOR
                      02    11  01A5    434          BRB     5$                             ;
                                01A7    435 IOC$REQDATAP::                                  ;REQUEST BUFFERED DATAPATH WAIT
                      01    DD  01A7    436          PUSHL   #1                             ;SET WAIT INDICATOR
                50    20 A5   D0  01A9    437 5$:      MOVL    UCB$L_CRB(R5),R0               ;GET ADDRESS OF CRB
              28 27 A0    07    E0  01AD    438          BBS     #VEC$V_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R0),20$ ;IF SET, PERMANENT
                51    28 A0   D0  01B2    439          MOVL    CRB$L_INTD+VEC$L_ADP(R0),R1    ;GET ADDRESS OF ADP
        52    24 A1    10   00    EA  01B6    440          FFS     #0,#ADP$C_NUMDATAP,ADP$W_DPBITMAP(R1),R2 ;SEARCH FOR FREE DATAPATH
                      0B    13  01BC    441          BEQL    10$                            ;IF EQL NONE FOUND
                   00    52   F0  01BE    442          INSV    R2,#VEC$V_DATAPATH,-           ;STORE ASSIGNED
                      05        01C1    443                  #VEC$S_DATAPATH,-              ; DATAPATH NUMBER
                   27 A0        01C2    444                  CRB$L_INTD+VEC$B_DATAPATH(R0)
              11 24 A1    52    E4  01C4    445          BBSC    R2,ADP$W_DPBITMAP(R1),20$      ;CLEAR DATAPATH BIT AND EXIT
                   13 8E    E9  01C9    446 10$:     BLBC    (SP)+,30$                      ;IF LBC NO WAIT SPECIFIED
                10 A5    53    7D  01CC    447          MOVQ    R3,UCB$L_FR3(R5)               ;SAVE DRIVER PROCESS CONTEXT
                   0C A5 8ED0   01D0    448          POPL    UCB$L_FPC(R5)                  ;SAVE DATAPATH WAIT RETURN ADDRESS
                18 B1    65    0E  01D4    449          INSQUE  UCB$L_FQFL(R5),@ADP$L_DPQBL(R1) ;INSERT DRIVER PROCESS IN DATAPATH W
                      7E    D5  01D8    450          TSTL    -(SP)                          ;PUSH DUMMY INDICATOR ON STACK
                   50    00'   9A  01DA    451 20$:     MOVZBL  S^#SS$_NORMAL,R0               ;SET SUCCESS INDICATOR
                      8E    D5  01DD    452          TSTL    (SP)+                          ;REMOVE WAIT INDICATOR FROM STACK
                            05  01DF    453 30$:     RSB                                    ;
                                01E0    454          .DSABL  LSB
```

```
                                01E0    456          .SBTTL  RELEASE UNIBUS MAP REGISTERS
                                01E0    457 ;+
                                01E0    458 ; IOC$RELMAPREG - RELEASE UNIBUS MAP REGISTERS
                                01E0    459 ;
                                01E0    460 ; THIS ROUTINE IS CALLED TO RELEASE UNIBUS MAP REGISTERS THAT WERE PREVIOUSLY
                                01E0    461 ; ASSIGNED FOR AN I/O TRANSFER.
                                01E0    462 ;
                                01E0    463 ; INPUTS:
                                01E0    464 ;
                                01E0    465 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                01E0    466 ;
                                01E0    467 ;       IT IS ASSUMED THAT THE CALLER STILL OWNS THE I/O CHANNEL ON WHICH
                                01E0    468 ;       THE TRANSFER TOOK PLACE.
                                01E0    469 ;
                                01E0    470 ; OUTPUTS:
                                01E0    471 ;
                                01E0    472 ;       IF THE MAPPING REGISTERS HAVE BEEN PERMANENTLY ASSIGNED TO THE ASSO-
                                01E0    473 ;       CIATED I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                                01E0    474 ;       ELSE THE MAPPING REGISTERS ARE RELEASED AND AN ATTEMPT IS MADE TO
                                01E0    475 ;       REMOVE THE NEXT DRIVER PROCESS FROM THE MAP REGISTER WAIT QUEUE. IF
                                01E0    476 ;       A DRIVER PROCESS IS WAITING, THEN IT IS REMOVED FROM THE MAP REGISTER
                                01E0    477 ;       WAIT QUEUE AND AN ATTEMPT IS MADE TO ASSIGN THE REQUESTED NUMBER OF
                                01E0    478 ;       CONTIGUOUS MAP REGISTERS TO THE TRANSFER. IF THE ALLOCATION IS SUCCESS-
                                01E0    479 ;       FUL, THEN THE DRIVER IS CALLED VIA A JSB TO ITS MAP REGISTER WAIT
                                01E0    480 ;       RETURN ADDRESS. WHEN THE DRIVER RETURNS ANOTHER ATTEMPT WILL BE MADE
                                01E0    481 ;       TO ALLOCATE MAP REGISTERS TO THE NEXT DRIVER PROCESS IN THE MAP REGISTER
                                01E0    482 ;       WAIT QUEUE. THE PROCESS OF ATTEMPTING TO ALLOCATE REGISTERS TO WAIT-
                                01E0    483 ;       ING DRIVER PROCESSES IS CONTINUED UNTIL EITHER THERE ARE NO DRIVER
                                01E0    484 ;       PROCESSES WAITING OR AN ALLOCATION FAILURE OCCURS. IN THE CASE OF
                                01E0    485 ;       AN ALLOCATION FAILURE THE DRIVER PROCESS IS REINSERTED AT THE FRONT
                                01E0    486 ;       OF THE MAP REGISTER WAIT QUEUE.
                                01E0    487 ;-
                                01E0    488
                                01E0    489 IOC$RELMAPREG::                                     ;RELEASE UNIBUS MAP REGISTERS
   51    20 A5    D0   01E0    490          MOVL    UCB$L_CRB(R5),R1                        ;GET ADDRESS OF CRB
30 24 A1    0F    E0   01E4    491          BBS     #VEC$V_MAPLOCK,CRB$L_INTD+VEC$W_MAPREG(R1),40$ ;IF SET, PERMANENT
   0078 8F      BB   01E9    492          PUSHR   #^M<R3,R4,R5,R6>                        ;SAVE REGISTERS
   52    28 A1    D0   01ED    493          MOVL    CRB$L_INTD+VEC$L_ADP(R1),R2 ;GET ADDRESS OF ADP
   56    52       D0   01F1    494          MOVL    R2,R6                                   ;SAVE ADDRESS OF ADP
   54    24 A1    3C   01F4    495          MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4 ;GET STARTING MAP REGISTER NUMBER
   50    00       D2   01F8    496          MCOML   #0,R0                                   ;SET ALTER PATTERN
         2F       10   01FB    497          BSBB    IOC$ALTUBAMAP                           ;ALTER MAP REGISTER BITMAP
   55    1C B6    0F   01FD    498 10$:     REMQUE  @ADP$L_MRQFL(R6),R5                     ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
         12       1D   0201    499          BVS     30$                                     ;IF VS NO DRIVER PROCESS WAITING
         49       10   0203    500          BSBB    IOC$ALOUBAMAP                           ;SEARCH MAP REGISTER BITMAP AND ALLOCATE
      09 50       E9   0205    501          BLBC    R0,20$                                  ;IF LBC ALLOCATION FAILURE
   53    10 A5    7D   0208    502          MOVQ    UCB$L_FR3(R5),R3                        ;RESTORE DRIVE PROCESS CONTEXT
      0C B5       16   020C    503          JSB     @UCB$L_FPC(R5)                          ;CALL DRIVER AT MAP REGISTER WAIT RETURN ADD
         EC       11   020F    504          BRB     10$                                     ;
   1C A6    65     0E   0211    505 20$:     INSQUE  UCB$L_FQFL(R5),ADP$L_MRQFL(R6) ;REINSERT DRIVER PROCESS AT FRONT OF
      0078 8F      BA   0215    506 30$:     POPR    #^M<R3,R4,R5,R6>                        ;RESTORE REGISTERS
                   05   0219    507 40$:     RSB                                             ;
```

```
                     021A   509          .SBTTL  REQUEST UNIBUS MAP REGISTERS
                     021A   510 ;+
                     021A   511 ; IOC$REQMAPREG - REQUEST UNIBUS MAP REGISTERS
                     021A   512 ;
                     021A   513 ; THIS ROUTINE IS CALLED TO REQUEST UNIBUS MAP REGISTERS TO PERFORM AN
                     021A   514 ; I/O TRANSFER.
                     021A   515 ;
                     021A   516 ; INPUTS:
                     021A   517 ;
                     021A   518 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     021A   519 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                     021A   520 ;
                     021A   521 ;       IT IS ASSUMED THAT THE CALLER OWNS THE I/O CHANNEL ON WHICH THE
                     021A   522 ;       TRANSFER IS TO OCCUR ON.
                     021A   523 ;
                     021A   524 ; OUTPUTS:
                     021A   525 ;
                     021A   526 ;       IF MAP REGISTERS HAVE BEEN PERMANENTLY ASSIGNED TO THE ASSOCIATED
                     021A   527 ;       I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
                     021A   528 ;       ELSE AN ATTEMPT IS MADE TO ALLOCATE THE REQUESTED NUMBER OF MAP REG-
                     021A   529 ;       ISTERS. IF SUFFICIENT CONTIGUOUS MAP REGISTERS ARE FOUND, THEN THEY
                     021A   530 ;       ARE ASSIGNED TO THE ASSOCIATED I/O CHANNEL AND CONTROL IS RETURNED
                     021A   531 ;       TO THE CALLER. ELSE THE DRIVER PROCESS CONTEXT IS SAVED IN ITS FORK
                     021A   532 ;       BLOCK, THE FORK BLOCK IS INSERTED IN THE MAP REGISTER WAIT QUEUE,
                     021A   533 ;       AND A RETURN TO THE DRIVER PROCESS' CALLER IS EXECUTED.
                     021A   534 ;-
                     021A   535
                     021A   536 IOC$REQMAPREG::                                ;REQUEST UNIBUS MAP REGISTERS
           32   10  021A   537          BSBB    IOC$ALOUBAMAP                 ; ALLOCATE UBA MAP REGISTER
        0C 50   E8  021C   538          BLBS    R0,10$                        ;IF LBS SUCCESSFUL ALLOCATION
10 A5      53   7D  021F   539          MOVQ    R3,UCB$L_FR3(R5)              ;SAVE DRIVER PROCESS CONTEXT
        0C A5 8ED0  0223   540          POPL    UCB$L_FPC(R5)                 ;SAVE MAP REGISTER WAIT RETURN ADDRESS
20 B2      65   0E  0227   541          INSQUE  UCB$L_FQFL(R5),@ADP$L_MRQBL(R2) ;INSERT PROCESS IN MAP REGISTER WAIT
                05  022B   542 10$:     RSB                                   ;
```

```
                                022C   544          .SBTTL  ALTER UBA MAP REGISTER BITMAP
                                022C   545 ;+
                                022C   546 ; IOC$ALTUBAMAP - ALTER UBA MAP REGISTER BIT MAP
                                022C   547 ;
                                022C   548 ; THIS ROUTINE IS CALLED TO EITHER CLEAR OF SET A FIELD OF BITS IN THE UBA MAP
                                022C   549 ; REGISTER ALLOCATION BITMAP.
                                022C   550 ;
                                022C   551 ; INPUTS:
                                022C   552 ;
                                022C   553 ;       R0 = ALTERATION BIT MASK.
                                022C   554 ;       R1 = ADDRESS OF CRB.
                                022C   555 ;       R2 = ADDRESS OF ADP.
                                022C   556 ;       R4 = STARTING MAP REGISTER NUMBER.
                                022C   557 ;
                                022C   558 ; OUTPUTS:
                                022C   559 ;
                                022C   560 ;       THE SPECIFIED BIT FIELD IN THE UBA MAP ALLOCATION BIT MAP IS EITHER SET
                                022C   561 ;       OR CLEARED DEPENDING ON THE STATE OF THE ALTERATION MASK.
                                022C   562 ;
                                022C   563 ;       R3 AND R4 ARE DESTROYED.
                                022C   564 ;-
                                022C   565
                                022C   566 IOC$ALTUBAMAP::                           ;ALTER MAP REGISTER BIT MAP
          53    26 A1    9A     022C   567          MOVZBL  CRB$L_INTD+VEC$B_NUMREG(R1),R3 ;GET NUMBER OF BITS TO ALTER
             53    20    D1     0230   568 10$:     CMPL    #32,R3                  ;MORE THAN LONGWORD LEFT?
                   0E    18     0233   569          BGEQ    20$                     ;IF GEQ NO
    26 A2    20    54    50     0235   570          INSV    R0,R4,#32,ADP$W_MRBITMAP(R2) ;ALTER BITMAP WITH SUPPLIED PATTERN
                54    20    C0     023B   571          ADDL    #32,R4                  ;UPDATE STARTING BIT POSITION
                53    20    C2     023E   572          SUBL    #32,R3                  ;REDUCE NUMBER OF BITS TO ALTER
                      ED    11     0241   573          BRB     10$                     ;
    26 A2    53    54    50     0243   574 20$:     INSV    R0,R4,R3,ADP$W_MRBITMAP(R2) ;ALTER BITMAP WITH SUPPLIED PATTERN
                            05     0249   575          RSB                             ;
```

```
                                024A   577          .SBTTL  SEARCH MAP REGISTER BITMAP AND ALLOCATE MAP REGISTERS
                                024A   578 ;+
                                024A   579 ; IOC$ALOUBAMAP - ALLOCATE UBA MAP REGISTERS (CRB DATABASE SPECIFIED)
                                024A   580 ; IOC$ALOUBAMAPN - ALLOCATE UBA MAP REGISTERS (ARGUMENT SPECIFIED)
                                024A   581 ;
                                024A   582 ; THIS ROUTINE IS CALLED TO ALLOCATE UBA MAP REGISTERS AND TO MARK THE ALLOCATION
                                024A   583 ; IN THE UBA MAP REGISTER ALLOCATION BITMAP.
                                024A   584 ;
                                024A   585 ; INPUTS:
                                024A   586 ;
                                024A   587 ;       R3 = NUMBER OF MAP REGISTERS TO ALLOCATE (ARGUMENT SPECIFIED ENTRY).
                                024A   588 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                024A   589 ;
                                024A   590 ; OUTPUTS:
                                024A   591 ;
                                024A   592 ;       R0 = SUCCESS INDICATION.
                                024A   593 ;-
                                024A   594
                                024A   595          .ENABL  LSB
                                024A   596 IOC$ALOUBAMAPN::                         ;ALLOCATE UBA MAP REGISTERS ARGUMENT SPECIFI
                    38    BB    024A   597          PUSHR   #^M<R3,R4,R5>           ;SAVE REGISTERS
                    15    11    024C   598          BRB     5$                      ;
                                024E   599 IOC$ALOUBAMAP::                          ;ALLOCATE UBA MAP REGISTERS CRB SPECIFIED
                    38    BB    024E   600          PUSHR   #^M<R3,R4,R5>           ;SAVE REGISTERS
              53    6E A5   3C  0250   601          MOVZWL  UCB$W_BCNT(R5),R3       ;GET TRANSFER BYTE COUNT
              54    6C A5   3C  0254   602          MOVZWL  UCB$W_BOFF(R5),R4       ;GET BYTE OFFSET IN PAGE
        53     03FF C344    9E  0258   603          MOVAB   ^X3FF(R3)[R4],R3        ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
     53    53    F7 8F    78  025E   604          ASHL    #-9,R3,R3               ;CALCULATE NUMBER OF MAP REGISTERS REQUIRED
                 50    D4  0263   605 5$:         CLRL    R0                      ;ASSUME ALLOCATION FAILURE
              51    20 A5   D0  0265   606          MOVL    UCB$L_CRB(R5),R1        ;GET ADDRESS OF CRB
              52    28 A1   D0  0269   607          MOVL    CRB$L_INTD+VEC$L_ADP(R1),R2 ;GET ADDRESS OF ADP
           39 24 A1    0F    E0  026D   608          BBS     #VEC$V_MAPLOCK,CRB$L_INTD+VEC$W_MAPREG(R1),40$ ;IF SET, PERMANENT
              26 A1    53    90  0272   609          MOVB    R3,CRB$L_INTD+VEC$B_NUMREG(R1) ;SET NUMBER OF MAP REGISTERS ALLOCATE
                       54    D4  0276   610          CLRL    R4                      ;CLEAR STARTING BIT POSITION
              55    54       53    C1  0278   611 10$:        ADDL3   R3,R4,R5        ;CALCULATE HIGHEST BIT IN REQUIRED SCAN
              01F0 8F       55    B1  027C   612          CMPW    R5,#496         ;BEYOND END OF ALLOCATION BITMAP?
                            2A    14  0281   613          BGTR    50$             ;IF GTR YES
     54    26 A2    20      54    EA  0283   614          FFS     R4,#32,ADP$W_MRBITMAP(R2),R4 ;FIND A SET BIT
                            ED    13  0289   615          BEQL    10$             ;IF EQL BIT NOT FOUND
              55    54       53    C1  028B   616          ADDL3   R3,R4,R5        ;CALCULATE HIGH BIT FOR SUCCESSFUL ALLOCATIO
                 24 A1       54    B0  028F   617          MOVW    R4,CRB$L_INTD+VEC$W_MAPREG(R1) ;SAVE STARTING BIT NUMBER
     54    26 A2    20      54    EB  0293   618 20$:        FFC     R4,#32,ADP$W_MRBITMAP(R2),R4 ;FIND A CLEAR BIT
                       55    54  D1  0299   619          CMPL    R4,R5           ;ENOUGH SET BITS SCANNED OVER?
                            07    18  029C   620          BGEQ    30$             ;IF GEQ YES
           F0 26 A2       54    E0  029E   621          BBS     R4,ADP$W_MRBITMAP(R2),20$ ;IF SET, CONTINUE SCAN
                            D3    11  02A3   622          BRB     10$             ;
              54    24 A1       3C  02A5   623 30$:        MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4 ;RETRIEVE STARTING MAP REGISTER
                            81    10  02A9   624          BSBB    IOC$ALTUBAMAP   ;ALTER MAP REGISTER BITMAP
                            50    D6  02AB   625 40$:        INCL    R0              ;SET SUCCESS INDICATOR
                            38    BA  02AD   626 50$:        POPR    #^M<R3,R4,R5>   ;RESTORE REGISTERS
                            05    02AF   627          RSB                     ;
                                  02B0   628          .DSABL  LSB
```

```
                     02B0    630          .SBTTL  RETURN TO CALLER
                     02B0    631 ;+
                     02B0    632 ; IOC$RETURN - RETURN TO CALLER
                     02B0    633 ;
                     02B0    634 ; THIS ROUTINE IS CALLED AS A RESULT OF A DDT DISPATCH TO A NULL ENTRY. ITS
                     02B0    635 ; FUNCTION IS MERELY TO RETURN TO ITS CALLER.
                     02B0    636 ;
                     02B0    637 ; INPUTS:
                     02B0    638 ;
                     02B0    639 ;       NONE.
                     02B0    640 ;
                     02B0    641 ; OUTPUTS:
                     02B0    642 ;
                     02B0    643 ;       NONE.
                     02B0    644 ;-
                     02B0    645
                     02B0    646 IOC$RETURN::                            ;RETURN TO CALLER
                  05 02B0    647          RSB                            ;
```

```
                                02B1    649          .SBTTL  WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
                                02B1    650 ;+
                                02B1    651 ; IOC$WFIKPCH - WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
                                02B1    652 ;
                                02B1    653 ; THIS ROUTINE IS CALLED TO SOFTWARE ENABLE INTERRUPTS AND TIMEOUT ON
                                02B1    654 ; A DEVICE UNIT AND TO KEEP THE CHANNEL. THIS ROUTINE CAN BE CALLED AT
                                02B1    655 ; EITHER FORK OR DEVICE INTERRUPT LEVEL.
                                02B1    656 ;
                                02B1    657 ; INPUTS:
                                02B1    658 ;
                                02B1    659 ;       00(SP) = RETURN ADDRESS OF CALLER.
                                02B1    660 ;       04(SP) = TIMEOUT VALUE IN SECONDS.
                                02B1    661 ;       08(SP) = IPL TO LOWER TO AFTER SETTING WAIT.
                                02B1    662 ;       12(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                02B1    663 ;
                                02B1    664 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                02B1    665 ;
                                02B1    666 ; OUTPUTS:
                                02B1    667 ;
                                02B1    668 ;       THE TIMEOUT VALUE IS COMPUTED AND STORED IN DUE TIME, REGISTERS R3 AND
                                02B1    669 ;       R4 ALONG WITH THE RETURN PC ARE SAVED IN THE FORK BLOCK, INTERRUPTS AND
                                02B1    670 ;       TIMEOUT ARE ENABLED, AND A RETURN TO THE CALLER'S CALLER IS EXECUTED.
                                02B1    671 ;-
                                02B1    672
                                02B1    673 IOC$WFIKPCH::                               ;WAITFOR INTERRUPT/TIMEOUT AND KEEP CHANNEL
                   6E   02   C0 02B1    674          ADDL    #2,(SP)                    ;CALCULATE OFFSET TO NORMAL RETURN
                10 A5   53   7D 02B4    675          MOVQ    R3,UCB$L_FR3(R5)           ;SAVE REGISTERS R3 AND R4
                   0C A5 8ED0   02B8    676          POPL    UCB$L_FPC(R5)              ;SAVE INTERRUPT RETURN ADDRESS
                58 A5   03   A8 02BC    677          BISW    #UCB$M_INT!UCB$M_TIM,UCB$W_STS(R5) ;ENABLE INTERRUPT AND TIMEOUT
5C A5   00000000'EF   8E   C1 02C0    678          ADDL3   (SP)+,EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ;SET TIMEOUT TIME
             58 A5 0040 8F   AA 02C9    679          BICW    #UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR UNIT TIMED OUT
                               02CF    680          ENBINT                             ;ENABLE INTERRUPTS
                   12   8E   DA 02CF                 MTPR    (SP)+,S^#PR$_IPL
                               05 02D2    681          RSB                             ;
```

```
                        02D3    683          .SBTTL  WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
                        02D3    684 ;+
                        02D3    685 ; IOC$WFIRLCH - WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
                        02D3    686 ;
                        02D3    687 ; THIS ROUTINE IS CALLED TO SOFTWARE ENABLE INTERRUPTS AND TIMEOUT ON A DEVICE
                        02D3    688 ; UNIT AND TO RELEASE THE CHANNEL. THIS ROUTINE CAN ONLY BE CALLED AT FORK LEVEL.
                        02D3    689 ;
                        02D3    690 ; INPUTS:
                        02D3    691 ;
                        02D3    692 ;       00(SP) = RETURN ADDRESS OF CALLER.
                        02D3    693 ;       04(SP) = TIMEOUT VALUE IN SECONDS.
                        02D3    694 ;       08(SP) = IPL TO LOWER TO AFTER SETTING WAIT.
                        02D3    695 ;       12(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                        02D3    696 ;
                        02D3    697 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                        02D3    698 ;
                        02D3    699 ; OUTPUTS:
                        02D3    700 ;
                        02D3    701 ;       THE TIMEOUT VALUE IS COMPUTED AND STORED IN DUE TIME, REGISTERS R3 AND
                        02D3    702 ;       R4 ALONG WITH THE RETURN PC ARE SAVED IN THE FORK BLOCK, INTERRUPTS AND
                        02D3    703 ;       TIMEOUT ARE ENABLED, THE CHANNEL IS RELEASED, AND A RETURN TO THE CALLER'S
                        02D3    704 ;       CALLER IS EXECUTED.
                        02D3    705 ;-
                        02D3    706
                        02D3    707 IOC$WFIRLCH::                               ;WAITFOR INTERRUPT/TIMEOUT AND RELEASE CHANN
            6E  02  C0  02D3    708          ADDL    #2,(SP)                    ;CALCULATE OFFSET TO NORMAL RETURN
          10 A5 53  7D  02D6    709          MOVQ    R3,UCB$L_FR3(R5)           ;SAVE REGISTERS R3 AND R4
             0C A5 8ED0 02DA    710          POPL    UCB$L_FPC(R5)              ;SAVE INTERRUPT RETURN ADDRESS
          58 A5 03  A8  02DE    711          BISW    #UCB$M_INT!UCB$M_TIM,UCB$W_STS(R5) ;ENABLE INTERRUPT AND TIMEOUT
5C A5 00000000'EF  8E  C1  02E2    712          ADDL3   (SP)+,EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ;SET TIMEOUT TIME
      58 A5 0040 8F  AA  02EB    713          BICW    #UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR UNIT TIMED OUT
                        02F1    714          ENBINT                             ;ENABLE INTERRUPTS
            12  8E  DA  02F1                 MTPR    (SP)+,S^#PR$_IPL
               FD5B 31  02F4    715          BRW     IOC$RELCHAN                ;RELEASE ALL CHANNELS AND RETURN TO CALLER
                        02F7    716
                        02F7    717
```

```
                        02F7    719          .SBTTL  ALLOCATE SYSTEM PAGE TABLE
                        02F7    720 ;+
                        02F7    721 ; IOC$ALLOSPT - ALLOCATE SYSTEM PAGE TABLE
                        02F7    722 ;
                        02F7    723 ; THIS ROUTINE ALLOCATES SYSTEM PAGE TABLE (SPT) ENTRIES.
                        02F7    724 ;
                        02F7    725 ; INPUTS:
                        02F7    726 ;
                        02F7    727 ;       R1 = NUMBER OF SPT ENTRIES TO BE ALLOCATED
                        02F7    728 ;
                        02F7    729 ;       BOO$GL_SPTFREL = LOWEST FREE VPN
                        02F7    730 ;       BOO$GL_SPTFREH = HIGHEST FREE VPN
                        02F7    731 ;
                        02F7    732 ;       IT IS ASSUMED THAT THE CALLER IS RUNNING AT IPL$_SYNCH.
                        02F7    733 ;
                        02F7    734 ; OUTPUTS:
                        02F7    735 ;
                        02F7    736 ;       R0 = SUCCESS INDICATION.
                        02F7    737 ;       R2 = STARTING PAGE NUMBER ALLOCATED (SVPN).
                        02F7    738 ;       R3 = ADDRESS OF BASE OF SYSTEM PAGE TABLE (MMG$GL_SPTBASE).
                        02F7    739 ;
                        02F7    740 ;       R1 IS PRESERVED ACROSS CALL.
                        02F7    741 ;-
                        02F7    742 IOC$ALLOSPT::                               ;ALLOCATE SYSTEM PAGE TABLE
     53  00000000'EF    D0  02F7    743          MOVL    L^MMG$GL_SPTBASE,R3     ; Get addr of base of SPT
         50    01       D0  02FE    744          MOVL    #1,R0                   ; Set success
                        0301    745 10$:
                        05  0301    746          RSB                             ;
                            0302    747
                            0302    748          .END
```

```
ADP$B_NUMBER                    0000000B    D
ADP$B_PORT                      00000020    D
ADP$B_TYPE                      0000000A    D
ADP$C_DRADPLEN                  00000014    D
ADP$C_MBAADPLEN                 00000014    D
ADP$C_MPMADPLEN                 00000070    D
ADP$C_NUMDATAP                =  00000010    D
ADP$C_UBAADPLEN                 00000070    D
ADP$K_DRADPLEN                  00000014    D
ADP$K_MBAADPLEN                 00000014    D
ADP$K_MPMADPLEN                 00000070    D
ADP$K_UBAADPLEN                 00000070    D
ADP$L_CRB                       00000010    D
ADP$L_CSR                       00000000    D
ADP$L_DPQBL                     00000018    D
ADP$L_DPQFL                     00000014    D
ADP$L_INTD                      00000064    D
ADP$L_LINK                      00000004    D
ADP$L_MRQBL                     00000020    D
ADP$L_MRQFL                     0000001C    D
ADP$L_PRQQBL                    00000018    D
ADP$L_PRQQFL                    00000014    D
ADP$L_SHB                       0000001C    D
ADP$L_VECTOR                    00000010    D
ADP$W_ADPTYPE                   0000000E    D
ADP$W_DPBITMAP                  00000024    D
ADP$W_MRBITMAP                  00000026    D
ADP$W_SIZE                      00000008    D
ADP$W_TR                        0000000C    D
BUG$CHECK                       ********  X   02
CRB$B_MASK                      0000000E    D
CRB$B_TYPE                      0000000A    D
CRB$C_LENGTH                    00000038    D
CRB$K_LENGTH                    00000038    D
CRB$L_INTD                      00000014    D
CRB$L_INTD2                     00000038    D
CRB$L_LINK                      00000010    D
CRB$L_WQBL                      00000004    D
CRB$L_WQFL                      00000000    D
CRB$M_BSY                     =  00000001    D
CRB$V_BSY                     =  00000000    D
CRB$W_REFC                      0000000C    D
CRB$W_SIZE                      00000008    D
DDB$B_ACPCLASS                  00000013    D
DDB$B_TYPE                      0000000A    D
DDB$C_LENGTH                    00000034    D
DDB$K_LENGTH                    00000034    D
DDB$L_ACPD                      00000010    D
DDB$L_DDT                       0000000C    D
DDB$L_LINK                      00000000    D
DDB$L_UCB                       00000004    D
DDB$T_DRVNAME                   00000024    D
DDB$T_NAME                      00000014    D
DDB$W_SIZE                      00000008    D
DDT$L_ALTSTART                  0000001C    D
DDT$L_CANCEL                    0000000C    D
DDT$L_FDT                       00000008    D
DDT$L_REGDUMP                   00000010    D
DDT$L_START                     00000000    D
DDT$L_UNITINIT                  00000018    D
DDT$L_UNSOLINT                  00000004    D
DDT$W_DIAGBUF                   00000014    D
DDT$W_ERRORBUF                  00000016    D
EMB$B_DV_ERTCNT               =  00000010    D
EMB$Q_DV_IOSB                 =  00000012    D
EMB$W_DV_STS                  =  0000001A    D
ERL$RELEASEMB                   ********    X   02
EXE$GL_ABSTIM                   ********    X   02
EXE$GQ_SYSTIME                  ********    X   02
IDB$B_TYPE                      0000000A    D
IDB$C_LENGTH                    00000034    D
IDB$K_LENGTH                    00000034    D
IDB$L_ADP                       00000010    D
IDB$L_CSR                       00000000    D
IDB$L_OWNER                     00000004    D
IDB$L_UCBLST                    00000014    D
IDB$W_SIZE                      00000008    D
IDB$W_UNITS                     0000000C    D
IOC$ALLOSPT                     000002F7  RG  D   02
IOC$ALOUBAMAP                   0000024E  RG  D   02
IOC$ALOUBAMAPN                  0000024A  RG  D   02
IOC$ALTUBAMAP                   0000022C  RG  D   02
IOC$CANCELIO                    00000000  RG  D   02
IOC$DIAGBUFILL                  00000017  RG  D   02
IOC$GL_PSBL                     ********    X     02
IOC$INITIATE                    00000120  RG  D   02
IOC$RELCHAN                     00000052  RG  D   02
IOC$RELDATAP                    00000153  RG  D   02
IOC$RELMAPREG                   000001E0  RG  D   02
IOC$RELSCHAN                    00000048  RG  D   02
IOC$REQCOM                      000000DD  RG  D   02
IOC$REQDATAP                    000001A7  RG  D   02
IOC$REQDATAPNW                  000001A3  RG  D   02
IOC$REQMAPREG                   0000021A  RG  D   02
IOC$REQPCHANH                   000000A9  RG  D   02
IOC$REQPCHANL                   000000B2  RG  D   02
IOC$REQSCHANH                   00000095  RG  D   02
IOC$REQSCHANL                   0000009F  RG  D   02
IOC$RETURN                      000002B0  RG  D   02
IOC$WFIKPCH                     000002B1  RG  D   02
IOC$WFIRLCH                     000002D3  RG  D   02
IPL$_IOPOST                   =  00000004    D
IRP$B_CARCON                    00000038    D
IRP$B_EFN                       00000022    D
IRP$B_PRI                       00000023    D
IRP$B_RMOD                      0000000B    D
IRP$B_TYPE                      0000000A    D
IRP$C_LENGTH                    0000005C    D
IRP$K_LENGTH                    0000005C    D
IRP$L_ARB                       00000050    D
IRP$L_AST                       00000010    D
IRP$L_ASTPRM                    00000014    D
IRP$L_DIAGBUF                   00000044    D
IRP$L_EXTEND                    0000004C    D
```

IRP$L_IOQBL		00000004	D		PCB$W_ASTCNT		00000038	D		
IRP$L_IOQFL		00000000	D		PCB$W_BIOCNT		0000003A	D		
IRP$L_IOSB		00000024	D		PCB$W_BIOLM		0000003C	D		
IRP$L_IOST1		00000034	D		PCB$W_DIOCNT		0000003E	D		
IRP$L_IOST2		00000038	D		PCB$W_DIOLM		00000040	D		
IRP$L_MEDIA		00000034	D		PCB$W_GPGCNT		00000034	D		
IRP$L_PID		0000000C	D		PCB$W_GRP		0000008A	D		
IRP$L_SEGVBN		00000040	D		PCB$W_MEM		00000088	D		
IRP$L_SEQNUM		00000048	D		PCB$W_MTXCNT		0000000E	D		
IRP$L_SVAPTE		0000002C	D		PCB$W_PPGCNT		00000036	D		
IRP$L_TT_TERM		00000038	D		PCB$W_PRCCNT		00000042	D		
IRP$L_UCB		0000001C	D		PCB$W_SIZE		00000008	D		
IRP$L_WIND		00000018	D		PCB$W_STATE		0000002C	D		
IRP$Q_NT_PRVMSK		0000003C	D		PCB$W_TMBU		00000032	D		
IRP$V_DIAGBUF	=	00000007	D		PR$_IPL	=	00000012	D		
IRP$W_ABCNT		0000003C	D		PR$_SIRR	=	00000014	D		
IRP$W_BCNT		00000032	D		RELEASE		0000011D	R	D	02
IRP$W_BOFF		00000030	D		SIZ...	=	00000001		D	
IRP$W_CHAN		00000028	D		SS$_NORMAL		********		X	02
IRP$W_FUNC		00000020	D		UCB$B_AMOD		00000053	D		
IRP$W_OBCNT		0000003E	D		UCB$B_CEX		00000077	D		
IRP$W_SIZE		00000008	D		UCB$B_CM1		0000004A	D		
IRP$W_STS		0000002A	D		UCB$B_CM2		0000004B	D		
IRP$W_TT_PRMPT		0000003C	D		UCB$B_DEVCLASS		00000038	D		
MMG$GL_SPTBASE		********	X	02	UCB$B_DEVTYPE		00000039	D		
PCB$B_ASTACT		0000000C	D		UCB$B_DIPL		00000052	D		
PCB$B_ASTEN		0000000D	D		UCB$B_DX_SCTCNT		000000A6	D		
PCB$B_PRI		0000000B	D		UCB$B_ERTCNT		00000070	D		
PCB$B_PRIB		0000002F	D		UCB$B_ERTMAX		00000071	D		
PCB$B_TYPE		0000000A	D		UCB$B_FEX		00000076	D		
PCB$B_WEFC		0000002E	D		UCB$B_FIPL		0000000B	D		
PCB$C_LENGTH		0000008C	D		UCB$B_LOCSRV		0000003C	D		
PCB$K_LENGTH		0000008C	D		UCB$B_OFFNDX		00000094	D		
PCB$L_ARB		00000084	D		UCB$B_OFFRTC		00000095	D		
PCB$L_ASTQBL		00000014	D		UCB$B_REMSRV		0000003D	D		
PCB$L_ASTQFL		00000010	D		UCB$B_SECTORS		0000003C	D		
PCB$L_EFC2P		00000058	D		UCB$B_SLAVE		00000074	D		
PCB$L_EFC3P		0000005C	D		UCB$B_SPR		00000075	D		
PCB$L_EFCS		00000050	D		UCB$B_STATE		00000052	D		
PCB$L_EFCU		00000054	D		UCB$B_TRACKS		0000003D	D		
PCB$L_EFWM		0000004C	D		UCB$B_TT_CRFILL		0000009D	D		
PCB$L_JIB		00000078	D		UCB$B_TT_DECRF		000000A1	D		
PCB$L_OWNER		0000001C	D		UCB$B_TT_DELFF		000000A2	D		
PCB$L_PHD		00000064	D		UCB$B_TT_DESPEE		000000A0	D		
PCB$L_PHYPCB		00000018	D		UCB$B_TT_DETYPE		000000A4	D		
PCB$L_PID		00000060	D		UCB$B_TT_LFFILL		0000009E	D		
PCB$L_PQB		0000004C	D		UCB$B_TT_SPEED		0000009C	D		
PCB$L_SQBL		00000004	D		UCB$B_TYPE		0000000A	D		
PCB$L_SQFL		00000000	D		UCB$B_VERTSZ		0000003F	D		
PCB$L_STS		00000024	D		UCB$C_LENGTH		00000074	D		
PCB$L_UIC		00000088	D		UCB$C_MB_LENGTH		00000090	D		
PCB$L_WSSWP		00000020	D		UCB$C_TT_LENGTH		000000BC	D		
PCB$L_WTIME		00000028	D		UCB$K_LENGTH		00000074	D		
PCB$Q_PRIV		0000007C	D		UCB$K_MB_LENGTH		00000090	D		
PCB$T_LNAME		00000068	D		UCB$K_TT_LENGTH		000000BC	D		
PCB$T_TERMINAL		00000044	D		UCB$L_AMB		00000054	D		
PCB$W_APTCNT		00000030	D		UCB$L_ASTQBL		00000010	D		

UCB$L_ASTQFL		0000000C	D	UCB$W_BYTESTOGO		0000003E	D
UCB$L_CPID		0000005C	D	UCB$W_CHARGE		0000004A	D
UCB$L_CRB		00000020	D	UCB$W_CYLINDERS		0000003E	D
UCB$L_DDB		00000024	D	UCB$W_DA		0000008C	D
UCB$L_DEVCHAR		00000034	D	UCB$W_DC		0000008E	D
UCB$L_DEVDEPEND		0000003C	D	UCB$W_DEVBUFSIZ		0000003A	D
UCB$L_DPC		00000080	D	UCB$W_DEVSTS		0000005A	D
UCB$L_DUETIM		0000005C	D	UCB$W_DIRSEQ		00000088	D
UCB$L_DX_BFPNT		0000009C	D	UCB$W_DSTADDR		00000018	D
UCB$L_DX_BUF		00000098	D	UCB$W_DX_BCR		000000A4	D
UCB$L_DX_RXDB		000000A0	D	UCB$W_EC1		00000090	D
UCB$L_EMB		00000078	D	UCB$W_EC2		00000092	D
UCB$L_FIRST		00000014	D	UCB$W_ERRCNT		00000072	D
UCB$L_FPC		0000000C	D	UCB$W_FUNC		0000007E	D
UCB$L_FQBL		00000004	D	UCB$W_MB_SEED		00000000	D
UCB$L_FQFL		00000000	D	UCB$W_MSGCNT		00000016	D
UCB$L_FR3		00000010	D	UCB$W_MSGMAX		00000014	D
UCB$L_FR4		00000014	D	UCB$W_NT_CHAN		0000007C	D
UCB$L_IOQBL		00000044	D	UCB$W_OFFSET		0000008A	D
UCB$L_IOQFL		00000040	D	UCB$W_REFC		00000050	D
UCB$L_IRP		0000004C	D	UCB$W_SIZE		00000008	D
UCB$L_LINK		0000002C	D	UCB$W_SRCADDR		0000001A	D
UCB$L_LOGADR		00000064	D	UCB$W_STS		00000058	D
UCB$L_MAXBLOCK		00000084	D	UCB$W_TT_DESIZE		000000A5	D
UCB$L_MB_MBX		0000007C	D	UCB$W_UNIT		00000048	D
UCB$L_MB_PORT		0000008C	D	UCB$W_VPROT		0000001A	D
UCB$L_MB_RAST		00000078	D	VEC$B_DATAPATH		00000013	D
UCB$L_MB_SMB		00000080	D	VEC$B_NUMREG		00000012	D
UCB$L_MB_WAST		00000074	D	VEC$C_LENGTH		00000024	D
UCB$L_MB_WIOQBL		00000088	D	VEC$K_LENGTH		00000024	D
UCB$L_MB_WIOQFL		00000084	D	VEC$L_ADP		00000014	D
UCB$L_MEDIA		0000008C	D	VEC$L_IDB		00000008	D
UCB$L_NT_DATSSB		00000074	D	VEC$L_INITIAL		0000000C	D
UCB$L_NT_INTSSB		00000078	D	VEC$L_START		0000001C	D
UCB$L_OPCNT		00000060	D	VEC$L_UNITDISC		00000020	D
UCB$L_OWNUIC		0000001C	D	VEC$L_UNITINIT		00000018	D
UCB$L_PID		00000028	D	VEC$Q_DISPATCH		00000000	D
UCB$L_RQBL		00000004	D	VEC$S_DATAPATH	=	00000005	D
UCB$L_RQFL		00000000	D	VEC$V_DATAPATH	=	00000000	D
UCB$L_SVAPTE		00000068	D	VEC$V_MAPLOCK	=	0000000F	D
UCB$L_SVPN		00000064	D	VEC$V_PATHLOCK	=	00000007	D
UCB$L_TT_DECHAR		000000A8	D	VEC$W_MAPREG		00000010	D
UCB$L_TT_RDUE		0000008C	D				
UCB$L_TT_RTIMOU		000000B8	D				
UCB$L_VCB		00000030	D				
UCB$M_BSY	=	00000100	D				
UCB$M_CANCEL	=	00000008	D				
UCB$M_INT	=	00000002	D				
UCB$M_TIM	=	00000001	D				
UCB$M_TIMOUT	=	00000040	D				
UCB$T_PARTNER		0000000C	D				
UCB$V_BSY	=	00000008	D				
UCB$V_ERLOGIP	=	00000002	D				
UCB$W_BCNT		0000006E	D				
UCB$W_BCR		00000096	D				
UCB$W_BOFF		0000006C	D				
UCB$W_BUFQUO		00000018	D				

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000302 (770.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ADP$C_NUMDATAP	=00000010		#-440 (1)
ADP$L_DPQBL	00000018		449 (1)
ADP$L_DPQFL	00000014		386 (1)
ADP$L_MRQBL	00000020		541 (1)
ADP$L_MRQFL	0000001C		498 (1) 505 (1)
ADP$W_DPBITMAP	00000024		398 (1) 440 (1) 445 (1)
ADP$W_MRBITMAP	00000026		570 (1) 574 (1) 614 (1) 618 (1)
			621 (1)
BUG$CHECK	00000000-XR		399 (1)
CRB$B_MASK	0000000E		196 (1) #-211 (1) 261 (1)
CRB$L_INTD	00000014		#-197 (1) #-260 (1) #-378 (1) #-379 (1)
			383 (1) 393 (1) 438 (1) #-439 (1)
			444 (1) 491 (1) #-493 (1) #-495 (1)
			#-567 (1) #-607 (1) 608 (1) #-609 (1)
			#-617 (1) #-623 (1)
CRB$L_LINK	00000010		#-188 (1) #-192 (1) #-247 (1) #-251 (1)
CRB$L_WQBL	00000004		#-259 (1)
CRB$L_WQFL	00000000		200 (1) 267 (1)
CRB$M_BSY	=00000001		#-211 (1)
CRB$V_BSY	=00000000		#-196 (1) #-261 (1)
DDB$L_DDT	0000000C		#-153 (1) #-348 (1)
DDT$L_REGDUMP	00000010		#-154 (1)
DDT$L_START	00000000		#-349 (1)
EMB$B_DV_ERTCNT	=00000010		#-308 (1)
EMB$Q_DV_IOSB	=00000012		#-309 (1)
EMB$W_DV_STS	=0000001A		#-307 (1)
ERL$RELEASEMB	00000000-XR		310 (1)
EXE$GL_ABSTIM	00000000-XR		#-678 (1) #-712 (1)
EXE$GQ_SYSTIME	00000000-XR		#-150 (1) #-346 (1)
IDB$L_CSR	00000000		#-205 (1) #-262 (1)
IDB$L_OWNER	00000004		#-198 (1) #-206 (1) #-210 (1) #-263 (1)
			#-268 (1)
IOC$ALLOSPT	000002F7-R 742 (1)		
IOC$ALOUBAMAP	0000024E-R 599 (1)		#-500 (1) #-537 (1)
IOC$ALOUBAMAPN	0000024A-R 596 (1)		
IOC$ALTUBAMAP	0000022C-R 566 (1)		#-497 (1) #-624 (1)
IOC$CANCELIO	00000000-R 116 (1)		
IOC$DIAGBUFILL	00000017-R 145 (1)		
IOC$GL_PSBL	00000000-XR		302 (1)
IOC$INITIATE	00000120-R 338 (1)		#-312 (1)
IOC$RELCHAN	00000052-R 190 (1)		#-315 (1) #-715 (1)
IOC$RELDATAP	00000153-R 376 (1)		
IOC$RELMAPREG	000001E0-R 489 (1)		
IOC$RELSCHAN	00000048-R 186 (1)		
IOC$REQCOM	000000DD-R 296 (1)		
IOC$REQDATAP	000001A7-R 435 (1)		
IOC$REQDATAPNW	000001A3-R 432 (1)		
IOC$REQMAPREG	0000021A-R 536 (1)		
IOC$REQPCHANH	000000A9-R 253 (1)		
IOC$REQPCHANL	000000B2-R 257 (1)		

```
IOC$REQSCHANH                   00000095-R   245    (1)
IOC$REQSCHANL                   0000009F-R   249    (1)
IOC$RETURN                      000002B0-R   646    (1)
IOC$WFIKPCH                     000002B1-R   673    (1)
IOC$WFIRLCH                     000002D3-R   707    (1)
IPL$_IOPOST                    =00000004                 #-304    (1)
IRP$L_DIAGBUF                   00000044                 #-148    (1)    #-345    (1)
IRP$L_MEDIA                     00000034                 #-298    (1)
IRP$L_PID                       0000000C                 #-118    (1)
IRP$L_SVAPTE                    0000002C                 #-342    (1)
IRP$V_DIAGBUF                  =00000007                 #-147    (1)    #-344    (1)
IRP$W_CHAN                      00000028                 #-120    (1)
IRP$W_STS                       0000002A                   147    (1)      344    (1)
MMG$GL_SPTBASE                  00000000-XR              #-743    (1)
PCB$L_PID                       00000060                 #-118    (1)
PR$_IPL                        =00000012                 #-680    (1)    #-714    (1)
PR$_SIRR                       =00000014                 #-304    (1)
RELEASE                         0000011D-R   314    (1)  #-269    (1)
SS$_NORMAL                      00000000-XR              #-451    (1)
UCB$B_ERTCNT                    00000070                 #-151    (1)    #-308    (1)
UCB$L_CRB                       00000020                 #-187    (1)    #-191    (1)    #-195    (1)    #-246    (1)
                                                         #-250    (1)    #-254    (1)    #-258    (1)    #-377    (1)
                                                         #-390    (1)    #-437    (1)    #-490    (1)    #-606    (1)
UCB$L_DDB                       00000024                 #-152    (1)    #-347    (1)
UCB$L_DUETIM                    0000005C                 #-678    (1)    #-712    (1)
UCB$L_EMB                       00000078                 #-306    (1)
UCB$L_FPC                       0000000C                   207    (1)    #-266    (1)      395    (1)    #-448    (1)
                                                           503    (1)    #-540    (1)    #-676    (1)    #-710    (1)
UCB$L_FQFL                      00000000                   267    (1)      449    (1)      505    (1)      541    (1)
UCB$L_FR3                       00000010                 #-204    (1)    #-265    (1)    #-394    (1)    #-447    (1)
                                                         #-502    (1)    #-539    (1)    #-675    (1)    #-709    (1)
UCB$L_IOQFL                     00000040                   311    (1)
UCB$L_IRP                       0000004C                 #-146    (1)    #-297    (1)    #-339    (1)
UCB$L_OPCNT                     00000060                 #-299    (1)
UCB$L_SVAPTE                    00000068                 #-342    (1)
UCB$M_BSY                      =00000100                 #-313    (1)
UCB$M_CANCEL                   =00000008                 #-122    (1)    #-343    (1)
UCB$M_INT                      =00000002                 #-677    (1)    #-711    (1)
UCB$M_TIM                      =00000001                 #-677    (1)    #-711    (1)
UCB$M_TIMOUT                   =00000040                 #-343    (1)    #-679    (1)    #-713    (1)
UCB$V_BSY                      =00000008                 #-117    (1)
UCB$V_ERLOGIP                  =00000002                 #-305    (1)
UCB$W_BCNT                      0000006E                 #-601    (1)
UCB$W_BOFF                      0000006C                 #-602    (1)
UCB$W_STS                       00000058                   117    (1)    #-122    (1)      305    (1)    #-307    (1)
                                                         #-313    (1)    #-343    (1)    #-677    (1)    #-679    (1)
                                                         #-711    (1)    #-713    (1)
VEC$B_DATAPATH                  00000013                 #-379    (1)      383    (1)      393    (1)      438    (1)
                                                           444    (1)
VEC$B_NUMREG                    00000012                 #-567    (1)    #-609    (1)
VEC$L_ADP                       00000014                 #-378    (1)    #-439    (1)    #-493    (1)    #-607    (1)
VEC$L_IDB                       00000008                 #-197    (1)    #-260    (1)
VEC$S_DATAPATH                 =00000005                 #-382    (1)    #-385    (1)    #-392    (1)    #-443    (1)
VEC$V_DATAPATH                 =00000000                 #-381    (1)    #-384    (1)    #-391    (1)    #-442    (1)
VEC$V_MAPLOCK                  =0000000F                 #-491    (1)    #-608    (1)
VEC$V_PATHLOCK                 =00000007                 #-438    (1)
VEC$W_MAPREG                    00000010                   491    (1)    #-495    (1)      608    (1)    #-617    (1)
```

#-623 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ADPDEF	2	75 (1)	75 (1)
$CADEF	1	76 (1)	76 (1)
$CRBDEF	1	77 (1)	77 (1)
$DDBDEF	1	78 (1)	78 (1)
$DDTDEF	1	79 (1)	79 (1)
$DEFINI	1	75 (1)	75 (1) 76 (1) 77 (1) 78 (1) 79 (1)
			80 (1) 81 (1) 82 (1) 83 (1) 84 (1)
			85 (1) 86 (1) 87 (1) 88 (1) 89 (1)
$EMBDEF	1	80 (1)	80 (1)
$EMBDVDEF	2	80 (1)	80 (1)
$EMBETDEF	4	80 (1)	80 (1)
$EMBHDDEF	1	80 (1)	80 (1)
$EMBTSDEF	1	80 (1)	80 (1)
$IDBDEF	1	81 (1)	81 (1)
$IPLDEF	1	82 (1)	82 (1)
$IRPDEF	4	83 (1)	83 (1)
$JIBDEF	3	84 (1)	84 (1)
$LOGDEF	1	85 (1)	85 (1)
$PCBDEF	4	86 (1)	86 (1)
$PRDEF	5	87 (1)	87 (1)
$UCBDEF	10	88 (1)	88 (1)
$VECDEF	2	89 (1)	89 (1)
BUG_CHECK	1	399 (1)	399 (1)
ENBINT	1	680 (1)	680 (1) 714 (1)
SOFTINT	1	304 (1)	304 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	149 (1) 156 (1) 351 (1) 571 (1) 674 (1) 708 (1)
ADDL3	00C1	611 (1) 616 (1) 678 (1) 712 (1)
ASHL	0078	604 (1)
BBC	00E1	117 (1) 147 (1) 196 (1) 344 (1)
BBCC	00E5	305 (1)
BBCS	00E3	398 (1)
BBS	00E0	155 (1) 350 (1) 438 (1) 491 (1) 608 (1) 621 (1)
BBSC	00E4	445 (1)
BBSS	00E2	261 (1)
BEQL	0013	193 (1) 269 (1) 441 (1) 615 (1)
BGEQ	0018	569 (1) 620 (1)
BGTR	0014	613 (1)
BICB	008A	211 (1)
BICW	00AA	313 (1) 343 (1) 679 (1) 713 (1)
BISW	00A8	122 (1) 677 (1) 711 (1)
BLBC	00E9	446 (1) 501 (1)
BLBS	00E8	538 (1)
BLSS	0019	380 (1)
BNEQ	0012	119 (1) 121 (1) 199 (1) 303 (1)
BRB	0011	189 (1) 248 (1) 252 (1) 256 (1) 434 (1) 504 (1) 573 (1)
		598 (1) 622 (1)
BRW	0031	315 (1) 715 (1)
BSBB	0010	194 (1) 497 (1) 500 (1) 537 (1) 624 (1)
BVC	001C	312 (1)
BVS	001D	201 (1) 387 (1) 499 (1)
CLRL	00D4	210 (1) 605 (1) 610 (1)
CMPL	00D1	118 (1) 198 (1) 268 (1) 568 (1) 619 (1)
CMPW	00B1	120 (1) 612 (1)
CVTBL	0098	379 (1)
EXTZV	00EF	384 (1)
FFC	00EB	618 (1)
FFS	00EA	440 (1) 614 (1)
INCL	00D6	299 (1) 625 (1)
INSQUE	000E	267 (1) 302 (1) 449 (1) 505 (1) 541 (1)
INSV	00F0	381 (1) 391 (1) 442 (1) 570 (1) 574 (1)
JMP	0017	352 (1)
JSB	0016	157 (1) 207 (1) 310 (1) 395 (1) 399 (1) 503 (1)
MCOML	00D2	496 (1)
MOVAB	009E	603 (1)
MOVB	0090	609 (1)
MOVL	00D0	146 (1) 148 (1) 152 (1) 153 (1) 154 (1) 187 (1) 188 (1)
		191 (1) 192 (1) 195 (1) 197 (1) 203 (1) 204 (1) 205 (1)
		206 (1) 246 (1) 247 (1) 250 (1) 251 (1) 254 (1) 255 (1)
		258 (1) 259 (1) 260 (1) 262 (1) 263 (1) 265 (1) 297 (1)
		306 (1) 339 (1) 345 (1) 347 (1) 348 (1) 349 (1) 377 (1)
		378 (1) 389 (1) 390 (1) 437 (1) 439 (1) 490 (1) 493 (1)
		494 (1) 606 (1) 607 (1) 743 (1) 744 (1)
MOVQ	007D	150 (1) 298 (1) 309 (1) 342 (1) 346 (1) 394 (1) 447 (1)
		502 (1) 539 (1) 675 (1) 709 (1)
MOVW	00B0	307 (1) 308 (1) 617 (1)

MOVZBL	009A	451	(1)	567	(1)											
MOVZWL	003C	151	(1)	495	(1)	601	(1)	602	(1)	623	(1)					
MTPR	00DA	304	(1)	680	(1)	714	(1)									
POPL	8ED0	266	(1)	448	(1)	540	(1)	676	(1)	710	(1)					
POPR	00BA	208	(1)	396	(1)	506	(1)	626	(1)							
PUSHL	00DD	433	(1)	436	(1)											
PUSHR	00BB	202	(1)	388	(1)	492	(1)	597	(1)	600	(1)					
REMQUE	000F	200	(1)	311	(1)	386	(1)	498	(1)							
RSB	0005	123	(1)	158	(1)	209	(1)	212	(1)	264	(1)	270	(1)	397	(1)	
		400	(1)	453	(1)	507	(1)	542	(1)	575	(1)	627	(1)	647	(1)	
		681	(1)	746	(1)											
SUBL	00C2	572	(1)													
TSTL	00D5	450	(1)	452	(1)											

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	399	(1)														
.DSABL	213	(1)	271	(1)	454	(1)	628	(1)								
.ENABL	185	(1)	244	(1)	431	(1)	595	(1)								
.END	748	(1)														
.ENDC	680	(1)	714	(1)												
.ENDM	304	(1)	399	(1)	680	(1)	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)
	80	(1)	81	(1)	82	(1)	83	(1)	84	(1)	85	(1)	86	(1)	87	(1)
	88	(1)	89	(1)												
.ENDR	80	(1)														
.IDENT	6	(1)														
.IF	680	(1)	714	(1)												
.IFF	680	(1)	714	(1)												
.IRP	80	(1)														
.NOCROSS	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)	80	(1)	81	(1)	82	(1)
	83	(1)	84	(1)	85	(1)	86	(1)	87	(1)	88	(1)	89	(1)		
.PAGE	124	(1)	159	(1)	214	(1)	272	(1)	316	(1)	353	(1)	401	(1)	455	(1)
	508	(1)	543	(1)	576	(1)	629	(1)	648	(1)	682	(1)	69	(1)	718	(1)
	90	(1)														
.PSECT	115	(1)														
.RESTORE	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)	80	(1)	81	(1)	82	(1)
	83	(1)	84	(1)	85	(1)	86	(1)	87	(1)	88	(1)	89	(1)		
.SAVE	75	(1)	76	(1)	77	(1)	78	(1)	79	(1)	80	(1)	81	(1)	82	(1)
	83	(1)	84	(1)	85	(1)	86	(1)	87	(1)	88	(1)	89	(1)		
.SBTTL	125	(1)	160	(1)	215	(1)	273	(1)	317	(1)	354	(1)	402	(1)	456	(1)
	509	(1)	544	(1)	577	(1)	630	(1)	649	(1)	683	(1)	719	(1)	91	(1)
.TITLE	5	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	55	#-148	(1)	#-149	(1)	#-150	(1)	#-151	(1)	#-187	(1)	#-188	(1)
		#-191	(1)	#-192	(1)	#-195	(1)	196	(1)	#-197	(1)	200	(1)
		#-211	(1)	#-246	(1)	#-247	(1)	#-250	(1)	#-251	(1)	#-254	(1)
		#-255	(1)	#-258	(1)	#-259	(1)	#-260	(1)	261	(1)	#-298	(1)
		#-309	(1)	#-345	(1)	#-346	(1)	#-347	(1)	#-348	(1)	#-349	(1)
		#-351	(1)	#-377	(1)	#-378	(1)	#-379	(1)	383	(1)	#-386	(1)
		#-389	(1)	#-437	(1)	438	(1)	#-439	(1)	444	(1)	#-451	(1)
		#-496	(1)	#-501	(1)	#-538	(1)	#-570	(1)	#-574	(1)	#-605	(1)
		#-625	(1)	#-744	(1)								
R1	37	#-154	(1)	155	(1)	#-156	(1)	157	(1)	#-197	(1)	#-198	(1)
		#-205	(1)	#-206	(1)	#-210	(1)	#-260	(1)	#-262	(1)	#-263	(1)
		#-268	(1)	#-349	(1)	350	(1)	#-351	(1)	352	(1)	#-378	(1)
		386	(1)	#-390	(1)	393	(1)	398	(1)	#-439	(1)	440	(1)
		445	(1)	449	(1)	#-490	(1)	491	(1)	#-493	(1)	#-495	(1)
		#-567	(1)	#-606	(1)	#-607	(1)	608	(1)	#-609	(1)	#-617	(1)
		#-623	(1)										
R2	32	#-120	(1)	#-152	(1)	#-153	(1)	#-154	(1)	#-156	(1)	#-200	(1)
		#-203	(1)	#-255	(1)	#-259	(1)	267	(1)	#-306	(1)	#-307	(1)
		#-308	(1)	#-309	(1)	#-379	(1)	385	(1)	#-391	(1)	#-398	(1)
		#-440	(1)	#-442	(1)	#-445	(1)	#-493	(1)	#-494	(1)	541	(1)
		570	(1)	574	(1)	#-607	(1)	614	(1)	618	(1)	621	(1)
R3	43	#-118	(1)	#-120	(1)	#-146	(1)	147	(1)	#-148	(1)	#-202	(1)
		#-204	(1)	#-208	(1)	#-265	(1)	#-297	(1)	#-298	(1)	302	(1)
		#-311	(1)	#-339	(1)	#-342	(1)	344	(1)	#-345	(1)	#-388	(1)
		#-394	(1)	#-396	(1)	#-447	(1)	#-492	(1)	#-502	(1)	#-506	(1)
		#-539	(1)	#-567	(1)	#-568	(1)	#-572	(1)	#-574	(1)	#-597	(1)
		#-600	(1)	#-601	(1)	603	(1)	#-604	(1)	#-609	(1)	#-611	(1)
		#-616	(1)	#-626	(1)	#-675	(1)	#-709	(1)	#-743	(1)		
R4	29	#-118	(1)	#-202	(1)	#-205	(1)	#-208	(1)	#-262	(1)	#-388	(1)
		#-396	(1)	#-492	(1)	#-495	(1)	#-506	(1)	#-570	(1)	#-571	(1)
		#-574	(1)	#-597	(1)	#-600	(1)	#-602	(1)	603	(1)	#-610	(1)
		#-611	(1)	#-614	(1)	#-616	(1)	#-617	(1)	#-618	(1)	#-619	(1)
		#-621	(1)	#-623	(1)	#-626	(1)						
R5	77	117	(1)	#-122	(1)	#-146	(1)	#-151	(1)	#-152	(1)	#-187	(1)
		#-191	(1)	#-195	(1)	#-198	(1)	#-202	(1)	#-203	(1)	#-204	(1)
		#-206	(1)	207	(1)	#-208	(1)	#-246	(1)	#-250	(1)	#-254	(1)
		#-258	(1)	#-263	(1)	#-265	(1)	#-266	(1)	267	(1)	#-268	(1)
		#-297	(1)	#-299	(1)	305	(1)	#-306	(1)	#-307	(1)	#-308	(1)
		311	(1)	#-313	(1)	#-339	(1)	#-342	(1)	#-343	(1)	#-347	(1)
		#-377	(1)	#-388	(1)	#-389	(1)	#-390	(1)	#-394	(1)	395	(1)
		#-396	(1)	#-437	(1)	#-447	(1)	#-448	(1)	449	(1)	#-490	(1)
		#-492	(1)	#-498	(1)	#-502	(1)	503	(1)	505	(1)	#-506	(1)
		#-539	(1)	#-540	(1)	541	(1)	#-597	(1)	#-600	(1)	#-601	(1)
		#-602	(1)	#-606	(1)	#-611	(1)	#-612	(1)	#-616	(1)	#-619	(1)
		#-626	(1)	#-675	(1)	#-676	(1)	#-677	(1)	#-678	(1)	#-679	(1)
		#-709	(1)	#-710	(1)	#-711	(1)	#-712	(1)	#-713	(1)		
R6	5	#-492	(1)	#-494	(1)	498	(1)	505	(1)	#-506	(1)		
SP	9	#-446	(1)	#-450	(1)	#-452	(1)	#-674	(1)	#-678	(1)	#-680	(1)
		#-708	(1)	#-712	(1)	#-714	(1)						

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.03	00:00:00.18
Command processing	878	00:00:00.22	00:00:00.68
Pass 1	631	00:00:02.70	00:00:04.13
Symbol table sort	0	00:00:00.20	00:00:00.19
Pass 2	9	00:00:00.75	00:00:01.45
Symbol table output	0	00:00:00.05	00:00:00.21
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	13	00:00:00.20	00:00:00.36
Assembler run totals	1556	00:00:04.15	00:00:07.20

The working set limit was 3400 pages.
140774 bytes (275 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 747 non-local and 35 local symbols.
748 source lines were read in Pass 1, producing 53 object records in Pass 2.
104 pages of virtual memory were used to define 31 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	10
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	10
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	27

1149 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:IOSNPG.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 97 CONVERT DEVICE NAME AND UNIT
(1) 135 FIND FREE I/O CHANNEL
(1) 171 SEARCH FOR DEVICE
(1) 319 UNLOCK I/O DATA BASE AND RETURN STATUS
(1) 341 VERIFY I/O CHANNEL NUMBER

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element IOSPGD.MAR
0000     2 ; *1     6-FEB-1986 15:19:02 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element IOSPGD.MAR
0000     4         .TITLE  IOSPGD  *** IOSPGD services for QIO
0000     5         .IDENT  /05-02/
0000     6         .LIST   MEB
0000     7         .NLIST  CND
0000     8
0000     9 ;
0000    10 ;*****************************************************************************
0000    11 ;*                                                                           *
0000    12 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                    *
0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    14 ;*                                                                           *
0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    20 ;*  TRANSFERRED.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    24 ;*  CORPORATION.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    28 ;*                                                                           *
0000    29 ;*****************************************************************************
0000    30 ;
0000    31 ; D. N. CUTLER 13-JUN-76
0000    32 ;
0000    33 ; PAGED I/O RELATED SUBROUTINES
0000    34 ;
0000    35 ;
0000    36 ;
0000    37 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
0000    38 ;
0000    39 ;               Dave Butenhof   14-may-1980, Version 5.4
0000    40 ;       02      Modify VMS V2.0 source for Supervisor environment
0000    41 ;       V04     LMK0001         LEN KAWELL      27-JUL-1979
0000    42 ;               CHANGE INTERFACE TO IOC$CREATE_UCB TO ASSUME THAT THE
0000    43 ;               I/O DATABASE IS LOCKED FOR WRITE ACCESS.
0000    44 ;
0000    45 ;       V03     SPR23081        LEN KAWELL      28-MAR-1979
0000    46 ;               CHANGE OVERFLOW CHECK IN IOC$SEARCHDEV SO THAT UNIT
0000    47 ;               NUMBERS BETWEEN 32768 AND 65535 ARE VALID.
0000    48 ;
0000    49 ;       V02     SPR20488                LEN KAWELL      7-FEB-1979
0000    50 ;               RESTORE PCB ADDRESS WHEN CHECKING FOR ALLSPOOL
0000    51 ;               PRIVILEGE DURING GENERIC ALLOCATION OF A SPOOLED DEVICE.
0000    52 ;
0000    53 ; MACRO LIBRARY CALLS
0000    54 ;
0000    58
0000    59         $ACBDEF                         ;DEFINE AST CONTROL BLOCK
0000               .SAVE   LOCAL_BLOCK
```

```
                     0000              .IIF    NB,ACB$L_ASTQFL,        ACB$L_ASTQFL:
00000004             0000                      .BLKL   1
                     0004              .IIF    NB,ACB$L_ASTQBL,        ACB$L_ASTQBL:
00000008             0004                      .BLKL   1
                     0008              .IIF    NB,ACB$W_SIZE,  ACB$W_SIZE:
0000000A             0008                      .BLKW   1
                     000A              .IIF    NB,ACB$B_TYPE,  ACB$B_TYPE:
0000000B             000A                      .BLKB   1
                     000B              .IIF    NB,ACB$B_RMOD,  ACB$B_RMOD:
0000000C             000B                      .BLKB   1
                     000C              .IIF    NB,ACB$L_PID,   ACB$L_PID:
00000010             000C                      .BLKL   1
                     0010              .IIF    NB,ACB$L_AST,   ACB$L_AST:
00000014             0010                      .BLKL   1
                     0014              .IIF    NB,ACB$L_ASTPRM,        ACB$L_ASTPRM:
00000018             0014                      .BLKL   1
                     0018              .IIF    NB,ACB$C_LENGTH,        ACB$C_LENGTH:
                     0018              .IIF    NB,ACB$K_LENGTH,        ACB$K_LENGTH:
                     0018              .IIF    NB,ACB$L_KAST,  ACB$L_KAST:
0000001C             0018                      .BLKL   1
                     0000              .RESTORE
                     0000      60      $CCBDEF                         ;DEFINE CCB OFFSETS
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             001C              .=0
                     0000              .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
00000004             0000                      .BLKL   1
                     0004              .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
00000008             0004                      .BLKL   1
                     0008              .IIF    NB,CCB$B_STS,   CCB$B_STS:
00000009             0008                      .BLKB   1
                     0009              .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
0000000A             0009                      .BLKB   1
                     000A              .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
0000000C             000A                      .BLKW   1
                     000C              .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
00000010             000C                      .BLKL   1
                     0010              .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                     0010              .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                     0000              .RESTORE
                     0000      61      $CRBDEF                         ;DEFINE CRB OFFSETS
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             001C              .=0
                     0000              .IIF    NB,CRB$L_WQFL,  CRB$L_WQFL:
00000004             0000                      .BLKL   1
                     0004              .IIF    NB,CRB$L_WQBL,  CRB$L_WQBL:
00000008             0004                      .BLKL   1
                     0008              .IIF    NB,CRB$W_SIZE,  CRB$W_SIZE:
0000000A             0008                      .BLKW   1
                     000A              .IIF    NB,CRB$B_TYPE,  CRB$B_TYPE:
0000000B             000A                      .BLKB   1
0000000C             000B                      .BLKB   1
                     000C              .IIF    NB,CRB$W_REFC,  CRB$W_REFC:
0000000E             000C                      .BLKW   1
                     000E              .IIF    NB,CRB$B_MASK,  CRB$B_MASK:
0000000F             000E                      .BLKB   1
```

```
                    00000010  000F                        .BLKB   1
                              0010              .IIF    NB,CRB$L_LINK,  CRB$L_LINK:
                    00000014  0010                        .BLKL   1
                              0014              .IIF    NB,CRB$L_INTD,  CRB$L_INTD:
                    00000038  0014                        .BLKL   9
                              0038              .IIF    NB,CRB$C_LENGTH,        CRB$C_LENGTH:
                              0038              .IIF    NB,CRB$K_LENGTH,        CRB$K_LENGTH:
                              0038              .IIF    NB,CRB$L_INTD2, CRB$L_INTD2:
                    0000005C  0038                        .BLKL   9
                              0000              .RESTORE
                              0000      62      $DDBDEF                         ;DEFINE DDB OFFSETS
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  005C              .=0
                              0000              .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
                    00000004  0000                        .BLKL   1
                              0004              .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
                    00000008  0004                        .BLKL   1
                              0008              .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
                    0000000A  0008                        .BLKW   1
                              000A              .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
                    0000000B  000A                        .BLKB   1
                    0000000C  000B                        .BLKB   1
                              000C              .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
                    00000010  000C                        .BLKL   1
                              0010              .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
                    00000013  0010                        .BLKB   3
                              0013              .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
                    00000014  0013                        .BLKB   1
                              0014              .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
                    00000015  0014                        .BLKB   1
                    00000024  0015                        .BLKB   15
                              0024              .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
                    00000025  0024                        .BLKB   1
                    00000034  0025                        .BLKB   15
                              0034              .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                              0034              .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                              0000              .RESTORE
                              0000      63      $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  005C              .=0
                              0000              .RESTORE
                              0000      64      $IPLDEF                         ;DEFINE IPL LEVELS
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  005C              .=0
                              0000              .RESTORE
                              0000      65      $LOGDEF                         ;DEFINE LOG OFFSETS
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  005C              .=0
                              0000              .RESTORE
                              0000      66      $PCBDEF                         ;DEFINE PCB OFFSETS
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  005C              .=0
```

```
                    0000            .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004            0000                    .BLKL   1
                    0004            .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008            0004                    .BLKL   1
                    0008            .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A            0008                    .BLKW   1
                    000A            .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B            000A                    .BLKB   1
                    000B            .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C            000B                    .BLKB   1
                    000C            .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D            000C                    .BLKB   1
                    000D            .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E            000D                    .BLKB   1
                    000E            .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010            000E                    .BLKW   1
                    0010            .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014            0010                    .BLKL   1
                    0014            .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018            0014                    .BLKL   1
                    0018            .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C            0018                    .BLKL   1
                    001C            .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020            001C                    .BLKL   1
                    0020            .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024            0020                    .BLKL   1
                    0024            .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028            0024                    .BLKL   1
                    0028            .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C            0028                    .BLKL   1
                    002C            .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E            002C                    .BLKW   1
                    002E            .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F            002E                    .BLKB   1
                    002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030            002F                    .BLKB   1
                    0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032            0030                    .BLKW   1
                    0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034            0032                    .BLKW   1
                    0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036            0034                    .BLKW   1
                    0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038            0036                    .BLKW   1
                    0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A            0038                    .BLKW   1
                    003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C            003A                    .BLKW   1
                    003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E            003C                    .BLKW   1
                    003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040            003E                    .BLKW   1
                    0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042            0040                    .BLKW   1
                    0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044            0042                    .BLKW   1
                    0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
```

```
0000004C  0044                          .BLKB   8
          004C          .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
          004C          .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                          .BLKL   1
          0050          .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                          .BLKL   1
          0054          .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                          .BLKL   1
          0058          .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                          .BLKL   1
          005C          .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                          .BLKL   1
          0060          .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                          .BLKL   1
          0064          .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                          .BLKL   1
          0068          .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                          .BLKB   16
          0078          .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084  007C                          .BLKQ   1
          0084          .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
          0088          .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                          .BLKW   1
          008A          .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                          .BLKW   1
          008C          .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
          008C          .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
          0000          .RESTORE
          0000    67    $PRDEF                          ;DEFINE PROCESSOR REGISTERS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .RESTORE
          0000    68    $PRVDEF                         ;DEFINE PRIVILEGE BITS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .RESTORE
          0000    69    $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .RESTORE
          0000    70    $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .RESTORE
          0000    71    $UCBDEF                         ;DEFINE UCB OFFSETS
          0000          .SAVE   LOCAL_BLOCK
          0000          .PSECT  $ABS$,ABS
00000000  008C          .=0
          0000          .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
```

```
                    0000            .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004            0000                    .BLKL   1
                    0004            .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                    0004            .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008            0004                    .BLKL   1
                    0008            .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
0000000A            0008                    .BLKW   1
                    000A            .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
0000000B            000A                    .BLKB   1
                    000B            .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
0000000C            000B                    .BLKB   1
                    000C            .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C            .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C            .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
0000000D            000C                    .BLKB   1
00000010            000D                    .BLKB   3
                    0010            .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010            .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014            0010                    .BLKL   1
                    0014            .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014            .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014            .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016            0014                    .BLKW   1
                    0016            .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018            0016                    .BLKW   1
                    0018            .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018            .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
0000001A            0018                    .BLKW   1
                    001A            .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A            .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
0000001C            001A                    .BLKW   1
                    001C            .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
00000020            001C                    .BLKL   1
                    0020            .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
00000024            0020                    .BLKL   1
                    0024            .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
00000028            0024                    .BLKL   1
                    0028            .IIF    NB,UCB$L_PID,   UCB$L_PID:
0000002C            0028                    .BLKL   1
                    002C            .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030            002C                    .BLKL   1
                    0030            .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034            0030                    .BLKL   1
                    0034            .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038            0034                    .BLKL   1
                    0038            .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039            0038                    .BLKB   1
                    0039            .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A            0039                    .BLKB   1
                    003A            .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C            003A                    .BLKW   1
                    003C            .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C            .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C            .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D            003C                    .BLKB   1
                    003D            .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                    003D            .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
```

```
                    0000003E  003D                        .BLKB   1
                              003E              .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                              003E              .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
                    0000003F  003E                        .BLKB   1
                              003F              .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
                    00000040  003F                        .BLKB   1
                              0040              .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
                    00000044  0040                        .BLKL   1
                              0044              .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
                    00000048  0044                        .BLKL   1
                              0048              .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
                    0000004A  0048                        .BLKW   1
                              004A              .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                              004A              .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
                    0000004B  004A                        .BLKB   1
                              004B              .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
                    0000004C  004B                        .BLKB   1
                              004C              .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
                    00000050  004C                        .BLKL   1
                              0050              .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
                    00000052  0050                        .BLKW   1
                              0052              .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                              0052              .IIF    NB,UCB$B_STATE, UCB$B_STATE:
                    00000053  0052                        .BLKB   1
                              0053              .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
                    00000054  0053                        .BLKB   1
                              0054              .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
                    00000058  0054                        .BLKL   1
                              0058              .IIF    NB,UCB$W_STS,   UCB$W_STS:
                    0000005A  0058                        .BLKW   1
                              005A              .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
                    0000005C  005A                        .BLKW   1
                              005C              .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                              005C              .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
                    00000060  005C                        .BLKL   1
                              0060              .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
                    00000064  0060                        .BLKL   1
                              0064              .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                              0064              .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
                    00000068  0064                        .BLKL   1
                              0068              .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
                    0000006C  0068                        .BLKL   1
                              006C              .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
                    0000006E  006C                        .BLKW   1
                              006E              .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
                    00000070  006E                        .BLKW   1
                              0070              .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
                    00000071  0070                        .BLKB   1
                              0071              .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
                    00000072  0071                        .BLKB   1
                              0072              .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
                    00000074  0072                        .BLKW   1
                              0074              .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                              0074              .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
                    00000000  0074              . = 0
                              0000              .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
                    00000002  0000                        .BLKW   1
```

```
                    00000074  0002              . = 116
                              0074              .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
                    00000078  0074                              .BLKL   1
                              0078              .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
                    0000007C  0078                              .BLKL   1
                              007C              .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
                    00000080  007C                              .BLKL   1
                              0080              .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
                    00000084  0080                              .BLKL   1
                              0084              .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
                    00000088  0084                              .BLKL   1
                              0088              .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
                    0000008C  0088                              .BLKL   1
                              008C              .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
                    00000090  008C                              .BLKL   1
                              0090              .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                              0090              .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
                    00000074  0090              . = 116
                              0074              .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
                    00000075  0074                              .BLKB   1
                              0075              .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
                    00000076  0075                              .BLKB   1
                              0076              .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
                    00000077  0076                              .BLKB   1
                              0077              .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
                    00000078  0077                              .BLKB   1
                              0078              .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
                    0000007C  0078                              .BLKL   1
                    0000007E  007C                              .BLKW   1
                              007E              .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
                    00000080  007E                              .BLKW   1
                              0080              .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
                    00000084  0080                              .BLKL   1
                    00000080  0084              . = 128
                    00000084  0080                              .BLKL   1
                              0084              .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
                    00000088  0084                              .BLKL   1
                              0088              .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
                    0000008A  0088                              .BLKW   1
                              008A              .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
                    0000008C  008A                              .BLKW   1
                              008C              .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                              008C              .IIF    NB,UCB$W_DA,    UCB$W_DA:
                    0000008E  008C                              .BLKW   1
                              008E              .IIF    NB,UCB$W_DC,    UCB$W_DC:
                    00000090  008E                              .BLKW   1
                              0090              .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
                    00000092  0090                              .BLKW   1
                              0092              .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
                    00000094  0092                              .BLKW   1
                              0094              .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
                    00000095  0094                              .BLKB   1
                              0095              .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
                    00000096  0095                              .BLKB   1
                              0096              .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
                    00000098  0096                              .BLKW   1
                    00000096  0098              . = 150
```

```
00000098  0096                                  .BLKW   1
          0098                  .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                  .BLKL   1
          00A0                  .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                                  .BLKL   1
          00A4                  .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                                  .BLKW   1
          00A6                  .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                                  .BLKB   1
000000A8  00A7                                  .BLKB   1
00000074  00A8                  . = 116
00000078  0074                                  .BLKL   1
0000007C  0078                                  .BLKL   1
00000080  007C                                  .BLKL   1
00000084  0080                                  .BLKL   1
00000088  0084                                  .BLKL   1
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                  .BLKL   1
00000094  0090                                  .BLKL   1
00000098  0094                                  .BLKL   1
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                  .BLKB   1
          009D                  .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                  .BLKB   1
          009E                  .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                  .BLKB   1
000000A0  009F                                  .BLKB   1
          00A0                  .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                  .BLKB   1
          00A1                  .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                  .BLKB   1
          00A2                  .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                  .BLKB   1
000000A4  00A3                                  .BLKB   1
          00A4                  .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                  .BLKB   1
          00A5                  .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                  .BLKW   1
000000A8  00A7                                  .BLKB   1
          00A8                  .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                  .BLKL   1
000000B0  00AC                                  .BLKL   1
000000B4  00B0                                  .BLKL   1
000000B8  00B4                                  .BLKL   1
          00B8                  .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                  .BLKL   1
          00BC                  .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC                  .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC                  . = 116
          0074                  .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                  .BLKL   1
```

```
                          007C              .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
                 0000007E 007C                              .BLKW   1
                 00000080 007E                              .BLKW   1
                          0000              .RESTORE
                          0000     75
                          0000     76 ;
                          0000     77 ; LOCAL SYMBOLS
                          0000     78 ;
                          0000     79 ; CHARACTER DEFINITIONS
                          0000     80 ;
                          0000     81
                 0000003A 0000     82 COLON=58                                ;COLON
                 00000000 0000     83              .PSECT  SEP, SHR, EXE, WRT, LONG
                 00000039 0000     84 NINE=57                                 ;DIGIT 9
                 00000041 0000     85 UCA=65                                  ;UPPER CASE A
                 00000064 0000     86 UCZ=100                                 ;UPPER CASE Z
                 0000005F 0000     87 UNDERSCORE=95                           ;UNDERSCORE
                 00000030 0000     88 ZERO=48                                 ;DIGIT 0
                          0000     89
                          0000     90 ;
                          0000     91 ; LOCAL DATA
                          0000     92 ;
                          0000     93
3A 57 55 21 43 41 21 5F 00' 0000     94 DEVCTL: .ASCIC  /_!AC!UW:/              ;DEVICE NAME CONVERSION CONTROL STRING
                       08 0000
```

```
                                  0009          .SBTTL  CONVERT DEVICE NAME AND UNIT
                                  0009    ;+
                                  0009    ; IOC$CVT_DEVNAM - CONVERT DEVICE NAME AND UNIT
                                  0009    ;
                                  0009    ; THIS ROUTINE IS CALLED TO CONVERT A DEVICE NAME AND UNIT NUMBER TO A PHYSICAL
                                  0009    ; DEVICE NAME STRING.
                                  0009    ;
                                  0009    ; INPUTS:
                                  0009    ;
                                  0009    ;       R0 = LENGTH OF OUTPUT BUFFER.
                                  0009    ;       R1 = ADDRESS OF OUTPUT BUFFER.
                                  0009    ;       R6 = ADDRESS OF DEVICE UCB.
                                  0009    ;
                                  0009    ; OUTPUTS:
                                  0009    ;
                                  0009    ;       THE DEVICE NAME AND UNIT NUMBER ARE CONVERTED AND STORED IN THE SPECIFIED
                                  0009    ;       OUTPUT BUFFER. THE FOLLOWING REGISTER VALUES ARE RETURNED:
                                  0009    ;
                                  0009    ;               R0 = FINAL CONVERSION STATUS.
                                  0009    ;               R1 = LENGTH OF CONVERSION STRING.
                                  0009    ;
                                  0009    ;       R3 IS PRESERVED ACROSS CALL.
                                  0009    ;-
                                  0009
                                  0009    IOC$CVT_DEVNAM::                        ;CONVERT DEVICE NAME AND UNIT
          7E    50    7D          0009            MOVQ    R0,-(SP)                ;SAVE OUTPUT BUFFER DESCRIPTOR
          52    5E    D0          000C            MOVL    SP,R2                   ;SET ADDRESS OF OUTPUT BUFFER DESCRIPTOR
     51    EE    AF    9E          000F            MOVAB   DEVCTL,R1               ;GET ADDRESS OF CONVERSION CONTROL STRING
          50    81    9A          0013            MOVZBL  (R1)+,R0                ;GET LENGTH OF CONVERSION CONTROL STRING
          7E    50    7D          0016            MOVQ    R0,-(SP)                ;SAVE CONVERSION CONTROL STRING DESCRIPTOR
          51    5E    D0          0019            MOVL    SP,R1                   ;SET ADDRESS OF CONTROL STRING DESCRIPTOR
     50    24    A6    D0          001C            MOVL    UCB$L_DDB(R6),R0        ;GET ADDRESS OF DDB
     50    14    A0    9E          0020            MOVAB   DDB$T_NAME(R0),R0       ;GET ADDRESS OF DEVICE NAME
                                  0024            $FAO_S  (R1),(R1),(R2),R0,UCB$W_UNIT(R6) ;CONVERT DEVICE AND UNIT
          48    A6    DD          0024                    PUSHL   UCB$W_UNIT(R6)
                50    DD          0027                    PUSHL   R0
                62    7F          0029                    PUSHAQ  (R2)
                61    3F          002B                    PUSHAW  (R1)
                61    7F          002D                    PUSHAQ  (R1)
00000000'GF     05    FB          002F                    CALLS   #$$T2,G^SYS$FAO
          51    6E    3C          0036            MOVZWL  (SP),R1                 ;GET LENGTH OF OUTPUT STRING
          5E    10    C0          0039            ADDL    #16,SP                  ;REMOVE DESCRIPTORS FROM STACK
                      05          003C            RSB                             ;
```

```
                                003D   135          .SBTTL  FIND FREE I/O CHANNEL
                                003D   136 ;+
                                003D   137 ; IOC$FFCHAN - FIND FREE I/O CHANNEL
                                003D   138 ;
                                003D   139 ; THIS ROUTINE IS CALLED TO SEARCH THE I/O CHANNEL TABLE FOR A FREE CHANNEL.
                                003D   140 ;
                                003D   141 ; INPUTS:
                                003D   142 ;
                                003D   143 ;       NONE.
                                003D   144 ;
                                003D   145 ; OUTPUTS:
                                003D   146 ;
                                003D   147 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO FIND FREE I/O CHANNEL.
                                003D   148 ;
                                003D   149 ;              R0 = SS$_NOIOCHAN - NO I/O CHANNEL AVAILABLE.
                                003D   150 ;
                                003D   151 ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                                003D   152 ;
                                003D   153 ;              R1 = AVAILABLE CHANNEL NUMBER.
                                003D   154 ;
                                003D   155 ;       R3 IS PRESERVED ACROSS CALL.
                                003D   156 ;-
                                003D   157
                                003D   158 IOC$FFCHAN::                                 ;FIND FREE I/O CHANNEL
     50   09   00000000'EF   C1  003D   159          ADDL3   CTL$GL_CCBBASE,#CCB$B_AMOD,R0   ;BASE AND OFFSET TO TEST ASSIGNMENT
                    51   10   CE  0045   160          MNEGL   #CCB$C_LENGTH,R1        ;SET STARTING CHANNEL INDEX
          52   00000000'9F   3C  0048   161          MOVZWL  @#CTL$GW_NMIOCH,R2      ;GET NUMBER OF I/O CHANNELS
                      6041   95  004F   162 10$:     TSTB    (R0)[R1]                ;CHANNEL ASSIGNED?
                        0C   13  0052   163          BEQL    20$                     ;IF EQL NO
                51    10   C2  0054   164          SUBL    #CCB$C_LENGTH,R1        ;CALCULATE NEXT CHANNEL INDEX
                  F5  52   F5  0057   165          SOBGTR  R2,10$                  ;ANY MORE CCB'S TO EXAMINE?
          50   01B4  8F   3C  005A   166          MOVZWL  #SS$_NOIOCHAN,R0        ;INDICATE FAILURE
                             05  005F   167          RSB                             ;
                50   01   3C  0060   168 20$:     MOVZWL  #SS$_NORMAL,R0          ;INDICATE SUCCESS
                             05  0063   169          RSB                             ;
```

```
                     0064   171          .SBTTL  SEARCH FOR DEVICE
                     0064   172 ;+
                     0064   173 ; IOC$SEARCHDEV - SEARCH FOR PHYSICAL DEVICE
                     0064   174 ; IOC$SEARCHGEN - SEARCH FOR GENERIC DEVICE
                     0064   175 ;
                     0064   176 ; THIS ROUTINE IS CALLED TO SEARCH THE DEVICE DATA BASE FOR A SPECIFIED
                     0064   177 ; DEVICE. IT IS ASSUMED THAT THE DEVICE DATA BASE HAS BEEN LOCKED FOR
                     0064   178 ; READ ACCESS.
                     0064   179 ;
                     0064   180 ; INPUTS:
                     0064   181 ;
                     0064   182 ;       R1 = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
                     0064   183 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0064   184 ;
                     0064   185 ; OUTPUTS:
                     0064   186 ;
                     0064   187 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO FIND DEVICE.
                     0064   188 ;
                     0064   189 ;               R0 = SS$_IVDEVNAM - INVALID DEVICE NAME.
                     0064   190 ;               R0 = SS$_NONLOCAL - NONLOCAL DEVICE.
                     0064   191 ;               R0 = SS$_NOSUCHDEV - NO SUCH DEVICE.
                     0064   192 ;
                     0064   193 ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                     0064   194 ;
                     0064   195 ;               R1 = UCB ADDRESS OF DEVICE UNIT.
                     0064   196 ;-
                     0064   197
                     0064   198          .ENABL  LSB
                     0064   199 IOC$SEARCHDEV::                          ;SEARCH FOR PHYSICAL DEVICE
     52   02   D0     0064   200          MOVL    #2,R2                   ;INDICATE PHYSICAL DEVICE SEARCH
          03   11     0067   201          BRB     5$                      ;
                      0069   202 IOC$SEARCHGEN::                         ;SEARCH FOR GENERIC DEVICE
     52   01   D0     0069   203          MOVL    #1,R2                   ;INDICATE GENERIC DEVICE SEARCH
     01F4 8F   BB     006C   204 5$:      PUSHR   #^M<R2,R4,R5,R6,R7,R8>  ;SAVE REGISTERS
 52    40   8F   9A   0070   205          MOVZBL  #LOG$C_NAMLENGTH,R2     ;SET MAXIMUM LENGTH OF RESULT STRING
     5E   52   C2     0074   206          SUBL    R2,SP                   ;ALLOCATE SPACE FOR RESULT STRING
     54   61   3C     0077   207          MOVZWL  (R1),R4                 ;GET LENGTH OF NAME STRING IN BYTES
          69   13     007A   208          BEQL    70$                     ;IF EQL INVALID DEVICE NAME
 55    04   A1   D0   007C   209          MOVL    4(R1),R5                ;GET ADDRESS OF NAME STRING
 85    5F   8F   91   0080   210          CMPB    #UNDERSCORE,(R5)+       ;DEVICE NAME START WITH UNDERSCORE?
          06   12     0084   211          BNEQ    7$                      ;IF NEQ NO
          54   D7     0086   212          DECL    R4                      ;REDUCE LENGTH OF DEVICE NAME
          5B   13     0088   213          BEQL    70$                     ;IF EQL INVALID DEVICE NAME
          10   11     008A   214          BRB     10$                     ;
75   54   3A   3A     008C   215 7$:      LOCC    #^A/:/,R4,-(R5)         ;SEARCH STRING FOR A COLON
          0A   13     0090   216          BEQL    10$                     ;IF EQL COLON NOT FOUND
          50   D7     0092   217          DECL    R0                      ;POSSIBLY A NODE NAME?
          06   13     0094   218          BEQL    10$                     ;IF EQL NO
 01   A1   3A   91     0096   219          CMPB    #^A/:/,1(R1)            ;NEXT CHARACTER A COLON?
          42   13     009A   220          BEQL    50$                     ;IF EQL YES
     50   54   7D     009C   221 10$:     MOVQ    R4,R0                   ;COPY DEVICE NAME PARAMETERS
 61    64   8F   91   009F   222 20$:     CMPB    #UCZ,(R1)               ;POSSIBLY UPPER CASE ALPHABETIC?
          40   1F     00A3   223          BLSSU   70$                     ;IF LSSU NO
 61    41   8F   91   00A5   224          CMPB    #UCA,(R1)               ;UPPER CASE ALPHABETIC?
          05   1A     00A9   225          BGTRU   30$                     ;IF GTRU NO
          51   D6     00AB   226          INCL    R1                      ;POINT TO NEXT CHARACTER
     EF   50   F5     00AD   227          SOBGTR  R0,20$                  ;ANY MORE CHARACTERS TO SCAN?
```

```
                56   D4   00B0    228 30$:    CLRL    R6                          ;CLEAR UNIT NUMBER
              54 50   C2   00B2    229         SUBL    R0,R4                       ;CALCULATE LENGTH OF DEVICE NAME
                 2E   13   00B5    230         BEQL    70$                         ;IF EQL NO DEVICE NAME SPECIFIED
                 50   D7   00B7    231 40$:    DECL    R0                          ;ANY MORE CHARACTERS IN STRING?
                 31   19   00B9    232         BLSS    80$                         ;IF LSS NO
              52 81   9A   00BB    233         MOVZBL  (R1)+,R2                    ;GET NEXT CHARACTER
              52 3A   91   00BE    234         CMPB    #COLON,R2                   ;DEVICE NAME TERMINATOR?
                 29   13   00C1    235         BEQL    80$                         ;IF EQL YES
           40 AE   02   C8   00C3    236         BISL    #2,LOG$C_NAMLENGTH(SP)      ;SET EXPLICIT UNIT NUMBER FLAG
              52 30   82   00C7    237         SUBB    #ZERO,R2                    ;POSSIBLY A DECIMAL DIGIT?
                 19   19   00CA    238         BLSS    70$                         ;IF LSS NO
              52 09   91   00CC    239         CMPB    #NINE-ZERO,R2               ;DECIMAL DIGIT?
                 14   19   00CF    240         BLSS    70$                         ;IF LSS NO
              56 05   A4   00D1    241         MULW    #5,R6                       ;SCALE CURRENT UNIT NUMBER BY 10
                           00D4    242                                             ; IN TWO STEPS, FOR OVERFLOW CHECK
                 0F   1D   00D4    243         BVS     70$                         ;IF VS OVERFLOW - NUMBER TOO BIG
              56 56   A0   00D6    244         ADDW    R6,R6                       ;SECOND STEP OF SCALE BY 10
              56 52   C0   00D9    245         ADDL    R2,R6                       ;ADD NEW DIGIT TO ACCUMULATION
                 D9   11   00DC    246         BRB     40$                         ;
                           00DE    247
                           00DE    248 ;
                           00DE    249 ; NONLOCAL DEVICE
                           00DE    250 ;
                           00DE    251
        50 08F0 8F   3C   00DE    252 50$:    MOVZWL  #SS$_NONLOCAL,R0            ;SET NONLOCAL DEVICE
                 35   11   00E3    253 60$:    BRB     120$                        ;
                           00E5    254
                           00E5    255 ;
                           00E5    256 ; INVALID DEVICE NAME
                           00E5    257 ;
                           00E5    258
     50 0144 8F   3C   00E5    259 70$:    MOVZWL  #SS$_IVDEVNAM,R0            ;SET INVALID DEVICE NAME
                 2E   11   00EA    260         BRB     120$                        ;
                           00EC    261
                           00EC    262 ;
                           00EC    263 ; SEARCH DEVICE DATA BASE FOR NAME/UNIT MATCH
                           00EC    264 ;
                           00EC    265
           57   01   D0   00EC    266 80$:    MOVL    #1,R7                       ;SET DEVICE NAME LENGTH ADJUSTMENT VALUE
58 00000000'EF   DE   00EF    267         MOVAL   L^IOC$GL_DEVLIST-DDB$L_LINK,R8 ;GET ADDRESS OF I/O DATABASE LISTHEAD
                 2B   10   00F6    268 90$:    BSBB    SEARCHDEV                   ;SEARCH FOR DEVICE NAME MATCH
                 09   12   00F8    269         BNEQ    100$                        ;IF NEQ MATCH NOT FOUND
                 43   10   00FA    270 95$:    BSBB    SEARCHUNIT                  ;SEARCH FOR UNIT NUMBER MATCH
              1B 50   E8   00FC    271         BLBS    R0,120$                     ;IF LBS UNIT NUMBER MATCH FOUND
           F3 40 AE   E8   00FF    272         BLBS    LOG$C_NAMLENGTH(SP),90$     ;IF LBS GENERIC DEVICE NAME SEARCH
                 57   D4   0103    273 100$:   CLRL    R7                          ;CLEAR DEVICE NAME LENGTH ADJUSTMENT VALUE
58 00000000'EF   DE   0105    274         MOVAL   L^IOC$GL_DEVLIST-DDB$L_LINK,R8 ;GET ADDRESS OF I/O DATABASE LISTHEAD
                 15   10   010C    275         BSBB    SEARCHDEV                   ;SEARCH FOR DEVICE NAME MATCH
                 05   12   010E    276         BNEQ    110$                        ;IF NEQ MATCH NOT FOUND
                 2D   10   0110    277         BSBB    SEARCHUNIT                  ;SEARCH FOR UNIT NUMBER MATCH
              05 50   E8   0112    278         BLBS    R0,120$                     ;IF LBS UNIT NUMBER MATCH FOUND
     50 0908 8F   3C   0115    279 110$:   MOVZWL  #SS$_NOSUCHDEV,R0           ;SET NO SUCH DEVICE
        5E   40 AE   9E   011A    280 120$:   MOVAB   LOG$C_NAMLENGTH(SP),SP      ;REMOVE DEVICE NAME STRING FROM STACK
           01F4 8F   BA   011E    281         POPR    #^M<R2,R4,R5,R6,R7,R8>      ;RESTORE REGISTERS
                 05   0122    282         RSB                                 ;
                           0123    283         .DSABL  LSB
                           0123    284
```

```
                                0123    285 ;
                                0123    286 ; SUBROUTINE TO SEARCH FOR DEVICE NAME MATCH
                                0123    287 ;
                                0123    288
                                0123    289 SEARCHDEV:                                  ;SEARCH FOR DEVICE NAME
        58    68    D0          0123    290 10$:    MOVL    DDB$L_LINK(R8),R8           ;GET ADDRESS OF NEXT DDB
              14    13          0126    291         BEQL    20$                         ;IF EQL END OF LIST
     50    14 A8    DE          0128    292         MOVAL   DDB$T_NAME(R8),R0           ;GET ADDRESS OF GENERIC DEVICE NAME
  51    80    57    83          012C    293         SUBB3   R7,(R0)+,R1                 ;CALCULATE LENGTH OF STRING TO COMPARE
        54    51    91          0130    294         CMPB    R1,R4                       ;LENGTH OF NAMES MATCH?
              EE    12          0133    295         BNEQ    10$                         ;IF NEQ NO
  65    60    54    29          0135    296         CMPC    R4,(R0),(R5)                ;COMPARE DEVICE NAMES
              E8    12          0139    297         BNEQ    10$                         ;IF NEQ NAMES DO NOT MATCH
                    05          013B    298         RSB                                 ;
              58    D6          013C    299 20$:    INCL    R8                          ;INDICATE SEARCH FAILURE
                    05          013E    300         RSB                                 ;
                                013F    301
                                013F    302 ;
                                013F    303 ; SUBROUTINE TO SEARCH FOR UNIT NUMBER MATCH
                                013F    304 ;
                                013F    305
                                013F    306 SEARCHUNIT:                                 ;SEARCH FOR UNIT NUMBER
              50    D4          013F    307         CLRL    R0                          ;ASSUME SEARCH FAILURE
     51    D8 A8    DE          0141    308         MOVAL   DDB$L_UCB-UCB$L_LINK(R8),R1 ;GET ADDRESS OF NEXT UCB ADDRESS
     51    2C A1    D0          0145    309 10$:    MOVL    UCB$L_LINK(R1),R1           ;GET ADDRESS OF NEXT UCB
              0F    13          0149    310         BEQL    40$                         ;IF EQL END OF LIST
  08 44 AE    01    E1          014B    311         BBC     #1,LOG$C_NAMLENGTH+4(SP),20$ ;IF CLR, GENERIC SEARCH
     48 A1    56    B1          0150    312         CMPW    R6,UCB$W_UNIT(R1)           ;UNIT NUMBER MATCH?
              02    13          0154    313         BEQL    30$                         ;IF EQL YES
              ED    11          0156    314         BRB     10$                         ;
                                0158    315 20$:
              50    D6          0158    316 30$:    INCL    R0                          ;INDICATE UNIT NUMBER MATCH
                    05          015A    317 40$:    RSB                                 ;
```

```
                        015B   319          .SBTTL  UNLOCK I/O DATA BASE AND RETURN STATUS
                        015B   320 ;+
                        015B   321 ; IOC$UNLOCK - UNLOCK I/O DATA BASE AND RETURN STATUS
                        015B   322 ;
                        015B   323 ; THIS ROUTINE IS JUMPED TO AT THE END OF AN I/O RELATED SYSTEM SERVICE TO
                        015B   324 ; UNLOCK THE I/O DATA BASE, SET THE CURRENT PROCESSOR PRIORITY TO ZERO,
                        015B   325 ; AND TO RETURN STATUS TO THE CHANGE MODE DISPATCHER.
                        015B   326 ;
                        015B   327 ; INPUTS:
                        015B   328 ;
                        015B   329 ;       R0 = FINAL SYSTEM SERVICE STATUS VALUE.
                        015B   330 ;
                        015B   331 ; OUTPUTS:
                        015B   332 ;
                        015B   333 ;       THE I/O DATA BASE IS UNLOCKED, THE CURRENT PROCESSOR PRIORITY IS SET
                        015B   334 ;       TO ZERO, AND A RETURN TO THE CHANGE MODE DISPATCHER IS EXECUTED.
                        015B   335 ;-
                        015B   336
                        015B   337 IOC$UNLOCK::                                ;UNLOCK I/O DATA BASE AND RETURN STATUS
                        015B   338          SETIPL  #0                         ;ALLOW ALL INTERRUPTS
                    12  015B                       MTPR    #0,S^#PR$_IPL
                    04  015E   339          RET                                ;
```

```
                                015F   341          .SBTTL  VERIFY I/O CHANNEL NUMBER
                                015F   342 ;+
                                015F   343 ; IOC$VERIFYCHAN - VERIFY I/O CHANNEL NUMBER
                                015F   344 ;
                                015F   345 ; THIS ROUTINE IS CALLED TO VERIFY AND TRANSLATE AN I/O CHANNEL NUMBER TO
                                015F   346 ; A CCB ADDRESS. THE CHANNEL IS CHECKED FOR ACCESSIBILITY BY THE PREVIOUS
                                015F   347 ; ACCESS MODE.
                                015F   348 ;
                                015F   349 ; INPUTS:
                                015F   350 ;
                                015F   351 ;       R0 = I/O CHANNEL NUMBER.
                                015F   352 ;
                                015F   353 ; OUTPUTS:
                                015F   354 ;
                                015F   355 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO VERIFY.
                                015F   356 ;
                                015F   357 ;               R0 = SS$_IVCHAN - INVALID CHANNEL NUMBER.
                                015F   358 ;               R0 = SS$_NOPRIV - NO PRIVILEGE TO ACCESS CHANNEL.
                                015F   359 ;               R1 = ADDRESS OF CCB IF R0 = SS$_NOPRIV
                                015F   360 ;
                                015F   361 ;       R0 LOW BIT SET INDICATES VERIFY SUCCESS WITH:
                                015F   362 ;
                                015F   363 ;               R1 = ADDRESS OF CCB.
                                015F   364 ;               R2 = CHANNEL INDEX.
                                015F   365 ;-
                                015F   366
                                015F   367 IOC$VERIFYCHAN::                             ;VERIFY I/O CHANNEL NUMBER
          50    0F    AA      015F   368          BICW    #CCB$C_LENGTH-1,R0           ;CLEAR EXTRANEOUS LOW ORDER BITS
                28    13      0162   369          BEQL    10$                          ;IF EQL INVALID CHANNEL
  00000000'9F   50    B1      0164   370          CMPW    R0,@#CTL$GW_CHINDX           ;LEGAL CHANNEL NUMBER?
                1F    1E      016B   371          BGEQU   10$                          ;IF GEQU NO
          52    50    CE      016D   372          MNEGL   R0,R2                        ;CONVERT TO CHANNEL INDEX
51  00000000'FF42    9E      0170   373          MOVAB   @CTL$GL_CCBBASE[R2],R1       ;GET ADDRESS OF CORRESPONDING CCB
                53    DC      0178   374          MOVPSL  R3                           ;READ CURRENT PSL
53  53    02    16    EF      017A   375          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R3,R3 ;EXTRACT PREVIOUS MODE FIELD
          50    24    3C      017F   376          MOVZWL  #SS$_NOPRIV,R0               ;ASSUME CALLER DOES NOT HAVE PRIVILEGE
        09 A1   53    91      0182   377          CMPB    R3,CCB$B_AMOD(R1)            ;CALLER HAVE PRIVILEGE TO ACCESS CHANNEL?
                09    18      0186   378          BGEQ    20$                          ;IF GEQ NO
        05 50   00    E3      0188   379          BBCS    #0,R0,20$                    ;INDICATE SUCCESS
      50    013C 8F   3C      018C   380 10$:     MOVZWL  #SS$_IVCHAN,R0               ;SET INVALID CHANNEL
                      05      0191   381 20$:     RSB                                  ;
                              0192   382
                              0192   383          .END
```

```
$$T2                    = 00000005    D
ACB$B_RMOD                0000000B    D
ACB$B_TYPE                0000000A    D
ACB$C_LENGTH              00000018    D
ACB$K_LENGTH              00000018    D
ACB$L_AST                 00000010    D
ACB$L_ASTPRM              00000014    D
ACB$L_ASTQBL              00000004    D
ACB$L_ASTQFL              00000000    D
ACB$L_KAST                00000018    D
ACB$L_PID                 0000000C    D
ACB$W_SIZE                00000008    D
CCB$B_AMOD                00000009    D
CCB$B_STS                 00000008    D
CCB$C_LENGTH              00000010    D
CCB$K_LENGTH              00000010    D
CCB$L_DIRP                0000000C    D
CCB$L_UCB                 00000000    D
CCB$L_WIND                00000004    D
CCB$W_IOC                 0000000A    D
COLON                   = 0000003A    D
CRB$B_MASK                0000000E    D
CRB$B_TYPE                0000000A    D
CRB$C_LENGTH              00000038    D
CRB$K_LENGTH              00000038    D
CRB$L_INTD                00000014    D
CRB$L_INTD2               00000038    D
CRB$L_LINK                00000010    D
CRB$L_WQBL                00000004    D
CRB$L_WQFL                00000000    D
CRB$W_REFC                0000000C    D
CRB$W_SIZE                00000008    D
CTL$GL_CCBBASE            ********    X   02
CTL$GW_CHINDX             ********    X   02
CTL$GW_NMIOCH             ********    X   02
DDB$B_ACPCLASS            00000013    D
DDB$B_TYPE                0000000A    D
DDB$C_LENGTH              00000034    D
DDB$K_LENGTH              00000034    D
DDB$L_ACPD                00000010    D
DDB$L_DDT                 0000000C    D
DDB$L_LINK                00000000    D
DDB$L_UCB                 00000004    D
DDB$T_DRVNAME             00000024    D
DDB$T_NAME                00000014    D
DDB$W_SIZE                00000008    D
DEVCTL                    00000000 R  D   02
IOC$CVT_DEVNAM            00000009 RG D   02
IOC$FFCHAN                0000003D RG D   02
IOC$GL_DEVLIST            ********    X   02
IOC$SEARCHDEV             00000064 RG D   02
IOC$SEARCHGEN             00000069 RG D   02
IOC$UNLOCK                0000015B RG D   02
IOC$VERIFYCHAN            0000015F RG D   02
LOG$C_NAMLENGTH         = 00000040    D
NINE                    = 00000039    D
PCB$B_ASTACT              0000000C    D
PCB$B_ASTEN               0000000D    D
PCB$B_PRI                 0000000B    D
PCB$B_PRIB                0000002F    D
PCB$B_TYPE                0000000A    D
PCB$B_WEFC                0000002E    D
PCB$C_LENGTH              0000008C    D
PCB$K_LENGTH              0000008C    D
PCB$L_ARB                 00000084    D
PCB$L_ASTQBL              00000014    D
PCB$L_ASTQFL              00000010    D
PCB$L_EFC2P               00000058    D
PCB$L_EFC3P               0000005C    D
PCB$L_EFCS                00000050    D
PCB$L_EFCU                00000054    D
PCB$L_EFWM                0000004C    D
PCB$L_JIB                 00000078    D
PCB$L_OWNER               0000001C    D
PCB$L_PHD                 00000064    D
PCB$L_PHYPCB              00000018    D
PCB$L_PID                 00000060    D
PCB$L_PQB                 0000004C    D
PCB$L_SQBL                00000004    D
PCB$L_SQFL                00000000    D
PCB$L_STS                 00000024    D
PCB$L_UIC                 00000088    D
PCB$L_WSSWP               00000020    D
PCB$L_WTIME               00000028    D
PCB$Q_PRIV                0000007C    D
PCB$T_LNAME               00000068    D
PCB$T_TERMINAL            00000044    D
PCB$W_APTCNT              00000030    D
PCB$W_ASTCNT              00000038    D
PCB$W_BIOCNT              0000003A    D
PCB$W_BIOLM               0000003C    D
PCB$W_DIOCNT              0000003E    D
PCB$W_DIOLM               00000040    D
PCB$W_GPGCNT              00000034    D
PCB$W_GRP                 0000008A    D
PCB$W_MEM                 00000088    D
PCB$W_MTXCNT              0000000E    D
PCB$W_PPGCNT              00000036    D
PCB$W_PRCCNT              00000042    D
PCB$W_SIZE                00000008    D
PCB$W_STATE               0000002C    D
PCB$W_TMBU                00000032    D
PR$_IPL                 = 00000012    D
PSL$S_PRVMOD            = 00000002    D
PSL$V_PRVMOD            = 00000016    D
SEARCHDEV                 00000123 R  D   02
SEARCHUNIT                0000013F R  D   02
SIZ...                  = 00000002    D
SS$_IVCHAN              = 0000013C    D
SS$_IVDEVNAM            = 00000144    D
SS$_NOIOCHAN            = 000001B4    D
SS$_NONLOCAL            = 000008F0    D
SS$_NOPRIV              = 00000024    D
SS$_NORMAL              = 00000001    D
```

SS$_NOSUCHDEV	= 00000908	D		UCB$L_FR3	00000010	D
SYS$FAO	********	X	02	UCB$L_FR4	00000014	D
UCA	= 00000041	D		UCB$L_IOQBL	00000044	D
UCB$B_AMOD	00000053	D		UCB$L_IOQFL	00000040	D
UCB$B_CEX	00000077	D		UCB$L_IRP	0000004C	D
UCB$B_CM1	0000004A	D		UCB$L_LINK	0000002C	D
UCB$B_CM2	0000004B	D		UCB$L_LOGADR	00000064	D
UCB$B_DEVCLASS	00000038	D		UCB$L_MAXBLOCK	00000084	D
UCB$B_DEVTYPE	00000039	D		UCB$L_MB_MBX	0000007C	D
UCB$B_DIPL	00000052	D		UCB$L_MB_PORT	0000008C	D
UCB$B_DX_SCTCNT	000000A6	D		UCB$L_MB_RAST	00000078	D
UCB$B_ERTCNT	00000070	D		UCB$L_MB_SHB	00000080	D
UCB$B_ERTMAX	00000071	D		UCB$L_MB_WAST	00000074	D
UCB$B_FEX	00000076	D		UCB$L_MB_WIOQBL	00000088	D
UCB$B_FIPL	0000000B	D		UCB$L_MB_WIOQFL	00000084	D
UCB$B_LOCSRV	0000003C	D		UCB$L_MEDIA	0000008C	D
UCB$B_OFFNDX	00000094	D		UCB$L_NT_DATSSB	00000074	D
UCB$B_OFFRTC	00000095	D		UCB$L_NT_INTSSB	00000078	D
UCB$B_REMSRV	0000003D	D		UCB$L_OPCNT	00000060	D
UCB$B_SECTORS	0000003C	D		UCB$L_OWNUIC	0000001C	D
UCB$B_SLAVE	00000074	D		UCB$L_PID	00000028	D
UCB$B_SPR	00000075	D		UCB$L_RQBL	00000004	D
UCB$B_STATE	00000052	D		UCB$L_RQFL	00000000	D
UCB$B_TRACKS	0000003D	D		UCB$L_SVAPTE	00000068	D
UCB$B_TT_CRFILL	0000009D	D		UCB$L_SVPN	00000064	D
UCB$B_TT_DECRF	000000A1	D		UCB$L_TT_DECHAR	000000A8	D
UCB$B_TT_DELFF	000000A2	D		UCB$L_TT_RDUE	0000008C	D
UCB$B_TT_DESPEE	000000A0	D		UCB$L_TT_RTIMOU	000000B8	D
UCB$B_TT_DETYPE	000000A4	D		UCB$L_VCB	00000030	D
UCB$B_TT_LFFILL	0000009E	D		UCB$T_PARTNER	0000000C	D
UCB$B_TT_SPEED	0000009C	D		UCB$W_BCNT	0000006E	D
UCB$B_TYPE	0000000A	D		UCB$W_BCR	00000096	D
UCB$B_VERTSZ	0000003F	D		UCB$W_BOFF	0000006C	D
UCB$C_LENGTH	00000074	D		UCB$W_BUFQUO	00000018	D
UCB$C_MB_LENGTH	00000090	D		UCB$W_BYTESTOGO	0000003E	D
UCB$C_TT_LENGTH	000000BC	D		UCB$W_CHARGE	0000004A	D
UCB$K_LENGTH	00000074	D		UCB$W_CYLINDERS	0000003E	D
UCB$K_MB_LENGTH	00000090	D		UCB$W_DA	0000008C	D
UCB$K_TT_LENGTH	000000BC	D		UCB$W_DC	0000008E	D
UCB$L_AMB	00000054	D		UCB$W_DEVBUFSIZ	0000003A	D
UCB$L_ASTQBL	00000010	D		UCB$W_DEVSTS	0000005A	D
UCB$L_ASTQFL	0000000C	D		UCB$W_DIRSEQ	00000088	D
UCB$L_CPID	0000005C	D		UCB$W_DSTADDR	00000018	D
UCB$L_CRB	00000020	D		UCB$W_DX_BCR	000000A4	D
UCB$L_DDB	00000024	D		UCB$W_EC1	00000090	D
UCB$L_DEVCHAR	00000034	D		UCB$W_EC2	00000092	D
UCB$L_DEVDEPEND	0000003C	D		UCB$W_ERRCNT	00000072	D
UCB$L_DPC	00000080	D		UCB$W_FUNC	0000007E	D
UCB$L_DUETIM	0000005C	D		UCB$W_MB_SEED	00000000	D
UCB$L_DX_BFPNT	0000009C	D		UCB$W_MSGCNT	00000016	D
UCB$L_DX_BUF	00000098	D		UCB$W_MSGMAX	00000014	D
UCB$L_DX_RXDB	000000A0	D		UCB$W_NT_CHAN	0000007C	D
UCB$L_EMB	00000078	D		UCB$W_OFFSET	0000008A	D
UCB$L_FIRST	00000014	D		UCB$W_REFC	00000050	D
UCB$L_FPC	0000000C	D		UCB$W_SIZE	00000008	D
UCB$L_FQBL	00000004	D		UCB$W_SRCADDR	0000001A	D
UCB$L_FQFL	00000000	D		UCB$W_STS	00000058	D

```
UCB$W_TT_DESIZE                   000000A5    D
UCB$W_UNIT                        00000048    D
UCB$W_VPROT                       0000001A    D
UCZ                             = 00000064    D
UNDERSCORE                      = 0000005F    D
ZERO                            = 00000030    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000000BC (188.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000192 (402.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T2	=00000005	130 (1)	130 (1)
CCB$B_AMOD	00000009		#-159 (1) #-377 (1)
CCB$C_LENGTH	00000010		#-160 (1) #-164 (1) #-368 (1)
COLON	=0000003A	82 (1)	#-234 (1)
CTL$GL_CCBBASE	00000000-XR		#-159 (1) 373 (1)
CTL$GW_CHINDX	00000000-XR		#-370 (1)
CTL$GW_NMIOCH	00000000-XR		#-161 (1)
DDB$L_LINK	00000000		267 (1) 274 (1) #-290 (1)
DDB$L_UCB	00000004		308 (1)
DDB$T_NAME	00000014		129 (1) 292 (1)
DEVCTL	00000000-R	94 (1)	124 (1)
IOC$CVT_DEVNAM	00000009-R	121 (1)	
IOC$FFCHAN	0000003D-R	158 (1)	
IOC$GL_DEVLIST	00000000-XR		267 (1) 274 (1)
IOC$SEARCHDEV	00000064-R	199 (1)	
IOC$SEARCHGEN	00000069-R	202 (1)	
IOC$UNLOCK	0000015B-R	337 (1)	
IOC$VERIFYCHAN	0000015F-R	367 (1)	
LOG$C_NAMLENGTH	=00000040		#-205 (1) #-236 (1) #-272 (1) 280 (1) 311 (1)
NINE	=00000039	84 (1)	#-239 (1)
PR$_IPL	=00000012		#-338 (1)
PSL$S_PRVMOD	=00000002		#-375 (1)
PSL$V_PRVMOD	=00000016		#-375 (1)
SEARCHDEV	00000123-R	289 (1)	#-268 (1) #-275 (1)
SEARCHUNIT	0000013F-R	306 (1)	#-270 (1) #-277 (1)
SS$_IVCHAN	=0000013C		#-380 (1)
SS$_IVDEVNAM	=00000144		#-259 (1)
SS$_NOIOCHAN	=000001B4		#-166 (1)
SS$_NONLOCAL	=000008F0		#-252 (1)
SS$_NOPRIV	=00000024		#-376 (1)
SS$_NORMAL	=00000001		#-168 (1)
SS$_NOSUCHDEV	=00000908		#-279 (1)
SYS$FAO	00000000-XR		130 (1)
UCA	=00000041	85 (1)	#-224 (1)
UCB$L_DDB	00000024		#-128 (1)
UCB$L_LINK	0000002C		308 (1) #-309 (1)
UCB$W_UNIT	00000048		#-130 (1) #-312 (1)
UCZ	=00000064	86 (1)	#-222 (1)
UNDERSCORE	=0000005F	87 (1)	#-210 (1)
ZERO	=00000030	88 (1)	#-237 (1) #-239 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	59 (1)	59 (1)
$CCBDEF	1	60 (1)	60 (1)
$CRBDEF	1	61 (1)	61 (1)
$DDBDEF	1	62 (1)	62 (1)
$DEFINI	1	59 (1)	59 (1) 60 (1) 61 (1) 62 (1) 63 (1)
			64 (1) 65 (1) 66 (1) 67 (1) 68 (1)
			69 (1) 70 (1) 71 (1)
$DEVDEF	1	63 (1)	63 (1)
$FAO_S	2	130 (1)	130 (1)
$IPLDEF	1	64 (1)	64 (1)
$LOGDEF	1	65 (1)	65 (1)
$PCBDEF	4	66 (1)	66 (1)
$PRDEF	5	67 (1)	67 (1)
$PRVDEF	4	68 (1)	68 (1)
$PSLDEF	2	69 (1)	69 (1)
$PUSHADR	1	130 (1)	130 (1)
$SSDEF	21	70 (1)	70 (1)
$UCBDEF	10	71 (1)	71 (1)
SETIPL	1	338 (1)	338 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	132 (1) 245 (1)
ADDL3	00C1	159 (1)
ADDW	00A0	244 (1)
BBC	00E1	311 (1)
BBCS	00E3	379 (1)
BEQL	0013	163 (1) 208 (1) 213 (1) 216 (1) 218 (1) 220 (1) 230 (1)
		235 (1) 291 (1) 310 (1) 313 (1) 369 (1)
BGEQ	0018	378 (1)
BGEQU	001E	371 (1)
BGTRU	001A	225 (1)
BICW	00AA	368 (1)
BISL	00C8	236 (1)
BLBS	00E8	271 (1) 272 (1) 278 (1)
BLSS	0019	232 (1) 238 (1) 240 (1)
BLSSU	001F	223 (1)
BNEQ	0012	211 (1) 269 (1) 276 (1) 295 (1) 297 (1)
BRB	0011	201 (1) 214 (1) 246 (1) 253 (1) 260 (1) 314 (1)
BSBB	0010	268 (1) 270 (1) 275 (1) 277 (1)
BVS	001D	243 (1)
CALLS	00FB	130 (1)
CLRL	00D4	228 (1) 273 (1) 307 (1)
CMPB	0091	210 (1) 219 (1) 222 (1) 224 (1) 234 (1) 239 (1) 294 (1)
		377 (1)
CMPC	0029	296 (1)
CMPW	00B1	312 (1) 370 (1)
DECL	00D7	212 (1) 217 (1) 231 (1)
EXTZV	00EF	375 (1)
INCL	00D6	226 (1) 299 (1) 316 (1)
LOCC	003A	215 (1)
MNEGL	00CE	160 (1) 372 (1)
MOVAB	009E	124 (1) 129 (1) 280 (1) 373 (1)
MOVAL	00DE	267 (1) 274 (1) 292 (1) 308 (1)
MOVL	00D0	123 (1) 127 (1) 128 (1) 200 (1) 203 (1) 209 (1) 266 (1)
		290 (1) 309 (1)
MOVPSL	00DC	374 (1)
MOVQ	007D	122 (1) 126 (1) 221 (1)
MOVZBL	009A	125 (1) 205 (1) 233 (1)
MOVZWL	003C	131 (1) 161 (1) 166 (1) 168 (1) 207 (1) 252 (1) 259 (1)
		279 (1) 376 (1) 380 (1)
MTPR	00DA	338 (1)
MULW	00A4	241 (1)
POPR	00BA	281 (1)
PUSHAQ	007F	130 (1)
PUSHAW	003F	130 (1)
PUSHL	00DD	130 (1)
PUSHR	00BB	204 (1)
RET	0004	339 (1)
RSB	0005	133 (1) 167 (1) 169 (1) 282 (1) 298 (1) 300 (1) 317 (1)
		381 (1)
SOBGTR	00F5	165 (1) 227 (1)

```
SUBB             0082      237     (1)
SUBB3            0083      293     (1)
SUBL             00C2      164     (1)    206     (1)     229     (1)
TSTB             0095      162     (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	94	(1)														
.DSABL	283	(1)														
.ENABL	198	(1)														
.END	383	(1)														
.ENDC	130	(1)	338	(1)												
.ENDM	130	(1)	338	(1)	59	(1)	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)
	65	(1)	66	(1)	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)		
.IDENT	5	(1)														
.IF	130	(1)	338	(1)												
.IFF	130	(1)	338	(1)												
.IRP	130	(1)														
.LIST	57	(1)	6	(1)	73	(1)	74	(1)								
.NLIST	55	(1)	56	(1)	7	(1)	72	(1)								
.NOCROSS	59	(1)	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)
	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)						
.PAGE	134	(1)	170	(1)	318	(1)	340	(1)	95	(1)	96	(1)				
.PSECT	83	(1)														
.RESTORE	59	(1)	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)
	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)						
.SAVE	59	(1)	60	(1)	61	(1)	62	(1)	63	(1)	64	(1)	65	(1)	66	(1)
	67	(1)	68	(1)	69	(1)	70	(1)	71	(1)						
.SBTTL	135	(1)	171	(1)	319	(1)	341	(1)	97	(1)						
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	32	#-122	(1)	#-125	(1)	#-126	(1)	#-128	(1)	129	(1)	#-130	(1)
		#-159	(1)	#-162	(1)	#-166	(1)	#-168	(1)	#-217	(1)	#-221	(1)
		#-227	(1)	#-229	(1)	#-231	(1)	#-252	(1)	#-259	(1)	#-271	(1)
		#-278	(1)	#-279	(1)	#-292	(1)	#-293	(1)	296	(1)	#-307	(1)
		#-316	(1)	#-368	(1)	#-370	(1)	#-372	(1)	#-376	(1)	379	(1)
		#-380	(1)										
R1	24	#-124	(1)	#-125	(1)	#-127	(1)	130	(1)	#-131	(1)	#-160	(1)
		#-162	(1)	#-164	(1)	#-207	(1)	#-209	(1)	#-219	(1)	#-222	(1)
		#-224	(1)	#-226	(1)	#-233	(1)	#-293	(1)	#-294	(1)	#-308	(1)
		#-309	(1)	#-312	(1)	#-373	(1)	#-377	(1)				
R2	17	#-123	(1)	130	(1)	#-161	(1)	#-165	(1)	#-200	(1)	#-203	(1)
		#-204	(1)	#-205	(1)	#-206	(1)	#-233	(1)	#-234	(1)	#-237	(1)
		#-239	(1)	#-245	(1)	#-281	(1)	#-372	(1)	373	(1)		
R3	4	#-374	(1)	375	(1)	#-377	(1)						
R4	9	#-204	(1)	#-207	(1)	#-212	(1)	#-215	(1)	#-221	(1)	#-229	(1)
		#-281	(1)	#-294	(1)	#-296	(1)						
R5	6	#-204	(1)	#-209	(1)	#-210	(1)	215	(1)	#-281	(1)	296	(1)
R6	10	#-128	(1)	#-130	(1)	#-204	(1)	#-228	(1)	#-241	(1)	#-244	(1)
		#-245	(1)	#-281	(1)	#-312	(1)						
R7	5	#-204	(1)	#-266	(1)	#-273	(1)	#-281	(1)	#-293	(1)		
R8	9	#-204	(1)	#-267	(1)	#-274	(1)	#-281	(1)	#-290	(1)	292	(1)
		#-299	(1)	308	(1)								
SP	12	#-122	(1)	#-123	(1)	#-126	(1)	#-127	(1)	#-131	(1)	#-132	(1)
		#-206	(1)	#-236	(1)	#-272	(1)	280	(1)	311	(1)		

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.04	00:00:00.20
Command processing	889	00:00:00.24	00:00:00.68
Pass 1	801	00:00:02.79	00:00:04.19
Symbol table sort	0	00:00:00.30	00:00:00.29
Pass 2	52	00:00:00.97	00:00:01.55
Symbol table output	2	00:00:00.04	00:00:00.10
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	5	00:00:00.15	00:00:00.27
Assembler run totals	1775	00:00:04.54	00:00:07.29

The working set limit was 2900 pages.
168114 bytes (329 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1027 non-local and 25 local symbols.
383 source lines were read in Pass 1, producing 67 object records in Pass 2.
79 pages of virtual memory were used to define 25 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              8
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           2
SYS$COMMON:[SYSLIB]STARLET.MLB;2                      11
TOTALS (all libraries)                                21

1350 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:IOSPGD.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1)	67	APPLY ECC CORRECTION
(1)	129	CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
(1)	167	MAP VIRTUAL TO LOGICAL BLOCK
(1)	278	UPDATE TRANSFER PARAMETERS
(1)	328	SENSE DISK'S SIZE FDT ROUTINE

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element IOSRAM.MAR
0000     2 ; *1     6-FEB-1986 15:19:05 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element IOSRAM.MAR
0000     4         .TITLE  IOSRAM  *** IOSRAM random access I/O service routines
0000     5         .IDENT  /05-02/
0000     6         .LIST   MEB
0000     7         .NLIST  CND
0000     8
0000     9 ;
0000    10 ;*****************************************************************************
0000    11 ;*                                                                           *
0000    12 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                    *
0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    14 ;*                                                                           *
0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    20 ;*  TRANSFERRED.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    24 ;*  CORPORATION.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    28 ;*                                                                           *
0000    29 ;*****************************************************************************
0000    30 ;
0000    31 ; D. N. CUTLER 16-MAR-77
0000    32 ;
0000    33 ; MODIFIED BY  T.L. SOUTTER  20-FEB-78
0000    34 ;
0000    35 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
0000    36 ;
0000    37 ;               Dave Butenhof   14-may-1980, Version 5.4
0000    38 ;       02      Modified VMS V2.0 source for Supervisor environment
0000    39 ;
0000    40 ; NONPAGED RANDOM ACCESS MASS STORAGE I/O RELATED ROUTINES
0000    41 ;
0000    42 ; MACRO LIBRARY CALLS
0000    43 ;
0000    44
0000    45
0000    49
0000    50         $DEVDEF                         ;DEFINE DEVICE CHARACTERISTIC BITS
0000               .SAVE   LOCAL_BLOCK
0000               .RESTORE
0000    51         $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
0000               .SAVE   LOCAL_BLOCK
0000               .RESTORE
0000    52         $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
0000               .SAVE   LOCAL_BLOCK
0000               .RESTORE
0000    53         $IRPDEF                         ;DEFINE IRP OFFSETS
0000               .SAVE   LOCAL_BLOCK
```

```
                        0000            .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
00000004  0000                                   .BLKL   1
                        0004            .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
00000008  0004                                   .BLKL   1
                        0008            .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
0000000A  0008                                   .BLKW   1
                        000A            .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
0000000B  000A                                   .BLKB   1
                        000B            .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
0000000C  000B                                   .BLKB   1
                        000C            .IIF    NB,IRP$L_PID,   IRP$L_PID:
00000010  000C                                   .BLKL   1
                        0010            .IIF    NB,IRP$L_AST,   IRP$L_AST:
00000014  0010                                   .BLKL   1
                        0014            .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
00000018  0014                                   .BLKL   1
                        0018            .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
0000001C  0018                                   .BLKL   1
                        001C            .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
00000020  001C                                   .BLKL   1
                        0020            .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
00000022  0020                                   .BLKW   1
                        0022            .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
00000023  0022                                   .BLKB   1
                        0023            .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
00000024  0023                                   .BLKB   1
                        0024            .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
00000028  0024                                   .BLKL   1
                        0028            .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
0000002A  0028                                   .BLKW   1
                        002A            .IIF    NB,IRP$W_STS,   IRP$W_STS:
0000002C  002A                                   .BLKW   1
                        002C            .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
00000030  002C                                   .BLKL   1
                        0030            .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
00000032  0030                                   .BLKW   1
                        0032            .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
00000034  0032                                   .BLKW   1
                        0034            .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
                        0034            .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
00000038  0034                                   .BLKL   1
                        0038            .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
                        0038            .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
                        0038            .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
00000039  0038                                   .BLKB   1
0000003C  0039                                   .BLKB   3
                        003C            .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
                        003C            .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
                        003C            .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                                   .BLKW   1
                        003E            .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                                   .BLKW   1
                        0040            .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                                   .BLKL   1
                        0044            .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                                   .BLKL   1
                        0048            .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
```

```
0000004C  0048                                  .BLKL   1
          004C                  .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                                  .BLKL   1
00000058  0054                                  .BLKL   1
0000005C  0058                                  .BLKL   1
          005C                  .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
          005C                  .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
          0000                  .RESTORE
          0000      54          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  005C                  .=0
          0000                  .RESTORE
          0000      55          $PTEDEF                         ;DEFINE PAGE TABLE ENTRY FIELDS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  005C                  .=0
          0000                  .RESTORE
          0000      56          $RVTDEF                         ;DEFINE RVT OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  005C                  .=0
          0000                  .RESTORE
          0000      57          $UCBDEF                         ;DEFINE UCB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  005C                  .=0
          0000                  .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
          0000                  .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
          0004                  .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
          000C                  .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
          000C                  .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
0000000D  000C                                  .BLKB   1
00000010  000D                                  .BLKB   3
          0010                  .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
          0010                  .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
          0014                  .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
          0014                  .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016  0014                                  .BLKW   1
          0016                  .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018  0016                                  .BLKW   1
          0018                  .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
          0018                  .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
```

```
                    0000001A  0018                        .BLKW   1
                              001A          .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                              001A          .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                    0000001C  001A                        .BLKW   1
                              001C          .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                    00000020  001C                        .BLKL   1
                              0020          .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                    00000024  0020                        .BLKL   1
                              0024          .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                    00000028  0024                        .BLKL   1
                              0028          .IIF    NB,UCB$L_PID,   UCB$L_PID:
                    0000002C  0028                        .BLKL   1
                              002C          .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
                    00000030  002C                        .BLKL   1
                              0030          .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
                    00000034  0030                        .BLKL   1
                              0034          .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
                    00000038  0034                        .BLKL   1
                              0038          .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
                    00000039  0038                        .BLKB   1
                              0039          .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
                    0000003A  0039                        .BLKB   1
                              003A          .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
                    0000003C  003A                        .BLKW   1
                              003C          .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                              003C          .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                              003C          .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
                    0000003D  003C                        .BLKB   1
                              003D          .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                              003D          .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
                    0000003E  003D                        .BLKB   1
                              003E          .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                              003E          .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
                    0000003F  003E                        .BLKB   1
                              003F          .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
                    00000040  003F                        .BLKB   1
                              0040          .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
                    00000044  0040                        .BLKL   1
                              0044          .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
                    00000048  0044                        .BLKL   1
                              0048          .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
                    0000004A  0048                        .BLKW   1
                              004A          .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                              004A          .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
                    0000004B  004A                        .BLKB   1
                              004B          .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
                    0000004C  004B                        .BLKB   1
                              004C          .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
                    00000050  004C                        .BLKL   1
                              0050          .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
                    00000052  0050                        .BLKW   1
                              0052          .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                              0052          .IIF    NB,UCB$B_STATE, UCB$B_STATE:
                    00000053  0052                        .BLKB   1
                              0053          .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
                    00000054  0053                        .BLKB   1
                              0054          .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
```

```
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
          005A                  .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
          005C                  .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C                  .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                  .BLKL   1
          0060                  .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                                  .BLKL   1
          0064                  .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
          0064                  .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                                  .BLKL   1
          0068                  .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                                  .BLKL   1
          006C                  .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                                  .BLKW   1
          006E                  .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                                  .BLKW   1
          0070                  .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                                  .BLKB   1
          0071                  .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                                  .BLKB   1
          0072                  .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                                  .BLKW   1
          0074                  .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
          0074                  .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074                  . = 0
          0000                  .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                                  .BLKW   1
00000074  0002                  . = 116
          0074                  .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                                  .BLKL   1
          0080                  .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                                  .BLKL   1
          0090                  .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
          0090                  .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                  . = 116
          0074                  .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                                  .BLKB   1
          0075                  .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                                  .BLKB   1
          0076                  .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                                  .BLKB   1
          0077                  .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                                  .BLKB   1
          0078                  .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
```

```
0000007C  0078                          .BLKL   1
0000007E  007C                          .BLKW   1
          007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                          .BLKW   1
          0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                          .BLKL   1
00000080  0084          . = 128
00000084  0080                          .BLKL   1
          0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                          .BLKW   1
          008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                          .BLKW   1
          008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                          .BLKW   1
          008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                          .BLKW   1
          0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                          .BLKW   1
          0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                          .BLKW   1
          0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                          .BLKB   1
          0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                          .BLKB   1
          0096          .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                          .BLKW   1
00000096  0098          . = 150
00000098  0096                          .BLKW   1
          0098          .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                          .BLKL   1
          009C          .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                          .BLKL   1
          00A0          .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                          .BLKL   1
          00A4          .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                          .BLKW   1
          00A6          .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                          .BLKB   1
000000A8  00A7                          .BLKB   1
00000074  00A8          . = 116
00000078  0074                          .BLKL   1
0000007C  0078                          .BLKL   1
00000080  007C                          .BLKL   1
00000084  0080                          .BLKL   1
00000088  0084                          .BLKL   1
0000008C  0088                          .BLKL   1
          008C          .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                          .BLKL   1
00000094  0090                          .BLKL   1
00000098  0094                          .BLKL   1
0000009C  0098                          .BLKL   1
          009C          .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                          .BLKB   1
          009D          .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
```

```
0000009E  009D                                  .BLKB   1
          009E                  .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                  .BLKB   1
000000A0  009F                                  .BLKB   1
          00A0                  .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                  .BLKB   1
          00A1                  .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                  .BLKB   1
          00A2                  .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                  .BLKB   1
000000A4  00A3                                  .BLKB   1
          00A4                  .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                  .BLKB   1
          00A5                  .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                  .BLKW   1
000000A8  00A7                                  .BLKB   1
          00A8                  .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                  .BLKL   1
000000B0  00AC                                  .BLKL   1
000000B4  00B0                                  .BLKL   1
000000B8  00B4                                  .BLKL   1
          00B8                  .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                  .BLKL   1
          00BC                  .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC                  .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC                  . = 116
          0074                  .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                  .BLKW   1
00000080  007E                                  .BLKW   1
          0000                  .RESTORE
          0000     58           $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .RESTORE
          0000     59           $WCBDEF                         ;DEFINE WCB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .IIF    NB,WCB$L_WLFL,  WCB$L_WLFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,WCB$L_WLBL,  WCB$L_WLBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,WCB$W_SIZE,  WCB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,WCB$B_TYPE,  WCB$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,WCB$B_ACCESS,        WCB$B_ACCESS:
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,WCB$L_PID,   WCB$L_PID:
0000000E  000C                                  .BLKB   2
          000E                  .IIF    NB,WCB$W_REFCNT,        WCB$W_REFCNT:
00000010  000E                                  .BLKW   1
```

```
                        0010            .IIF    NB,WCB$L_ORGUCB,        WCB$L_ORGUCB:
00000014                0010                    .BLKL   1
                        0014            .IIF    NB,WCB$W_ACON,  WCB$W_ACON:
00000016                0014                    .BLKW   1
                        0016            .IIF    NB,WCB$W_NMAP,  WCB$W_NMAP:
00000018                0016                    .BLKW   1
                        0018            .IIF    NB,WCB$L_FCB,   WCB$L_FCB:
0000001C                0018                    .BLKL   1
                        001C            .IIF    NB,WCB$L_RVT,   WCB$L_RVT:
00000020                001C                    .BLKL   1
                        0020            .IIF    NB,WCB$L_STVBN, WCB$L_STVBN:
00000024                0020                    .BLKL   1
                        0024            .IIF    NB,WCB$C_MAP,   WCB$C_MAP:
                        0024            .IIF    NB,WCB$K_MAP,   WCB$K_MAP:
                        0024            .IIF    NB,WCB$C_LENGTH,        WCB$C_LENGTH:
                        0024            .IIF    NB,WCB$K_LENGTH,        WCB$K_LENGTH:
                        0024            .IIF    NB,WCB$W_P1_COUNT,      WCB$W_P1_COUNT:
00000026                0024                    .BLKW   1
                        0026            .IIF    NB,WCB$L_P1_LBN,        WCB$L_P1_LBN:
0000002A                0026                    .BLKL   1
                        002A            .IIF    NB,WCB$W_P2_COUNT,      WCB$W_P2_COUNT:
0000002C                002A                    .BLKW   1
                        002C            .IIF    NB,WCB$L_P2_LBN,        WCB$L_P2_LBN:
00000030                002C                    .BLKL   1
00000000                0030            . = 0
                        0000            .IIF    NB,WCB$W_COUNT, WCB$W_COUNT:
00000002                0000                    .BLKW   1
                        0002            .IIF    NB,WCB$L_LBN,   WCB$L_LBN:
00000006                0002                    .BLKL   1
00000000                0006            . = 0
FFFFFFFA                0000                    .BLKB   -6
                        FFFA            .IIF    NB,WCB$W_PREVCOUNT,     WCB$W_PREVCOUNT:
FFFFFFFC                FFFA                    .BLKW   1
                        FFFC            .IIF    NB,WCB$L_PREVLBN,       WCB$L_PREVLBN:
00000000                FFFC                    .BLKL   1
                        0000            .RESTORE
```

```
                    00000000     64          .PSECT  SEP, SHR, EXE, WRT, LONG
                        0000     65
```

```
                     0000     67          .SBTTL  APPLY ECC CORRECTION
                     0000     68 ;+
                     0000     69 ; IOC$APPLYECC - APPLY ECC CORRECTION
                     0000     70 ;
                     0000     71 ; THIS ROUTINE IS CALLED TO APPLY AN ECC CORRECTION TO DATA THAT HAS BEEN
                     0000     72 ; TRANSFERED INTO MEMORY FROM A DISK DEVICE.
                     0000     73 ;
                     0000     74 ; INPUTS:
                     0000     75 ;
                     0000     76 ;       R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED UP TO, BUT NOT
                     0000     77 ;            INCLUDING, BLOCK TO BE CORRECTED (MUST BE A MULTIPLE OF 512
                     0000     78 ;            BYTES).
                     0000     79 ;       R5 = DEVICE UNIT UCB ADDRESS.
                     0000     80 ;
                     0000     81 ;       UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
                     0000     82 ;       UCB$W_EC1(R5) = STARTING BIT NUMBER OF ERROR BURST.
                     0000     83 ;       UCB$W_EC2(R5) = EXCLUSIVE OR CORRECTION PATTERN.
                     0000     84 ;       UCB$L_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
                     0000     85 ;            THE TRANSFER.
                     0000     86 ;
                     0000     87 ; OUTPUTS:
                     0000     88 ;
                     0000     89 ;       THE CORRECTION PATTERN IS EXCLUSIVE OR'ED WITH THE DATA IN MEMORY
                     0000     90 ;       PROVIDING THE NECESSARY CORRECTION.
                     0000     91 ;
                     0000     92 ;       R3 IS PRESERVED ACROSS CALL.
                     0000     93 ;-
                     0000     94
                     0000     95 IOC$APPLYECC::                               ;APPLY ECC CORRECTION
            18   BB  0000     96          PUSHR   #^M<R3,R4>                  ;SAVE REGISTERS
       52   0090 C5  3C  0002     97          MOVZWL  UCB$W_EC1(R5),R2            ;GET STARTING BIT NUMBER OF ERROR BURST
            52   D7  0007     98          DECL    R2                          ;CONVERT TO RELATIVE BIT NUMBER
   51   52    F8 8F  8B  0009     99          BICB3   #^XF8,R2,R1                 ;ISOLATE PATTERN SHIFT COUNT
        52    08 C6  000E    100          DIVL    #8,R2                       ;CALCULATE RELATIVE BYTE OFFSET IN BLOCK
        52    50 C0  0011    101          ADDL    R0,R2                       ;CALCULATE RELATIVE OFFSET IN BUFFER
   50   0092 C5  3C  0014    102          MOVZWL  UCB$W_EC2(R5),R0            ;GET EXCLUSIVE OR CORRECTION PATTERN
   50   50   51  78  0019    103          ASHL    R1,R0,R0                    ;SHIFT PATTERN TO PROPER POSITION
        54   03  D0  001D    104          MOVL    #3,R4                       ;SET LOOP COUNT
     6E A5   52  B1  0020    105 10$:     CMPW    R2,UCB$W_BCNT(R5)           ;BYTE OFFSET WITHIN RANGE?
            47  1E  0024    106          BGEQU   40$                         ;IF GEQU NO
      51   6C A5  3C  0026    107          MOVZWL  UCB$W_BOFF(R5),R1           ;GET BYTE OFFSET IN PAGE
        51   52  C0  002A    108          ADDL    R2,R1                       ;CALCULATE BYTE OFFSET OF TRANFER PTE
   51   51    F7 8F  78  002D    109          ASHL    #-VA$S_BYTE,R1,R1           ;CALCULATE LONGWORD OFFSET TO TRANSFER PTE
   53   68 B541  D0  0032    110          MOVL    @UCB$L_SVAPTE(R5)[R1],R3    ;GET TRANSFER PTE
            03  19  0037    111          BLSS    20$                         ;IF LSS VALID PTE
          FFC4'  30  0039    112          BSBW    IOC$PTETOPFN                ;CONVERT TO VALID PTE
   51   64 A5   04  C5  003C    113 20$:     MULL3   #4,UCB$L_SVPN(R5),R1        ;CALCULATE BYTE OFFSET TO SYSTEM PTE
        00   53  F0  0041    114          INSV    R3,#PTE$V_PFN,-             ;MOVE TRANSFER PTE INTO SYSTEM PAGE TABLE
00000000'FF41   15       0044    115                  #PTE$S_PFN,@MMG$GL_SPTBASE[R1]  ;
      51   51   07  78  004B    116          ASHL    #VA$S_BYTE-2,R1,R1          ;CONVERT SVPN TO SYSTEM VIRTUAL ADDRESS
        00 51   1F  E2  004F    117          BBSS    #VA$V_SYSTEM,R1,30$         ;SET SYSTEM VIRTUAL ADDRESS BIT
                    0053    118 30$:     INVALID R1                          ;INVALIDATE TRANSLATION BUFFER
        3A   51  DA  0053                    MTPR    R1,S^#PR$_TBIS
   53   52    6C A5  C1  0056    119          ADDL3   UCB$W_BOFF(R5),R2,R3        ;CALCULATE BYTE OFFSET IN BLOCK
51   09   00   53  F0  005B    120          INSV    R3,#VA$V_BYTE,#VA$S_BYTE,R1 ;INSERT BYTE OFFSET IN BLOCK
        61   50  8C  0060    121          XORB    R0,(R1)                     ;CORRECT MEMORY BYTE
   50   50    F8 8F  78  0063    122          ASHL    #-8,R0,R0                   ;SHIFT NEXT CORRECTION BYTE INTO PLACE
```

```
                52   D6  0068   123          INCL    R2                              ;UPDATE OFFSET IN BUFFER
             B3 54   F5  006A   124          SOBGTR  R4,10$                          ;ANY MORE CORRECTIONS TO MAKE?
                18   BA  006D   125 40$:      POPR    #^M<R3,R4>                      ;RESTORE REGISTERS
          5A A5 01   A8  006F   126          BISW    #UCB$M_ECC,UCB$W_DEVSTS(R5) ;SET ECC CORRECTION MADE
                05   0073   127          RSB                                     ;
```

```
                                0074   129          .SBTTL  CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                                0074   130 ;+
                                0074   131 ; IOC$CVTLOGPHY - CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                                0074   132 ;
                                0074   133 ; THIS ROUTINE IS CALLED TO CONDITIONALLY CONVERT A LOGICAL BLOCK NUMBER
                                0074   134 ; TO A PHYSICAL DISK ADDRESS AND STORE THE RESULT IN THE I/O PACKET.
                                0074   135 ;
                                0074   136 ; INPUTS:
                                0074   137 ;
                                0074   138 ;       R0 = LOGICAL BLOCK NUMBER TO BE CONVERTED.
                                0074   139 ;       R3 = I/O PACKET ADDRESS.
                                0074   140 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                0074   141 ;
                                0074   142 ; OUTPUTS:
                                0074   143 ;
                                0074   144 ;       IF UCB$V_NOCNVRT IS CLEAR IN UCB$W_DEVSTS, THE LOGICAL BLOCK NUMBER
                                0074   145 ;       IS CONVERTED TO A PHYSICAL DISK ADDRESS USING THE DISK GEOMETRY PARA-
                                0074   146 ;       METERS IN THE UCB. THE RESULT IS STORED IN THE MEDIA ADDRESS LONGWORD
                                0074   147 ;       OF THE I/O PACKET.
                                0074   148 ;
                                0074   149 ;       IF UCB$V_NOCNVRT IS SET, THE BLOCK NUMBER IS STORED IN THE MEDIA ADDRESS
                                0074   150 ;       LONGWORD WITHOUT CONVERSION.
                                0074   151 ;
                                0074   152 ;       R3 IS PRESERVED ACROSS CALL.
                                0074   153 ;-
                                0074   154
                                0074   155 IOC$CVTLOGPHY::                            ;CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
          34 A3    50    D0   0074   156          MOVL    R0,IRP$L_MEDIA(R3)         ;ASSUME NO CONVERSION
       1D 5A A5    02    E0   0078   157          BBS     #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),10$ ;BYPASS CONVERSION IF SET
          52    3C A5    9A   007D   158          MOVZBL  UCB$L_DEVDEPEND(R5),R2     ;GET NUMBER OF SECTORS PER TRACK
                   51    D4   0081   159          CLRL    R1                         ;CLEAR HIGH PART OF DIVIDEND
34 A3     50    50    52    7B   0083   160          EDIV    R2,R0,R0,IRP$L_MEDIA(R3)   ;CALCULATE SECTOR NUMBER AND STORE
          52    3D A5    9A   0089   161          MOVZBL  UCB$L_DEVDEPEND+1(R5),R2   ;GET NUMBER OF TRACKS PER CYLINDER
   51     50    50    52    7B   008D   162          EDIV    R2,R0,R0,R1                ;CALCULATE TRACK AND CYLINDER
          35 A3    51    90   0092   163          MOVB    R1,IRP$L_MEDIA+1(R3)       ;STORE TRACK NUMBER
          36 A3    50    B0   0096   164          MOVW    R0,IRP$L_MEDIA+2(R3)       ;STORE CYLINDER NUMBER
                   05         009A   165 10$:      RSB                                ;
```

```
                                009B   167          .SBTTL  MAP VIRTUAL TO LOGICAL BLOCK
                                009B   168 ;+
                                009B   169 ; IOC$MAPVBLK - MAP VIRTUAL TO LOGICAL BLOCK
                                009B   170 ;
                                009B   171 ; THIS ROUTINE IS CALLED TO MAP A VIRTUAL BLOCK TO A LOGICAL BLOCK USING A
                                009B   172 ; MAPPING WINDOW.
                                009B   173 ;
                                009B   174 ; INPUTS:
                                009B   175 ;
                                009B   176 ;       R0 = VIRTUAL BLOCK NUMBER.
                                009B   177 ;       R1 = NUMBER OF BYTES TO MAP.
                                009B   178 ;       R2 = ADDRESS OF WINDOW MAPPING BLOCK.
                                009B   179 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                009B   180 ;
                                009B   181 ; OUTPUTS:
                                009B   182 ;
                                009B   183 ;       R0 LOW BIT CLEAR INDICATES A TOTAL MAPPING FAILURE.
                                009B   184 ;
                                009B   185 ;               R2 = NUMBER OF UNMAPPED BYTES.
                                009B   186 ;
                                009B   187 ;       R0 LOW BIT SET INDICATES PARTIAL MAP WITH:
                                009B   188 ;
                                009B   189 ;               R1 = LOGICAL BLOCK NUMBER OF FIRST BLOCK.
                                009B   190 ;               R2 = NUMBER OF UNMAPPED BYTES.
                                009B   191 ;               R5 = UCB ADDRESS OF DEVICE UNIT (POSSIBLY MODIFIED).
                                009B   192 ;
                                009B   193 ;       R3 IS PRESERVED ACROSS CALL.
                                009B   194 ;-
                                009B   195
                                009B   196 IOC$MAPVBLK::                                ;MAP VIRTUAL TO LOGICAL BLOCK
    12    0A A2    91          009B   197          CMPB    WCB$B_TYPE(R2),#DYN$C_WCB ;SEE IF THIS IS REALLY A WINDOW
          77 12                009F   198          BNEQ    110$                      ;IF NEQ NO
                               00A1   199 10$:     DSBINT  UCB$B_FIPL(R5)            ;SYNCHRONIZE ACCESS TO SYSTEM DATABASE
       7E   12    DB          00A1                       MFPR    S^#PR$_IPL,-(SP)
    12    0B A5    DA          00A4                       MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
          1A       BB          00A8   200          PUSHR   #^M<R1,R3,R4>             ;SAVE REGISTERS
    53    16 A2    3C          00AA   201          MOVZWL  WCB$W_NMAP(R2),R3         ;GET COUNT OF RETRIEVAL POINTERS
          1B       13          00AE   202          BEQL    30$                       ;BRANCH IF EMPTY WINDOW
    54    20 A2    DE          00B0   203          MOVAL   WCB$L_STVBN(R2),R4        ;POINT TO STARTING VBN
       50    84    C2          00B4   204          SUBL    (R4)+,R0                  ;SUBTRACT STARTING VBN FROM DESIRED
 13 34 A5    05    E0          00B7   205          BBS     #DEV$V_SQD,UCB$L_DEVCHAR(R5),40$ ;IF SET, SEQUENTIAL DEVICE
          0D       1F          00BC   206          BLSSU   30$                       ;BRANCH IF VBN PRECEDES WINDOW
                               00BE   207
                               00BE   208 ;
                               00BE   209 ; SCAN THE WINDOW, SUBTRACTING THE COUNT FIELD OF EACH POINTER FROM THE
                               00BE   210 ; CURRENT RELATIVE BLOCK NUMBER.
                               00BE   211 ;
                               00BE   212
       51    84    3C          00BE   213 20$:     MOVZWL  (R4)+,R1                  ;GET COUNT FIELD OF RETRIEVAL POINTER
       50    51    C2          00C1   214          SUBL    R1,R0                     ;SUBTRACT FROM RELATIVE BLOCK NUMBER
          0E       1F          00C4   215          BLSSU   50$                       ;BRANCH IF VBN LOCATED IN THIS POINTER
          84       D5          00C6   216          TSTL    (R4)+                     ;SKIP LBN FIELD OF POINTER
       F3 53       F5          00C8   217          SOBGTR  R3,20$                    ;LOOP THRU WINDOW
          50       D4          00CB   218 30$:     CLRL    R0                        ;VBN IS BEYOND WINDOW
          43       11          00CD   219          BRB     100$                      ;RETURN FAILURE
                               00CF   220
                               00CF   221 ;
```

```
                                00CF   222 ; DEVICE IS A SEQUENTIAL DEVICE. FIRST MAPPING POINTER CONTAINS THE UCB ADDRESS
                                00CF   223 ; OF THE CURRENT VOLUME THAT IS BEING PROCESSED. ALL BYTES ALWAYS MAP.
                                00CF   224 ;
                                00CF   225
                     55   64   D0  00CF   226 40$:    MOVL    (R4),R5                 ;GET CURRENT VOLUME UCB ADDRESS
                          35   11  00D2   227         BRB     80$                     ;
                                00D4   228
                                00D4   229 ;
                                00D4   230 ; FOUND THE RETRIEVAL POINTER CONTAINING THE STARTING VBN. R0 NOW
                                00D4   231 ; CONTAINS A NEGATIVE VALUE WHICH IS THE NUMBER OF BLOCKS BETWEEN
                                00D4   232 ; THE STARTING VBN AND THE END OF THE POINTER.
                                00D4   233 ;
                                00D4   234
                          50    DD  00D4   235 50$:    PUSHL   R0                      ;SAVE # BLOCKS MAPPED PAST START VBN
                     51   84    C0  00D6   236         ADDL    (R4)+,R1                ;FIRST LBN BEYOND THIS POINTER
                     50   51    C0  00D9   237         ADDL    R1,R0                   ;COMPUTE STARTING LBN
                                00DC   238
                                00DC   239 ;
                                00DC   240 ; IF THE NEXT RETRIEVAL POINTER IS CONTIGUOUS WITH THE ONE FOUND, ADD
                                00DC   241 ; IN ITS COUNT TO HANDLE THE CASE WHERE A TRANSFER SPANS TWO POINTERS.
                                00DC   242 ; NOTE THAT THE GREATEST NUMBER OF CONTIGUOUS POINTERS A TRANSFER CAN
                                00DC   243 ; SPAN IS TWO.
                                00DC   244 ;
                                00DC   245
                          53    D7  00DC   246         DECL    R3                      ;SEE IF THERE IS ANOTHER POINTER
                          0B    15  00DE   247         BLEQ    60$                     ;BRANCH IF NONE
                     53   84    3C  00E0   248         MOVZWL  (R4)+,R3                ;GET COUNT OF NEXT RETRIEVAL POINTER
                     64   51    D1  00E3   249         CMPL    R1,(R4)                 ;SEE IF THE NEXT POINTER IS CONTIGUOUS
                          03    12  00E6   250         BNEQ    60$                     ;BRANCH IF NOT
                     6E   53    C2  00E8   251         SUBL    R3,(SP)                 ;ADD TO # BLOCKS MAPPED (NEGATIVE)
                                00EB   252
                                00EB   253 ;
                                00EB   254 ; EXTRACT THE LBN AND RVN COMPONENTS OF THE STARTING "LBN" AND SWITCH
                                00EB   255 ; TO THE RIGHT UCB IF THIS IS A MULTI-VOLUME SET.
                                00EB   256 ;
                                00EB   257
51   50   18   00    EF  00EB   258 60$:    EXTZV   #0,#24,R0,R1            ;EXTRACT LBN PART
50   50   08   18    EF  00F0   259         EXTZV   #24,#8,R0,R0            ;EXTRACT RVN
                09    13  00F5   260         BEQL    70$                     ;BRANCH IF NOT VOLUME SET
          52   1C A2    D0  00F7   261         MOVL    WCB$L_RVT(R2),R2        ;GET RELATIVE VOLUME TABLE ADDR
     55   40 A240    D0  00FB   262         MOVL    RVT$L_UCBLST-4(R2)[R0],R5 ;GET THE RIGHT UCB ADDRESS
                                0100   263
                                0100   264 ;
                                0100   265 ; SEE IF THE ENTIRE TRANSFER IS MAPPED CONTIGUOUSLY.
                                0100   266 ;
                                0100   267
     6E   6E    09    78  0100   268 70$:    ASHL    #9,(SP),(SP)            ;CONVERT TO # BYTES MAPPED
          6E    8E    C0  0104   269         ADDL    (SP)+,(SP)              ;SUBTRACT FROM BYTES DESIRED
                02    18  0107   270         BGEQ    90$                     ;BRANCH IF NOT TOTAL MAP
                6E    D4  0109   271 80$:    CLRL    (SP)                    ;ZERO INDICATES COMPLETE MAP
50   00000000'8F    D0  010B   272 90$:    MOVL    #SS$_NORMAL,R0          ;INDICATE SUCCESS
                1C    BA  0112   273 100$:   POPR    #^M<R2,R3,R4>           ;RESTORE REGISTERS
                            0114   274         ENBINT                          ;ALLOW INTERRUPTS
          12    8E    DA  0114                       MTPR    (SP)+,S^#PR$_IPL
                      05  0117   275         RSB                             ;
                            0118   276 110$:   BUG_CHECK STRNOTWCB,FATAL       ;STRUCTURE IS NOT A WINDOW BLOCK
     00000000'9F    16  0118                       JSB     @#BUG$CHECK
```

```
46 2C 42 43 57 54 4F 4E 52 54 53 00' 011E              .ASCIC  "STRNOTWCB,FATAL"
                         4C 41 54 41  012A
                               0F     011E
```

```
                                012E   278          .SBTTL  UPDATE TRANSFER PARAMETERS
                                012E   279 ;+
                                012E   280 ; IOC$UPDATRANSP - UPDATE TRANSFER PARAMETERS
                                012E   281 ;
                                012E   282 ; THIS ROUTINE IS CALLED TO UPDATE THE TRANSFER PARAMETERS AFTER A DISK ERROR
                                012E   283 ; HAS BEEN DISCOVERED BUT GOOD DATA WAS TRANSFERED.
                                012E   284 ;
                                012E   285 ; INPUTS:
                                012E   286 ;
                                012E   287 ;       R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED (MUST BE A MULTIPLE
                                012E   288 ;             OF 512 BYTES).
                                012E   289 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                012E   290 ;
                                012E   291 ;       UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
                                012E   292 ;       UCB$W_DA(R5) = CURRENT SECTOR AND TRACK ADDRESS.
                                012E   293 ;       UCB$W_DC(R5) = CURRENT CYLINDER ADDRESS.
                                012E   294 ;       UCB$L_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
                                012E   295 ;             THE TRANSFER.
                                012E   296 ;
                                012E   297 ; OUTPUTS:
                                012E   298 ;
                                012E   299 ;       THE NUMBER OF BYTES REMAINING TO BE TRANSFERED, THE SYSTEM VIRTUAL
                                012E   300 ;       ADDRESS OF THE NEXT PTE, AND THE CURRENT DISK ADDRESS OF THE TRANSFER
                                012E   301 ;       ARE UPDATED.
                                012E   302 ;
                                012E   303 ;       R3 IS PRESERVED ACROSS CALL.
                                012E   304 ;-
                                012E   305
                                012E   306 IOC$UPDATRANSP::                          ;UPDATE TRANSFER PARAMETERS
      6E A5    50    A2         012E   307          SUBW    R0,UCB$W_BCNT(R5)       ;CALCULATE REMAINING BYTES TO TRANSFER
50    50   F9 8F    78         0132   308          ASHL    #-7,R0,R0               ;CALCULATE PTE LONGWORDS TO SKIP OVER
      68 A5    50    C0         0137   309          ADDL    R0,UCB$L_SVAPTE(R5)     ;UPDATE SYSTEM VIRTUAL ADDRESS OF NEXT PTE
         50    04    C6         013B   310          DIVL    #4,R0                   ;CALCULATE NUMBER OF SECTORS TRANSFERED
   008C C5    50    80         013E   311          ADDB    R0,UCB$W_DA(R5)         ;UPDATE SECTOR ADDRESS
                                0143   312
                                0143   313 ;
                                0143   314 ; RIPPLE CARRY FROM SECTOR TO TRACK AND FROM TRACK TO CYLINDER
                                0143   315 ;
                                0143   316
008C C5    3C A5    91         0143   317 10$:     CMPB    UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SECTOR OVERFLOW?
           1E       1A         0149   318          BGTRU   20$                     ;IF GTRU NO
008C C5    3C A5    82         014B   319          SUBB    UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SUBTRACT OUT A TRACK
      008D C5       96         0151   320          INCB    UCB$W_DA+1(R5)          ;INCREMENT TRACK ADDRESS
008D C5    3D A5    91         0155   321          CMPB    UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;TRACK OVERFLOW?
           E6       1A         015B   322          BGTRU   10$                     ;IF GTRU NO
008D C5    3D A5    82         015D   323          SUBB    UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;SUBTRACT OUT A CYLINDER
      008E C5       B6         0163   324          INCW    UCB$W_DC(R5)            ;UPDATE CYLINDER ADDRESS
           DA       11         0167   325          BRB     10$                     ;
                    05         0169   326 20$:     RSB                             ;
```

```
                     016A   328          .SBTTL  SENSE DISK'S SIZE FDT ROUTINE
                     016A   329 ;+
                     016A   330 ; IOC$SENSEDISK - SENSE DISK'S SIZE FDT ROUTINE
                     016A   331 ;
                     016A   332 ; THIS ROUTINE IS THE STANDARD SENSEMODE/SENSECHAR FDT ROUTINE FOR
                     016A   333 ; DISK DEVICES.  IT OBTAINS THE DISK'S SIZE, IN LOGICAL BLOCKS, FROM THE
                     016A   334 ; UCB (UCB$L_MAXBLOCK) AND IMMEDIATELY COMPLETES THE I/O REQUEST WITH
                     016A   335 ; THE SECOND LONGWORD OF THE FINAL I/O STATUS EQUAL TO THE DISK'S SIZE.
                     016A   336 ;
                     016A   337 ; INPUTS:
                     016A   338 ;
                     016A   339 ;       R0 = SCRATCH.
                     016A   340 ;       R1 = SCRATCH.
                     016A   341 ;       R2 = SCRATCH.
                     016A   342 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     016A   343 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     016A   344 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     016A   345 ;       R6 = ADDRESS OF CCB.
                     016A   346 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     016A   347 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     016A   348 ;       R9 = SCRATCH.
                     016A   349 ;       R10 = SCRATCH.
                     016A   350 ;       R11 = SCRATCH.
                     016A   351 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     016A   352 ;
                     016A   353 ; OUTPUTS:
                     016A   354 ;
                     016A   355 ;       THE DISK'S SIZE, IN LOGICAL BLOCKS, IS OBTAINED FROM THE UCB
                     016A   356 ;       AND THE I/O IS COMPLETED WITH THE SECOND I/O STATUS LONGWORD
                     016A   357 ;       EQUAL TO THE DISK'S SIZE.
                     016A   358 ;-
                     016A   359
                     016A   360 IOC$SENSEDISK::                              ;SENSE DISK'S SIZE
 51   0084 C5   D0   016A   361          MOVL    UCB$L_MAXBLOCK(R5),R1       ;GET DISK'S SIZE IN LOGICAL BLOCKS
 50     00'  3C     016F   362          MOVZWL  S^#SS$_NORMAL,R0            ;SET NORMAL COMPLETION STATUS
    FE8B'  31     0172   363          BRW     EXE$FINISHIO                ;FINISH I/O OPERATION
                     0175   364
                     0175   365          .END
```

BUG$CHECK	********	X	02	UCB$B_DEVCLASS	00000038	D	UCB$L_MAXBLOCK	00000084	D
DEV$V_SQD	= 00000005	D		UCB$B_DEVTYPE	00000039	D	UCB$L_MB_MBX	0000007C	D
DYN$C_WCB	= 00000012	D		UCB$B_DIPL	00000052	D	UCB$L_MB_PORT	0000008C	D
EXE$FINISHIO	********	X	02	UCB$B_DX_SCTCNT	000000A6	D	UCB$L_MB_RAST	00000078	D
IOC$APPLYECC	00000000 RG	D	02	UCB$B_ERTCNT	00000070	D	UCB$L_MB_SHB	00000080	D
IOC$CVTLOGPHY	00000074 RG	D	02	UCB$B_ERTMAX	00000071	D	UCB$L_MB_WAST	00000074	D
IOC$MAPVBLK	0000009B RG	D	02	UCB$B_FEX	00000076	D	UCB$L_MB_WIOQBL	00000088	D
IOC$PTETOPFN	********	X	02	UCB$B_FIPL	0000000B	D	UCB$L_MB_WIOQFL	00000084	D
IOC$SENSEDISK	0000016A RG	D	02	UCB$B_LOCSRV	0000003C	D	UCB$L_MEDIA	0000008C	D
IOC$UPDATRANSP	0000012E RG	D	02	UCB$B_OFFNDX	00000094	D	UCB$L_NT_DATSSB	00000074	D
IRP$B_CARCON	00000038	D		UCB$B_OFFRTC	00000095	D	UCB$L_NT_INTSSB	00000078	D
IRP$B_EFN	00000022	D		UCB$B_REMSRV	0000003D	D	UCB$L_OPCNT	00000060	D
IRP$B_PRI	00000023	D		UCB$B_SECTORS	0000003C	D	UCB$L_OWNUIC	0000001C	D
IRP$B_RMOD	0000000B	D		UCB$B_SLAVE	00000074	D	UCB$L_PID	00000028	D
IRP$B_TYPE	0000000A	D		UCB$B_SPR	00000075	D	UCB$L_RQBL	00000004	D
IRP$C_LENGTH	0000005C	D		UCB$B_STATE	00000052	D	UCB$L_RQFL	00000000	D
IRP$K_LENGTH	0000005C	D		UCB$B_TRACKS	0000003D	D	UCB$L_SVAPTE	00000068	D
IRP$L_ARB	00000050	D		UCB$B_TT_CRFILL	0000009D	D	UCB$L_SVPN	00000064	D
IRP$L_AST	00000010	D		UCB$B_TT_DECRF	000000A1	D	UCB$L_TT_DECHAR	000000A8	D
IRP$L_ASTPRM	00000014	D		UCB$B_TT_DELFF	000000A2	D	UCB$L_TT_RDUE	0000008C	D
IRP$L_DIAGBUF	00000044	D		UCB$B_TT_DESPEE	000000A0	D	UCB$L_TT_RTIMOU	000000B8	D
IRP$L_EXTEND	0000004C	D		UCB$B_TT_DETYPE	000000A4	D	UCB$L_VCB	00000030	D
IRP$L_IOQBL	00000004	D		UCB$B_TT_LFFILL	0000009E	D	UCB$M_ECC	= 00000001	D
IRP$L_IOQFL	00000000	D		UCB$B_TT_SPEED	0000009C	D	UCB$T_PARTNER	0000000C	D
IRP$L_IOSB	00000024	D		UCB$B_TYPE	0000000A	D	UCB$V_NOCNVRT	= 00000002	D
IRP$L_IOST1	00000034	D		UCB$B_VERTSZ	0000003F	D	UCB$W_BCNT	0000006E	D
IRP$L_IOST2	00000038	D		UCB$C_LENGTH	00000074	D	UCB$W_BCR	00000096	D
IRP$L_MEDIA	00000034	D		UCB$C_MB_LENGTH	00000090	D	UCB$W_BOFF	0000006C	D
IRP$L_PID	0000000C	D		UCB$C_TT_LENGTH	000000BC	D	UCB$W_BUFQUO	00000018	D
IRP$L_SEGVBN	00000040	D		UCB$K_LENGTH	00000074	D	UCB$W_BYTESTOGO	0000003E	D
IRP$L_SEQNUM	00000048	D		UCB$K_MB_LENGTH	00000090	D	UCB$W_CHARGE	0000004A	D
IRP$L_SVAPTE	0000002C	D		UCB$K_TT_LENGTH	000000BC	D	UCB$W_CYLINDERS	0000003E	D
IRP$L_TT_TERM	00000038	D		UCB$L_AMB	00000054	D	UCB$W_DA	0000008C	D
IRP$L_UCB	0000001C	D		UCB$L_ASTQBL	00000010	D	UCB$W_DC	0000008E	D
IRP$L_WIND	00000018	D		UCB$L_ASTQFL	0000000C	D	UCB$W_DEVBUFSIZ	0000003A	D
IRP$Q_NT_PRVMSK	0000003C	D		UCB$L_CPID	0000005C	D	UCB$W_DEVSTS	0000005A	D
IRP$W_ABCNT	0000003C	D		UCB$L_CRB	00000020	D	UCB$W_DIRSEQ	00000088	D
IRP$W_BCNT	00000032	D		UCB$L_DDB	00000024	D	UCB$W_DSTADDR	00000018	D
IRP$W_BOFF	00000030	D		UCB$L_DEVCHAR	00000034	D	UCB$W_DX_BCR	000000A4	D
IRP$W_CHAN	00000028	D		UCB$L_DEVDEPEND	0000003C	D	UCB$W_EC1	00000090	D
IRP$W_FUNC	00000020	D		UCB$L_DPC	00000080	D	UCB$W_EC2	00000092	D
IRP$W_OBCNT	0000003E	D		UCB$L_DUETIM	0000005C	D	UCB$W_ERRCNT	00000072	D
IRP$W_SIZE	00000008	D		UCB$L_DX_BFPNT	0000009C	D	UCB$W_FUNC	0000007E	D
IRP$W_STS	0000002A	D		UCB$L_DX_BUF	00000098	D	UCB$W_MB_SEED	00000000	D
IRP$W_TT_PRMPT	0000003C	D		UCB$L_DX_RXDB	000000A0	D	UCB$W_MSGCNT	00000016	D
MMG$GL_SPTBASE	********	X	02	UCB$L_EMB	00000078	D	UCB$W_MSGMAX	00000014	D
PR$_IPL	= 00000012	D		UCB$L_FIRST	00000014	D	UCB$W_NT_CHAN	0000007C	D
PR$_TBIS	= 0000003A	D		UCB$L_FPC	0000000C	D	UCB$W_OFFSET	0000008A	D
PTE$S_PFN	= 00000015	D		UCB$L_FQBL	00000004	D	UCB$W_REFC	00000050	D
PTE$V_PFN	= 00000000	D		UCB$L_FQFL	00000000	D	UCB$W_SIZE	00000008	D
RVT$L_UCBLST	= 00000044	D		UCB$L_FR3	00000010	D	UCB$W_SRCADDR	0000001A	D
SIZ...	= 00000001	D		UCB$L_FR4	00000014	D	UCB$W_STS	00000058	D
SS$_NORMAL	********	X	02	UCB$L_IOQBL	00000044	D	UCB$W_TT_DESIZE	000000A5	D
UCB$B_AMOD	00000053	D		UCB$L_IOQFL	00000040	D	UCB$W_UNIT	00000048	D
UCB$B_CEX	00000077	D		UCB$L_IRP	0000004C	D	UCB$W_VPROT	0000001A	D
UCB$B_CM1	0000004A	D		UCB$L_LINK	0000002C	D	VA$S_BYTE	= 00000009	D
UCB$B_CM2	0000004B	D		UCB$L_LOGADR	00000064	D	VA$V_BYTE	= 00000000	D

```
VA$V_SYSTEM     = 0000001F     D
WCB$B_ACCESS      0000000B     D
WCB$B_TYPE        0000000A     D
WCB$C_LENGTH      00000024     D
WCB$C_MAP         00000024     D
WCB$K_LENGTH      00000024     D
WCB$K_MAP         00000024     D
WCB$L_FCB         00000018     D
WCB$L_LBN         00000002     D
WCB$L_ORGUCB      00000010     D
WCB$L_P1_LBN      00000026     D
WCB$L_P2_LBN      0000002C     D
WCB$L_PID         0000000C     D
WCB$L_PREVLBN     FFFFFFFC     D
WCB$L_RVT         0000001C     D
WCB$L_STVBN       00000020     D
WCB$L_WLBL        00000004     D
WCB$L_WLFL        00000000     D
WCB$W_ACON        00000014     D
WCB$W_COUNT       00000000     D
WCB$W_NMAP        00000016     D
WCB$W_P1_COUNT    00C00024     D
WCB$W_P2_COUNT    0000002A     D
WCB$W_PREVCOUNT   FFFFFFFA     D
WCB$W_REFCNT      0000000E     D
WCB$W_SIZE        00000008     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000175 (373.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
BUG$CHECK	00000000-XR		276 (1)
DEV$V_SQD	=00000005		#-205 (1)
DYN$C_WCB	=00000012		#-197 (1)
EXE$FINISHIO	00000000-XR		#-363 (1)
IOC$APPLYECC	00000000-R	95 (1)	
IOC$CVTLOGPHY	00000074-R	155 (1)	
IOC$MAPVBLK	0000009B-R	196 (1)	
IOC$PTETOPFN	00000000-XR		#-112 (1)
IOC$SENSEDISK	0000016A-R	360 (1)	
IOC$UPDATRANSP	0000012E-R	306 (1)	
IRP$L_MEDIA	00000034		#-156 (1) #-160 (1) #-163 (1) #-164 (1)
MMG$GL_SPTBASE	00000000-XR		115 (1)
PR$_IPL	=00000012		#-199 (1) #-274 (1)
PR$_TBIS	=0000003A		#-118 (1)
PTE$S_PFN	=00000015		#-115 (1)
PTE$V_PFN	=00000000		#-114 (1)
RVT$L_UCBLST	=00000044		#-262 (1)
SS$_NORMAL	00000000-XR		#-272 (1) #-362 (1)
UCB$B_FIPL	0000000B		#-199 (1)
UCB$L_DEVCHAR	00000034		205 (1)
UCB$L_DEVDEPEND	0000003C		#-158 (1) #-161 (1) #-317 (1) #-319 (1) #-321 (1)
			#-323 (1)
UCB$L_MAXBLOCK	00000084		#-361 (1)
UCB$L_SVAPTE	00000068		#-110 (1) #-309 (1)
UCB$L_SVPN	00000064		#-113 (1)
UCB$M_ECC	=00000001		#-126 (1)
UCB$V_NOCNVRT	=00000002		#-157 (1)
UCB$W_BCNT	0000006E		#-105 (1) #-307 (1)
UCB$W_BOFF	0000006C		#-107 (1) #-119 (1)
UCB$W_DA	0000008C		#-311 (1) #-317 (1) #-319 (1) #-320 (1) #-321 (1)
			#-323 (1)
UCB$W_DC	0000008E		#-324 (1)
UCB$W_DEVSTS	0000005A		#-126 (1) 157 (1)
UCB$W_EC1	00000090		#-97 (1)
UCB$W_EC2	00000092		#-102 (1)
VA$S_BYTE	=00000009		#-109 (1) #-116 (1) #-120 (1)
VA$V_BYTE	=00000000		#-120 (1)
VA$V_SYSTEM	=0000001F		#-117 (1)
WCB$B_TYPE	0000000A		#-197 (1)
WCB$L_RVT	0000001C		#-261 (1)
WCB$L_STVBN	00000020		203 (1)
WCB$W_NMAP	00000016		#-201 (1)

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$DEFINI         1           50      (1)     50      (1)     51      (1)     52      (1)     53      (1)     54      (1)
                                            55      (1)     56      (1)     57      (1)     58      (1)     59      (1)
$DEVDEF         1           50      (1)     50      (1)
$DYNDEF         2           51      (1)     51      (1)
$IPLDEF         1           52      (1)     52      (1)
$IRPDEF         4           53      (1)     53      (1)
$PRDEF          5           54      (1)     54      (1)
$PTEDEF         3           55      (1)     55      (1)
$RVTDEF         1           56      (1)     56      (1)
$UCBDEF         10          57      (1)     57      (1)
$VADEF          1           58      (1)     58      (1)
$WCBDEF         3           59      (1)     59      (1)
BUG_CHECK       1           276     (1)     276     (1)
DSBINT          1           199     (1)     199     (1)
ENBINT          1           274     (1)     274     (1)
INVALID         1           118     (1)     118     (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDB	0080	311 (1)
ADDL	00C0	101 (1) 108 (1) 236 (1) 237 (1) 269 (1) 309 (1)
ADDL3	00C1	119 (1)
ASHL	0078	103 (1) 109 (1) 116 (1) 122 (1) 268 (1) 308 (1)
BBS	00E0	157 (1) 205 (1)
BBSS	00E2	117 (1)
BEQL	0013	202 (1) 260 (1)
BGEQ	0018	270 (1)
BGEQU	001E	106 (1)
BGTRU	001A	318 (1) 322 (1)
BICB3	008B	99 (1)
BISW	00A8	126 (1)
BLEQ	0015	247 (1)
BLSS	0019	111 (1)
BLSSU	001F	206 (1) 215 (1)
BNEQ	0012	198 (1) 250 (1)
BRB	0011	219 (1) 227 (1) 325 (1)
BRW	0031	363 (1)
BSBW	0030	112 (1)
CLRL	00D4	159 (1) 218 (1) 271 (1)
CMPB	0091	197 (1) 317 (1) 321 (1)
CMPL	00D1	249 (1)
CMPW	00B1	105 (1)
DECL	00D7	246 (1) 98 (1)
DIVL	00C6	100 (1) 310 (1)
EDIV	007B	160 (1) 162 (1)
EXTZV	00EF	258 (1) 259 (1)
INCB	0096	320 (1)
INCL	00D6	123 (1)
INCW	00B6	324 (1)
INSV	00F0	114 (1) 120 (1)
JSB	0016	276 (1)
MFPR	00DB	199 (1)
MOVAL	00DE	203 (1)
MOVB	0090	163 (1)
MOVL	00D0	104 (1) 110 (1) 156 (1) 226 (1) 261 (1) 262 (1) 272 (1) 361 (1)
MOVW	00B0	164 (1)
MOVZBL	009A	158 (1) 161 (1)
MOVZWL	003C	102 (1) 107 (1) 201 (1) 213 (1) 248 (1) 362 (1) 97 (1)
MTPR	00DA	118 (1) 199 (1) 274 (1)
MULL3	00C5	113 (1)
POPR	00BA	125 (1) 273 (1)
PUSHL	00DD	235 (1)
PUSHR	00BB	200 (1) 96 (1)
RSB	0005	127 (1) 165 (1) 275 (1) 326 (1)
SOBGTR	00F5	124 (1) 217 (1)
SUBB	0082	319 (1) 323 (1)
SUBL	00C2	204 (1) 214 (1) 251 (1)
SUBW	00A2	307 (1)

```
TSTL          00D5      216    (1)
XORB          008C      121    (1)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ASCIC          276     (1)
.END            365     (1)
.ENDC           118     (1)     199     (1)     274     (1)
.ENDM           118     (1)     199     (1)     274     (1)     276     (1)     50      (1)     51      (1)     52      (1)     53      (1)
                54      (1)     55      (1)     56      (1)     57      (1)     58      (1)     59      (1)
.IDENT          5       (1)
.IF             118     (1)     199     (1)     274     (1)
.IFF            118     (1)     199     (1)     274     (1)
.LIST           48      (1)     6       (1)     61      (1)     62      (1)
.NLIST          46      (1)     47      (1)     60      (1)     7       (1)
.NOCROSS        50      (1)     51      (1)     52      (1)     53      (1)     54      (1)     55      (1)     56      (1)     57      (1)
                58      (1)     59      (1)
.PAGE           128     (1)     166     (1)     277     (1)     327     (1)     63      (1)     66      (1)
.PSECT          64      (1)
.RESTORE        50      (1)     51      (1)     52      (1)     53      (1)     54      (1)     55      (1)     56      (1)     57      (1)
                58      (1)     59      (1)
.SAVE           50      (1)     51      (1)     52      (1)     53      (1)     54      (1)     55      (1)     56      (1)     57      (1)
                58      (1)     59      (1)
.SBTTL          129     (1)     167     (1)     278     (1)     328     (1)     67      (1)
.TITLE          4       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	30	#-101	(1)	#-102	(1)	#-103	(1)	#-121	(1)	#-122	(1)	#-156	(1)
		#-160	(1)	#-162	(1)	#-164	(1)	#-204	(1)	#-214	(1)	#-218	(1)
		#-235	(1)	#-237	(1)	258	(1)	259	(1)	#-262	(1)	#-272	(1)
		#-307	(1)	#-308	(1)	#-309	(1)	#-310	(1)	#-311	(1)	#-362	(1)
R1	26	#-103	(1)	#-107	(1)	#-108	(1)	#-109	(1)	#-110	(1)	#-113	(1)
		115	(1)	#-116	(1)	117	(1)	#-118	(1)	120	(1)	#-121	(1)
		#-159	(1)	#-162	(1)	#-163	(1)	#-200	(1)	#-213	(1)	#-214	(1)
		#-236	(1)	#-237	(1)	#-249	(1)	#-258	(1)	#-361	(1)	#-99	(1)
R2	20	#-100	(1)	#-101	(1)	#-105	(1)	#-108	(1)	#-119	(1)	#-123	(1)
		#-158	(1)	#-160	(1)	#-161	(1)	#-162	(1)	#-197	(1)	#-201	(1)
		203	(1)	#-261	(1)	#-262	(1)	#-273	(1)	#-97	(1)	#-98	(1)
		#-99	(1)										
R3	17	#-110	(1)	#-114	(1)	#-119	(1)	#-120	(1)	#-125	(1)	#-156	(1)
		#-160	(1)	#-163	(1)	#-164	(1)	#-200	(1)	#-201	(1)	#-217	(1)
		#-246	(1)	#-248	(1)	#-251	(1)	#-273	(1)	#-96	(1)		
R4	14	#-104	(1)	#-124	(1)	#-125	(1)	#-200	(1)	#-203	(1)	#-204	(1)
		#-213	(1)	#-216	(1)	#-226	(1)	#-236	(1)	#-248	(1)	#-249	(1)
		#-273	(1)	#-96	(1)								
R5	29	#-102	(1)	#-105	(1)	#-107	(1)	#-110	(1)	#-113	(1)	#-119	(1)
		#-126	(1)	157	(1)	#-158	(1)	#-161	(1)	#-199	(1)	205	(1)
		#-226	(1)	#-262	(1)	#-307	(1)	#-309	(1)	#-311	(1)	#-317	(1)
		#-319	(1)	#-320	(1)	#-321	(1)	#-323	(1)	#-324	(1)	#-361	(1)
		#-97	(1)										
SP	8	#-199	(1)	#-251	(1)	#-268	(1)	#-269	(1)	#-271	(1)	#-274	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.00	00:00:00.20
Command processing	879	00:00:00.22	00:00:00.72
Pass 1	574	00:00:02.22	00:00:03.53
Symbol table sort	0	00:00:00.15	00:00:00.14
Pass 2	9	00:00:00.61	00:00:01.05
Symbol table output	0	00:00:00.03	00:00:00.06
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	5	00:00:00.10	00:00:00.25
Assembler run totals	1491	00:00:03.33	00:00:05.95

The working set limit was 2900 pages.
138352 bytes (271 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 608 non-local and 18 local symbols.
365 source lines were read in Pass 1, producing 44 object records in Pass 2.
88 pages of virtual memory were used to define 23 macros.

```
                                    +----------------------------+
                                    ! Macro library statistics !
                                    +----------------------------+

Macro library name                   Macros defined
------------------                   --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5          6
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6            1
SYS$COMMON:[SYSLIB]LIB.MLB;2                       6
SYS$COMMON:[SYSLIB]STARLET.MLB;2                   6
TOTALS (all libraries)                            19

902 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:IOSRAM.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents


```
(1)      685        Libraries and Macros
(1)      764        Equated Symbols
(1)      822        Data Psect Declarations
(1)      845        Work Psect Declarations
(1)     1020        GLOBAL PSECT DECLARATIONS FOR VSDP                          ;G:28
(1)     1043        Data Psect Declarations
(5)     1065        DIAGNOSTIC SUPERVISOR BOOTSTRAP ROUTINE.
(5)     1324        USEP KERNEL ROUTINE
(5)     1570        COMMON KERNEL REFRESH ENTRY POINT.
(6)     1798        INIT_CONTEXT    Initialize process context
(6)     1924        CONTROL-C AST HANDLER
(6)     1971        KEYBOARD CHECK ROUTINE
(6)     2174        GET_TIME        Convert TODR to current system time         ;G:16
(6)     2284        DSX$CNTRLC      Program enable for Control-C intercept
(6)     2334        DSV$EXIT, Process EXIT command
(6)     2409        EXIT$HANDLR     Process EXIT handler
(6)     2475        DSR$Check_MenuTest_Off_Set Routine
(6)     2555        DSR$Check_AutoTest_Off_Set Routine
(8)     2629        DSX$GETTERM  Get User Terminal Characteristics              [87]
```

```
ZZ-ENSAA-10.10  KERNEL *** KERNEL Main control routine          3-MAR-1988       Fiche 13 Frame F8        Sequence 2568
KERNEL                        *** KERNEL Main control routine     18-NOV-1987 15:15:02  VAX/VMS Macro V04-00        Page   1
10.9-36                                                           18-NOV-1987 15:05:32  KERNEL.MAR;2                      (1)

                                1 ; DEC/CMS REPLACEMENT HISTORY, Element KERNEL.MAR
                                2 ; *38   11-NOV-1987 08:59:20 GEARIN "ADD CODE TO remember SCORPIO CONSOLE LOAD PATH"
                                3 ;  37B1 27-AUG-1987 10:05:53 MEIER "CHANGES TO DS"
                                4 ;  37A1 27-AUG-1987 07:40:04 GEARIN "ALLOW VDS TO remember CONSOLE LOAD DISK # FOR S
                                5 ; *37   10-AUG-1987 17:15:34 BARANSKI "Merging Gen:35A1 and Gen:36 to Gen:37"
                                6 ;  35A1 10-AUG-1987 15:05:36 GEARIN "SAVED R3 UPON BOOTING IF BOOTED ON 8200 [R3 HAS
                                7 ; *36   27-JUL-1987 08:46:52 MEIER "CHANGED A SYMBOL"
                                8 ; *35    1-MAY-1987 11:01:06 MEIER "REMOVED REFERENCES TO LLL PRODUCT NAME"
                                9 ; *34   29-APR-1987 16:02:29 MARTIN "MODIFIED FOR PSTAR"
                               10 ; *33   26-JAN-1987 15:21:13 MARTIN "FIXED THE VDS HEADER"
                               11 ; *32   26-JAN-1987 14:04:30 MEIER "LLL FIX DONE"
                               12 ; *31   24-DEC-1986 10:01:16 MUSE "RECOMPILE"
                               13 ; *30   22-DEC-1986 13:51:05 MARTIN "NO CHANGES"
                               14 ; *29   10-DEC-1986 15:50:16 MARTIN "LINED UP DISPLAY OF LEGAL STATEMENT"
                               15 ; *28    5-DEC-1986 10:01:00 BARANSKI "Delete MUSE changes, Add SILVER changes."
                               16 ; *27   22-OCT-1986 09:14:57 MUSE "CAPTURE THE AN RPB VALUE ON BOOT FOR PABTDRIVR"
                               17 ; *26   20-JUN-1986 12:55:18 SALEM "DONE"
                               18 ; *25   19-JUN-1986 12:48:50 SALEM "<no reason given>"
                               19 ; *24   18-JUN-1986 14:51:27 SALEM "DONE"
                               20 ; *23   18-JUN-1986 14:07:13 SALEM "DONE"
                               21 ; *22    5-JUN-1986 14:17:07 BAGGETT "<no reason given>"
                               22 ; *21    5-JUN-1986 13:49:45 BAGGETT "<no reason given>"
                               23 ; *20    5-JUN-1986 13:41:21 BAGGETT "<no reason given>"
                               24 ; *19    5-JUN-1986 12:20:31 ANDELLA "<no reason given>"
                               25 ; *18    2-JUN-1986 10:15:04 BAGGETT "FINISHED"
                               26 ; *17   28-MAY-1986 13:30:49 BERGAZZI "/TIME CHANGE"
                               27 ; *16   20-MAY-1986 10:25:24 BERGAZZI "/TIME"
                               28 ; *15   14-MAY-1986 16:07:40 SALEM "DONE WITH SYSTYPE CHECK"
                               29 ; *14   28-APR-1986 14:59:47 SHELHAMER "<no reason given>"
                               30 ; *13   25-APR-1986 13:34:50 SHELHAMER "<no reason given>"
                               31 ; *12   23-APR-1986 11:41:34 SHELHAMER "<no reason given>"
                               32 ; *11   23-APR-1986 10:51:07 SHELHAMER "<no reason given>"
                               33 ; *10   18-APR-1986 15:41:18 SHELHAMER "<no reason given>"
                               34 ; *9    16-APR-1986 14:53:41 SHELHAMER "<no reason given>"
                               35 ; *8    15-APR-1986 12:46:20 SHELHAMER "<no reason given>"
                               36 ; *7    10-APR-1986 16:57:49 SHELHAMER "<no reason given>"
                               37 ; *6    20-MAR-1986 09:39:41 MUSE "<no reason given>"
                               38 ; *5    10-MAR-1986 17:08:45 SHELHAMER "<no reason given>"
                               39 ; *4    10-MAR-1986 16:50:33 SHELHAMER "<no reason given>"
                               40 ; *3    10-MAR-1986 10:56:17 SHELHAMER "<no reason given>"
                               41 ; *2    12-FEB-1986 09:22:13 BERGAZZI "REMOVAL OF LIB$GET_FOREIGN CALL"
                               42 ; *1     6-FEB-1986 15:19:09 DS ""
                               43 ; DEC/CMS REPLACEMENT HISTORY, Element KERNEL.MAR
                               44         .Title  KERNEL  *** KERNEL Main control routine
                               45         .Ident  /10.9-36/                                                    ;G:35A
                               46         .NoShow Conditionals
                               47
                               48 ;++
                               49 ; Copyright (c) 1977, 1983, 1984, 1985, 1986, 1987                        ;G:33
                               50 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                               51 ;
                               52 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                               53 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                               54 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                               55 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                               56 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                               57 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
```

```
                                0000     58 ; REMAIN IN DEC.
                                0000     59 ;
                                0000     60 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                                0000     61 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                                0000     62 ; CORPORATION.
                                0000     63 ;
                                0000     64 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                                0000     65 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                0000     66 ;--
```

```
0000     68 ;**
0000     69 ; Facility:
0000     70 ;
0000     71 ;       VAX Diagnostic Supervisor.
0000     72 ;
0000     73 ; Abstract:
0000     74 ;
0000     75 ;       This module contains the main control routines for the VAX
0000     76 ;       Diagnostic Supervisor.
0000     77 ;
0000     78 ; Environment:
0000     79 ;
0000     80 ;       Not applicable.
0000     81 ;
0000     82 ; Author:
0000     83 ;
0000     84 ;       Ken Chapman
0000     85 ;
0000     86 ; Date:
0000     87 ;
0000     88 ;       26-Oct-77
0000     89 ;
0000     90 ; Version:
0000     91 ;
0000     92 ;       Version 01.
0000     93 ;
```

```
0000     95 ;
0000     96 ; Modified By:
0000     97 ;       01      - Nick Howgate, 15-nov-1977, Version 02.
0000     98 ;               APT INTERFACE UPDATES.
0000     99 ;
0000    100 ;       02      - Tom Soutter, 21-nov-1977, Version 03.
0000    101 ;               DSPR #14  DEFAULT FLAG SETTING.
0000    102 ;
0000    103 ;       03      - Ken Chapman, 06-dec-1977, Version 04.
0000    104 ;               REMOVED MEMORY MAPS.
0000    105 ;
0000    106 ;       04      - Ken Chapman, 06-dec-1977, Version 04.
0000    107 ;               DSPR #3  MODIFIED ^C BREAK TO CLI.
0000    108 ;
0000    109 ;       05      - Nick Howgate, 19-dec-1977, Version 05.
0000    110 ;               MADE AX_BUFF, USTKPTR, SSTKPTR, ESTKPTR GLOBAL FOR MEMMGT.
0000    111 ;
0000    112 ;       06      - Nick Howgate, 19-dec-1977, Version 05.
0000    113 ;               MADE ALL STACKS 2 PAGES.
0000    114 ;
0000    115 ;       07      - Nick Howgate, 30-Dec-1977, Version 06 (ESSAA-3.06).
0000    116 ;               DELETE SYMBOL DS$GL_PASSES
0000    117 ;
0000    118 ;       08      - Nick Howgate, 06-Feb-1978, Version 07 (ESSAA-4.00).
0000    119 ;               INCLUDE QIO SUPPORT
0000    120 ;
0000    121 ;               - Tom Soutter, 03-Mar-1978
0000    122 ;       09      ADDED SOFTWARE PCB FOR AST HANDLING
0000    123 ;       10      FIXED SOME BUGS IN PCB SETUP CODE
0000    124 ;       11      DEFINED DIAGNOSTIC PID, SET UP SAME IN SOFTWARE PCB
0000    125 ;
0000    126 ;       12      - Ken Chapman, 23-Mar-1978, Version 08 (ESSAA-4.01).
0000    127 ;               ADDED BUFFER INIT CODE.
0000    128 ;               REMOVED MOVC5 AND LDPCTX INSTRUCTIONS.
0000    129 ;
0000    130 ;               - Tom Soutter, 30-Mar-1978
0000    131 ;       13      ADDED CODE TO INITIALIZE THE SYSTEM TIMER
0000    132 ;       14      REMOVED STACK BUFFER PAGES, REDUCED USER STACK TO ONE
0000    133 ;               PAGE AND OVER-MAPPED THE SUPERVISOR & EXECUTIVE STACKS
0000    134 ;               ONTO THE USER STACK
0000    135 ;
0000    136 ;       15      - Nick Howgate, 18-May-1978, Version 09 (ESSAA-4.02).
0000    137 ;               PLACE IPL LEVEL AT 1F AT SYSTEM START UP
0000    138 ;               LOWER IPL LEVEL ONLY AFTER SOFTWARE INITIALIZATION
0000    139 ;
0000    140 ;       16      - Nick Howgate, 09-Jun-1978, Version 10 (ESSAA-4.03).
0000    141 ;               FIX APT ^C PROBLEM
0000    142 ;
0000    143 ;       17      - Nick Howgate, 14-Jun-1978
0000    144 ;               CLEAR EVENTS FLAGS AT SUPERVISOR START
```

```
0000   146 ;                    - Roger Riggs, 10-Aug-1978
0000   147 ;          18        Revised basic program loop
0000   148 ;          19        Fix for changed position of year in DS$GT_NEWYEAR
0000   149 ;                    Added separate allocations for USER, EXEC, and SUPER stacks
0000   150 ;          20        Added call to INI$PTABLE to initialize P-table list at
0000   151 ;                    supervisor start.
0000   152 ;          21        Added code to Auto configure/ATTACH the default load device
0000   153 ;                    when loaded with DIAGBOOT.
0000   154 ;          22        Added process exit handler to execute use cleanup and
0000   155 ;                    deallocate device allocated by select.
0000   156 ;
0000   157 ;          23        - Roger Riggs, 31-Aug-1979
0000   158 ;                    Added initial VDT breakpoint and saved VMB boot flags
0000   159 ;
0000   160 ;          24        - Roger Riggs, 21-Jan-1980, Version 5.2
0000   161 ;                    Added code to always prompt APT on start, also copy
0000   162 ;                    processor type code to EXE$GB_CPUTYPE
0000   163 ;
0000   164 ;          25        - David R. Butenhof, 06-feb-1980
0000   165 ;                    Routine INIT_CONTEXT should force AST delivery enable.
0000   166 ;
0000   167 ;          26        - John Ciukaj, 25-Jan-1980, Version 5.2
0000   168 ;                    Added location BOOT$L_DEVTYP to be loaded from RPB$B_DEVTYP.
0000   169 ;
0000   170 ;          27        - Roger Riggs, 12-Mar-1980
0000   171 ;                    Clear DS$GB_MM_ENB on START 10000, as well as memory managment
0000   172 ;
0000   173 ;                    - Roger Riggs, 24-Mar-1980
0000   174 ;          28        Added code to verify that system identification register
0000   175 ;                    agrees with current supervisor.
0000   176 ;          29        Added code to type message and exit if SYS$INPUT is not
0000   177 ;                    a terminal or a mailbox.  Fix for SPR#11-28511.
0000   178 ;                    Later fix will use RMS for SYS$INPUT I/O in usermode.
0000   179 ;
0000   180 ;                    - Dave Butenhof, -Mar-1980
0000   181 ;          30        Add RMS setup in usermode, to allow supervisor execution
0000   182 ;                    via VMS command file/batch.
0000   183 ;          31        Fix CTL$GL_CCBBASE label define to point to a pointer
0000   184 ;                    longword.  And fix order of arguments to PRINT on SID
0000   185 ;                    mis-match.
0000   186 ;
0000   187 ;          32        - Roger Riggs, 21-May-1980, Version 5.4
0000   188 ;                    Changed order of stacks to allow for memory protection
0000   189 ;                    when MM is on
```

```
                                0000   191 ;
                                0000   192 ;       33      Moved IOC$GL_PSFL/BL to Entry
                                0000   193 ;       34      Added JIB
                                0000   194 ;
                                0000   195 ;       35      - Dave Butenhof, 28-aug-1980, Version 6.0
                                0000   196 ;               Add call to RMS$INIT in standalone initialization to
                                0000   197 ;               clear RMS file table.
                                0000   198 ;
                                0000   199 ;               - Roger Riggs, 08-Sept-1980, Version 6.0
                                0000   200 ;       36      Added code to set and restore on exit the default directory
                                0000   201 ;               and disk.
                                0000   202 ;       37      Move translation of load device name to DEF$Q_DEV on
                                0000   203 ;               usemode entry
```

```
                        0000   205 ;
                        0000   206 ;       38      - Roger Riggs, 17-Oct-1980
                        0000   207 ;               Made reference to EXE$GQ_SYSTIME L^ offset.
                        0000   208 ;
                        0000   209 ;               - Dave Butenhof, 17-oct-1980, version 6.1
                        0000   210 ;       39      Add new argument to DSX$CNTRLC; DISABL--if low bit is set,
                        0000   211 ;               disables control C handling, if clear, enables it.  Also
                        0000   212 ;               KB_CHECK ignores control C flag if handling is disabled.
                        0000   213 ;               Also, if user control C routine return failure code, the
                        0000   214 ;               Supervisor will proceed to handle the control C normally
                        0000   215 ;               itself.
                        0000   216 ;       40      Add label BEGIN_BLISS (same as BEGIN) for BLISS routines to
                        0000   217 ;               use after INIT_CONTEXT, since BEGIN is reserved word.
```

```
                        0000    219 ;
                        0000    220 ;       41      - Dave Butenhof, 18-feb-1981, version 6.3
                        0000    221 ;               Delete use of SEP psect.
                        0000    222 ;
                        0000    223 ;       42      - Dave Butenhof, 16-mar-1981, version 6.3
                        0000    224 ;               Fix ^C handling; move flag clears to make user status=0
                        0000    225 ;               return work under APT.
                        0000    226 ;
                        0000    227 ;       43      - Dave Butenhof, 23-mar-1981, version 6.3
                        0000    228 ;               Expand Exec/Supervisor stack to 3 pages each.
```

```
0000    230 ;       44      - Jack Stansbury, 21-Oct-1981, Version 6.5
0000    231 ;               Added calls to QA$MAIN for the QA enhancement.
0000    232 ;               Also added .LIBRARY statements for $DS and $DIAG.
0000    233 ;               Also changed one of the error messages to lower case.
0000    234 ;               Also fixed some truncation errors.
0000    235 ;
0000    236 ;       45      - Jack Stansbury, 10-Nov-1981, Version 6.5
0000    237 ;               Changed ALL W^ to L^.
0000    238 ;
0000    239 ;       46      - Dave Butenhof, 12-Nov-1981, version 6.5
0000    240 ;               Add DS$GB_TYPECODE for 'SET BINARY' function.
0000    241 ;
0000    242 ;       47      - Dave Butenhof, 13-Nov-1981, version 6.5
0000    243 ;               Implement EXIT command (DSV$EXIT).  It will do an
0000    244 ;               $EXIT in user mode, or a HALT in standalone; after
0000    245 ;               doing program cleanup, etc.
0000    246 ;
0000    247 ;       48      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000    248 ;               Commented out several calls to QA_MAIN because it appears as
0000    249 ;               though they will not be needed for the 6.5 DS version.
```

```
                        0000    251 ;
                        0000    252 ;       49      - Dave Butenhof, 14-Dec-1981, version 6.6
                        0000    253 ;               Modify startup to use $GETSYI call in usermode, to get
                        0000    254 ;               the actual processor SID into DSA$GL_SID.
                        0000    255 ;
                        0000    256 ;       50      - Dave Butenhof, 29-Dec-1981, version 6.6
                        0000    257 ;               Add locations to store error counts for HARD, SOFT, DEVICE,
                        0000    258 ;               SYSTEM, and SOFTWARE (ERRSUP) errors.
                        0000    259 ;
                        0000    260 ;       51      - Jack Stansbury, 10-Feb-1982, Version 6.6X
                        0000    261 ;               Added another Control-C handler for the DS to use. This is for
                        0000    262 ;               QA to trap control-c's, but can be used by any other
                        0000    263 ;               DS-integrated facility. See the KB_Check routine for the code.
                        0000    264 ;
                        0000    265 ;       52      - Dave Butenhof, 09-Mar-1982, version 6.6
                        0000    266 ;               Add support for rooted systems.  When booted from DIAGBOOT,
                        0000    267 ;               startup code will do 'SET LOAD' command on the filespec in the
                        0000    268 ;               RPB$T_FILE field.
```

```
                        0000    270 ;       53      - Dave Butenhof, 26-Mar-1982, version 6.7
                        0000    271 ;               Fix rooted system directory support...redefine SYS$DISK when
                        0000    272 ;               translating SYS$SYSROOT.  SYS$SYSROOT:[SYSO.SYSMAINT] don't
                        0000    273 ;               Added SYS$LIBRARY:LIB.
                        0000    274 ;
                        0000    275 ;       54      Marion Baggett, 31-Mar-1982, Version 6.7
                        0000    276 ;               Added SYS$LIBRARY:LIB
                        0000    277 ;
                        0000    278 ;       55      Marion Baggett, 2-Apr-1982, Version 6.7
                        0000    279 ;               Added DS$GL_PREPERR_COUNT field to count device preparation     ;G:2
                        0000    280 ;               errors.                                                         ;G:2
                        0000    281 ;
                        0000    282 ;       56      Marion Baggett, 5-Apr-1982, Version 6.7
                        0000    283 ;               Changed $DS_PRINT to $PRINT macros.
                        0000    284 ;
                        0000    285 ;       57      Marion Baggett, 7-Apr-1982, Version 6.7
                        0000    286 ;               Added $DS_TYPEDEF to define type codes.
                        0000    287 ;
                        0000    288 ;       58      Dave Butenhof, 08-Apr-1982, version 6.7
                        0000    289 ;               Fix truncation error.
                        0000    290 ;
                        0000    291 ;       59      Jack Stansbury, 9-Apr-1982, Version 6.7
                        0000    292 ;               Added $CRD Library statement. Added support for the
                        0000    293 ;               Customer Runnable Diagnostic package.
                        0000    294 ;
                        0000    295 ;       60      Dave Butenhof, 13-Apr-1982, version 6.7
                        0000    296 ;               Add setup of DS$GB_INHIBIT_NAMING, to enable/disable
                        0000    297 ;               generic name enforcement. (Set in APT mode).
```

```
                        0000    299 ;
                        0000    300 ;       61      Dave Butenhof, 17-May-1982, version 6.8
                        0000    301 ;               Fix size of Kernel stack (make it larger to minimize trouble).
                        0000    302 ;
                        0000    303 ;       62      Dave Butenhof, 21-May-1982, version 6.8
                        0000    304 ;               Automatically attach console device at startup, to make
                        0000    305 ;               RMS and DIRECTORY a bit simpler (no special case for device
                        0000    306 ;               without Ptable).
                        0000    307 ;
                        0000    308 ;       63      Jack Stansbury, 7-June-1982, Version 6.8
                        0000    309 ;               Added code in DSV$Exit routine that will not allow the user
                        0000    310 ;               to Continue (console command) once a HALT instruction has been
                        0000    311 ;               executed - only if CRD-AutoTest bits are set in the R5 passed
                        0000    312 ;               to the DS.
                        0000    313 ;
                        0000    314 ;       64      Jack Stansbury, 9-Jun-1982, Version 6.8
                        0000    315 ;               Added code in the DSV$Exit routine that will allow output of
                        0000    316 ;               pertinent CRD variables when a CRD-Internal error occurs. This
                        0000    317 ;               is indicated by CRD changing the byte in its own dispatch vector
                        0000    318 ;               area (DS$GA_CRD_Starting_Address + 1) to be one more than the ;G:2
                        0000    319 ;               absolute value of its dispatch vectors + 2 byte. That is, CRD ;G:2
                        0000    320 ;               will increment the "positive version number" in its dispatch  ;G:2
                        0000    321 ;               vector area by one when it detects a internal error. CRD will ;G:2
                        0000    322 ;               then put the values of a select number of pertinent variables ;G:2
                        0000    323 ;               into its own dispatch vector area. DSV$Exit will print these  ;G:2
                        0000    324 ;               out when the user types a console Continue command.           ;G:2
                        0000    325 ;
                        0000    326 ;       65      Marion Baggett, 9-June-1982, Version 6.8
                        0000    327 ;               Increase the size of the interupt stack to 4 pages so the
                        0000    328 ;               Vax Architecture exerciser will run.
                        0000    329 ;
                        0000    330 ;       66      Richard Brown, 15-June-82, Version 6.8
                        0000    331 ;               Addition of IMAGE_HEADER, which is a buffer to hold the
                        0000    332 ;               image header of a diagnostic file, if the file was linked
                        0000    333 ;               with a header.  The header is used by the debugger.
                        0000    334 ;
                        0000    335 ;       67      Jack Stansbury, 28-June-1982, Version 6.8
                        0000    336 ;               Disabled ^C's in stand-alone initialization code so that CRD  ;G:2
                        0000    337 ;               will have a chance to set up a ^C handler. This takes care of ;G:2
                        0000    338 ;               the problem of ^C's being typed while DIAGBOOT is running, and ;G:2
                        0000    339 ;               then having the DS see the ^C before CRD is loaded in (thus   ;G:2
                        0000    340 ;               causing the DS> prompt to appear without the DS_Start message ;G:2
                        0000    341 ;               being printed). ^C's are re-enabled in the Begin_0 code.      ;G:2
```

```
                        0000    343 ;
                        0000    344 ;       68      Richard Brown, 16-July-82, Version 6.9
                        0000    345 ;               Addition of code in INIT_CONTEXT to see if the T-bit is set
                        0000    346 ;               in the PSL and if so to set it in the new PSL.  This is so
                        0000    347 ;               that if the debugger has set the T-bit, it will not be cleared
                        0000    348 ;               out when the context is initialized.
                        0000    349 ;
                        0000    350 ;       69      Jack Stansbury, 24-July-1982, Version 6.9
                        0000    351 ;               Added two new routine that return a 1 in R0 iff the correct   ;G:2
                        0000    352 ;               bits are set in Boot$L_R5 for CRD_AutoTest and for            ;G:2
                        0000    353 ;               CRD_MenuTest. Used these routines in the Begin0 part.         ;G:2
                        0000    354 ;
                        0000    355 ;       70      Jack Stansbury, 27-July-1982, Version 6.9
                        0000    356 ;               Added another ^C handler for the VDS. This allows CRD Menu Test ;G:2
                        0000    357 ;               to let the User handle a ^C if the user has established a     ;G:2
                        0000    358 ;               handler routine; but at the same time, if this user routine   ;G:2
                        0000    359 ;               returns failure meaning they did not want to handle it, let the ;G:2
                        0000    360 ;               "third" handler (VDS's second handler) have it. Leave DS$Cli  ;G:2
                        0000    361 ;               as the last chance handler.                                   ;G:2
                        0000    362 ;
                        0000    363 ;       71      Jack Stansbury, 8-September-1982, Version 6.9
                        0000    364 ;               Made some changes to the DSV$Exit routine for CRD Menu Test.  ;G:2
                        0000    365 ;               Also made a few changes to the routines mentioned in 69 above ;G:2
                        0000    366 ;               - they were looking at the wrong bits.                        ;G:2
                        0000    367 ;
                        0000    368 ;       72      Jack Stansbury, 15-September-1982, Version 6.9
                        0000    369 ;               Cleared out the three ^C handler addresses in the VDS         ;G:2
                        0000    370 ;               stand-alone initialization code.                              ;G:2
                        0000    371 ;
                        0000    372 ;       73      Jack Stansbury, September 15, 1982, Version 6.9
                        0000    373 ;               Made the DS$L_UserCntrlC longword global, for use by the CLI  ;G:2
                        0000    374 ;               routine.                                                      ;G:2
                        0000    375 ;
                        0000    376 ;       74      Jack Stansbury, September 23, 1982, Version 6.9
                        0000    377 ;               Added a print statement to be printed if the CRD               ;G:2
                        0000    378 ;               initialization fails.                                         ;G:2
                        0000    379 ;
                        0000    380 ;       75      Jack Stansbury, September 24, 1982, Version 6.9
                        0000    381 ;               Added code to check to see if both CRD bits were set in the R5 ;G:2
                        0000    382 ;               passed to the VDS from Diagboot. If so, print an error message ;G:2
                        0000    383 ;               and exit.                                                     ;G:2
                        0000    384 ;
                        0000    385 ;       76      Jack Stansbury, September 28, 1982, Version 6.9
                        0000    386 ;               Added code to take care of the following: Start up the VDS.   ;G:2
                        0000    387 ;               Type CRD. Hit ^P while CRD is running - get back to ">>>"     ;G:2
                        0000    388 ;               console prompt. Type S 10000 - get a DS> prompt. Type CRD     ;G:2
                        0000    389 ;               again,but the CRD image is already loaded. This was causing a ;G:2
                        0000    390 ;               halt at PC = 00000001. So, the solution is to call the        ;G:2
                        0000    391 ;               Unload_CRD routine when the bits are set in the DSA flags     ;G:2
                        0000    392 ;               longword when the Supervisor is being inited (it is           ;G:2
                        0000    393 ;               normally inited without any of the CRD bits being set).
                        0000    394 ;
                        0000    395 ;       77      Bob Bergazzi    7-Oct-82        Version 6.9
                        0000    396 ;               Changed the way system service vectors are handled; after wecopy ;G:
                        0000    397 ;               the VMS system service vectors to 10200 (in order to get the
                        0000    398 ;               correct word mask), modify each entry point to immediately JMP
                        0000    399 ;               to the corresponding entry point in the real system service   ;G:36
```

```
                        400 ;               table at 80000xxx. This solves the problem of code in the system
                        401 ;               service vectors which branches around outside of the three pages
                        402 ;               which we were copying before.
                        403 ;
                        404 ;       78      Bob Bergazzi    24-Nov-82       Version 6.10
                        405 ;               Added a call to load the PCS on startup (DS$_LOADPCS_S)
                        406 ;
                        407 ;       79      Jack Stansbury, 4-Mar-83, Version 6.11
                        408 ;               Changed the code in the DSR$Check routines (for CRD) - they must
                        409 ;               check the DSA bits while online, not the Boot$L_R5 longword.
                        410 ;
                        411 ;       80      John Ciukaj,    2-Apr.-1983     Version 6.11
                        412 ;               - Changed code that referenced CRD bits in DSA Flags.
                        413 ;                 The bits were redefined to distinguish between
                        414 ;                 online and offline.
                        415 ;               - Changed name of DSR$Check_... routines to distinguish
                        416 ;                 between Online and Offline.
                        417 ;
                        418 ;       81      Bob Bergazzi    8-Apr-1983      Version 6.11
                        419 ;               Made on-line mode start up with terminal width set to 80.
                        420 ;
                        421 ;       82      Bob Bergazzi    25-April-1983   Version 6.11
                        422 ;               During boot of ESSAA, we must determine if this is 780 or a
                        423 ;               785, and if it is a 785, then we need to change
                        424 ;               the clock overhead factor (DS$GK_USOVR) in the WAITUS routine.
                        425 ;               This is done by referencing the symbol DSX$WAITUS_USOVR which
                        426 ;               addresses the beginning of the instruction which uses the
                        427 ;               overhead factor, and modifying the instruction. This is done
                        428 ;               in order to avoid adding more instructions to the DSX$WAITUS
                        429 ;               routine which would change the value of the DS$GK_USOVR for
                        430 ;               existing processors.
                        431 ;
                        432 ;       83      Bob Bergazzi    25-April-1983   Version 6.11
                        433 ;               Increased SCB size to 4 pages for VENUS. Related changes in
                        434 ;               SCB module.
                        435 ;
                        436 ;       84      John Ciukaj     29-Apr-83       version 6.11
                        437 ;                                   13-May-83
                        438 ;               DSR$Check.. routines: check both DSA Flags and RPB
                        439 ;               regardless of Standalone or Online.
                        440 ;
                        441 ;       85      Bob Bergazzi    May 16,1983     Version 6.11
                        442 ;               Changed the order of the .LIB statements.
                        443 ;
                        444 ;       86      M.Baggett       May 16, 1983    Version 6.11
                        445 ;               Get RPB$L_BOOTR2 field from RPB.
                        446 ;
                        447 ;       87      Richard Brown   June 22, 1983   Version 6.12
                        448 ;               Added code to fetch user terminal characteristics
                        449 ;               and store them in $DSGL_TERMCHAR so the new service
                        450 ;               $DS_GETTERM can pass them to a diagnostic.
                        451 ;
                        452 ;               Added new system service $DS_GETTERM, which returns
                        453 ;               the characteristics of the user terminal.
                        454 ;
                        455 ;       88      Bob Bergazzi    Sep 22, 1983    Version 6.13
                        456 ;               Added a $CANTIM if a ^C is typed to cancel a $SETIMR
```

```
0000   457 ;                which may have been done at program start for the
0000   458 ;                /TIME switch.
0000   459 ;                Commented out for version 6.13 and 6.14, and 7.0
0000   460 ;
0000   461 ;        89      Bob Bergazzi    11-11-83        Version 6.14
0000   462 ;                Clear the DS$M_CTRLC and DS$M_CTRLO flags before we call
0000   463 ;                a control-c handler. This fixes the problem of terminal
0000   464 ;                output in a ^C handler being suppressed in S/A.
0000   465 ;
0000   466 ;        90      Bob Bergazzi    12-28-83        Version 6.14
0000   467 ;                The definition of JIB$L_ARB in the JIB was removed from VMS V4,
0000   468 ;                so define it to its old offset value (the space in the JIB is
0000   469 ;                still there).
0000   470 ;
0000   471 ;        91      Bob Bergazzi    2-14-84         Version 6.14
0000   472 ;                Moved the code which builds ptables for the console load device
0000   473 ;                and the boot device into the CONFIG module, called by
0000   474 ;                JSB INI$LOAD_DEVICE. This was done so SET MEM could call it.  ;G:2
0000   475 ;
0000   476 ;        92      Richard Brown   May 14, 1984    Version 7.0
0000   477 ;                Added display of legal statement on startup.
0000   478 ;
0000   479 ;        93      M. Baggett      May-17-1984     Version 7.0
0000   480 ;                Allow word length unit to be save in RPB.
0000   481 ;
0000   482 ;        94      Bob Bergazzi    May 18, 1984    Version 7.0
0000   483 ;                Made legal display not print out when running under APT.
0000   484 ;
0000   485 ;        95      Bob Bergazzi    June 19, 1984   Version 7.0
0000   486 ;                Also, do not display legal message when CRD is running.
0000   487 ;
0000   488 ;        96      R.E. Muse       July 13, 1984   Version 7.0
0000   489 ;                Search order bit mask added to $TRNLOG system service.
0000   490 ;                SYS$DISK is now search for in the system logical name
0000   491 ;                table exclusively.
0000   492 ;
0000   493 ;        97      R.E. Muse       19 July 1984    version 7.0
0000   494 ;                logical translation for SYS$SYSROOT added and SYS$DISK
0000   495 ;                removed. Now default directory is "[sys0.sysmaint].
0000   496 ;
0000   497 ;        98      Bob Bergazzi    July 27, 1984   Version 7.1
0000   498 ;                Moved $DS_LOADPCS to after memory is initialized and ptables
0000   499 ;                are built and RMS is initialized, so that we can load
0000   500 ;                patches from a file.
0000   501 ;
0000   502 ;        99      Bob Bergazzi    Aug. 30, 1984   Version 7.1
0000   503 ;                Added call to HDW_CLOCK_SET to do cpu specific clock setup.
0000   504 ;
0000   505 ;        100     Ron Parker      November 1,1984 Version 7.1
0000   506 ;                Added support for CRD_Menutest_Online
0000   507 ;
0000   508 ;        101     RE Muse         Novemeber 7,1984 version 7.1
0000   509 ;                removed PSECT _LAST and placed in each machine specific
0000   510 ;                ONLYnnn module. This was done in particular to accomodate
0000   511 ;                the large SCB page allocation for NAUTILUS, otherwise all
0000   512 ;                supervisor's would carry more SCB page allocation than
0000   513 ;                really needed. ie. nautilus needs 37 pages for the scb.
```

```
0000    514 ;
0000    515 ;       102     M. Baggett      9-Nov-1984      Version 7.1
0000    516 ;               Improved the coding of CASE statement.
0000    517 ;
0000    518 ;       103     Jim Shelhamer   Feb 11, 1985    Version 8.1
0000    519 ;               Added code for Attached Processor EXIT instruction.
0000    520 ;
0000    521 ;       104     Bob Bergazzi    Feb. 13, 1985   Version 8.1
0000    522 ;               Changed location of call to HDW_CLOCK_SET, since it no
0000    523 ;               longer changes DS$GT_NEWYEAR+7.
0000    524 ;
0000    525 ;       105     Ted Salem       Feb. 25, 1985   Version 8.1
0000    526 ;               Added code for simulation of the NI - i.e. may be temporary.
0000    527 ;
0000    528 ;       106     Bob Bergazzi    Feb. 26, 1985   Version 8.1
0000    529 ;               Fix edit [88] (/TIME).
0000    530 ;
0000    531 ;       107     Ted Salem       Feb. 26, 1985   Version 8.1
0000    532 ;               Made NDS$OPEN, RECEIVE_POLL weak symbols.
0000    533 ;
0000    534 ;       108     Bob Bergazzi    Mar. 4, 1985    Version 8.1
0000    535 ;               Commented out edit 106.
0000    536 ;
0000    537 ;       109     M. Baggett      6-Mar-1985      Version 8.1
0000    538 ;               Expanded legal notice to show copyright year.
0000    539 ;
0000    540 ;       110     Ted Salem       March 14, 1985  Version 8.2
0000    541 ;               Added code to clear the MP$R_ACTIVE_SERVICES bitvector.
0000    542 ;
0000    543 ;       111     Domenic Andella 22-May-1985     Version 8.2
0000    544 ;               Added multi-processing support to KB_CHECK.
0000    545 ;
0000    546 ;       112     Domenic Andella 11-Jun-1985     Version 9.0
0000    547 ;               Temporarily add mp symbol MP$GR_BOOTED_PROCESSORS
0000    548 ;               until all mp modules are included in link. Add
0000    549 ;               WEAK symbol definitions.
0000    550 ;
0000    551 ;       113     Bob Bergazzi    11-June-1985    Version 9.0
0000    552 ;               Set up EXE$GL_TENUSEC and EXE$GL_UBDELAY in the
0000    553 ;               BOOTDRIVR module for use by the TIMEDWAIT macro
0000    554 ;               which is in some bootdrivrs.
0000    555 ;
0000    556 ;       114     M. Baggett      19-June-1985    Version 9.0
0000    557 ;               Fix directory if booting from ULTRIX disk.
0000    558 ;
0000    559 ;       115     Jim Shelhamer   24-JUN-1985     Version 9.0
0000    560 ;               Made ds$gt_copyear gloabl, it is referenced by APBOOT8200.
0000    561 ;
0000    562 ;       117     Jim Shelhamer   2-Aug-1985      Version 9.0
0000    563 ;               Took out JSB to VRSETWIDTH.
0000    564 ;               This was messng up characteristics of MBX for VSDP.
0000    565 ;
0000    566 ;       118     M. Baggett      13-Aug-1985     Version 9.0
0000    567 ;               Added field for RPB bootr1.
0000    568 ;
0000    569 ;       119     Bob Bergazzi    30-Sep-1985     Version 9.0
0000    570 ;               T/M for SCORPIO should merely do a normal supervisor
```

```
0000    571 ;                   boot, not invoking CRD MENU mode.
0000    572 ;
0000    573 ;       120         Jim Shelhamer   28-Oct-1985     Version 9.1
0000    574 ;                   During boot of EDSAA, we must determine if this is 8600 or a
0000    575 ;                   8650, and if it is a 8650, then we need to change
0000    576 ;                   the clock overhead factor (DS$GK_USOVR) in the WAITUS routine.
0000    577 ;                   This is done by referencing the symbol DSX$WAITUS_USOVR which
0000    578 ;                   addresses the beginning of the instruction which uses the
0000    579 ;                   overhead factor, and modifying the instruction. This is done
0000    580 ;                   in order to avoid adding more instructions to the DSX$WAITUS
0000    581 ;                   routine which would change the value of the DS$GK_USOVR for
0000    582 ;                   existing processors.
0000    583 ;
0000    584 ;       121         Jim Baranski     1-Nov-1985     Version 9.1
0000    585 ;                   Un Comment Out START/TIME.
0000    586 ;
0000    587 ;       122         Bob Bergazzi    Jan 17, 1986    Version 9.1
0000    588 ;                   For a NAUTILUS supervisor, change the default directory
0000    589 ;                   in the online mode from USERFILES to SYSMAINT.
0000    590 ;
0000    591 ;       122         Bob Bergazzi    Jan 17, 1986    Version 9.1
0000    592 ;                   For a NAUTILUS supervisor, change the default directory
0000    593 ;                   in the online mode from USERFILES to SYSMAINT.
0000    594 ;
0000    595 ;       123         Ted Salem       30-Jan-1986     Version 9.2
0000    596 ;                   Moved code in change #110 from Init_common to Begin0.
0000    597 ;
0000    598 ;       124         Bob Bergazzi    12-Feb-1986     Version 10.0                ;G:2
0000    599 ;                   Remove call to LIB$GET_FOREIGN. This was put in as part of  ;G:2
0000    600 ;                   an online CRD enhancement, which is no longer needed.       ;G:2
0000    601 ;                                                                               ;G:3
0000    602 ;       125         Jim Shelhamer   5-Mar-1986      Version 10.0                ;G:3
0000    603 ;                   On EXIT from AP, request that PP set up RXCD intr vector    ;G:3
0000    604 ;                   to handle console prompt which will occur on HALT inst.     ;G:3
0000    605 ;                   Added $PR8SSDEF.                                            ;G:3
0000    606 ;                                                                               ;G:7
0000    607 ;       126         Jim Shelhamer   10-Apr-1986     Version 10.0                ;G:7
0000    608 ;                   Added call to ONLY_CPU, which will be located in            ;G:7
0000    609 ;                   each ONLYxxx module. This will provide an area for          ;G:7
0000    610 ;                   CPU specific start up code. Also, ignore RXCD intrs.        ;G:13
0000    611 ;                                                                               ;G:15
0000    612 ;       127         Ted Salem       4-May-1986      Version 10.0                ;G:15
0000    613 ;                   Added structures and functionality to read the system       ;G:15
0000    614 ;                   type for MicroVax2 based systems.  This is used to          ;G:36
0000    615 ;                   distinguish an Aurora from an Andromeda system.             ;G:15
0000    616 ;                   Also, added code to read Module Id for Andromeda.           ;G:15
0000    617 ;                                                                               ;G:16
0000    618 ;       128         Bob Bergazzi    15-May-1986     Version 10.1                ;G:16
0000    619 ;                   Added support to KB_CHECK for /TIME.                        ;G:16
0000    620 ;                                                                               ;G:17
0000    621 ;       129         Bob Bergazzi    28-May-1986     Version 10.1                ;G:17
0000    622 ;                   Changed /TIME under VMS from elapsed time to execution time.;G:17
0000    623 ;                                                                               ;G:18
0000    624 ;       130         M. Baggett      2-Jun-1986      Version 10.1                ;G:18
0000    625 ;                   Added support for RT-VAX.                                   ;G:18
0000    626 ;                                                                               ;G:19
0000    627 ;       131         Domenic Andella 2-June-1986     Version 10.2                ;G:19
```

```
0000  ;                 Save, restore R0, and R1 at several places                  ;G:19
0000  ;                 in routine KB_CHECK.                                        ;G:36
0000  ;                                                                             ;G:20
0000  ;        132      M. Baggett      5-Jun-1986      Version 10.2               ;G:20
0000  ;                 Allow AURORA to have to SID values.                         ;G:20
0000  ;                                                                             ;G:25
0000  ;        133      Ted Salem       19-June-1986    Version 10.2               ;G:25
0000  ;                 Changed all access of Andromeda IOspace to                  ;G:36
0000  ;                 longword access only. (FBI is longword access).             ;G:25
0000  ;                                                                             ;G:28
0000  ;        134      Doug Silver     20-Sep-1986     Version 10.3               ;G:28
0000  ;                 Made PASSNO,TESTNO,SUBTESTNO global variables               ;G:28
0000  ;                 for VSDP in GLBL_SECT_AREA.                                 ;G:28
0000  ;                                                                             ;G:28
0000  ;        135      Doug Silver     20-Nov-1986     Version 10.3               ;G:28
0000  ;                 Made default load path the logical name                     ;G:28
0000  ;                 "SYS$MAINTENANCE:".                                         ;G:36
0000  ;                                                                             ;G:36
0000  ;        136      Stephen Meier   5-Jan-1987      Version 10.5               ;G:32
0000  ;                 Add LLL specific code to take the pointer                   ;G:34
0000  ;                 to the console communications area out of the               ;G:36
0000  ;                 RPB and save it in a global location.                       ;G:32
0000  ;                                                                             ;G:33
0000  ;        137      Ingrid Martin   26-JAN-1987     Version 10.5               ;G:33
0000  ;                 Fixed the VDS header using spaces rather than               ;G:33
0000  ;                 tabs.                                                       ;G:33
0000  ;                                                                             ;G:34
0000  ;        138      ingrid martin   31-mar-1987     version 10.7               ;G:34
0000  ;                 updates for rrr machine:  get_time                          ;G:34
0000  ;                                                                             ;G:34
0000  ;        139      David Gearin    1-Apr-1987      Version 10.7               ;G:34
0000  ;                 Add code to permit booting and usage of csa2                ;G:34
0000  ;                 (RX50 console drive) on a Scorpio.                          ;G:34
0000  ;                                                                             ;G:34
0000  ;        140      Ingrid Martin   7-April-1987    Version 10.7               ;G:34
0000  ;                 Fixed a line that got jumbled.                              ;G:34
0000  ;                                                                             ;G:35
0000  ;        Edit #'s by generation # from now on....                             ;G:35
0000  ;                                                                             ;G:36
0000  ;        35       Stephen Meier   1-may-87        Version10.7                ;G:35
0000  ;                 Removed references to the LLL product name.                 ;G:35
0000  ;                                                                             ;G:36
0000  ;        36       Stephen Meier   27-Jul-87       Version 10.10              ;G:36
0000  ;                 Removed an LLL specific dummy symbol and replaced it with   ;G:36
0000  ;                 the real value.                                             ;G:36
0000  ;                                                                             ;G:36
0000  ;                                                                             ;G:37
0000  ;        36       David Gearin    3-Aug-87        Version 10.10              ;G:35A
0000  ;                 Continuation of Edit #139. Saved contents of R3 upon        ;G:35A
0000  ;                 entering the main code.  R3 contains the # of disk drive    ;G:35A
0000  ;                 used if Scorpio was booted via the console.                 ;G:35A
0000  ;                                                                             ;G:38
0000  ;                 David Gearin    11-Nov-1987     Version 10.10              ;G:38
0000  ;                 Add code to "remember" which drive # scorpio booted         ;G:38
0000  ;                 from, if booted from console rx50.                          ;G:38
0000  ;;--                                                                          ;G:25
```

```
                                0000   685          .Subtitle       Libraries and Macros
                                0000   686 ;
                                0000   687 ; INCLUDE FILES:
                                0000   688 ;
                                0000   689          .Library        /SYS$LIBRARY:LIB/        ;                                [54]  ;G:32
                                0000   690          .Library        /$CRD/                   ; CRD-specific defns             [59]  ;G:2
                                0000   691          .Library        /$DS/                    ;                                [44]  ;G:2
                                0000   692          .Library        /$DIAG/                  ;                                [44]  ;G:2
                                0000   693
                                0000   694 ;
                                0000   695 ; WEAK EXTERNALS
                                0000   696 ;
                                0000   697
                                0000   698          .WEAK   XDELBPT,XDELTBIT,NDS$OPEN,RECEIVE_POLL  ;                       [107] ;G:16
                                0000   699          .WEAK   MP$_PRIMARY_CPU_CHECK   ; Checks and acts on mp flags.          [112] ;G:16
                                0000   700          .WEAK   MP$_RESUME_ATTACHED_PROCESSORS; Resumes attached processors[112] ;G:
                                0000   701          .WEAK   MP$R_ACTIVE_SERVICES    ; Bitvector used in interlock                  ;G:16
                                0000   702                                          ; routine                        [110] ;G:16
                                0000   703          .WEAK   MP$R_INTERLOCK_END      ; Endmark of all interlocking                  ;G:16
                                0000   704                                          ; structures                     [110] ;G:16
                                0000   705          .WEAK   ap$i_prompt_check,ap$b_prompt_buffer_count,ap$b_prompt_count  ;G:5
                                0000   706          .WEAK   PPA$GB_MID, PPA$GB_CPU  ; Module # and ID # for Andromeda [127] ;G:1
                                0000   707          .WEAK   TXCS$V_RDY,TODR_STORE,TODR_BYTE_CNT     ; 8800 specific [128] ;G:16
                                0000   708                                                                                 ;G:36
                                0000   709 ; EXTERNALS
                                0000   710 ;
                                0000   711              .EXTRN  RECEIVE_POLL,NDS$OPEN   ; TO OPEN SIMULATION FILES FOR NI [105]
                                0000   712              ;.EXTRN MP$GR_BOOTED_PROCESSORS ; Bitvector to indicate attached processors
                                0000   713              .EXTRN  MP$_PRIMARY_CPU_CHECK; Checks and acts on mp flags.            [111
                                0000   714              .EXTRN  MP$_RESUME_ATTACHED_PROCESSORS; Resumes attached processors    [111
                                0000   715 ;
                                0000   716 ; EQUATED SYMBOLS:
                                0000   717 ;
                                0000   718                                                  ; *** System type for RT-VAX [130] ;
00000010                        0000   719 PR$_SID_TYPRTUV2 ==^X10                        ; ***** TEMPORARY UNTIL DEFINED [130
00000002                        0000   720 PR$_SYS_TYP800 == 2                           ; System Type for Aurora         [127] ;G:15
00000003                        0000   721 PR$_SYS_TYP900 == 3                           ; System Type for Andromeda      [127] ;G:15
0000000A                        0000   722 PR$_SYS_TYP_LLL == 10                        ; System type for LLL            [136] ;G:35
00000011                        0000   723 PR$_SID_TYP8RR  =  17                        ; System type for RRR            [138] ;G:34
00000006                        0000   724 PR$_SID_TYP8NN  =  6                         ; System type for nautilus       [138] ;G:34
20040004                        0000   725 SYS_TYPE = ^x20040004                       ; System Type in IOSPACE   [127][133] ;G:26
3FFFE400                        0000   726 PPA_SPID = ^x3FFFE400                       ; Secondary ID for Andromeda     [127] ;G:23
00000000                        0000   727 RPB$V_MENUTEST = 0                          ; This is missing from the
00000001                        0000   728 RPB$M_MENUTEST = 1@RPB$V_MENUTEST           ; SYS$LIBRARY:LIB.MLB library            [119
                                0000   729
00000001                        0000   730 SS$_NORMAL=1                               ; SUCCESS
                                0000   731              $DS_DSDEF                   ; SHOULD PRECEDE "IOC$K_DIAGPID"
00660000                        0000   732 IOC$K_DIAGPID == DS$K_SUBSYS@16           ; DIAGNOSTIC PROCESS I.D.
                                0000   733              $ARBDEF                                                            ;G:17
                                0000                    .SAVE   LOCAL_BLOCK
                                0000                    .RESTORE
                                0000   734              $CCBDEF
                                0000                    .SAVE   LOCAL_BLOCK
                                0000                    .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
00000004                        0000                            .BLKL   1
                                0004                    .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
00000008                        0004                            .BLKL   1
```

```
                     0008                .IIF    NB,CCB$B_STS,   CCB$B_STS:
00000009             0008                        .BLKB   1
                     0009                .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
0000000A             0009                        .BLKB   1
                     000A                .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
0000000C             000A                        .BLKW   1
                     000C                .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
00000010             000C                        .BLKL   1
                     0010                .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                     0010                .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                     0000                .RESTORE
                     0000    735         $DEVDEF                         ; Define device type codes              ;G:17
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0010                .=0
                     0000                .RESTORE
                     0000    736         $DIBDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0010                .=0
                     0000                .RESTORE
                     0000    737         $JIBDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0010                .=0
                     0000                .RESTORE
00000000             0000    738                 JIB$L_ARB = 0           ; This was removed from VMS v4.0 [90] ;G:16
                     0000    739         $JPIDEF                         ; For $GETJPIW service          [129] ;G:17
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0010                .=0
                     0000                .RESTORE
                     0000    740         $PCBDEF                                                         ;G:17
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0010                .=0
                     0000                .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004             0000                        .BLKL   1
                     0004                .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008             0004                        .BLKL   1
                     0008                .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A             0008                        .BLKW   1
                     000A                .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B             000A                        .BLKB   1
                     000B                .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C             000B                        .BLKB   1
                     000C                .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D             000C                        .BLKB   1
                     000D                .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E             000D                        .BLKB   1
                     000E                .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010             000E                        .BLKW   1
                     0010                .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014             0010                        .BLKL   1
                     0014                .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018             0014                        .BLKL   1
                     0018                .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
```

```
0000001C  0018                          .BLKL   1
          001C          .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020  001C                          .BLKL   1
          0020          .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024  0020                          .BLKL   1
          0024          .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028  0024                          .BLKL   1
          0028          .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C  0028                          .BLKL   1
          002C          .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E  002C                          .BLKW   1
          002E          .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                          .BLKB   1
          002F          .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                          .BLKB   1
          0030          .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                          .BLKW   1
          0032          .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                          .BLKW   1
          0034          .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                          .BLKW   1
          0036          .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                          .BLKW   1
          0038          .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                          .BLKW   1
          003A          .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                          .BLKW   1
          003C          .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                          .BLKW   1
          003E          .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                          .BLKW   1
          0040          .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                          .BLKW   1
          0042          .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                          .BLKW   1
          0044          .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                          .BLKB   8
          004C          .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
          004C          .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                          .BLKL   1
          0050          .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                          .BLKL   1
          0054          .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                          .BLKL   1
          0058          .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                          .BLKL   1
          005C          .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                          .BLKL   1
          0060          .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                          .BLKL   1
          0064          .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                          .BLKL   1
          0068          .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                          .BLKB   16
          0078          .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
```

```
00000084  007C                                  .BLKQ   1
          0084                  .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
          0088                  .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                                  .BLKW   1
          008A                  .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                                  .BLKW   1
          008C                  .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
          008C                  .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
          0000                  .RESTORE
          0000    741           $PHDDEF                                                         ;G:17
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  008C                  .=0
          0000                  .RESTORE
          0000    742           $PR780DEF                       ; Define 780 specific IPRs      [126] ;G:17
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  008C                  .=0
          0000                  .RESTORE
          0000    743           $PR8SSDEF                       ; Define Scorpio specifics      [125] ;G:17
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  008C                  .=0
          0000                  .RESTORE
          0000    744           $PRDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  008C                  .=0
          0000                  .RESTORE
          0000    745           $PSLDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  008C                  .=0
          0000                  .RESTORE
          0000    746           $RPBDEF                         ; Define offsets in RPB                 ;G:17
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  008C                  .=0
          0000                  .IIF    NB,RPB$L_BASE,  RPB$L_BASE:
00000004  0000                  .IIF    NB,.BLKL,       .BLKL
          0004                  .IIF    NB,RPB$L_RESTART,       RPB$L_RESTART:
00000008  0004                  .IIF    NB,.BLKL,       .BLKL
          0008                  .IIF    NB,RPB$L_CHKSUM,        RPB$L_CHKSUM:
0000000C  0008                  .IIF    NB,.BLKL,       .BLKL
          000C                  .IIF    NB,RPB$L_RSTRTFLG,      RPB$L_RSTRTFLG:
00000010  000C                  .IIF    NB,.BLKL,       .BLKL
          0010                  .IIF    NB,RPB$L_HALTPC,        RPB$L_HALTPC:
00000014  0010                  .IIF    NB,.BLKL,       .BLKL
          0014                  .IIF    NB,RPB$L_HALTPSL,       RPB$L_HALTPSL:
00000018  0014                  .IIF    NB,.BLKL,       .BLKL
          0018                  .IIF    NB,RPB$L_HALTCODE,      RPB$L_HALTCODE:
0000001C  0018                  .IIF    NB,.BLKL,       .BLKL
          001C                  .IIF    NB,RPB$L_BOOTR0,        RPB$L_BOOTR0:
          001C                  .IIF    NB,RPB$B_R0DEVTYP,      RPB$B_R0DEVTYP:
0000001D  001C                  .IIF    NB,.BLKB,       .BLKB
```

```
0000001E  001D                                  .BLKB   1
          001E                  .IIF    NB,RPB$W_R0UBVEC,       RPB$W_R0UBVEC:
00000020  001E                  .IIF    NB,.BLKW,       .BLKW
          0020                  .IIF    NB,RPB$L_BOOTR1,        RPB$L_BOOTR1:
00000024  0020                  .IIF    NB,.BLKL,       .BLKL
          0024                  .IIF    NB,RPB$L_BOOTR2,        RPB$L_BOOTR2:
00000028  0024                  .IIF    NB,.BLKL,       .BLKL
          0028                  .IIF    NB,RPB$L_BOOTR3,        RPB$L_BOOTR3:
0000002C  0028                  .IIF    NB,.BLKL,       .BLKL
          002C                  .IIF    NB,RPB$L_BOOTR4,        RPB$L_BOOTR4:
00000030  002C                  .IIF    NB,.BLKL,       .BLKL
          0030                  .IIF    NB,RPB$L_BOOTR5,        RPB$L_BOOTR5:
00000034  0030                  .IIF    NB,.BLKL,       .BLKL
          0034                  .IIF    NB,RPB$L_IOVEC, RPB$L_IOVEC:
00000038  0034                  .IIF    NB,.BLKL,       .BLKL
          0038                  .IIF    NB,RPB$L_IOVECSZ,       RPB$L_IOVECSZ:
0000003C  0038                  .IIF    NB,.BLKL,       .BLKL
          003C                  .IIF    NB,RPB$L_FILLBN,        RPB$L_FILLBN:
00000040  003C                  .IIF    NB,.BLKL,       .BLKL
          0040                  .IIF    NB,RPB$L_FILSIZ,        RPB$L_FILSIZ:
00000044  0040                  .IIF    NB,.BLKL,       .BLKL
          0044                  .IIF    NB,RPB$Q_PFNMAP,        RPB$Q_PFNMAP:
0000004C  0044                  .IIF    NB,.BLKQ,       .BLKQ
          004C                  .IIF    NB,RPB$L_PFNCNT,        RPB$L_PFNCNT:
00000050  004C                  .IIF    NB,.BLKL,       .BLKL
          0050                  .IIF    NB,RPB$L_SVASPT,        RPB$L_SVASPT:
00000054  0050                  .IIF    NB,.BLKL,       .BLKL
          0054                  .IIF    NB,RPB$L_CSRPHY,        RPB$L_CSRPHY:
00000058  0054                  .IIF    NB,.BLKL,       .BLKL
          0058                  .IIF    NB,RPB$L_CSRVIR,        RPB$L_CSRVIR:
0000005C  0058                  .IIF    NB,.BLKL,       .BLKL
          005C                  .IIF    NB,RPB$L_ADPPHY,        RPB$L_ADPPHY:
00000060  005C                  .IIF    NB,.BLKL,       .BLKL
          0060                  .IIF    NB,RPB$L_ADPVIR,        RPB$L_ADPVIR:
00000064  0060                  .IIF    NB,.BLKL,       .BLKL
          0064                  .IIF    NB,RPB$W_UNIT,  RPB$W_UNIT:
00000066  0064                  .IIF    NB,.BLKW,       .BLKW
          0066                  .IIF    NB,RPB$B_DEVTYP,        RPB$B_DEVTYP:
00000067  0066                  .IIF    NB,.BLKB,       .BLKB
          0067                  .IIF    NB,RPB$B_SLAVE, RPB$B_SLAVE:
00000068  0067                  .IIF    NB,.BLKB,       .BLKB
          0068                  .IIF    NB,RPB$T_FILE,  RPB$T_FILE:
00000090  0068                  .IIF    NB,.BLKB,       .BLKB   40
          0090                  .IIF    NB,RPB$B_CONFREG,       RPB$B_CONFREG:
000000A0  0090                  .IIF    NB,.BLKB,       .BLKB   16
          00A0                  .IIF    NB,RPB$B_HDRPGCNT,      RPB$B_HDRPGCNT:
000000A1  00A0                  .IIF    NB,.BLKB,       .BLKB
          00A1                  .IIF    NB,RPB$B_BOOTNDT,       RPB$B_BOOTNDT:
000000A2  00A1                  .IIF    NB,.BLKB,       .BLKB
000000A4  00A2                                  .BLKW   1
          00A4                  .IIF    NB,RPB$L_ISP,   RPB$L_ISP:
000000A8  00A4                  .IIF    NB,.BLKL,       .BLKL
          00A8                  .IIF    NB,RPB$L_PCBB,  RPB$L_PCBB:
000000AC  00A8                  .IIF    NB,.BLKL,       .BLKL
          00AC                  .IIF    NB,RPB$L_SBR,   RPB$L_SBR:
000000B0  00AC                  .IIF    NB,.BLKL,       .BLKL
          00B0                  .IIF    NB,RPB$L_SCBB,  RPB$L_SCBB:
```

```
                    000000B4  00B0          .IIF    NB,.BLKL,          .BLKL
                              00B4          .IIF    NB,RPB$L_SISR,     RPB$L_SISR:
                    000000B8  00B4          .IIF    NB,.BLKL,          .BLKL
                              00B8          .IIF    NB,RPB$L_SLR,      RPB$L_SLR:
                    000000BC  00B8          .IIF    NB,.BLKL,          .BLKL
                              00BC          .IIF    NB,RPB$L_MEMDSC,           RPB$L_MEMDSC:
                    000000FC  00BC          .IIF    NB,.BLKL,          .BLKL   16
                              00FC          .IIF    NB,RPB$L_BUGCHK,           RPB$L_BUGCHK:
                    00000100  00FC          .IIF    NB,.BLKL,          .BLKL
```

```
                     0100                .IIF    NB,RPB$B_WAIT,  RPB$B_WAIT:
00000104             0100                .IIF    NB,.BLKB,       .BLKB   4
                     0104                .IIF    NB,RPB$C_LENGTH,        RPB$C_LENGTH:
                     0104                .IIF    NB,RPB$K_LENGTH,        RPB$K_LENGTH:
                     0000                .RESTORE
                     0000    747         $SECDEF                         ; Define section codes          [134] ;G:28
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0104                .=0
                     0000                .RESTORE
                     0000    748         $SFDEF                          ;                               [128] ;G:16
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0104                .=0
                     0000                .RESTORE
                     0000    749         $SHRDEF                         ; Define shareable status codes
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0104                .=0
                     0000                .RESTORE
                     0000    750         $SYIDEF                         ; Define SYI codes              [49]  ;G:16
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0104                .=0
                     0000                .RESTORE
                     0000    751         $DS_SCBDEF                                                            ;G:35A
                     0000    752         CLIDEF                          ; Define CLI data base offsets  [52]  ;G:16
                     0000    753         CMKDEF
                     0000    754         $DS_DSADEF                                                            ;G:17
                     0000    755         DSFDEF
                     0000    756         ENVDEF
```

```
                    0000   757        DSQA                                  ; QA routine name constants     [44]  ;G:16
                    0000   758        $DS_HPODEF                            ; P-table offsets                     ;G:17
                    0000              .SAVE   LOCAL_BLOCK
                    0000              .PSECT  $ABS$,ABS
          00000000  FE70              .=0
                    0000              .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
          00000008  0000              .IIF    NB,.BLKQ,       .BLKQ   1
                    0008              .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
          0000000A  0008              .IIF    NB,.BLKW,       .BLKW   1
                    000A              .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
          0000000B  000A              .IIF    NB,.BLKB,       .BLKB   1
                    000B              .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
          0000000C  000B              .IIF    NB,.BLKB,       .BLKB   1
                    000C              .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
          00000018  000C              .IIF    NB,.BLKB,       .BLKB   12
                    0018              .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
          0000001C  0018              .IIF    NB,.BLKL,       .BLKL   1
                    001C              .IIF    NB,HP$A_DVA,    HP$A_DVA:
          00000020  001C              .IIF    NB,.BLKL,       .BLKL   1
                    0020              .IIF    NB,HP$A_LINK,   HP$A_LINK:
          00000024  0020              .IIF    NB,.BLKL,       .BLKL   1
                    0024              .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
          00000026  0024              .IIF    NB,.BLKW,       .BLKW   1
                    0026              .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
          00000032  0026              .IIF    NB,.BLKB,       .BLKB   12
                    0032              .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                    0000              .RESTORE
                    0000   759        $DS_TYPEDEF                           ; Define type codes             [57]  ;G:16
                    0000   760        $CRD_Literals                         ; Define the CRD literals       [59]  ;G:16
                    0000   761        APDEF                                 ;                               [103] ;G:16
                    0000   762                                                                                      ;G:36
```

```
                 0000          .Subtitle       Equated Symbols
                 0000                                                                                  ;G:36
                 0000
00003000         0000  DSA$M_DFLTFLGS  == DSA$M_OPER ! DSA$M_PROMPT
                 0000
00000200         0000  DS$A_PRGBGN     == ^X200
0000F800         0000  DS$K_PRGSIZ     == ^XFA00 - DS$A_PRGBGN                                         ;G:36
                 0000
0000000D         0000          CR      = 13
0000000A         0000          LF      = 10
                 0000
                 0000  ; INITIAL FREE-CORE REQUEST SIZE
                 0000
00000098         0000  UCB$SIZE        =       ^X98
00000018         0000  DDB$SIZE        =       ^X18
0000002C         0000  CRB$SIZE        =       ^X2C
00000020         0000  IDB$SIZE        =       ^X20
00000050         0000  IRP$SIZE        =       ^X50
00000010         0000  CCB$SIZE        =       ^X10                                                    ;G:36
00000050         0000  VCB$SIZE        =       ^X50
00000030         0000  WCB$SIZE        =       ^X30
00000030         0000  FCB$SIZE        =       ^X30
                 0000
00000040         0000  SGN$C_IRPCNT    ==      64
                 0000  SGN$C_NPAGEDYN  ==      <UCB$SIZE*24>           -; 24 UCB'S
                 0000                          +<DDB$SIZE*5>           -;  5 DDB'S
                 0000                          +<CRB$SIZE*3>           -;  3 CRB'S
                 0000                          +<IDB$SIZE*5>           -;  5 IDB'S
                 0000                          +<IRP$SIZE*SGN$C_IRPCNT>-; IRP'S
                 0000                          +<CCB$SIZE*16>          -; 16 CCB'S
                 0000                          +<VCB$SIZE*2>           -;  2 VCB'S
                 0000                          +<WCB$SIZE*2>           -;  2 WCB'S
0000263C         0000                          +<FCB$SIZE*2>            ;  2 FCB'S
00000800         0000  DB_SIZ  = ^X800                          ; Size of DBDRIVER
00000800         0000  DL_SIZ  = ^X800                          ; Size of DLDRIVER
00000A00         0000  DM_SIZ  = ^XA00                          ; Size of DMDRIVER
00000800         0000  DR_SIZ  = ^X800                          ; Size of DRDRIVER
00001000         0000  TM_SIZ  = ^X1000                         ; Size of TMDRIVER
00000600         0000  XF_SIZ  = ^X600                          ; Size of XFDRIVER
                 0000
00005E3C         0000  QIO$MINCORE==SGN$C_NPAGEDYN + <DB_SIZ+DL_SIZ+DM_SIZ+DR_SIZ+TM_SIZ+XF_SIZ>
                 0000
                 0000  ;
                 0000  ; ALLOCATION GRANULARITY MASK
                 0000  ;
                 0000
0000000F         0000  MASK    = ^XF
                 0000
                 0000  ;
                 0000  ; STARLET VECTOR LOCAL DEFINITION OVERRIDES DS STARLET SERVICE VECTOR.
                 0000  ;
                 0000
800000C8         0000  SYS$CRETVA      = ^X800000C8
800000F0         0000  SYS$DXLEXH      = ^X800000F0
80000110         0000  SYS$DELTVA      = ^X80000110
80000208         0000  SYS$SETEXV      = ^X80000208
80000230         0000  SYS$SETPRT      = ^X80000230
```

```
                                        0000   822          .Subtitle       Data Psect Declarations
                                    00000000   823          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                        0000   824 ;
                                        0000   825 ; Read only data
                                        0000   826 ;
                                        0000   827 CRD_Switch_1:
                                  44 52 43 2F   0000   828          .ASCII  '/CRD'                                          ;       [100
                                        0004   829 CRD_Switch_2:
      45 43 41 52 54 5F 44 52 43 2F             0004   830          .ASCII  '/CRD_TRACE'
                                        000E   831 T_WRONGCPU:
44 49 20 6D 65 74 73 79 53 07 07 00'    000E   832          .ASCIC  <7><7>"System ID register type = !UB, expected !UB!/"   ;       [44]
79 74 20 72 65 74 73 69 67 65 72 20     001A
78 65 20 2C 42 55 21 20 3D 20 65 70     0026
2F 21 42 55 21 20 64 65 74 63 65 70     0032
                                  2F    000E
                                        003E   833
                                        003E   834 T_Both_CRD_Bits_Set:                                                 ;       [75]
4E 49 20 52 4F 52 52 45 20 3F 3F 00'    003E   835          .Ascic  "?? ERROR IN STARTING CRD - INCORRECT BITS SET!/"       ;       [75]
52 43 20 47 4E 49 54 52 41 54 53 20     004A
43 45 52 52 4F 43 4E 49 20 2D 20 44     0056
2F 21 54 45 53 20 53 54 49 42 20 54     0062
                                  2F    003E
                                        006E   836 DS$GT_LEGAL::                                                       ;       [92]
20 20 20 20 20 20 20 2F 21 2F 21 00'    006E   837          .ASCIC  "!/!/                       VAX DIAGNOSTIC SOFTWARE" -          ;
20 20 20 20 20 20 20 20 20 20 20 20     007A
53 4F 4E 47 41 49 44 20 58 41 56 20     0086
45 52 41 57 54 46 4F 53 20 43 49 54     0092
20 20 20 20 20 20 20 20 20 20 2F 21     009E   838                  "!/                                PROPERTY OF" -          ;
20 20 20 20 20 20 20 20 20 20 20 20     00AA
59 54 52 45 50 4F 52 50 20 20 20 20     00B6
                                  46 4F 20    00C2
20 20 20 20 20 20 20 20 20 20 2F 21     00C5   839                  "!/                  DIGITAL EQUIPMENT CORPORATION" -          ;
54 49 47 49 44 20 20 20 20 20 20 20     00D1
54 4E 45 4D 50 49 55 51 45 20 4C 41     00DD
4E 4F 49 54 41 52 4F 50 52 4F 43 20     00E9
20 20 20 20 20 20 20 20 2F 21 2F 21     00F5   840                  "!/!/                ***CONFIDENTIAL AND PROPRIETARY***" -      ;
4E 4F 43 2A 2A 2A 20 20 20 20 20 20     0101
4E 41 20 4C 41 49 54 4E 45 44 49 46     010D
52 41 54 45 49 52 50 4F 52 50 20 44     0119
                              2A 2A 2A 59   0125
75 41 20 65 73 55 2F 21 2F 21 2F 21     0129   841                  "!/!/!/Use Authorized Only Pursuant to a Valid Right-to-Use License"
6C 6E 4F 20 64 65 7A 69 72 6F 68 74     0135
74 20 74 6E 61 75 73 72 75 50 20 79     0141
69 52 20 64 69 6C 61 56 20 61 20 6F     014D
4C 20 65 73 55 2D 6F 74 2D 74 68 67     0159
                  65 73 6E 65 63 69     0165
                                 FC     006E
                                        016B   842
                                        016B   843
```

```
                                    016B   845          .Subtitle       Work Psect Declarations
                                00000000   846          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                                    0000   847 DS$GT_COPYRIGHT::
                                4C' 0000   848          .BYTE   DS$GT_LEGAL_END - .-1            ; Form ASCIC format              [109
2C 74 68 67 69 72 79 70 6F 43 2F 21  0001   849          .ASCII  "!/Copyright, Digital Equipment Corporation,  "          ;       [109
75 71 45 20 6C 61 74 69 67 69 44 20  000D
6F 70 72 6F 43 20 74 6E 65 6D 70 69  0019
         20 20 2C 6E 6F 69 74 61 72  0025
                                    002E   850 DS$GT_COPYEAR::                                                          ; [115]
                        36 38 39 31  002E   851          .ASCII  "1986"                                                          ;G:19
74 68 67 69 52 20 6C 6C 41 20 20 2E  0032   852          .ASCII  ".  All Rights Reserved.!/!/"
21 2E 64 65 76 72 65 73 65 52 20 73  003E
                           2F 21 2F  004A
                                    004D   853 DS$GT_LEGAL_END::
                                    004D   854 ULTRIX_FILE:
                                0A' 004D   855          .BYTE   ULTRIX_END - .-1                 ; Form ASCIC format              [114
                           2C 30 28  004E   856          .ASCII  "(0,"
                                    0051   857 ULTRIX_PARTITION:
              64 6C 65 69 66 29 36  0051   858          .ASCII  "6)field"
                                    0058   859 ULTRIX_END:
                                    0058   860 CRD_DESC:
                               000A  0058   861          .WORD   10                                                          ; [ 100 ]
                               010E  005A   862          .WORD   ^X010E                                                      ; [ 100 ]
                           00000060' 005C   863          .ADDRESS        CRD_DESC_BUF                                        ; [ 100 ]
                                    0060   864 CRD_DESC_BUF:
                           0000006A  0060   865          .BLKB   10                                                          ; [ 100 ]
                                    006A   866          .align  long
                                    006C   867 DS$FAB_INPUT::
                                    006C   868          $FAB    FAC=GET, FNA=SYS$INPUT+1, FNS=9
                                    006C               .SAVE   LOCAL_BLOCK
                                    006C               .PSECT  $ABS$,ABS
                           00000000  FE70               .=0
                                0000006C               .RESTORE
                                 03  006C               .BYTE FAB$C_BID
                                 50  006D               .BYTE FAB$C_BLN
                           000000BC  006E               .BLKB FAB$C_BLN-2
                           00000070  00BC               .=$$.TAB+FAB$L_FOP
                           00000000' 0070               .ADDRESS        $$.TMP
                           0000007C  0074               .=$$.TAB+FAB$L_ALQ
                           00000000' 007C               .ADDRESS        0
                               0000  0080               .WORD   0
                                 02  0082               .BYTE   $$.TMP
                                 00  0083               .BYTE   $$.TMP
                           00000000' 0084               .ADDRESS        0
                                 00  0088               .BYTE   0
                                 00  0089               .BYTE   FAB$C_SEQ
                                 00  008A               .BYTE   $$.TMP
                                 02  008B               .BYTE   FAB$C_VAR
                                 00  008C               .BYTE   $$.TMP
                                 00  008D               .BYTE   0
                               0000  008E               .WORD
                           00000000' 0090               .ADDRESS        0
                           00000000' 0094               .ADDRESS        0
                           0000018B' 0098               .ADDRESS        SYS$INPUT+1
                           00000000' 009C               .ADDRESS        0
                                 09  00A0               .BYTE   9
                                 00  00A1               .BYTE   0
```

```
          0000  00A2                .WORD   0
    00000000'  00A4                .ADDRESS        0
          0000  00A8                .WORD   0
            00  00AA                .BYTE   0
            00  00AB                .BYTE   0
    000000B4   00AC                .=$$.TAB+FAB$W_GBC
          0000  00B4                .WORD   0
            00  00B6                .BYTE   <<0aFAB$V_LNM_MODE> + <0aFAB$V_CHAN_MODE> + -
    000000BC   00B7                .=$$.TABEND
               00BC   869
               00BC   870 DS$RAB_INPUT::
               00BC   871          $RAB    FAB=DS$FAB_INPUT, ROP=CVT
               00BC                .SAVE   LOCAL_BLOCK
               00BC                .PSECT  $ABS$,ABS
    00000000   FE70                .=0
          000000BC                .RESTORE
            01  00BC                .BYTE RAB$C_BID
            44  00BD                .BYTE RAB$C_BLN
    00000100   00BE                .BLKB RAB$C_BLN-2
    000000C0   0100                .=$$.TAB+RAB$L_ROP
    04000000'  00C0                .ADDRESS        $$.TMP
    000000D4   00C4                .=$$.TAB+RAB$L_CTX
    00000000'  00D4                .ADDRESS        0
    000000DA   00D8                .=$$.TAB+RAB$B_RAC
            00  00DA                .BYTE   RAB$C_SEQ
            00  00DB                .BYTE   0
          0000  00DC                .WORD   0
          0000  00DE                .WORD   0
    00000000'  00E0                .ADDRESS        0
    00000000'  00E4                .ADDRESS        0
    00000000'  00E8                .ADDRESS        0
    00000000'  00EC                .ADDRESS        0
            00  00F0                .BYTE   0
            00  00F1                .BYTE   0
            00  00F2                .BYTE   0
            00  00F3                .BYTE   0
    00000000'  00F4                .ADDRESS        0
    0000006C'  00F8                .ADDRESS        DS$FAB_INPUT
    00000000'  00FC                .ADDRESS        0
               0100   872
               0100   873 DS$FAB_OUTPUT::
               0100   874          $FAB    FAC=PUT, FNA=SYS$OUTPUT+1, FNS=10, FOP=CIF
            03  0100                .BYTE FAB$C_BID
            50  0101                .BYTE FAB$C_BLN
    00000150   0102                .BLKB FAB$C_BLN-2
    00000104   0150                .=$$.TAB+FAB$L_FOP
    02000000'  0104                .ADDRESS        $$.TMP
    00000110   0108                .=$$.TAB+FAB$L_ALQ
    00000000'  0110                .ADDRESS        0
          0000  0114                .WORD   0
            01  0116                .BYTE   $$.TMP
            00  0117                .BYTE   $$.TMP
    00000000'  0118                .ADDRESS        0
            00  011C                .BYTE   0
            00  011D                .BYTE   FAB$C_SEQ
            00  011E                .BYTE   $$.TMP
            02  011F                .BYTE   FAB$C_VAR
```

```
                      00  0120                .BYTE   $$.TMP
                      00  0121                .BYTE   0
                    0000  0122                .WORD
                00000000' 0124                .ADDRESS        0
                00000000' 0128                .ADDRESS        0
                00000195' 012C                .ADDRESS        SYS$OUTPUT+1
                00000000' 0130                .ADDRESS        0
                      0A  0134                .BYTE   10
                      00  0135                .BYTE   0
                    0000  0136                .WORD   0
                00000000' 0138                .ADDRESS        0
                    0000  013C                .WORD   0
                      00  013E                .BYTE   0
                      00  013F                .BYTE   0
                00000148  0140                .=$$.TAB+FAB$W_GBC
                    0000  0148                .WORD   0
                      00  014A                .BYTE   <<0aFAB$V_LNM_MODE> + <0aFAB$V_CHAN_MODE> + -
                00000150  014B                .=$$.TABEND
                          0150    875
                          0150    876 DS$RAB_OUTPUT::
                          0150    877         $RAB    FAB=DS$FAB_OUTPUT, ROP=CCO
                      01  0150                .BYTE RAB$C_BID
                      44  0151                .BYTE RAB$C_BLN
                00000194  0152                .BLKB RAB$C_BLN-2
                00000154  0194                .=$$.TAB+RAB$L_ROP
                80000000' 0154                .ADDRESS        $$.TMP
                00000168  0158                .=$$.TAB+RAB$L_CTX
                00000000' 0168                .ADDRESS        0
                0000016E  016C                .=$$.TAB+RAB$B_RAC
                      00  016E                .BYTE   RAB$C_SEQ
                      00  016F                .BYTE   0
                    0000  0170                .WORD   0
                    0000  0172                .WORD   0
                00000000' 0174                .ADDRESS        0
                00000000' 0178                .ADDRESS        0
                00000000' 017C                .ADDRESS        0
                00000000' 0180                .ADDRESS        0
                      00  0184                .BYTE   0
                      00  0185                .BYTE   0
                      00  0186                .BYTE   0
                      00  0187                .BYTE   0
                00000000' 0188                .ADDRESS        0
                00000100' 018C                .ADDRESS        DS$FAB_OUTPUT
                00000000' 0190                .ADDRESS        0
                          0194    878
                          0194    879
                          0194    880 DS$GA_DS_Ctrl_C_First::             ; Address of 1st DS Control-C Handler   [70]
                00000000  0194    881         .Long   0                   ; No handler by default.                [51]
                          0198    882
                          0198    883 DS$GA_DS_Ctrl_C_Second::            ; Address of 2nd DS Control-C handler   [70]
                00000000  0198    884         .Long   0                   ; No handler by default.                [70]
                          019C    885
                          019C    886 DS$L_UserCntrlC::                   ;                                       [73]
                00000000  019C    887         .Long   0                   ; Address of user Control-C handler
                          01A0    888
                          01A0    889 EXIT$DESBLK:                        ; DESCRIPTOR OF EXIT HANDLER
                00000000  01A0    890         .LONG   0                   ; FORWARD LINK
```

```
00000E70' 01A4   891          .ADDRESS        EXIT$HANDLR     ; ADDRESS OF EXIT HANDLER
00000001  01A8   892          .LONG   1                       ; ONE PARAMETER
000001B0' 01AC   893          .ADDRESS        10$             ; ADDRESS TO STORE EXIT REASON
00000000  01B0   894 10$:     .LONG   0
          01B4   895
000001B8  01B4   896 SAVEFP:  .BLKL   1                       ; INITIAL CONTENTS OF FP
          01B8   897
000001E0  01B8   898 DS$AT_DDIR:      .BLKW   20              ; Saved default directory
          01E0   899
00000208  01E0   900 DS$AT_DDEV:      .BLKW   20              ; Saved default device
          0208   901
00000408  0208   902 IMAGE_HEADER::   .BLKB   512             ; Storage for diagnostic's image header
          0408   903 IMAGE_HEADER_END::                       ; End of buffer.
          0408   904
          0408   905 ; The following location is for temporary usage until all mp modules are linked. [11
          0408   906                                                                              ;G:6
00000000  0408   907 MP$GR_BOOTED_PROCESSORS::       .LONG 0 ; Structure to denote active mp system [112]
          040C   908
       00 040C   909 DS$GB_SYSTYPE:: .BYTE   0               ; SYSTEM TYPE (for MicroVax CPU)  [127] ;G:1
00000000  040D   910 DS$GL_SYSTYPE:  .LONG   0               ; and longword for FBI access     [133] ;G:2
          0411   911                                                                              ;G:15
00000000  0411   912 START_RUN_PC:   .LONG   0               ; PC in KB_CHECK                  [128] ;G:17
          0415   913                                         ; Item list for $GETJPIW call     [129] ;G:17
     0004 0415   914 CPUTIM:         .WORD   4               ; buffer length                   [129] ;G:17
     0407 0417   915                 .WORD   JPI$_CPUTIM     ; Item                            [129] ;G:17
00000425' 0419   916                 .ADDRESS JPICPUTIM      ; address of buffer               [129] ;G:17
00000429' 041D   917                 .ADDRESS JPICPUTIMS     ; return buffer length            [129] ;G:17
00000000  0421   918                 .LONG   0               ; end of list                     [129] ;G:17
00000000  0425   919 JPICPUTIM:      .LONG   0               ; Contains cpu time               [129] ;G:17
00000000  0429   920 JPICPUTIMS:     .LONG   0               ; Return # bytes of cpu time      [129] ;G:17
```

```
                     042D   922
                     042D   923 ;
                     042D   924 ; PERMANENT CCB DATA BASE
                     042D   925 ;
            00000040 042D   926 DS$K_CCB         ==      64              ; NUMBER OF CCB'S
                     042D   927
                     042D   928 CTL$GL_CCB::
            0000082D 042D   929         .BLKB   DS$K_CCB * CCB$C_LENGTH
                     082D   930
                     082D   931 CTL$GL_CCBBASE::
            0000082D' 082D  932         .ADDRESS        CTL$GL_CCBBASE
                     0831   933
                     0831   934 SCH$GL_CURPCB::          ; And Incidentially the current PCB address
                     0831   935 DS$GL_PCBVEC::           ; List of PCB's (indexed) (DS has only 1)
            00000000' 0831  936         .ADDRESS DS$AX_SOFTPCB          ; Address of software PCB
                     0835   937
                     0835   938 ;
                     0835   939 ; FORK QUEUE LISTHEADS
                     0835   940 ;
                     0835   941
                     0835   942 ;       .ALIGN  QUAD
                     0835   943 SWI$GL_FQFL::                            ; FORWARD LINK
            00000835' 0835  944 A:      .LONG   A                       ; IPL-6 LISTHEAD
                     0839   945
                     0839   946 SWI$GL_FQBL::                            ; BACKWARD LINK
            00000835' 0839  947         .LONG   A                       ;
      0000083D'0000083D' 083D 948 2$:   .LONG   2$,2$                   ; IPL-7 LISTHEAD
      00000845'00000845' 0845 949 3$:   .LONG   3$,3$                   ; IPL-8 LISTHEAD
      0000084D'0000084D' 084D 950 4$:   .LONG   4$,4$                   ; IPL-9 LISTHEAD
      00000855'00000855' 0855 951 5$:   .LONG   5$,5$                   ; IPL-10 LISTHEAD
      0000085D'0000085D' 085D 952 6$:   .LONG   6$,6$                   ; IPL-11 LISTHEAD
                     0865   953
            00000869 0865   954 BOOT$L_ADP::     .BLKL   1              ; Address of BOOT adapter                [91]
            0000086D 0869   955 BOOT$L_CSR::     .BLKL   1              ; Address of BOOT device csr             [91]
            00000871 086D   956 BOOT$L_UNIT::    .BLKL   1              ; Unit number of Boot device             [91]
            00000875 0871   957 BOOT$L_R1::      .BLKL   1              ; BI port to boot from                   [117
            00000879 0875   958 BOOT$L_R2::      .BLKL   1              ; CI port station to boot from      [86][91]
            0000087D 0879   959 Boot$L_R5::      .BLKL   1              ; R5 with switches from VMB/DIAGBOOT     [71]
            00000881 087D   960 BOOT$L_DEVTYP::  .BLKL   1              ; device type of BOOT media:             [91]
                     0881   961                                        ; 0=massbus,1=RK06/07,2=RL01/02       ;G:36
                     0881   962 BOOT$L_SCORPIO_R3::                                                          ;G:34
            00000885 0881   963                 .BLKL   1              ; R3 contains #of console drive booted from
                     0885   964                                        ; 0=massbus,1=RK06/07,2=RL01/02       ;G:35A
                     0885   965 RX50_DISK_SELECT::                                                          ;G:35A
            00000889 0885   966                 .BLKL   1              ; contains #of console drive booted from (sc
            0000088D 0889   967 BOOT$A_CCA::     .BLKL   1              ; Addr of LLL's console communications area.
                     088D   968 DS$GQ_EFL::                             ; EVENT FLAGS, LOGICAL.
            00000891 088D   969 DS$GL_EFC0::     .BLKL   1              ; EVENT FLAG CLUSTER 0.
            00000895 0891   970 DS$GL_EFC1::     .BLKL   1              ; EVENT FLAG CLUSTER 1.
            00000899 0895   971 DS$GA_LASTADR::  .BLKL   1              ; LAST ADDRESS USED BY SUPERVISOR.
```

```
                     0899   973 ;
                     0899   974 ; DYNAMIC EXECUTIVE STORAGE.
                     0899   975 ;
                     0899   976
0000000F             0899   977 DS$K_ASCIBUF     == 15
00000200             0899   978 DS$K_BUFSIZ      == 512
00000080             0899   979 DS$K_STRBUF      == 128
                     0899   980
0000089D             0899   981 DS$GL_FLAGS::    .BLKL   1       ; SUPERVISOR INTERNAL FLAGS
0000089F             089D   982 DS$GW_TTIN::     .BLKW   1
000008A1             089F   983 DS$GW_TTOUT::    .BLKW   1
000008A5             08A1   984 DS$GL_ERRCNT::   .BLKL   1       ; RUNNING ERROR COUNT.
                     08A5   985 DS$GL_HARDERR_COUNT::        ;                                       [50]
000008A9             08A5   986                  .BLKL   1       ; Summary of Hard Errors                [50]
                     08A9   987 DS$GL_PREPERR_COUNT::        ;                                       [55]
000008AD             08A9   988                  .BLKL   1       ; Summary of Device Peparation Errors   [55]
                     08AD   989 DS$GL_SOFTERR_COUNT::        ;                                       [50]
000008B1             08AD   990                  .BLKL   1       ; Summary of Soft Errors                [50]
                     08B1   991 DS$GL_DEVERR_COUNT::         ;                                       [50]
000008B5             08B1   992                  .BLKL   1       ; Summary of Device Errors              [50]
                     08B5   993 DS$GL_SYSERR_COUNT::         ;                                       [50]
000008B9             08B5   994                  .BLKL   1       ; Summary of System Errors              [50]
                     08B9   995 DS$GL_ERRSUP_COUNT::         ;                                       [50]
000008BD             08B9   996                  .BLKL   1       ; Summary of ErrSup Errors              [50]
000008C1             08BD   997 DS$GL_NUMTEST::  .BLKL   1       ; NUMBER OF TESTS IN PROGRAM.
000008C5             08C1   998 DS$GL_FSTTEST::  .BLKL   1       ; FIRST TEST TO EXECUTE
000008C9             08C5   999 DS$GL_LSTTEST::  .BLKL   1       ; LAST TEST TO EXECUTE
000008CD             08C9  1000 DS$GL_SUBTEST::  .BLKL   1       ; SUBTEST NUMBER FOR LOOPING.
000008D1             08CD  1001 DS$GA_CHKLPPC::  .BLKL   1       ; ADDR OF CURRENT 'CHKLOOP' CALL
000008D5             08D1  1002 DS$GA_LOOPADR::  .BLKL   1       ; ADDRESS OF CURRENT SUBTEST
000008D9             08D5  1003 DS$GL_RUNTIM::   .BLKL   1       ; LOCATION TO HOLD ELAPSED RUN TIME
000008DD             08D9  1004 DS$GL_TIMESEC::  .BLKL   1       ; CONVERTED RUN TIME IN SECONDS
000008E1             08DD  1005 DS$GL_WAITIM::   .BLKL   1       ; NUMBER OF MICROSECONDS TO DELAY
000008E5             08E1  1006 DS$GL_RADIX::    .BLKL   1       ; DEFAULT RADIX.
000008E9             08E5  1007 DS$GL_SIZE::     .BLKL   1       ; DEFAULT DATA SIZE.
                     08E9  1008 DS$GQ_BUFQWD::                   ; BUFFER QUADWORD DESCRIPTOR.
000008ED             08E9  1009 DS$GL_BUFLEN::   .BLKL   1       ; LENGTH OF CURRENT BUFFER CONTENTS.
000008F1             08ED  1010 DS$GA_BUFPTR::   .BLKL   1       ; POINTS TO CURRENT POSITION
000008F2             08F1  1011 DS$GB_BYTEBUF::  .BLKB   1       ; ONE BYTE BUFFER FOR CHK K.B.
000008F3             08F2  1012 DS$GB_TYPECODE::.BLKB    1       ; Type code for 'SET BINARY' output     [46]
                     08F3  1013 DS$GB_INHIBIT_NAMING::           ; Flag byte for ATTACH                  [60]
000008F4             08F3  1014                  .BlkB   1       ;                                       [60]
00000903             08F4  1015 DS$GT_ASCIBUF::  .BLKB   DS$K_ASCIBUF    ; 15 BYTES FOR NUMERICAL ASCII
00000B03             0903  1016 DS$GT_BUFFER::   .BLKB   DS$K_BUFSIZ     ; INPUT-OUTPUT BUFFER
00000B83             0B03  1017 DS$GT_STRBUF::   .BLKB   DS$K_STRBUF     ; COUNTED ASCII STRING BUFFER
00000B8B             0B83  1018 DS$GL_TERMCHAR::.BLKQ    1       ; Characteristics of user terminal  [87]
```

```
                                0B8B  1020          .SBTTL  GLOBAL PSECT DECLARATIONS FOR VSDP                          ;G:28
                        00000000  0B8B  1021 DEVCHAR:    .LONG        0                                  ; [134]      ;G:28
                        00000000  0B8F  1022 PID:     .LONG   0                                          ; [134]      ;G:28
                        00000B97  0B93  1023 PIDNO:   .BLKL   1                                          ; [134]      ;G:28
                        00000B9B  0B97  1024 PIDNOS:  .BLKL   1                                          ; [134]      ;G:28
                            000C  0B9B  1025 ITEMS:   .WORD   12                                         ; [134]      ;G:28
                            0319  0B9D  1026          .WORD   JPI$_PID                                   ; [134]      ;G:28
                        00000B93' 0B9F  1027          .ADDRESS PIDNO                                                  ;G:28
                        00000B97' 0BA3  1028          .ADDRESS PIDNOS                                                 ;G:28
                        00000000  0BA7  1029          .LONG   0                                          ; [134]      ;G:28
                        00000BB3  0BAB  1030 IOSB:    .BLKQ   1                                          ; [134]      ;G:28
                                  0BB3  1031 GLBL_SECT_NAME:                                                            ;G:28
                        0000000C  0BB3  1032          .LONG   12                                         ; [134]      ;G:28
                        00000BC3' 0BB7  1033          .ADDRESS FAOBUF                                    ; [134]      ;G:28
                                  0BBB  1034 GLBL_SECT_AREA:                                                            ;G:28
                        00000BC3  0BBB  1035          .BLKL   2                                                       ;G:28
                        00000BCF  0BC3  1036 FAOBUF:  .BLKB   12                                         ; [134]      ;G:28
                        00000BD1  0BCF  1037 FAOLEN:  .BLKW   1                                                       ;G:28
                        00000BD3  0BD1  1038          .BLKW   1                                          ; [134]      ;G:28
                                  0BD3  1039 CTRL_STR:                                                                  ;G:28
58 21 5F 43 53 47 00000BDB'010E0000' 0BD3  1040          .ASCID /GSC_!XL/                                   ; [134]      ;G:28
                              4C  0BE1
                                  0BE2  1041 ;                                                                          ;G:28
```

```
ZZ-ENSAA-10.10  Data Psect Declarations                          3-MAR-1988       Fiche 13 Frame B11        Sequence 2603
KERNEL                        *** KERNEL Main control routine     18-NOV-1987 15:15:02  VAX/VMS Macro V04-00        Page  36
10.9-36                       Data Psect Declarations             18-NOV-1987 15:05:32  KERNEL.MAR;2                      (1)

                                0BE2  1043          .Subtitle       Data Psect Declarations
                             0000016B 1044          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                016B  1045
                                016B  1046          MODNAM  KERNEL
                 4C 45 4E 52 45 4B 00' 016B  $MODULE:          .ASCIC  \KERNEL\
                                06    016B
                                0172  1047
                    00000000'00000000' 0172  1048 SYSVEC: .LONG   DS$AQ_SYSSRV, DS$AQ_SSEND       ; SYSTEM SERVICE VECTORS.
                                017A  1049
                                017A  1050 SSPROT: $CRETVA INADR=SYSVEC
                          00000003' 017A                    .ADDRESS        3
                          00000172' 017E                    .ADDRESS        SYSVEC
                          00000000' 0182                    .ADDRESS        0
                          00000000' 0186                    .ADDRESS        0
                                018A  1051
```

```
                018A  1053 SYS$INPUT:
```

```
      54 55 50 4E 49 24 53 59 53 00' 018A  1055          .ASCIC  "SYS$INPUT"
                                 09  018A
                                     0194  1056 SYS$OUTPUT:
   54 55 50 54 55 4F 24 53 59 53 00' 0194  1057          .ASCIC  "SYS$OUTPUT"
                                 0A  0194
                                     019F  1058
                0000FE00 0000FE00  019F  1059 SUPBUF: .LONG   DSA$AL_APTMAIL,DSA$AL_APTMAIL ; APT MAILBOX AREA
```

```
                        01A7  1061
0000F9FF 00000200       01A7  1062 PRGBUF: .LONG   DS$A_PRGBGN,DS$A_PRGBGN+DS$K_PRGSIZ-1 ; PROGRAM OVERLAY AREA
                        01AF  1063 ;                                                        [101]
```

```
                    01AF  1065          .SBTTL  DIAGNOSTIC SUPERVISOR BOOTSTRAP ROUTINE.
                00000000  1066          .PSECT  CODE, EXE, NOWRT, SHR, LONG
                    0000  1067 ;++
                    0000  1068 ; FUNCTIONAL DESCRIPTION:
                    0000  1069 ;
                    0000  1070 ;       INITIAL SETUP ROUTINE FOR STAND ALONE VERSIONS OF THE DIAGNOSTIC
                    0000  1071 ;       SUPERVISOR.
                    0000  1072 ;
                    0000  1073 ; CALLING SEQUENCE:
                    0000  1074 ;
                    0000  1075 ;       NONE
                    0000  1076 ;
                    0000  1077 ; INPUT PARAMETERS:
                    0000  1078 ;
                    0000  1079 ;       NONE
                    0000  1080 ;
                    0000  1081 ; IMPLICIT INPUTS:
                    0000  1082 ;
                    0000  1083 ;       NONE
                    0000  1084 ;
                    0000  1085 ; OUTPUT PARAMETERS:
                    0000  1086 ;
                    0000  1087 ;       NONE
                    0000  1088 ;
                    0000  1089 ; IMPLICIT OUTPUTS:
                    0000  1090 ;
                    0000  1091 ;       NONE
                    0000  1092 ;
                    0000  1093 ; COMPLETION CODES:
                    0000  1094 ;
                    0000  1095 ;       NONE
                    0000  1096 ;
                    0000  1097 ; SIDE EFFECTS:
                    0000  1098 ;
                    0000  1099 ;       Plenty!
                    0000  1100 ;
                    0000  1101 ;--
```

```
                                 0000  1103 ;+
                                 0000  1104 ; Normal Diagnostic Supervisor (in stand-alone mode) entry from START 10000
                                 0000  1105 ;-
                                 0000  1106
                                 0000  1107 BOOT::
                      12   1F   DA  0000  1108          MTPR    #31,#PR$_IPL                    ; RAISE IPL TO 1F
              5E  00000000'8F   D0  0003  1109          MOVL    #ISTKPTR, SP                    ; INITIALIZE INTERRUPT STACK
                      52   50   D0  000A  1110          MOVL    R0,R2                           ; Save Flag
                   50   01   06   78  000D  1111          ASHL    #9-3, #1, R0                    ; PAGE OF QUADWORDS COUNT.
              51   0000FE00'EF   DE  0011  1112          MOVAL   DSA$AL_APTMAIL, R1              ; START OF APT MAILBOX, ETC.
                         81   7C  0018  1113 10$:       CLRQ    (R1)+                           ; CLEAR IT.
                      FB 50   F5  001A  1114          SOBGTR  R0, 10$                         ; UNTIL END.
    0000FE00'EF   00003000 8F   C8  001D  1115          BISL2   #DSA$M_DFLTFLGS, DSA$GL_FLAGS   ; SET DEFAULT FLAGS.
                                 0028  1116
                   00000865'EF   D5  0028  1117          TSTL    BOOT$L_ADP                      ; Check one-shot flag
                         08   12  002E  1118          BNEQ    12$                             ; Branch if shot                [114
                      50   3E   DB  0030  1119          MFPR    #PR$_SID,R0                     ; Get SID register
                      52   50   D1  0033  1120          CMPL    R0,R2                           ; Entered from DIAGBOOT?
                         03   13  0036  1121          BEQL    15$                             ;                               [114
                       009E   31  0038  1122 12$:       BRW     30$                             ; Branch if not                 [114
          00000865'EF   5C AB   D0  003B  1123 15$:       MOVL    RPB$L_ADPPHY(R11),L^BOOT$L_ADP  ; Save adapter address
          00000869'EF   54 AB   D0  0043  1124          MOVL    RPB$L_CSRPHY(R11),L^BOOT$L_CSR  ; Save device csr address
          0000086D'EF   64 AB   3C  004B  1125          MOVZWL  RPB$W_UNIT(R11),L^BOOT$L_UNIT   ; Save boot unit number        [93]
          00000871'EF   20 AB   D0  0053  1126          MOVL    RPB$L_BOOTR1(R11),L^BOOT$L_R1   ; Save BI number               [117
          00000875'EF   24 AB   D0  005B  1127          MOVL    RPB$L_BOOTR2(R11),L^BOOT$L_R2   ; Save CI port number          [86]
          00000879'EF   30 AB   D0  0063  1128          MOVL    RPB$L_BOOTR5(R11),L^BOOT$L_R5   ; Save flags from boot
          0000087D'EF   66 AB   9A  006B  1129          MOVZBL  RPB$B_DEVTYP(R11), -
                                    0073  1130                  L^BOOT$L_DEVTYP                 ; Save device type from boot
               50   50   E8 8F   78  0073  1131          ASHL    #-24,R0,R0                      ; Get upper byte of PR$_SID     [136
                         0A   50   91  0078  1132          CMPB    R0,#PR$_SYS_TYP_LLL             ; Are we a LLL system?  [136] ;G:35
                         08   12  007B  1133          BNEQ    17$                             ; If not skip next line         [136
          00000889'EF   2C AB   D0  007D  1134          MOVL    RPB$L_BOOTR4(R11),BOOT$A_CCA    ; Save addr of LLL's CCA.       [136
                      5A   68 AB   9E  0085  1135 17$:       MovAB   RPB$T_File(R11), R10            ; point to ASCIC file name      [52]
                         59   8A   9A  0089  1136          MovZBL  (R10)+, R9                      ; get length/address in R9/R10  [52]
                         1D   12  008C  1137          BNEQ    20$                             ; Branch if present            [114
    50    00000879'EF   04   1C   EE  008E  1138          EXTV    #28,#4,L^BOOT$L_R5,R0           ; Get partition value          [114
          00000051'EF   50   30   81  0097  1139          ADDB3   #^A"0",R0,L^ULTRIX_PARTITION    ; Put ascii number in text     [114
               5A  0000004D'EF   9E  009F  1140          MOVAB   ULTRIX_FILE,R10                 ; Address of default specs     [114
                         50   8A   9A  00A6  1141          MOVZBL  (R10)+,R0                       ; Length of string             [114
                         1A   11  00A9  1142          BRB     25$                             ; Set ULTRIX default           [114
                      50   59   D0  00AB  1143 20$:       MOVL    R9,R0                           ; Length of string             [114
                      6A   28   91  00AE  1144          CMPB    #^A"(",(R10)                    ; Check if ULTRIX              [114
                         12   13  00B1  1145          BEQL    25$                             ; Branch to use entire string  [114
                      50   6A   9A  00B3  1146          MovZBL  (R10), R0                       ; Get first character of name  [52]
                      50   02   80  00B6  1147          AddB2   #2, R0                          ; Make ] or > for end of dir.  [52]
                   6A   59   50   3A  00B9  1148          LocC    R0, R9, (R10)                   ; Search for end of directory  [52]
                         1A   13  00BD  1149          BEql    30$                             ; Exit if not found            [52]
                   50   51   5A   C3  00BF  1150          SubL3   R10, R1, R0                     ; Compute length of the        [52]
                         50   D6  00C3  1151          IncL    R0                              ; .. file name string          [52]
                      51   5A   D0  00C5  1152 25$:       MovL    R10, R1                         ; Get address of string        [52]
               52   00000000'EF   DE  00C8  1153          MovAL   L^Ds$GL_CliBase, R2             ; Point to CLI data base       [52]
                     08 A2   50   7D  00CF  1154          MovQ    R0, Cli$Q_File(R2)              ; Set it up for SET LOAD       [52]
                    00000000'EF   16  00D3  1155          Jsb     L^Dsv$SetLoad                   ; And do the SET LOAD command  [58]
                                    00D9  1156 30$:
```

```
                                00D9  1158 ;+
                                00D9  1159 ; APT ENTRY POINT FROM START 10004
                                00D9  1160 ;-
                                00D9  1161 APT::
                  12   1F   DA  00D9  1162          MTPR    #31,#PR$_IPL                    ; RAISE IPL TO HOLD OFF
                                00DC  1163                                                  ; ANY INTERRUPTS
         5E   00000000'8F   D0  00DC  1164          MOVL    #ISTKPTR,SP                     ; SET INTERRUPT STACK
                     5D   D4  00E3  1165          CLRL    FP                              ; NO PREVIOUS STACK FRAME
      0000FE14'EF    3E   DB  00E5  1166          MFPR    #PR$_SID,DSA$GL_SID             ; SID SAVED IN FULL
                                00EC  1167 ;+                                                                   ;G:34
                                00EC  1168 ;  If Scorpio save R3                                                ;G:34
                                00EC  1169 ;-                                                                   ;G:34
         05   0000FE17'EF   91  00EC  1170          CMPB    DSA$GL_SID+3,#PR$_SID_TYP8SS    ; Is it a Scorpio?              [139
                     10   12  00F3  1171          BNEQ    1$                              ; If NO branch, ELSE save       [139
         02   00000885'EF   91  00F5  1172          CMPB    RX50_DISK_SELECT,#2             ; Already booted from csa2?   ;G:37A
                     07   13  00FC  1173          BEQL    1$                              ; Yes? Leave this as the drive # ;G:
      00000885'EF    53   D0  00FE  1174          MOVL    R3,RX50_DISK_SELECT             ; # of drive booted from        [139
00000000'EF   0000FE17'EF   90  0105  1175 1$:     MOVB    DSA$GL_SID+3, -                                         ;G:34
                                0110  1176                  EXE$GB_CPUTYPE              ; Load processor type for BOOTDRIVR
         10   00000000'EF   91  0110  1177          CMPB    EXE$GB_CPUTYPE,#PR$_SID_TYPRTUV2; Is CPU type a RT  MicroVax2? [130]
                     09   13  0117  1178          BEQL    2$                              ; Yes                           [130] ;G:
         08   00000000'EF   91  0119  1179          CMPB    EXE$GB_CPUTYPE,#PR$_SID_TYPUV2  ; Is CPU type a MicroVax2? [127] ;G:
                     3D   12  0120  1180          BNEQ    3$                              ; NO, let SYSTYPE = 0           [127] ;G:
0000040D'EF   20040004   9F   D0  0122  1181 2$:     MOVL    @#SYS_TYPE,DS$GL_SYSTYPE        ; Get Sys Type from IOSPACE[127][133
0000040C'EF   00000410'EF   90  012D  1182          MOVB    DS$GL_SYSTYPE+3,DS$GB_SYSTYPE   ; Move upper byte of longw.[133] ;G:
         03   0000040C'EF   91  0138  1183          CMPB    DS$GB_SYSTYPE,#PR$_SYS_TYP900   ; Is this an Andromeda sys [127] ;G:
                     1E   12  013F  1184          BNEQ    3$                              ; NO, then branch               [127] ;G:
         50   3FFFE400   9F   D0  0141  1185          MOVL    @#PPA_SPID,R0                   ; Get CPU ID from IOspace  [127][133
              00000000'EF   50   F6  0148  1186          CVTLB   R0,PPA$GB_CPU               ; Move it into a byte      [133] ;G:
50    00000000'EF    06   02   EF  014F  1187          EXTZV   #2,#6,PPA$GB_CPU,R0         ; EXTRACT the Module #     [133] ;G:
              00000000'EF   50   F6  0158  1188          CVTLB   R0,PPA$GB_MID               ; and move the lower byte  [133] ;G:
                                015F  1189                                                                      ;G:15
 00000899'EF   00000082   8F   D0  015F  1190 3$:     MOVL    #DS$M_CMDFLG ! DS$M_LODFLG, -   ; SET COMMAND AND LOAD MODE FLAGS. ;
                                016A  1191                    L^DS$GL_FLAGS
      000008E1'EF    10   D0  016A  1192          MOVL    #16, L^DS$GL_RADIX              ; SET UP DEFAULT RADIX.
         11   00000000'8F   DA  0171  1193          MTPR    #SCB_IMAGE, #PR$_SCBB           ; SYSTEM CONTROL BLOCK BASE.
         10   00000000'8F   DA  0178  1194          MTPR    #PCB_BASE, #PR$_PCBB            ; PROCESS CONTROL BLOCK BASE.
                                017F  1195 ;
                                017F  1196 ; CLEAR ALL BUFFER AREAS.
                                017F  1197 ;
         50   00000000'EF   DE  017F  1198          MOVAL   AX_BUFF, R0                     ; START OF BUFFERS.
                     80   7C  0186  1199 10$:     CLRQ    (R0)+                           ; CLEAR IT.
      00000000'8F    50   D1  0188  1200          CMPL    R0, #POPT_BASE                  ; CHECK FOR ENOUGH.             ;G:34
                     F5   19  018F  1201          BLSS    10$                             ; LOOP TIL DONE.
                                0191  1202
                                0191  1203 ;
                                0191  1204 ; SYSTEM CONTROL BLOCK INITIALIZATION.
                                0191  1205 ;
         50   00000000'8F   D0  0191  1206          MOVL    #XDELTBIT,R0                    ; Get address of Tbit handler
                     1E   13  0198  1207          BEQL    20$                             ; Branch if no XDELTA linked
         51   00000000'EF   DE  019A  1208          MOVAL   SCB_IMAGE,R1                    ; Point to SCBB
               28 A1   60   DE  01A1  1209          MOVAL   (R0),SCB$L_TBIT(R1)             ; Set tbit vector
      2C A1   00000000'EF   9E  01A5  1210          MOVAB   XDELBPT,SCB$L_BREAK(R1)         ; Set breakpoint vector
      03 00000879'EF    02   E1  01AD  1211          BBC     #RPB$V_INIBPT,BOOT$L_R5,20$     ; Branch if no initial breakpoint
                    010A   30  01B5  1212          BSBW    INI$BRK                         ; Initial XDELTA breakpoint
                                01B8  1213 20$:
                                01B8  1214 ;
```

```
                                01B8  1215 ; Disable recognition of ^C's for the duration of this stand-alone          [66]
                                01B8  1216 ; initialization. Note that the KB_Check routine MUST NOT be called before  [66]
                                01B8  1217 ; this is executed!!!!!                                                     [66]
                                01B8  1218 ;
                                01B8  1219
                                01B8  1220          $DS_CntrlC_S    Disabl=#1                ; Set Disable flag          [66]
                     01    DD   01B8                        PUSHL   #1
                     00    DD   01BA                        PUSHL   #0
        00000000'9F  02    FB   01BC                        CALLS   #2, a#DS$CNTRLC
                                01C3  1221
            00000194'EF    D4   01C3  1222          ClrL    L^DS$GA_DS_Ctrl_C_First          ; Clear the VDS's 1st ^C routine[72]
            00000198'EF    D4   01C9  1223          ClrL    L^DS$GA_DS_Ctrl_C_Second         ; Clear the VDS's 2nd ^C routine[72]
            0000019C'EF    D4   01CF  1224          ClrL    L^DS$L_UserCntrlC                ; Clear the User's ^C handler   [72]
                                01D5  1225
                                01D5  1226 ;
                                01D5  1227 ; Check correct processor type for this supervisor
                                01D5  1228 ;
             50   00'8F    9A   01D5  1229          MOVZBL  #<DS$GK_SIDa-24>,R0              ; Get expected value for type
        50   0000FE17'EF   91   01D9  1230          CMPB    DSA$GL_SID+3,R0                  ; Same as current?
                     23    13   01E0  1231          BEQL    30$                              ; Branch if so
                 50  10    91   01E2  1232          CMPB    #PR$_SID_TYPRTUV2,R0             ; Was RT-VAX expected          [132
                     09    12   01E5  1233          BNEQ    28$                              ; Branch if not. error.        [132
        0000FE17'EF  08    91   01E7  1234          CMPB    #PR$_SID_TYPUV2,DSA$GL_SID+3     ; Is it AURORA ?               [132
                     15    13   01EE  1235          BEQL    30$                              ; Allow both AURORA and RT-VAX  [132
                                01F0  1236                                                                                ;G:20
                                01F0  1237 28$:     $PRINTI_S       T_WRONGCPU, -                                          ;G:20
                                01F0  1238                   DSA$GL_SID+3,R0                 ; No, inform the operator
                     50    DD   01F0                        PUSHL   R0
            0000FE17'EF    DD   01F2                        PUSHL   DSA$GL_SID+3
            0000000E'EF    9F   01F8                        PUSHAB  T_WRONGCPU
        00000000'EF  03    FB   01FE                        CALLS   #$$N, DSX$PRINTI
                                0205  1239 ;
                                0205  1240 ; Do CPU specific stuff
                                0205  1241 ;
            00000000'EF    16   0205  1242 30$:     JSB     ONLY_CPU                         ;                              [126
                                020B  1243                                                                                ;G:7
                                020B  1244 ;                                                                                ;G:7
                                020B  1245 ; Initialize the SCB                                                             ;G:7
                                020B  1246 ;                                                                                ;G:7
                                020B  1247          $DS_INITSCB_G                            ; Initialize SCB               ;G:7
        00000000'9F  6E    FA   020B                        CALLG   (SP), a#DS$INITSCB
                                0212  1248
                                0212  1249                                                                                ;G:8
                                0212  1250 ;
                                0212  1251 ; MEMORY MANAGEMENT INITIALIZATION.
                                0212  1252 ;
                                0212  1253
            00000000'EF    16   0212  1254 40$:     JSB     L^MAPMEM                         ; Map physical memory          ;G:8
                                0218  1255
        00000000'EF  6E    FA   0218  1256          CALLG   (SP),QIO$INITIALIZE              ; SETUP QIO DATABASE
            00000000'EF    7C   021F  1257          CLRQ    L^DS$GL_BUFCNT                   ; INIT BUFFER COUNTS, P0 + P1.
            00000008'EF    7C   0225  1258          CLRQ    L^DS$GL_BUFCNT+8                 ;  SYS AND (UNUSED).
                 38  00    DA   022B  1259          MTPR    #0, #PR$_MAPEN                   ;
            00000000'EF    94   022E  1260          CLRB    DS$GB_MM_ENB             ; Memory management not enabled
                                0234  1261
                                0234  1262 ;
                                0234  1263 ; INITIALIZE PROCESS CONTROL BLOCK.
```

```
                                0234  1264 ;
                                0234  1265
                                0234  1266 ;
                                0234  1267 ; INITIALIZE EXE$GQ_SYSTIME (SYSTEM DATE/TIME QUADWORD)
                                0234  1268 ;
                                0234  1269
00000007'EF   0000000F'EF   D0  0234  1270          MOVL    L^DS$GQ_DAYTIM+15, -                ; GET YEAR FROM LINK DATE
                                023F  1271                    L^DS$GT_NEWYEAR+7
    00000000'EF   0000'8F   3C  023F  1272          MOVZWL  #DS$K_NYSIZ, L^DS$GQ_CURYEAR        ; BUILD A STRING DESCRIPTOR
00000004'EF   00000000'EF   DE  0248  1273          MOVAL   L^DS$GT_NEWYEAR, -
                                0253  1274                    L^DS$GQ_CURYEAR+4
                                0253  1275          $BINTIM_S  L^DS$GQ_CURYEAR, L^DS$GQ_CURYEAR
              00000000'EF   7F  0253                          PUSHAQ  L^DS$GQ_CURYEAR
              00000000'EF   7F  0259                          PUSHAQ  L^DS$GQ_CURYEAR
         00000000'GF   02   FB  025F                          CALLS   #2,G^SYS$BINTIM
00000000'EF   00000000'EF   7D  0266  1276          MOVQ    L^DS$GQ_CURYEAR, EXE$GQ_SYSTIME ; PRIME SYSTIME WITH THIS YEAR
              00000000'EF   16  0271  1277          JSB     HDW_CLOCK_SET                       ; Do cpu specific clock setup.  [104
                                0277  1278                                                      ; Reads/sets WATCH chip/TODR    [104
                                0277  1279                                                      ; for 11/zzz,11/bbb.            [104
                                0277  1280                                                                                  ;G:8
              00000000'EF   16  0277  1281          JSB     L^CLK$RE_INIT                       ; INIT INTERVAL TIMER AND CLOCK
              00000000'EF   16  027D  1282          JSB     L^DSR$SETIMR_INI                    ; INIT THE TIMER QUEUE'S
                                0283  1283
00000000'EF   00000000'EF   D0  0283  1284          MOVL    ONLY$GL_TENUSEC,EXE$GL_TENUSEC      ; For TIMEDWAIT macro           [113
00000000'EF   00000000'EF   D0  028E  1285          MOVL    ONLY$GL_UBDELAY,EXE$GL_UBDELAY      ; For TIMEDWAIT macro           [113
                                0299  1286
                                0299  1287 ;        PUSHL   #0                                  ; CLEAR CONTROL-C HANDLER       [66]
                                0299  1288 ;        CALLS   #1,L^DSX$CNTRLC                                                     [66]
                                0299  1289
              00000000'EF   16  0299  1290          JSB     L^INI$PTABLE                        ; Clear P-table pointers
                                029F  1291
              0000088A'EF   16  029F  1292          JSB     L^INIT_CONTEXT                      ; Initialize Context....
                                02A5  1293
              00000000'EF   16  02A5  1294 50$:     JSB     L^INI$LOAD_DEVICE                   ; Make ptables for console and  [91]
                                02AB  1295                                                      ; boot devices                  [91]
                                02AB  1296
                                02AB  1297 ; Size the console terminal.                                                 [87]
                                02AB  1298
00000000'EF   00   FB           02AB  1299          CALLS   #0,SIZE_TERM                ;                               [87]
                                02B2  1300
                                02B2  1301 ;
                                02B2  1302 ; Initialize RMS file control block pointer table (RMS$FILE_TABLE)
                                02B2  1303 ;
              00000000'EF   16  02B2  1304          JSB     L^RMS$INIT                          ; Call initialization routine
                                02B8  1305
                                02B8  1306 ;
                                02B8  1307 ; Load cpu microcode patches                                                 [98]
                                02B8  1308 ;
                                02B8  1309
                                02B8  1310          $DS_LOADPCS_S                               ; Loads patches from onboard    [78]
00000000'9F   00   FB           02B8                        CALLS   #0, a#DS$LOADPCS
                                02BF  1311                                                      ; supervisor image if a COMET   [78]
                                02BF  1312                                                      ; looks for a file if 11/ZZZ    [98]
          03BD   31             02BF  1313          BRW     BEGIN0
                                02C2  1314
                                02C2  1315 INI$BRK::
                          03    02C2  1316          BPT                                         ; Known XDELTA breakpoint
```

```
                        05  02C3  1317          RSB                                 ; Return
                            02C4  1318 DS$SOFTIPL5::
          00000000'8F   D5  02C4  1319          TSTL    #XDELBPT                    ; XDELTA present?
                    02  13  02CA  1320          BEQL    10$                         ; Branch if not
                    F4  10  02CC  1321          BSBB    INI$BRK                     ; Take XDELTA breakpoint
                        02  02CE  1322 10$:     REI                                 ; Return from IPL 5
```

```
                        02CF  1324          .SBTTL  USEP KERNEL ROUTINE
                   000002CF   1325          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                        02CF  1326 ;++
                        02CF  1327 ; FUNCTIONAL DESCRIPTION:
                        02CF  1328 ;
                        02CF  1329 ;       Entry from command mode, perform usermode only initialization.
                        02CF  1330 ;
                        02CF  1331 ; CALLING SEQUENCE:
                        02CF  1332 ;
                        02CF  1333 ;       CALLS   #0,DS$USERMODE
                        02CF  1334 ;
                        02CF  1335 ; INPUT PARAMETERS:     NONE
                        02CF  1336 ;
                        02CF  1337 ; IMPLICIT INPUTS:      NONE
                        02CF  1338 ;
                        02CF  1339 ; OUTPUT PARAMETERS:    NONE
                        02CF  1340 ;
                        02CF  1341 ; IMPLICIT OUTPUTS:     NONE
                        02CF  1342 ;
                        02CF  1343 ; COMPLETION CODES:     NONE
                        02CF  1344 ;
                        02CF  1345 ; SIDE EFFECTS:         NONE
                        02CF  1346 ;
                        02CF  1347 ;--
```

```
                     0FFC  02CF  1349 .ENTRY  DS$USERMODE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          00000899'EF   D4  02D1  1350        CLRL    DS$GL_FLAGS                ; Clear flags initially
                        02D7  1351        $CRETVA_S  L^SUPBUF                  ; GET ACCESS TO APT MAILBOX AREA
                  00    DD  02D7                   PUSHL   #0
                  00    DD  02D9                   PUSHL   #0
          0000019F'EF   7F  02DB                   PUSHAQ  L^SUPBUF
      800000C8'GF   03    FB  02E1                   CALLS   #3,G^SYS$CRETVA
                        02E8  1352        $CRETVA_S  L^PRGBUF                  ; GET ACCESS TO PROGRAM OVERLAY AREA
                  00    DD  02E8                   PUSHL   #0
                  00    DD  02EA                   PUSHL   #0
          000001A7'EF   7F  02EC                   PUSHAQ  L^PRGBUF
      800000C8'GF   03    FB  02F2                   CALLS   #3,G^SYS$CRETVA
                        02F9  1353        $SETEXV_S #1,COND_DEBUG              ; SET UP EXCEPTION HANDLER FOR DEBUG
                  00    DD  02F9                   PUSHL   #0
                  00    DD  02FB                   PUSHL   #0
          00000000'EF   DF  02FD                   PUSHAL  COND_DEBUG
                  01    DD  0303                   PUSHL   #1
      80000208'GF   04    FB  0305                   CALLS   #4,G^SYS$SETEXV
      6D  00000000'EF   DE  030C  1354        MOVAL   L^COND_HANDLR,(FP)       ; SET UP SUPERVISOR'S EXCEPTION HANDLER
                        0313  1355        $DELTVA_G       L^SSPROT             ; UNPROTECT SYSTEM SERVICE VECTORS.
80000110'GF   0000017A'EF   FA  0313                   CALLG   L^SSPROT,G^SYS$DELTVA
                        031E  1356        $CRETVA_G       L^SSPROT             ; RECAPTURE SPACE.
800000C8'GF   0000017A'EF   FA  031E                   CALLG   L^SSPROT,G^SYS$CRETVA
                        0329  1357        $DCLEXH_S EXIT$DESBLK                ; Declare image exit handler
          000001A0'EF   DF  0329                   PUSHAL  EXIT$DESBLK
      00000000'GF   01    FB  032F                   CALLS   #1,G^SYS$DCLEXH
  80000000'EF   0000'8F   28  0336  1358        MOVC3   #DS$K_SSSIZE, -          ; COPY STARLET SYSTEM SERVICE VECTORS.
          00000000'EF       033F  1359                  ^X80000000, L^DS$AQ_SYSSRV    ; IN ORDER TO GET THE CORRECT
                            0344  1360                                           ; WORD MASK FOR CALLx
                            0344  1361 ;
                            0344  1362 ; SET UP THE COPIED STARLET SYSTEM SERVICE VECTORS TO JMP TO 80000xxx          [77]
                            0344  1363 ; SO WE DON'T HAVE TO WORRY THAT THE SYSTEM SERVICE VECTOR CODE REMAINS IN     [77]
                            0344  1364 ; 3 PAGES (WHICH IS ALL WE HAVE ROOM FOR IN ENTRY)                            [77]
                            0344  1365 ;
      50    80000002 8F   D0  0344  1366        MOVL    #^X80000002,R0           ; BEGINNING OF SYSTEM SERVICE VECTOR  [77]
      51    00000000'EF   DE  034B  1367        MOVAL   DS$AQ_SYSSRV,R1          ; 10200                               [77]
                            0352  1368                                           ; QIOW                                [77]
       02 A1    9F17 8F   B0  0352  1369        MOVW    #^X9F17,2(R1)            ; JMP OPCODE, SKIPPING WORD MASK      [77]
            04 A1    50   D0  0358  1370        MOVL    R0,4(R1)                 ; ABSOLUTE ADDRESS 80000002           [77]
                            035C  1371                                           ; CALL_HANDL                          [77]
               50    0E   C0  035C  1372        ADDL2   #^XE,R0                  ; 80000010                            [77]
       10 A1    9F17 8F   B0  035F  1373        MOVW    #^X9F17,^X10(R1)         ; 10210                               [77]
            12 A1    50   D0  0365  1374        MOVL    R0,^X12(R1)              ; 80000010                            [77]
                            0369  1375                                           ; THE REST CAN BE LOOPED              [77]
               51    1A   C0  0369  1376        ADDL2   #^X1A,R1                 ; START AT 1021A                      [77]
               50    0A   C0  036C  1377        ADDL2   #^XA,R0                  ; JUMPING TO 8000001A                 [77]
          61    9F17 8F   B0  036F  1378 2$:    MOVW    #^X9F17,(R1)             ; SET UP THE JMP                      [77]
            02 A1    50   D0  0374  1379        MOVL    R0,2(R1)                 ; SET UP THE ADDRESS TO JMP TO        [77]
               50    08   C0  0378  1380        ADDL2   #8,R0                    ; NEXT...                             [77]
FFEA 51    08    00000000'8F   F1  037B  1381        ACBL    #DS$AQ_SSEND,#8,R1,2$    ; DONE ?                              [77]
                            0385  1382 ;
      50    0000018A'EF   9E  0385  1383        MOVAB   L^SYS$INPUT,R0           ; ADDRESS OF LOGICAL NAME
                  02A8    30  038C  1384        BSBW    TRAN_ASSIGN              ; TRANSLATE AND ASSIGN
                06 50     E8  038F  1385        BLBS    R0,100$                  ; Branch if success                   [87]
          0000041C'EF   17  0392  1386        JMP     10$                      ; Jump if failure                     [87]
                            0398  1387 100$:                                     ;                                     [87]
      0000089D'EF   51    B0  0398  1388        MOVW    R1,L^DS$GW_TTIN          ; SAVE INPUT CHANNEL NUMBER
```

```
                       7E   D5  039F  1389          TSTL    -(SP)                   ; Allocate space for characteristics
                       7E   7F  03A1  1390          PUSHAQ  -(SP)                   ; Allocate 8 more bytes and push addr
                       0C   DD  03A3  1391          PUSHL   #12                     ; Length of Device characteristics buf
                  50   6E   7E  03A5  1392          MOVAQ   (SP),R0                 ; Point to DIB descriptor
                                03A8  1393          $GETCHN_S R1,(R0),(R0)          ; Get characteristics of SYS$INPUT:
                       00   DD  03A8                        PUSHL   #0
                       00   DD  03AA                        PUSHL   #0
                       60   7F  03AC                        PUSHAQ  (R0)
                       60   3F  03AE                        PUSHAW  (R0)
                    7E   51   3C  03B0                        MOVZWL  R1,-(SP)
          00000000'GF   05   FB  03B3                        CALLS   #5,G^SYS$GETCHN
                  65 50   E9  03BA  1394          BLBC    R0,17$                  ; Branch if fails                       [87]
                       1F   BA  03BD  1395          POPR    #^M<R0,R1,R2,R3,R4>     ; R0=len, R1=adr  R2,R3,R4=DEVchar
          00000B8B'EF   52   D0  03BF  1396          MOVL    R2,DEVCHAR              ; Save device characteristics   [134] ;G:28
                48 52   0E   E1  03C6  1397 3$:      BBC     S^#DEV$V_FOD, R2, 5$    ; Branch if not command file            ;G:28
                                03CA  1398          $OPEN   DS$FAB_INPUT            ; Open command file
               0000006C'EF   DF  03CA                        PUSHAL  DS$FAB_INPUT
          00000000'GF   01   FB  03D0                        CALLS   #$$.TMP1,G^SYS$OPEN
                  42 50   E9  03D7  1399          BLBC    R0, 10$                 ; Exit if error
                                03DA  1400          $CONNECT        DS$RAB_INPUT    ; Connect record block
               000000BC'EF   DF  03DA                        PUSHAL  DS$RAB_INPUT
          00000000'GF   01   FB  03E0                        CALLS   #$$.TMP1,G^SYS$CONNECT
                  32 50   E9  03E7  1401          BLBC    R0, 10$                 ; Exit if error
                                03EA  1402          $CREATE DS$FAB_OUTPUT           ; Open output file
               00000100'EF   DF  03EA                        PUSHAL  DS$FAB_OUTPUT
          00000000'GF   01   FB  03F0                        CALLS   #$$.TMP1,G^SYS$CREATE
                  22 50   E9  03F7  1403          BLBC    R0, 10$                 ; Exit if error
                                03FA  1404          $CONNECT        DS$RAB_OUTPUT   ; Connect record block
               00000150'EF   DF  03FA                        PUSHAL  DS$RAB_OUTPUT
          00000000'GF   01   FB  0400                        CALLS   #$$.TMP1,G^SYS$CONNECT
                  12 50   E9  0407  1405          BLBC    R0, 10$                 ; Exit if error
          00000899'EF   16   E3  040A  1406          BBCS    S^#DS$V_BATCH, -
                             6E   0411  1407                   L^DS$GL_FLAGS, 20$   ; Set 'RMS' mode bit & skip
                                0412  1408 5$:
                                0412  1409
     50   00000194'EF   9E   0412  1410          MOVAB   L^SYS$OUTPUT,R0         ; ADDRESS OF LOGICAL NAME FOR OUTPUT
                     021B   30  0419  1411          BSBW    TRAN_ASSIGN             ; TRANSLATE AND ASSIGN
                                041C  1412 10$:
                  08 50   E8  041C  1413          BLBS    R0,15$                  ; Branch if no error
                     0214   31  041F  1414          BRW     DS$USERMODE_X           ; Exit w/ error
                                0422  1415
                5E   14   C0  0422  1416 17$:      ADDL    #20,SP                  ; Fix stack after $GETCHN error         [87]
                     F5   11  0425  1417          BRB     10$                     ; and leave.                            [87]
                                0427  1418 15$:
          0000089F'EF   51   B0  0427  1419          MOVW    R1,L^DS$GW_TTOUT        ; SAVE OUTPUT CHANNEL NUMBER
                                042E  1420
                       7E   D5  042E  1421          TSTL    -(SP)                   ; Allocate space for characteristics    [87]
                       7E   7F  0430  1422          PUSHAQ  -(SP)                   ; Allocate 8 more bytes and push addr   [87]
                       0C   DD  0432  1423          PUSHL   #12                     ; Length of Device characteristics buf  [87]
                  50   6E   7E  0434  1424          MOVAQ   (SP),R0                 ; Point to DIB descriptor               [87]
                                0437  1425          $GETCHN_S R1,(R0),(R0)          ; Get characteristics of SYS$OUTPUT:    [87]
                       00   DD  0437                        PUSHL   #0
                       00   DD  0439                        PUSHL   #0
                       60   7F  043B                        PUSHAQ  (R0)
                       60   3F  043D                        PUSHAW  (R0)
                    7E   51   3C  043F                        MOVZWL  R1,-(SP)
          00000000'GF   05   FB  0442                        CALLS   #5,G^SYS$GETCHN
```

```
                D6 50    E9  0449  1426          BLBC    R0,17$                          ; Branch if fails                                [87]
                   1F    BA  044C  1427          POPR    #^M<R0,R1,R2,R3,R4>             ; R0=len, R1=adr  R2,R3,R4=DEVchar               [87]
        00000B83'EF   53    7D  044E  1428          MOVQ    R3,DS$GL_TERMCHAR               ; Save terminal characteristics (R3,R4)          [87]
                              0455  1429
                50    7E    7E  0455  1430          MOVAQ   -(SP),R0                        ; Address of quad status block
                              0458  1431          $QIOW_S  ,L^DS$GW_TTIN,#IO$_SETMODE!IO$M_CTRLCAST,(R0),,,L^RCTRLC
                      7E    7C  0458                        CLRQ    -(SP)
                      7E    7C  045A                        CLRQ    -(SP)
                      00    DD  045C                        PUSHL   #0
             000009E9'EF    DF  045E                        PUSHAL  L^RCTRLC
                      7E    7C  0464                        CLRQ    -(SP)
                      60    7F  0466                        PUSHAQ  (R0)
          7E    0000'8F    3C  0468                        MOVZWL  #IO$_SETMODE!IO$M_CTRLCAST,-(SP)
       7E   0000089D'EF    3C  046D                        MOVZWL  L^DS$GW_TTIN,-(SP)
                      00    DD  0474                        PUSHL   #0
        00000000'GF   0C    FB  0476                        CALLS   #12,G^SYS$QIOW
                50    8E    7D  047D  1432          MOVQ    (SP)+,R0                        ; Get status
                              0480  1433 20$:
                              0480  1434          $CRETVA_S  L^SUPBUF                     ; GET ACCESS TO APT MAILBOX AREA
                      00    DD  0480                        PUSHL   #0
                      00    DD  0482                        PUSHL   #0
             0000019F'EF    7F  0484                        PUSHAQ  L^SUPBUF
        800000C8'GF   03    FB  048A                        CALLS   #3,G^SYS$CRETVA
                              0491  1435          $CRETVA_S  L^PRGBUF                     ; GET ACCESS TO PROGRAM OVERLAY AREA
                      00    DD  0491                        PUSHL   #0
                      00    DD  0493                        PUSHL   #0
             000001A7'EF    7F  0495                        PUSHAQ  L^PRGBUF
        800000C8'GF   03    FB  049B                        CALLS   #3,G^SYS$CRETVA
     7A 00000B8B'EF   14    E1  04A2  1436          BBC     S^#DEV$V_MBX, DEVCHAR, 19$                                              ;G:28
                              04AA  1437                                                  ; Branch if no mailbox               [134] ;G:28
                              04AA  1438          $GETJPI_S -                             ; Get current process id             [134] ;G:28
                              04AA  1439                  EFN = #1, -                                                                 ;G:28
                              04AA  1440                  PIDADR = PID, -                                                             ;G:28
                              04AA  1441                  ITMLST = ITEMS, -                                                           ;G:28
                              04AA  1442                  IOSB = IOSB                                                                 ;G:28
                      7E    7C  04AA                        CLRQ    -(SP)
             00000BAB'EF    7F  04AC                        PUSHAQ  IOSB
             00000B9B'EF    DF  04B2                        PUSHAL  ITEMS
                      00    DD  04B8                        PUSHL   #0
             00000B8F'EF    DF  04BA                        PUSHAL  PID
                      01    DD  04C0                        PUSHL   #1
          00000000'GF   07    FB  04C2                        CALLS   #7,G^SYS$GETJPI
                              04C9  1443          $FAO_S -                                ; Create global section name         [134] ;G:28
                              04C9  1444                  CTRSTR = CTRL_STR, -                                                        ;G:28
                              04C9  1445                  OUTBUF = GLBL_SECT_NAME, -                                                  ;G:28
                              04C9  1446                  OUTLEN = FAOLEN, -                                                          ;G:28
                              04C9  1447                  P1 = PIDNO                                                                  ;G:28
             00000B93'EF    DD  04C9                        PUSHL   PIDNO
             00000BB3'EF    7F  04CF                        PUSHAQ  GLBL_SECT_NAME
             00000BCF'EF    3F  04D5                        PUSHAW  FAOLEN
             00000BD3'EF    7F  04DB                        PUSHAQ  CTRL_STR
          00000000'GF   04    FB  04E1                        CALLS   #$$T2,G^SYS$FAO
00000BBB'EF    FE4B 8F    3C  04E8  1448          MOVZWL  #^XFE4B, GLBL_SECT_AREA ; Global area's starting address [134] ;G:28
00000BBF'EF    FE54 8F    3C  04F1  1449          MOVZWL  #^XFE54, GLBL_SECT_AREA+4 ; Global area's ending address [134] ;G:28
                              04FA  1450          $CRMPSC_S -                             ; Make it global                     [134] ;G:28
                              04FA  1451                  INADR = GLBL_SECT_AREA, -                                                   ;G:28
                              04FA  1452                  FLAGS = #SEC$M_WRT!SEC$M_GBL!SEC$M_PAGFIL, -                                ;G:28
```

```
                                04FA  1453                GSDNAM = GLBL_SECT_NAME, -                                  ;G:28
                                04FA  1454                CHAN = #0, -                                                ;G:28
                                04FA  1455                PAGCNT = #1                                                 ;G:28
                          7E    7C  04FA                  CLRQ    -(SP)
                          00    DD  04FC                  PUSHL   #0
                          01    DD  04FE                  PUSHL   #1
                       7E 00    3C  0500                  MOVZWL  #0,-(SP)
                          00    DD  0503                  PUSHL   #0
                          00    DD  0505                  PUSHL   #0
                00000BB3'EF     7F  0507                  PUSHAQ  GLBL_SECT_NAME
                00080009 8F     DD  050D                  PUSHL   #SEC$M_WRT!SEC$M_GBL!SEC$M_PAGFIL
                          00    DD  0513                  PUSHL   #0
                          00    DD  0515                  PUSHL   #0
                00000BBB'EF     7F  0517                  PUSHAQ  GLBL_SECT_AREA
           00000000'GF  0C      FB  051D                  CALLS   #12,G^SYS$CRMPSC
                                0524  1456                                                                            ;G:28
     1D 00000899'EF  1A      E1  0524  1457 19$:     BBC     #DS$V_NI,DS$GL_FLAGS,22$  ; BRANCH IF NOT OVER NI    [105]    ;G:28
                                052C  1458
                                052C  1459 ;;;
```

```
                00000000'EF   04   FB  05B6  1495          CallS   #4, Sys$CreLog          ; Create logical name                   [53]
                                  05BD  1496
                000001BA'EF   9F   05BD  1497          PUSHAB  L^DS$AT_DDIR+2          ; Address of area to save original default d
                        26    DD   05C3  1498          PUSHL   #38                     ; Length of area
                        6E    7F   05C5  1499          PUSHAQ  (SP)                    ; Address of buffer for old
                000001B8'EF   3F   05C7  1500          PUSHAW  L^DS$AT_DDIR            ; Address for length of buffer
                00000000'EF   7F   05CD  1501          PUSHAQ  DEF$Q_DIR               ; Address of descriptor of current dir
                00000000'EF   03   FB  05D3  1502          CALLS   #3,SYS$SETDDIR          ; Set new default directory
                                  05DA  1503
                                  05DA  1504 ;
                                  05DA  1505 ; Use the (all-new for V3.0 VMS) $GETSYI service to find the processor type
                                  05DA  1506 ; we are actually running on, and use this for the DSA$GL_SID longword.  If
                                  05DA  1507 ; the service fails for any reason, do the same thing we always did before, and
                                  05DA  1508 ; use the DS$GK_SID value, rather than fail the startup.
                                  05DA  1509 ;
                                  05DA  1510
                        7E    7C  05DA  1511          clrq    -(sp)                   ; Terminate argument list               [49]
                0000FE14'EF   9F   05DC  1512          pushab  L^dsa$gl_sid            ; Address of internal SID location      [49]
                    7E   1001 8F   B0  05E2  1513          movw    #syi$_sid, -(sp)        ; Push item code for full SID           [49]
                        7E    04   B0  05E7  1514          movw    #4, -(sp)               ; And length of buffer (longword)       [49]
                        50    6E   9E  05EA  1515          movab   (sp), r0                ; Remember where we are                 [49]
                                  05ED  1516          $getsyi_s       itmlst=(r0)     ; Get the information                   [49]
                                  05ED                        CLRQ    -(SP)
                                  05EF                        PUSHL   #0
                                  05F1                        PUSHAL  (r0)
                                  05F3                        PUSHL   #0
                                  05F5                        PUSHL   #0
                                  05F7                        PUSHL   #0
                00000000'GF   07   FB  05F9                        CALLS   #7,G^SYS$GETSYI
                            5E    10   C0  0600  1517          addl2   #4*4, sp                ; Clear off argument list               [49]
                                0B 50   E8  0603  1518          blbs    r0, 70$                 ; Continue if successful                [49]
0000FE14'EF     00000000'8F   D0  0606  1519          movl    #ds$gk_sid,dsa$gl_sid   ; Set pseudo SID value                  [49]
                                  0611  1520
0000FE00'EF     10003000 8F   C8  0611  1521 70$:     BISL2   #DSA$M_DFLTFLGS ! DSA$M_USER, -
                                  061C  1522                    L^DSA$GL_FLAGS          ; SET DEFAULT CONTROL FLAGS
                                  061C  1523
                00000000'EF   16  061C  1524          Jsb     L^MEMPOOL$INIT          ; SETUP MEMORY POOL                     [52]
                                  0622  1525
                        F9DB' 30  0622  1526          BSBW    INI$PTABLE              ; Clear P-table pointers
                                  0625  1527
                000008E1'EF   10  D0  0625  1528          MOVL    #16,L^DS$GL_RADIX       ; DEFAULT FOR EXAM/DEP FUNCTIONS
                000001B4'EF   5D  D0  062C  1529          MOVL    FP,L^SAVEFP             ; PRESERVE CLEAN STACK POINTER
                        0049  31  BW  0633  1530          BRW     BEGIN0                  ; AND START FRESH                       [52]
                                  0636  1531
                                  0636  1532 DS$USERMODE_X:
                                  04  0636  1533          RET                             ; RETURN
```

```
                                0637  1535 ;
                                0637  1536 ; TRANSLATE AND ASSIGN CHANNEL
                                0637  1537 ; INPUT:  R0    ADDR OF ASCIC LOGICAL NAME
                                0637  1538 ; OUTPUT: R0    SUCCESS/FAILURE
                                0637  1539 ;         R1    CHANNEL NUMBER
                                0637  1540 TRAN_ASSIGN:
                     5E   28    C2  0637  1541          SUBL    #40,SP
                          6E    9F  063A  1542          PUSHAB  (SP)                    ; MAKE DESCRIPTOR OF OUTPUT BUFFER
                          28    DD  063C  1543          PUSHL   #40                     ; LENGTH
                                    063E  1544                                          ; **** STACK CLEANED TO HERE AFTER CALL
                        01 A0   9F  063E  1545          PUSHAB  1(R0)                   ; INPUT DESCRIPTOR
                     7E   60    9A  0641  1546          MOVZBL  (R0),-(SP)              ; LENGTH
                          7E    D4  0644  1547          CLRL    -(SP)                   ; DUMMY ARG
                          7E    7C  0646  1548          CLRQ    -(SP)                   ; 2 DUMMY ARGS
                       14 AE    7F  0648  1549          PUSHAQ  20(SP)                  ; ADDRESS OF OUTPUT DESCRIPTOR
                       18 AE    DF  064B  1550          PUSHAL  24(SP)                  ; ADDRESS OF OUTPUT LENGTH
                       14 AE    7F  064E  1551          PUSHAQ  20(SP)                  ; ADDRESS OF INPUT DESCRIPTOR
          00000000'EF    08    FB  0651  1552          CALLS   #8,SYS$TRNLOG           ; TRANSLATE THE NAME
                       20 50    E9  0658  1553          BLBC    R0,120$                 ; BRANCH IF FAILED
                                    065B  1554
                  1B   08 AE    91  065B  1555          CMPB    8(SP),#^X1B             ; ESCAPE?
                          07    12  065F  1556          BNEQ    110$                    ; NO
                     6E   04    C2  0661  1557          SUBL    #4,(SP)                 ; SHORTEN STRING
                  04 AE   04    C0  0664  1558          ADDL    #4,4(SP)                ; UPDATE ADDRESS
                                    0668  1559 110$:
                          7E    D4  0668  1560          CLRL    -(SP)                   ; SPACE FOR CHANNEL NUMBER
                          7E    7C  066A  1561          CLRQ    -(SP)                   ; 2 DUMMY ARGS
                       08 AE    DF  066C  1562          PUSHAL  8(SP)                   ; ADDRESS TO RETURN CHANNEL NUMBER
                       10 AE    7F  066F  1563          PUSHAQ  16(SP)                  ; ADDRESS OF DESCRIPTOR
          00000000'EF    04    FB  0672  1564          CALLS   #4,SYS$ASSIGN           ; ASSIGN THE CHANNEL
                          02    BA  0679  1565          POPR    #^M<R1>                 ; GET CHANNEL ASSIGNED
                                    067B  1566 120$:
                  5E   30   C0    067B  1567          ADDL    #48,SP                  ; CLEANUP STACK
                               05 067E  1568          RSB                             ; RETURN
```

```
                        067F  1570          .SBTTL  COMMON KERNEL REFRESH ENTRY POINT.
                        067F  1571 ;++
                        067F  1572 ; FUNCTIONAL DESCRIPTION:
                        067F  1573 ;
                        067F  1574 ;       This routine is the basic loop of the supervisor.
                        067F  1575 ;       At begin it refreshes the image context and calls DS$CLI.
                        067F  1576 ;       The diagonostic is entered from the command processor.
                        067F  1577 ;
                        067F  1578 ; CALLING SEQUENCE:
                        067F  1579 ;
                        067F  1580 ;       ENTERED FROM "BOOT".
                        067F  1581 ;
                        067F  1582 ; INPUT PARAMETERS:     NONE
                        067F  1583 ;
                        067F  1584 ; IMPLICIT INPUTS:      NONE
                        067F  1585 ;
                        067F  1586 ; OUTPUT PARAMETERS:    NONE
                        067F  1587 ;
                        067F  1588 ; IMPLICIT OUTPUTS:     NONE
                        067F  1589 ;
                        067F  1590 ; COMPLETION CODES:     NONE
                        067F  1591 ;
                        067F  1592 ; SIDE EFFECTS:
                        067F  1593 ;
                        067F  1594 ;       Cleans stacks, registers, and ends up in Kernel mode.
                        067F  1595 ;       Call the CRD Image-Loading routine and CRD Initialization if
                        067F  1596 ;       either of the CRD bits have been set in the DSA flags.
                        067F  1597 ;--
```

```
                                  067F  BEGIN0: ; Only come here once ...                          [59]
                                  067F
                                  067F                                                  ; Only clear the structure once [123
                                  067F          Br_If_Not_MP    MP_clear_end            ; Branch if not an MP system
05000000 8F   00000000'8F   D1    067F          CMPL    #DS$GK_SID,#^X05000000          ; Branch if Scorpio [61]
                      27    13    068A          BEQL    30001$                          ; Yes, branch
06000000 8F   00000000'8F   D1    068C          CMPL    #DS$GK_SID,#^X06000000          ; Branch if Nautilus [61]
                      1A    13    0697          BEQL    30001$                          ; Yes, branch                  ;G:23
11000000 8F   00000000'8F   D1    0699          CMPL    #DS$GK_SID,#^X11000000          ; Branch if RRR [69]           ;G:20
                      0D    13    06A4          BEQL    30001$                                                         ;G:23
0A000000 8F   00000000'8F   D1    06A6          CMPL    #DS$GK_SID,#^X0A000000          ; Branch if LLL [69]           ;G:20
                      18    12    06B1          BNEQ    MP_clear_end
                                  06B3    30001$:
                      18    BB    06B3          PUSHR   #^M<R3,R4>                      ; Save registers                [110
          53  00000000'EF   DE    06B5          MOVAL   MP$R_ACTIVE_SERVICES,R3         ; Move address of Beggining     [110
          54  00000000'EF   DE    06BC          MOVAL   MP$R_INTERLOCK_END,R4           ; and end of strustures         [110
                      63    94    06C3  1$:     CLRB    (R3)                            ; then clear them               [110
                FA 53 54    F2    06C5          AOBLSS  R4,R3,1$                        ; all out.                      [110
                      18    BA    06C9          POPR    #^M<R3,R4>                      ; Restore registers             [110
                                  06CB
                                  06CB  MP_clear_end:
                                  06CB
                                  06CB
                                  06CB
                                  06CB          ;+
                                  06CB          ; Restore the CRD vectors that were originally
                                  06CB          ; set up in the DSVECS module.
                                  06CB          ;-
                                  06CB
          00000000'EF   00    FB  06CB          Calls   #0, L^DSR$Restore_CRD_Vectors
                                  06D2
                                  06D2          ;+
                                  06D2          ; Clear all the CRD flags in the DSA$GL_Flags longword
                                  06D2          ; Original flag settings still in RPB$ block, for offline only
                                  06D2          ;-
                                  06D2
0000FE00'EF   00070800 8F   CA    06D2          BICL2   #<DSA$M_CRD_Autotest_Off! -                    ; [ 100 ]
                                  06DD                    DSA$M_CRD_Menutest_Off! -                    ; [ 100 ]
                                  06DD                    DSA$M_CRD_Menutest_On ! -                    ; [ 100 ]
                                  06DD                    DSA$M_CRD_Trace>, -                          ; [ 100 ]
                                  06DD                  L^DSA$GL_Flags                                 ; [ 100 ]
                                  06DD
                                  06DD          ;+
                                  06DD          ;   Set up R4 to contain bits of interest from L^Boot$L_R5,
                                  06DD          ;   the RPB$ block, DSA$V_USER mode and LIB$GET_FOREIGN results.
                                  06DD          ;   A case statement will later take action based on these
                                  06DD          ;   combinations.
                                  06DD          ;
                                  06DD          ;   The last section, 3$, tests for being in USER mode (VMS) and
                                  06DD          ;   if so checks for the command string that envoked the VDS. The
                                  06DD          ;   command string will be $ESSAA/CRD.  The LIB$GET_FOREIGN
                                  06DD          ;   will return the /CRD part. This is tested against the required
                                  06DD          ;   text string to envoke CRD online.
                                  06DD          ;-
                                  06DD
                      54    D4    06DD          CLRL    R4                                                     ; [100] ;G:2
      02 00000879'EF  04    E1    06DF          BBC     #RPB$V_Diag, L^Boot$L_R5,1$     ; Check Diag bit       ; [100] ;G:3
```

```
                     54   96   06E7  1647          INCB    R4                                                      ; [100] ;G:3
          03 00000879'EF  00   E1   06E9  1648 1$:   BBC     #RPB$V_Menutest, L^Boot$L_R5,2$ ; Check Menutest_off bit; [100] ;G:3
                     54   02   80   06F1  1649          ADDB2   #2,R4                                                   ; [100] ;G:3
          03 00000879'EF  0F   E1   06F4  1650 2$:   BBC     #RPB$V_Autotest, L^Boot$L_R5,3$ ; Check Autotest_off bit; [100] ;G:3
                     54   04   80   06FC  1651          ADDB2   #4,R4                                                   ; [100] ;G:3
                                  06FF  1652 3$:                                                                        ;G:2
                                  06FF  1653                                    ;                           begin [124] ;G:2
                                  06FF  1654 ;        BBC     #DSA$V_User, L^DSA$GL_Flags,4$  ; Check VMS user mode   ; [100] ;G:3
                                  06FF  1655 ;        PUSHAB  CRD_DESC                        ; Now in USER mode      ; [100] ;G:2
                                  06FF  1656 ;        CALLS   #1,L^LIB$GET_FOREIGN            ; Get envoking command  ; [100] ;G:2
                                  06FF  1657 ;        BLBC    R0,4$                           ; Skip rest if not right; [100] ;G:2
                                  06FF  1658 ;        CMPC3   #4,a<CRD_DESC+4>,CRD_Switch_1   ; Compare command       ; [100] ;G:2
                                  06FF  1659 ;        TSTL    R0                              ; Check result          ; [100] ;G:2
                                  06FF  1660 ;        BNEQ    4$                              ; IF nonzero, no match  ; [100] ;G:2
                                  06FF  1661 ;        ADDB2   #8,R4                           ; A match, indicate so  ; [100] ;G:2
                                  06FF  1662                                    ;                             end [124] ;G:2
                                  06FF  1663 4$:
                                  06FF  1664          ;+
                                  06FF  1665          ; Test the R4 bits to determine which if any DSA$ bits to set and
                                  06FF  1666          ; any action to take.
                                  06FF  1667          ;-
                                  06FF  1668
        0F'8F    00   54   8F     06FF  1669          CASEB   R4,#0,#<<15$-10$>/2>-1                                  ; [100][102]  ;G:36
                                  0704  1670 10$:     .SIGNED_WORD -                                                  ; [100][102]  ;G:36
                         00D1'    0704  1671            100$-10$,-    ; 0, No bits set, skip CRD stuff                  ; [100] ;G:3
                                  0706  1672            100$-10$,-    ; 1, Only Diag bit set, skip CRD stuff            ; [100] ;G:3
                         00D1'    0706  1673             20$-10$,-    ; 2, ERROR                                        ; [100] ;G:3
                         0022'    0708  1674             21$-10$,-    ; 3, Diag and Menu offline, do action 21$         ; [100] ;G:3
                         003B'    070A  1675             20$-10$,-    ; 4, ERROR                                        ; [100] ;G:3
                         0022'    070C  1676             23$-10$,-    ; 5, Diag and auto, do action 23$                 ; [100] ;G:3
                         005B'    070E  1677             20$-10$,-    ; 6, ERROR                                        ; [100] ;G:3
                         0022'    0710  1678             20$-10$,-    ; 7, ERROR                                        ; [100] ;G:3
                         0022'    0712  1679             24$-10$,-    ; 8, Menu online, ignore other bits               ; [100] ;G:3
                         0068'    0714  1680             24$-10$,-    ; 9, Menu online, ignore other bits               ; [100] ;G:3
                         0068'    0716  1681             24$-10$,-    ;10, Menu online, ignore other bits               ; [100] ;G:3
                         0068'    0718  1682             24$-10$,-    ;11, Menu online, ignore other bits               ; [100] ;G:3
                         0068'    071A  1683             24$-10$,-    ;12, Menu online, ignore other bits               ; [100] ;G:3
                         0068'    071C  1684             24$-10$,-    ;13, Menu online, ignore other bits               ; [100] ;G:3
                         0068'    071E  1685             24$-10$,-    ;14, Menu online, ignore other bits               ; [100] ;G:3
                0068'    0068'    0720  1686             24$-10$      ;15, Menu online, ignore other bits               ; [100] ;G:3
                                  0724  1687 15$:
                          00   11  0724  1688          BRB     20$     ; This should never happen                        ; [100] ;G:3
                                  0726  1689
                                  0726  1690          ;+
                                  0726  1691          ; The case action routines
                                  0726  1692          ;-
                                  0726  1693
                                  0726  1694 20$:     $Print  #DS$K_Type_General_Error, #DS$K_PrintF, L^T_Both_CRD_Bits_Set ;[75]
                0000003E'EF       9F    0726                  PUSHAB  L^T_Both_CRD_Bits_Set
                     7E   01      9A    072C                  movzbl  #DS$K_PrintF, -(sp)
                     7E   03      98    072F                  cvtbl   #DS$K_Type_General_Error, -(sp)
          00000000'GF     03      FB    0732                  CALLS   #$$N+2, G^DSX$PRINT
               00000DEF'EF       17    0739  1695          JMP     L^DSV$Exit                      ; Bail out - exit to console    ; [75]
          05   0000FE17'EF       91    073F  1696 21$:     CMPB    DSA$GL_SID+3,#PR$_SID_TYP8SS    ; If SCORPIO, then               [119]
                          0A      12    0746  1697          BNEQ    22$                             ; don't do CRD MENU, instead     [119]
          00000879'EF     01      CA    0748  1698          BICL2   #RPB$M_MENUTEST,BOOT$L_R5       ; do normal VDS boot            [119]
                     0083         31    074F  1699          BRW     100$                            ; Continue with VDS boot        [119]
```

```
0000FE00'EF   00010000 8F   C8   0752  1700 22$:    BISL2   #DSA$M_CRD_Menutest_Off, L^DSA$GL_Flags                    ; [ 100 ]
                       39   11   075D  1701         BRB     30$                                                       ; [ 100 ]
0000FE00'EF   00000800 8F   C8   075F  1702 23$:    BISL2   #DSA$M_CRD_Autotest_Off, L^DSA$GL_Flags                   ; [ 100 ]
                       2C   11   076A  1703         BRB     30$                                                       ; [ 100 ]
0000FE00'EF   00040000 8F   C8   076C  1704 24$:    BISL2   #DSA$M_CRD_Menutest_On,  L^DSA$GL_Flags                   ; [ 100 ]
0000005C'FF   00000058'EF   29   0777  1705         CMPC3   CRD_DESC,a<CRD_DESC+4>,CRD_Switch_2; Compare command      ; [ 100 ]
              00000004'EF        0782
                       50   D5   0787  1706         TSTL    R0                                  ; Check result        ; [ 100 ]
                       0B   12   0789  1707         BNEQ    25$                                 ; IF nonzero, no match; [ 100 ][11
0000FE00'EF   00020000 8F   C8   078B  1708         BISL2   #DSA$M_CRD_Trace, L^DSA$GL_Flags                          ; [ 1000 ]
                       00   11   0796  1709 25$:    BRB     30$                                                       ; [ 100 ]
                                 0798  1710
```

```
                                0798  1712          ;+
                                0798  1713          ; Take this path to do CRD stuff
                                0798  1714          ;-
                                0798  1715
          00000000'EF    00   FB  0798  1716 30$:    CALLS   #0, L^DSR$Load_CRD      ; Load the CRD image file       [59]
                                079F  1717
                                079F  1718          ;+
                                079F  1719          ; Call the main CRD routine and have it initialize itself. This routine
                                079F  1720          ; will return failure when its initialization fails for some reason. It
                                079F  1721          ; will return success otherwise. If it fails, the DS image will exit.
                                079F  1722          ; If it succeeds, the DS will continue normally.
                                079F  1723          ;-
                                079F  1724
                          00    DD  079F  1725          PUSHL   #0                      ; Push 0 as param-4             [59]
                          00    DD  07A1  1726          PUSHL   #0                      ; Push 0 as param-3             [59]
                          00    DD  07A3  1727          PUSHL   #CRD$K_CRD_Initialization ; Push Initialization as param-2      [59]
                          00    DD  07A5  1728          PUSHL   #CRD$K_Hook_Point       ; Push Hook_Point as param-1    [59]
          00000000'FF     04    FB  07A7  1729          CALLS   #4, @L^DS$GA_CRD_DS_Interface                           ; [59]
                       24 50    E8  07AE  1730          BLBS    R0, 100$                ; If routine returns success, branch    [59]
                                07B1  1731
                                07B1  1732          ;+
                                07B1  1733          ; The init routine failed - print an error message and exit the VDS     [74]
                                07B1  1734          ; Clear the DSA Flags for CRD stuff                                     ; [
                                07B1  1735          ;-
                                07B1  1736
    0000FE00'EF    00050802 8F   CA  07B1  1737          BICL2   #<DSA$M_Binary          ! -                             ; [ 100 ]
                                07BC  1738                    DSA$M_CRD_Menutest_Off ! -                            ; [ 100 ]
                                07BC  1739                    DSA$M_CRD_Autotest_Off ! -                            ; [ 100 ]
                                07BC  1740                    DSA$M_CRD_Menutest_On >, -                            ; [ 100 ]
                                07BC  1741                  L^DSA$GL_Flags                                          ; [ 100 ]
                                07BC  1742
                                07BC  1743          $Print  #DS$K_Type_General_Error, -     ; Print an error message         [74]
                                07BC  1744                    #DS$K_PrintF, -           ; . . . use PrintF               [74]
                                07BC  1745                    L^GT_Menu_Error           ; . . . the format string        [74]
                00000000'EF     9F  07BC                    PUSHAB  L^GT_Menu_Error
                     7E     01  9A  07C2                    movzbl  #DS$K_PrintF, -(sp)
                     7E     03  98  07C5                    cvtbl   #DS$K_Type_General_Error, -(sp)
             00000000'GF     03  FB  07C8                    CALLS   #$$N+2, G^DSX$PRINT
                                07CF  1746
                00000DEF'EF     16  07CF  1747          JSb     DSV$Exit                ; Exit if routine failed        [59]
                                07D5  1748
    0000FE00'EF    80050800 8F   D3  07D5  1749 100$:   BITL    #DSA$M_CRD_AUTOTEST_OFF!-; Don't display legal statement if     [95]
                                07E0  1750                    DSA$M_CRD_MENUTEST_OFF! -; CRD MENU or AUTO or if APT           [95]
                                07E0  1751                    DSA$M_APT! -
                                07E0  1752                    DSA$M_CRD_MENUTEST_ON, -                        ; 100
                                07E0  1753                    L^DSA$GL_FLAGS ;                                [95]
                          31    12  07E0  1754          BNEQ    150$                    ;                               [95]
                                07E2  1755          $PRINT  #DS$K_TYPE_DS_START, -  ; Display statement from legal dept.    [92]
                                07E2  1756                    #DS$K_PRINTF,-          ; ... use PRINTF                [92]
                                07E2  1757                    DS$GT_LEGAL             ; ... format statement          [92]
                0000006E'EF     9F  07E2                    PUSHAB  DS$GT_LEGAL
                     7E     01  9A  07E8                    movzbl  #DS$K_PRINTF, -(sp)
                     7E     1D  98  07EB                    cvtbl   #DS$K_TYPE_DS_START, -(sp)
             00000000'GF     03  FB  07EE                    CALLS   #$$N+2, G^DSX$PRINT
    0000002E'EF    0000000F'EF  D0  07F5  1758                    MOVL    L^DS$GQ_DAYTIM+15, -    ; GET YEAR FROM LINK DATE       [109
                                0800  1759                      L^DS$GT_COPYEAR       ; Insert year for link          [109
                                0800  1760          $PRINT  #DS$K_TYPE_DS_START, -  ; Display statement from legal dept.    [109
```

```
                          0800  1761                    #DS$K_PRINTF,-          ; ... use PRINTF                        [109
                          0800  1762                    DS$GT_COPYRIGHT         ; ... format statement                  [109
          00000000'EF   9F  0800                        PUSHAB  DS$GT_COPYRIGHT
              7E    01  9A  0806                        movzbl  #DS$K_PRINTF, -(sp)
              7E    1D  98  0809                        cvtbl   #DS$K_TYPE_DS_START, -(sp)
    00000000'GF     03  FB  080C                        CALLS   #$$N+2, G^DSX$PRINT
                          0813  1763
                          0813  1764 150$:   $Print    #DS$K_TYPE_DS_START,-   ; Print startup message                [56]
                          0813  1765                      #DS$K_PRINTF,-        ; ... use PrintF
                          0813  1766                      L^DS$GT_START,-       ; ... the format string
                          0813  1767                      #0                    ; ... the current time
                    00    DD  0813                        PUSHL   #0
          00000000'EF   9F  0815                        PUSHAB  L^DS$GT_START
              7E    01  9A  081B                        movzbl  #DS$K_PRINTF, -(sp)
              7E    1D  98  081E                        cvtbl   #DS$K_TYPE_DS_START, -(sp)
    00000000'GF     04  FB  0821                        CALLS   #$$N+2, G^DSX$PRINT
                          0828  1768
          00000000'EF   16  0828  1769          JSB     SCRIPT$INIT             ; Init script flags and pointers        [44]
                    52    D4  082E  1770          CLRL    R2                      ; START AT EFN ZERO
                          0830  1771
                          0830  1772 200$:   $CLREF_S          R2              ; CLEAR THE EVENT FLAG
                    52    DD  0830                        PUSHL   R2
    00000000'GF     01    FB  0832                        CALLS   #1,G^SYS$CLREF
              F3 52   3F    F3  0839  1773          AOBLEQ  #63, R2, 200$           ; CLEAR ALL 64 FLAGS
                          083D  1774
                          083D  1775          $DS_CntrlC_S    Disabl=#0       ; Re-enable catching of ^C's - clear    [66]
                    00    DD  083D                        PUSHL   #0
                    00    DD  083F                        PUSHL   #0
        00000000'9F     02  FB  0841                        CALLS   #2, a#DS$CNTRLC
                          0848  1776                                          ; ... the Disable flag.                 [66]
                          0848  1777
    0000FE00'EF   01    1F  EF  0848  1778          ExTZV   #Dsa$V_Apt, #1, -       ; Extract the APT mode bit              [60]
                    50        0850  1779                    L^DSA$GL_Flags, R0    ; .. into R0                            [60]
        000008F3'EF     50  90  0851  1780          MovB    R0, -                   ; .. and copy byte into flag            [60]
                          0858  1781                    L^DS$GB_Inhibit_Naming;                                         [60]
                          0858  1782
                          0858  1783          Br_If_Not_APT   Begin           ; Branch around if not in APT-mode      [59]
          0000FE00'EF   1F  E1  0858                    BBC     #DSA$V_Apt, -           ; If bit not set, branch
                    0D        085F                            L^DSA$GL_Flags, Begin   ;
                          0860  1784          $PRINTI_S DS$GT_PROMPT          ; Type prompt on APT start
              00000000'EF   9F  0860                        PUSHAB  DS$GT_PROMPT
          00000000'EF   01    FB  0866                    CALLS   #$$N, DSX$PRINTI
```

```
                                086D  1786 BEGIN::
                                086D  1787 BEGIN_BLISS::
                          1B    10  086D  1788          BSBB    INIT_CONTEXT            ; REINITIALIZE IMAGE CONTEXT
                          7E    DC  086F  1789          MOVPSL  -(SP)                   ; CURRENT PSL
                7E  00000888'EF   9E  0871  1790          MOVAB   20$,-(SP)               ; CURRENT PC
                        7FFF  8F  BB  0878  1791          PUSHR   #^X7FFF                 ; SAVE R0-R14
                00000000'EF   6E  FA  087C  1792          CALLG   (SP),DS$CLI             ; ENTER COMMAND INTERPRETER
                        7FFF  8F  BA  0883  1793          POPR    #^X7FFF                 ; RESTORE R0-R14
                                  02  0887  1794          REI                             ; NOTE: USES PC/PSL SAVED/MODIFIED ABOVE
                                      0888  1795
                          E3    11  0888  1796 20$:       BRB     BEGIN                   ; IF DS$CLI EVER RETURNS DO IT AGAIN
```

```
                        088A  1798          .SBTTL  INIT_CONTEXT    Initialize process context
                        088A  1799 ;++
                        088A  1800 ; FUNCTIONAL DESCRIPTION:
                        088A  1801 ;
                        088A  1802 ;       This routine is called to reinitialize the entire process context.
                        088A  1803 ;       It initializes the contents of the registers and resets the stack
                        088A  1804 ;       pointers.  In stand alone it also resets the SYSTEM, P0, and P1 base
                        088A  1805 ;       and length registers, as well as all the stack pointers in all modes.
                        088A  1806 ;       It does not effect the mapping or availablity of memory or the current
                        088A  1807 ;       state of memory management.
                        088A  1808 ;
                        088A  1809 ; CALLING SEQUENCE:
                        088A  1810 ;
                        088A  1811 ;       BSBW    INIT_CONTEXT
                        088A  1812 ;
                        088A  1813 ; INPUT PARAMETERS:     NONE
                        088A  1814 ;
                        088A  1815 ; IMPLICIT INPUTS:
                        088A  1816 ;
                        088A  1817 ;       Initial contents of system, P0, and P1 base registers, stack pointers
                        088A  1818 ;       etc.
                        088A  1819 ;
                        088A  1820 ; OUTPUT PARAMETERS:    NONE
                        088A  1821 ;
                        088A  1822 ; IMPLICIT OUTPUTS:
                        088A  1823 ;
                        088A  1824 ;       The stack and frame pointer are reset it initial values, all registers
                        088A  1825 ;       are modified.
                        088A  1826 ;
                        088A  1827 ; COMPLETION CODES:     NONE
                        088A  1828 ;
                        088A  1829 ; SIDE EFFECTS:
                        088A  1830 ;
                        088A  1831 ;       The entire process context is refreshed.
                        088A  1832 ;--
```

```
                                088A  1834 INIT_CONTEXT::
                                088A  1835          QA_MAIN Kernel_Init_Context     ;                                                [44]
                          08    DD  088A                  PUSHL   #DSQA$K_Kernel_Init_Context
              00000000'9F 01    FB  088C                  CALLS   #1, a#QA$MAIN
              0000FE00'EF 1C    E1  0893  1836            BBC     #DSA$V_USER, L^DSA$GL_FLAGS -
                          03        089A  1837                    , 10$                   ; BRANCH IF STAND ALONE
                       00F9 31      089B  1838            BRW     INIT_USERMODE           ; DO USER MODE STUFF
                                089E  1839
                                089E  1840 10$:
        50    00000000'EF 9E      089E  1841            MOVAB   DS$AX_SOFTPCB, R0               ; GET BASE ADDRESS.
     60 A0    00660000 8F D0      08A5  1842            MOVL    #IOC$K_DIAGPID, PCB$L_PID(R0)   ; SET UP DIAG PID.
           10 A0    10 A0 DE      08AD  1843            MOVAL   PCB$L_ASTQFL(R0), PCB$L_ASTQFL(R0) ; INIT AST DELIVERY QUEUE.
           14 A0    10 A0 DE      08B2  1844            MOVAL   PCB$L_ASTQFL(R0), PCB$L_ASTQBL(R0) ;
                    0C A0 94      08B7  1845            CLRB    PCB$B_ASTACT(R0)                ; Clear AST actives
                                08BA  1846 ;
                                08BA  1847 ; Initialize JIB and ARB
                                08BA  1848 ;
        53    00000000'EF 9E      08BA  1849            MOVAB   DS$AX_JIB,R3            ; Point to JIB
              78 A0    53 D0      08C1  1850            MOVL    R3,PCB$L_JIB(R0)        ; Make PCB point to JIB
        52    00000000'EF 9E      08C5  1851            MOVAB   DS$AX_ARB,R2            ; Point to ARB
           0084 C0     52 D0      08CC  1852            MOVL    R2,PCB$L_ARB(R0)        ; Make PCB point to ARB
                 63    52 D0      08D1  1853            MOVL    R2,JIB$L_ARB(R3)        ; Make JIB point to ARB
                                08D4  1854
        50    00000000'EF DE      08D4  1855            MOVAL   PHD_BASE, R0                    ; GET BASE POINTER.
                    00C0 C0 8ED0  08DB  1856            POPL    PHD$L_PC(R0)                    ; SAVE RETURN PC.
              00000993'EF DC      08E0  1857            MOVPSL  PSL_SAVE                        ; GET CURRENT PSL IN R4 [68]
                                08E6  1858            CMK     REFRESH
                          7E    DC  08E6                  MOVPSL  -(SP)
                       06 6E    1A  E0  08E8              BBS     #PSL$V_IS, (SP), 30002$
                          8E    D4  08EC                  CLRL    (SP)+
                          05    BC  08EE                  CHMK    #CMK$_REFRESH
                          06    11  08F0                  BRB     30003$
                    0000097B'EF 16  08F2        30002$:   JSB     CMK$REFRESH
                                08F8        30003$:
           04       00000000'8F DA  08F8  1859            MTPR    #ISTKPTR, #PR$_ISP              ; RESET INTERRUPT STACK.
           5E       00000000'EF DE  08FF  1860            MOVAL   KSTKPTR, SP                     ; RESET KERNEL STACK POINTER.
                       00    5E DA  0906  1861            MTPR    SP, #PR$_KSP                    ; INITIALIZE KERNEL STACK
                     78 A0    00 DB  0909  1862            MFPR    #PR$_KSP, PHD$L_KSP(R0)         ; KERNEL STACK POINTER.
           01       00000000'8F DA  090D  1863            MTPR    #ESTKPTR, #PR$_ESP              ; INITIALIZE EXECUTIVE STACK
                     7C A0    01 DB  0914  1864            MFPR    #PR$_ESP, PHD$L_ESP(R0)         ; EXECUTIVE STACK POINTER.
           02       00000000'8F DA  0918  1865            MTPR    #SSTKPTR, #PR$_SSP              ; INITIALIZE SUPERVISOR STACK
                    0080 C0    02 DB  091F  1866            MFPR    #PR$_SSP, PHD$L_SSP(R0)         ; SUPERVISOR STACK POINTER.
           03       00000000'8F DA  0924  1867            MTPR    #USTKPTR, #PR$_USP              ; INITIALIZE USER STACK
                    0084 C0    03 DB  092B  1868            MFPR    #PR$_USP, PHD$L_USP(R0)         ; USER STACK POINTER.
                                0930  1869
                          5D    D4  0930  1870            CLRL    FP                              ; NO PREVIOUS STACK FRAME
                                0932  1871
           51       00B8 C0    DE  0932  1872            MOVAL   PHD$L_R12(R0),R1                ; POINT TO REGISTER AREA FOR R12
           61        CCCCCCCC 8F D0  0937  1873            MOVL    #^XCCCCCCCC,(R1)                ; STORE INITIAL AP
71         61        11111111 8F C3  093E  1874 20$:       SUBL3   #^X11111111,(R1),-(R1)          ; STORE NEXT LOWER REGISTER
                          F6    12  0946  1875            BNEQ    20$                             ; CONTINUE UNTIL R0 STORED
                    00C4 C0    DC  0948  1876            MOVPSL  PHD$L_PSL(R0)                   ; PROCESSOR STATUS LONGWORD.
                 00C8 C0    08 DB  094C  1877            MFPR    #PR$_P0BR, PHD$L_P0BR(R0)       ; P0 BASE REGISTER.
                 00CC C0    09 DB  0951  1878            MFPR    #PR$_P0LR, PHD$L_P0LRASTL(R0)   ; P0 LENGTH REGISTER.
                       13    04 DA  0956  1879            MTPR    #4,#PR$_ASTLVL          ; Set ASTLVL
                 00CF C0    04 90  0959  1880            MOVB    #4, PHD$B_ASTLVL(R0)            ; SET NO PENDING AST.
                 00D0 C0    0A DB  095E  1881            MFPR    #PR$_P1BR, PHD$L_P1BR(R0)       ; P1 BASE REGISTER.
```

```
                00D4 C0    0B    DB   0963  1882          MFPR    #PR$_P1LR, PHD$L_P1LR(R0)       ; P1 LENGTH REGISTER.
          0088 C0   12345678 8F    D0   0968  1883          MOVL    #^X12345678, PHD$L_R0(R0)       ; ^X12345678 INTO R0 IMAGE.
                       12    00    DA   0971  1884          MTPR    #0,#PR$_IPL                     ; NOW LOWER IPL
                                        0974  1885
                50     00C0 C0    D0   0974  1886          MOVL    PHD$L_PC(R0),R0                 ; GET RETURN ADDRESS
                             29    11   0979  1887          BRB     INIT_COMMON                     ; DO COMMON INITIALIZATION
                                        097B  1888 CMK$REFRESH::
                       12    1F    DA   097B  1889          MTPR    #31,#PR$_IPL            ; Set IPL to 31 so REI doesn't fault
            04 AE    001F0000 8F    D0   097E  1890          MOVL    #PSL$M_IPL,4(SP)                ; SET TO KERNEL MODE & IPL=31
            04 00000993'EF    04    E1   0986  1891          BBC     #PSL$V_TBIT,PSL_SAVE,10$        ;IF T-BIT SET IN OLD PSL[68]
                     04 AE    10    C8   098E  1892          BISL    #1@PSL$V_TBIT,4(SP)             ;THEN SET IT IN NEW PSL [68]
                                        0992  1893 10$:                                            ;ELSE DON'T.            [68]
                                   02   0992  1894          REI
                                        0993  1895
                           00000000     0993  1896 PSL_SAVE:      .LONG   0                       ;TEMP. STORAGE FOR PSL  [68]
```

```
                                0997  1898 INIT_USERMODE:
                     50 8ED0    0997  1899          POPL    R0                              ; GET RETURN ADDRESS
          5D 000001B4'EF  D0    099A  1900          MOVL    L^SAVEFP,FP                     ; RESET FRAME POINTER
               5E    5D  D0    09A1  1901          MOVL    FP,SP                           ; DITTO FOR STACK POINTER
                                09A4  1902
                                09A4  1903 INIT_COMMON:
                                09A4  1904
          000008F2'EF   94      09A4  1905          clrb    L^ds$gb_typecode                ; Clear the SET BINARY code     [46]
                        01      09AA  1906          PUSHR   #^M<R0>                         ; Save return address
                                09AC  1907          $SETAST_S       #1                      ; Enable AST delivery
                        01      09AC                PUSHL   #1
          00000000'GF   01      09AE                CALLS   #1,G^SYS$SETAST
                        01      09B5  1908          POPR    #^M<R0>                         ; Restore return address
          000009C1'EF   00      09B7  1909          CALLS   #0,10$                          ; SETUP DUMMY CALL FRAME
                      FEAC      09BE  1910          BRW     BEGIN                           ; IN THE UNLIKELY EVENT IT RETURN
                                09C1  1911
                      0000      09C1  1912 10$:     .WORD   0                               ; SAVE NO REGISTERS
                        50      09C3  1913          PUSHL   R0                              ; PUT RETURN ADDRESS BACK
          6D 00000000'EF        09C5  1914          MOVAB   L^COND_HANDLR,(FP)              ; STACK BASED CONDITION HANDLER
                                09CC  1915
          7E CCCCCCCC 8F        09CC  1916          MOVL    #^XCCCCCCCC,-(SP)               ; SET INITIAL AP
    7E    6E 11111111 8F        09D3  1917 20$:     SUBL3   #^X11111111,(SP),-(SP)          ; NEXT LOWER REGISTER
                        F6      09DB  1918          BNEQ    20$                             ; UNTIL R0
                      1FFF 8F   09DD  1919          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
          50 12345678 8F        09E1  1920          MOVL    #^X12345678,R0                  ; INITIAL R0
                                09E8  1921
                        05      09E8  1922          RSB                                     ; RETURN ALL REFRESHED
```

```
                                09E9  1924          .SBTTL  CONTROL-C AST HANDLER
                                09E9  1925 ;++
                                09E9  1926 ; FUNCTIONAL DESCRIPTION:
                                09E9  1927 ;
                                09E9  1928 ;       This is the usermode CONTROL-C handler. It sets
                                09E9  1929 ;       DS$V_CTRLC in DS$GL_FLAGS and renables the ^C AST.
                                09E9  1930 ;
                                09E9  1931 ; CALLING SEQUENCE:
                                09E9  1932 ;
                                09E9  1933 ;       CALLS   #0,RCTRLC
                                09E9  1934 ;
                                09E9  1935 ; INPUT PARAMETERS:     NONE
                                09E9  1936 ;
                                09E9  1937 ; IMPLICIT INPUTS:      NONE
                                09E9  1938 ;
                                09E9  1939 ; OUTPUT PARAMETERS:    NONE
                                09E9  1940 ;
                                09E9  1941 ; IMPLICIT OUTPUTS:     NONE
                                09E9  1942 ;
                                09E9  1943 ; COMPLETION CODES:     NONE
                                09E9  1944 ;
                                09E9  1945 ; SIDE EFFECTS:
                                09E9  1946 ;
                                09E9  1947 ;       SETS DS$V_CTRLC IN DS$GL_FLAGS
                                09E9  1948 ;--
                                09E9  1949
                          0000  09E9  1950 .ENTRY  RCTRLC,^M<>
5E    00000074 8F    C2   09EB  1951          SUBL    #DIB$K_LENGTH,SP        ; Space for device characteristics
                6E    9F   09F2  1952          PUSHAB  (SP)                    ; Make descriptor of DIB
      00000074 8F    DD   09F4  1953          PUSHL   #DIB$K_LENGTH           ; Set length
           50  5E    D0   09FA  1954          MOVL    SP,R0                   ; Save address of data area
                           09FD  1955          $GETCHN_S L^DS$GW_TTIN,(R0),(R0); Get characterictics
                   00    DD   09FD                   PUSHL   #0
                   00    DD   09FF                   PUSHL   #0
                   60    7F   0A01                   PUSHAQ  (R0)
                   60    3F   0A03                   PUSHAW  (R0)
7E    0000089D'EF    3C   0A05                   MOVZWL  L^DS$GW_TTIN,-(SP)
00000000'GF    05    FB   0A0C                   CALLS   #5,G^SYS$GETCHN
         5E    08    C0   0A13  1956          ADDL    #8,SP                   ; Remove descriptor
           11  50    E8   0A16  1957          BLBS    R0,10$                  ; Branch if ok
                           0A19  1958          ERRSUP_S
      0000016B'EF    DF   0A19                   PUSHAL  $MODULE
                   00    DD   0A1F                   PUSHL   #0
                   01    DD   0A21                   PUSHL   #$ER
00000000'EF    03    FB   0A23                   CALLS   #3, DS_ERRSUP
       30 6E    14    E1   0A2A  1959 10$:     BBC     #DEV$V_MBX, -
                           0A2E  1960                  DIB$L_DEVCHAR(SP),30$   ; Branch if not mailbox
             08 AE    B7   0A2E  1961 20$:     DECW    DIB$L_DEVDEPEND(SP)     ; Decrement count of messaged
                2B    19   0A31  1962          BLSS    30$                     ; Finish
                           0A33  1963          $QIOW_S ,L^DS$GW_TTIN,#IO$_READVBLK!IO$M_NOW,,,,L^DS$GT_BUFFER,#80
                   7E    7C   0A33                   CLRQ    -(SP)
                   7E    7C   0A35                   CLRQ    -(SP)
      00000050 8F    DD   0A37                   PUSHL   #80
      00000903'EF    DF   0A3D                   PUSHAL  L^DS$GT_BUFFER
                   7E    7C   0A43                   CLRQ    -(SP)
                   00    DD   0A45                   PUSHL   #0
    7E    0000'8F    3C   0A47                   MOVZWL  #IO$_READVBLK!IO$M_NOW,-(SP)
```

```
                7E   0000089D'EF   3C   0A4C                        MOVZWL  L^DS$GW_TTIN,-(SP)
                             00    DD   0A53                        PUSHL   #0
                00000000'GF   0C   FB   0A55                        CALLS   #12,G^SYS$QIOW
                              D0   11   0A5C  1964          BRB     20$                             ; Try again
                                        0A5E  1965 30$:
                                        0A5E  1966          $QIOW_S ,L^DS$GW_TTIN,#IO$_SETMODE!IO$M_CTRLCAST,,,,RCTRLC
                              7E   7C   0A5E                        CLRQ    -(SP)
                              7E   7C   0A60                        CLRQ    -(SP)
                              00   DD   0A62                        PUSHL   #0
                           82 AF   DF   0A64                        PUSHAL  RCTRLC
                              7E   7C   0A67                        CLRQ    -(SP)
                              00   DD   0A69                        PUSHL   #0
                    7E   0000'8F   3C   0A6B                        MOVZWL  #IO$_SETMODE!IO$M_CTRLCAST,-(SP)
                7E   0000089D'EF   3C   0A70                        MOVZWL  L^DS$GW_TTIN,-(SP)
                             00    DD   0A77                        PUSHL   #0
                00000000'GF   0C   FB   0A79                        CALLS   #12,G^SYS$QIOW
                00000899'EF   01   C8   0A80  1967          BISL2   #DS$M_CTRLC,L^DS$GL_FLAGS       ; SET CONTROL-C FLAG
                                        0A87  1968
                                   04   0A87  1969          RET                                     ; RETURN
```

```
                        1971          .SBTTL  KEYBOARD CHECK ROUTINE
                        1972 ;**
                        1973 ; FUNCTIONAL DESCRIPTION:
                        1974 ;
                        1975 ;       This routine is called to check for ^C, if it has been typed
                        1976 ;       control is passes to command mode by calling DS$CLI.
                        1977 ;       If Any ASTS are active (S/A only) the call is ignored
                        1978 ;
                        1979 ; CALLING SEQUENCE:
                        1980 ;
                        1981 ;       BSBW    KB_CHECK
                        1982 ;
                        1983 ; INPUT PARAMETERS:
                        1984 ;
                        1985 ;       NONE
                        1986 ;
                        1987 ; IMPLICIT INPUTS:
                        1988 ;
                        1989 ;       NONE
                        1990 ;
                        1991 ; OUTPUT PARAMETERS:
                        1992 ;
                        1993 ;       NONE
                        1994 ;
                        1995 ; IMPLICIT OUTPUTS:
                        1996 ;
                        1997 ;       NONE
                        1998 ;
                        1999 ; COMPLETION CODES:
                        2000 ;
                        2001 ;       NONE
                        2002 ;
                        2003 ; SIDE EFFECTS:
                        2004 ;
                        2005 ;       NONE
                        2006 ;--
```

```
                                0A88  2008 KB_CHECK::                                                                  ;G:36
      03 00000899'EF   19   E0  0A88  2009          BBS     #DS$V_TIMED,DS$GL_FLAGS,1$ ; TIMED?                  [128] ;G:16
                     004D   31  0A90  2010          BRW     20$                        ; Branch if no /TIME          [128] ;G:16
                                0A93  2011                                                                                ;G:16
                       03   BB  0A93  2012 1$:      PUSHR   #^M<R0,R1>                 ; Used by GET_TIME            [128] ;G:17
              00000000'EF   DF  0A95  2013          PUSHAL  POLL_TIME                  ; Address to store result     [128] ;G:16
         00000C99'EF   01   FB  0A9B  2014          CALLS   #1,[illegible]_TIME              ; Use TODR to calculate time  [128] ;G:16
                       03   BA  0AA2  2015          POPR    #^M[illegible]0,R1>               ; Used by GET_TIME            [128] ;G:17
                                0AA4  2016                                                                                ;G:16
  00000004'EF   00000004'EF   D1  0AA4  2017          CMPL    L^DUE_TIME+4, -            ; See if time is due          [128] ;G:16
                                0AAF  2018                  L^POLL_TIME+4              ; Compare high order of quad  [128] ;G:16
                       0F   1F  0AAF  2019          BLSSU   10$                        ; If LSSU, then time is due   [128] ;G:16
                       2D   1A  0AB1  2020          BGTRU   20$                        ; If GTRU, then time is not due [128] ;G:16
  00000000'EF   00000000'EF   D1  0AB3  2021          CMPL    L^DUE_TIME,L^POLL_TIME     ; Compare low order           [128] ;G:16
                       20   1A  0ABE  2022          BGTRU   20$                        ; If GTRU, then time is not due [128] ;G:16
                                0AC0  2023 10$:
  00000899'EF   02000000 8F   CA  0AC0  2024          BICL2   #DS$M_TIMED,DS$GL_FLAGS ; Clear flag until a CONTINUE [128] ;G:16
  00000899'EF   08000000 8F   C8  0ACB  2025          BISL2   #DS$M_TIMED_OUT,DS$GL_FLAGS ; Set flag for ABORT      [128] ;G:16
                 50   10 AD   D0  0AD6  2026          MOVL    SF$L_SAVE_PC(FP),R0        ; /TIME is due, set up R0 for [128] ;G:16
              00000000'EF   17  0ADA  2027          JMP     DSI$ABORT                  ; DSI$ABORT. Abort the program [128] ;G:16
                                0AE0  2028 20$:                                                                           ;G:36
              0000FE48'EF   D6  0AE0  2029          INCL    L^DSA$GL_EVENT             ; INCREMENT EVENT COUNTER
                                0AE6  2030 KB_CHECK_APT::
         0000FE00'EF   1C   E0  0AE6  2031          BBS     #DSA$V_USER, -             ; CHECK FOR USER ENVIRONMENT.
                       24       0AED  2032                    L^DSA$GL_FLAGS, 10$
              0000000C'EF   95  0AEE  2033          TSTB    DS$AX_SOFTPCB+PCB$B_ASTACT ; Any ASTS active?
                       03   13  0AF4  2034          BEQL    5$                         ; Continue if not             [51]  ;G:16
                     019F   31  0AF6  2035          BRW     KB_CHECK_X                 ; Exit immediately if so      [51]  ;G:16
                                0AF9  2036
              00000000'EF   16  0AF9  2037 5$:      JSB     L^KB_POLL                  ; Poll console
                                0AFF  2038
              00000408'EF   D5  0AFF  2039          TSTL    MP$GR_BOOTED_PROCESSORS            ; Are we an mp system? [111]
                       0B   13  0B05  2040          BEQL    10$                                ; Not mp system        [111]
                       03   BB  0B07  2041          PUSHR   #^M<R0,R1>                 ; Save Registers        [131]      ;G:19
         00000000'EF   00   FB  0B09  2042          CALLS   #0,MP$_PRIMARY_CPU_CHECK           ; Call mp code         [111]
                       03   BA  0B10  2043          POPR    #^M<R0,R1>                 ; Restore Registers     [131]      ;G:19
         00000899'EF   00   E0  0B12  2044 10$:     BBS     #DS$V_CTRLC, -             ; CHECK CONTROL-C FLAG
                       03       0B19  2045                    L^DS$GL_FLAGS, 12$         ; If set, continue on         [51]  ;G:16
                     017B   31  0B1A  2046          BRW     KB_CHECK_X                 ; If clear, exit routine      [51]  ;G:16
                                0B1D  2047
         00000899'EF   18   E1  0B1D  2048 12$:     BBC     #DS$V_DISABLCC, -          ; Continue if ^C enabled      [70]  ;G:16
                       03       0B24  2049                    DS$GL_FLAGS, -             ; . . .                       [51]  ;G:16
                                0B25  2050                    13$                        ; . . .                       [70]  ;G:16
                     0170   31  0B25  2051          Brw     KB_Check_X                 ; Exit if ^C is disabled      [70]  ;G:16
                                0B28  2052
              00000000'EF   16  0B28  2053 13$:     JSB     SCRIPT$STOP                ; Stop script input           [44]  ;G:16
```

```
                        0B2E  2055          ;+
                        0B2E  2056          ; Algorithm:                                                    [70]  ;G:16
                        0B2E  2057          ; 1. Check the 1st DS ^C handler.                               [70]  ;G:16
                        0B2E  2058          ; 2. If there is not one, go to 4.                              [70]  ;G:16
                        0B2E  2059          ; 3. If there is one, do the following:                         [70]  ;G:16
                        0B2E  2060          ;       a. Call the 1st DS handler.                             [70]  ;G:16
                        0B2E  2061          ;       b. If it returns failure, go to 4.                      [70]  ;G:16
                        0B2E  2062          ;       c. If it returns success, clear the ^C flag and return. [70]  ;G:16
                        0B2E  2063          ; 4. Check the user ^C handler.                                 [70]  ;G:16
                        0B2E  2064          ; 5. If there is not one, go to 7.                              [70]  ;G:16
                        0B2E  2065          ; 6. If there is one, do the following:                         [70]  ;G:16
                        0B2E  2066          ;       a. Call the user handler.                               [70]  ;G:16
                        0B2E  2067          ;       b. If it returns failure, go to 7.                      [70]  ;G:16
                        0B2E  2068          ;       c. If it returns success, clear the ^C flag and return. [70]  ;G:16
                        0B2E  2069          ; 7. Check the 2nd DS ^C handler.                               [70]  ;G:16
                        0B2E  2070          ; 8. If there is not one, go to 10.                             [70]  ;G:16
                        0B2E  2071          ; 9. If there is one, do the following:                         [70]  ;G:16
                        0B2E  2072          ;       a. Call the 2nd DS handler.                             [70]  ;G:16
                        0B2E  2073          ;       b. If it returns failure, go to 10.                     [70]  ;G:16
                        0B2E  2074          ;       c. If it returns success, clear the ^C flag and return. [70]  ;G:16
                        0B2E  2075          ; 10. Set up the stack in preparation for doing an REI.         [70]  ;G:16
                        0B2E  2076          ; 11. Call DS$Cli for VDS command mode.                         [70]  ;G:16
                        0B2E  2077          ; 12. REI.                                                      [70]  ;G:16
                        0B2E  2078          ;--                                                             [70]  ;G:16
                        0B2E  2079                                                                                ;G:36
        00000194'EF  D5 0B2E  2080          TSTL    L^DS$GA_DS_Ctrl_C_First ; Check 1st DS Control-C Handler?[70] ;G:16
                  1F 13 0B34  2081          BEQL    15$                     ; If not one, branch          [70]  ;G:16
     7E 00000194'EF  D0 0B36  2082          MOVL    L^DS$GA_DS_Ctrl_C_First,-; Move the 1st handler address [70] ;G:16
                     0B3D  2083            -(SP)                   ; . . . to the stack                  [70]  ;G:16
        00000194'EF  D4 0B3D  2084          CLRL    L^DS$GA_DS_Ctrl_C_First ; Clear out the 1st handler address[70] ;G:1
        00010001 8F  CA 0B43  2085          BICL2   #DS$M_CTRLO!DS$M_CTRLC,-; Clear the ^C and ^O bits    [89]  ;G:16
        00000899'EF     0B49  2086            L^DS$GL_FLAGS           ; . . .                               [89]  ;G:16
            9E   6E  FA 0B4E  2087          CALLG   (SP), a(SP)+            ; Enter the first DS handler    [70]  ;G:16
              01 50  E9 0B51  2088          BLBC    R0, 15$                 ; If 1st 'failed', check user's [70]  ;G:16
                     05 0B54  2089          RSB                             ; . . . and return to the caller.[70] ;G:16
                     0B55  2090                                                                               ;G:36
        0000019C'EF  D5 0B55  2091 15$:     TSTL    L^DS$L_UserCntrlC       ; ANY USER HANDLER?             [70]  ;G:16
                  1F 13 0B5B  2092          BEQL    20$                     ; NO, go check DS's second handler[70] ;G:16
     7E 0000019C'EF  D0 0B5D  2093          MOVL    L^DS$L_UserCntrlC,-(SP) ; GET ADDR OF USER CNTRL-C ROUTINE[70] ;G:16
        0000019C'EF  D4 0B64  2094          CLRL    L^DS$L_UserCntrlC       ; CLEAR HANDLER ADDRESS         [70]  ;G:16
00000899'EF  00010001 8F  CA 0B6A  2095          BICL2   #DS$M_CTRLO!DS$M_CTRLC, -       ;                       [89]  ;G:16
                     0B75  2096            L^DS$GL_FLAGS           ; CLEAR CONTROL-C PENDING       [89]  ;G:16
            9E   6E  FA 0B75  2097          CALLG   (SP),a(SP)+             ; ENTER HANDLER                 ;G:36
              01 50  E9 0B78  2098          BLBC    R0, 20$                 ; If handler 'failed', let DS handle
                     05 0B7B  2099          RSB                             ; RETURN
                     0B7C  2100
        00000198'EF  D5 0B7C  2101 20$:     TstL    L^DS$GA_DS_Ctrl_C_Second; Is there a 2nd DS Cntrl-C Handler?[70] ;G:
                  1F 13 0B82  2102          Beql    50$                     ; If not, call DS$Cli.          [70]  ;G:
     7E 00000198'EF  D0 0B84  2103          MovL    L^DS$GA_DS_Ctrl_C_Second,-; Move the 2nd handler address   [70] ;G:
                     0B8B  2104            -(SP)                   ; . . . to the stack                  [70]  ;G:
        00000198'EF  D4 0B8B  2105          ClrL    L^DS$GA_DS_Ctrl_C_Second; Clear out the 2nd handler address [70] ;G:
        00010001 8F  CA 0B91  2106          BicL2   #DS$M_CtrlO!DS$M_CtrlC,-; Otherwise, clear ^C and ^O bits   [89] ;G:
        00000899'EF     0B97  2107            L^DS$GL_Flags           ; . . .                               [89]  ;G:
            9E   6E  FA 0B9C  2108          CallG   (SP), a(SP)+            ; Enter the second DS handler   [70]  ;G:
              B3 50  E9 0B9F  2109          BlbC    R0, 15$                 ; If 2nd 'failed', call DS$Cli  [70]  ;G:16
                     05 0BA2  2110          Rsb                             ; . . . and return to the caller.[70] ;G:16
                     0BA3  2111                                                                               ;G:36
```

```
00000899'EF   02000000 8F   CA   0BA3  2112 50$:    BICL2   #DS$M_TIMED,DS$GL_FLAGS ; /TIME is off until a CONTINUE [128] ;G:16
                      01    BB   0BAE  2113         PUSHR   #^M<R0>                 ; Save original value           [131] ;G:19
         50   00000000'EF   DE   0BB0  2114         MOVAL   DS$GL_CLIBASE,R0        ; base of CLI data structure    [128] ;G:16
          07 0444 C0   03   E1   0BB7  2115         BBC     #CLI$V_START_OR_RUN, -  ; If the last command was a RUN [128] ;G:16
                                0BBD  2116                 CLI$L_FLAGS2(R0),53$    ; or a START,                           ;G:16
         00000411'EF   6E   D0   0BBD  2117         MOVL    (SP),START_RUN_PC       ; Save PC in case of a CONTINUE [128] ;G:16
                      01    BA   0BC4  2118 53$:    POPR    #^M<R0>                 ; Restore original value        [131] ;G:19
                  7E  6E   D0   0BC6  2119         MOVL    (SP),-(SP)              ; COPY RETURN ADDRESS           [131] ;G:19
                    04 AE  DC   0BC9  2120         MOVPSL  4(SP)                   ; SAVE CURRENT PSL
                    08 AE  9F   0BCC  2121         PUSHAB  8(SP)                   ; TRUE STACK POINTER AT ENTRY
                  3FFF 8F   BB   0BCF  2122         PUSHR   #^X3FFF                 ; STACK REGISTERS
         00000000'EF   6E   FA   0BD3  2123         CALLG   (SP),L^DS$CLI           ; ENTER COMMAND DECODER
                                0BDA  2124
       00000411'EF   3C AE   D1   0BDA  2125         CMPL    60(SP),START_RUN_PC     ; If we are CONTINUING from a   [128] ;G:16
                      03    13   0BE2  2126         BEQL    55$                     ; RUN or START command, check   [128] ;G:16
                   00A0     31   0BE4  2127         BRW     75$                     ; for /TIME and MP startup      [128] ;G:16
                                0BE7  2128                                                                                 ;G:16
00000000'EF   00000000'EF   7D   0BE7  2129 55$:    MOVQ    L^BIN_TIME,L^BIN_TIME   ; Leave if /TIME not given      [88]  ;G:16
                      03    12   0BF2  2130         BNEQ    61$                     ;                               [128] ;G:16
                   0075     31   0BF4  2131         BRW     70$                                                            ;G:16
00000000'EF   00000000'EF   C2   0BF7  2132 61$:    SUBL2   L^BEGIN_TIME, -         ; Calculate remaining time for  [88]  ;G:36
                                0C02  2133                 L^POLL_TIME             ; /TIME                         [88][128] ;G:
00000004'EF   00000004'EF   D9   0C02  2134         SBWC    L^BEGIN_TIME+4, -       ; Subtract 2 quads              [88]  ;G:16
                                0C0D  2135                 L^POLL_TIME+4           ; Result is time actually used.[88][128] ;G:
00000000'EF   00000000'EF   C0   0C0D  2136         ADDL2   L^POLL_TIME, -          ; Now, calculate how much more [88][128] ;G:
                                0C18  2137                 L^BIN_TIME              ; to let it run. BIN_TIME is a  [88]  ;G:16
00000004'EF   00000004'EF   D8   0C18  2138         ADWC    L^POLL_TIME+4, -        ; delta time, thus is negative [88][128] ;G:
                                0C23  2139                 L^BIN_TIME+4            ;                               [88]  ;G:16
                                0C23  2140                                         ; Calculate new due time        [128] ;G:16
                      03    BB   0C23  2141         PUSHR   #^M<R0,R1>              ; Used by GET_TIME              [128] ;G:17
            00000000'EF   DF   0C25  2142         PUSHAL  BEGIN_TIME              ; Address to store result       [128] ;G:16
      00000C99'EF   01    FB   0C2B  2143         CALLS   #1,GET_TIME             ; Use TODR to calculate SYSTIME [128] ;G:16
                      03    BA   0C32  2144         POPR    #^M<R0,R1>              ; Used by GET_TIME              [128] ;G:17
                                0C34  2145                                                                                 ;G:16
00000000'EF   00000000'EF   7D   0C34  2146         MOVQ    L^BEGIN_TIME,L^DUE_TIME ; Initialize DUE_TIME           [128] ;G:16
00000000'EF   00000000'EF   C2   0C3F  2147         SUBL2   L^BIN_TIME,L^DUE_TIME   ; BIN_TIME is negative, so      [128] ;G:16
                                0C4A  2148                                         ; subtracting it is really      [128] ;G:16
00000004'EF   00000004'EF   D9   0C4A  2149         SBWC    L^BIN_TIME+4,L^DUE_TIME+4 ; adding it.                  [128] ;G:16
00000899'EF   02000000 8F   C8   0C55  2150         BISL2   #DS$M_TIMED,DS$GL_FLAGS ; This run is timed             [128] ;G:16
         50   00000000'EF   DE   0C60  2151         MOVAL   DS$GL_CLIBASE,R0        ; base of CLI data structure    [128] ;G:16
            0444 C0   08   C8   0C67  2152         BISL2   #CLI$M_START_OR_RUN, -  ; Pretend the last command was  [128] ;G:16
                                0C6C  2153                 CLI$L_FLAGS2(R0)        ; a START or RUN, so that we can[128] ;G:16
                                0C6C  2154                                         ; CONTINUE from another ^C.     [128] ;G:16
                                0C6C  2155 70$:
            00000408'EF   D5   0C6C  2156         TSTL    MP$GR_BOOTED_PROCESSORS         ; Are we an mp system?  [111] ;G:19
                      13    13   0C72  2157         BEQL    75$                     ; Not mp system                 [111] ;G:16
         00000000'EF   00    FB   0C74  2158         CALLS   #0,MP$_RESUME_ATTACHED_PROCESSORS ; If CONT, then restart [111] ;G:1
                                0C7B  2159                                         ; all Ap's                      [111] ;G:16
         50   00000000'EF   DE   0C7B  2160         MOVAL   DS$GL_CLIBASE,R0        ; base of CLI data structure    [131] ;G:19
            0444 C0   08   C8   0C82  2161         BISL2   #CLI$M_START_OR_RUN, -  ; Pretend the last command was  [128] ;G:16
                                0C87  2162                 CLI$L_FLAGS2(R0)        ; a START or RUN, so that we can[128] ;G:16
                                0C87  2163                                         ; CONTINUE from another ^C.     [128] ;G:16
                                0C87  2164                                                                                 ;G:16
            50   38 AE   D0   0C87  2165 75$:    MOVL    14*4(SP),R0             ; GET NEW STACK POINTER         [111] ;G:16
            70   3C AE   7D   0C8B  2166         MOVQ    15*4(SP),-(R0)          ; PUT NEW PC/PSL THERE
            38 AE   50   D0   0C8F  2167         MOVL    R0,14*4(SP)             ; PUT UPDATED NEW STACK POINTER BACK
               7FFF 8F   BA   0C93  2168         POPR    #^X7FFF                 ; RESTORE REGISTERS, CHANGES SP
```

```
                02  0C97  2169          REI                             ; USE NEW PC/PSL
                    0C98  2170
                    0C98  2171 KB_CHECK_X:
                05  0C98  2172          RSB                             ; RETURN
```

```
                                0C99  2174          .SBTTL  GET_TIME        Convert TODR to current system time          ;G:16
                                0C99  2175 ;++                                                              begin   [128] ;G:16
                                0C99  2176 ; FUNCTIONAL DESCRIPTION:                                                      ;G:16
                                0C99  2177 ;                                                                              ;G:16
                                0C99  2178 ;       This routine returns the system time.                                  ;G:17
                                0C99  2179 ;       If running under VMS, returns execution time.                          ;G:17
                                0C99  2180 ;       If standalone, execution time and elapsed time are the same. For       ;G:17
                                0C99  2181 ;       non-8800 systems, the time is obtained from EXE$GQ_SYSTIME, unless the ;G:17
                                0C99  2182 ;       DS$V_TIMRON flag indicates that the diagnostic is using the interval   ;G:17
                                0C99  2183 ;       clock, in which case EXE$GQ_SYSTIME is not current. In that case, read ;G:17
                                0C99  2184 ;       the TODR, and convert it to the current system time.                    ;G:17
                                0C99  2185 ;       For 8800 systems, a different algorithm is followed because too        ;G:16
                                0C99  2186 ;       many reads of the TODR can drastically slow down execution time.       ;G:16
                                0C99  2187 ;       The time is obtained from EXE$GQ_SYSTIME regardless of whether the     ;G:16
                                0C99  2188 ;       diagnostic is using the interval timer (DS$V_TIMRON). This does        ;G:16
                                0C99  2189 ;       not seem to affect the accuracy of the /TIME switch too much. Only     ;G:16
                                0C99  2190 ;       when CONTINUing from a ^C is the TODR read.                            ;G:16
                                0C99  2191 ;                                                                              ;G:16
                                0C99  2192 ; CALLING SEQUENCE:                                                            ;G:16
                                0C99  2193 ;                                                                              ;G:16
                                0C99  2194 ;       CALLS   #1,GET_TIME                                                    ;G:16
                                0C99  2195 ;                                                                              ;G:16
                                0C99  2196 ; INPUT PARAMETERS:                                                            ;G:16
                                0C99  2197 ;                                                                              ;G:16
                                0C99  2198 ;       4(AP)   Address of quadword which receives current system time         ;G:16
                                0C99  2199 ;                                                                              ;G:16
                                0C99  2200 ; IMPLICIT INPUTS:      NONE                                                   ;G:16
                                0C99  2201 ;                                                                              ;G:16
                                0C99  2202 ; OUTPUT PARAMETERS:    NONE                                                   ;G:16
                                0C99  2203 ;                                                                              ;G:16
                                0C99  2204 ; IMPLICIT OUTPUTS:     NONE                                                   ;G:16
                                0C99  2205 ;                                                                              ;G:16
                                0C99  2206 ; COMPLETION CODES:     NONE                                                   ;G:16
                                0C99  2207 ;                                                                              ;G:16
                                0C99  2208 ; SIDE EFFECTS:         NONE                                                   ;G:16
                                0C99  2209 ;--                                                                            ;G:16
                          0028  0C99  2210 .ENTRY  GET_TIME,^M<R3,R5>                                                     ;G:16
                                0C9B  2211          BR_IF_NOT_USER  50$             ;                          [129] ;G:17
0000FE00'EF     1C    E1      0C9B                BBC     #DSA$V_User, -                  ; If bit not set, branch
                38              0CA2                        L^DSA$GL_Flags, 50$     ;
                                0CA3  2212          $GETJPIW_S ITMLST=CPUTIM        ; Returns process's accumulated [129] ;G:17
                7E    7C        0CA3                        CLRQ    -(SP)
                00    DD        0CA5                        PUSHL   #0
     00000415'EF      DF        0CA7                        PUSHAL  CPUTIM
                00    DD        0CAD                        PUSHL   #0
                00    DD        0CAF                        PUSHL   #0
                00    DD        0CB1                        PUSHL   #0
00000000'GF     07    FB        0CB3                        CALLS   #7,G^SYS$GETJPIW
                                0CBA  2213                                      ; cpu time in 10ms units        [129] ;G:17
           14 50      E8        0CBA  2214          BLBS    R0,10$              ;                               [129] ;G:17
                                0CBD  2215          ERRSUP_S                    ; Couldn't get cpu time         [129] ;G:17
     0000016B'EF      DF        0CBD                        PUSHAL  $MODULE
                00    DD        0CC3                        PUSHL   #0
                02    DD        0CC5                        PUSHL   #$ER
00000000'EF     03    FB        0CC7                        CALLS   #3, DS_ERRSUP
          00F2  31    0CCE  2216          BRW     1000$               ;                               [129] ;G:17
50   0000425'EF     D0        0CD1  2217 10$:     MOVL    JPICPUTIM,R0            ; R0 gets cpu time              [129] ;G:17
```

```
                  00C5   31  0CD8  2218          BRW     751$                      ;                                [129] ;G:17
                             0CDB  2219                                                                                  ;G:17
      06   0000FE17'EF   91  0CDB  2220 50$:     CMPB    DSA$GL_SID+3,#PR$_SID_TYP8NN ; 8800? [138]                    ;G:36
                    3A   13  0CE2  2221          BEQL    500$                                                          ;G:34
      11   0000FE17'EF   91  0CE4  2222          CMPB    DSA$GL_SID+3,#PR$_SID_TYP8RR ; RRR? [138]                     ;G:34
                    31   13  0CEB  2223          BEQL    500$                                                          ;G:34
                             0CED  2224                                                                                  ;G:16
                             0CED  2225 100$:                                      ; not an 8800                         ;G:17
      00000899'EF   11   E0  0CED  2226          BBS     #DS$V_TIMRON, -           ; Use EXE$GQ_SYSTIME unless someone   ;G:16
                    0B       0CF4  2227                  DS$GL_FLAGS,200$          ; is using the interval clock         ;G:17
                             0CF5  2228                                                                                  ;G:16
   04 BC   00000000'EF   7D  0CF5  2229          MOVQ    EXE$GQ_SYSTIME,a4(AP)     ; Store current time                  ;G:16
                  00C3   31  0CFD  2230          BRW     1000$                     ; Exit                                ;G:16
                             0D00  2231                                                                                  ;G:36
              53   1B   D0  0D00  2232 200$:    MOVL    #PR780$_TODR,R3           ; Standalone, use TODR                ;G:17
              55   01   D0  0D03  2233          MOVL    #1,R5                     ; Code for EXAMINE command            ;G:16
                             0D06  2234          CMK     SGIPR                     ; Must be in KERNEL mode              ;G:16
                    7E   DC  0D06                MOVPSL  -(SP)
           06 6E   1A   E0  0D08                BBS     #PSL$V_IS, (SP), 30004$
                    8E   D4  0D0C                CLRL    (SP)+
                    0A   BC  0D0E                CHMK    #CMK$_SGIPR
                    06   11  0D10                BRB     30005$
          00000000'EF   16  0D12       30004$:  JSB     CMK$SGIPR
                             0D18       30005$:
              50   51   D0  0D18  2235          MOVL    R1,R0                     ; Put TODR contents in R0             ;G:16
                  007B   31  0D1B  2236          BRW     750$                      ; Finish with common code             ;G:16
                             0D1E  2237                                                                                  ;G:16
                             0D1E  2238                                                                                  ;G:16
                             0D1E  2239 500$:                                      ; 8800 and RRR routine [138]          ;G:34
   0B 00000899'EF   19   E1  0D1E  2240          BBC     #DS$V_TIMED,DS$GL_FLAGS,510$ ; Flag is clear if this routine  ;G:16
                             0D26  2241                                            ; is called as a result of a CONTINUE ;G:16
                             0D26  2242                                            ; command                             ;G:16
                             0D26  2243                                                                                  ;G:16
                             0D26  2244                                            ; If not from a CONTINUE command,     ;G:16
                             0D26  2245                                            ; it is polling in KB_CHECK,          ;G:16
   04 BC   00000000'EF   7D  0D26  2246          MOVQ    EXE$GQ_SYSTIME,a4(AP)     ; store current time                  ;G:16
                  0092   31  0D2E  2247          BRW     1000$                     ; Exit                                ;G:16
                             0D31  2248                                                                                  ;G:16
                             0D31  2249                                            ; Fetch TODR from console             ;G:16
                             0D31  2250                                            ; This is only done if this routine   ;G:16
                             0D31  2251                                            ; is called as a result of a CONTINUE ;G:16
                             0D31  2252                                            ; command                             ;G:16
                             0D31  2253 510$:                                                                            ;G:16
          00000000'EF   D4  0D31  2254          CLRL    TODR_BYTE_CNT             ; init byte/loop count                ;G:16
                             0D37  2255 520$:                                                                            ;G:16
              53   22   D0  0D37  2256          MOVL    #PR$_TXCS,R3              ; Standalone, use TODR                ;G:16
              55   01   D0  0D3A  2257          MOVL    #1,R5                     ; Code for EXAMINE command            ;G:16
                             0D3D  2258          CMK     SGIPR                     ; Must be in KERNEL mode              ;G:16
                    7E   DC  0D3D                MOVPSL  -(SP)
           06 6E   1A   E0  0D3F                BBS     #PSL$V_IS, (SP), 30006$
                    8E   D4  0D43                CLRL    (SP)+
                    0A   BC  0D45                CHMK    #CMK$_SGIPR
                    06   11  0D47                BRB     30007$
          00000000'EF   16  0D49       30006$:  JSB     CMK$SGIPR
                             0D4F       30007$:
   E0 51   00000000'8F   E1  0D4F  2259          BBC     #TXCS$V_RDY,R1,520$       ; branch if not ready to xmit         ;G:16
                    55   D4  0D57  2260          CLRL    R5                        ; CMK$SGIPR code for DEPOSIT function ;G:16
```

```
                51   00000F08 8F    D0   0D59  2261           MOVL    #^X00000F08,R1          ; TOY read request                ;G:16
                     53    23      D0   0D60  2262           MOVL    #PR$_TXDB,R3            ; IPR to write to                 ;G:16
                                   0D63  2263           CMK     SGIPR                   ; Must be in KERNEL mode          ;G:16
                           7E      DC   0D63                    MOVPSL  -(SP)
                     06 6E 1A      E0   0D65                    BBS     #PSL$V_IS, (SP), 30008$
                           8E      D4   0D69                    CLRL    (SP)+
                           0A      BC   0D6B                    CHMK    #CMK$_SGIPR
                           06      11   0D6D                    BRB     30009$
                00000000'EF        16   0D6F         30008$: JSB     CMK$SGIPR
                                   0D75         30009$:
                                   0D75  2264 530$:                                                       ;G:16
                                   0D75  2265           CMK     RCONSOLE                ; get binary data                 ;G:16
                           7E      DC   0D75                    MOVPSL  -(SP)
                     06 6E 1A      E0   0D77                    BBS     #PSL$V_IS, (SP), 30010$
                           8E      D4   0D7B                    CLRL    (SP)+
                           01      BC   0D7D                    CHMK    #CMK$_RCONSOLE
                           06      11   0D7F                    BRB     30011$
                00000000'EF        16   0D81         30010$: JSB     CMK$RCONSOLE
                                   0D87         30011$:
                04 00000000'EF     D1   0D87  2266           CMPL    TODR_BYTE_CNT,#4        ; do we have all four bytes of    ;G:16
                                   0D8E  2267                   ; toy data                                         ;G:16
                           02      13   0D8E  2268           BEQL    540$                    ; yes then continue               ;G:16
                           E3      11   0D90  2269           BRB     530$                    ; branch if not                   ;G:16
                                   0D92  2270 540$:                                                       ;G:16
                50 00000000'EF     D0   0D92  2271           MOVL    TODR_STORE,R0           ; get toy data                    ;G:16
                                   0D99  2272                                                             ;G:16
                50 00000000'8F     C2   0D99  2273 750$:      SUBL2   #TODR_BIAS,R0           ; Remove validation bias          ;G:16
                     50    01      CA   0DA0  2274 751$:      BICL    #1,R0                   ; Clear low bit          [129]    ;G:17
                 50  50    FF 8F   9C   0DA3  2275           ROTL    #-1,R0,R0               ; Divide by 2 unsigned            ;G:16
50   00           50 00000000'8F   7A   0DA8  2276           EMUL    #2*TICK_NS,R0,#0,R0     ; Convert to SYSTIME format       ;G:16
                                   0DB1  2277                   ; 10ms to 100 ns                                   ;G:16
                50 00000000'EF     C0   0DB1  2278           ADDL2   L^DS$GQ_CURYEAR,R0      ; Adjust to current time/date     ;G:16
                51 00000004'EF     D8   0DB8  2279           ADWC    L^DS$GQ_CURYEAR+4,R1    ; (all 64 bits)                   ;G:16
                     04 BC   50    7D   0DBF  2280           MOVQ    R0,@4(AP)               ; Return result                   ;G:16
                                   04   0DC3  2281 1000$:     RET                             ;                        end      [128] ;G:16
                                   0DC4  2282                                                             ;G:16
```

```
                                ODC4  2284          .SBTTL  DSX$CNTRLC      Program enable for Control-C intercept
                                ODC4  2285 ;++
                                ODC4  2286 ; FUNCTIONAL DESCRIPTION:
                                ODC4  2287 ;
                                ODC4  2288 ;       This routine enables the diagnostic to intercept the next
                                ODC4  2289 ;       Control-C typed by the operator.
                                ODC4  2290 ;       If a routine has been specified when the next Control-C is
                                ODC4  2291 ;       typed, the enable is cleared and the routine called.
                                ODC4  2292 ;       The enable is effective until a Control-C is typed
                                ODC4  2293 ;       or canceled by calling with a zero routine_address.
                                ODC4  2294 ;
                                ODC4  2295 ;       The routine also allows enabling or disabling control C handling.  If
                                ODC4  2296 ;       the DISABL argument has low bit SET, ^C handling is DISABLED.
                                ODC4  2297 ;
                                ODC4  2298 ; CALLING SEQUENCE:
                                ODC4  2299 ;
                                ODC4  2300 ;       DS$CNTRLC(routine_address)
                                ODC4  2301 ;
                                ODC4  2302 ; INPUT PARAMETERS:
                                ODC4  2303 ;
                                ODC4  2304 ;       4(AP)   Address of routine to call on Control-C
                                ODC4  2305 ;       8(AP)   DISABL flag (LBS to disable, LBC to enable)
                                ODC4  2306 ;
                                ODC4  2307 ; IMPLICIT INPUTS:      NONE
                                ODC4  2308 ;
                                ODC4  2309 ; OUTPUT PARAMETERS:    NONE
                                ODC4  2310 ;
                                ODC4  2311 ; IMPLICIT OUTPUTS:
                                ODC4  2312 ;
                                ODC4  2313 ;       DS$L_USERCNTRLC is set to value of routine_address
                                ODC4  2314 ;
                                ODC4  2315 ; COMPLETION CODES:     NONE
                                ODC4  2316 ;
                                ODC4  2317 ; SIDE EFFECTS:         NONE
                                ODC4  2318 ;--
                                ODC4  2319
                          0000  ODC4  2320 .ENTRY  DSX$CNTRLC,^M<>
                                ODC6  2321
 0000019C'EF    04 AC    D0    ODC6  2322          MOVL    4(AP),L^DS$L_USERCNTRLC ; SET USER CONTROL-C ADDRESS
                50       D4    ODCE  2323          CLRL    R0
             02 6C       91    ODD0  2324          CMPB    (AP),#2                 ; Is DISABL included?
                16       1F    ODD3  2325          BLSSU   10$                     ; No: don't use it
    51   00000899'EF    9E    ODD5  2326          MOVAB   L^DS$GL_FLAGS, R1       ; Point to internal flags
 50   61    01    18    EF    ODDC  2327          EXTZV   #DS$V_DISABLCC,#1,(R1),R0; Extract current value of disable bit
61   01    18    08 AC    F0    ODE1  2328          INSV    8(AP),#DS$V_DISABLCC,#1,(R1); Insert new value
       50    50    03    78    ODE7  2329          ASHL    #3,R0,R0                ; Make old value into a
             50    01    88    ODEB  2330 10$:     BISB    #1,R0                   ;  WASSET or WASCLR return code
                         04    ODEE  2331          RET
                               ODEF  2332                                                                  ;G:3
```

```
                        .Sbttl  DSV$EXIT, Process EXIT command
                ;++
                ; Functional Description:
                ;
                ;       This routine is called from CLI when an EXIT command is
                ;       typed.  It will call DS_CLEANUP, INIT_CONTEXT; then if
                ;       in standalone, it will HALT; if on-line, it will do a
                ;       $EXIT call. If the bits in the R5 that was passed from the
                ;       BOOT routines are setup for CRD-AutoTest, then do some
                ;       special things.
                ;
                ; Inputs:
                ;
                ;       None
                ;
                ; Outputs:
                ;
                ;       None
                ;
                ;--
```

```
                                0DEF  DSV$EXIT::                                  ;                                                [47]
                                0DEF          Br_If_Not_LodFlg        10$         ; Branch if no program loaded                    [63]
        00000899'EF   01   E1   0DEF          BBC     #DS$V_LodFlg, -                     ; If bit not set, branch                 [29]
                      06        0DF6                    L^DS$GL_Flags, 10$                  ;                                        [29]
             00000000'EF   16   0DF7          jsb     Ds_Cleanup                  ; Conditionally call                [47]  ;G:3
                                0DFD                                              ;  program cleanup                  [47][125] ;G
                                0DFD
                    FA8A   30   0DFD  10$:    Bsbw    Init_Context                ; Clean up after ourselves                       [47]
                                0E00
                                0E00          ;+
                                0E00          ; Call the Unload_CRD routine - this routine will check to see if CRD           [71]
                                0E00          ; been running. If it hasn't, it will return. If CRD has been running,          [71]
                                0E00          ; the routine will deallocate all the memory that was allocated for             [71]
                                0E00          ; the CRD image. In this case, it will not return to here! It goes to           [71]
                                0E00          ; either the Begin_Bliss label, or it does a Halt.                               [71]
                                0E00          ;-
                                0E00
                      00   DD   0E00          PushL   #0                          ; If the CRD image is in memory, then            [76]
                                0E02                                              ; . . . don't return to here.                    [76]
        00000000'EF   01   FB   0E02          Calls   #1, DSR$Unload_CRD          ; 'Unload' the CRD image from memory             [76]
                                0E09
                                0E09          ;+
                                0E09          ; If it comes to here, then the CRD image was not in memory.                     [76]
                                0E09          ;-
                                0E09
                                0E09          Br_If_User      60$                 ; If in User-Mode, don't try to halt             [63]
        0000FE00'EF   1C   E0   0E09          BBS     #DSA$V_User, -                      ; If bit set, branch                     [28]
                      55        0E10                    L^DSA$GL_Flags, 60$                 ;                                        [28]
                                0E11
                                0E11          BR_IF_NOT_AP    20$                 ; Not AP, normal exit                [125] ;G:3
     49 00000000'EF   08   E1   0E11          BBC     #AP$V_AP_Running,AP$GW_Data,20$
        7E   0000005E 8F   DB   0E19          MFPR    #PR8SS$_BINID,-(SP)                 ; Get our node id
   00000000'EF   04   04   ED   0E20          CMPZV   #AP$V_AP_Node_Id,#AP$S_AP_Node_Id, - ; Is our node id the same as
                      8E        0E28                    AP$GW_Data, (SP)+                   ; the AP node id?
                      37   12   0E29          BNEQ    20$                                 ; No, Branch
                                0E2B  ;+                                                                                          ;G:3
                                0E2B  ; If we are running in the AP, request the PP to set up its RXCD intr   [125] ;G:3
                                0E2B  ; to handle the console prompt, as well as getting ready to verify that [125] ;G:3
                                0E2B  ; >>> did come over, and clear the AP_RUNNING bit and HALT                [102][125] ;
                                0E2B  ;-                                                                                          ;G:3
        00000000'EF   01   D0   0E2B          MOVL    #AP$_REQ_RXCDVEC, -         ; Set request type                [125] ;G:3
                                0E32                    AP$GR_SHARED_DATA           ;                                 [125] ;G:3
 00000004'EF   00000001'EF   DE   0E32          MOVAL   AP$I_PROMPT_CHECK+1, -      ; PP will get console prompt      [125] ;G:3
                                0E3D                    AP$GR_SHARED_DATA+4         ;                                 [125] ;G:3
   00000000'EF   0400 8F   A8   0E3D          BISW2   #AP$M_REQUEST, -            ; Set request bit                 [125] ;G:3
                                0E46                    AP$GW_DATA                  ;                                 [125] ;G:3
        00000000'EF   0A   E0   0E46  15$:    BBS     #AP$V_REQUEST, -            ; Loop until PP finishes req      [125] ;G:3
                      F8        0E4D                    AP$GW_DATA, 15$             ;                                 [125] ;G:3
             00000000'EF   94   0E4E          CLRB    AP$B_PROMPT_BUFFER_COUNT ; Clear buffer pointer               [125] ;G:3
             00000000'EF   94   0E54          CLRB    AP$B_PROMPT_COUNT           ; Clear count of '>'s recvd       [125] ;G:3
                                0E5A                                                                                              ;G:3
     00 00000000'EF   08   E4   0E5A          bbsc    #ap$v_ap_running,AP$gw_data,20$ ; clear ap_running bit  [102] ;G:3
                                0E62
                      00        0E62  20$:    Halt                                ; Halt processor in standalone                   [63]
                    FA07   31   0E63          BrW     Begin                       ; ReStart the VDS if console Continue            [71]
                                0E66
                                0E66          ;+
```

```
                                0E66  2403          ; We are in User-Mode. Do a system service exit.                    [63]
                                0E66  2404          ;-
                                0E66  2405
                                0E66  2406 60$:     $Exit_S                          ; Exit with success code            [63]
                     01    DD   0E66                          PUSHL   #1
        00000000'GF  01    FB   0E68                          CALLS   #1,G^SYS$EXIT
                           05   0E6F  2407          Rsb                              ; (just in case)                    [47]
```

```
                        0E70  2409          .SBTTL  EXIT$HANDLR     Process EXIT handler
                        0E70  2410 ;++
                        0E70  2411 ; FUNCTIONAL DESCRIPTION:
                        0E70  2412 ;
                        0E70  2413 ;       This routine is called from the command interpreter
                        0E70  2414 ;       when the image is forced to exit.  It's sole purpose is
                        0E70  2415 ;       to execute the diagnostic program cleanup code.
                        0E70  2416 ;       DS_CLEANUP is called only if a program is loaded.
                        0E70  2417 ;       In addition any devices allocated by select are deallocated.
                        0E70  2418 ;
                        0E70  2419 ; CALLING SEQUENCE:
                        0E70  2420 ;
                        0E70  2421 ;       EXIT$HANDLR()
                        0E70  2422 ;
                        0E70  2423 ; INPUT PARAMETERS:     NONE
                        0E70  2424 ;
                        0E70  2425 ; IMPLICIT INPUTS:
                        0E70  2426 ;
                        0E70  2427 ;       DS$GL_FLAGS     DS$V_DONFLG, DS$V_LODFLG, DS$V_STRFLG
                        0E70  2428 ;
                        0E70  2429 ; OUTPUT PARAMETERS:    NONE
                        0E70  2430 ;
                        0E70  2431 ; IMPLICIT OUTPUTS:     NONE
                        0E70  2432 ;
                        0E70  2433 ; COMPLETION CODES:     NONE
                        0E70  2434 ;--
                        0E70  2435
                        0E70  2436 EXIT$HANDLR:
                  0FFC  0E70  2437          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
06 00000899'EF    01    E1  0E72  2438          BBC     #DS$V_LODFLG,L^DS$GL_FLAGS,10$ ; Branch if not program loaded
       00000000'EF    16  0E7A  2439          JSB     DS_CLEANUP                      ; Conditionally execute program cleanup ;G:4
                        0E80  2440
   55  00000000'EF    DE  0E80  2441 10$:     MOVAL   DS$GA_PTABLE,R5                 ; Point to P-table list
   53  00000000'EF    DE  0E87  2442          MOVAL   DS$GA_SELECTS,R3                ; Point to select bits
           54    65    D0  0E8E  2443          MOVL    (R5),R4                         ; Get max count of units
                        0E91  2444
         52    6544    D0  0E91  2445 20$:     MOVL    (R5)[R4],R2                     ; Get address of P-table
                 15    13  0E95  2446          BEQL    30$                             ; Branch if no P-table
      10 0A A2   00    E1  0E97  2447          BBC     #HP$V_ALLOC, -
                        0E9C  2448                  HP$B_FLAGS(R2),30$              ; Branch if should not be deallocated
      0B 0A A2   01    E1  0E9C  2449          BBC     #HP$V_WASALL, -
                        0EA1  2450                  HP$B_FLAGS(R2),30$              ; Branch if is not allocated
                        0EA1  2451          $DALLOC_S HP$Q_DEVICE(R2)               ; Deallocate device
                 00    DD  0EA1                      PUSHL   #0
                 62    7F  0EA3                      PUSHAQ  HP$Q_DEVICE(R2)
00000000'GF      02    FB  0EA5                      CALLS   #2,G^SYS$DALLOC
            E2 54    F5  0EAC  2452 30$:     SOBGTR  R4,20$                          ; Count and branch
                        0EAF  2453 ;
                        0EAF  2454 ; Restore sys$disk and original default directory                        ;G:28
                        0EAF  2455 ;
   50  000001E0'EF    9E  0EAF  2456          MOVAB   L^DS$AT_DDEV,R0                 ; Address of new logical name   [135] ;G:28
              02 A0    9F  0EB6  2457          PUSHAB  2(R0)
           7E   60    3C  0EB9  2458          MOVZWL  (R0),-(SP)                      ; Length of new logical name    [135] ;G:28
                 7E    D4  0EBC  2459          CLRL    -(SP)                           ; Access mode                   [135] ;G:28
              04 AE    7F  0EBE  2460          PUSHAQ  4(SP)                           ; point to descriptor of new name [135] ;G:2
       00000000'EF    7F  0EC1  2461          PUSHAQ  SYS$DISK                        ; Address of descriptor of logical name [135
                 00    DD  0EC7  2462          PUSHL   #0                              ; In process logical name table [135] ;G:28
```

```
                00000000'EF   04   FB  0EC9  2463          CALLS   #4,SYS$CRELOG            ; Create logical name          [135] ;G:28
                                      0ED0  2464
                50   000001B8'EF   9E  0ED0  2465          MOVAB   L^DS$AT_DDIR,R0          ; Address of old directory
                   04 AE    02 A0   9E  0ED7  2466          MOVAB   2(R0),4(SP)              ; Fill in descriptor
                         6E   60   3C  0EDC  2467          MOVZWL  (R0),(SP)                ; Fill in length
                              7E   7C  0EDF  2468          CLRQ    -(SP)                    ; Null parameters
                           08 AE   7F  0EE1  2469          PUSHAQ  8(SP)                    ; Address of descriptor
                00000000'EF   03   FB  0EE4  2470          CALLS   #3,SYS$SETDDIR           ; Set default directory back
                         5E   08   C0  0EEB  2471          ADDL    #8,SP                    ; Remove extra stuff
                                      0EEE  2472
                                   04 0EEE  2473          RET
```

```
                                0EEF          .SubTitle       DSR$Check_MenuTest_Off_Set Routine
                                0EEF  ;++
                                0EEF  ; FUNCTIONAL DESCRIPTION:
                                0EEF  ;
                                0EEF  ;       This routine checks to see if CRD MenuTest Offline is active.
                                0EEF  ;       It checks the Boot$L_R5 longword and DSA Flags.
                                0EEF  ;       The value in Boot$L_R5 is the value that
                                0EEF  ;       is passes to the VDS from DiagBoot (in R5).
                                0EEF  ;
                                0EEF  ; CALLING SEQUENCE:
                                0EEF  ;
                                0EEF  ;       Jsb/Bsbw/Bsbb DSR$Check_MenuTest_Off_Set
                                0EEF  ;
                                0EEF  ; INPUT PARAMETERS:
                                0EEF  ;
                                0EEF  ;       NONE
                                0EEF  ;
                                0EEF  ; IMPLICIT INPUTS:
                                0EEF  ;
                                0EEF  ;       Boot$L_R5
                                0EEF  ;       DSA$GL_Flags
                                0EEF  ;
                                0EEF  ; OUTPUT PARAMETERS:
                                0EEF  ;
                                0EEF  ;       NONE
                                0EEF  ;
                                0EEF  ; IMPLICIT OUTPUTS:
                                0EEF  ;
                                0EEF  ;       NONE
                                0EEF  ;
                                0EEF  ; COMPLETION CODES:
                                0EEF  ;
                                0EEF  ;       R0 bit 0 set:
                                0EEF  ;               CRD MENU active.
                                0EEF  ;       R0 bit 1 set:
                                0EEF  ;               The correct bits are set  in Boot$L_R5.
                                0EEF  ;               This indicates that CRD MENU
                                0EEF  ;               was invoked from 'T/M' console command. R0 bit
                                0EEF  ;               1 clear indicates that CRD MENU was invoked by VDS
                                0EEF  ;               'CRD' command.
                                0EEF  ;       R0 = 0: CRD MENU not active.
                                0EEF  ;--
                                0EEF  DSR$Check_MenuTest_Off_Set::            ;                                    [80]
                                0EEF
                     50   D4    0EEF          ClrL    R0                      ; Assume they are not set            [69]
                                0EF1
0000FE00'EF          10   E1    0EF1          Bbc     #DSA$V_CRD_MenuTest_Off, - ; If the CRD_MenuTest_Off
                     03         0EF8                  L^DSA$GL_Flags,    -  ; ...bit is not set in the DSA flags
                                0EF9                  50$                    ; ... branch to 50$                  [84]
                                0EF9
                50   01   C8    0EF9          BISL2   #1, R0          ; CRD MENU active; set the flag              [84]
                                0EFC
                                0EFC          ;+                                                                   [69]
                                0EFC          ; If the DIAG and the MenuTest bits are both set in the R5 passed from [71]
                                0EFC          ; DiagBoot, set the low bit in R0 and return.                        [69]
                                0EFC          ;-                                                                   [69]
                                0EFC
```

```
                00000879'EF    04    E1    0EFC  2532 50$:    Bbc     #Rpb$V_Diag, -          ; If the DIAG bit is clear ...          [79]
                               0E          0F03  2533           L^Boot$L_R5, -          ; ... in the R5 passed from DiagBoot,   [69]
                                           0F04  2534           100$                    ; ... branch to 100$.                   [69]
                                           0F04  2535
                                           0F04  2536       ;+                                                                  [69]
                                           0F04  2537       ; The DIAG bit is set.                                              [71]
                                           0F04  2538       ;-                                                                  [69]
                                           0F04  2539
                00000879'EF    00    E1    0F04  2540       Bbc     #Rpb$V_MenuTest, -      ; If the MenuTest bit is clear ...      [69]
                               06          0F0B  2541           L^Boot$L_R5, -          ; ... in the R5 passed from DiagBoot,   [69]
                                           0F0C  2542           100$                    ; ... branch to 100$.                   [69]
                                           0F0C  2543
                                           0F0C  2544       ;+                                                                  [69]
                                           0F0C  2545       ; The MenuTest bit is also set. This indicates that CRD-MenuTest    [69]
                                           0F0C  2546       ; should be run.                                                    [84]
                                           0F0C  2547       ;-                                                                  [69]
                                           0F0C  2548
                          50   01    C8    0F0C  2549 75$:    BISL2   #1, R0                  ; CRD MENU active                       [84]
                          50   02    C8    0F0F  2550         BISL2   #2, R0                  ; CRD MENU invoked by 'T/M' cosole command
                                           0F12  2551                                         ;                                       [84]
                                           0F12  2552
                                     05    0F12  2553 100$:   Rsb                             ; Return to caller                      [69]
```

```
                        0F13  2555          .SubTitle       DSR$Check_AutoTest_Off_Set Routine
                        0F13  2556 ;++
                        0F13  2557 ; FUNCTIONAL DESCRIPTION:
                        0F13  2558 ;
                        0F13  2559 ;       This routine checks to see if CRD AutoTest Offline is active.
                        0F13  2560 ;       It checks the Boot$L_R5 longword or DSA Flags.
                        0F13  2561 ;       The value in Boot$L_R5 is the value that
                        0F13  2562 ;       is passes to the VDS from DiagBoot (in R5).
```

```
                        0F13  2564 ;
                        0F13  2565 ; CALLING SEQUENCE:
                        0F13  2566 ;
                        0F13  2567 ;       Jsb/Bsbw/Bsbb DSR$Check_AutoTest_Off_Set
```

```
                                0F13  ;
                                0F13  ; INPUT PARAMETERS:
                                0F13  ;
                                0F13  ;       NONE
                                0F13  ;
                                0F13  ; IMPLICIT INPUTS:
                                0F13  ;
                                0F13  ;       Boot$L_R5
                                0F13  ;       DSA$GL_Flags
                                0F13  ;
                                0F13  ; OUTPUT PARAMETERS:
                                0F13  ;
                                0F13  ;       NONE
                                0F13  ;
                                0F13  ; IMPLICIT OUTPUTS:
                                0F13  ;
                                0F13  ;       NONE
                                0F13  ;
                                0F13  ; COMPLETION CODES:
                                0F13  ;
                                0F13  ;       R0 = 1: CRD AUTO active
                                0F13  ;       R0 = 0: CRD AUTO not active
                                0F13  ;--
                                0F13  DSR$Check_AutoTest_Off_Set::          ;                                    [80]
                                0F13
                     50    D4   0F13          ClrL    R0                     ; Assume they are not set            [69]
                                0F15
        0000FE00'EF  0B    E1   0F15          Bbc     #DSA$V_CRD_AutoTest_Off, - ; If the CRD_AutoTest_Off
                     02         0F1C                    L^DSA$GL_Flags,    -  ; ... bit is not set in the DSA flags
                                0F1D                    50$                   ; ... branch to 50$.                 [84]
                                0F1D
                     10    11   0F1D          Brb     75$                    ; The bit is set; set the flag       [79]
                                0F1F
                                0F1F          ;+                                                                  [69]
                                0F1F          ; If the AutoTest and the DIAG bits are both set in the R5 passed from [69]
                                0F1F          ; DiagBoot, set the low bit in R0 and return.                        [69]
                                0F1F          ;-                                                                  [69]
                                0F1F
        00000879'EF  04    E1   0F1F  50$:    Bbc     #Rpb$V_Diag, -         ; If the DIAG bit is clear ...       [79]
                     0A         0F26                    L^Boot$L_R5, -        ; ... in the R5 passed from DiagBoot, [69]
                                0F27                    100$                  ; ... branch to 100$.                [69]
                                0F27
                                0F27          ;+                                                                  [69]
                                0F27          ; The DIAG bit is set.                                              [69]
                                0F27          ;-                                                                  [69]
                                0F27
        00000879'EF  0F    E1   0F27          Bbc     #Rpb$V_AutoTest, -     ; If the AutoTest bit is clear ...   [69]
                     02         0F2E                    L^Boot$L_R5, -        ; ... in the R5 passed from DiagBoot, [69]
                                0F2F                    100$                  ; ... branch to 100$.                [69]
                                0F2F
                                0F2F          ;+                                                                  [69]
                                0F2F          ; The AutoTest bit is also set. This indicates that CRD-AutoTest    [69]
                                0F2F          ; should be run. Return a 1 in R0.                                  [69]
                                0F2F          ;-                                                                  [69]
                                0F2F
                     50    D6   0F2F  75$:    IncL    R0                     ; CRD AutoTest is running.           [79]
                                0F31
```

```
 05  0F31  2626 100$:   Rsb                          ; Return to caller                    [69]
     0F32  2627
```

```
                                0F32  2629 .sbttl DSX$GETTERM  Get User Terminal Characteristics                    [87]
                                0F32  2630
                                0F32  2631 ;++                                                                    [87]
                                0F32  2632 ; FUNCTIONAL DESCRIPTION:                                              [87]
                                0F32  2633 ;                                                                      [87]
                                0F32  2634 ;       This routine simply passes the user terminal's characteristics [87]
                                0F32  2635 ;       to the caller.  The characteristics are stored in bits 0 through[87]
                                0F32  2636 ;       23.  They are defined by the $TTDEF macro of VMS.  The charac- [87]
                                0F32  2637 ;       teristics are determined at startup time and stored in         [87]
                                0F32  2638 ;       DS$GL_TERMCHAR.                                                [87]
                                0F32  2639 ;                                                                      [87]
                                0F32  2640 ; CALLING SEQUENCE:                                                    [87]
                                0F32  2641 ;                                                                      [87]
                                0F32  2642 ;       PUSHAL  addr                                                   [87]
                                0F32  2643 ;       CALLS   #1,DSX$GETTERM                                         [87]
                                0F32  2644 ;                                                                      [87]
                                0F32  2645 ; INPUT PARAMETERS:                                                    [87]
                                0F32  2646 ;                                                                      [87]
                                0F32  2647 ;       4(AP)   Address of longword in which to load characteristics   [87]
                                0F32  2648 ;                                                                      [87]
                                0F32  2649 ; IMPLICIT INPUTS:                                                     [87]
                                0F32  2650 ;                                                                      [87]
                                0F32  2651 ;       DS$GL_TERMCHAR  Longword containing terminal characteristics   [87]
                                0F32  2652 ;                                                                      [87]
                                0F32  2653 ; OUTPUT PARAMETERS:                                                   [87]
                                0F32  2654 ;                                                                      [87]
                                0F32  2655 ;       NONE                                                           [87]
                                0F32  2656 ;                                                                      [87]
                                0F32  2657 ; INPLICIT OUTPUTS:                                                    [87]
                                0F32  2658 ;                                                                      [87]
                                0F32  2659 ;       NONE                                                           [87]
                                0F32  2660 ;                                                                      [87]
                                0F32  2661 ; COMPLETION CODES:                                                    [87]
                                0F32  2662 ;                                                                      [87]
                                0F32  2663 ;       SS$_NORMAL - Service successfully completed.                   [87]
                                0F32  2664 ;                                                                      [87]
                                0F32  2665 ; SIDE EFFECTS:                                                        [87]
                                0F32  2666 ;                                                                      [87]
                                0F32  2667 ;       NONE                                                           [87]
                                0F32  2668 ;                                                                      [87]
                                0F32  2669
                                0F32  2670
                          0000  0F32  2671 .ENTRY DSX$GETTERM, ^M<>                 ;                            [87]
                                0F34  2672
04 BC    00000B83'EF      7D   0F34  2673         MOVQ    DS$GL_TERMCHAR, @4(AP)  ; Load term. characteristics  [87]
            50    01      D0   0F3C  2674         MOVL    #SS$_NORMAL,R0          ; Indicate successful return. [87]
                  04      04   0F3F  2675         RET                             ; Done.                       [87]
                               0F40  2676
                               0F40  2677         .END
```

```
$$.TAB                          = 00000150 R  D  03
$$.TABEND                       = 00000194 R  D  03
$$.TMP                          = 80000000    D
$$.TMP1                         = 00000001    D
$$.TMP2                         = 000000CF
$$ARGS                          = 00000003    D
$$N                             = 00000001    D
$$T1                            = 00000001    D
$$T2                            = 00000004    D
$ER                             = 00000003    D
$MODULE                           0000016B R  D  02
A                                 00000835 R  D  03
AP$B_PROMPT_BUFFER_COUNT          ********W GX   00
AP$B_PROMPT_COUNT                 ********W GX   00
AP$GK_PROMPT_WAIT               = 00FFFFFF    D
AP$GR_SHARED_DATA                 ********   X   04
AP$GW_DATA                        ********   X   04
AP$I_PROMPT_CHECK                 ********W GX   00
AP$M_AP_NODE_ID                 = 000000F0    D
AP$M_AP_RUNNING                 = 00000100    D
AP$M_BUFFER_FULL                = 00000800    D
AP$M_DATA_READY                 = 00000200    D
AP$M_PP_NODE_ID                 = 0000000F    D
AP$M_REQUEST                    = 00000400    D
AP$S_AP_NODE_ID                 = 00000004    D
AP$S_PP_NODE_ID                 = 00000004    D
AP$V_AP_NODE_ID                 = 00000004    D
AP$V_AP_RUNNING                 = 00000008    D
AP$V_BUFFER_FULL                = 0000000B    D
AP$V_DATA_READY                 = 00000009    D
AP$V_PP_NODE_ID                 = 00000000    D
AP$V_REQUEST                    = 0000000A    D
AP$_REQ_CONSRX                  = 00000000  G D
AP$_REQ_RXCDVEC                 = 00000001  G D
APT                               000000D9 RG D  04
AX_BUFF                           ********   X   04
BEGIN                             0000086D RG D  04
BEGIN0                            0000067F R  D  04
BEGIN_BLISS                       0000086D RG D  04
BEGIN_TIME                        ********   X   04
BIN_TIME                          ********   X   04
BIT...                          = 0000000C    D
BOOT                              00000000 RG D  04
BOOT$A_CCA                        00000889 RG D  03
BOOT$L_ADP                        00000865 RG D  03
BOOT$L_CSR                        00000869 RG D  03
BOOT$L_DEVTYP                     0000087D RG D  03
BOOT$L_R1                         00000871 RG D  03
BOOT$L_R2                         00000875 RG D  03
BOOT$L_R5                         00000879 RG D  03
BOOT$L_SCORPIO_R3                 00000881 RG D  03
BOOT$L_UNIT                       0000086D RG D  03
CCB$B_AMOD                        00000009    D
CCB$B_STS                         00000008    D
CCB$C_LENGTH                      00000010    D
CCB$K_LENGTH                      00000010    D
CCB$L_DIRP                        0000000C    D
CCB$L_UCB                         00000000    D
CCB$L_WIND                        00000004    D
CCB$SIZE                        = 00000010    D
CCB$W_IOC                         0000000A    D
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        0000044C    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_FLAGS2                      00000444    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEMP_BASE                   00000448    D
CLI$L_TEST                        00000020    D
CLI$M_START_OR_RUN              = 00000008    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BASE_QUAL                 = 00000002    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NEXT                      = 00000001    D
CLI$V_NOALLOC                   = 00000000    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
```

```
CLI$V_START_OR_RUN              = 00000003     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
CLK$RE_INIT                       ********   X   04
CMK$RCONSOLE                      ********   X   04
CMK$REFRESH                       0000097B RG  D   04
CMK$SGIPR                         ********   X   04
CMK$_APBOOT                     = 00000004     D
CMK$_ASTEXIT                    = 00000000     D
CMK$_CANTIM                     = 0000000B     D
CMK$_HIBER                      = 00000007     D
CMK$_INITSCB                    = 00000008     D
CMK$_LAST                       = 0000000E     D
CMK$_MMENABLE                   = 00000003     D
CMK$_RCONSOLE                   = 00000001     D
CMK$_REFRESH                    = 00000005     D
CMK$_SCHDWK                     = 00000006     D
CMK$_SETIMR                     = 0000000C     D
CMK$_SETIPL                     = 00000009     D
CMK$_SETPRT                     = 0000000D     D
CMK$_SGIPR                      = 0000000A     D
CMK$_TCONSOLE                   = 00000002     D
COND_DEBUG                        ********   X   04
COND_HANDLR                       ********   X   04
CPUTIM                            00000415 R   D   03
CR                              = 0000000D     D
CRB$SIZE                        = 0000002C     D
CRD$K_CRD_INITIALIZATION        = 00000000   G D
CRD$K_DS_CONTROL_C              = 00000001   G D
CRD$K_DS_FLAGS                  = 00000000   G D
CRD$K_FAILURE                   = 00000000     D
CRD$K_HOOK_POINT                = 00000000   G D
CRD$K_LAST_ACCESS_CODE          = 00000002   G D
CRD$K_LAST_DS_VARIABLE          = 00000002   G D
CRD$K_LAST_FUNCTION             = 00000002   G D
CRD$K_LAST_HOOK_POINT           = 00000001   G D
CRD$K_READ                      = 00000000   G D
CRD$K_SUCCESS                   = 00000001     D
CRD$K_TYPECODE                  = 00000001   G D
CRD$K_WRITE                     = 00000001   G D
CRD_DESC                          00000058 R   D   03
CRD_DESC_BUF                      00000060 R   D   03
CRD_SWITCH_1                      00000000 R   D   02
CRD_SWITCH_2                      00000004 R   D   02
CRETVA$_ACMODE                  = 0000000C     D
CRETVA$_INADR                   = 00000004     D
CRETVA$_NARGS                   = 00000003     D
CRETVA$_RETADR                  = 00000008     D
CTL$GL_CCB                        0000042D RG  D   03
CTL$GL_CCBBASE                    0000082D RG  D   03
CTRL_STR                          00000BD3 R   D   03
DB_SIZ                          = 00000800     D
DDB$SIZE                        = 00000018     D
DEF$Q_DEV                         ********   X   04
DEF$Q_DIR                         ********   X   04
DEV$V_FOD                       = 0000000E     D
DEV$V_MBX                       = 00000014     D
DEVCHAR                           00000B8B R   D   03
DIB$K_LENGTH                    = 00000074     D
DIB$L_DEVCHAR                   = 00000000     D
DIB$L_DEVDEPEND                 = 00000008     D
DL_SIZ                          = 00000800     D
DM_SIZ                          = 00000A00     D
DR_SIZ                          = 00000800     D
DS$AQ_SSEND                       ********   X   02
DS$AQ_SYSSRV                      ********   X   02
DS$AT_DDEV                        000001E0 R   D   03
DS$AT_DDIR                        000001B8 R   D   03
DS$AX_ARB                         ********   X   04
DS$AX_JIB                         ********   X   04
DS$AX_SOFTPCB                     ********   X   03
DS$A_PRGBGN                     = 00000200   G D
DS$CLI                            ********   X   04
DS$CNTRLC                         ********   X   04
DS$FAB_INPUT                      0000006C RG  D   03
DS$FAB_OUTPUT                     00000100 RG  D   03
DS$GA_BUFPTR                      000008ED RG  D   03
DS$GA_CHKLPPC                     000008CD RG  D   03
DS$GA_CRD_DS_INTERFACE            ********   X   04
DS$GA_DS_CTRL_C_FIRST             00000194 RG  D   03
DS$GA_DS_CTRL_C_SECOND            00000198 RG  D   03
DS$GA_LASTADR                     00000895 RG  D   03
DS$GA_LOOPADR                     000008D1 RG  D   03
DS$GA_PTABLE                      ********   X   04
DS$GA_SELECTS                     ********   X   04
DS$GB_BYTEBUF                     000008F1 RG  D   03
DS$GB_INHIBIT_NAMING              000008F3 RG  D   03
DS$GB_MM_ENB                      ********   X   04
DS$GB_SYSTYPE                     0000040C RG  D   03
DS$GB_TYPECODE                    000008F2 RG  D   03
DS$GK_SID                         ********   X   04
DS$GL_BUFCNT                      ********   X   04
DS$GL_BUFLEN                      000008E9 RG  D   03
DS$GL_CLIBASE                     ********   X   04
DS$GL_DEVERR_COUNT                000008B1 RG  D   03
DS$GL_EFC0                        0000088D RG  D   03
DS$GL_EFC1                        00000891 RG  D   03
DS$GL_ERRCNT                      000008A1 RG  D   03
DS$GL_ERRSUP_COUNT                000008B9 RG  D   03
DS$GL_FLAGS                       00000899 RG  D   03
DS$GL_FSTTEST                     000008C1 RG  D   03
DS$GL_HARDERR_COUNT               000008A5 RG  D   03
DS$GL_LSTTEST                     000008C5 RG  D   03
DS$GL_NUMTEST                     000008BD RG  D   03
DS$GL_PCBVEC                      00000831 RG  D   03
DS$GL_PREPERR_COUNT               000008A9 RG  D   03
DS$GL_RADIX                       000008E1 RG  D   03
DS$GL_RUNTIM                      000008D5 RG  D   03
DS$GL_SIZE                        000008E5 RG  D   03
DS$GL_SOFTERR_COUNT               000008AD RG  D   03
DS$GL_SUBTEST                     000008C9 RG  D   03
DS$GL_SYSERR_COUNT                000008B5 RG  D   03
DS$GL_SYSTYPE                     0000040D R   D   03
DS$GL_TERMCHAR                    00000B83 RG  D   03
```

```
DS$GL_TIMESEC                   000008D9 RG D  03
DS$GL_WAITIM                    000008DD RG D  03
DS$GQ_BUFQWD                    000008E9 RG D  03
DS$GQ_CURYEAR                   ******** X    04
DS$GQ_DAYTIM                    ******** X    04
DS$GQ_EFL                       0000088D RG D  03
DS$GT_ASCIBUF                   000008F4 RG D  03
DS$GT_BUFFER                    00000903 RG D  03
DS$GT_COPYEAR                   0000002E RG D  03
DS$GT_COPYRIGHT                 00000000 RG D  03
DS$GT_LEGAL                     0000006E RG D  02
DS$GT_LEGAL_END                 0000004D RG D  03
DS$GT_NEWYEAR                   ******** X    04
DS$GT_PROMPT                    ******** X    04
DS$GT_START                     ******** X    04
DS$GT_STRBUF                    00000B03 RG D  03
DS$GW_TTIN                      0000089D RG D  03
DS$GW_TTOUT                     0000089F RG D  03
DS$INITSCB                      ******** X    04
DS$K_ASCIBUF                  = 0000000F  G D
DS$K_BUFSIZ                   = 00000200  G D
DS$K_CCB                      = 00000040  G D
DS$K_ERROR                    = 00000002    D
DS$K_NORMAL                   = 00000001    D
DS$K_NYSIZ                      ******** X    04
DS$K_PRGSIZ                   = 0000F800  G D
DS$K_PRINTB                   = 00000002    D
DS$K_PRINTF                   = 00000001    D
DS$K_PRINTI                   = 00000000    D
DS$K_PRINTX                   = 00000003    D
DS$K_SEVERE                   = 00000004    D
DS$K_SSSIZE                     ******** X    04
DS$K_STRBUF                   = 00000080  G D
DS$K_SUBSYS                   = 00000066    D
DS$K_TYPE_ABORT_PROGRAM       = 00000014    D
DS$K_TYPE_ABORT_TEST          = 00000013    D
DS$K_TYPE_COMMAND_ERR         = 00000015    D
DS$K_TYPE_COMMAND_OUT         = 00000016    D
DS$K_TYPE_CRD_AUTOTEST        = 0000001A    D
DS$K_TYPE_DS_PROMPT           = 00000001    D
DS$K_TYPE_DS_START            = 0000001D    D
DS$K_TYPE_ERRDEV              = 00000008    D
DS$K_TYPE_ERRHARD             = 00000006    D
DS$K_TYPE_ERROR_BODY          = 00000009    D
DS$K_TYPE_ERROR_END           = 0000000A    D
DS$K_TYPE_ERRPREP             = 0000001B    D
DS$K_TYPE_ERRSOFT             = 00000007    D
DS$K_TYPE_ERRSUP              = 00000004    D
DS$K_TYPE_ERRSYS              = 00000005    D
DS$K_TYPE_ERR_HALT            = 0000000D    D
DS$K_TYPE_EXCEPTION           = 0000000C    D
DS$K_TYPE_EXCEPTION_HEAD      = 0000000B    D
DS$K_TYPE_FIRST_PASS          = 00000011    D
DS$K_TYPE_GENERAL             = 00000000    D
DS$K_TYPE_GENERAL_ERROR       = 00000003    D
DS$K_TYPE_NO_TESTS            = 00000012    D
DS$K_TYPE_PARAM_ERROR         = 0000001C    D
DS$K_TYPE_PROGRAM_END         = 00000010    D
DS$K_TYPE_PROGRAM_INFO        = 00000017    D
DS$K_TYPE_PROGRAM_START       = 0000000F    D
DS$K_TYPE_QIO_INVADP          = 00000024    D
DS$K_TYPE_QIO_NODRIVER        = 00000022    D
DS$K_TYPE_QIO_WRONGVER        = 00000023    D
DS$K_TYPE_SCRIPT_ECHO         = 00000021    D
DS$K_TYPE_SCRIPT_PMF          = 0000001E    D
DS$K_TYPE_SCRIPT_PROMPT       = 00000020    D
DS$K_TYPE_SCRIPT_SKIP         = 0000001F    D
DS$K_TYPE_SEQUENCE_ERROR      = 00000019    D
DS$K_TYPE_START_ERR           = 00000018    D
DS$K_TYPE_START_LIST          = 00000025    D
DS$K_TYPE_SUMMARY             = 0000000E    D
DS$K_TYPE_USER_PROMPT         = 00000002    D
DS$K_WARNING                  = 00000000    D
DS$LOADPCS                      ******** X    04
DS$L_USERCNTRLC                 0000019C RG D  03
DS$M_ABRTFLG                  = 00000040    D
DS$M_BADTIME                  = 00100000    D
DS$M_BATCH                    = 00400000    D
DS$M_BRKCLR                   = 00001000    D
DS$M_BRKPT                    = 00000800    D
DS$M_CHARFLG                  = 00000100    D
DS$M_CMDFLG                   = 00000080    D
DS$M_CTRLC                    = 00000001    D
DS$M_CTRLO                    = 00010000    D
DS$M_DEVFLG                   = 00000200    D
DS$M_DISABLCC                 = 01000000    D
DS$M_DONFLG                   = 00002000    D
DS$M_ERRFLG                   = 00000010    D
DS$M_EXCEPT                   = 00080000    D
DS$M_EXETST                   = 00040000    D
DS$M_HLTFLG                   = 00000008    D
DS$M_LODFLG                   = 00000002    D
DS$M_MEMMGT                   = 00008000    D
DS$M_NI                       = 04000000    D
DS$M_OUTPUT                   = 00800000    D
DS$M_RUBFLG                   = 00000020    D
DS$M_SCRIPT                   = 00200000    D
DS$M_SETIMR                   = 02000000    D
DS$M_STRFLG                   = 00000004    D
DS$M_SUBT                     = 00004000    D
DS$M_SYSFLG                   = 00000400    D
DS$M_TIMED                    = 02000000    D
DS$M_TIMED_OUT                = 08000000    D
DS$M_TIMRON                   = 00020000    D
DS$RAB_INPUT                    000000BC RG D  03
DS$RAB_OUTPUT                   00000150 RG D  03
DS$SOFTIPL5                     000002C4 RG D  04
DS$USERMODE                     000002CF RG D  04
DS$USERMODE_X                   00000636 R  D  04
DS$V_ABRTFLG                  = 00000006    D
DS$V_BADTIME                  = 00000014    D
DS$V_BATCH                    = 00000016    D
DS$V_BRKCLR                   = 0000000C    D
DS$V_BRKPT                    = 0000000B    D
```

```
DS$V_CHARFLG                    =  00000008     D
DS$V_CMDFLG                     =  00000007     D
DS$V_CTRLC                      =  00000000     D
DS$V_CTRLO                      =  00000010     D
DS$V_DEVFLG                     =  00000009     D
DS$V_DISABLCC                   =  00000018     D
DS$V_DONFLG                     =  0000000D     D
DS$V_ERRFLG                     =  00000004     D
DS$V_EXCEPT                     =  00000013     D
DS$V_EXETST                     =  00000012     D
DS$V_HLTFLG                     =  00000003     D
DS$V_LODFLG                     =  00000001     D
DS$V_MEMMGT                     =  0000000F     D
DS$V_NI                         =  0000001A     D
DS$V_OUTPUT                     =  00000017     D
DS$V_RUBFLG                     =  00000005     D
DS$V_SCRIPT                     =  00000015     D
DS$V_SETIMR                     =  00000019     D
DS$V_STRFLG                     =  00000002     D
DS$V_SUBT                       =  0000000E     D
DS$V_SYSFLG                     =  0000000A     D
DS$V_TIMED                      =  00000019     D
DS$V_TIMED_OUT                  =  0000001B     D
DS$V_TIMRON                     =  00000011     D
DS$_AP_NORMAL_BREAK             =  00660150     D
DS$_ARITH                       =  006600D0     D
DS$_ASBE                        =  00660118     D
DS$_BADLINK                     =  006600F0     D
DS$_BADTYPE                     =  006600E8     D
DS$_BIIC                        =  00660120     D
DS$_CHME                        =  006600A8     D
DS$_CHMK                        =  006600E0     D
DS$_DEVNAME                     =  00660108     D
DS$_ERROR                       =  00660002     D
DS$_FHWE                        =  00660068     D
DS$_FRAGBUF                     =  00660080     D
DS$_ICBUSY                      =  006600C8     D
DS$_ICERR                       =  006600C0     D
DS$_IHWE                        =  00660060     D
DS$_ILLCHAR                     =  00660018     D
DS$_ILLPAGCNT                   =  00660078     D
DS$_ILLUNIT                     =  00660100     D
DS$_INIT_FAIL                   =  00660140     D
DS$_INSFMEM                     =  00660050     D
DS$_INVCPU                      =  00660130     D
DS$_IPL2HI                      =  006600B8     D
DS$_IVADDR                      =  00660040     D
DS$_IVVECT                      =  00660038     D
DS$_KRNLSTK                     =  00660090     D
DS$_LOGIC                       =  00660070     D
DS$_MCHK                        =  00660088     D
DS$_MEM_ALLOC_ERR               =  00660138     D
DS$_MMOFF                       =  00660058     D
DS$_NEEDUNIT                    =  006600F8     D
DS$_NODE                        =  00660128     D
DS$_NOPCS                       =  00660110     D
DS$_NORMAL                      =  00660001     D
DS$_NOSUPPORT                   =  006600B1     D
DS$_NOTALLOWMP                  =  00660148     D
DS$_NOTDON                      =  00660030     D
DS$_NOTIMP                      =  006600B0     D
DS$_NULLSTR                     =  00660010     D
DS$_OVERFLOW                    =  00660008     D
DS$_POWER                       =  00660098     D
DS$_PROGERR                     =  00660020     D
DS$_SEVERE                      =  00660004     D
DS$_TRANSL                      =  006600A0     D
DS$_TRUNCATE                    =  00660028     D
DS$_UNEXPINT                    =  006600D8     D
DS$_UNSUPPDEV                   =  00660158     D
DS$_VASFULL                     =  00660048     D
DS$_WARNING                     =  00660000     D
DSA$AL_APTMAIL                     0000FE00     D
DSA$AT_APTTXT                      0000FA00     D
DSA$GL_APTCOM                      0000FE04     D
DSA$GL_DEVLEN                      0000FE58     D
DSA$GL_ERRNO                       0000FE44     D
DSA$GL_EVENT                       0000FE48     D
DSA$GL_FLAGS                       0000FE00     D
DSA$GL_MSGTYP                      0000FE40     D
DSA$GL_PASSES                      0000FE08     D
DSA$GL_PASSNO                      0000FE54     D
DSA$GL_SECTNO                      0000FE10     D
DSA$GL_SID                         0000FE14     D
DSA$GL_SUBTNO                      0000FE4C     D
DSA$GL_TESTNO                      0000FE50     D
DSA$GL_UNITS                       0000FE0C     D
DSA$GQ_MSGPTR                      0000FE68     D
DSA$GT_DEVNAM                      0000FE5C     D
DSA$M_APT                       =  80000000     D
DSA$M_BINARY                    =  00000002     D
DSA$M_CRD_AUTOTEST_OFF          =  00000800     D
DSA$M_CRD_MENUTEST_OFF          =  00010000     D
DSA$M_CRD_MENUTEST_ON           =  00040000     D
DSA$M_CRD_TRACE                 =  00020000     D
DSA$M_DFLTFLGS                  =  00003000   G D
DSA$M_OPER                      =  00001000     D
DSA$M_PROMPT                    =  00002000     D
DSA$M_USER                      =  10000000     D
DSA$V_APT                       =  0000001F     D
DSA$V_CRD_AUTOTEST_OFF          =  0000000B     D
DSA$V_CRD_MENUTEST_OFF          =  00000010     D
DSA$V_USER                      =  0000001C     D
DSI$ABORT                          ********   X    04
DSQA$K_DISPAT_AFTER_INIT        =  00000014     D
DSQA$K_DISPAT_BEFORE_INIT       =  00000013     D
DSQA$K_DISPAT_CALLTEST          =  00000015     D
DSQA$K_DISPAT_DONE_TESTS        =  00000018     D
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =  00000001     D
DSQA$K_DISPAT_DSX$ENDPASS_END   =  00000002     D
DSQA$K_DISPAT_DSX$ENDPASS_MID   =  00000000     D
DSQA$K_DISPAT_DS_CLEANUP_BGN    =  00000003     D
DSQA$K_DISPAT_DS_CLEANUP_END    =  00000004     D
DSQA$K_DUMMY_1                  =  00000007     D
```

```
DSQA$K_ERROR_ERROR_BGN          =  00000006     D
DSQA$K_ERROR_ERROR_END          =  00000005     D
DSQA$K_KERNEL_INIT_CONTEXT      =  00000008     D
DSQA$K_LOOP_DSX$BGNSUB          =  00000009     D
DSQA$K_LOOP_DSX$CKLOOP          =  0000000B     D
DSQA$K_LOOP_DSX$ENDSUB          =  0000000A     D
DSQA$K_PARAM_DSX$GETADDRESS     =  0000000E     D
DSQA$K_PARAM_DSX$GETDATA        =  0000000C     D
DSQA$K_PARAM_DSX$GETLOGICAL     =  0000000F     D
DSQA$K_PARAM_DSX$GETSTRING      =  00000010     D
DSQA$K_PARAM_DSX$GETVIELD       =  0000000D     D
DSQA$K_QA_DSX$BRANCH            =  00000011     D
DSQA$K_START_BEFORE_HEADER      =  00000017     D
DSQA$K_START_VRRESTART          =  00000012     D
DSQA$K_START_VRSTART            =  00000016     D
DSR$CHECK_AUTOTEST_OFF_SET        00000F13 RG D  04
DSR$CHECK_MENUTEST_OFF_SET        00000EEF RG D  04
DSR$LOAD_CRD                      ********   X   04
DSR$RESTORE_CRD_VECTORS           ********   X   04
DSR$SETIMR_INI                    ********   X   04
DSR$UNLOAD_CRD                    ********   X   04
DSV$EXIT                          00000DEF RG D  04
DSV$SETLOAD                       ********   X   04
DSX$CNTRLC                        00000DC4 RG D  04
DSX$GETTERM                       00000F32 RG D  04
DSX$PRINT                         ********   X   04
DSX$PRINTI                        ********   X   04
DS_CLEANUP                        ********   X   04
DS_ERRSUP                         ********   X   04
DUE_TIME                          ********   X   04
ESTKPTR                           ********   X   04
EXE$GB_CPUTYPE                    ********   X   04
EXE$GL_TENUSEC                    ********   X   04
EXE$GL_UBDELAY                    ********   X   04
EXE$GQ_SYSTIME                    ********   X   04
EXIT$DESBLK                       000001A0 R   D  03
EXIT$HANDLR                       00000E70 R   D  04
FAB$C_BID                    =  00000003     D
FAB$C_BLN                    =  00000050     D
FAB$C_SEQ                    =  00000000     D
FAB$C_VAR                    =  00000002     D
FAB$L_ALQ                    =  00000010     D
FAB$L_FOP                    =  00000004     D
FAB$V_CHAN_MODE              =  00000002     D
FAB$V_CIF                    =  00000019     D
FAB$V_FILE_MODE              =  00000004     D
FAB$V_GET                    =  00000001     D
FAB$V_LNM_MODE               =  00000000     D
FAB$V_PUT                    =  00000000     D
FAB$W_GBC                    =  00000048     D
FALSE                        =  00000000     D
FAOBUF                            00000BC3 R   D  03
FAOLEN                            00000BCF R   D  03
FCB$SIZE                     =  00000030     D
GET_TIME                          00000C99 RG D  04
GLBL_SECT_AREA                    00000BBB R   D  03
GLBL_SECT_NAME                    00000BB3 R   D  03
GT_MENU_ERROR                     ********   X   04
HDW_CLOCK_SET                     ********   X   04
HP$A_DEPENDENT                    00000032     D
HP$A_DEVICE                       00000018     D
HP$A_DVA                          0000001C     D
HP$A_LINK                         00000020     D
HP$B_DRIVE                        0000000B     D
HP$B_FLAGS                        0000000A     D
HP$Q_DEVICE                       00000000     D
HP$T_DEVICE                       0000000C     D
HP$T_TYPE                         00000026     D
HP$V_ALLOC                   =  00000000     D
HP$V_WASALL                  =  00000001     D
HP$W_SIZE                         00000008     D
HP$W_VECTOR                       00000024     D
IDB$SIZE                     =  00000020     D
IMAGE_HEADER                      00000208 RG D  03
IMAGE_HEADER_END                  00000408 RG D  03
INI$BRK                           000002C2 RG D  04
INI$LOAD_DEVICE                   ********   X   04
INI$PTABLE                        ********   X   04
INIT_COMMON                       000009A4 R   D  04
INIT_CONTEXT                      0000088A RG D  04
INIT_USERMODE                     00000997 R   D  04
IO$M_CTRLCAST                     ********   X   04
IO$M_NOW                          ********   X   04
IO$_READVBLK                      ********   X   04
IO$_SETMODE                       ********   X   04
IOC$K_DIAGPID                =  00660000   G D
IOSB                              00000BAB R   D  03
IRP$SIZE                     =  00000050     D
ISTKPTR                           ********   X   04
ITEMS                             00000B9B R   D  03
JIB$L_ARB                    =  00000000     D
JPI$_CPUTIM                  =  00000407     D
JPI$_PID                     =  00000319     D
JPICPUTIM                         00000425 R   D  03
JPICPUTIMS                        00000429 R   D  03
KB_CHECK                          00000A88 RG D  04
KB_CHECK_APT                      00000AE6 RG D  04
KB_CHECK_X                        00000C98 R   D  04
KB_POLL                           ********   X   04
KSTKPTR                           ********   X   04
LF                           =  0000000A     D
MAPMEM                            ********   X   04
MASK                         =  0000000F     D
MEMPOOL$INIT                      ********   X   04
MP$GR_BOOTED_PROCESSORS           00000408 RG D  03
MP$R_ACTIVE_SERVICES              ********'W GX   00
MP$R_INTERLOCK_END                ********'W GX   00
MP$_PRIMARY_CPU_CHECK             ********'W GX   00
MP$_RESUME_ATTACHED_PROCESSORS    ********'W GX   00
MP_CLEAR_END                      000006CB R   D  04
NDS$OPEN                          ********'W GX   00
OFF                          =  00000000     D
ON                           =  00000001     D
ONLY$GL_TENUSEC                   ********   X   04
```

```
ONLY$GL_UBDELAY                 ********  X   04
ONLY_CPU                        ********  X   04
P0PT_BASE                       ********  X   04
PCB$B_ASTACT                    0000000C    D
PCB$B_ASTEN                     0000000D    D
PCB$B_PRI                       0000000B    D
PCB$B_PRIB                      0000002F    D
PCB$B_TYPE                      0000000A    D
PCB$B_WEFC                      0000002E    D
PCB$C_LENGTH                    0000008C    D
PCB$K_LENGTH                    0000008C    D
PCB$L_ARB                       00000084    D
PCB$L_ASTQBL                    00000014    D
PCB$L_ASTQFL                    00000010    D
PCB$L_EFC2P                     00000058    D
PCB$L_EFC3P                     0000005C    D
PCB$L_EFCS                      00000050    D
PCB$L_EFCU                      00000054    D
PCB$L_EFWM                      0000004C    D
PCB$L_JIB                       00000078    D
PCB$L_OWNER                     0000001C    D
PCB$L_PHD                       00000064    D
PCB$L_PHYPCB                    00000018    D
PCB$L_PID                       00000060    D
PCB$L_PQB                       0000004C    D
PCB$L_SQBL                      00000004    D
PCB$L_SQFL                      00000000    D
PCB$L_STS                       00000024    D
PCB$L_UIC                       00000088    D
PCB$L_WSSWP                     00000020    D
PCB$L_WTIME                     00000028    D
PCB$Q_PRIV                      0000007C    D
PCB$T_LNAME                     00000068    D
PCB$T_TERMINAL                  00000044    D
PCB$W_APTCNT                    00000030    D
PCB$W_ASTCNT                    00000038    D
PCB$W_BIOCNT                    0000003A    D
PCB$W_BIOLM                     0000003C    D
PCB$W_DIOCNT                    0000003E    D
PCB$W_DIOLM                     00000040    D
PCB$W_GPGCNT                    00000034    D
PCB$W_GRP                       0000008A    D
PCB$W_MEM                       00000088    D
PCB$W_MTXCNT                    0000000E    D
PCB$W_PPGCNT                    00000036    D
PCB$W_PRCCNT                    00000042    D
PCB$W_SIZE                      00000008    D
PCB$W_STATE                     0000002C    D
PCB$W_TMBU                      00000032    D
PCB_BASE                        ********  X   04
PHD$B_ASTLVL                  = 000000CF    D
PHD$L_ESP                     = 0000007C    D
PHD$L_KSP                     = 00000078    D
PHD$L_P0BR                    = 000000C8    D
PHD$L_P0LRASTL                = 000000CC    D
PHD$L_P1BR                    = 000000D0    D
PHD$L_P1LR                    = 000000D4    D
PHD$L_PC                      = 000000C0    D
PHD$L_PSL                     = 000000C4    D
PHD$L_R0                      = 00000088    D
PHD$L_R12                     = 000000B8    D
PHD$L_SSP                     = 00000080    D
PHD$L_USP                     = 00000084    D
PHD_BASE                        ********  X   04
PID                             00000B8F R  D   03
PIDNO                           00000B93 R  D   03
PIDNOS                          00000B97 R  D   03
POLL_TIME                       ********  X   04
PPA$GB_CPU                      ********W GX   00
PPA$GB_MID                      ********W GX   00
PPA_SPID                      = 3FFFE400    D
PR$_ASTLVL                    = 00000013    D
PR$_ESP                       = 00000001    D
PR$_IPL                       = 00000012    D
PR$_ISP                       = 00000004    D
PR$_KSP                       = 00000000    D
PR$_MAPEN                     = 00000038    D
PR$_P0BR                      = 00000008    D
PR$_P0LR                      = 00000009    D
PR$_P1BR                      = 0000000A    D
PR$_P1LR                      = 0000000B    D
PR$_PCBB                      = 00000010    D
PR$_SCBB                      = 00000011    D
PR$_SID                       = 0000003E    D
PR$_SID_TYP8NN                = 00000006    D
PR$_SID_TYP8RR                = 00000011    D
PR$_SID_TYP8SS                = 00000005    D
PR$_SID_TYPRTUV2              = 00000010  G D
PR$_SID_TYPUV2                = 00000008    D
PR$_SSP                       = 00000002    D
PR$_SYS_TYP800                = 00000002  G D
PR$_SYS_TYP900                = 00000003  G D
PR$_SYS_TYP_LLL               = 0000000A  G D
PR$_TXCS                      = 00000022    D
PR$_TXDB                      = 00000023    D
PR$_USP                       = 00000003    D
PR780$_TODR                   = 0000001B    D
PR8SS$_BINID                  = 0000005E    D
PRGBUF                          000001A7 R  D   02
PSL$M_IPL                     = 001F0000    D
PSL$V_IS                      = 0000001A    D
PSL$V_TBIT                    = 00000004    D
PSL_SAVE                        00000993 R  D   04
QA$MAIN                         ********    X   04
QIO$INITIALIZE                  ********    X   04
QIO$MINCORE                   = 00005E3C  G D
RAB$B_RAC                     = 0000001E    D
RAB$C_BID                     = 00000001    D
RAB$C_BLN                     = 00000044    D
RAB$C_SEQ                     = 00000000    D
RAB$L_CTX                     = 00000018    D
RAB$L_ROP                     = 00000004    D
RAB$V_CCO                     = 0000001F    D
RAB$V_CVT                     = 0000001A    D
```

```
RCTRLC                          000009E9 RG D  04
RECEIVE_POLL                    ********W GX   00
RMS$INIT                        ********  X    04
RPB$B_BOOTNDT                   000000A1    D
RPB$B_CONFREG                   00000090    D
RPB$B_DEVTYP                    00000066    D
RPB$B_HDRPGCNT                  000000A0    D
RPB$B_RODEVTYP                  0000001C    D
RPB$B_SLAVE                     00000067    D
RPB$B_WAIT                      00000100    D
RPB$C_LENGTH                    00000104    D
RPB$K_LENGTH                    00000104    D
RPB$L_ADPPHY                    0000005C    D
RPB$L_ADPVIR                    00000060    D
RPB$L_BASE                      00000000    D
RPB$L_BOOTR0                    0000001C    D
RPB$L_BOOTR1                    00000020    D
RPB$L_BOOTR2                    00000024    D
RPB$L_BOOTR3                    00000028    D
RPB$L_BOOTR4                    0000002C    D
RPB$L_BOOTR5                    00000030    D
RPB$L_BUGCHK                    000000FC    D
RPB$L_CHKSUM                    00000008    D
RPB$L_CSRPHY                    00000054    D
RPB$L_CSRVIR                    00000058    D
RPB$L_FILLBN                    0000003C    D
RPB$L_FILSIZ                    00000040    D
RPB$L_HALTCODE                  00000018    D
RPB$L_HALTPC                    00000010    D
RPB$L_HALTPSL                   00000014    D
RPB$L_IOVEC                     00000034    D
RPB$L_IOVECSZ                   00000038    D
RPB$L_ISP                       000000A4    D
RPB$L_MEMDSC                    000000BC    D
RPB$L_PCBB                      000000A8    D
RPB$L_PFNCNT                    0000004C    D
RPB$L_RESTART                   00000004    D
RPB$L_RSTRTFLG                  0000000C    D
RPB$L_SBR                       000000AC    D
RPB$L_SCBB                      000000B0    D
RPB$L_SISR                      000000B4    D
RPB$L_SLR                       000000B8    D
RPB$L_SVASPT                    00000050    D
RPB$M_MENUTEST                = 00000001    D
RPB$Q_PFNMAP                    00000044    D
RPB$T_FILE                      00000068    D
RPB$V_AUTOTEST                = 0000000F    D
RPB$V_DIAG                    = 00000004    D
RPB$V_INIBPT                  = 00000002    D
RPB$V_MENUTEST                = 00000000    D
RPB$W_ROUBVEC                   0000001E    D
RPB$W_UNIT                      00000064    D
RX50_DISK_SELECT                00000885 RG D  03
SAVEFP                          000001B4 R  D  03
SCB$L_ACCESS                    00000020    D
SCB$L_ARITH                     00000034    D
SCB$L_BREAK                     0000002C    D
SCB$L_CHME                                                00000044    D
SCB$L_CHMK                                                00000040    D
SCB$L_CHMS                                                00000048    D
SCB$L_CHMU                                                0000004C    D
SCB$L_COMPAT                                              00000030    D
SCB$L_KNLSTK                                              00000008    D
SCB$L_MACHCHK                                             00000004    D
SCB$L_OPCCUS                                              00000014    D
SCB$L_OPCDEC                                              00000010    D
SCB$L_POWER                                               0000000C    D
SCB$L_RADRMOD                                             0000001C    D
SCB$L_ROPRAND                                             00000018    D
SCB$L_RXDB                                                000000F8    D
SCB$L_SFTLVL1                                             00000084    D
SCB$L_SFTLVL10                                            000000A8    D
SCB$L_SFTLVL11                                            000000AC    D
SCB$L_SFTLVL12                                            000000B0    D
SCB$L_SFTLVL13                                            000000B4    D
SCB$L_SFTLVL14                                            000000B8    D
SCB$L_SFTLVL15                                            000000BC    D
SCB$L_SFTLVL2                                             00000088    D
SCB$L_SFTLVL3                                             0000008C    D
SCB$L_SFTLVL4                                             00000090    D
SCB$L_SFTLVL5                                             00000094    D
SCB$L_SFTLVL6                                             00000098    D
SCB$L_SFTLVL7                                             0000009C    D
SCB$L_SFTLVL8                                             000000A0    D
SCB$L_SFTLVL9                                             000000A4    D
SCB$L_TBIT                                                00000028    D
SCB$L_TIMER                                               000000C0    D
SCB$L_TRANSL                                              00000024    D
SCB$L_TXDB                                                000000FC    D
SCB$L_VM                                                  00000038    D
SCB$L_ZERO                                                00000000    D
SCB_IMAGE                                                 ********  X    04
SCH$GL_CURPCB                                             00000831 RG D  03
SCRIPT$INIT                                               ********  X    04
SCRIPT$STOP                                               ********  X    04
SEC$M_GBL                                       =         00000001    D
SEC$M_PAGFIL                                    =         00080000    D
SEC$M_WRT                                       =         00000008    D
SF$L_SAVE_PC                                    =         00000010    D
SGN$C_IRPCNT                                    =         00000040  G D
SGN$C_NPAGEDYN                                  =         0000263C  G D
SIZ...                                          =         00000001    D
SIZE_TERM                                                 ********  X    04
SS$_NORMAL                                      =         00000001    D
SSPROT                                                    0000017A R  D  02
SSTKPTR                                                   ********  X    04
START_RUN_PC                                              00000411 R  D  03
SUPBUF                                                    0000019F R  D  02
SWI$GL_FQBL                                               00000839 RG D  03
SWI$GL_FQFL                                               00000835 RG D  03
SYI$_SID                                        =         00001001    D
SYS$ASSIGN                                                ********  X    04
SYS$BINTIM                                                ********  GX   04
SYS$CLREF                                                 ********  GX   04
```

```
SYS$CONNECT                    ********  GX   04
SYS$CREATE                     ********  GX   04
SYS$CRELOG                     ********   X   04
SYS$CRETVA                   = 800000C8  G D
SYS$CRMPSC                     ********  GX   04
SYS$DALLOC                     ********  GX   04
SYS$DCLEXH                     ********  GX   04
SYS$DELTVA                   = 80000110  G D
SYS$DISK                       ********   X   04
SYS$DXLEXH                   = 800000F0    D
SYS$EXIT                       ********  GX   04
SYS$FAO                        ********   X   04
SYS$GETCHN                     ********  GX   04
SYS$GETJPI                     ********  GX   04
SYS$GETJPIW                    ********  GX   04
SYS$GETSYI                     ********  GX   04
SYS$INPUT                      0000018A R  D  02
SYS$OPEN                       ********  GX   04
SYS$OUTPUT                     00000194 R  D  02
SYS$QIOW                       ********  GX   04
SYS$SETAST                     ********  GX   04
SYS$SETDDIR                    ********   X   04
SYS$SETEXV                   = 80000208  G D
SYS$SETPRT                   = 80000230    D
SYS$TRNLOG                     ********   X   04
SYSVEC                         00000172 R  D  02
SYS_TYPE                     = 20040004    D
TICK_NS                        ********   X   04
TM_SIZ                       = 00001000    D
TODR_BIAS                      ********   X   04
TODR_BYTE_CNT                  ********W GX   00
TODR_STORE                     ********W GX   00
TRAN_ASSIGN                    00000637 R  D  04
TRUE                         = 00000001    D
TXCS$V_RDY                     ********W GX   00
T_BOTH_CRD_BITS_SET            0000003E R  D  02
T_WRONGCPU                     0000000E R  D  02
UCB$SIZE                     = 00000098    D
ULTRIX_END                     00000058 R  D  03
ULTRIX_FILE                    0000004D R  D  03
ULTRIX_PARTITION               00000051 R  D  03
USTKPTR                        ********   X   04
VCB$SIZE                     = 00000050    D
V_ENV                        = 00000001    D
V_LVL                        = 00000000    D
WCB$SIZE                     = 00000030    D
XDELBPT                        ********W GX   00
XDELTBIT                       ********W GX   00
XF_SIZ                       = 00000600    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	000001AF (431.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
WORK	00000BE2 (3042.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	00000F40 (3904.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$.TAB	=00000150-R	877 (1)	868 (1) 871 (1) 874 (1) 877 (1)
$$.TABEND	=00000194-R	877 (1)	868 (1) 871 (1) 874 (1) 877 (1)
$$.TMP	=80000000	877 (1)	868 (1) 871 (1) 874 (1) 877 (1)
$$.TMP1	=00000001	1404 (5)	1398 (5) 1400 (5) 1402 (5) 1404 (5)
$$.TMP2	=000000CF	1404 (5)	1398 (5) 1400 (5) 1402 (5) 1404 (5)
$$ARGS	=00000003	1050 (1)	1050 (1)
$$N	=00000001	1784 (6)	1238 (5) #-1694 (5) #-1745 (6) #-1757 (6)
			#-1762 (6) 1767 (6) #-1784 (6)
$$T1	=00000001	2212 (6)	1050 (1) 1353 (5) 1431 (5) 1442 (5)
			1455 (5) 1516 (5) 1963 (6) 1966 (6)
			2212 (6)
$$T2	=00000004	1447 (5)	1447 (5)
$ER	=00000003	2215 (6)	#-1958 (6) #-2215 (6)
$MODULE	0000016B-R	1046 (1)	1958 (6) 2215 (6)
A	00000835-R	944 (1)	944 (1) 947 (1)
AP$B_PROMPT_BUFFER_COUNT	00000000-XR		#-2394 (6) 705 (1)
AP$B_PROMPT_COUNT	00000000-XR		#-2395 (6) 705 (1)
AP$GK_PROMPT_WAIT	=00FFFFFF	761 (1)	
AP$GR_SHARED_DATA	00000000-XR		#-2387 (6) #-2389 (6)
AP$GW_DATA	00000000-XR		2380 (6) #-2391 (6) 2393 (6) 2397 (6)
AP$I_PROMPT_CHECK	00000000-XR		2388 (6) 705 (1)
AP$M_AP_NODE_ID	=000000F0	761 (1)	
AP$M_AP_RUNNING	=00000100	761 (1)	
AP$M_BUFFER_FULL	=00000800	761 (1)	
AP$M_DATA_READY	=00000200	761 (1)	
AP$M_PP_NODE_ID	=0000000F	761 (1)	
AP$M_REQUEST	=00000400	761 (1)	#-2390 (6)
AP$S_AP_NODE_ID	=00000004	761 (1)	#-2380 (6)
AP$S_PP_NODE_ID	=00000004	761 (1)	
AP$V_AP_NODE_ID	=00000004	761 (1)	#-2380 (6)
AP$V_AP_RUNNING	=00000008	761 (1)	#-2380 (6) #-2397 (6)
AP$V_BUFFER_FULL	=0000000B	761 (1)	
AP$V_DATA_READY	=00000009	761 (1)	
AP$V_PP_NODE_ID	=00000000	761 (1)	
AP$V_REQUEST	=0000000A	761 (1)	#-2392 (6)
AP$_REQ_CONSRX	=00000000	761 (1)	
AP$_REQ_RXCDVEC	=00000001	761 (1)	#-2386 (6)
APT	000000D9-R	1161 (5)	
AX_BUFF	00000000-XR		1198 (5)
BEGIN	0000086D-R	1786 (6)	#-1783 (6) #-1796 (6) #-1910 (6) #-2400 (6)
BEGIN0	0000067F-R	1599 (5)	#-1313 (5) #-1530 (5)
BEGIN_BLISS	0000086D-R	1787 (6)	
BEGIN_TIME	00000000-XR		#-2132 (6) #-2134 (6) 2142 (6) #-2146 (6)
BIN_TIME	00000000-XR		#-2129 (6) #-2137 (6) #-2139 (6) #-2147 (6)
			#-2149 (6)
BIT...	=0000000C	761 (1)	731 (1) 753 (1) 755 (1) 757 (1)
			759 (1) 760 (1) 761 (1)
BOOT	00000000-R	1107 (5)	
BOOT$A_CCA	00000889-R	967 (1)	#-1134 (5)
BOOT$L_ADP	00000865-R	954 (1)	#-1117 (5) #-1123 (5)

```
BOOT$L_CSR                  00000869-R   955    (1)    #-1124   (5)
BOOT$L_DEVTYP               0000087D-R   960    (1)    #-1130   (5)
BOOT$L_R1                   00000871-R   957    (1)    #-1126   (5)
BOOT$L_R2                   00000875-R   958    (1)    #-1127   (5)
BOOT$L_R5                   00000879-R   959    (1)    #-1128   (5)      1138   (5)      1211   (5)      1646   (5)
                                                         1648   (5)      1650   (5)    #-1698   (5)      2533   (6)
                                                         2541   (6)      2608   (8)      2616   (8)
BOOT$L_SCORPIO_R3           00000881-R   962    (1)
BOOT$L_UNIT                 0000086D-R   956    (1)    #-1125   (5)
CCB$C_LENGTH                00000010                     929    (1)
CCB$SIZE                   =00000010     782    (1)      793    (1)
CLI$K_BUFSIZ               =00000100     752    (1)      752    (1)
CLI$K_SIZE                  0000044C     752    (1)
CLI$L_ADDRESS               00000018     752    (1)
CLI$L_COMMAND               00000004     752    (1)
CLI$L_DATA                  0000001C     752    (1)
CLI$L_FLAGS                 00000000     752    (1)
CLI$L_FLAGS2                00000444     752    (1)      2116   (6)    #-2153   (6)    #-2162   (6)
CLI$L_LAST                  00000024     752    (1)
CLI$L_NEXT                  00000030     752    (1)
CLI$L_PASS                  0000002C     752    (1)
CLI$L_SUBT                  00000028     752    (1)
CLI$L_TEMP_BASE             00000448     752    (1)
CLI$L_TEST                  00000020     752    (1)
CLI$M_START_OR_RUN         =00000008     752    (1)    #-2152   (6)    #-2161   (6)
CLI$Q_BUFQWD                00000034     752    (1)
CLI$Q_FILE                  00000008     752    (1)    #-1154   (5)
CLI$Q_SECTION               00000010     752    (1)
CLI$Q_TIME                  0000043C     752    (1)
CLI$T_BUFFER                0000003C     752    (1)
CLI$V_ADAPTER              =00000018     752    (1)
CLI$V_ADR                  =0000000B     752    (1)
CLI$V_ASCII                =00000013     752    (1)
CLI$V_BASE_QUAL            =00000002     752    (1)
CLI$V_BREAK                =0000000A     752    (1)
CLI$V_BRIEF                =0000001B     752    (1)
CLI$V_BYTE                 =0000000D     752    (1)
CLI$V_CLEAR                =00000002     752    (1)
CLI$V_DEC                  =00000010     752    (1)
CLI$V_DEFAULT              =0000000C     752    (1)
CLI$V_DEPOSIT              =00000019     752    (1)
CLI$V_EVENT                =00000008     752    (1)
CLI$V_EXAM                 =00000005     752    (1)
CLI$V_FLAGS                =00000009     752    (1)
CLI$V_HEX                  =00000012     752    (1)
CLI$V_KERNEL               =00000017     752    (1)
CLI$V_LOAD                 =00000006     752    (1)
CLI$V_LONG                 =0000000F     752    (1)
CLI$V_NEXT                 =00000001     752    (1)
CLI$V_NOALLOC              =00000000     752    (1)
CLI$V_NOTNUF               =00000001     752    (1)
CLI$V_OCT                  =00000011     752    (1)
CLI$V_PREG                 =0000001A     752    (1)
CLI$V_QA                   =00000007     752    (1)
CLI$V_QACKLOOPLOOPS        =0000001C     752    (1)
CLI$V_QAERRORPRINTS        =0000001B     752    (1)
CLI$V_QAMULTIPLEPASS       =0000001F     752    (1)
```

```
CLI$V_QASUBTESTLOOPS            =0000001E     752    (1)
CLI$V_QATESTLOOPS               =0000001D     752    (1)
CLI$V_REG                       =00000014     752    (1)
CLI$V_REQUIRED                  =00000000     752    (1)
CLI$V_RUN                       =00000015     752    (1)
CLI$V_SET                       =00000003     752    (1)
CLI$V_SHOW                      =00000004     752    (1)
CLI$V_START_OR_RUN              =00000003     752    (1)    #-2115   (6)      752    (1)
CLI$V_VALSEC                    =00000016     752    (1)
CLI$V_WORD                      =0000000E     752    (1)
CLK$RE_INIT                      00000000-XR                  1281   (5)
CMK$RCONSOLE                     00000000-XR                  2265   (6)
CMK$REFRESH                      0000097B-R   1888   (6)     1858   (6)
CMK$SGIPR                        00000000-XR                  2234   (6)      2258   (6)      2263   (6)
CMK$_APBOOT                     =00000004     753    (1)
CMK$_ASTEXIT                    =00000000     753    (1)
CMK$_CANTIM                     =0000000B     753    (1)
CMK$_HIBER                      =00000007     753    (1)
CMK$_INITSCB                    =00000008     753    (1)
CMK$_LAST                       =0000000E     753    (1)
CMK$_MMENABLE                   =00000003     753    (1)
CMK$_RCONSOLE                   =00000001     753    (1)    #-2265   (6)
CMK$_REFRESH                    =00000005     753    (1)    #-1858   (6)
CMK$_SCHDWK                     =00000006     753    (1)
CMK$_SETIMR                     =0000000C     753    (1)
CMK$_SETIPL                     =00000009     753    (1)
CMK$_SETPRT                     =0000000D     753    (1)
CMK$_SGIPR                      =0000000A     753    (1)    #-2234   (6)    #-2258   (6)    #-2263   (6)
CMK$_TCONSOLE                   =00000002     753    (1)
COND_DEBUG                       00000000-XR                  1353   (5)
COND_HANDLR                      00000000-XR                  1354   (5)      1914   (6)
CPUTIM                           00000415-R    914    (1)     2212   (6)
CR                              =0000000D     772    (1)
CRB$SIZE                        =0000002C     779    (1)      790   (1)
CRD$K_CRD_INITIALIZATION        =00000000     760    (1)    #-1727   (6)
CRD$K_DS_CONTROL_C              =00000001     760    (1)
CRD$K_DS_FLAGS                  =00000000     760    (1)
CRD$K_FAILURE                   =00000000     760    (1)
CRD$K_HOOK_POINT                =00000000     760    (1)    #-1728   (6)
CRD$K_LAST_ACCESS_CODE          =00000002     760    (1)
CRD$K_LAST_DS_VARIABLE          =00000002     760    (1)
CRD$K_LAST_FUNCTION             =00000002     760    (1)
CRD$K_LAST_HOOK_POINT           =00000001     760    (1)
CRD$K_READ                      =00000000     760    (1)
CRD$K_SUCCESS                   =00000001     760    (1)
CRD$K_TYPECODE                  =00000001     760    (1)
CRD$K_WRITE                     =00000001     760    (1)
CRD_DESC                         00000058-R    860    (1)    #-1705   (5)
CRD_DESC_BUF                     00000060-R    864    (1)      863    (1)
CRD_SWITCH_1                     00000000-R    827    (1)
CRD_SWITCH_2                     00000004-R    829    (1)     1705   (5)
CRETVA$_ACMODE                  =0000000C     1050   (1)
CRETVA$_INADR                   =00000004     1050   (1)
CRETVA$_NARGS                   =00000003     1050   (1)
CRETVA$_RETADR                  =00000008     1050   (1)
CTL$GL_CCB                       0000042D-R    928    (1)
CTL$GL_CCBBASE                   0000082D-R    931    (1)      932    (1)
```

```
CTRL_STR                        00000BD3-R   1039    (1)       1447    (5)
DB_SIZ                         =00000800     797    (1)        804    (1)
DDB$SIZE                       =00000018     778    (1)        789    (1)
DEF$Q_DEV                       00000000-XR                  #-1484    (5)    #-1485    (5)    #-1486    (5)    #-1487    (5)
                                                             #-1488    (5)      1492    (5)
DEF$Q_DIR                       00000000-XR                  #-1489    (5)      1501    (5)
DEV$V_FOD                      =0000000E                     #-1397    (5)
DEV$V_MBX                      =00000014                     #-1436    (5)    #-1959    (6)
DEVCHAR                         00000B8B-R   1021    (1)    #-1396    (5)      1436    (5)
DIB$K_LENGTH                   =00000074                     #-1951    (6)    #-1953    (6)
DIB$L_DEVCHAR                  =00000000                       1960    (6)
DIB$L_DEVDEPEND                =00000008                     #-1961    (6)
DL_SIZ                         =00000800     798    (1)        804    (1)
DM_SIZ                         =00000A00     799    (1)        804    (1)
DR_SIZ                         =00000800     800    (1)        804    (1)
DS$AQ_SSEND                     00000000-XR                    1048    (1)    #-1381    (5)
DS$AQ_SYSSRV                    00000000-XR                    1048    (1)      1359    (5)      1367    (5)
DS$AT_DDEV                      000001E0-R    900    (1)       1473    (5)      2456    (6)
DS$AT_DDIR                      000001B8-R    898    (1)       1497    (5)      1500    (5)      2465    (6)
DS$AX_ARB                       00000000-XR                    1851    (6)
DS$AX_JIB                       00000000-XR                    1849    (6)
DS$AX_SOFTPCB                   00000000-XR                    1841    (6)    #-2033    (6)       936    (1)
DS$A_PRGBGN                    =00000200     769    (1)       1062    (5)       770    (1)
DS$CLI                          00000000-XR                    1792    (6)      2123    (6)
DS$CNTRLC                       00000000-XR                    1220    (5)      1775    (6)
DS$FAB_INPUT                    0000006C-R    867    (1)       1398    (5)       871    (1)
DS$FAB_OUTPUT                   00000100-R    873    (1)       1402    (5)       877    (1)
DS$GA_BUFPTR                    000008ED-R   1010    (1)
DS$GA_CHKLPPC                   000008CD-R   1001    (1)
DS$GA_CRD_DS_INTERFACE          00000000-XR                    1729    (6)
DS$GA_DS_CTRL_C_FIRST           00000194-R    880    (1)    #-1222    (5)    #-2080    (6)    #-2082    (6)    #-2084    (6)
DS$GA_DS_CTRL_C_SECOND          00000198-R    883    (1)    #-1223    (5)    #-2101    (6)    #-2103    (6)    #-2105    (6)
DS$GA_LASTADR                   00000895-R    971    (1)
DS$GA_LOOPADR                   000008D1-R   1002    (1)
DS$GA_PTABLE                    00000000-XR                    2441    (6)
DS$GA_SELECTS                   00000000-XR                    2442    (6)
DS$GB_BYTEBUF                   000008F1-R   1011    (1)
DS$GB_INHIBIT_NAMING            000008F3-R   1013    (1)    #-1781    (6)
DS$GB_MM_ENB                    00000000-XR                  #-1260    (5)
DS$GB_SYSTYPE                   0000040C-R    909    (1)    #-1182    (5)    #-1183    (5)
DS$GB_TYPECODE                  000008F2-R   1012    (1)    #-1905    (6)
DS$GK_SID                       00000000-XR                  #-1229    (5)    #-1519    (5)    #-1602    (5)
DS$GL_BUFCNT                    00000000-XR                  #-1257    (5)    #-1258    (5)
DS$GL_BUFLEN                    000008E9-R   1009    (1)
DS$GL_CLIBASE                   00000000-XR                    1153    (5)      2114    (6)      2151    (6)      2160    (6)
DS$GL_DEVERR_COUNT              000008B1-R    991    (1)
DS$GL_EFC0                      0000088D-R    969    (1)
DS$GL_EFC1                      00000891-R    970    (1)
DS$GL_ERRCNT                    000008A1-R    984    (1)
DS$GL_ERRSUP_COUNT              000008B9-R    995    (1)
DS$GL_FLAGS                     00000899-R    981    (1)    #-1191    (5)    #-1350    (5)      1407    (5)      1457    (5)
                                                              1462    (5)      1463    (5)    #-1967    (6)      2009    (6)
                                                             #-2024    (6)    #-2025    (6)      2045    (6)      2049    (6)
                                                             #-2086    (6)    #-2096    (6)    #-2107    (6)    #-2112    (6)
                                                             #-2150    (6)      2227    (6)      2240    (6)      2326    (6)
                                                              2356    (6)      2438    (6)
DS$GL_FSTTEST                   000008C1-R    998    (1)
```

```
DS$GL_HARDERR_COUNT             000008A5-R   985    (1)
DS$GL_LSTTEST                   000008C5-R   999    (1)
DS$GL_NUMTEST                   000008BD-R   997    (1)
DS$GL_PCBVEC                    00000831-R   935    (1)
DS$GL_PREPERR_COUNT             000008A9-R   987    (1)
DS$GL_RADIX                     000008E1-R   1006   (1)   #-1192   (5)   #-1528   (5)
DS$GL_RUNTIM                    000008D5-R   1003   (1)
DS$GL_SIZE                      000008E5-R   1007   (1)
DS$GL_SOFTERR_COUNT             000008AD-R   989    (1)
DS$GL_SUBTEST                   000008C9-R   1000   (1)
DS$GL_SYSERR_COUNT              000008B5-R   993    (1)
DS$GL_SYSTYPE                   0000040D-R   910    (1)   #-1181   (5)   #-1182   (5)
DS$GL_TERMCHAR                  00000B83-R   1018   (1)   #-1428   (5)   #-2673   (8)
DS$GL_TIMESEC                   000008D9-R   1004   (1)
DS$GL_WAITIM                    000008DD-R   1005   (1)
DS$GQ_BUFQWD                    000008E9-R   1008   (1)
DS$GQ_CURYEAR                   00000000-XR                #-1272   (5)   #-1274   (5)   1275   (5)   #-1276   (5)
                                                           #-2278   (6)   #-2279   (6)
DS$GQ_DAYTIM                    00000000-XR                #-1270   (5)   #-1758   (6)
DS$GQ_EFL                       0000088D-R   968    (1)
DS$GT_ASCIBUF                   000008F4-R   1015   (1)
DS$GT_BUFFER                    00000903-R   1016   (1)     1963   (6)
DS$GT_COPYEAR                   0000002E-R   850    (1)   #-1759   (6)
DS$GT_COPYRIGHT                 00000000-R   847    (1)     1762   (6)
DS$GT_LEGAL                     0000006E-R   836    (1)     1757   (6)
DS$GT_LEGAL_END                 0000004D-R   853    (1)     848    (1)
DS$GT_NEWYEAR                   00000000-XR                #-1271   (5)     1273   (5)
DS$GT_PROMPT                    00000000-XR                  1784   (6)
DS$GT_START                     00000000-XR                  1767   (6)
DS$GT_STRBUF                    00000B03-R   1017   (1)
DS$GW_TTIN                      0000089D-R   982    (1)   #-1388   (5)   #-1431   (5)   #-1955   (6)   #-1963   (6)
                                                           #-1966   (6)
DS$GW_TTOUT                     0000089F-R   983    (1)   #-1419   (5)
DS$INITSCB                      00000000-XR                  1247   (5)
DS$K_ASCIBUF                    =0000000F    977    (1)     1015   (1)
DS$K_BUFSIZ                     =00000200    978    (1)     1016   (1)
DS$K_CCB                        =00000040    926    (1)     929    (1)
DS$K_ERROR                      =00000002    731    (1)
DS$K_NORMAL                     =00000001    731    (1)
DS$K_NYSIZ                      00000000-XR                #-1272   (5)
DS$K_PRGSIZ                     =0000F800    770    (1)     1062   (5)
DS$K_PRINTB                     =00000002    759    (1)
DS$K_PRINTF                     =00000001    759    (1)   #-1694   (5)   #-1745   (6)   #-1757   (6)   #-1762   (6)
                                                           #-1767   (6)
DS$K_PRINTI                     =00000000    759    (1)
DS$K_PRINTX                     =00000003    759    (1)
DS$K_SEVERE                     =00000004    731    (1)
DS$K_SSSIZE                     00000000-XR                #-1358   (5)
DS$K_STRBUF                     =00000080    979    (1)     1017   (1)
DS$K_SUBSYS                     =00000066    731    (1)     731    (1)     732    (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014    759    (1)
DS$K_TYPE_ABORT_TEST            =00000013    759    (1)
DS$K_TYPE_COMMAND_ERR           =00000015    759    (1)
DS$K_TYPE_COMMAND_OUT           =00000016    759    (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A    759    (1)
DS$K_TYPE_DS_PROMPT             =00000001    759    (1)
DS$K_TYPE_DS_START              =0000001D    759    (1)   #-1757   (6)   #-1762   (6)   #-1767   (6)
```

```
DS$K_TYPE_ERRDEV                =00000008      759     (1)
DS$K_TYPE_ERRHARD               =00000006      759     (1)
DS$K_TYPE_ERROR_BODY            =00000009      759     (1)
DS$K_TYPE_ERROR_END             =0000000A      759     (1)
DS$K_TYPE_ERRPREP               =0000001B      759     (1)
DS$K_TYPE_ERRSOFT               =00000007      759     (1)
DS$K_TYPE_ERRSUP                =00000004      759     (1)
DS$K_TYPE_ERRSYS                =00000005      759     (1)
DS$K_TYPE_ERR_HALT              =0000000D      759     (1)
DS$K_TYPE_EXCEPTION             =0000000C      759     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      759     (1)
DS$K_TYPE_FIRST_PASS            =00000011      759     (1)
DS$K_TYPE_GENERAL               =00000000      759     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003      759     (1)     #-1694    (5)     #-1745    (6)
DS$K_TYPE_NO_TESTS              =00000012      759     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C      759     (1)
DS$K_TYPE_PROGRAM_END           =00000010      759     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017      759     (1)
DS$K_TYPE_PROGRAM_START         =0000000F      759     (1)
DS$K_TYPE_QIO_INVADP            =00000024      759     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022      759     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023      759     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021      759     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E      759     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      759     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      759     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      759     (1)
DS$K_TYPE_START_ERR             =00000018      759     (1)
DS$K_TYPE_START_LIST            =00000025      759     (1)
DS$K_TYPE_SUMMARY               =0000000E      759     (1)
DS$K_TYPE_USER_PROMPT           =00000002      759     (1)
DS$K_WARNING                    =00000000      731     (1)
DS$LOADPCS                       00000000-XR                    1310    (5)
DS$L_USERCNTRLC                  0000019C-R     886     (1)     #-1224    (5)     #-2091    (6)     #-2093    (6)     #-2094    (6)
                                                                #-2322    (6)
DS$M_ABRTFLG                    =00000040      755     (1)
DS$M_BADTIME                    =00100000      755     (1)
DS$M_BATCH                      =00400000      755     (1)
DS$M_BRKCLR                     =00001000      755     (1)
DS$M_BRKPT                      =00000800      755     (1)
DS$M_CHARFLG                    =00000100      755     (1)
DS$M_CMDFLG                     =00000080      755     (1)     #-1190    (5)
DS$M_CTRLC                      =00000001      755     (1)     #-1967    (6)     #-2085    (6)     #-2095    (6)     #-2106    (6)
DS$M_CTRLO                      =00010000      755     (1)     #-2085    (6)     #-2095    (6)     #-2106    (6)
DS$M_DEVFLG                     =00000200      755     (1)
DS$M_DISABLCC                   =01000000      755     (1)
DS$M_DONFLG                     =00002000      755     (1)
DS$M_ERRFLG                     =00000010      755     (1)
DS$M_EXCEPT                     =00080000      755     (1)
DS$M_EXETST                     =00040000      755     (1)
DS$M_HLTFLG                     =00000008      755     (1)
DS$M_LODFLG                     =00000002      755     (1)     #-1190    (5)
DS$M_MEMMGT                     =00008000      755     (1)
DS$M_NI                         =04000000      755     (1)
DS$M_OUTPUT                     =00800000      755     (1)
DS$M_RUBFLG                     =00000020      755     (1)
DS$M_SCRIPT                     =00200000      755     (1)
```

```
DS$M_SETIMR                 =02000000     755    (1)
DS$M_STRFLG                 =00000004     755    (1)
DS$M_SUBT                   =00004000     755    (1)
DS$M_SYSFLG                 =00000400     755    (1)
DS$M_TIMED                  =02000000     755    (1)    #-2024   (6)    #-2112   (6)    #-2150   (6)
DS$M_TIMED_OUT              =08000000     755    (1)    #-2025   (6)
DS$M_TIMRON                 =00020000     755    (1)
DS$RAB_INPUT                 000000BC-R   870    (1)      1400   (5)
DS$RAB_OUTPUT                00000150-R   876    (1)      1404   (5)
DS$SOFTIPL5                  000002C4-R   1318   (5)
DS$USERMODE                  000002CF-R   1349   (5)
DS$USERMODE_X                00000636-R   1532   (5)    #-1414   (5)
DS$V_ABRTFLG                =00000006     755    (1)
DS$V_BADTIME                =00000014     755    (1)
DS$V_BATCH                  =00000016     755    (1)    #-1406   (5)
DS$V_BRKCLR                 =0000000C     755    (1)
DS$V_BRKPT                  =0000000B     755    (1)
DS$V_CHARFLG                =00000008     755    (1)
DS$V_CMDFLG                 =00000007     755    (1)
DS$V_CTRLC                  =00000000     755    (1)    #-2044   (6)
DS$V_CTRLO                  =00000010     755    (1)
DS$V_DEVFLG                 =00000009     755    (1)
DS$V_DISABLCC               =00000018     755    (1)    #-2048   (6)    #-2327   (6)    #-2328   (6)
DS$V_DONFLG                 =0000000D     755    (1)
DS$V_ERRFLG                 =00000004     755    (1)
DS$V_EXCEPT                 =00000013     755    (1)
DS$V_EXETST                 =00000012     755    (1)
DS$V_HLTFLG                 =00000003     755    (1)
DS$V_LODFLG                 =00000001     755    (1)    #-2356   (6)    #-2438   (6)
DS$V_MEMMGT                 =0000000F     755    (1)
DS$V_NI                     =0000001A     755    (1)    #-1457   (5)    #-1462   (5)    #-1463   (5)
DS$V_OUTPUT                 =00000017     755    (1)
DS$V_RUBFLG                 =00000005     755    (1)
DS$V_SCRIPT                 =00000015     755    (1)
DS$V_SETIMR                 =00000019     755    (1)
DS$V_STRFLG                 =00000002     755    (1)
DS$V_SUBT                   =0000000E     755    (1)
DS$V_SYSFLG                 =0000000A     755    (1)
DS$V_TIMED                  =00000019     755    (1)    #-2009   (6)    #-2240   (6)
DS$V_TIMED_OUT              =0000001B     755    (1)
DS$V_TIMRON                 =00000011     755    (1)    #-2226   (6)
DS$_AP_NORMAL_BREAK         =00660150     731    (1)
DS$_ARITH                   =006600D0     731    (1)
DS$_ASBE                    =00660118     731    (1)
DS$_BADLINK                 =006600F0     731    (1)
DS$_BADTYPE                 =006600E8     731    (1)
DS$_BIIC                    =00660120     731    (1)
DS$_CHME                    =006600A8     731    (1)
DS$_CHMK                    =006600E0     731    (1)
DS$_DEVNAME                 =00660108     731    (1)
DS$_ERROR                   =00660002     731    (1)
DS$_FHWE                    =00660068     731    (1)
DS$_FRAGBUF                 =00660080     731    (1)
DS$_ICBUSY                  =006600C8     731    (1)
DS$_ICERR                   =006600C0     731    (1)
DS$_IHWE                    =00660060     731    (1)
DS$_ILLCHAR                 =00660018     731    (1)
```

```
DS$_ILLPAGCNT                =00660078     731    (1)
DS$_ILLUNIT                  =00660100     731    (1)
DS$_INIT_FAIL                =00660140     731    (1)
DS$_INSFMEM                  =00660050     731    (1)
DS$_INVCPU                   =00660130     731    (1)
DS$_IPL2HI                   =006600B8     731    (1)
DS$_IVADDR                   =00660040     731    (1)
DS$_IVVECT                   =00660038     731    (1)
DS$_KRNLSTK                  =00660090     731    (1)
DS$_LOGIC                    =00660070     731    (1)
DS$_MCHK                     =00660088     731    (1)
DS$_MEM_ALLOC_ERR            =00660138     731    (1)
DS$_MMOFF                    =00660058     731    (1)
DS$_NEEDUNIT                 =006600F8     731    (1)
DS$_NODE                     =00660128     731    (1)
DS$_NOPCS                    =00660110     731    (1)
DS$_NORMAL                   =00660001     731    (1)
DS$_NOSUPPORT                =006600B1     731    (1)
DS$_NOTALLOWMP               =00660148     731    (1)
DS$_NOTDON                   =00660030     731    (1)
DS$_NOTIMP                   =006600B0     731    (1)      731    (1)
DS$_NULLSTR                  =00660010     731    (1)
DS$_OVERFLOW                 =00660008     731    (1)
DS$_POWER                    =00660098     731    (1)
DS$_PROGERR                  =00660020     731    (1)
DS$_SEVERE                   =00660004     731    (1)
DS$_TRANSL                   =006600A0     731    (1)
DS$_TRUNCATE                 =00660028     731    (1)
DS$_UNEXPINT                 =006600D8     731    (1)
DS$_UNSUPPDEV                =00660158     731    (1)
DS$_VASFULL                  =00660048     731    (1)
DS$_WARNING                  =00660000     731    (1)      731    (1)
DSA$AL_APTMAIL                0000FE00                     1059   (3)      1112   (5)
DSA$GL_EVENT                  0000FE48                   #-2029   (6)
DSA$GL_FLAGS                  0000FE00                   #-1115   (5)    #-1522   (5)    #-1630   (5)    #-1700   (5)
                                                         #-1702   (5)    #-1704   (5)    #-1708   (5)    #-1741   (6)
                                                         #-1753   (6)      1779   (6)      1783   (6)      1836   (6)
                                                           2032   (6)      2211   (6)      2378   (6)      2522   (6)
                                                           2597   (8)
DSA$GL_SID                    0000FE14                   #-1166   (5)    #-1170   (5)    #-1175   (5)    #-1230   (5)
                                                         #-1234   (5)    #-1238   (5)      1512   (5)    #-1519   (5)
                                                         #-1696   (5)    #-2220   (6)    #-2222   (6)
DSA$M_APT                    =80000000                   #-1751   (6)
DSA$M_BINARY                 =00000002                   #-1737   (6)
DSA$M_CRD_AUTOTEST_OFF       =00000800                   #-1626   (5)    #-1702   (5)    #-1739   (6)    #-1749   (6)
DSA$M_CRD_MENUTEST_OFF       =00010000                   #-1627   (5)    #-1700   (5)    #-1738   (6)    #-1750   (6)
DSA$M_CRD_MENUTEST_ON        =00040000                   #-1628   (5)    #-1704   (5)    #-1740   (6)    #-1752   (6)
DSA$M_CRD_TRACE              =00020000                   #-1629   (5)    #-1708   (5)
DSA$M_DFLTFLGS               =00003000     767    (1)    #-1115   (5)    #-1521   (5)
DSA$M_OPER                   =00001000                     767    (1)
DSA$M_PROMPT                 =00002000                     767    (1)
DSA$M_USER                   =10000000                   #-1521   (5)
DSA$V_APT                    =0000001F                   #-1778   (6)    #-1783   (6)
DSA$V_CRD_AUTOTEST_OFF       =0000000B                   #-2596   (8)
DSA$V_CRD_MENUTEST_OFF       =00000010                   #-2521   (6)
DSA$V_USER                   =0000001C                   #-1836   (6)    #-2031   (6)    #-2211   (6)    #-2378   (6)
DSI$ABORT                     00000000-XR                  2027   (6)
```

```
DSQA$K_DISPAT_AFTER_INIT        =00000014      757     (1)
DSQA$K_DISPAT_BEFORE_INIT       =00000013      757     (1)
DSQA$K_DISPAT_CALLTEST          =00000015      757     (1)
DSQA$K_DISPAT_DONE_TESTS        =00000018      757     (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001      757     (1)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002      757     (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000      757     (1)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003      757     (1)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004      757     (1)
DSQA$K_DUMMY_1                  =00000007      757     (1)
DSQA$K_ERROR_ERROR_BGN          =00000006      757     (1)
DSQA$K_ERROR_ERROR_END          =00000005      757     (1)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008      757     (1)     #-1835    (6)
DSQA$K_LOOP_DSX$BGNSUB          =00000009      757     (1)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B      757     (1)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A      757     (1)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E      757     (1)
DSQA$K_PARAM_DSX$GETDATA        =0000000C      757     (1)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F      757     (1)
DSQA$K_PARAM_DSX$GETSTRING      =00000010      757     (1)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D      757     (1)
DSQA$K_QA_DSX$BRANCH            =00000011      757     (1)
DSQA$K_START_BEFORE_HEADER      =00000017      757     (1)
DSQA$K_START_VRRESTART          =00000012      757     (1)
DSQA$K_START_VRSTART            =00000016      757     (1)
DSR$CHECK_AUTOTEST_OFF_SET       00000F13-R    2592    (8)
DSR$CHECK_MENUTEST_OFF_SET       00000EEF-R    2517    (6)
DSR$LOAD_CRD                     00000000-XR                   1716    (6)
DSR$RESTORE_CRD_VECTORS          00000000-XR                   1619    (5)
DSR$SETIMR_INI                   00000000-XR                   1282    (5)
DSR$UNLOAD_CRD                   00000000-XR                   2372    (6)
DSV$EXIT                         00000DEF-R    2355    (6)     1695    (5)     1747    (6)
DSV$SETLOAD                      00000000-XR                   1155    (5)
DSX$CNTRLC                       00000DC4-R    2320    (6)
DSX$GETTERM                      00000F32-R    2671    (8)
DSX$PRINT                        00000000-XR                   1694    (5)     1745    (6)     1757    (6)     1762    (6)
                                                               1767    (6)
DSX$PRINTI                       00000000-XR                   1238    (5)     1784    (6)
DS_CLEANUP                       00000000-XR                   2357    (6)     2439    (6)
DS_ERRSUP                        00000000-XR                   1958    (6)     2215    (6)
DUE_TIME                         00000000-XR                 #-2017    (6)   #-2021    (6)   #-2146    (6)   #-2147    (6)
                                                             #-2149    (6)
ESTKPTR                          00000000-XR                 #-1863    (6)
EXE$GB_CPUTYPE                   00000000-XR                 #-1176    (5)   #-1177    (5)   #-1179    (5)
EXE$GL_TENUSEC                   00000000-XR                 #-1284    (5)
EXE$GL_UBDELAY                   00000000-XR                 #-1285    (5)
EXE$GQ_SYSTIME                   00000000-XR                 #-1276    (5)   #-2229    (6)   #-2246    (6)
EXIT$DESBLK                      000001A0-R    889     (1)     1357    (5)
EXIT$HANDLR                      00000E70-R    2436    (6)      891    (1)
FAB$C_BID                       =00000003                      868    (1)      874    (1)
FAB$C_BLN                       =00000050                      868    (1)      874    (1)
FAB$C_SEQ                       =00000000                      868    (1)      874    (1)
FAB$C_VAR                       =00000002                      868    (1)      874    (1)
FAB$L_ALQ                       =00000010                      868    (1)      874    (1)
FAB$L_FOP                       =00000004                      868    (1)      874    (1)
FAB$V_CHAN_MODE                 =00000002                      868    (1)      874    (1)
FAB$V_CIF                       =00000019                      874    (1)
```

```
FAB$V_FILE_MODE                 =00000004                       868    (1)      874    (1)
FAB$V_GET                       =00000001                       868    (1)
FAB$V_LNM_MODE                  =00000000                       868    (1)      874    (1)
FAB$V_PUT                       =00000000                       874    (1)
FAB$W_GBC                       =00000048                       868    (1)      874    (1)
FALSE                           =00000000       760    (1)
FAOBUF                           00000BC3-R     1036   (1)     1033    (1)
FAOLEN                           00000BCF-R     1037   (1)     1447    (5)
FCB$SIZE                        =00000030       785    (1)      796    (1)
GET_TIME                         00000C99-R     2210   (6)     2014    (6)     2143    (6)
GLBL_SECT_AREA                   00000BBB-R     1034   (1)    *-1448   (5)    *-1449   (5)     1455   (5)
GLBL_SECT_NAME                   00000BB3-R     1031   (1)     1447    (5)     1455    (5)
GT_MENU_ERROR                    00000000-XR                     1745    (6)
HDW_CLOCK_SET                    00000000-XR                     1277    (5)
HP$B_FLAGS                       0000000A                        2448    (6)     2450    (6)
HP$Q_DEVICE                      00000000                        2451    (6)
HP$V_ALLOC                      =00000000                       *-2447   (6)
HP$V_WASALL                     =00000001                       *-2449   (6)
IDB$SIZE                        =00000020       780    (1)      791    (1)
IMAGE_HEADER                     00000208-R     902    (1)
IMAGE_HEADER_END                 00000408-R     903    (1)
INI$BRK                          000002C2-R     1315   (5)    *-1212   (5)    *-1321   (5)
INI$LOAD_DEVICE                  00000000-XR                     1294    (5)
INI$PTABLE                       00000000-XR                     1290    (5)    *-1526   (5)
INIT_COMMON                      000009A4-R     1903   (6)    *-1887   (6)
INIT_CONTEXT                     0000088A-R     1834   (6)     1292    (5)    *-1788   (6)    *-2360   (6)
INIT_USERMODE                    00000997-R     1898   (6)    *-1838   (6)
IO$M_CTRLCAST                    00000000-XR                    *-1431   (5)    *-1966   (6)
IO$M_NOW                         00000000-XR                    *-1963   (6)
IO$_READVBLK                     00000000-XR                    *-1963   (6)
IO$_SETMODE                      00000000-XR                    *-1431   (5)    *-1966   (6)
IOC$K_DIAGPID                   =00660000       732    (1)    *-1842   (6)
IOSB                             00000BAB-R     1030   (1)     1442    (5)
IRP$SIZE                        =00000050       781    (1)      792    (1)
ISTKPTR                          00000000-XR                    *-1109   (5)    *-1164   (5)    *-1859   (6)
ITEMS                            00000B9B-R     1025   (1)     1442    (5)
JIB$L_ARB                       =00000000       738    (1)    *-1853   (6)
JPI$_CPUTIM                     =00000407                       915    (1)
JPI$_PID                        =00000319                      1026    (1)
JPICPUTIM                        00000425-R     919    (1)    *-2217   (6)      916    (1)
JPICPUTIMS                       00000429-R     920    (1)      917    (1)
KB_CHECK                         00000A88-R     2008   (6)
KB_CHECK_APT                     00000AE6-R     2030   (6)
KB_CHECK_X                       00000C98-R     2171   (6)    *-2035   (6)    *-2046   (6)    *-2051   (6)
KB_POLL                          00000000-XR                     2037    (6)
KSTKPTR                          00000000-XR                     1860    (6)
LF                              =0000000A       773    (1)
MAPMEM                           00000000-XR                     1254    (5)
MASK                            =0000000F       810    (1)
MEMPOOL$INIT                     00000000-XR                     1524    (5)
MP$GR_BOOTED_PROCESSORS          00000408-R     907    (1)    *-2039   (6)    *-2156   (6)
MP$R_ACTIVE_SERVICES             00000000-XR                     1604    (5)      701    (1)
MP$R_INTERLOCK_END               00000000-XR                     1605    (5)      703    (1)
MP$_PRIMARY_CPU_CHECK            00000000-XR                     2042    (6)      699    (1)      713    (1)
MP$_RESUME_ATTACHED_PROCESSORS   00000000-XR                     2158    (6)      700    (1)      714    (1)
MP_CLEAR_END                     000006CB-R     1610   (5)    *-1602   (5)
NDS$OPEN                         00000000-XR                    *-1464   (5)      698    (1)      711    (1)
```

```
OFF                          =00000000     760    (1)
ON                           =00000001     760    (1)
ONLY$GL_TENUSEC               00000000-XR                 #-1284   (5)
ONLY$GL_UBDELAY               00000000-XR                 #-1285   (5)
ONLY_CPU                      00000000-XR                   1242   (5)
P0PT_BASE                     00000000-XR                 #-1200   (5)
PCB$B_ASTACT                  0000000C                    #-1845   (6)    #-2033   (6)
PCB$L_ARB                     00000084                    #-1852   (6)
PCB$L_ASTQBL                  00000014                    #-1844   (6)
PCB$L_ASTQFL                  00000010                      1843   (6)      1844   (6)
PCB$L_JIB                     00000078                    #-1850   (6)
PCB$L_PID                     00000060                    #-1842   (6)
PCB_BASE                      00000000-XR                 #-1194   (5)
PHD$B_ASTLVL                 =000000CF                    #-1880   (6)
PHD$L_ESP                    =0000007C                    #-1864   (6)
PHD$L_KSP                    =00000078                    #-1862   (6)
PHD$L_P0BR                   =000000C8                    #-1877   (6)
PHD$L_P0LRASTL               =000000CC                    #-1878   (6)
PHD$L_P1BR                   =000000D0                    #-1881   (6)
PHD$L_P1LR                   =000000D4                    #-1882   (6)
PHD$L_PC                     =000000C0                    #-1856   (6)    #-1886   (6)
PHD$L_PSL                    =000000C4                    #-1876   (6)
PHD$L_R0                     =00000088                    #-1883   (6)
PHD$L_R12                    =000000B8                      1872   (6)
PHD$L_SSP                    =00000080                    #-1866   (6)
PHD$L_USP                    =00000084                    #-1868   (6)
PHD_BASE                      00000000-XR                   1855   (6)
PID                           00000B8F-R    1022   (1)      1442   (5)
PIDNO                         00000B93-R    1023   (1)      1027   (1)    #-1447   (5)
PIDNOS                        00000B97-R    1024   (1)      1028   (1)
POLL_TIME                     00000000-XR                   2013   (6)    #-2018   (6)    #-2021   (6)    #-2133   (6)
                                                          #-2135   (6)    #-2136   (6)    #-2138   (6)
PPA$GB_CPU                    00000000-XR                 #-1186   (5)      1187   (5)      706    (1)
PPA$GB_MID                    00000000-XR                 #-1188   (5)      706    (1)
PPA_SPID                     =3FFFE400     726    (1)    #-1185   (5)
PR$_ASTLVL                   =00000013                    #-1879   (6)
PR$_ESP                      =00000001                    #-1863   (6)    #-1864   (6)
PR$_IPL                      =00000012                    #-1108   (5)    #-1162   (5)    #-1884   (6)    #-1889   (6)
PR$_ISP                      =00000004                    #-1859   (6)
PR$_KSP                      =00000000                    #-1861   (6)    #-1862   (6)
PR$_MAPEN                    =00000038                    #-1259   (5)
PR$_P0BR                     =00000008                    #-1877   (6)
PR$_P0LR                     =00000009                    #-1878   (6)
PR$_P1BR                     =0000000A                    #-1881   (6)
PR$_P1LR                     =0000000B                    #-1882   (6)
PR$_PCBB                     =00000010                    #-1194   (5)
PR$_SCBB                     =00000011                    #-1193   (5)
PR$_SID                      =0000003E                    #-1119   (5)    #-1166   (5)
PR$_SID_TYP8NN               =00000006     724    (1)    #-2220   (6)
PR$_SID_TYP8RR               =00000011     723    (1)    #-2222   (6)
PR$_SID_TYP8SS               =00000005                    #-1170   (5)    #-1696   (5)
PR$_SID_TYPRTUV2             =00000010     719    (1)    #-1177   (5)    #-1232   (5)
PR$_SID_TYPUV2               =00000008                    #-1179   (5)    #-1234   (5)
PR$_SSP                      =00000002                    #-1865   (6)    #-1866   (6)
PR$_SYS_TYP800               =00000002     720    (1)
PR$_SYS_TYP900               =00000003     721    (1)    #-1183   (5)
PR$_SYS_TYP_LLL              =0000000A     722    (1)    #-1132   (5)
```

```
PR$_TXCS                        =00000022                       #-2256  (6)
PR$_TXDB                        =00000023                       #-2262  (6)
PR$_USP                         =00000003                       #-1867  (6)     #-1868  (6)
PR780$_TODR                     =0000001B                       #-2232  (6)
PR8SS$_BINID                    =0000005E                       #-2380  (6)
PRGBUF                           000001A7-R   1062   (5)          1352  (5)       1435  (5)
PSL$M_IPL                       =001F0000                       #-1890  (6)
PSL$V_IS                        =0000001A     753   (1)       #-1858  (6)     #-2234  (6)     #-2258  (6)     #-2263  (6)
                                                                #-2265  (6)
PSL$V_TBIT                      =00000004                       #-1891  (6)     #-1892  (6)
PSL_SAVE                         00000993-R   1896   (6)       #-1857  (6)       1891  (6)
QA$MAIN                          00000000-XR                      1835  (6)
QIO$INITIALIZE                   00000000-XR                      1256  (5)
QIO$MINCORE                     =00005E3C     804   (1)
RAB$B_RAC                       =0000001E                        871  (1)       877  (1)
RAB$C_BID                       =00000001                        871  (1)       877  (1)
RAB$C_BLN                       =00000044                        871  (1)       877  (1)
RAB$C_SEQ                       =00000000                        871  (1)       877  (1)
RAB$L_CTX                       =00000018                        871  (1)       877  (1)
RAB$L_ROP                       =00000004                        871  (1)       877  (1)
RAB$V_CCO                       =0000001F                        877  (1)
RAB$V_CVT                       =0000001A                        871  (1)
RCTRLC                           000009E9-R   1950   (6)          1431  (5)       1966  (6)
RECEIVE_POLL                     00000000-XR                      1465  (5)        698  (1)       711  (1)
RMS$INIT                         00000000-XR                      1304  (5)
RPB$B_DEVTYP                     00000066                        #-1129  (5)
RPB$L_ADPPHY                     0000005C                        #-1123  (5)
RPB$L_BOOTR1                     00000020                        #-1126  (5)
RPB$L_BOOTR2                     00000024                        #-1127  (5)
RPB$L_BOOTR4                     0000002C                        #-1134  (5)
RPB$L_BOOTR5                     00000030                        #-1128  (5)
RPB$L_CSRPHY                     00000054                        #-1124  (5)
RPB$M_MENUTEST                  =00000001     728   (1)       #-1698  (5)
RPB$T_FILE                       00000068                         1135  (5)
RPB$V_AUTOTEST                  =0000000F                       #-1650  (5)     #-2615  (8)
RPB$V_DIAG                      =00000004                       #-1646  (5)     #-2532  (6)     #-2607  (8)
RPB$V_INIBPT                    =00000002                       #-1211  (5)
RPB$V_MENUTEST                  =00000000     727   (1)       #-1648  (5)     #-2540  (6)       728  (1)
RPB$W_UNIT                       00000064                        #-1125  (5)
RX50_DISK_SELECT                 00000885-R   965    (1)       #-1172  (5)     #-1174  (5)
SAVEFP                           000001B4-R   896    (1)       #-1529  (5)     #-1900  (6)
SCB$L_BREAK                      0000002C                        #-1210  (5)
SCB$L_TBIT                       00000028                        #-1209  (5)
SCB_IMAGE                        00000000-XR                     #-1193  (5)       1208  (5)
SCH$GL_CURPCB                    00000831-R   934    (1)
SCRIPT$INIT                      00000000-XR                      1769  (6)
SCRIPT$STOP                      00000000-XR                      2053  (6)
SEC$M_GBL                       =00000001                       #-1455  (5)
SEC$M_PAGFIL                    =00080000                       #-1455  (5)
SEC$M_WRT                       =00000008                       #-1455  (5)
SF$L_SAVE_PC                    =00000010                       #-2026  (6)
SGN$C_IRPCNT                    =00000040     787   (1)          792  (1)
SGN$C_NPAGEDYN                  =0000263C     796   (1)          804  (1)
SIZ...                          =00000001     761   (1)          755  (1)       761  (1)
SIZE_TERM                        00000000-XR                      1299  (5)
SS$_NORMAL                      =00000001     730   (1)       #-2674  (8)
SSPROT                           0000017A-R   1050   (1)         1355  (5)       1356  (5)
```

```
SSTKPTR                    00000000-XR                  #-1865   (6)
START_RUN_PC               00000411-R   912    (1)     #-2117   (6)     #-2125   (6)
SUPBUF                     0000019F-R   1059   (3)       1351   (5)       1434   (5)
SWI$GL_FQBL                00000839-R   946    (1)
SWI$GL_FQFL                00000835-R   943    (1)
SYI$_SID                  =00001001                    #-1513   (5)
SYS$ASSIGN                 00000000-XR                   1564   (5)
SYS$BINTIM                 00000000-XR                   1275   (5)
SYS$CLREF                  00000000-XR                   1772   (6)
SYS$CONNECT                00000000-XR                   1400   (5)       1404   (5)
SYS$CREATE                 00000000-XR                   1402   (5)
SYS$CRELOG                 00000000-XR                   1495   (5)       2463   (6)
SYS$CRETVA                =800000C8     816    (1)       1351   (5)       1352   (5)       1356   (5)       1434   (5)
                                                         1435   (5)
SYS$CRMPSC                 00000000-XR                   1455   (5)
SYS$DALLOC                 00000000-XR                   2451   (6)
SYS$DCLEXH                 00000000-XR                   1357   (5)
SYS$DELTVA                =80000110     818    (1)       1355   (5)
SYS$DISK                   00000000-XR                   1481   (5)       1493   (5)       2461   (6)
SYS$DXLEXH                =800000F0     817    (1)
SYS$EXIT                   00000000-XR                   2406   (6)
SYS$FAO                    00000000-XR                   1447   (5)
SYS$GETCHN                 00000000-XR                   1393   (5)       1425   (5)       1955   (6)
SYS$GETJPI                 00000000-XR                   1442   (5)
SYS$GETJPIW                00000000-XR                   2212   (6)
SYS$GETSYI                 00000000-XR                   1516   (5)
SYS$INPUT                  0000018A-R   1053   (2)       1383   (5)       868    (1)
SYS$OPEN                   00000000-XR                   1398   (5)
SYS$OUTPUT                 00000194-R   1056   (3)       1410   (5)       874    (1)
SYS$QIOW                   00000000-XR                   1431   (5)       1963   (6)       1966   (6)
SYS$SETAST                 00000000-XR                   1907   (6)
SYS$SETDDIR                00000000-XR                   1502   (5)       2470   (6)
SYS$SETEXV                =80000208     819    (1)       1353   (5)
SYS$SETPRT                =80000230     820    (1)
SYS$TRNLOG                 00000000-XR                   1482   (5)       1552   (5)
SYSVEC                     00000172-R   1048   (1)       1050   (1)
SYS_TYPE                  =20040004     725    (1)     #-1181   (5)
TICK_NS                    00000000-XR                 #-2276   (6)
TM_SIZ                    =00001000     801    (1)       804    (1)
TODR_BIAS                  00000000-XR                 #-2273   (6)
TODR_BYTE_CNT              00000000-XR                 #-2254   (6)     #-2266   (6)       707    (1)
TODR_STORE                 00000000-XR                 #-2271   (6)       707    (1)
TRAN_ASSIGN                00000637-R   1540   (5)     #-1384   (5)     #-1411   (5)
TRUE                      =00000001     760    (1)
TXCS$V_RDY                 00000000-XR                 #-2259   (6)       707    (1)
T_BOTH_CRD_BITS_SET        0000003E-R   834    (1)       1694   (5)
T_WRONGCPU                 0000000E-R   831    (1)       1238   (5)
UCB$SIZE                  =00000098     777    (1)       788    (1)
ULTRIX_END                 00000058-R   859    (1)       855    (1)
ULTRIX_FILE                0000004D-R   854    (1)       1140   (5)
ULTRIX_PARTITION           00000051-R   857    (1)     #-1139   (5)
USTKPTR                    00000000-XR                 #-1867   (6)
VCB$SIZE                  =00000050     783    (1)       794    (1)
V_ENV                     =00000001     756    (1)
V_LVL                     =00000000     756    (1)
WCB$SIZE                  =00000030     784    (1)       795    (1)
XDELBPT                    00000000-XR                   1210   (5)     #-1319   (5)       698    (1)
```

```
XDELTBIT                    00000000-XR                 #-1206   (5)      698    (1)
XF_SIZ                     =00000600     802    (1)      804    (1)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO                    SIZE        DEFINITION      REFERENCES...
-----                    ----        ----------      -------------
$$R_TABINIT              1           868    (1)      868    (1)      871    (1)      874    (1)      877    (1)
$$R_VBFSET               1           868    (1)      868    (1)      871    (1)      874    (1)      877    (1)
$ARBDEF                  1           733    (1)      733    (1)
$ASNPUSH                 1           1455   (5)      1455   (5)
$BINTIM_S                1           1275   (5)      1275   (5)
$CCBDEF                  1           734    (1)      734    (1)
$CLREF_S                 1           1772   (6)      1772   (6)
$CONNECT                 1           1400   (5)      1400   (5)      1404   (5)
$CRD_LITERALS            5           760    (1)      760    (1)
$CREATE                  1           1402   (5)      1402   (5)
$CRETVA                  1           1050   (1)      1050   (1)
$CRETVADEF               1           1050   (1)      1050   (1)
$CRETVA_G                1           1356   (5)      1356   (5)
$CRETVA_S                1           1351   (5)      1351   (5)      1352   (5)      1434   (5)      1435   (5)
$CRMPSC_S                1           1451   (5)      1451   (5)
$D1_PRINT_S              1           1238   (5)      1238   (5)      1694   (5)      1745   (6)      1757   (6)
                                                     1762   (6)      1767   (6)      1784   (6)
$DALLOC_S                1           2451   (6)      2451   (6)
$DCLEXH_S                1           1357   (5)      1357   (5)
$DEF                     1           761    (1)
$DEFINI                  1           733    (1)      733    (1)      734    (1)      735    (1)      736    (1)
                                                     737    (1)      739    (1)      740    (1)      741    (1)
                                                     742    (1)      743    (1)      744    (1)      745    (1)
                                                     746    (1)      747    (1)      748    (1)      749    (1)
                                                     750    (1)      751    (1)      754    (1)      758    (1)
                                                     868    (1)      871    (1)
$DELTVA_G                1           1355   (5)      1355   (5)
$DEVDEF                  1           735    (1)      735    (1)
$DIBDEF                  2           736    (1)      736    (1)
$DS_CNTRLC_S             1           1220   (5)      1220   (5)      1775   (6)
$DS_DSADEF               5           754    (1)      754    (1)
$DS_DSDEF                3           731    (1)      731    (1)
$DS_HPODEF               2           758    (1)      758    (1)
$DS_INITSCB_G            1           1247   (5)      1247   (5)
$DS_LOADPCS_S            1           1310   (5)      1310   (5)
$DS_SCBDEF               3           751    (1)      751    (1)
$DS_TYPEDEF              4           759    (1)      759    (1)
$EQU                     1           761    (1)      731    (1)      753    (1)      757    (1)      759    (1)
                                                     760    (1)
$EQULS1                  1           760    (1)      731    (1)      753    (1)      757    (1)      759    (1)
                                                     760    (1)
$EQULST                  1           731    (1)      731    (1)      753    (1)      757    (1)      759    (1)
                                                     760    (1)
$EXIT_S                  1           2406   (6)      2406   (6)
$FAB                     4           868    (1)      868    (1)      874    (1)
$FABDEF                  1           868    (1)      868    (1)      874    (1)
$FAO_S                   2           1444   (5)      1444   (5)
$GBLINI                  2           731    (1)      731    (1)      753    (1)      755    (1)      757    (1)
                                                     759    (1)      760    (1)      761    (1)
$GETCHN_S                1           1393   (5)      1393   (5)      1425   (5)      1955   (6)
```

$GETJPIW_S	1	2212	(6)	2212	(6)							
$GETJPI_S	1	1439	(5)	1439	(5)							
$GETSYI_S	1	1516	(5)	1516	(5)							
$JIBDEF	3	737	(1)	737	(1)							
$JPIDEF	6	739	(1)	739	(1)							
$OFFDEF	1	1050	(1)	1050	(1)							
$OPEN	1	1398	(5)	1398	(5)							
$PCBDEF	4	740	(1)	740	(1)							
$PHDDEF	6	741	(1)	741	(1)							
$PR780DEF	1	742	(1)	742	(1)							
$PR8SSDEF	5	743	(1)	743	(1)							
$PRDEF	5	744	(1)	744	(1)							
$PRINT	2	1694	(5)	1694	(5)	1743	(6)	1755	(6)	1760	(6)	
				1764	(6)							
$PRINTI_S	1	1237	(5)	1237	(5)	1784	(6)					
$PSLDEF	2	745	(1)	745	(1)							
$PUSHADR	1	1220	(5)	1220	(5)	1275	(5)	1351	(5)	1352	(5)	
				1353	(5)	1357	(5)	1393	(5)	1425	(5)	
				1431	(5)	1434	(5)	1435	(5)	1442	(5)	
				1447	(5)	1455	(5)	1516	(5)	1775	(6)	
				1955	(6)	1958	(6)	1963	(6)	1966	(6)	
				2212	(6)	2215	(6)	2451	(6)			
$PUSHTWO	1	1431	(5)	1431	(5)	1455	(5)	1963	(6)	1966	(6)	
$QIOPUSH	1	1353	(5)	1353	(5)	1431	(5)	1442	(5)	1516	(5)	
				1963	(6)	1966	(6)	2212	(6)			
$QIOW_S	1	1431	(5)	1431	(5)	1963	(6)	1966	(6)			
$RAB	2	871	(1)	871	(1)	877	(1)					
$RABDEF	1	871	(1)	871	(1)	877	(1)					
$RMSCALL	2	1398	(5)	1398	(5)	1400	(5)	1402	(5)	1404	(5)	
$RPBDEF	5	746	(1)	746	(1)							
$SECDEF	3	747	(1)	747	(1)							
$SETAST_S	1	1907	(6)	1907	(6)							
$SETEXV_S	1	1353	(5)	1353	(5)							
$SFDEF	2	748	(1)	748	(1)							
$SHRDEF	6	749	(1)	749	(1)							
$SYIDEF	18	750	(1)	750	(1)							
$VIELD	1			755	(1)	761	(1)					
$VIELD1	1	761	(1)	755	(1)	761	(1)					
APDEF	2	761	(1)	761	(1)							
BR_IF_NOT_AP	1	2380	(6)	2380	(6)							
BR_IF_NOT_APT	1	1783	(6)	1783	(6)							
BR_IF_NOT_AP_RUNNING	1	2380	(6)	2380	(6)							
BR_IF_NOT_LODFLG	1	2356	(6)	2356	(6)							
BR_IF_NOT_MP	1	1602	(5)	1602	(5)							
BR_IF_NOT_USER	1	2211	(6)	2211	(6)							
BR_IF_USER	1	2378	(6)	2378	(6)							
CLIDEF	4	752	(1)	752	(1)							
CMK	1	1858	(6)	1858	(6)	2234	(6)	2258	(6)	2263	(6)	
				2265	(6)							
CMKDEF	1	753	(1)	753	(1)							
DSFDEF	3	755	(1)	755	(1)							
DSQA	2	757	(1)	757	(1)							
ENVDEF	1	756	(1)	756	(1)							
ERRSUP_S	1	1958	(6)	1958	(6)	2215	(6)					
MODNAM	1	1046	(1)	1046	(1)							
QA_MAIN	1	1835	(6)	1835	(6)							
SET_NI	1	1462	(5)	1462	(5)							

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ACBL	00F1	1381 (5)
ADDB2	0080	1147 (5) 1649 (5) 1651 (5)
ADDB3	0081	1139 (5)
ADDL	00C0	1416 (5) 1558 (5) 1567 (5) 1956 (6) 2471 (6)
ADDL2	00C0	1372 (5) 1376 (5) 1377 (5) 1380 (5) 1517 (5) 2136 (6) 2278 (6)
ADWC	00D8	2138 (6) 2279 (6)
AOBLEQ	00F3	1773 (6)
AOBLSS	00F2	1607 (5)
ASHL	0078	1111 (5) 1131 (5) 2329 (6)
BBC	00E1	1211 (5) 1397 (5) 1436 (5) 1457 (5) 1646 (5) 1648 (5) 1650 (5)
		1783 (6) 1836 (6) 1891 (6) 1959 (6) 2048 (6) 2115 (6) 2211 (6)
		2240 (6) 2259 (6) 2356 (6) 2380 (6) 2438 (6) 2447 (6) 2449 (6)
		2521 (6) 2532 (6) 2540 (6) 2596 (8) 2607 (8) 2615 (8)
BBCS	00E3	1406 (5)
BBS	00E0	1858 (6) 2009 (6) 2031 (6) 2044 (6) 2226 (6) 2234 (6) 2258 (6)
		2263 (6) 2265 (6) 2378 (6) 2392 (6)
BBSC	00E4	2397 (6)
BBSS	00E2	1462 (5) 1463 (5)
BEQL	0013	1121 (5) 1145 (5) 1149 (5) 1173 (5) 1178 (5) 1207 (5) 1231 (5)
		1235 (5) 1320 (5) 1602 (5) 2034 (6) 2040 (6) 2081 (6) 2092 (6)
		2102 (6) 2126 (6) 2157 (6) 2221 (6) 2223 (6) 2268 (6) 2446 (6)
BGTRU	001A	2020 (6) 2022 (6)
BICL	00CA	2274 (6)
BICL2	00CA	1626 (5) 1698 (5) 1737 (6) 2024 (6) 2085 (6) 2095 (6) 2106 (6)
		2112 (6)
BISB	0088	2330 (6)
BISL	00C8	1892 (6)
BISL2	00C8	1115 (5) 1521 (5) 1700 (5) 1702 (5) 1704 (5) 1708 (5) 1967 (6)
		2025 (6) 2150 (6) 2152 (6) 2161 (6) 2525 (6) 2549 (6) 2550 (6)
BISW2	00A8	2390 (6)
BITL	00D3	1749 (6)
BLBC	00E9	1394 (5) 1399 (5) 1401 (5) 1403 (5) 1405 (5) 1426 (5) 1553 (5)
		2088 (6) 2098 (6) 2109 (6)
BLBS	00E8	1385 (5) 1413 (5) 1518 (5) 1730 (6) 1957 (6) 2214 (6)
BLSS	0019	1201 (5) 1962 (6)
BLSSU	001F	2019 (6) 2325 (6)
BNEQ	0012	1118 (5) 1133 (5) 1137 (5) 1171 (5) 1180 (5) 1184 (5) 1233 (5)
		1556 (5) 1602 (5) 1697 (5) 1707 (5) 1754 (6) 1875 (6) 1918 (6)
		2130 (6) 2380 (6)
BPT	0003	1316 (5)
BRB	0011	1142 (5) 1417 (5) 1688 (5) 1701 (5) 1703 (5) 1709 (5) 1796 (6)
		1858 (6) 1887 (6) 1964 (6) 2234 (6) 2258 (6) 2263 (6) 2265 (6)
		2269 (6) 2600 (8)
BRW	0031	1122 (5) 1313 (5) 1414 (5) 1530 (5) 1699 (5) 1838 (6) 1910 (6)
		2010 (6) 2035 (6) 2046 (6) 2051 (6) 2127 (6) 2131 (6) 2216 (6)
		2218 (6) 2230 (6) 2236 (6) 2247 (6) 2400 (6)
BSBB	0010	1321 (5) 1788 (6)
BSBW	0030	1212 (5) 1384 (5) 1411 (5) 1526 (5) 2360 (6)
CALLG	00FA	1247 (5) 1256 (5) 1355 (5) 1356 (5) 1792 (6) 2087 (6) 2097 (6)
		2108 (6) 2123 (6)

```
CALLS           00FB    1220 (5)    1238 (5)    1275 (5)    1299 (5)    1310 (5)    1351 (5)    1352 (5)
                        1353 (5)    1357 (5)    1393 (5)    1398 (5)    1400 (5)    1402 (5)    1404 (5)
                        1425 (5)    1431 (5)    1434 (5)    1435 (5)    1442 (5)    1447 (5)    1455 (5)
                        1465 (5)    1482 (5)    1495 (5)    1502 (5)    1516 (5)    1552 (5)    1564 (5)
                        1619 (5)    1694 (5)    1716 (6)    1729 (6)    1745 (6)    1757 (6)    1762 (6)
                        1767 (6)    1772 (6)    1775 (6)    1784 (6)    1835 (6)    1907 (6)    1909 (6)
                        1955 (6)    1958 (6)    1963 (6)    1966 (6)    2014 (6)    2042 (6)    2143 (6)
                        2158 (6)    2212 (6)    2215 (6)    2372 (6)    2406 (6)    2451 (6)    2463 (6)
                        2470 (6)
CASEB           008F    1669 (5)
CHMK            00BC    1858 (6)    2234 (6)    2258 (6)    2263 (6)    2265 (6)
CLRB            0094    1260 (5)    1606 (5)    1845 (6)    1905 (6)    2394 (6)    2395 (6)
CLRL            00D4    1165 (5)    1222 (5)    1223 (5)    1224 (5)    1350 (5)    1477 (5)    1491 (5)
                        1547 (5)    1560 (5)    1645 (5)    1770 (6)    1858 (6)    1870 (6)    2084 (6)
                        2094 (6)    2105 (6)    2234 (6)    2254 (6)    2258 (6)    2260 (6)    2263 (6)
                        2265 (6)    2323 (6)    2459 (6)    2519 (6)    2594 (8)
CLRQ            007C    1113 (5)    1199 (5)    1257 (5)    1258 (5)    1431 (5)    1442 (5)    1455 (5)
                        1511 (5)    1516 (5)    1548 (5)    1561 (5)    1963 (6)    1966 (6)    2212 (6)
                        2468 (6)
CLRW            00B4    1489 (5)
CMPB            0091    1132 (5)    1144 (5)    1170 (5)    1172 (5)    1177 (5)    1179 (5)    1183 (5)
                        1230 (5)    1232 (5)    1234 (5)    1555 (5)    1696 (5)    2220 (6)    2222 (6)
                        2324 (6)
CMPC3           0029    1705 (5)
CMPL            00D1    1120 (5)    1200 (5)    1602 (5)    2017 (6)    2021 (6)    2125 (6)    2266 (6)
CMPZV           00ED    2380 (6)
CVTBL           0098    1694 (5)    1745 (6)    1757 (6)    1762 (6)    1767 (6)
CVTLB           00F6    1186 (5)    1188 (5)
DECW            00B7    1961 (6)
EMUL            007A    2276 (6)
EXTV            00EE    1138 (5)
EXTZV           00EF    1187 (5)    1778 (6)    2327 (6)
HALT            0000    2399 (6)
INCB            0096    1647 (5)
INCL            00D6    1151 (5)    2029 (6)    2624 (8)
INSV            00F0    2328 (6)
JMP             0017    1386 (5)    1695 (5)    2027 (6)
JSB             0016    1155 (5)    1242 (5)    1254 (5)    1277 (5)    1281 (5)    1282 (5)    1290 (5)
                        1292 (5)    1294 (5)    1304 (5)    1524 (5)    1747 (6)    1769 (6)    1858 (6)
                        2037 (6)    2053 (6)    2234 (6)    2258 (6)    2263 (6)    2265 (6)    2357 (6)
                        2439 (6)
LOCC            003A    1148 (5)
MFPR            00DB    1119 (5)    1166 (5)    1862 (6)    1864 (6)    1866 (6)    1868 (6)    1877 (6)
                        1878 (6)    1881 (6)    1882 (6)    2380 (6)
MOVAB           009E    1135 (5)    1140 (5)    1210 (5)    1383 (5)    1410 (5)    1473 (5)    1515 (5)
                        1790 (6)    1841 (6)    1849 (6)    1851 (6)    1914 (6)    2326 (6)    2456 (6)
                        2465 (6)    2466 (6)
MOVAL           00DE    1112 (5)    1153 (5)    1198 (5)    1208 (5)    1209 (5)    1273 (5)    1354 (5)
                        1367 (5)    1604 (5)    1605 (5)    1843 (6)    1844 (6)    1855 (6)    1860 (6)
                        1872 (6)    2114 (6)    2151 (6)    2160 (6)    2388 (6)    2441 (6)    2442 (6)
MOVAQ           007E    1392 (5)    1424 (5)    1430 (5)
MOVB            0090    1175 (5)    1182 (5)    1780 (6)    1880 (6)
MOVC3           0028    1358 (5)
MOVL            00D0    1109 (5)    1110 (5)    1123 (5)    1124 (5)    1126 (5)    1127 (5)    1128 (5)
                        1134 (5)    1143 (5)    1152 (5)    1164 (5)    1174 (5)    1181 (5)    1185 (5)
                        1190 (5)    1192 (5)    1206 (5)    1270 (5)    1284 (5)    1285 (5)    1366 (5)
                        1370 (5)    1374 (5)    1379 (5)    1396 (5)    1485 (5)    1486 (5)    1487 (5)
```

		1488	(5)	1519	(5)	1528	(5)	1529	(5)	1758	(6)	1842	(6)	1850	(6)
		1852	(6)	1853	(6)	1873	(6)	1883	(6)	1886	(6)	1890	(6)	1900	(6)
		1901	(6)	1916	(6)	1920	(6)	1954	(6)	2026	(6)	2082	(6)	2093	(6)
		2103	(6)	2117	(6)	2119	(6)	2165	(6)	2167	(6)	2217	(6)	2232	(6)
		2233	(6)	2235	(6)	2256	(6)	2257	(6)	2261	(6)	2262	(6)	2271	(6)
		2322	(6)	2386	(6)	2443	(6)	2445	(6)	2674	(8)				
MOVPSL	00DC	1789	(6)	1857	(6)	1858	(6)	1876	(6)	2120	(6)	2234	(6)	2258	(6)
		2263	(6)	2265	(6)										
MOVQ	007D	1154	(5)	1276	(5)	1428	(5)	1432	(5)	2129	(6)	2146	(6)	2166	(6)
		2229	(6)	2246	(6)	2280	(6)	2673	(8)						
MOVW	00B0	1369	(5)	1373	(5)	1378	(5)	1388	(5)	1419	(5)	1484	(5)	1513	(5)
		1514	(5)												
MOVZBL	009A	1129	(5)	1136	(5)	1141	(5)	1146	(5)	1229	(5)	1546	(5)	1694	(5)
		1745	(6)	1757	(6)	1762	(6)	1767	(6)						
MOVZWL	003C	1125	(5)	1272	(5)	1393	(5)	1425	(5)	1431	(5)	1448	(5)	1449	(5)
		1455	(5)	1955	(6)	1963	(6)	1966	(6)	2458	(6)	2467	(6)		
MTPR	00DA	1108	(5)	1162	(5)	1193	(5)	1194	(5)	1259	(5)	1859	(6)	1861	(6)
		1863	(6)	1865	(6)	1867	(6)	1879	(6)	1884	(6)	1889	(6)		
POPL	8ED0	1856	(6)	1899	(6)										
POPR	00BA	1395	(5)	1427	(5)	1565	(5)	1608	(5)	1793	(6)	1908	(6)	1919	(6)
		2015	(6)	2043	(6)	2118	(6)	2144	(6)	2168	(6)				
PUSHAB	009F	1238	(5)	1474	(5)	1497	(5)	1512	(5)	1542	(5)	1545	(5)	1694	(5)
		1745	(6)	1757	(6)	1762	(6)	1767	(6)	1784	(6)	1952	(6)	2121	(6)
		2457	(6)												
PUSHAL	00DF	1353	(5)	1357	(5)	1398	(5)	1400	(5)	1402	(5)	1404	(5)	1431	(5)
		1442	(5)	1516	(5)	1550	(5)	1562	(5)	1958	(6)	1963	(6)	1966	(6)
		2013	(6)	2142	(6)	2212	(6)	2215	(6)						
PUSHAQ	007F	1275	(5)	1351	(5)	1352	(5)	1390	(5)	1393	(5)	1422	(5)	1425	(5)
		1431	(5)	1434	(5)	1435	(5)	1442	(5)	1447	(5)	1455	(5)	1479	(5)
		1481	(5)	1492	(5)	1493	(5)	1499	(5)	1501	(5)	1549	(5)	1551	(5)
		1563	(5)	1955	(6)	2451	(6)	2460	(6)	2461	(6)	2469	(6)		
PUSHAW	003F	1393	(5)	1425	(5)	1447	(5)	1480	(5)	1500	(5)	1955	(6)		
PUSHL	00DD	1220	(5)	1238	(5)	1351	(5)	1352	(5)	1353	(5)	1391	(5)	1393	(5)
		1423	(5)	1425	(5)	1431	(5)	1434	(5)	1435	(5)	1442	(5)	1447	(5)
		1455	(5)	1464	(5)	1475	(5)	1476	(5)	1478	(5)	1494	(5)	1498	(5)
		1516	(5)	1543	(5)	1725	(6)	1726	(6)	1727	(6)	1728	(6)	1767	(6)
		1772	(6)	1775	(6)	1835	(6)	1907	(6)	1913	(6)	1953	(6)	1955	(6)
		1958	(6)	1963	(6)	1966	(6)	2212	(6)	2215	(6)	2370	(6)	2406	(6)
		2451	(6)	2462	(6)										
PUSHR	00BB	1603	(5)	1791	(6)	1906	(6)	2012	(6)	2041	(6)	2113	(6)	2122	(6)
		2141	(6)												
REI	0002	1322	(5)	1794	(6)	1894	(6)	2169	(6)						
RET	0004	1533	(5)	1969	(6)	2281	(6)	2331	(6)	2473	(6)	2675	(8)		
ROTL	009C	2275	(6)												
RSB	0005	1317	(5)	1568	(5)	1922	(6)	2089	(6)	2099	(6)	2110	(6)	2172	(6)
		2407	(6)	2553	(6)	2626	(8)								
SBWC	00D9	2134	(6)	2149	(6)										
SOBGTR	00F5	1114	(5)	2452	(6)										
SUBL	00C2	1541	(5)	1557	(5)	1951	(6)								
SUBL2	00C2	2132	(6)	2147	(6)	2273	(6)								
SUBL3	00C3	1150	(5)	1874	(6)	1917	(6)								
TSTB	0095	2033	(6)												
TSTL	00D5	1117	(5)	1319	(5)	1389	(5)	1421	(5)	1706	(5)	2039	(6)	2080	(6)
		2091	(6)	2101	(6)	2156	(6)								

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	1027	(1)	1028	(1)	1033	(1)	1050	(1)	863	(1)	868	(1)	871	(1)	874	(1)
	877	(1)	891	(1)	893	(1)	916	(1)	917	(1)	932	(1)	936	(1)		
.ALIGN	866	(1)														
.ASCIC	1046	(1)	1055	(3)	1057	(3)	832	(1)	835	(1)	837	(1)				
.ASCID	1040	(1)														
.ASCII	828	(1)	830	(1)	849	(1)	851	(1)	852	(1)	856	(1)	858	(1)		
.BLKB	1011	(1)	1012	(1)	1014	(1)	1015	(1)	1016	(1)	1017	(1)	1036	(1)	865	(1)
	868	(1)	871	(1)	874	(1)	877	(1)	902	(1)	929	(1)				
.BLKL	1000	(1)	1001	(1)	1002	(1)	1003	(1)	1004	(1)	1005	(1)	1006	(1)	1007	(1)
	1009	(1)	1010	(1)	1023	(1)	1024	(1)	1035	(1)	752	(1)	896	(1)	954	(1)
	955	(1)	956	(1)	957	(1)	958	(1)	959	(1)	960	(1)	963	(1)	966	(1)
	967	(1)	969	(1)	970	(1)	971	(1)	981	(1)	984	(1)	986	(1)	988	(1)
	990	(1)	992	(1)	994	(1)	996	(1)	997	(1)	998	(1)	999	(1)		
.BLKQ	1018	(1)	1030	(1)												
.BLKW	1037	(1)	1038	(1)	898	(1)	900	(1)	982	(1)	983	(1)				
.BYTE	848	(1)	855	(1)	868	(1)	871	(1)	874	(1)	877	(1)	909	(1)		
.END	2677	(8)														
.ENDC	1220	(5)	1238	(5)	1275	(5)	1351	(5)	1352	(5)	1353	(5)	1357	(5)	1393	(5)
	1398	(5)	1400	(5)	1402	(5)	1404	(5)	1425	(5)	1431	(5)	1434	(5)	1435	(5)
	1442	(5)	1447	(5)	1455	(5)	1516	(5)	1694	(5)	1745	(6)	1757	(6)	1762	(6)
	1767	(6)	1775	(6)	1784	(6)	1835	(6)	1955	(6)	1958	(6)	1963	(6)	1966	(6)
	2212	(6)	2215	(6)	2451	(6)	731	(1)	753	(1)	755	(1)	757	(1)	759	(1)
	760	(1)	761	(1)	868	(1)	871	(1)	874	(1)	877	(1)				
.ENDM	1046	(1)	1050	(1)	1220	(5)	1237	(5)	1238	(5)	1247	(5)	1275	(5)	1310	(5)
	1351	(5)	1353	(5)	1355	(5)	1356	(5)	1357	(5)	1393	(5)	1398	(5)	1400	(5)
	1402	(5)	1431	(5)	1439	(5)	1444	(5)	1447	(5)	1451	(5)	1455	(5)	1462	(5)
	1516	(5)	1602	(5)	1694	(5)	1772	(6)	1783	(6)	1835	(6)	1858	(6)	1907	(6)
	1958	(6)	2211	(6)	2212	(6)	2356	(6)	2378	(6)	2380	(6)	2406	(6)	2451	(6)
	731	(1)	733	(1)	734	(1)	735	(1)	736	(1)	737	(1)	739	(1)	740	(1)
	741	(1)	742	(1)	743	(1)	744	(1)	745	(1)	746	(1)	747	(1)	748	(1)
	749	(1)	750	(1)	751	(1)	752	(1)	753	(1)	754	(1)	755	(1)	756	(1)
	757	(1)	758	(1)	759	(1)	760	(1)	761	(1)	868	(1)	871	(1)		
.ENDR	1238	(5)	1694	(5)	1745	(6)	1757	(6)	1762	(6)	1767	(6)	1784	(6)	731	(1)
	753	(1)	755	(1)	757	(1)	759	(1)	760	(1)	761	(1)	868	(1)	871	(1)
	874	(1)	877	(1)												
.ENTRY	1349	(5)	1950	(6)	2210	(6)	2320	(6)	2671	(8)						
.EXTRN	711	(1)	713	(1)	714	(1)										
.GLOBL	1275	(5)	1351	(5)	1352	(5)	1353	(5)	1355	(5)	1356	(5)	1357	(5)	1393	(5)
	1398	(5)	1400	(5)	1402	(5)	1404	(5)	1425	(5)	1431	(5)	1434	(5)	1435	(5)
	1442	(5)	1455	(5)	1516	(5)	1772	(6)	1907	(6)	1955	(6)	1963	(6)	1966	(6)
	2212	(6)	2406	(6)	2451	(6)										
.IDENT	45	(1)														
.IF	1220	(5)	1238	(5)	1275	(5)	1351	(5)	1352	(5)	1353	(5)	1357	(5)	1393	(5)
	1398	(5)	1400	(5)	1402	(5)	1404	(5)	1425	(5)	1431	(5)	1434	(5)	1435	(5)
	1442	(5)	1447	(5)	1455	(5)	1516	(5)	1694	(5)	1745	(6)	1757	(6)	1762	(6)
	1767	(6)	1775	(6)	1784	(6)	1835	(6)	1955	(6)	1958	(6)	1963	(6)	1966	(6)
	2212	(6)	2215	(6)	2451	(6)	731	(1)	753	(1)	755	(1)	757	(1)	759	(1)
	760	(1)	761	(1)	868	(1)	871	(1)	874	(1)	877	(1)				
.IFF	1220	(5)	1275	(5)	1351	(5)	1352	(5)	1353	(5)	1357	(5)	1393	(5)	1398	(5)
	1400	(5)	1402	(5)	1404	(5)	1425	(5)	1431	(5)	1434	(5)	1435	(5)	1442	(5)

```
                1447  (5)     1455  (5)     1516  (5)     1775  (6)     1955  (6)     1958  (6)     1963  (6)     1966  (6)
                2212  (6)     2215  (6)     2451  (6)     731   (1)     753   (1)     755   (1)     757   (1)     759   (1)
                760   (1)     761   (1)     868   (1)     871   (1)     874   (1)     877   (1)
.IF_FALSE       1835  (6)
.IIF            1958  (6)     2215  (6)     731   (1)     753   (1)     755   (1)     757   (1)     759   (1)     760   (1)
                761   (1)     868   (1)     871   (1)     874   (1)     877   (1)
.IRP            1050  (1)     1238  (5)     1447  (5)     1694  (5)     1745  (6)     1757  (6)     1762  (6)     1767  (6)
                1784  (6)     731   (1)     753   (1)     755   (1)     757   (1)     759   (1)     760   (1)     761   (1)
                868   (1)     871   (1)     874   (1)     877   (1)
.LIBRARY        689   (1)     690   (1)     691   (1)     692   (1)
.LIST           1050  (1)     751   (1)     752   (1)     754   (1)
.LONG           1021  (1)     1022  (1)     1029  (1)     1032  (1)     1048  (1)     1059  (3)     1062  (5)     1896  (6)
                881   (1)     884   (1)     887   (1)     890   (1)     892   (1)     894   (1)     907   (1)     910   (1)
                912   (1)     918   (1)     919   (1)     920   (1)     944   (1)     947   (1)     948   (1)     949   (1)
                950   (1)     951   (1)     952   (1)
.MACRO          731   (1)     753   (1)     755   (1)     757   (1)     759   (1)     760   (1)     761   (1)
.MDELETE        731   (1)     753   (1)     761   (1)
.NLIST          1050  (1)     751   (1)     752   (1)     754   (1)
.NOCROSS        733   (1)     734   (1)     735   (1)     736   (1)     737   (1)     739   (1)     740   (1)     741   (1)
                742   (1)     743   (1)     744   (1)     745   (1)     746   (1)     747   (1)     748   (1)     749   (1)
                750   (1)     751   (1)     754   (1)     758   (1)     868   (1)     871   (1)
.NOSHOW         46    (1)
.NTYPE          1398  (5)     1400  (5)     1402  (5)     1404  (5)
.PAGE           1019  (1)     1042  (1)     1064  (5)     1102  (5)     1157  (5)     1323  (5)     1348  (5)     145   (1)
                1534  (5)     1569  (5)     1598  (5)     1785  (6)     1797  (6)     1833  (6)     1897  (6)     190   (1)
                1923  (6)     1970  (6)     2007  (6)     204   (1)     2054  (6)     2173  (6)     218   (1)     2283  (6)
                229   (1)     2333  (6)     2354  (6)     2408  (6)     2474  (6)     250   (1)     2554  (6)     2628  (8)
                269   (1)     298   (1)     342   (1)     67    (1)     684   (1)     763   (1)     821   (1)     844   (1)
                921   (1)     94    (1)     972   (1)
.PSECT          1044  (1)     1066  (5)     1325  (5)     752   (1)     823   (1)     846   (1)
.RESTORE        733   (1)     734   (1)     735   (1)     736   (1)     737   (1)     739   (1)     740   (1)     741   (1)
                742   (1)     743   (1)     744   (1)     745   (1)     746   (1)     747   (1)     748   (1)     749   (1)
                750   (1)     751   (1)     752   (1)     754   (1)     758   (1)     868   (1)     871   (1)
.SAVE           733   (1)     734   (1)     735   (1)     736   (1)     737   (1)     739   (1)     740   (1)     741   (1)
                742   (1)     743   (1)     744   (1)     745   (1)     746   (1)     747   (1)     748   (1)     749   (1)
                750   (1)     751   (1)     752   (1)     754   (1)     758   (1)     868   (1)     871   (1)
.SBTTL          1020  (1)     1065  (5)     1324  (5)     1570  (5)     1798  (6)     1924  (6)     1971  (6)     2174  (6)
                2284  (6)     2334  (6)     2409  (6)     2629  (8)
.SIGNED_WORD    1671  (5)
.SUBTITLE       1043  (1)     2475  (6)     2555  (6)     685   (1)     764   (1)     822   (1)     845   (1)
.TITLE          44    (1)
.WEAK           698   (1)     699   (1)     700   (1)     701   (1)     703   (1)     705   (1)     706   (1)     707   (1)
.WORD           1025  (1)     1026  (1)     1912  (6)     2437  (6)     861   (1)     862   (1)     868   (1)     871   (1)
                874   (1)     877   (1)     914   (1)     915   (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	8	#-1919 (6) #-2229 (6) #-2246 (6) #-2280 (6) #-2322 (6) #-2324 (6) #-2328 (6) #-2673 (8)
FP	8	#-1165 (5) #-1354 (5) #-1529 (5) #-1870 (6) #-1900 (6) #-1901 (6) #-1914 (6) #-2026 (6)
R0	155	#-1110 (5) #-1111 (5) #-1114 (5) #-1119 (5) #-1120 (5) #-1131 (5) #-1132 (5) #-1138 (5) #-1139 (5) #-1141 (5) #-1143 (5) #-1146 (5) #-1147 (5) #-1148 (5) #-1150 (5) #-1151 (5) #-1154 (5) #-1185 (5) #-1186 (5) #-1187 (5) #-1188 (5) #-1198 (5) #-1199 (5) #-1200 (5) #-1206 (5) 1209 (5) #-1229 (5) #-1230 (5) #-1232 (5) #-1238 (5) #-1366 (5) #-1370 (5) #-1372 (5) #-1374 (5) #-1377 (5) #-1379 (5) #-1380 (5) #-1383 (5) #-1385 (5) #-1392 (5) 1393 (5) #-1394 (5) #-1395 (5) #-1399 (5) #-1401 (5) #-1403 (5) #-1405 (5) #-1410 (5) #-1413 (5) #-1424 (5) 1425 (5) #-1426 (5) #-1427 (5) #-1430 (5) 1431 (5) #-1432 (5) #-1515 (5) 1516 (5) #-1518 (5) 1545 (5) #-1546 (5) #-1553 (5) #-1706 (5) #-1730 (6) #-1779 (6) #-1780 (6) #-1841 (6) #-1842 (6) 1843 (6) 1844 (6) #-1845 (6) #-1850 (6) #-1852 (6) #-1855 (6) #-1856 (6) #-1862 (6) #-1864 (6) #-1866 (6) #-1868 (6) 1872 (6) #-1876 (6) #-1877 (6) #-1878 (6) #-1880 (6) #-1881 (6) #-1882 (6) #-1883 (6) #-1886 (6) #-1899 (6) #-1906 (6) #-1908 (6) #-1913 (6) #-1919 (6) #-1920 (6) #-1954 (6) 1955 (6) #-1957 (6) #-2012 (6) #-2015 (6) #-2026 (6) #-2041 (6) #-2043 (6) #-2088 (6) #-2098 (6) #-2109 (6) #-2113 (6) #-2114 (6) 2116 (6) #-2118 (6) #-2141 (6) #-2144 (6) #-2151 (6) #-2153 (6) #-2160 (6) #-2162 (6) #-2165 (6) #-2166 (6) #-2167 (6) #-2214 (6) #-2217 (6) #-2235 (6) #-2271 (6) #-2273 (6) #-2274 (6) #-2275 (6) #-2276 (6) #-2278 (6) #-2280 (6) #-2323 (6) #-2327 (6) #-2329 (6) #-2330 (6) #-2456 (6) 2457 (6) #-2458 (6) #-2465 (6) 2466 (6) #-2467 (6) #-2519 (6) #-2525 (6) #-2549 (6) #-2550 (6) #-2594 (8) #-2624 (8) #-2674 (8)
R1	41	#-1112 (5) #-1113 (5) #-1150 (5) #-1152 (5) #-1208 (5) #-1209 (5) #-1210 (5) #-1367 (5) #-1369 (5) #-1370 (5) #-1373 (5) #-1374 (5) #-1376 (5) #-1378 (5) #-1379 (5) #-1381 (5) #-1388 (5) #-1393 (5) #-1395 (5) #-1419 (5) #-1425 (5) #-1427 (5) #-1565 (5) #-1872 (6) #-1873 (6) #-1874 (6) #-1919 (6) #-2012 (6) #-2015 (6) #-2041 (6) #-2043 (6) #-2141 (6) #-2144 (6) #-2235 (6) 2259 (6) #-2261 (6) #-2279 (6) #-2326 (6) 2327 (6) 2328 (6)
R10	12	#-1135 (5) #-1136 (5) #-1140 (5) #-1141 (5) #-1144 (5) #-1146 (5) 1148 (5) #-1150 (5) #-1152 (5) 1349 (5) #-1919 (6) 2437 (6)
R11	12	#-1123 (5) #-1124 (5) #-1125 (5) #-1126 (5) #-1127 (5) #-1128 (5) #-1129 (5) #-1134 (5) 1135 (5) 1349 (5) #-1919 (6) 2437 (6)
R2	24	#-1110 (5) #-1120 (5) #-1153 (5) #-1154 (5) 1349 (5) #-1395 (5) #-1396 (5) 1397 (5) #-1427 (5) #-1473 (5) 1474 (5) 1480 (5) #-1770 (6) #-1772 (6) #-1773 (6) #-1851 (6) #-1852 (6) #-1853 (6) #-1919 (6) 2437 (6) #-2445 (6) 2448 (6) 2450 (6) 2451 (6)
R3	20	#-1174 (5) 1349 (5) #-1395 (5) #-1427 (5) #-1428 (5) #-1603 (5) #-1604 (5) #-1606 (5) #-1607 (5) #-1608 (5) #-1849 (6) #-1850 (6) #-1853 (6) #-1919 (6) 2210 (6) #-2232 (6) #-2256 (6) #-2262 (6) 2437 (6) #-2442 (6)
R4	17	1349 (5) #-1395 (5) #-1427 (5) #-1603 (5) #-1605 (5) #-1607 (5) #-1608 (5) #-1645 (5) #-1647 (5) #-1649 (5) #-1651 (5) #-1669 (5)

```
                          #-1919   (6)      2437   (6)    #-2443   (6)    #-2445   (6)    #-2452   (6)
R5              10          1349   (5)    #-1919   (6)      2210   (6)    #-2233   (6)    #-2257   (6)    #-2260   (6)
                            2437   (6)    #-2441   (6)    #-2443   (6)    #-2445   (6)
R6               3          1349   (5)    #-1919   (6)      2437   (6)
R7               3          1349   (5)    #-1919   (6)      2437   (6)
R8               3          1349   (5)    #-1919   (6)      2437   (6)
R9               6         #-1136   (5)    #-1143   (5)    #-1148   (5)      1349   (5)    #-1919   (6)      2437   (6)
SP             132         #-1109   (5)    #-1164   (5)      1247   (5)      1256   (5)    #-1389   (5)      1390   (5)
                            1392   (5)    #-1393   (5)    #-1416   (5)    #-1421   (5)      1422   (5)      1424   (5)
                          #-1425   (5)      1430   (5)    #-1431   (5)    #-1432   (5)    #-1442   (5)    #-1455   (5)
                          #-1477   (5)      1479   (5)    #-1491   (5)      1499   (5)    #-1511   (5)    #-1513   (5)
                          #-1514   (5)      1515   (5)    #-1516   (5)    #-1517   (5)    #-1541   (5)      1542   (5)
                          #-1546   (5)    #-1547   (5)    #-1548   (5)      1549   (5)      1550   (5)      1551   (5)
                          #-1555   (5)    #-1557   (5)    #-1558   (5)    #-1560   (5)    #-1561   (5)      1562   (5)
                            1563   (5)    #-1567   (5)    #-1694   (5)    #-1745   (6)    #-1757   (6)    #-1762   (6)
                          #-1767   (6)    #-1789   (6)    #-1790   (6)      1792   (6)    #-1858   (6)    #-1860   (6)
                          #-1861   (6)    #-1890   (6)    #-1892   (6)    #-1901   (6)    #-1916   (6)    #-1917   (6)
                          #-1951   (6)      1952   (6)    #-1954   (6)    #-1955   (6)    #-1956   (6)      1960   (6)
                          #-1961   (6)    #-1963   (6)    #-1966   (6)    #-2083   (6)      2087   (6)    #-2093   (6)
                            2097   (6)    #-2104   (6)      2108   (6)    #-2117   (6)    #-2119   (6)    #-2120   (6)
                            2121   (6)      2123   (6)    #-2125   (6)    #-2165   (6)    #-2166   (6)    #-2167   (6)
                          #-2212   (6)    #-2234   (6)    #-2258   (6)    #-2263   (6)    #-2265   (6)    #-2380   (6)
                          #-2458   (6)    #-2459   (6)      2460   (6)    #-2466   (6)    #-2467   (6)    #-2468   (6)
                            2469   (6)    #-2471   (6)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    23    00:00:00.03    00:00:00.23
Command processing               874    00:00:00.21    00:00:01.12
Pass 1                          1794    00:00:08.30    00:00:15.60
Symbol table sort                  0    00:00:00.64    00:00:00.64
Pass 2                           144    00:00:02.30    00:00:04.92
Symbol table output                2    00:00:00.14    00:00:00.51
Psect synopsis output              2    00:00:00.00    00:00:00.00
Cross-reference output            28    00:00:01.03    00:00:01.54
Assembler run totals            2868    00:00:12.66    00:00:24.57

The working set limit was 4096 pages.
430317 bytes (841 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2328 non-local and 116 local symbols.
2677 source lines were read in Pass 1, producing 172 object records in Pass 2.
256 pages of virtual memory were used to define 89 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	12
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	20
DISK$DS:[DS.LOCAL.LIBRARY]CRD.MLB;4	2
SYS$COMMON:[SYSLIB]LIB.MLB;2	3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	49
TOTALS (all libraries)	86

2810 GETS were required to define 86 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:KERNEL.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 136 Libraries, Equated Symbols
(1) 160 Work Psect Declarations
(1) 174 Data Psect Declarations
(1) 191 DSV$SETLOAD SET THE DEFAULT LOAD DEVICE/DIRECTORY
(1) 317 DSV$SHOWLOAD Display default load device
(1) 351 DSR$IFLOADDEV Is it the load device?
(1) 399 DSV$LOAD PROGRAM IMAGE LOAD SUBROUTINE

```
0000     1          .Title  LOAD    *** LOAD Set/Show load
0000     2          .Ident  '10.9-03'
0000     3          .NoShow Conditionals
0000     4 ;
0000     5 ; COPYRIGHT (C) 1977, 1980, 1982, 1985, 1986, 1987
0000     6 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     7 ;
0000     8 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000     9 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    10 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    11 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    12 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    13 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    14 ; REMAIN IN DEC.
0000    15 ;
0000    16 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    17 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    18 ; CORPORATION.
0000    19 ;
0000    20 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    22
0000    23 ;++
0000    24 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
0000    25 ;
0000    26 ; ABSTRACT:
0000    27 ;                    This routine is designed to read diagonostic
0000    28 ;       programs into memory.  It removes ALL breakpoints from
0000    29 ;       the previous program and opens the file specified by
0000    30 ;       the LOAD command and reads it starting at memory address
0000    31 ;       DS$GA_PRGBGN (^X200).
0000    32 ;
0000    33 ; ENVIRONMENT:
0000    34 ;       Loads programs in user mode
0000    35 ; and   Loads from floppy and RMS disks in stand-alone
0000    36 ;
0000    37 ; AUTHOR:       TOM SOUTTER     10-NOV-77       VERSION 01
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;                    TOM SOUTTER     21-NOV-77       VERSION 02
0000    41 ;       01           SPR #16  "N.Y.I." MESSAGES FOR STAND-ALONE 'RUN' OR 'LOAD'.
0000    42 ;
0000    43 ;                    TOM SOUTTER     24-MAR-78       VERSION 03 (ESSAA-4.01)
0000    44 ;       02           FIX TO LOAD ALGORITHM TO READ FILE BLOCKS "IN USE" ONLY
0000    45 ;
0000    46 ;                    Roger Riggs     06-Jul-78       Version 04 (ESSAA-4.03)
0000    47 ;       03           Modify load sequence to use $READ and to read whole
0000    48 ;                    file in one operation. verification is equivalent
0000    49 ;
0000    50 ;       04           Eliminate message for successful program load.  /TLS
0000    51 ;                    Roger Riggs     26-Oct-78       Version 05 (ESSAA-5.01)
0000    52 ;       05           Add DSX$LOAD to allow diagnostic to read files into memory
0000    53 ;                    Include facility for stand alone load from ODS-1/ODS-2 structures
0000    54 ;       06           Add floppy support in stand alone
0000    55 ;       07           Add SET LOAD and SHOW LOAD commands
0000    56 ;       08           Add DSR$IFLOADDEV to let attach(select) know if it is
0000    57 ;                    dealing with the load device.
```

```
                        0000     58 ;        09      REMOVE W^ from calls to DSP$XX_READ
                        0000     59 ;
                        0000     60 ;                JOHN CIUKAJ     NOV-79
                        0000     61 ;        10      Modify DSX$LOAD for loadable drivers. BOO$QIO is used for
                        0000     62 ;                resident driver. Pseudo restart parameter block is developed
                        0000     63 ;                for use in BOO$QIO.
                        0000     64 ;
                        0000     65 ;        11      John Ciukaj     2-15-80
                        0000     66 ;                Change RPB block building in LOADFILE_SA to reflect new
                        0000     67 ;                structure of BOOTDRIVER module.
                        0000     68 ;                Roger Riggs     29-FEB-80
                        0000     69 ;        12      Added code to support 7th argument to DS$LOAD allowing
                        0000     70 ;                the program to read sections of the file, the LOAD$_LBN
                        0000     71 ;                argument supplies the starting logical block number of the
                        0000     72 ;                first block to be read from the file.
                        0000     73 ;                Roger Riggs, 5-June-1980, Version 5.4
                        0000     74 ;        13      Moved code to reset stacks and mode to before DS_CLEANUP is
                        0000     75 ;                called.
                        0000     76 ;
                        0000     77 ;                Dave Butenhof, 27-aug-1980
                        0000     78 ;        14      Define DEF$Q_DIR to be global, so it is accessible to
                        0000     79 ;                new standalone RMS routines.
                        0000     80 ;                Also deleted DSX$LOAD routine it has been moved to DSLOAD.
                        0000     81 ;                Roger Riggs, 17-Sep-1980, Version 6.0
                        0000     82 ;        15      Deleted Reference to L$A_LASTADR in DSV$LOAD, MAPFREE
                        0000     83 ;                now checks the loaded bit and L$A_LASTADR
                        0000     84 ;
                        0000     85 ;        16      - Jack Stansbury, 26-Oct-1981, Version 6.-
                        0000     86 ;                Added .LIBRARY statements for $DS and $DIAG.
                        0000     87 ;                Also fixed truncation errors.
                        0000     88 ;
                        0000     89 ;        17      Marion Baggett, 31-Mar-1982, Version 6.7
                        0000     90 ;                Added sys$library:lib .
                        0000     91 ;
                        0000     92 ;        18      Marion Baggett, 7-Apr-1982, Version 6.7
                        0000     93 ;                Added $DS_TYPEDEF to define type def codes.
                        0000     94 ;
                        0000     95 ;        19      Jack Stansbury, 13-April-1982, Version 6.7
                        0000     96 ;                Fixed truncation error.
                        0000     97 ;
                        0000     98 ;        20      Richard Brown, 15-June-82, Version 6.8
                        0000     99 ;                Clear image header storage buffer before loading diag.
                        0000    100 ;                Then save the name of the file being loaded, so the debugger
                        0000    101 ;                can use it later.
                        0000    102 ;                Also, clear debugger-resident flag before loading diag.
                        0000    103 ;
                        0000    104 ;        21      Richard Brown, 8-Mar-84, Version 6.15
                        0000    105 ;                Add code to clear all event flags before a new program
                        0000    106 ;                is loaded.
                        0000    107 ;
                        0000    108 ;        22      RE MUSE         15-MAY-84       VERSION 7.0
                        0000    109 ;                added support for longfile names
                        0000    110 ;
                        0000    111 ;        23      M. Baggett      19-FEB-1985     VERSION 8.1
                        0000    112 ;                Added support for ULTRIX set load.
                        0000    113 ;
                        0000    114 ;        24      M. Baggett      21-May-1985     Version 8.2
```

```
0000    115 ;                   Added more flags for ULTRIX operations
0000    116 ;
0000    117 ;       25          M. Baggett      3-Jun-1985      Version 8.3
0000    118 ;                   Changes to set load to support ULTRIX, remove unneed flags,
0000    119 ;                   and change [23].
0000    120 ;
0000    121 ;       26          Bob Bergazzi    7-Jan-1986      Version 9.1
0000    122 ;                   Removed DEF$Q_DEV, DEF$Q_DIR, SYS$SYSROOT, and SYS$DISK
0000    123 ;                   from this module to the ONLY modules,
0000    124 ;                   so that the default directory for each system could differ.
0000    125 ;
0000    126 ;                   Steven Fortuna  22-May-1987     Version  10.9
0000    127 ;                   Add the capablity for giving a SET LOAD command to a device
0000    128 ;                   name or logical path name without the need for the input
0000    129 ;                   string to be terminated with a ':'.  Symptom was the
0000    130 ;                   SET LOAD command would accept input without a ':' but the
0000    131 ;                   default directory did not change nor was an error reported
0000    132 ;                   to the user.
0000    133 ;
0000    134 ;--
```

```
0000   136          .SBTTL  Libraries, Equated Symbols
0000   137 ;
0000   138 ; INCLUDE FILES:
0000   139 ;
0000   140          .LIBRARY        /SYS$LIBRARY:LIB/                ;              [17]
0000   141          .LIBRARY        /$DS/                            ;              [16]
0000   142          .LIBRARY        /$DIAG/                          ;              [16]
0000   143
0000   144 ;
0000   145 ; EQUATED SYMBOLS:
0000   146 ;
0000   147
0000   148          $DS_HDRDEF                      ; PROGRAM HEADER BLOCK SYMBOLS
0000   149          CLIDEF                          ; CLI COMMAND BLOCK OFFSETS
0000   150          DSFDEF                          ; CONTROL FLAG DEFINITIONS
0000   151          $DS_DSADEF                      ; CONTROL FLAG DEFINITIONS
0000   152          $DS_TYPEDEF                     ; DEFINE TYPE CODES            [18]
0000   153
0000   154 ;Global References                                                    [20]
0000   155
0000   156          .GLOBAL IMAGE_HEADER            ;IMAGE HEADER STORAGE BUFFER   [20]
0000   157          .GLOBAL IMAGE_HEADER_END        ;                              [20]
0000   158
```

```
                 0000          .Subtitle       Work Psect Declarations
                 0000   ;
                 0000   ; OWN STORAGE:
                 0000   ;
                 0000
             00000000          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG
                 0000   ULTRIX_FLAGS::                          ;                                [24]
      00000001   0000          .BLKB   1                       ; FLAG FOR ULTRIX                [23]
      00000000   0001   ULTRIX_V_MODE == 0                      ; BIT POSITION FOR ULTRIX        [24]
      00000056   0001   DIAG_FILE_NAME::.BLKB 39+39+5+2         ; PLACE TO SAVE FILENAME OF LAST[22]
                 0056                                           ; DIAGNOSTIC LOADED              [20]
                 0056
                 0056          .ALIGN  LONG                    ; VMS REQUIREMENT
```

```
                                        0058   174          .Subtitle       Data Psect Declarations
                                        0058   175
                                    00000000   176          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                                        0000   177 DEF_EXE:
30 3B 45 58 45 2E 00000008'010E0000'    0000   178          .ASCID  \.EXE;0\                 ; Default for loaded files
                                        000E   179 FIL$GT_DDDEV::
                                     00'    000E   180          .ASCIC  \\                       ; NO DEFAULT DEVICE
                                     00     000E
                                        000F   181 FIL$GT_DDSTRING::
   5D 54 4E 49 41 4D 53 59 53 5B     00'    000F   182          .ASCIC  \[SYSMAINT]\            ; DEFAULT DIRECTORY IN STAND ALONE
                                     0A     000F
                                        001A   183 T_SHOWLOAD:
         2F 21 53 41 21 53 41 21 00'    001A   184          .ASCIC  "!AS!AS!/"
                                     08     001A
                                        0023   185 T_TRUNCATED:
61 6C 20 6F 6F 74 20 65 6C 69 46 00'    0023   186          .ASCIC  "File too large = !XL, max = !XL!/"
6D 20 2C 4C 58 21 20 3D 20 65 67 72     002F
      2F 21 4C 58 21 20 3D 20 78 61     003B
                                     21     0023
                                        0045   187 CLREF_ERROR_TEXT:                                   ; [21]
6C 63 20 72 6F 72 72 45 20 3F 3F 00'    0045   188          .ASCIC  "?? Error clearing event flags."  ; [21]
74 6E 65 76 65 20 67 6E 69 72 61 65     0051
               2E 73 67 61 6C 66 20     005D
                                     1E     0045
                                        0064   189
```

```
                0064   191          .SBTTL  DSV$SETLOAD     SET THE DEFAULT LOAD DEVICE/DIRECTORY
           00000000    192          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                0000   193 ;++
                0000   194 ; FUNCTIONAL DESCRIPTION:
                0000   195 ;
                0000   196 ;       This routine is called form CLI to process a "SET LOAD <filespec>"
                0000   197 ;       command.  It saves the string pointed to by CLI$Q_FILE(R2) in
                0000   198 ;       DEF$Q_DEV and DEF$Q_DIR if applicable.
                0000   199 ;
                0000   200 ; CALLING SEQUCENCE:
                0000   201 ;
                0000   202 ;       BSBW    DSV$SETLOAD
                0000   203 ;
                0000   204 ; INPUT PARAMETERS:     NONE
                0000   205 ;
                0000   206 ; IMPLICIT INPUTS:
                0000   207 ;
                0000   208 ;       R2      Base of CLI DATA BASE
                0000   209 ;       *       CLI DATA BASE
                0000   210 ;
                0000   211 ; OUTPUT PARAMETERS:    NONE
                0000   212 ;
                0000   213 ; IMPLICIT OUTPUTS:
                0000   214 ;
                0000   215 ;       DEF$Q_DEV gets descriptor of device type
                0000   216 ;               after device is copied to local storage.
                0000   217 ;       DEF$Q_DIR gets descriptor of spec typed.
                0000   218 ;               after spec is copied to local storage
                0000   219 ;
                0000   220 ; COMPLETION CODES:     N/A
```

```
                        0000   222 DSV$SETLOAD::
              3C   BB   0000   223         PUSHR   #^M<R2,R3,R4,R5>        ; Save
           50 05   D0   0002   224         MOVL    #5, R0                  ; Setup to clear 5 quadwords
                        0005   225
              7E   7C   0005   226 10$:    CLRQ    -(SP)                   ; Clear a quadword
            FB 50   F5   0007   227         SOBGTR  R0, 10$                 ; Count and Branch if more
                        000A   228
              6E   7F   000A   229         PUSHAQ  (SP)                    ; Address of descriptors
        7E  08 A2   7D   000C   230         MOVQ    CLI$Q_FILE(R2), -(SP)   ; length, address of new default
00000000'EF 03   FB   0010   231         CALLS   #3, BREAKUP_FILE        ; Merge in new fields
                        0017   232
           50   8E   7D   0017   233         MOVQ    (SP)+,R0                ; Get length/address of new device
              11   13   001A   234         BEQL    20$                     ; Branch if none
00000000'EF   50   D0   001C   235         MOVL    R0,DEF$Q_DEV            ; Set length of new device name
54  00000000'EF   7D   0023   236         MOVQ    DEF$Q_DEV,R4            ; Length/Address to save device
            0106   30   002A   237         BSBW    90$                     ; Copy the device name
                        002D   238
           50   8E   7D   002D   239 20$:   MOVQ    (SP)+, R0               ; Length/addr of directory or partition
              11   13   0030   240         BEQL    25$                     ; Branch if none
00000000'EF   50   D0   0032   241         MOVL    R0,DEF$Q_DIR            ; Set new length of default directory
54  00000000'EF   7D   0039   242         MOVQ    DEF$Q_DIR,R4            ; Length/Address to save directory
            00F0   30   0040   243         BSBW    90$                     ; Copy directory name in                [25]
                        0043   244
                        0043   245 25$:                                    ; Assume file name is a device name
                        0043   246                                        ; due to the fact the device name
           50   6E   7D   0043   247         MOVQ    (SP),R0                 ; was entered with out a ':'.
              34   13   0046   248         BEQL    30$                     ; Branch if no filename
              0C   BB   0048   249         PUSHR   #^M<R2,R3>              ; Save register
        52  10 AE   7D   004A   250         MOVQ    16(SP),R2               ; Check for file type part
              2A   12   004E   251         BNEQ    26$                     ; If present, error, branch out
        52  18 AE   7D   0050   252         MOVQ    24(SP),R2               ; Check for file version
              24   12   0054   253         BNEQ    26$                     ; If present, error, branch out
              52   D4   0056   254         CLRL    R2                      ;
              53   D4   0058   255         CLRL    R3                      ; Clear R2 and R3 for temporary usage
           52 50   B0   005A   256         MOVW    R0,R2                   ; Get the length of the input string.
        53 51 52   C1   005D   257         ADDL3   R2,R1,R3                ; Find the byte one past the end of
           63   3A   90   0061   258         MOVB    #^A":",(R3)             ; string and append a ':'.
              52   B6   0064   259         INCW    R2                      ; Increment the string's length
           50 52   B0   0066   260         MOVW    R2,R0                   ; and move it back to R0.
00000000'EF   50   D0   0069   261         MOVL    R0,DEF$Q_DEV            ; Set length of new device name
54  00000000'EF   7D   0070   262         MOVQ    DEF$Q_DEV,R4            ; Length/Address to save device
            00B9   30   0077   263         BSBW    90$                     ; Copy the device name
                        007A   264
              0C   BA   007A   265 26$:    POPR    #^M<R2,R3>              ; Restore register
                        007C   266 30$:
54  00000000'EF   7D   007C   267         MOVQ    DEF$Q_DIR,R4            ; Length/Address to save directory      [25]
              77   13   0083   268         BEQL    40$                     ; Branch if none                        [25]
        65 54   29   3A   0085   269         LOCC    #^A")",R4,(R5)          ; Check for ultrix                      [25]
              69   13   0089   270         Beql    38$                     ; Branch if not ULTRIX format and       [25]
                        008B   271                                        ; Default to ODS                        [25]
0E 00000000'EF   00   E2   008B   272         BBSS    #ULTRIX_V_MODE,ULTRIX_FLAGS,35$ ; Branch if currently Ultrix   [25]
                        0093   273                                        ; Reset LC only on change over          [25]
           52   6E   7D   0093   274         MOVQ    (SP), R2                ; Get length/address of directory       [25]
        04 83   06   E1   0096   275 31$:    BBC     #6, (R3)+, 32$          ; Branch if not alpha char              [25]
        FF A3   20   88   009A   276         BISB2   #32, -1(R3)             ; Set to lower case                     [25]
            F5 52   F4   009E   277 32$:    SOBGEQ  R2, 31$                 ; Reset all                             [25]
                        00A1   278
```

```
                    6E   D5  00A1   279 35$:   TSTL    (SP)                      ;                                      [25]
                    1E   13  00A3   280        BEQL    37$                       ; Branch if not present                [25]
  00000000'EF   51  55   C3  00A5   281        SUBL3   R5,R1,DEF$Q_DIR           ; Retain length of partition only      [24]
          00000000'EF      D6  00AD   282        INCL    DEF$Q_DIR                 ; Plus ')'                             [24]
            55   01 A1   9E  00B3   283        MOVAB   1(R1),R5                  ; Address  of ')' plus 1               [24]
     00000000'EF   6E   C0  00B7   284        ADDL2   (SP),DEF$Q_DIR            ; Update length of parti/direc ULTRIX  [23]
                50   6E   7D  00BE   285        MOVQ    (SP),R0                   ; Get ULTRIX directory  --/--/--       [24]
                    70   10  00C1   286        BSBB    90$                       ; Copy directory                       [23]
                             00C3   287
                08 AE   D5  00C3   288 37$:   TSTL    8(SP)                     ; Check if 'file' only given           [25]
                    62   13  00C6   289        BEQL    70$                       ; Branch if not present                [25]
  00000000'EF   51  55   C3  00C8   290        SUBL3   R5,R1,DEF$Q_DIR           ; Retain length of partition only      [25]
          00000000'EF      D6  00D0   291        INCL    DEF$Q_DIR                 ; Plus ')'                             [25]
            55   01 A1   9E  00D6   292        MOVAB   1(R1),R5                  ; Address  of ')' plus 1               [25]
   00000000'EF   08 AE   C0  00DA   293        ADDL2   8(SP),DEF$Q_DIR           ; Update length of parti/direc ULTRIX  [25]
          00000000'EF      D7  00E2   294        DECL    DEF$Q_DIR                 ; Do not include '/' prefixed          [25]
            50   08 AE   7D  00E8   295        MOVQ    8(SP),R0                  ; Get ULTRIX directory  ----           [25]
                    51   D6  00EC   296        INCL    R1                        ; Do not include '/' prefixed by the   [25]
                    50   D7  00EE   297        DECL    R0                        ; Breakup routine                      [25]
                    41   10  00F0   298        BSBB    90$                       ; Copy directory                       [25]
                             00F2   299
                    36   11  00F2   300        BRB     70$                       ; EXIT                                 [23]
   00 00000000'EF   00   E5  00F4   301 38$:   BBcc    #ULTRIX_V_MODE,ULTRIX_FLAGS,40$ ; Clear ultrix flag            [25]
                             00FC   302 40$:   Br_If_Not_User  70$               ; Branch if not running in user-mode.  [18]
     0000FE00'EF   1C   E1  00FC               BBC     #DSA$V_User, -                   ; If bit not set, branch
                    26       0103                         L^DSA$GL_Flags, 70$            ;
                             0104   303
                             0104   304        $CRELOG_S #2,SYS$DISK,DEF$Q_DEV ; Set new SYS$DISK
                    00   DD  0104                       PUSHL   #0
          00000000'EF   7F  0106                       PUSHAQ  DEF$Q_DEV
          00000000'EF   7F  010C                       PUSHAQ  SYS$DISK
                    02   DD  0112                       PUSHL   #2
     00000000'GF   04   FB  0114                       CALLS   #4,G^SYS$CRELOG
                    7E   7C  011B   305        CLRQ    -(SP)                     ; Last two parameters null
          00000000'EF   9F  011D   306        PUSHAB  DEF$Q_DIR                 ; Address of new default directory
     00000000'EF   03   FB  0123   307        CALLS   #3,SYS$SETDDIR            ; Set new default directory
                             012A   308
            5E   18   C0  012A   309 70$:   ADDL    #3*8, SP                  ; Remove rest of parameters
                    3C   BA  012D   310        POPR    #^M<R2,R3,R4,R5>          ; Restore
                         05  012F   311        RSB
                             0130   312
            85   81   90  0130   313 80$:   MOVB    (R1)+, (R5)+              ; Copy a character
              FA 50   F4  0133   314 90$:   SOBGEQ  R0,80$                    ; Count and copy if more
                         05  0136   315        RSB
```

```
                     0137   317          .SBTTL  DSV$SHOWLOAD    Display default load device
                     0137   318 ;++
                     0137   319 ; FUNCTIONAL DESCRIPTION:
                     0137   320 ;
                     0137   321 ;       This routine is called from CLI to display the current defaults
                     0137   322 ;       for device and directory.
                     0137   323 ;
                     0137   324 ; CALLING SEQUENCE:
                     0137   325 ;
                     0137   326 ;       BSBW    DSV$SHOWLOAD
                     0137   327 ;
                     0137   328 ; INPUT PARAMETERS:     NONE
                     0137   329 ;
                     0137   330 ; IMPLICIT INPUTS:
                     0137   331 ;
                     0137   332 ;       R2      Address of CLI data base
                     0137   333 ;       .       contents of CLI$Q_FILE
                     0137   334 ;
                     0137   335 ; OUTPUT PARAMETERS:    NONE
                     0137   336 ;
                     0137   337 ; IMPLICIT OUTPUTS:     NONE
                     0137   338 ;
                     0137   339 ; COMPLETION CODES:     N/A
                     0137   340 ;--
                     0137   341
                     0137   342 DSV$SHOWLOAD::
00000000'EF    9F    0137   343          PUSHAB  L^DEF$Q_DIR                             ;               [16]
00000000'EF    7F    013D   344          PUSHAQ  L^DEF$Q_DEV                             ;               [16]
0000001A'EF    9F    0143   345          PUSHAB  L^T_SHOWLOAD                            ;               [16]
      7E    01    9A 0149   346          movzbl  #DS$K_PRINTF,-(sp)                      ;               [17]
      7E    16    98 014C   347          cvtbl   #DS$K_TYPE_COMMAND_OUT,-(sp)            ;               [17]
00000000'EF    05    FB 014F   348          CALLS   #5,DSX$PRINT                            ;               [17]
                     05 0156   349          RSB
```

```
                                0157   351          .SBTTL  DSR$IFLOADDEV   Is it the load device?
                                0157   352 ;**
                                0157   353 ; FUNCTIONAL DESCRIPTION:
                                0157   354 ;
                                0157   355 ;       This routine is called to determine if the device described
                                0157   356 ;       is the load device or not.
                                0157   357 ;
                                0157   358 ; CALLING SEQUENCE:
                                0157   359 ;
                                0157   360 ;       BSBW    DSR$IFLOADDEV
                                0157   361 ;
                                0157   362 ; INPUT PARAMETERS:     NONE
                                0157   363 ;
                                0157   364 ; IMPLICIT INPUTS:
                                0157   365 ;
                                0157   366 ;       R0      Length of device name
                                0157   367 ;       R1      Address of device name
                                0157   368 ;
                                0157   369 ; OUTPUT PARAMETERS:    NONE
                                0157   370 ;
                                0157   371 ; IMPLICIT OUTPUTS:     NONE
                                0157   372 ;
                                0157   373 ; COMPLETION CODES:
                                0157   374 ;
                                0157   375 ;       0       if device was not load device
                                0157   376 ;       1       If device was the load device
                                0157   377 ;
                                0157   378 ;--
                                0157   379
                                0157   380 DSR$IFLOADDEV::
                    3F    BB    0157   381          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save descriptor and regs
54  00000000'EF       7D    0159   382          MOVQ    L^DEF$Q_DEV,R4          ; Point to load device          [16]
    00000000'EF       16    0160   383          Jsb     L^SCAN$DEVICE           ; Extract device name           [18]
                      0C    BA    0166   384          POPR    #^M<R2,R3>              ; Restore desciptor desired
                      12    13    0168   385          BEQL    40$                     ; Branch if no device present
                52    50    B1    016A   386          CMPW    R0,R2                   ; length match?
                      0D    12    016D   387          BNEQ    40$                     ; Branch if not
                            016F   388
                83    81    91    016F   389 20$:     CMPB    (R1)+,(R3)+             ; Character match?
                      08    12    0172   390          BNEQ    40$                     ; Branch if not
                  F8  52    F5    0174   391          SOBGTR  R2,20$                  ; Branch if more to compare
                50    01    D0    0177   392          MOVL    #1,R0                   ; Success
                      02    11    017A   393          BRB     50$                     ; Restore regs and exit
                      50    D4    017C   394 40$:     CLRL    R0                      ; Not load device
                            017E   395
                      3C    BA    017E   396 50$:     POPR    #^M<R2,R3,R4,R5>        ; Restore rest of registers
                            05    0180   397          RSB
```

```
                     0181    399          .SBTTL  DSV$LOAD        PROGRAM IMAGE LOAD SUBROUTINE
                 00000181    400          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                     0181    401 ;++
                     0181    402 ; FUNCTIONAL DESCRIPTION:
                     0181    403 ;
                     0181    404 ;       READS AN PROGRAM IMAGE FILE ('.EXE/-HD') FROM A DISK
                     0181    405 ;       INTO A MEMORY OVERLAY AREA DEFINED BY 'DS$A_PRGBGN' AND
                     0181    406 ;       'DS$K_PRGSIZ'
                     0181    407 ;
                     0181    408 ; CALLING SEQUENCE:
                     0181    409 ;
                     0181    410 ;       BSBW    DSV$RUN                 TO LOAD THEN START
                     0181    411 ;       BSBW    DSV$LOAD
                     0181    412 ;
                     0181    413 ; INPUT PARAMETERS:     NONE
                     0181    414 ;
                     0181    415 ; IMPLICIT INPUTS:
                     0181    416 ;
                     0181    417 ;       CLI$Q_FILE = FILE SPEC (QUADWORD STRING DESCRIPTOR)
                     0181    418 ;
                     0181    419 ; OUTPUT PARAMETERS:    NONE
                     0181    420 ;
                     0181    421 ; IMPLICIT OUTPUTS:     NONE
                     0181    422 ;
                     0181    423 ; COMPLETION CODES:
                     0181    424 ;
                     0181    425 ;       R0      SS$_NORMAL      or failure  from DSX$LOAD
                     0181    426 ;
                     0181    427 ; SIDE EFFECTS:
                     0181    428 ;
                     0181    429 ;       PROGRAM AREA IS CLEARED BEFORE OVERLAY
                     0181    430 ;           (BREAKPOINT TABLE MUST ALSO BE CLEARED)
                     0181    431 ;--
```

```
                                0181   433 DSV$RUN::
                             01 0181   434         NOP                             ; 1 Inst so DSV$LOAD NEQ DSV$RUN
                                0182   435
                                0182   436 DSV$LOAD::
          00000000'EF        16 0182   437         Jsb     SCRIPT$FLUSH            ; Flush scripts if console command      [18]
                                0188   438
          00000000'EF        16 0188   439         Jsb     L^INIT_CONTEXT          ; Reinit stacks, PCB...etc              [18]
    00000000'EF   01        E5 018E   440         BBCC    #DS$V_LODFLG, -
                           06  0195   441                 L^DS$GL_FLAGS,5$        ; INDICATE NO IMAGE LOADED              [16]
          00000000'EF        16 0196   442         Jsb     L^DS_CLEANUP            ; Cleanup if necessary                  [18]
                                019C   443
                                019C   444 5$:     Br_If_User      10$             ; Branch if in user-mode                [18]
    0000FE00'EF   1C        E0 019C               BBS     #DSA$V_User, -          ;   If bit set, branch                  [28]
                           07  01A3                 L^DSA$GL_Flags, 10$     ;                                       [28]
    00000000'EF   6C        FA 01A4   445         CALLG   (AP),MAPFREE            ; Re-map free memory
                                01AB   446
                                01AB   447 10$:
 52 00000000'EF   9E        01AB   448         MOVAB   IMAGE_HEADER,R2         ; GET IMAGE HEADER STORAGE AREA ADDR.[20]
 53 00000000'EF   9E        01B2   449         MOVAB   IMAGE_HEADER_END,R3     ; END OF AREA                           [20]
       53   52   D1        01B9   450 100$:    CMPL    R2,R3                   ; WHILE NOT END OF BUFFER               [20]
            04   13        01BC   451         BEQL    200$                    ; DO                                    [20]
            82   D4        01BE   452         CLRL    (R2)+                   ;   CLEAR THE BUFFER                    [20]
            F7   11        01C0   453         BRB     100$                    ;   CHECK FOR DONE                      [20]
                           01C2   454 200$:                                   ;                                       [20]
 52 00000000'EF   DE        01C2   455         MOVAL   L^DS$GL_CLIBASE,R2      ; Reset R2 to point to cli database     [16]
    7E   0000'8F   3C        01C9   456         MOVZWL  #DS$A_PRGBGN,-(SP)      ; Address to load file
    7E   0000'8F   3C        01CE   457         MOVZWL  #DS$K_PRGSIZ,-(SP)      ; Length of load area
                           01D3   458                                         ; NOW SAVE DIAG. FILENAME FOR DEBUGGER[20]
            38   BB        01D3   459         PUSHR   #^M<R3,R4,R5>           ; SAVE SOME REGISTERS                   [20]
    53   08 A2   DE        01D5   460         MOVAL   CLI$Q_FILE(R2),R3       ; GET FILENAME DESCRIPTOR               [20]
54  00000001'EF   DE        01D9   461         MOVAL   DIAG_FILE_NAME,R4       ; GET STORAGE AREA                      [20]
       64   63   90        01E0   462         MOVB    (R3),(R4)               ; GET CHAR. COUNT                       [20]
       53   04   C0        01E3   463         ADDL2   #4,R3                   ; GET STRING POINTER                    [20]
       53   63   D0        01E6   464         MOVL    (R3),R3                 ; GET STRING                            [20]
       55   84   90        01E9   465         MOVB    (R4)+,R5                ; SET UP COUNTER OF CHARS TO MOVE[20]
                           01EC   466 300$:                                   ; REPEAT                                [20]
       84   83   90        01EC   467         MOVB    (R3)+,(R4)+             ;   MOVE A CHAR. TO SAVE AREA           [20]
            55   97        01EF   468         DECB    R5                      ;   COUNT A CHARACTER                   [20]
            F9   14        01F1   469         BGTR    300$                    ; UNTIL ENTIRE STRING COPIED            [20]
            38   BA        01F3   470         POPR    #^M<R3,R4,R5>           ; RESTORE SOME REGISTERS                [20]
    04000000   8F   CA        01F5   471         BICL    #1@DSA$V_DEBUG,-        ; CLEAR THE DEBUGGER-RESIDENT           [20]
    0000FE00'EF               01FB   472                 DSA$GL_FLAGS            ;   FLAG                               [20]
                           0200   473 ;
                           0200   474 ; Now clear all event flags.                                                      [21]
                           0200   475 ;
            08   BB        0200   476         PUSHR   #^M<R3>                 ; Use R3 to hold EF Number              [21]
       53   01   D0        0202   477         MOVL    #1, R3                  ; Start with EFN 1; VDS uses 0          [21]
            1C   E1        0205   478 400$:    BBC     #DSA$V_USER,-           ; IF user mode                          [21]
 0C 0000FE00'EF           0207   479                 DSA$GL_FLAGS,410$       ; THEN                                  [21]
       18   53   D1        020D   480         CMPL    R3, #24                 ;   IF EFN >= 24                        [21]
            07   19        0210   481         BLSS    410$                    ;   AND EFN <= 31                       [21]
       1F   53   D1        0212   482         CMPL    R3, #31                 ;   THEN                                [21]
            02   14        0215   483         BGTR    410$                    ;       Skip this EFN                   [21]
            0C   11        0217   484         BRB     420$                    ;                                       [21]
            53   DD        0219   485 410$:    PUSHL   R3                      ; Put EFN on stack                      [21]
00000000'EF   01   FB        021B   486         CALLS   #1, SYS$CLREF           ; Call service                          [21]
         08   50   E9        0222   487         BLBC    R0, 425$                ; Branch on error                       [21]
```

```
                DC 53   3F   F3  0225   488 420$:   AOBLEQ  #63, R3, 400$           ; Next EFN                          [21]
                        08   BA  0229   489           POPR    #^M<R3>               ; Restore R3                        [21]
                        17   11  022B   490           BRB     430$                  ; Continue                          [21]
                                 022D   491
                                 022D   492 425$:                                  ; Come here on $CLREF error         [21]
                        08   BA  022D   493           POPR    #^M<R3>               ; Restore R3                        [21]
          00000045'EF   9F   022F   494           PUSHAB  CLREF_ERROR_TEXT      ; Put address of msg. on stack      [21]
             7E   01   9A  0235   495           movzbl  #DS$K_PRINTF,-(sp)    ;                                   [21]
             7E   16   98  0238   496           cvtbl   #DS$K_TYPE_COMMAND_OUT,-(sp);                             [21]
       00000000'EF   03   FB  023B   497           CALLS   #3,DSX$PRINTF         ; Print error message               [21]
                        47   11  0242   498           BRB     DSV$LOAD_X            ; And leave.                        [21]
                                 0244   499
                                 0244   500 430$:
          00000000'EF   7F   0244   501           PUSHAQ  L^DEF_EXE             ; Address of defaults for file              [16]
                   08 A2   7F   024A   502           PUSHAQ  CLI$Q_FILE(R2)        ; Address of file descriptor
       00000000'EF   04   FB  024D   503           CALLS   #4,L^DSX$LOAD         ; Load the file specified           [16]
          00000000'EF   16   0254   504           Jsb     DSR$COMPLETION        ; Type error text if necessary      [18]
                                 025A   505
                      2E 50   E9  025A   506 30$:     BLBC    R0,DSV$LOAD_X         ; Branch if it failed
                                 025D   507           Set_LodFlg                    ; Indicate an image is loaded       [18]
       00000000'EF   01   E2  025D                 BBSS    #DS$V_LodFlg, -       ; Branch if LODFLG bit is set to    [29]
                        00   0264                   L^DS$GL_Flags, 30000$ ; ...                               [29]
                                 0265           30000$:                       ; ... here. Set bit regardless.     [29]
             53   0200'8F   3C  0265   508           MOVZWL  #DS$K_PRGSIZ*512,R3   ; Save max size
       51   00000214'EF   D0  026A   509           MOVL    L$A_LASTADR,R1        ; Get last address
                   51   53   D1  0271   510           CMPL    R3,R1                 ; Compare max to current
                        15   1E  0274   511           BGEQU   DSV$LOAD_X            ; Small enough, continue
                        0A   BB  0276   512           PUSHR   #^M<R1,R3>            ; Push Max size then current size
          00000023'EF   9F   0278   513           PUSHAB  T_TRUNCATED           ; Text of message
             7E   01   9A  027E   514           movzbl  #DS$K_PRINTF,-(sp)    ;                                   [17]
             7E   16   98  0281   515           cvtbl   #DS$K_TYPE_COMMAND_OUT,-(sp)  ;                           [17]
       00000000'EF   05   FB  0284   516           CALLS   #5,DSX$PRINT          ; Tell the operator                 [17]
                                 028B   517
                                 028B   518 DSV$LOAD_X:
          00000000'EF   9F   028B   519           PUSHAB  L^BEGIN               ; Fake JSB for VRSTART or error     [18]
                      0D 50   E9  0291   520           BLBC    R0,10$                ; If load failed, go back to BEGIN
    00000181'8F   04 A2   D1  0294   521           CMPL    CLI$L_COMMAND(R2), -
                                 029C   522                   #DSV$RUN              ; Was it run command?
                        03   12  029C   523           BNEQ    10$                   ; Branch if not run command
                   FD5F'   31  029E   524           BRW     VRSTART               ; Now issue START command
                        05   02A1   525 10$:    RSB                           ; Return to BEGIN
                                 02A2   526
                                 02A2   527           .END
```

```
$$T1                            = 00000000    D
BEGIN                             ********  X   04
BIT...                          = 00000004    D
BREAKUP_FILE                      ********  X   04
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        0000044C    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_FLAGS2                      00000444    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEMP_BASE                   00000448    D
CLI$L_TEST                        00000020    D
CLI$M_START_OR_RUN              = 00000008    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BASE_QUAL                 = 00000002    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NEXT                      = 00000001    D
CLI$V_NOALLOC                   = 00000000    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_START_OR_RUN              = 00000003    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
CLREF_ERROR_TEXT                  00000045 R  D   03
DEF$Q_DEV                         ********    X   04
DEF$Q_DIR                         ********    X   04
DEF_EXE                           00000000 R  D   03
DIAG_FILE_NAME                    00000001 RG D   02
DS$A_PRGBGN                       ********    X   04
DS$GL_CLIBASE                     ********    X   04
DS$GL_FLAGS                       ********    X   04
DS$K_PRGSIZ                       ********    X   04
DS$K_PRINTB                     = 00000002    D
DS$K_PRINTF                     = 00000001    D
DS$K_PRINTI                     = 00000000    D
DS$K_PRINTX                     = 00000003    D
DS$K_TYPE_ABORT_PROGRAM         = 00000014    D
DS$K_TYPE_ABORT_TEST            = 00000013    D
DS$K_TYPE_COMMAND_ERR           = 00000015    D
DS$K_TYPE_COMMAND_OUT           = 00000016    D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A    D
DS$K_TYPE_DS_PROMPT             = 00000001    D
DS$K_TYPE_DS_START              = 0000001D    D
DS$K_TYPE_ERRDEV                = 00000008    D
DS$K_TYPE_ERRHARD               = 00000006    D
DS$K_TYPE_ERROR_BODY            = 00000009    D
DS$K_TYPE_ERROR_END             = 0000000A    D
DS$K_TYPE_ERRPREP               = 0000001B    D
DS$K_TYPE_ERRSOFT               = 00000007    D
DS$K_TYPE_ERRSUP                = 00000004    D
DS$K_TYPE_ERRSYS                = 00000005    D
DS$K_TYPE_ERR_HALT              = 0000000D    D
DS$K_TYPE_EXCEPTION             = 0000000C    D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D
DS$K_TYPE_FIRST_PASS            = 00000011    D
DS$K_TYPE_GENERAL               = 00000000    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D
DS$K_TYPE_NO_TESTS              = 00000012    D
DS$K_TYPE_PARAM_ERROR           = 0000001C    D
DS$K_TYPE_PROGRAM_END           = 00000010    D
DS$K_TYPE_PROGRAM_INFO          = 00000017    D
DS$K_TYPE_PROGRAM_START         = 0000000F    D
DS$K_TYPE_QIO_INVADP            = 00000024    D
DS$K_TYPE_QIO_NODRIVER          = 00000022    D
DS$K_TYPE_QIO_WRONGVER          = 00000023    D
DS$K_TYPE_SCRIPT_ECHO           = 00000021    D
DS$K_TYPE_SCRIPT_PNF            = 0000001E    D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020    D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F    D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D
DS$K_TYPE_START_ERR             = 00000018    D
DS$K_TYPE_START_LIST            = 00000025    D
DS$K_TYPE_SUMMARY               = 0000000E    D
DS$K_TYPE_USER_PROMPT           = 00000002    D
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
```

```
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_NI                         = 04000000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMED                      = 02000000     D
DS$M_TIMED_OUT                  = 08000000     D
DS$M_TIMRON                     = 00020000     D
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_NI                         = 0000001A     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMED                      = 00000019     D
DS$V_TIMED_OUT                  = 0000001B     D
DS$V_TIMRON                     = 00000011     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$V_DEBUG                     = 0000001A     D
DSA$V_USER                      = 0000001C     D
DSR$COMPLETION                    ********   X    04
DSR$IFLOADDEV                     00000157 RG  D   04
DSV$LOAD                          00000182 RG  D   04
DSV$LOAD_X                        0000028B R   D   04
DSV$RUN                           00000181 RG  D   04
DSV$SETLOAD                       00000000 RG  D   04
DSV$SHOWLOAD                      00000137 RG  D   04
DSX$LOAD                          ********    X   04
DSX$PRINT                         ********    X   04
DSX$PRINTF                        ********    X   04
DS_CLEANUP                        ********    X   04
FIL$GT_DDDEV                      0000000E RG  D   03
FIL$GT_DDSTRING                   0000000F RG  D   03
IMAGE_HEADER                      ********   GX   00
IMAGE_HEADER_END                  ********   GX   00
INIT_CONTEXT                      ********    X   04
L$A_CCP                           00000240     D
L$A_DEVP                          0000021C     D
L$A_DREG                          00000224     D
L$A_DTP                           00000218     D
L$A_ICP                           0000023C     D
L$A_LASTADR                       00000214     D
L$A_NAME                          00000208     D
L$A_REPP                          00000244     D
L$A_SECNAM                        00000250     D
L$A_STATAB                        00000248     D
L$A_TSTCNT                        00000254     D
L$L_ENVIRON                       00000204     D
L$L_ERRTYP                        0000024C     D
L$L_HEADLENGTH                    00000200     D
L$L_REV                           0000020C     D
L$L_UNIT                          00000220     D
L$L_UNUSED                        00000228     D
L$L_UPDATE                        00000210     D
MAPFREE                           ********    X   04
SCAN$DEVICE                       ********    X   04
SCRIPT$FLUSH                      ********    X   04
SIZ...                          = 00000001     D
SYS$CLREF                         ********    X   04
SYS$CRELOG                        ********   GX   04
```

```
SYS$DISK                    ********   X   04
SYS$SETDDIR                 ********   X   04
T_SHOWLOAD                  0000001A R  D  03
T_TRUNCATED                 00000023 R  D  03
ULTRIX_FLAGS                00000000 RG D  02
ULTRIX_V_MODE             = 00000000  G D
VRSTART                     ********   X   04
```

```
+----------------+
! Psect synopsis !
+----------------+
```

PSECT name	Allocation		PSECT No.	Attributes										
. ABS .	00000000	(0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
ABS	0000FE70	(65136.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
WORK	00000058	(88.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	NOEXE	RD	WRT	NOVEC	LONG
DATA	00000064	(100.)	03 (3.)	NOPIC	USR	CON	REL	LCL	SHR	NOEXE	RD	NOWRT	NOVEC	BYTE
CODE	000002A2	(674.)	04 (4.)	NOPIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T1	=00000000	304 (1)	304 (1)
BEGIN	00000000-XR		519 (1)
BIT...	=00000004	152 (1)	150 (1) 152 (1)
BREAKUP_FILE	00000000-XR		231 (1)
CLI$K_BUFSIZ	=00000100	149 (1)	149 (1)
CLI$K_SIZE	0000044C	149 (1)	
CLI$L_ADDRESS	00000018	149 (1)	
CLI$L_COMMAND	00000004	149 (1)	#-521 (1)
CLI$L_DATA	0000001C	149 (1)	
CLI$L_FLAGS	00000000	149 (1)	
CLI$L_FLAGS2	00000444	149 (1)	
CLI$L_LAST	00000024	149 (1)	
CLI$L_NEXT	00000030	149 (1)	
CLI$L_PASS	0000002C	149 (1)	
CLI$L_SUBT	00000028	149 (1)	
CLI$L_TEMP_BASE	00000448	149 (1)	
CLI$L_TEST	00000020	149 (1)	
CLI$M_START_OR_RUN	=00000008	149 (1)	
CLI$Q_BUFQWD	00000034	149 (1)	
CLI$Q_FILE	00000008	149 (1)	#-230 (1) 460 (1) 502 (1)
CLI$Q_SECTION	00000010	149 (1)	
CLI$Q_TIME	0000043C	149 (1)	
CLI$T_BUFFER	0000003C	149 (1)	
CLI$V_ADAPTER	=00000018	149 (1)	
CLI$V_ADR	=0000000B	149 (1)	
CLI$V_ASCII	=00000013	149 (1)	
CLI$V_BASE_QUAL	=00000002	149 (1)	
CLI$V_BREAK	=0000000A	149 (1)	
CLI$V_BRIEF	=0000001B	149 (1)	
CLI$V_BYTE	=0000000D	149 (1)	
CLI$V_CLEAR	=00000002	149 (1)	
CLI$V_DEC	=00000010	149 (1)	
CLI$V_DEFAULT	=0000000C	149 (1)	
CLI$V_DEPOSIT	=00000019	149 (1)	
CLI$V_EVENT	=00000008	149 (1)	
CLI$V_EXAM	=00000005	149 (1)	
CLI$V_FLAGS	=00000009	149 (1)	
CLI$V_HEX	=00000012	149 (1)	
CLI$V_KERNEL	=00000017	149 (1)	
CLI$V_LOAD	=00000006	149 (1)	
CLI$V_LONG	=0000000F	149 (1)	
CLI$V_NEXT	=00000001	149 (1)	
CLI$V_NOALLOC	=00000000	149 (1)	
CLI$V_NOTNUF	=00000001	149 (1)	
CLI$V_OCT	=00000011	149 (1)	
CLI$V_PREG	=0000001A	149 (1)	
CLI$V_QA	=00000007	149 (1)	
CLI$V_QACKLOOPLOOPS	=0000001C	149 (1)	
CLI$V_QAERRORPRINTS	=0000001B	149 (1)	
CLI$V_QAMULTIPLEPASS	=0000001F	149 (1)	

```
CLI$V_QASUBTESTLOOPS            =0000001E     149    (1)
CLI$V_QATESTLOOPS               =0000001D     149    (1)
CLI$V_REG                       =00000014     149    (1)
CLI$V_REQUIRED                  =00000000     149    (1)
CLI$V_RUN                       =00000015     149    (1)
CLI$V_SET                       =00000003     149    (1)
CLI$V_SHOW                      =00000004     149    (1)
CLI$V_START_OR_RUN              =00000003     149    (1)     149    (1)
CLI$V_VALSEC                    =00000016     149    (1)
CLI$V_WORD                      =0000000E     149    (1)
CLREF_ERROR_TEXT                 00000045-R   187    (1)     494    (1)
DEF$Q_DEV                        00000000-XR                #-235   (1)    #-236   (1)    #-261   (1)    #-262   (1)
                                                             304   (1)     344   (1)    #-382   (1)
DEF$Q_DIR                        00000000-XR                #-241   (1)    #-242   (1)    #-267   (1)    #-281   (1)
                                                           #-282   (1)    #-284   (1)    #-290   (1)    #-291   (1)
                                                           #-293   (1)    #-294   (1)     306   (1)     343   (1)
DEF_EXE                          00000000-R   177    (1)     501   (1)
DIAG_FILE_NAME                   00000001-R   169    (1)     461   (1)
DS$A_PRGBGN                      00000000-XR                #-456   (1)
DS$GL_CLIBASE                    00000000-XR                 455   (1)
DS$GL_FLAGS                      00000000-XR                 441   (1)     507   (1)
DS$K_PRGSIZ                      00000000-XR                #-457   (1)    #-508   (1)
DS$K_PRINTB                     =00000002     152    (1)
DS$K_PRINTF                     =00000001     152    (1)    #-346   (1)    #-495   (1)    #-514   (1)
DS$K_PRINTI                     =00000000     152    (1)
DS$K_PRINTX                     =00000003     152    (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     152    (1)
DS$K_TYPE_ABORT_TEST            =00000013     152    (1)
DS$K_TYPE_COMMAND_ERR           =00000015     152    (1)
DS$K_TYPE_COMMAND_OUT           =00000016     152    (1)    #-347   (1)    #-496   (1)    #-515   (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     152    (1)
DS$K_TYPE_DS_PROMPT             =00000001     152    (1)
DS$K_TYPE_DS_START              =0000001D     152    (1)
DS$K_TYPE_ERRDEV                =00000008     152    (1)
DS$K_TYPE_ERRHARD               =00000006     152    (1)
DS$K_TYPE_ERROR_BODY            =00000009     152    (1)
DS$K_TYPE_ERROR_END             =0000000A     152    (1)
DS$K_TYPE_ERRPREP               =0000001B     152    (1)
DS$K_TYPE_ERRSOFT               =00000007     152    (1)
DS$K_TYPE_ERRSUP                =00000004     152    (1)
DS$K_TYPE_ERRSYS                =00000005     152    (1)
DS$K_TYPE_ERR_HALT              =0000000D     152    (1)
DS$K_TYPE_EXCEPTION             =0000000C     152    (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     152    (1)
DS$K_TYPE_FIRST_PASS            =00000011     152    (1)
DS$K_TYPE_GENERAL               =00000000     152    (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     152    (1)
DS$K_TYPE_NO_TESTS              =00000012     152    (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     152    (1)
DS$K_TYPE_PROGRAM_END           =00000010     152    (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     152    (1)
DS$K_TYPE_PROGRAM_START         =0000000F     152    (1)
DS$K_TYPE_QIO_INVADP            =00000024     152    (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     152    (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     152    (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     152    (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     152    (1)
```

```
DS$K_TYPE_SCRIPT_PROMPT         =00000020     152    (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     152    (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     152    (1)
DS$K_TYPE_START_ERR             =00000018     152    (1)
DS$K_TYPE_START_LIST            =00000025     152    (1)
DS$K_TYPE_SUMMARY               =0000000E     152    (1)
DS$K_TYPE_USER_PROMPT           =00000002     152    (1)
DS$M_ABRTFLG                    =00000040     150    (1)
DS$M_BADTIME                    =00100000     150    (1)
DS$M_BATCH                      =00400000     150    (1)
DS$M_BRKCLR                     =00001000     150    (1)
DS$M_BRKPT                      =00000800     150    (1)
DS$M_CHARFLG                    =00000100     150    (1)
DS$M_CMDFLG                     =00000080     150    (1)
DS$M_CTRLC                      =00000001     150    (1)
DS$M_CTRLO                      =00010000     150    (1)
DS$M_DEVFLG                     =00000200     150    (1)
DS$M_DISABLCC                   =01000000     150    (1)
DS$M_DOMFLG                     =00002000     150    (1)
DS$M_ERRFLG                     =00000010     150    (1)
DS$M_EXCEPT                     =00080000     150    (1)
DS$M_EXETST                     =00040000     150    (1)
DS$M_HLTFLG                     =00000008     150    (1)
DS$M_LODFLG                     =00000002     150    (1)
DS$M_MEMMGT                     =00008000     150    (1)
DS$M_NI                         =04000000     150    (1)
DS$M_OUTPUT                     =00800000     150    (1)
DS$M_RUBFLG                     =00000020     150    (1)
DS$M_SCRIPT                     =00200000     150    (1)
DS$M_SETIMR                     =02000000     150    (1)
DS$M_STRFLG                     =00000004     150    (1)
DS$M_SUBT                       =00004000     150    (1)
DS$M_SYSFLG                     =00000400     150    (1)
DS$M_TIMED                      =02000000     150    (1)
DS$M_TIMED_OUT                  =08000000     150    (1)
DS$M_TIMROM                     =00020000     150    (1)
DS$V_ABRTFLG                    =00000006     150    (1)
DS$V_BADTIME                    =00000014     150    (1)
DS$V_BATCH                      =00000016     150    (1)
DS$V_BRKCLR                     =0000000C     150    (1)
DS$V_BRKPT                      =0000000B     150    (1)
DS$V_CHARFLG                    =00000008     150    (1)
DS$V_CMDFLG                     =00000007     150    (1)
DS$V_CTRLC                      =00000000     150    (1)
DS$V_CTRLO                      =00000010     150    (1)
DS$V_DEVFLG                     =00000009     150    (1)
DS$V_DISABLCC                   =00000018     150    (1)
DS$V_DOMFLG                     =0000000D     150    (1)
DS$V_ERRFLG                     =00000004     150    (1)
DS$V_EXCEPT                     =00000013     150    (1)
DS$V_EXETST                     =00000012     150    (1)
DS$V_HLTFLG                     =00000003     150    (1)
DS$V_LODFLG                     =00000001     150    (1)    #-440    (1)    #-507    (1)
DS$V_MEMMGT                     =0000000F     150    (1)
DS$V_NI                         =0000001A     150    (1)
DS$V_OUTPUT                     =00000017     150    (1)
DS$V_RUBFLG                     =00000005     150    (1)
```

```
DS$V_SCRIPT                     =00000015      150     (1)
DS$V_SETIMR                     =00000019      150     (1)
DS$V_STRFLG                     =00000002      150     (1)
DS$V_SUBT                       =0000000E      150     (1)
DS$V_SYSFLG                     =0000000A      150     (1)
DS$V_TIMED                      =00000019      150     (1)
DS$V_TIMED_OUT                  =0000001B      150     (1)
DS$V_TIMRON                     =00000011      150     (1)
DSA$GL_FLAGS                     0000FE00                      302     (1)      444     (1)    #-472     (1)      479     (1)
DSA$V_DEBUG                     =0000001A                    #-471     (1)
DSA$V_USER                      =0000001C                    #-302     (1)    #-444     (1)    #-478     (1)
DSR$COMPLETION                   00000000-XR                  504     (1)
DSR$IFLOADDEV                    00000157-R     380     (1)
DSV$LOAD                         00000182-R     436     (1)
DSV$LOAD_X                       0000028B-R     518     (1)    #-498     (1)    #-506     (1)    #-511     (1)
DSV$RUN                          00000181-R     433     (1)    #-522     (1)
DSV$SETLOAD                      00000000-R     222     (1)
DSV$SHOWLOAD                     00000137-R     342     (1)
DSX$LOAD                         00000000-XR                  503     (1)
DSX$PRINT                        00000000-XR                  348     (1)      516     (1)
DSX$PRINTF                       00000000-XR                  497     (1)
DS_CLEANUP                       00000000-XR                  442     (1)
FIL$GT_DDDEV                     0000000E-R     179     (1)
FIL$GT_DDSTRING                  0000000F-R     181     (1)
IMAGE_HEADER                     00000000-XR                  156     (1)      448     (1)
IMAGE_HEADER_END                 00000000-XR                  157     (1)      449     (1)
INIT_CONTEXT                     00000000-XR                  439     (1)
L$A_LASTADR                      00000214                    #-509     (1)
MAPFREE                          00000000-XR                  445     (1)
SCAN$DEVICE                      00000000-XR                  383     (1)
SCRIPT$FLUSH                     00000000-XR                  437     (1)
SIZ...                          =00000001      150     (1)      150     (1)
SYS$CLREF                        00000000-XR                  486     (1)
SYS$CRELOG                       00000000-XR                  304     (1)
SYS$DISK                         00000000-XR                  304     (1)
SYS$SETDDIR                      00000000-XR                  307     (1)
T_SHOWLOAD                       0000001A-R     183     (1)      345     (1)
T_TRUNCATED                      00000023-R     185     (1)      513     (1)
ULTRIX_FLAGS                     00000000-R     166     (1)      272     (1)      301     (1)
ULTRIX_V_MODE                   =00000000      168     (1)    #-272     (1)    #-301     (1)
VRSTART                          00000000-XR                  #-524     (1)
```

```
                                        +-------------------------+
                                        ! Macros Cross Reference !
                                        +-------------------------+

MACRO            SIZE        DEFINITION      REFERENCES...
-----            ----        ----------      -------------
$ASNPUSH         1           304    (1)      304    (1)
$CRELOG_S        1           304    (1)      304    (1)
$DEF             1           152    (1)
$DEFINI          1           148    (1)      148    (1)      151    (1)
$DS_DSADEF       5           151    (1)      151    (1)
$DS_HDRDEF       2           148    (1)      148    (1)
$DS_TYPEDEF      4           152    (1)      152    (1)
$EQU             1           152    (1)      152    (1)
$EQULS1          1           152    (1)      152    (1)
$EQULST          1           152    (1)      152    (1)
$GBLINI          2                           150    (1)      152    (1)
$PUSHADR         1           304    (1)      304    (1)
$VIELD           1           150    (1)      150    (1)
$VIELD1          1           152    (1)      150    (1)
BR_IF_NOT_USER   1           302    (1)      302    (1)
BR_IF_USER       1           444    (1)      444    (1)
CLIDEF           4           149    (1)      149    (1)
DSFDEF           3           150    (1)      150    (1)
SET_LODFLG       1           507    (1)      507    (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	309 (1)
ADDL2	00C0	284 (1) 293 (1) 463 (1)
ADDL3	00C1	257 (1)
AOBLEQ	00F3	488 (1)
BBC	00E1	275 (1) 302 (1) 478 (1)
BBCC	00E5	301 (1) 440 (1)
BBS	00E0	444 (1)
BBSS	00E2	272 (1) 507 (1)
BEQL	0013	234 (1) 240 (1) 248 (1) 268 (1) 270 (1) 280 (1) 289 (1)
		385 (1) 451 (1)
BGEQU	001E	511 (1)
BGTR	0014	469 (1) 483 (1)
BICL	00CA	471 (1)
BISB2	0088	276 (1)
BLBC	00E9	487 (1) 506 (1) 520 (1)
BLSS	0019	481 (1)
BNEQ	0012	251 (1) 253 (1) 387 (1) 390 (1) 523 (1)
BRB	0011	300 (1) 393 (1) 453 (1) 484 (1) 490 (1) 498 (1)
BRW	0031	524 (1)
BSBB	0010	286 (1) 298 (1)
BSBW	0030	237 (1) 243 (1) 263 (1)
CALLG	00FA	445 (1)
CALLS	00FB	231 (1) 304 (1) 307 (1) 348 (1) 486 (1) 497 (1) 503 (1)
		516 (1)
CLRL	00D4	254 (1) 255 (1) 394 (1) 452 (1)
CLRQ	007C	226 (1) 305 (1)
CMPB	0091	389 (1)
CMPL	00D1	450 (1) 480 (1) 482 (1) 510 (1) 521 (1)
CMPW	00B1	386 (1)
CVTBL	0098	347 (1) 496 (1) 515 (1)
DECB	0097	468 (1)
DECL	00D7	294 (1) 297 (1)
INCL	00D6	282 (1) 291 (1) 296 (1)
INCW	00B6	259 (1)
JSB	0016	383 (1) 437 (1) 439 (1) 442 (1) 504 (1)
LOCC	003A	269 (1)
MOVAB	009E	283 (1) 292 (1) 448 (1) 449 (1)
MOVAL	00DE	455 (1) 460 (1) 461 (1)
MOVB	0090	258 (1) 313 (1) 462 (1) 465 (1) 467 (1)
MOVL	00D0	224 (1) 235 (1) 241 (1) 261 (1) 392 (1) 464 (1) 477 (1)
		509 (1)
MOVQ	007D	230 (1) 233 (1) 236 (1) 239 (1) 242 (1) 247 (1) 250 (1)
		252 (1) 262 (1) 267 (1) 274 (1) 285 (1) 295 (1) 382 (1)
MOVW	00B0	256 (1) 260 (1)
MOVZBL	009A	346 (1) 495 (1) 514 (1)
MOVZWL	003C	456 (1) 457 (1) 508 (1)
NOP	0001	434 (1)
POPR	00BA	265 (1) 310 (1) 384 (1) 396 (1) 470 (1) 489 (1) 493 (1)
PUSHAB	009F	306 (1) 343 (1) 345 (1) 494 (1) 513 (1) 519 (1)
PUSHAQ	007F	229 (1) 304 (1) 344 (1) 501 (1) 502 (1)

PUSHL	00DD	304	(1)	485	(1)								
PUSHR	00BB	223	(1)	249	(1)	381	(1)	459	(1)	476	(1)	512	(1)
RSB	0005	311	(1)	315	(1)	349	(1)	397	(1)	525	(1)		
SOBGEQ	00F4	277	(1)	314	(1)								
SOBGTR	00F5	227	(1)	391	(1)								
SUBL3	00C3	281	(1)	290	(1)								
TSTL	00D5	279	(1)	288	(1)								

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ALIGN	172	(1)														
.ASCIC	180	(1)	182	(1)	184	(1)	186	(1)	188	(1)						
.ASCID	178	(1)														
.BLKB	167	(1)	169	(1)												
.BLKL	149	(1)														
.END	527	(1)														
.ENDC	150	(1)	152	(1)	304	(1)										
.ENDM	148	(1)	149	(1)	150	(1)	151	(1)	152	(1)	302	(1)	304	(1)	444	(1)
	507	(1)														
.ENDR	150	(1)	152	(1)												
.GLOBAL	156	(1)	157	(1)												
.GLOBL	304	(1)														
.IDENT	2	(1)														
.IF	150	(1)	152	(1)	304	(1)										
.IFF	150	(1)	152	(1)	304	(1)										
.IIF	150	(1)	152	(1)												
.IRP	150	(1)	152	(1)												
.LIBRARY	140	(1)	141	(1)	142	(1)										
.LIST	148	(1)	149	(1)	151	(1)										
.MACRO	150	(1)	152	(1)												
.NLIST	148	(1)	149	(1)	151	(1)										
.NOCROSS	148	(1)	151	(1)												
.NOSHOW	3	(1)														
.PAGE	135	(1)	159	(1)	173	(1)	190	(1)	221	(1)	316	(1)	350	(1)	398	(1)
	432	(1)														
.PSECT	149	(1)	165	(1)	176	(1)	192	(1)	400	(1)						
.RESTORE	148	(1)	149	(1)	151	(1)										
.SAVE	148	(1)	149	(1)	151	(1)										
.SBTTL	136	(1)	191	(1)	317	(1)	351	(1)	399	(1)						
.SUBTITLE	160	(1)	174	(1)												
.TITLE	1	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	1	445 (1)
R0	21	#-224 (1) #-227 (1) #-233 (1) #-235 (1) #-239 (1) #-241 (1) #-247 (1) #-256 (1) #-260 (1) #-261 (1) #-285 (1) #-295 (1) #-297 (1) #-314 (1) #-381 (1) #-386 (1) #-392 (1) #-394 (1) #-487 (1) #-506 (1) #-520 (1)
R1	12	#-257 (1) #-281 (1) 283 (1) #-290 (1) 292 (1) #-296 (1) #-313 (1) #-381 (1) #-389 (1) #-509 (1) #-510 (1) #-512 (1)
R2	26	#-223 (1) #-230 (1) #-249 (1) #-250 (1) #-252 (1) #-254 (1) #-256 (1) #-257 (1) #-259 (1) #-260 (1) #-265 (1) #-274 (1) #-277 (1) #-310 (1) #-381 (1) #-384 (1) #-386 (1) #-391 (1) #-396 (1) #-448 (1) #-450 (1) #-452 (1) #-455 (1) 460 (1) 502 (1) #-521 (1)
R3	34	#-223 (1) #-249 (1) #-255 (1) #-257 (1) #-258 (1) #-265 (1) 275 (1) #-276 (1) #-310 (1) #-381 (1) #-384 (1) #-389 (1) #-396 (1) #-449 (1) #-450 (1) #-459 (1) #-460 (1) #-462 (1) #-463 (1) #-464 (1) #-467 (1) #-470 (1) #-476 (1) #-477 (1) #-480 (1) #-482 (1) #-485 (1) #-488 (1) #-489 (1) #-493 (1) #-508 (1) #-510 (1) #-512 (1)
R4	16	#-223 (1) #-236 (1) #-242 (1) #-262 (1) #-267 (1) #-269 (1) #-310 (1) #-381 (1) #-382 (1) #-396 (1) #-459 (1) #-461 (1) #-462 (1) #-465 (1) #-467 (1) #-470 (1)
R5	14	#-223 (1) 269 (1) #-281 (1) #-283 (1) #-290 (1) #-292 (1) #-310 (1) #-313 (1) #-381 (1) #-396 (1) #-459 (1) #-465 (1) #-468 (1) #-470 (1)
SP	25	#-226 (1) 229 (1) #-230 (1) #-233 (1) #-239 (1) #-247 (1) #-250 (1) #-252 (1) #-274 (1) #-279 (1) #-284 (1) #-285 (1) #-288 (1) #-293 (1) #-295 (1) #-305 (1) #-309 (1) #-346 (1) #-347 (1) #-456 (1) #-457 (1) #-495 (1) #-496 (1) #-514 (1) #-515 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.01	00:00:00.23
Command processing	877	00:00:00.30	00:00:00.83
Pass 1	635	00:00:01.59	00:00:03.56
Symbol table sort	0	00:00:00.08	00:00:00.08
Pass 2	21	00:00:00.38	00:00:00.72
Symbol table output	2	00:00:00.02	00:00:00.09
Psect synopsis output	2	00:00:00.01	00:00:00.03
Cross-reference output	7	00:00:00.22	00:00:00.34
Assembler run totals	1568	00:00:02.61	00:00:05.88

The working set limit was 2400 pages.
71433 bytes (140 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 290 non-local and 30 local symbols.
527 source lines were read in Pass 1, producing 35 object records in Pass 2.

48 pages of virtual memory were used to define 21 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	5
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	16

360 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:LOAD.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DI
```

Table of contents

(1) 75 LOAD MASSBUS ADAPTER MAP REGISTERS
(1) 120 LOAD UNIBUS ADAPTER MAP REGISTERS
(1) 197 GET PFN FROM INVALID PTE

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element LODMAP.MAR
0000     2 ; *1     6-FEB-1986 15:19:25 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element LODMAP.MAR
0000     4         .TITLE  LODMAP  *** LODMAP load adapter map registers
0000     5         .IDENT  /05-02/
0000     6         .LIST   MEB
0000     7         .NLIST  CND
0000     8
0000     9 ;
0000    10 ;*****************************************************************************
0000    11 ;*                                                                           *
0000    12 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    14 ;*                                                                           *
0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    20 ;*  TRANSFERRED.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    24 ;*  CORPORATION.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    28 ;*                                                                           *
0000    29 ;*****************************************************************************
0000    30 ;
0000    31 ; D. N. CUTLER 1-NOV-77
0000    32 ;
0000    33 ; LOAD MBA MAP REGISTERS
0000    34 ;
0000    35 ; MODIFIED BY:
0000    36 ;
0000    37 ;       ADAPTED TO DIAGNOSTIC SUPERVISOR ENVIRONMENT
0000    38 ;
0000    39 ; LOAD MBA AND UBA MAP REGISTERS
0000    40 ;               Dave Butenhof   15-may-1980, Version 5.4
0000    41 ;       02      Modify VMS V2.0 source to Supervisor Environment
0000    42 ;
0000    43 ;       V04     RLR-TS04        Robert L. Rappaport     7-DEC-1979
0000    44 ;               Added an alternate entrypoint to IOC$LOADUBAMAP to
0000    45 ;               support byte aligned UNIBUS DMA devices
0000    46 ;
0000    47 ;       V03     NPKCOMET        N. KRONENBERG   11-JUN-1979
0000    48 ;               RETURNED IOC$LOADUBAMAP TO MODULE LOADMREG.
0000    49 ;
0000    50 ;       V02     NPKCOMET        N. KRONENBERG   1-FEB-1979
0000    51 ;               REMOVED IOC$LOADUBAMAP TO MODULE LIOSUB.
0000    52 ;
```

```
                        0000     54 ;
                        0000     55 ; MACRO LIBRARY CALLS
                        0000     56 ;
                        0000     57
                        0000     58
                        0000     62
                        0000     63          $CRBDEF                             ;DEFINE CRB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,CRB$L_WQFL,  CRB$L_WQFL:
00000004                0000                         .BLKL   1
                        0004                 .IIF    NB,CRB$L_WQBL,  CRB$L_WQBL:
00000008                0004                         .BLKL   1
                        0008                 .IIF    NB,CRB$W_SIZE,  CRB$W_SIZE:
0000000A                0008                         .BLKW   1
                        000A                 .IIF    NB,CRB$B_TYPE,  CRB$B_TYPE:
0000000B                000A                         .BLKB   1
0000000C                000B                         .BLKB   1
                        000C                 .IIF    NB,CRB$W_REFC,  CRB$W_REFC:
0000000E                000C                         .BLKW   1
                        000E                 .IIF    NB,CRB$B_MASK,  CRB$B_MASK:
0000000F                000E                         .BLKB   1
00000010                000F                         .BLKB   1
                        0010                 .IIF    NB,CRB$L_LINK,  CRB$L_LINK:
00000014                0010                         .BLKL   1
                        0014                 .IIF    NB,CRB$L_INTD,  CRB$L_INTD:
00000038                0014                         .BLKL   9
                        0038                 .IIF    NB,CRB$C_LENGTH,        CRB$C_LENGTH:
                        0038                 .IIF    NB,CRB$K_LENGTH,        CRB$K_LENGTH:
                        0038                 .IIF    NB,CRB$L_INTD2, CRB$L_INTD2:
0000005C                0038                         .BLKL   9
                        0000                 .RESTORE
                        0000     64          $MBADEF                             ;DEFINE MBA REGISTER OFFSET DEFINITIONS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                005C                 .=0
                        0000                 .RESTORE
                        0000     65          $PTEDEF                             ;DEFINE PAGE TABLE ENTRY FIELDS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                005C                 .=0
                        0000                 .RESTORE
                        0000     66          $UBADEF                             ;DEFINE UCB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                005C                 .=0
                        0000                 .RESTORE
                        0000     67          $UCBDEF                             ;DEFINE UCB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                005C                 .=0
                        0000                 .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                        0000                 .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004                0000                         .BLKL   1
                        0004                 .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                        0004                 .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008                0004                         .BLKL   1
                        0008                 .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
```

```
                    0000000A  0008                          .BLKW   1
                              000A          .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                    0000000B  000A                          .BLKB   1
                              000B          .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                    0000000C  000B                          .BLKB   1
                              000C          .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                              000C          .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                              000C          .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                    0000000D  000C                          .BLKB   1
                    00000010  000D                          .BLKB   3
                              0010          .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                              0010          .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
                    00000014  0010                          .BLKL   1
                              0014          .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                              0014          .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                              0014          .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
                    00000016  0014                          .BLKW   1
                              0016          .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
                    00000018  0016                          .BLKW   1
                              0018          .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                              0018          .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
                    0000001A  0018                          .BLKW   1
                              001A          .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                              001A          .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                    0000001C  001A                          .BLKW   1
                              001C          .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                    00000020  001C                          .BLKL   1
                              0020          .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                    00000024  0020                          .BLKL   1
                              0024          .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                    00000028  0024                          .BLKL   1
                              0028          .IIF    NB,UCB$L_PID,   UCB$L_PID:
                    0000002C  0028                          .BLKL   1
                              002C          .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
                    00000030  002C                          .BLKL   1
                              0030          .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
                    00000034  0030                          .BLKL   1
                              0034          .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
                    00000038  0034                          .BLKL   1
                              0038          .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
                    00000039  0038                          .BLKB   1
                              0039          .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
                    0000003A  0039                          .BLKB   1
                              003A          .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
                    0000003C  003A                          .BLKW   1
                              003C          .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                              003C          .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                              003C          .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
                    0000003D  003C                          .BLKB   1
                              003D          .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                              003D          .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
                    0000003E  003D                          .BLKB   1
                              003E          .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                              003E          .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
                    0000003F  003E                          .BLKB   1
                              003F          .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
                    00000040  003F                          .BLKB   1
```

```
                        0040            .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                                  .BLKL   1
                        0044            .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                                  .BLKL   1
                        0048            .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                                  .BLKW   1
                        004A            .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                        004A            .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                                  .BLKB   1
                        004B            .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                                  .BLKB   1
                        004C            .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                  .BLKL   1
                        0050            .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                  .BLKW   1
                        0052            .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                        0052            .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                  .BLKB   1
                        0053            .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                  .BLKB   1
                        0054            .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                  .BLKL   1
                        0058            .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
                        005A            .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
                        005C            .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                        005C            .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                  .BLKL   1
                        0060            .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                                  .BLKL   1
                        0064            .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                        0064            .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                                  .BLKL   1
                        0068            .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                                  .BLKL   1
                        006C            .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                                  .BLKW   1
                        006E            .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                                  .BLKW   1
                        0070            .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                                  .BLKB   1
                        0071            .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                                  .BLKB   1
                        0072            .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                                  .BLKW   1
                        0074            .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                        0074            .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074          .       = 0
                        0000            .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                                  .BLKW   1
00000074  0002          .       = 116
                        0074            .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                                  .BLKL   1
                        0078            .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                                  .BLKL   1
                        007C            .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
```

```
00000080  007C                                  .BLKL   1
          0080                  .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                                  .BLKL   1
          0090                  .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
          0090                  .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                  . = 116
          0074                  .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                                  .BLKB   1
          0075                  .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                                  .BLKB   1
          0076                  .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                                  .BLKB   1
          0077                  .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                                  .BLKB   1
          0078                  .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                                  .BLKL   1
0000007E  007C                                  .BLKW   1
          007E                  .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                                  .BLKW   1
          0080                  .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                                  .BLKL   1
00000080  0084                  . = 128
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                                  .BLKW   1
          008A                  .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                                  .BLKW   1
          008C                  .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C                  .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                  .BLKW   1
          008E                  .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                  .BLKW   1
          0090                  .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                  .BLKW   1
          0092                  .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                  .BLKW   1
          0094                  .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                  .BLKB   1
          0095                  .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                  .BLKB   1
          0096                  .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                  .BLKW   1
00000096  0098                  . = 150
00000098  0096                                  .BLKW   1
          0098                  .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                  .BLKL   1
          00A0                  .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
```

```
000000A4  00A0                          .BLKL   1
          00A4          .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                          .BLKW   1
          00A6          .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                          .BLKB   1
000000A8  00A7                          .BLKB   1
00000074  00A8          . = 116
00000078  0074                          .BLKL   1
0000007C  0078                          .BLKL   1
00000080  007C                          .BLKL   1
00000084  0080                          .BLKL   1
00000088  0084                          .BLKL   1
0000008C  0088                          .BLKL   1
          008C          .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                          .BLKL   1
00000094  0090                          .BLKL   1
00000098  0094                          .BLKL   1
0000009C  0098                          .BLKL   1
          009C          .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                          .BLKB   1
          009D          .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                          .BLKB   1
          009E          .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                          .BLKB   1
000000A0  009F                          .BLKB   1
          00A0          .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                          .BLKB   1
          00A1          .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                          .BLKB   1
          00A2          .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                          .BLKB   1
000000A4  00A3                          .BLKB   1
          00A4          .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                          .BLKB   1
          00A5          .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                          .BLKW   1
000000A8  00A7                          .BLKB   1
          00A8          .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                          .BLKL   1
000000B0  00AC                          .BLKL   1
000000B4  00B0                          .BLKL   1
000000B8  00B4                          .BLKL   1
          00B8          .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                          .BLKL   1
          00BC          .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC          .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC          . = 116
          0074          .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                          .BLKL   1
          0078          .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                          .BLKW   1
00000080  007E                          .BLKW   1
          0000          .RESTORE
          0000     68   $VECDEF                         ;DEFINE CRB TRANSFER VECTOR OFFSETS
          0000          .SAVE   LOCAL_BLOCK
```

```
                    0000                .PSECT  $ABS$,ABS
          00000000  00BC                .=0
                    0000                .IIF    NB,VEC$Q_DISPATCH,      VEC$Q_DISPATCH:
          00000008  0000                        .BLKQ   1
                    0008                .IIF    NB,VEC$L_IDB,   VEC$L_IDB:
          0000000C  0008                        .BLKL   1
                    000C                .IIF    NB,VEC$L_INITIAL,       VEC$L_INITIAL:
          00000010  000C                        .BLKL   1
                    0010                .IIF    NB,VEC$W_MAPREG,        VEC$W_MAPREG:
          00000012  0010                        .BLKW   1
                    0012                .IIF    NB,VEC$B_NUMREG,        VEC$B_NUMREG:
          00000013  0012                        .BLKB   1
                    0013                .IIF    NB,VEC$B_DATAPATH,      VEC$B_DATAPATH:
          00000014  0013                        .BLKB   1
                    0014                .IIF    NB,VEC$L_ADP,   VEC$L_ADP:
          00000018  0014                        .BLKL   1
                    0018                .IIF    NB,VEC$L_UNITINIT,      VEC$L_UNITINIT:
          0000001C  0018                        .BLKL   1
                    001C                .IIF    NB,VEC$L_START, VEC$L_START:
          00000020  001C                        .BLKL   1
                    0020                .IIF    NB,VEC$L_UNITDISC,      VEC$L_UNITDISC:
          00000024  0020                        .BLKL   1
                    0024                .IIF    NB,VEC$C_LENGTH,        VEC$C_LENGTH:
                    0024                .IIF    NB,VEC$K_LENGTH,        VEC$K_LENGTH:
                    0000                .RESTORE
```

```
                        00000000     73          .PSECT  SEP, SHR, EXE, WRT, LONG
                            0000     74
                            0000     75          .SBTTL  LOAD MASSBUS ADAPTER MAP REGISTERS
                            0000     76 ;+
                            0000     77 ; IOC$LOADMBAMAP - LOAD MASSBUS ADAPTER MAP REGISTERS
                            0000     78 ;
                            0000     79 ; THIS ROUTINE IS CALLED TO LOAD THE MASSBUS ADAPTER MAP REGISTERS, THE
                            0000     80 ; BYTE COUNT REGISTER, AND THE VIRTUAL ADDRESS REGISTER.
                            0000     81 ;
                            0000     82 ; INPUTS:
                            0000     83 ;
                            0000     84 ;       R4 = ADDRESS OF MBA CONFIGURATION STATUS REGISTER.
                            0000     85 ;       R5 = UCB ADDRESS OF UNIT TRANSFER IS TO OCCUR ON.
                            0000     86 ;
                            0000     87 ; OUTPUTS:
                            0000     88 ;
                            0000     89 ;       THE TRANSFER BYTE COUNT, STARTING PAGE OFFSET, AND ADDRESS OF THE
                            0000     90 ;       PAGE TABLE ENTIRES THAT DESCRIBE THE TRANSFER ARE RETRIEVED FROM
                            0000     91 ;       THE SPECIFED UCB AND USED TO LOAD THE MBA BYTE COUNT, VIRTUAL ADDRESS,
                            0000     92 ;       AND MAP REGISTERS. ONE ADDITIONAL MAP REGISTER IS LOADED AS INVALID
                            0000     93 ;       TO STOP THE TRANSFER IF A HARDWARE FAILURE SHOULD OCCUR.
                            0000     94 ;
                            0000     95 ;       R3 IS PRESERVED ACROSS CALL.
                            0000     96 ;-
                            0000     97
                            0000     98 IOC$LOADMBAMAP::                         ;LOAD MASSBUS ADAPTER MAP REGISTERS
                     53     DD   0000     99          PUSHL   R3                  ;SAVE REGISTERS
                52   6E A5   3C   0002    100          MOVZWL  UCB$W_BCNT(R5),R2   ;GET TRANSFER BYTE COUNT
                10 A4   52   CE   0006    101          MNEGL   R2,MBA$L_BCR(R4)    ;LOAD BYTE COUNT REGISTER
                51   6C A5   3C   000A    102          MOVZWL  UCB$W_BOFF(R5),R1   ;GET BYTE OFFSET IN PAGE
                0C A4   51   D0   000E    103          MOVL    R1,MBA$L_VAR(R4)    ;LOAD STARTING VIRTUAL ADDRESS
                0C A4   51   D0   0012    104          MOVL    R1,MBA$L_VAR(R4)    ;*****TEMP UNTIL MBA ECO ******
             52   01FF C241   9E   0016    105          MOVAB   ^X1FF(R2)[R1],R2    ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
          52   52   F7 8F   78   001C    106          ASHL    #-9,R2,R2           ;CALCULATE NUMBER OF MAP REGISTERS TO LOAD
       51   0800 C4   DE   0021    107          MOVAL   MBA$L_MAP(R4),R1    ;GET ADDRESS OF MBA MAP REGISTERS
          50   68 A5   D0   0026    108          MOVL    UCB$L_SVAPTE(R5),R0 ;GET ADDRESS OF PAGE TABLE
             81   80   D0   002A    109 10$:     MOVL    (R0)+,(R1)+         ;LOAD MAP REGISTER
                  09   18   002D    110          BGEQ    30$                 ;IF GEQ PTE INVALID
                F8 52   F5   002F    111 20$:     SOBGTR  R2,10$              ;ANY MORE TO LOAD?
                  61   D4   0032    112          CLRL    (R1)                ;LOAD INVALID MAP ENTRY
                53   8E   D0   0034    113          MOVL    (SP)+,R3            ;RESTORE REGISTER
                       05   0037    114          RSB                         ;
          53   FC A0   D0   0038    115 30$:     MOVL    -4(R0),R3           ;GET THE PTE (NOT FROM MAP REGISTER!)
                009E   30   003C    116          BSBW    IOC$PTETOPFN        ;GET PFN FROM INVALID PTE
FC A1   53   80000000 8F   C9   003F    117          BISL3   #^X80000000,R3,-4(R1) ;AND LOAD THE MAP REGISTER
                  E5   11   0048    118          BRB     20$                 ;
```

```
                                004A   120          .SBTTL  LOAD UNIBUS ADAPTER MAP REGISTERS
                                004A   121 ;+
                                004A   122 ; IOC$LOADUBAMAP  - LOAD UNIBUS ADAPTER MAP REGISTERS
                                004A   123 ; IOC$LOADUBAMAPA - LOAD UNIBUS ADAPTER MAP REGISTERS ALTERNATE ENTRY FOR
                                004A   124 ;       BYTE ALIGNED UNIBUS DMA DEVICES WHICH NEVER WISH TO SET THE BYTE
                                004A   125 ;       OFFSET BIT IN MAP REGISTERS.  IN ALL OTHER RESPECTS THESE TWO
                                004A   126 ;       ENTRYPOINTS PRODUCE IDENTICAL RESULTS.
                                004A   127 ;
                                004A   128 ; THIS ROUTINE IS CALLED TO LOAD THE UNIBUS ADAPTER MAP REGISTERS.
                                004A   129 ;
                                004A   130 ; INPUTS:
                                004A   131 ;
                                004A   132 ;       R5 = UCB ADDRESS OF UNIT TRANSFER IS TO OCCUR ON.
                                004A   133 ;
                                004A   134 ;       IT IS ASSUMED THAT THE DATAPATH AND MAP REGISTERS HAVE BEEN PREVIOUSLY
                                004A   135 ;       ASSIGNED.
                                004A   136 ;
                                004A   137 ; OUTPUTS:
                                004A   138 ;
                                004A   139 ;       EACH MAP REGISTER IS LOADED WITH THE APPROPRIATE PAGE FRAME NUMBER
                                004A   140 ;       MERGED WITH THE DATAPATH DESIGNATOR AND BYTE OFFSET BIT. ONE ADDITIONAL
                                004A   141 ;       MAP REGISTER IS LOADED AS INVALID TO STOP THE TRANSFER IF A HARDWARE
                                004A   142 ;       FAILURE SHOULD OCCUR.
                                004A   143 ;
                                004A   144 ;       R3 IS PRESERVED ACROSS CALL.
                                004A   145 ;-
                                004A   146
                                004A   147          .ENABL  LSB
                                004A   148 IOC$LOADUBAMAPA::                      ;LOAD UNIBUS ADAPTER MAP REGISTERS - ALTERNATE
                                004A   149                                       ; HERE WE DUPLICATE THE CODE IN THE OTHER ENTRY
                                004A   150                                       ;  EXCEPT THAT WE DO NOT CHECK WHETHER THE BYTE
                                004A   151                                       ;  OFFSET IS ODD.  INSTEAD WE BRANCH DIRECTLY
                                004A   152                                       ;   PAST THE SETTING OF THE BYTE OFFSET BIT.
          7E    53    7D      004A   153          MOVQ    R3,-(SP)                 ;SAVE REGISTERS
        51    6C A5    3C      004D   154          MOVZWL  UCB$W_BOFF(R5),R1        ;GET BYTE OFFSET IN PAGE
        52    6E A5    3C      0051   155          MOVZWL  UCB$W_BCNT(R5),R2        ;GET TRANSFER BYTE COUNT
        53    20 A5    D0      0055   156          MOVL    UCB$L_CRB(R5),R3         ;GET ADDRESS OF CRB
                00    EF      0059   157          EXTZV   #VEC$V_DATAPATH,-        ;GET DATAPATH
                05            005B   158                  #VEC$S_DATAPATH,-        ; NUMBER
        54    27 A3           005C   159                  CRB$L_INTD+VEC$B_DATAPATH(R3),R4
                1B    11      005F   160          BRB     10$                      ; BRANCH AROUND TO JOIN COMMON CODE
                              0061   161 IOC$LOADUBAMAP::                         ;LOAD UNIBUS ADAPTER MAP REGISTERS
          7E    53    7D      0061   162          MOVQ    R3,-(SP)                 ;SAVE REGISTERS
        51    6C A5    3C      0064   163          MOVZWL  UCB$W_BOFF(R5),R1        ;GET BYTE OFFSET IN PAGE
        52    6E A5    3C      0068   164          MOVZWL  UCB$W_BCNT(R5),R2        ;GET TRANSFER BYTE COUNT
        53    20 A5    D0      006C   165          MOVL    UCB$L_CRB(R5),R3         ;GET ADDRESS OF CRB
                00    EF      0070   166          EXTZV   #VEC$V_DATAPATH,-        ;GET DATAPATH
                05            0072   167                  #VEC$S_DATAPATH,-        ; NUMBER
        54    27 A3           0073   168                  CRB$L_INTD+VEC$B_DATAPATH(R3),R4
              03 51    E9      0076   169          BLBC    R1,10$                   ;IF LBC WORD ALIGNED TRANSFER
          54    10    88      0079   170          BISB    #^X10,R4                 ;SET BYTE OFFSET BIT
    54    0400 8F    A8      007C   171 10$:     BISW    #^X400,R4                ;MERGE VALID WITH BYTE OFFSET AND DATAPATH
                05    E1      0081   172          BBC     #VEC$V_LWAE,-            ;BRANCH IF LONGWORD ACCESS NOT ENABLED
           03 27 A3           0083   173                  CRB$L_INTD+VEC$B_DATAPATH(R3),15$
          54    20    88      0086   174          BISB    #^X20,R4                 ;ELSE SET LWAE FOR MAP REG
  52    01FF C241    9E      0089   175 15$:     MOVAB   ^X1FF(R2)[R1],R2         ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
52    52    F7 8F    78      008F   176          ASHL    #-9,R2,R2                ;CALCULATE NUMBER OF MAP REGISTERS TO LOAD
```

```
                26 A3    52    91   0094   177          CMPB    R2,CRB$L_INTD+VEC$B_NUMREG(R3) ;ENOUGH MAP REGISTERS ASSIGNED?
                         2C    1E   0098   178          BGEQU   40$                            ;IF GEQU NO
                51   28 B3    D0   009A   179          MOVL    @CRB$L_INTD+VEC$L_ADP(R3),R1 ;GET ADDRESS OF CONFIGURATION REGISTER
                         00    EF   009E   180          EXTZV   #VEC$V_MAPREG,-                ;GET STARTING REGISTER
                         0F         00A0   181                  #VEC$S_MAPREG,-
                50   24 A3         00A1   182                  CRB$L_INTD+VEC$W_MAPREG(R3),R0
             51   0800 C140    DE   00A4   183          MOVAL   UBA$L_MAP(R1)[R0],R1     ;GET ADDRESS OF FIRST MAP REGISTER TO LOAD
                50   68 A5    D0   00AA   184          MOVL    UCB$L_SVAPTE(R5),R0      ;GET ADDRESS OF PAGE TABLE
                   53   80    D0   00AE   185 20$:     MOVL    (R0)+,R3                 ;GET NEXT PAGE TABLE ENTRY
                        02    19   00B1   186          BLSS    30$                      ;IF LSS VALID PAGE TABLE ENTRY
                        28    10   00B3   187          BSBB    IOC$PTETOPFN             ;GET PFN FROM INVALID PTE
          53   0B   15    54    F0   00B5   188 30$:     INSV    R4,#21,#11,R3            ;INSERT VALID, BYTE OFFSET, AND DATAPATH
                   81    53    D0   00BA   189          MOVL    R3,(R1)+                 ;LOAD UBA MAP REGISTER
                      EE 52    F5   00BD   190          SOBGTR  R2,20$                   ;ANY MORE TO LOAD?
                         61    D4   00C0   191          CLRL    (R1)                     ;LOAD INVALID MAP ENTRY
                   53   8E    7D   00C2   192          MOVQ    (SP)+,R3                 ;RESTORE REGISTERS
                         05    00C5   193          RSB                              ;
                               00C6   194 40$:     BUG_CHECK UBMAPEXCED,FATAL       ;UNIBUS MAP REGISTER ALLOCATION EXCEEDED
                00000000'9F    16   00C6                JSB     @#BUG$CHECK
2C 44 45 43 58 45 50 41 4D 42 55 00' 00CC                .ASCIC  "UBMAPEXCED,FATAL"
                   4C 41 54 41 46   00D8
                               10   00CC
                                    00DD   195          .DSABL  LSB
```

```
                        00DD   197          .SBTTL  GET PFN FROM INVALID PTE
                        00DD   198 ;+
                        00DD   199 ; IOC$PTETOPFN - GET PFN FROM INVALID PTE
                        00DD   200 ;
                        00DD   201 ; THIS ROUTINE IS CALLED TO RETURN THE PAGE FRAME NUMBER FROM A
                        00DD   202 ; PAGE TABLE ENTRY WHICH HAS ALREADY BEEN DETERMINED TO BE NOT VALID.
                        00DD   203 ;
                        00DD   204 ; INPUTS:
                        00DD   205 ;
                        00DD   206 ;       R3 = PAGE TABLE ENTRY
                        00DD   207 ;
                        00DD   208 ; OUTPUTS:
                        00DD   209 ;
                        00DD   210 ;       R3 = PAGE FRAME NUMBER AND MAY INCLUDE THE FOLLOWING FIELDS
                        00DD   211 ;            VALID BIT, MODIFY BIT, PROTECTION FIELD, OWNER FIELD
                        00DD   212 ;
                        00DD   213 ;       ALL OTHER REGISTERS PRESERVED
                        00DD   214 ;-
                        00DD   215
                        00DD   216          .ENABL  LSB
                        00DD   217 IOC$PTETOPFN::
        53   FB800000 8F   CA 00DD   218          BICL    #^C<PTE$M_TYP1 ! PTE$M_TYP0 !- ;PTE TYPE BITS
                        00E4   219                  PTE$M_GPTX>,R3          ;AND GPTX/PFN
         01 53   1A   E0 00E4   220          BBS     #PTE$V_TYP1,R3,20$      ;BRANCH IF BAD PTE FOR I/O
                     05 00E8   221 10$:      RSB
                        00E9   222 20$:      BUG_CHECK INVPTEFMT,FATAL       ;INVALID PAGE TABLE ENTRY FORMAT
                00000000'9F   16 00E9               JSB     @#BUG$CHECK
46 2C 54 4D 46 45 54 50 56 4E 49 00' 00EF           .ASCIC  "INVPTEFMT,FATAL"
                     4C 41 54 41   00FB
                     0F            00EF
                        00FF   223          .DSABL  LSB
                        00FF   224
                        00FF   225          .END
```

BUG$CHECK	********	X	02	UCB$C_TT_LENGTH	000000BC	D	UCB$W_CHARGE	0000004A	D						
CRB$B_MASK	0000000E	D		UCB$K_LENGTH	00000074	D	UCB$W_CYLINDERS	0000003E	D						
CRB$B_TYPE	0000000A	D		UCB$K_MB_LENGTH	00000090	D	UCB$W_DA	0000008C	D						
CRB$C_LENGTH	00000038	D		UCB$K_TT_LENGTH	000000BC	D	UCB$W_DC	0000008E	D						
CRB$K_LENGTH	00000038	D		UCB$L_AMB	00000054	D	UCB$W_DEVBUFSIZ	0000003A	D						
CRB$L_INTD	00000014	D		UCB$L_ASTQBL	00000010	D	UCB$W_DEVSTS	0000005A	D						
CRB$L_INTD2	00000038	D		UCB$L_ASTQFL	0000000C	D	UCB$W_DIRSEQ	00000088	D						
CRB$L_LINK	00000010	D		UCB$L_CPID	0000005C	D	UCB$W_DSTADDR	00000018	D						
CRB$L_WQBL	00000004	D		UCB$L_CRB	00000020	D	UCB$W_DX_BCR	000000A4	D						
CRB$L_WQFL	00000000	D		UCB$L_DDB	00000024	D	UCB$W_EC1	00000090	D						
CRB$W_REFC	0000000C	D		UCB$L_DEVCHAR	00000034	D	UCB$W_EC2	00000092	D						
CRB$W_SIZE	00000008	D		UCB$L_DEVDEPEND	0000003C	D	UCB$W_ERRCNT	00000072	D						
IOC$LOADMBAMAP	00000000	RG D	02	UCB$L_DPC	00000080	D	UCB$W_FUNC	0000007E	D						
IOC$LOADUBAMAP	00000061	RG D	02	UCB$L_DUETIM	0000005C	D	UCB$W_MB_SEED	00000000	D						
IOC$LOADUBAMAPA	0000004A	RG D	02	UCB$L_DX_BFPNT	0000009C	D	UCB$W_MSGCNT	00000016	D						
IOC$PTETOPFN	000000DD	RG D	02	UCB$L_DX_BUF	00000098	D	UCB$W_MSGMAX	00000014	D						
MBA$L_BCR =	00000010	D		UCB$L_DX_RXDB	000000A0	D	UCB$W_NT_CHAN	0000007C	D						
MBA$L_MAP =	00000800	D		UCB$L_EMB	00000078	D	UCB$W_OFFSET	0000008A	D						
MBA$L_VAR =	0000000C	D		UCB$L_FIRST	00000014	D	UCB$W_REFC	00000050	D						
PTE$M_GPTX =	003FFFFF	D		UCB$L_FPC	0000000C	D	UCB$W_SIZE	00000008	D						
PTE$M_TYP0 =	00400000	D		UCB$L_FQBL	00000004	D	UCB$W_SRCADDR	0000001A	D						
PTE$M_TYP1 =	04000000	D		UCB$L_FQFL	00000000	D	UCB$W_STS	00000058	D						
PTE$V_TYP1 =	0000001A	D		UCB$L_FR3	00000010	D	UCB$W_TT_DESIZE	000000A5	D						
SIZ... =	00000001	D		UCB$L_FR4	00000014	D	UCB$W_UNIT	00000048	D						
UBA$L_MAP =	00000800	D		UCB$L_IOQBL	00000044	D	UCB$W_VPROT	0000001A	D						
UCB$B_AMOD	00000053	D		UCB$L_IOQFL	00000040	D	VEC$B_DATAPATH	00000013	D						
UCB$B_CEX	00000077	D		UCB$L_IRP	0000004C	D	VEC$B_NUMREG	00000012	D						
UCB$B_CM1	0000004A	D		UCB$L_LINK	0000002C	D	VEC$C_LENGTH	00000024	D						
UCB$B_CM2	0000004B	D		UCB$L_LOGADR	00000064	D	VEC$K_LENGTH	00000024	D						
UCB$B_DEVCLASS	00000038	D		UCB$L_MAXBLOCK	00000084	D	VEC$L_ADP	00000014	D						
UCB$B_DEVTYPE	00000039	D		UCB$L_MB_MBX	0000007C	D	VEC$L_IDB	00000008	D						
UCB$B_DIPL	00000052	D		UCB$L_MB_PORT	0000008C	D	VEC$L_INITIAL	0000000C	D						
UCB$B_DX_SCTCNT	000000A6	D		UCB$L_MB_RAST	00000078	D	VEC$L_START	0000001C	D						
UCB$B_ERTCNT	00000070	D		UCB$L_MB_SHB	00000080	D	VEC$L_UNITDISC	00000020	D						
UCB$B_ERTMAX	00000071	D		UCB$L_MB_WAST	00000074	D	VEC$L_UNITINIT	00000018	D						
UCB$B_FEX	00000076	D		UCB$L_MB_WIOQBL	00000088	D	VEC$Q_DISPATCH	00000000	D						
UCB$B_FIPL	0000000B	D		UCB$L_MB_WIOQFL	00000084	D	VEC$S_DATAPATH =	00000005	D						
UCB$B_LOCSRV	0000003C	D		UCB$L_MEDIA	0000008C	D	VEC$S_MAPREG =	0000000F	D						
UCB$B_OFFNDX	00000094	D		UCB$L_NT_DATSSB	00000074	D	VEC$V_DATAPATH =	00000000	D						
UCB$B_OFFRTC	00000095	D		UCB$L_NT_INTSSB	00000078	D	VEC$V_LWAE =	00000005	D						
UCB$B_REMSRV	0000003D	D		UCB$L_OPCNT	00000060	D	VEC$V_MAPREG =	00000000	D						
UCB$B_SECTORS	0000003C	D		UCB$L_OWNUIC	0000001C	D	VEC$W_MAPREG	00000010	D						
UCB$B_SLAVE	00000074	D		UCB$L_PID	00000028	D									
UCB$B_SPR	00000075	D		UCB$L_RQBL	00000004	D									
UCB$B_STATE	00000052	D		UCB$L_RQFL	00000000	D									
UCB$B_TRACKS	0000003D	D		UCB$L_SVAPTE	00000068	D									
UCB$B_TT_CRFILL	0000009D	D		UCB$L_SVPN	00000064	D									
UCB$B_TT_DECRF	000000A1	D		UCB$L_TT_DECHAR	000000A8	D									
UCB$B_TT_DELFF	000000A2	D		UCB$L_TT_RDUE	0000008C	D									
UCB$B_TT_DESPEE	000000A0	D		UCB$L_TT_RTIMOU	000000B8	D									
UCB$B_TT_DETYPE	000000A4	D		UCB$L_VCB	00000030	D									
UCB$B_TT_LFFILL	0000009E	D		UCB$T_PARTNER	0000000C	D									
UCB$B_TT_SPEED	0000009C	D		UCB$W_BCNT	0000006E	D									
UCB$B_TYPE	0000000A	D		UCB$W_BCR	00000096	D									
UCB$B_VERTSZ	0000003F	D		UCB$W_BOFF	0000006C	D									
UCB$C_LENGTH	00000074	D		UCB$W_BUFQUO	00000018	D									
UCB$C_MB_LENGTH	00000090	D		UCB$W_BYTESTOGO	0000003E	D									

```
                                        +----------------+
                                        ! Psect synopsis !
                                        +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           000000BC  (  188.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
SEP                             000000FF  (  255.)  02 (  2.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD    WRT NOVEC LONG
```

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL          VALUE        DEFINITION      REFERENCES...
------          -----        ----------      -------------
BUG$CHECK       00000000-XR                  194    (1)      222    (1)
CRB$L_INTD      00000014                     159    (1)      168    (1)      173    (1)    #-177    (1)    #-179    (1)
                                             182    (1)
IOC$LOADMBAMAP  00000000-R   98     (1)
IOC$LOADUBAMAP  00000061-R   161    (1)
IOC$LOADUBAMAPA 0000004A-R   148    (1)
IOC$PTETOPFN    000000DD-R   217    (1)    #-116    (1)    #-187    (1)
MBA$L_BCR      =00000010                   #-101    (1)
MBA$L_MAP      =00000800                     107    (1)
MBA$L_VAR      =0000000C                   #-103    (1)    #-104    (1)
PTE$M_GPTX     =003FFFFF                   #-219    (1)
PTE$M_TYP0     =00400000                   #-218    (1)
PTE$M_TYP1     =04000000                   #-218    (1)
PTE$V_TYP1     =0000001A                   #-220    (1)
UBA$L_MAP      =00000800                     183    (1)
UCB$L_CRB       00000020                   #-156    (1)    #-165    (1)
UCB$L_SVAPTE    00000068                   #-108    (1)    #-184    (1)
UCB$W_BCNT      0000006E                   #-100    (1)    #-155    (1)    #-164    (1)
UCB$W_BOFF      0000006C                   #-102    (1)    #-154    (1)    #-163    (1)
VEC$B_DATAPATH  00000013                     159    (1)      168    (1)      173    (1)
VEC$B_NUMREG    00000012                   #-177    (1)
VEC$L_ADP       00000014                   #-179    (1)
VEC$S_DATAPATH =00000005                   #-158    (1)    #-167    (1)
VEC$S_MAPREG   =0000000F                   #-181    (1)
VEC$V_DATAPATH =00000000                   #-157    (1)    #-166    (1)
VEC$V_LWAE     =00000005                   #-172    (1)
VEC$V_MAPREG   =00000000                   #-180    (1)
VEC$W_MAPREG    00000010                     182    (1)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$CRBDEF         1           63     (1)      63     (1)
$DEFINI         1           63     (1)      63     (1)      64     (1)      65     (1)      66     (1)      67     (1)
                                            68     (1)
$MBADEF         5           64     (1)      64     (1)
$PTEDEF         3           65     (1)      65     (1)
$UBADEF         6           66     (1)      66     (1)
$UCBDEF         10          67     (1)      67     (1)
$VECDEF         2           68     (1)      68     (1)
BUG_CHECK       1           194    (1)      194    (1)      222    (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ASHL	0078	106	(1)	176	(1)										
BBC	00E1	172	(1)												
BBS	00E0	220	(1)												
BGEQ	0018	110	(1)												
BGEQU	001E	178	(1)												
BICL	00CA	218	(1)												
BISB	0088	170	(1)	174	(1)										
BISL3	00C9	117	(1)												
BISW	00A8	171	(1)												
BLBC	00E9	169	(1)												
BLSS	0019	186	(1)												
BRB	0011	118	(1)	160	(1)										
BSBB	0010	187	(1)												
BSBW	0030	116	(1)												
CLRL	00D4	112	(1)	191	(1)										
CMPB	0091	177	(1)												
EXTZV	00EF	157	(1)	166	(1)	180	(1)								
INSV	00F0	188	(1)												
JSB	0016	194	(1)	222	(1)										
MNEGL	00CE	101	(1)												
MOVAB	009E	105	(1)	175	(1)										
MOVAL	00DE	107	(1)	183	(1)										
MOVL	00D0	103	(1)	104	(1)	108	(1)	109	(1)	113	(1)	115	(1)	156	(1)
		165	(1)	179	(1)	184	(1)	185	(1)	189	(1)				
MOVQ	007D	153	(1)	162	(1)	192	(1)								
MOVZWL	003C	100	(1)	102	(1)	154	(1)	155	(1)	163	(1)	164	(1)		
PUSHL	00DD	99	(1)												
RSB	0005	114	(1)	193	(1)	221	(1)								
SOBGTR	00F5	111	(1)	190	(1)										

Directives Cross Reference

DIRECTIVE	REFERENCES...													
.ASCIC	194	(1)	222	(1)										
.DSABL	195	(1)	223	(1)										
.ENABL	147	(1)	216	(1)										
.END	225	(1)												
.ENDM	194	(1)	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)
.IDENT	5	(1)												
.LIST	6	(1)	61	(1)	70	(1)	71	(1)						
.NLIST	59	(1)	60	(1)	69	(1)	7	(1)						
.NOCROSS	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)		
.PAGE	119	(1)	196	(1)	53	(1)	72	(1)						
.PSECT	73	(1)												
.RESTORE	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)		
.SAVE	63	(1)	64	(1)	65	(1)	66	(1)	67	(1)	68	(1)		
.SBTTL	120	(1)	197	(1)	75	(1)								
.TITLE	4	(1)												

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
R0	7	#-108 (1) #-109 (1) #-115 (1) #-182 (1) 183 (1) #-184 (1) #-185 (1)
R1	17	#-102 (1) #-103 (1) #-104 (1) 105 (1) #-107 (1) #-109 (1) #-112 (1) #-117 (1) #-154 (1) #-163 (1) #-169 (1) 175 (1) #-179 (1) 183 (1) #-189 (1) #-191 (1)
R2	15	#-100 (1) #-101 (1) 105 (1) #-106 (1) #-111 (1) #-155 (1) #-164 (1) 175 (1) #-176 (1) #-177 (1) #-190 (1)
R3	20	#-113 (1) #-115 (1) #-117 (1) #-153 (1) #-156 (1) 159 (1) #-162 (1) #-165 (1) 168 (1) 173 (1) #-177 (1) #-179 (1) 182 (1) #-185 (1) 188 (1) #-189 (1) #-192 (1) #-219 (1) 220 (1) #-99 (1)
R4	10	#-101 (1) #-103 (1) #-104 (1) 107 (1) #-159 (1) #-168 (1) #-170 (1) #-171 (1) #-174 (1) #-188 (1)
R5	10	#-100 (1) #-102 (1) #-108 (1) #-154 (1) #-155 (1) #-156 (1) #-163 (1) #-164 (1) #-165 (1) #-184 (1)
SP	4	#-113 (1) #-153 (1) #-162 (1) #-192 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.21
Command processing	875	00:00:00.23	00:00:00.72
Pass 1	481	00:00:01.48	00:00:02.58
Symbol table sort	0	00:00:00.13	00:00:00.12
Pass 2	42	00:00:00.40	00:00:00.79
Symbol table output	2	00:00:00.02	00:00:00.02
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	5	00:00:00.08	00:00:00.10
Assembler run totals	1430	00:00:02.37	00:00:04.54

The working set limit was 2400 pages.
82994 bytes (163 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 481 non-local and 10 local symbols.
225 source lines were read in Pass 1, producing 39 object records in Pass 2.
41 pages of virtual memory were used to define 16 macros.

```
                                  +----------------------------+
                                  ! Macro library statistics !
                                  +----------------------------+

Macro library name                 Macros defined
------------------                 --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              3
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                1
SYS$COMMON:[SYSLIB]LIB.MLB;2                           3
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       5
TOTALS (all libraries)                                12

709 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:LODMAP.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 109 Declarations
(1) 150 DSX$Escape Routine
(1) 195 DSX$BgnSub Routine
(2) 301 DSX$EndSub Routine
(2) 491 DSX$CkLoop Routine
(3) 552 DSX$InLoop Routine

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element LOOP.MAR
0000     2 ; *1     6-FEB-1986 15:19:27 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element LOOP.MAR
0000     4         .Title  LOOP    *** LOOP Loop control
0000     5         .Ident  /07-19/
0000     6         .NoShow Conditionals
0000     7
0000     8 ;
0000     9 ; Copyright (c) 1977, 1981, 1982
0000    10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    11 ;
0000    12 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    13 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    14 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    15 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    16 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    17 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    18 ; REMAIN IN DEC.
0000    19 ;
0000    20 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    21 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    22 ; CORPORATION.
0000    23 ;
0000    24 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    26
0000    27 ;++
0000    28 ; FACILITY:
0000    29 ;       VAX DIAGNOSTIC SUPERVISOR.
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ; ENVIRONMENT:
0000    34 ;
0000    35 ; AUTHOR:
0000    36 ;       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    37 ;
```

```
0000     39 ; MODIFICATIONS:
0000     40 ;
0000     41 ;                 9-Mar-1978      Roger Riggs
0000     42 ;         01      Fixed DS$INLOOP and DS$CKLOOP
0000     43 ;         02      Fixed BGNSUB and ENDSUB to use .entry and .vector
0000     44 ;                 also changed names to DSX$...
0000     45 ;                 12-Jun-1979     Roger Riggs
0000     46 ;         03      Turn off ^O before typing final loop pass message
0000     47 ;                 Dave Butenhof, 1-Mar-1980
0000     48 ;         04      Changed all references to APT mailbox to use longword
0000     49 ;                 offsets.
0000     50 ;                 Roger Riggs, 18-Mar-1980
0000     51 ;         05      Changed loop on subtest to execute the subtest the
0000     52 ;                 specified number of times instead of the previous +1.
0000     53 ;                 also made post loop on subtest processing call DS_CLEANUP
0000     54 ;                 - Dave Butenhof, 22-Jun-1981, version 6.4
0000     55 ;         06      Fix edit 5: delete explicit setting of DS$V_DONFLG, which
0000     56 ;                 prevented DS_CLEANUP from actually executing the diagnostic's
0000     57 ;                 cleanup code.
0000     58 ;         07      Fix /subtest to work right with /test:n:m.  It will only loop
0000     59 ;                 on the last test selected.
0000     60 ;
0000     61 ;         08      - Jack Stansbury, 21-Oct-1981, Version 6.-
0000     62 ;                 Added calls to QA$MAIN for the QA enhancement.
0000     63 ;                 Also added .LIBRARY statements for $DS and $DIAG.
0000     64 ;                 Also changed the case of some of the error messages.
0000     65 ;
0000     66 ;         09      - Dave Butenhof, 02-Nov-1981, version 6.-
0000     67 ;                 Fix truncation errors throughout.
0000     68 ;
0000     69 ;         10      - Jack Stansbury, 21-Nov-1981, Version 6.5
0000     70 ;                 Commented out a call to QA_MAIN because it appears that it
0000     71 ;                 will not be needed for the 6.5 DS version.
0000     72 ;
0000     73 ;         11      - Jack Stansbury, 13-Jan-1982, Version 6.6
0000     74 ;                 Added comments galore to make things easier to understand.
0000     75 ;
0000     76 ;         12      - Jack Stansbury, 14-Jan-1982, Version 6.6
0000     77 ;                 Added a call to the Summary Code routine in the EndSub routine.
0000     78 ;                 The diagnostic's summary code will now be executed when
0000     79 ;                 the requested number of loops on subtest have been done.
0000     80 ;                 This is more consistent with the EndPass routine.
0000     81 ;
0000     82 ;         13      - Jack Stansbury, 15-Jan-1982, Version 6.6
0000     83 ;                 Changed the FMTLOOPDON message to make it more like the
0000     84 ;                 EndPass message. Also added "Test x.y" to the message.
0000     85 ;
0000     86 ;         14      - Jack Stansbury, 20-Jan-1982, Version 6.6
0000     87 ;                 Changed the code in the EndSub routine to not print the
0000     88 ;                 header line when the Loop_On_Subtest QA check routine is
0000     89 ;                 executing. This is for QA.
0000     90 ;
0000     91 ;         15      - Jack Stansbury, 29-Mar-1982, Version 6.?
0000     92 ;                 Added some typecoding.
0000     93 ;
0000     94 ;         16      Marion Baggett, 30-mar-1982,    Version 6.7
0000     95 ;                 Changed all $DS_PRINT statements to the new form $PRINT.
```

```
0000     96 ;
0000     97 ;       17      Dave Butenhof, 08-Apr-1982, version 6.7
0000     98 ;               Fix truncation error.
0000     99 ;
0000    100 ;       18      Jack Stansbury, 13-April-1982, Version 6.7
0000    101 ;               Fix two truncation errors.
0000    102 ;
0000    103 ;       19      Peter Green, 18-JUL-1983, Version 6.12
0000    104 ;               Changed $DS_INLOOP to return correct status
0000    105 ;               returns, (DS$_ERROR or DS$_NORMAL)
0000    106 ;
0000    107 ;--
```

```
                        0000    109          .SBTTL  Declarations
                        0000    110 ;
                        0000    111 ; INCLUDE FILES:
                        0000    112 ;
                        0000    113          .LIBRARY        /SYS$LIBRARY:LIB/       ;                       [16]
                        0000    114          .LIBRARY        /$DS/                   ;                       [08]
                        0000    115          .LIBRARY        /$DIAG/                 ;                       [08]
                        0000    116
                        0000    117 ;
                        0000    118 ; MACROS:
                        0000    119 ;
                        0000    120
                        0000    121 ;
                        0000    122 ; EQUATED SYMBOLS:
                        0000    123 ;
                00000003 0000   124          BPTCODE = 03            ; BREAKPOINT OPCODE
                        0000    125          $DS_DSADEF
                        0000    126          $DS_DSDEF
                        0000    127          $DS_HDRDEF
                        0000    128          APTDEF
                        0000    129          DSFDEF
                        0000    130          DSQA                    ; DSQA$K constant definitions           [08]
                        0000    131          DS_QADEFS               ; Other QA$K constant definitions       [14]
                        0000    132          $DS_TypeDef             ; Define the TypeCodes                  [15]
```

```
                                00000000   134          .PSECT  Data, Shr, NoExe, NoWrt, Byte
                                    0000   135
                                    0000   136 ;
                                    0000   137 ; OWN STORAGE:
                                    0000   138 ;
                                    0000   139
                                    0000   140          MODNAM  LOOP
                     50 4F 4F 4C 00'0000       $MODULE:        .ASCIC  \LOOP\
                                    04 0000
                                    0005   141
                                    0005   142 FMTSEQERR:
6E 65 75 71 65 53 20 3F 3F 2F 21 00'0005   143          .ASCIC  "!/?? Sequencing error. Should be in test !UW.!UW,"-    ;       [08]
20 2E 72 6F 72 72 65 20 67 6E 69 63 0011
6E 69 20 65 62 20 64 6C 75 6F 68 53 001D
55 21 2E 57 55 21 20 74 73 65 74 20 0029
74 20 64 65 68 63 61 65 72 20 2C 57 0035
2E 57 55 21 2E 57 55 21 20 74 73 65 0041
                           2F 21 004D
                              49 0005
                                    004F   144                  " reached test !UW.!UW.!/"                              ;       [08]
                                    004F   145 FMTLOOPDON:
66 6F 20 64 6E 45 20 2E 2E 2F 21 00'004F   146          .ASCIC  "!/.. End of run, !UW error!%S detected, !UL loop!%S,"- ;       [13]
72 65 20 57 55 21 20 2C 6E 75 72 20 005B
63 65 74 65 64 20 53 25 21 72 6F 72 0067
6F 6F 6C 20 4C 55 21 20 2C 64 65 74 0073
                  2C 53 25 21 70 007F
55 21 2E 4C 55 21 20 74 73 65 74 20 0084   147                  " test !UL.!UL,!/"-                                     ;       [13]
                           2F 21 2C 4C 0090
21 20 73 69 20 65 6D 69 74 20 20 20 0094   148                  "   time is !%D!/"                                      ;       [13]
                           2F 21 44 25 00A0
                                 54 004F
```

```
                       00A4   150          .SBTTL  DSX$Escape Routine
                   00000000   151          .PSECT  Code, Shr, Exe, NoWrt, Byte
                       0000   152 ;++
                       0000   153 ; FUNCTIONAL DESCRIPTION:
                       0000   154 ;
                       0000   155 ;
                       0000   156 ; CALLING SEQUENCE:
                       0000   157 ;
                       0000   158 ;       $ESCAPE ARG
                       0000   159 ;
                       0000   160 ; INPUT PARAMETERS:
                       0000   161 ;
                       0000   162 ;       AP      = Address of where to escape
                       0000   163 ;
                       0000   164 ; IMPLICIT INPUTS:
                       0000   165 ;
                       0000   166 ;       DS$GL_FLAGS <DS$V_ERRFLG>
                       0000   167 ;
                       0000   168 ; OUTPUT PARAMETERS:
                       0000   169 ;
                       0000   170 ;       NONE
                       0000   171 ;
                       0000   172 ; IMPLICIT OUTPUTS:
                       0000   173 ;
                       0000   174 ;       NONE
                       0000   175 ;
                       0000   176 ; COMPLETION CODES:
                       0000   177 ;
                       0000   178 ;       NONE
                       0000   179 ;
                       0000   180 ; SIDE EFFECTS:
                       0000   181 ;
                       0000   182 ;       NONE
                       0000   183 ;
                       0000   184 ;--
```

```
                        0000  0000   186          .ENTRY  DSX$ESCAPE,^M<>
                              0002   187
              00000000'EF   16  0002   188          Jsb     L^KB_CHECK                ; ^C check.                                  [18]
                              0008   189          Br_If_Not_ErrFlg  REscapeX       ; Exit if no error has occurred              [18]
          00000000'EF   04    E1  0008             BBC     #DS$V_ErrFlg, -           ; If bit not set, branch                     [29]
                        04        000F                       L^DS$GL_Flags, REscapeX          ;
            10 AD       5C    D0  0010   190          MOVL    AP, 16(FP)                ; Replace return PC with ESCAPE address [11]
                              0014   191
                              0014   192 RESCAPEX:
                              04  0014   193          RET                               ; Return to diagnostic
```

```
0015   195          .SBTTL  DSX$BgnSub Routine
0015   196 ;++
0015   197 ; FUNCTIONAL DESCRIPTION:
0015   198 ;
0015   199 ;       This routine insures that the diagnostic program is sequencing
0015   200 ;       through its subtests in numerical order.  If an error is detected
0015   201 ;       the operator is notified and the command mode is entered as if
0015   202 ;       "Halt On Error" was selected. This means that the operator may
0015   203 ;       examine the general purpose registers and continue at the next
0015   204 ;       instruction of the program, if desired.
0015   205 ;
0015   206 ; CALLING SEQUENCE:
0015   207 ;
0015   208 ;       $BGNSUB
0015   209 ;               CALLG   $$$, a#DS$BGNSUB
0015   210 ;
0015   211 ; INPUT PARAMETERS:
0015   212 ;
0015   213 ;       4(AP)   = CURRENT TEST NUMBER (PROGRAM)
0015   214 ;       8(AP)   = NEW SUBTEST NUMBER (PROGRAM)
0015   215 ;
0015   216 ; IMPLICIT INPUTS:
0015   217 ;
0015   218 ;       DSA$GL_TESTNO           = CURRENT TEST NUMBER (SUPERVISOR).
0015   219 ;       DS$GL_FLAGS <DS$V_SUBT> = SUBTEST FLAG.
0015   220 ;
0015   221 ; OUTPUT PARAMETERS:
0015   222 ;
0015   223 ;       NONE
0015   224 ;
0015   225 ; IMPLICIT OUTPUTS:
0015   226 ;
0015   227 ;       DSA$GL_SUBTNO               = SUBTEST NUMBER.
0015   228 ;       DS$GA_LOOPADR               = LOOP ADDRESS.
0015   229 ;       DS$GL_FLAGS <DS$V_ERRFLG> = ERROR FLAG.
0015   230 ;       DS$GL_FLAGS <DS$V_SUBT>   = SUBTEST FLAG.
0015   231 ;
0015   232 ; COMPLETION CODES:
0015   233 ;
0015   234 ;       NONE
0015   235 ;
0015   236 ; SIDE EFFECTS:
0015   237 ;
0015   238 ;       NONE
0015   239 ;
0015   240 ;--
```

```
                        0000  0015   242          .ENTRY  DSX$BGNSUB,^M<>
                              0017   243
                              0017   244          QA_MAIN Loop_DSX$BgnSub          ; Call QA$Main                          [08]
                    09   DD   0017                PUSHL   #DSQA$K_Loop_DSX$BgnSub
      00000000'9F   01   FB   0019                CALLS   #1, @#QA$MAIN
                              0020   245
      00000000'EF   0E   E2   0020   246          BBSS    #DS$V_SUBT, -            ; Set subtest flag. If it was already   [11]
                    1A        0027   247                    L^DS$GL_FLAGS, 20$     ; set, print sequence error.            [11]
                              0028   248
                              0028   249          Clear_ErrFlg                     ; Clear the error flag.                 [18]
      00000000'EF   04   E4   0028                BBSC    #DS$V_ErrFlg, -          ; Branch if ERRFLG bit is set to        [29]
                    00        002F                        L^DS$GL_Flags, 30000$ ; ...                                    [29]
                              0030       30000$:                                  ; ... here. Clear bit regardless.       [29]
                              0030   250
                              0030   251          ;+
                              0030   252          ; Save the PC that is stored in the call frame. This PC is for the      [11]
                              0030   253          ; first instruction after the $DS_BgnSub routine. Save it in            [11]
                              0030   254          ; DS$GA_LoopAdr so that the EndSub routine can branch back to this      [11]
                              0030   255          ; address if it needs to (because of looping on subtest, etc.).         [11]
                              0030   256          ;-
                              0030   257
   00000000'EF   10 AD   D0   0030   258          MOVL    16 (FP), -               ; Save return address of this subtest.  [11]
                              0038   259                    L^DS$GA_LOOPADR        ; LoopAdr is the address of where to    [11]
                              0038   260                                           ; loop back to from the EndSub routine. [11]
                              0038   261
   04 AC   0000FE50'EF   D1   0038   262          CMPL    L^DSA$GL_TESTNO, 4 (AP) ; Compare DS's TestNo with the progs.   [11]
                    62   13   0040   263          BEQL    50$                      ; Branch if they are the same (okay).   [11]
```

```
                        0042   265          ;+
                        0042   266          ; Sequencing error. The Supervisor's TestNo does not agree with the     [11]
                        0042   267          ; test number that the diagnostic passed to this routine (i.e., 16(AP)) [11]
                        0042   268          ; OR else the SubT flag was not set (indicating that two BgnSubs        [11]
                        0042   269          ; occurred without an intervening EndSub.                               [11]
                        0042   270          ;-
                        0042   271
                        0042   272 20$:      $Print  #DS$K_Type_Sequence_Error, -    ; There is a sequence error     [15]
                        0042   273                  #DS$K_PrintF, -                 ; ... use printf                [15]
                        0042   274                  L^FMTSEQERR, -                  ; ... the format string         [15]
                        0042   275                  L^DSA$GL_TESTNO,-               ; ... rest of parameters        [15]
                        0042   276                  L^DSA$GL_SUBTNO,-
                        0042   277                  4(AP), 8(AP)
          08 AC    DD   0042                       PUSHL   8(AP)
          04 AC    DD   0045                       PUSHL   4(AP)
0000FE4C'EF    DD   0048                       PUSHL   L^DSA$GL_SUBTNO
0000FE50'EF    DD   004E                       PUSHL   L^DSA$GL_TESTNO
00000005'EF    9F   0054                       PUSHAB  L^FMTSEQERR
          7E   01   9A   005A                  movzbl  #DS$K_PrintF, -(sp)
          7E   19   98   005D                  cvtbl   #DS$K_Type_Sequence_Error, -(sp)
00000000'GF    07   FB   0060                  CALLS   #$$N+2, G^DSX$PRINT
                        0067   278
                        0067   279          Set_Halt                                ; Set the HALT flag to halt on error.   [18]
0000FE00'EF    00   E2   0067                BBSS    #DSA$V_Halt, -                  ; Branch if Halt bit is set to          [29]
                00   006E                    L^DSA$GL_Flags, 30001$                ; ...
                        006F       30001$:                                         ; ... here. Set bit regardless.         [29]
00000000'EF    07   E3   006F   280          BBCS    #DS$V_CMDFLG, -                 ; Set the command flag. If not set,     [11]
                11   0076   281                    L^DS$GL_FLAGS, 40$              ; branch (okay).                        [09]
                        0077   282          ERRSUP_S                                ; The command flag was already set.     [09]
     00000000'EF    DF   0077                       PUSHAL  $MODULE
                00   DD   007D                       PUSHL   #0
                01   DD   007F                       PUSHL   #$ER
   00000000'EF    03   FB   0081                     CALLS   #3, DS_ERRSUP
                        0088   283
00000000'EF   10 BD   DE   0088   284 40$:   MOVAL   @16(FP), L^DS$AA_BPTADDR; Save return PC.                        [09]
00000000'EF   10 BD   90   0090   285        MOVB    @16(FP), L^DS$AB_BPTINST; Save the opcode located there.        [09]
          10 BD   03   90   0098   286        MOVB    #BPTCODE, @16(FP)       ; Store a breakpoint instruction at the [11]
                        009C   287                                            ; return location.                      [11]
0000FE50'EF   04 AC   D0   009C   288        MOVL    4(AP), DSA$GL_TESTNO    ; Reset the test number.                [11]
```

```
                        00A4   290          ;+
                        00A4   291          ; Fall through to here if there was a sequencing error, or branch to    [11]
                        00A4   292          ; here if there was not a sequence error.                               [11]
                        00A4   293          ;-
                        00A4   294
    0000FE4C'EF   08 AC  D0   00A4   295 50$:     MOVL    8 (AP), DSA$GL_SUBTNO   ; Reset the subtest nunmber.          [11]
          00000000'EF    16   00AC   296          Jsb     L^KB_CHECK              ; Check for TTY input.                 [18]
                        00B2   297
                        00B2   298 RBGNSUBX:
                     04 00B2   299          RET                                 ; Return to next subtest.              [11]
```

```
00B3    301          .SBTTL  DSX$EndSub Routine
00B3    302 ;++
00B3    303 ; FUNCTIONAL DESCRIPTION:
00B3    304 ;
00B3    305 ;       The end of subtest routine checks both test and subtest numbers
00B3    306 ;       to assure that the program sequence is correct.
00B3    307 ;
00B3    308 ; CALLING SEQUENCE:
00B3    309 ;
00B3    310 ;       $ENDSUB
00B3    311 ;               CALLG   $$$, a*DS$ENDSUB
00B3    312 ;
00B3    313 ; INPUT PARAMETERS:
00B3    314 ;
00B3    315 ;       4(AP)   = CURRENT TEST NUMBER (PROGRAM).
00B3    316 ;       8(AP)   = CURRENT SUBTEST NUMBER (PROGRAM).
00B3    317 ;
00B3    318 ; IMPLICIT INPUTS:
00B3    319 ;
00B3    320 ;       DSA$GL_TESTNO               = TEST NUMBER (SUPERVISOR).
00B3    321 ;       DSA$GL_SUBTNO               = SUBTEST NUMBER (SUPERVISOR).
00B3    322 ;       DSA$GL_FLAGS <DSA$V_LOOP>   = LOOP FLAG.
00B3    323 ;       DS$GL_SUBTEST               = SUBTEST NUMBER FOR LOOPING.
00B3    324 ;       DS$GA_LOOPADR               = LOOP ADDRESS.
00B3    325 ;       DS$GL_FLAGS <DS$V_SUBT>     = SUBTEST FLAG.
00B3    326 ;       DS$GL_FLAGS <DS$V_ERRFLG>   = ERROR FLAG.
00B3    327 ;
00B3    328 ; OUTPUT PARAMETERS:
00B3    329 ;
00B3    330 ;       NONE
00B3    331 ;
00B3    332 ; IMPLICIT OUTPUTS:
00B3    333 ;
00B3    334 ;       DS$GL_FLAGS<DS$V_SUBT>   = SUBTEST FLAG.
00B3    335 ;
00B3    336 ; COMPLETION CODES:
00B3    337 ;
00B3    338 ;       NONE
00B3    339 ;
00B3    340 ; SIDE EFFECTS:
00B3    341 ;
00B3    342 ;       NONE
00B3    343 ;
00B3    344 ;--
```

```
                          0000  00B3   346          .ENTRY  DSX$ENDSUB,^M<>
                                00B5   347
                                00B5   348          QA_MAIN Loop_DSX$EndSub          ; Call QA$Main                          [08]
                             0A   DD   00B5          PUSHL   #DSQA$K_Loop_DSX$EndSub
                00000000'9F  01   FB   00B7          CALLS   #1, a#QA$MAIN
                     00000000'EF  16   00BE   349    Jsb     L^KB_CHECK               ; Check for TTY input.                  [18]
                                00C4   350
                                00C4   351          ;+
                                00C4   352          ; Compare the Supervisor's test and subtest numbers with those of       [11]
                                00C4   353          ; the diagnostics (i.e., those that are passed). If they are not the    [11]
                                00C4   354          ; same, it is a Sequence Error.                                         [11]
                                00C4   355          ;-
                                00C4   356
          04 AC  0000FE50'EF    D1   00C4   357    CMPL    L^DSA$GL_TESTNO, 4(AP)   ; Compare DS's TestNo with the progs.   [11]
                             0A   12   00CC   358    BNEQ    3$                       ; Branch if they are not equal.         [11]
          08 AC  0000FE4C'EF    D1   00CE   359    CMPL    L^DSA$GL_SUBTNO, 8(AP)   ; Compare DS's SubtNo with the progs.   [11]
                             03   13   00D6   360    BEQL    4$                       ; Branch if no error                    [11]
                                00D8   361
                           00DE   31   00D8   362 3$: BRW     30$                      ; Branch if error in test or subtest
                                00DB   363
                                00DB   364          ;+
                                00DB   365          ; Check various flags.                                                  [11]
                                00DB   366          ;-
                                00DB   367
                                00DB   368 4$:                                       ; TestNo and SubtNo are both okay.      [11]
                00000000'EF  0E   E5   00DB   369    BBCC    #DS$V_SUBT, -            ; Branch if there was no BgnSub.       [11]
                             50        00E2   370              L^DS$GL_FLAGS, 17$     ;                                       [09]
                0000FE00'EF  02   E1   00E3   371    BBC     #DSA$V_LOOP, -           ; If Loop on Error flag not set,       [11]
                             08        00EA   372              L^DSA$GL_FLAGS, 5$     ; branch.                               [11]
                                00EB   373          Br_If_ErrFlg    10$              ; If error occurred, branch.           [18]
                00000000'EF  04   E0   00EB         BBS     #DS$V_ErrFlg, -          ;  If bit set, branch                  [29]
                             2F        00F2              L^DS$GL_Flags, 10$         ;                                      [29]
```

```
                        00F3   375          ;+
                        00F3   376          ; Come to here if the Loop on Error flag is not set OR                  [11]
                        00F3   377          ;                 if no errors have occurred.                           [11]
                        00F3   378          ;-
                        00F3   379
00000000'EF   08 AC   D1 00F3   380 5$:     CMPL    8 (AP), L^DS$GL_SUBTEST ; Is this the subtest to loop on?       [11]
                 35   12 00FB   381          BNEQ    15$                     ; If not, branch.                       [11]
00000000'EF   04 AC   D1 00FD   382          CMPL    4 (AP), L^DS$GL_LSTTEST ; Are we on the correct test?           [09]
                 2B   12 0105   383          BNEQ    15$                     ; If not on correct test, exit.         [07]
                        0107   384
                        0107   385          ;+
                        0107   386          ; Add one to the PassNo count. Thus, keep track of the number of times  [11]
                        0107   387          ; this subtest has been executed. Execute it only 'x' times, where x is [11]
                        0107   388          ; /PASS:x.                                                              [11]
                        0107   389          ;-
                        0107   390
   50   0000FE54'EF   DE 0107   391          MOVAL   L^DSA$GL_PASSNO, R0     ; Point to pass number.
0C 60   0000FE08'EF   F3 010E   392          AOBLEQ  L^DSA$GL_PASSES, (R0), 10$ ; Count and branch if not done.
                        0116   393
                        0116   394          ;+
                        0116   395          ; Are all done the requested number of passes. However, if the pass     [11]
                        0116   396          ; request is 0, keep chugging forever.                                  [11]
                        0116   397          ;-
                        0116   398
         0000FE08'EF   D5 0116   399          TSTL    L^DSA$GL_PASSES         ; If pass count 0
                     04   13 011C   400          BEQL    10$                     ; Continue forever
                     60   D7 011E   401          DECL    (R0)                    ; Make true count of passes
                     14   11 0120   402          BRB     20$                     ; Branch to done code
```

```
                                0122   404          ;+
                                0122   405          ; Come to here (if the Loop on Error flag is not set AND                [11]
                                0122   406          ;                 if an Error has occurred)                             [11]
                                0122   407          ;              OR (if the pass count has not reached its limit).        [11]
                                0122   408          ; Cause the loop to occur. That is, loop back to the instruction        [11]
                                0122   409          ; following the CALL to BgnSub.                                         [11]
                                0122   410          ;-
                                0122   411
      10 AD   00000000'EF   D0  0122   412 10$:     MOVL    L^DS$GA_LOOPADR, 16 (FP); Set PC to the saved LoopAdr.          [09]
                                012A   413          Set_SubT                        ; Set the SubT flag.                    [18]
         00000000'EF   0E   E2  012A               BBSS    #DS$V_Subt, -           ; Branch if SUBT bit is set to          [29]
                          00    0131                 L^DS$GL_Flags, 30002$ ; ...                                           [29]
                                0132         30002$:                               ; ... here. Set bit regardless.         [29]
                                0132   414
                             04 0132   415 15$:     RET                             ; Return with changed PC.               [11]
                                0133   416
                       0083  31 0133   417 17$:     BRW     30$                     ; Branch on                             [09]
```

```
                                0136   419          ;+
                                0136   420          ; Loop on subtest w/pass count exhausted. This should mirror the        [11]
                                0136   421          ; DSX$EndPass routine.                                                  [11]
                                0136   422          ;-
                                0136   423
     50   00000000'EF   DE      0136   424 20$:     MOVAL   L^DS$GL_FLAGS,R0        ; Point to flags.                      [09]
   60   01   07   01   F0      013D   425          INSV    #1, #DS$V_CMDFLG, #1, - ; Set Command mode.                    [06]
                                0142   426            (R0)                  ;                                              [11]
   60   01   10   00   F0      0142   427          INSV    #0, #DS$V_CTRLO, #1, -  ; Clear Cntrl-O flag.                  [06]
                                0147   428            (R0)                  ;                                              [11]
                                0147   429
                                0147   430          Br_If_Not_QA    23$             ; If not running under QA, branch      [18]
    0000FE00'EF   0F   E1       0147                BBC     #DSA$V_QA, -                    ; If bit not set, branch       [29]
                  0A            014E                  L^DSA$GL_Flags, 23$           ;                                      [29]
                  03   E1       014F   431          BBC     #QA$K_LOOP_ON_SUBTEST, -; If not running Loop on Subtest check, [14]
                                0151   432            L^QA$AOB_CHECK_STATE,-; branch.                                      [14]
      02 00000000'EF            0151   433            23$                   ;                                              [14]
                  2D   11       0157   434          BRB     24$                     ; Otherwise, do not print ending line. [14]
                                0159   435
                                0159   436 23$:     $PRINT  #DS$K_TYPE_PROGRAM_END,- ; PRINT "End of run ...".             [16]
                                0159   437            #DS$K_PRINTF,-        ; Use printf.                                  [16]
                                0159   438            L^FMTLOOPDON, -       ;                                              [14]
                                0159   439            L^DS$GL_ERRCNT, -     ; Current number of errors.                    [09]
                                0159   440            L^DSA$GL_PASSNO, -    ; Number of loops done.                        [11]
                                0159   441            L^DSA$GL_TESTNO, -    ; The test looped on.                          [13]
                                0159   442            L^DSA$GL_SUBTNO, -    ; The subtest looped on.                       [13]
                                0159   443            #0                    ; Print current time.                          [11]
                  00   DD       0159                PUSHL   #0
        0000FE4C'EF   DD       015B                PUSHL   L^DSA$GL_SUBTNO
        0000FE50'EF   DD       0161                PUSHL   L^DSA$GL_TESTNO
        0000FE54'EF   DD       0167                PUSHL   L^DSA$GL_PASSNO
        00000000'EF   DD       016D                PUSHL   L^DS$GL_ERRCNT
        0000004F'EF   9F       0173                PUSHAB  L^FMTLOOPDON
           7E   01   9A       0179                movzbl  #DS$K_PRINTF, -(sp)
           7E   10   98       017C                cvtbl   #DS$K_TYPE_PROGRAM_END, -(sp)
00000000'GF   08   FB       017F                CALLS   #$$N+2, G^DSX$PRINT
                                0186   444
     00000000'EF   16           0186   445 24$:     Jsb     L^INIT_CONTEXT          ; Reset stacks and processor mode.     [17]
                                018C   446          $DS_SUMMARY_S                 ; Exexcute the diagnostics summary code.[12]
00000000'9F   00   FB       018C                CALLS   #0, a#DS$SUMMARY
     00000000'EF   16           0193   447          Jsb     L^DS_CLEANUP            ; Execute users cleanup code.
                                0199   448
                                0199   449          Clear_CtrlO                   ; Clear the Control-O flag. This makes [18]
00000000'EF   10   E4       0199                BBSC    #DS$V_CtrlO, -          ; Branch if CTRLO bit is set to
                  00            01A0                  L^DS$GL_Flags, 30003$ ; ...
                                01A1       30003$:                              ; ... here. Clear bit regardless.
                                01A1   450                                      ; ... sure it is off after the summary. [18]
0000FE40'EF   0A   D0       01A1   451          MOVL    #APM$_DONE, -           ; Tell APT we're all done.             [11]
                                01A8   452            L^DSA$GL_MSGTYP       ;                                              [11]
                                01A8   453          Br_If_Not_QA    25$             ; If not running under QA, branch.     [18]
0000FE00'EF   0F   E1       01A8                BBC     #DSA$V_QA, -                    ; If bit not set, branch       [29]
                  03            01AF                  L^DSA$GL_Flags, 25$           ;                                      [29]
              FE4D'   31       01B0   454          BRW     VRRESTART               ; If running QA, branch to the START   [11]
                                01B3   455                                      ; routine. This should only happen on  [11]
                                01B3   456                                      ; the Loop_On_Subtest check routine.   [11]
     00000000'EF   16           01B3   457 25$:     Jsb     L^Begin                 ; Clean up the world. Go back to CLI.  [17]
```

```
                                01B9    459          ;+
                                01B9    460          ; Got here because not in subtest OR test or subtest numbers did not match.
                                01B9    461          ;-
                                01B9    462
                                01B9    463 30$:     $PRINT  #DS$K_TYPE_SEQUENCE_ERROR,-     ; PRINT "Sequencing error..."   [16]
                                01B9    464                    #DS$K_PRINTF,-           ; Use printf.                   [16]
                                01B9    465                    L^FMTSEQERR,-            ; ... the format string.        [18]
                                01B9    466                    L^DSA$GL_TESTNO,-        ; Current test number.          [11]
                                01B9    467                    L^DSA$GL_SUBTNO,-        ; Current subtest number.       [11]
                                01B9    468                    4(AP), 8(AP)             ; User's test and subtest number. [11]
                      08 AC     DD    01B9                     PUSHL   8(AP)
                      04 AC     DD    01BC                     PUSHL   4(AP)
              0000FE4C'EF       DD    01BF                     PUSHL   L^DSA$GL_SUBTNO
              0000FE50'EF       DD    01C5                     PUSHL   L^DSA$GL_TESTNO
              00000005'EF       9F    01CB                     PUSHAB  L^FMTSEQERR
                 7E     01     9A    01D1                     movzbl  #DS$K_PRINTF, -(sp)
                 7E     19     98    01D4                     cvtbl   #DS$K_TYPE_SEQUENCE_ERROR, -(sp)
          00000000'GF   07     FB    01D7                     CALLS   #$$N+2, G^DSX$PRINT
                                01DE    469
                                01DE    470          Set_Halt                             ; Set the Halt flag.            [18]
          0000FE00'EF   00     E2    01DE                     BBSS    #DSA$V_Halt, -       ; Branch if Halt bit is set to  [29]
                        00           01E5                       L^DSA$GL_Flags, 30004$       ; ...
                                01E6         30004$:                                      ; ... here. Set bit regardless. [29]
          00000000'EF   07     E3    01E6    471          BBCS    #DS$V_CMDFLG, -          ; Set command flag.             [11]
                        11           01ED    472                    L^DS$GL_FLAGS, 45$   ;                               [09]
                                01EE    473          ERRSUP_S                             ; Supervisor error: command flag set. [11]
              00000000'EF       DF    01EE                     PUSHAL  $MODULE
                        00      DD    01F4                     PUSHL   #0
                        02      DD    01F6                     PUSHL   #$ER
                00000000'EF     03    FB    01F8               CALLS   #3, DS_ERRSUP
                                01FF    474
                                01FF    475          ;+
                                01FF    476          ; Save the return PC in BptAddr. Save the opcode located at the return  [11]
                                01FF    477          ; PC location. Store a BPT instruction at that location. Reset the      [11]
                                01FF    478          ; test and subtest numbers. Return to the diagnostic, and execute the   [11]
                                01FF    479          ; breakpoint instruction, causing the Cli to take over.                 [11]
                                01FF    480          ;-
                                01FF    481
          00000000'EF   10 BD   DE    01FF    482 45$:     MOVAL   a16 (FP), -              ; Save return PC.               [09]
                                0207    483                    L^DS$AA_BPTADDR          ;                               [11]
          00000000'EF   10 BD   90    0207    484          MOVB    a16 (FP), -              ; Save the opcode located there. [09]
                                020F    485                    L^DS$AB_BPTINST          ;                               [11]
                   10 BD   03   90    020F    486          MOVB    #BPTCODE, a16 (FP)       ; Set a breakpoint at the return PC. [11]
          0000FE50'EF   04 AC   D0    0213    487          MOVL    4 (AP), L^DSA$GL_TESTNO  ; Set TestNo to the user's test number. [11]
          0000FE4C'EF   08 AC   D0    021B    488          MOVL    8 (AP), L^DSA$GL_SUBTNO  ; Set SubtNo to the user's subtest no. [11]
                        04      0223    489          RET                                  ; Return to the diagnostic.     [11]
```

```
0224    491          .SBTTL  DSX$CkLoop Routine
0224    492 ;++
0224    493 ; FUNCTIONAL DESCRIPTION:
0224    494 ;
0224    495 ;       Check for loop flag AND an error flag, and loop if both
0224    496 ;       conditions are met.  Also keep track of previous loop conditions
0224    497 ;       and modify the loop if a new error occurs.
0224    498 ;
0224    499 ; CALLING SEQUENCE:
0224    500 ;
0224    501 ;       $CKLOOP ADR
0224    502 ;               CALLG   ADR, a#DS$CKLOOP
0224    503 ;
0224    504 ; INPUT PARAMETERS:
0224    505 ;
0224    506 ;       AP      = LOOP ADDRESS
0224    507 ;
0224    508 ; IMPLICIT INPUTS:
0224    509 ;
0224    510 ;       DS$GA_CHKLPPC               = CHECK LOOP PC.
0224    511 ;       DSA$GL_FLAGS <DSA$V_LOOP>   = LOOP FLAG.
0224    512 ;       DS$GL_FLAGS <DS$V_ERRFLG>   = ERROR FLAG.
0224    513 ;
0224    514 ; OUTPUT PARAMETERS:
0224    515 ;
0224    516 ;       NONE
0224    517 ;
0224    518 ; IMPLICIT OUTPUTS:
0224    519 ;
0224    520 ;       NONE
0224    521 ;
0224    522 ; COMPLETION CODES:
0224    523 ;
0224    524 ;       NONE
0224    525 ;
0224    526 ; SIDE EFFECTS:
0224    527 ;
0224    528 ;       NONE
0224    529 ;
0224    530 ;--
```

```
                        0000  0224   532          .ENTRY  DSX$CKLOOP, ^M<>
                              0226   533
                              0226   534          QA_MAIN Loop_DSX$CkLoop          ; Call QaMain.                          [08]
                    0B   DD   0226                PUSHL   #DSQA$K_Loop_DSX$CkLoop
      00000000'9F   01   FB   0228                CALLS   #1, @#QA$MAIN
                              022F   535
          00000000'EF   16   022F   536          Jsb     L^KB_CHECK               ; Check for TTY input.                  [09]
      0000FE00'EF   02   E1   0235   537          BBC     #DSA$V_LOOP, -           ; If not looping on error, branch.      [11]
                    26        023C   538                    DSA$GL_FLAGS, RCKLOOPX
                              023D   539          Br_If_Not_ErrFlg  RCkLoopX       ; If no errors, branch.                 [18]
      00000000'EF   04   E1   023D                BBC     #DS$V_ErrFlg, -                  ; If bit not set, branch        [29]
                    1E        0244                        L^DS$GL_Flags, RCkLoopX                  ;
          00000000'EF   D5   0245   540          TSTL    L^DS$GA_CHKLPPC          ; Is this the first CHKLOOP after an    [09]
                              024B   541                                           ;   ERROR call?
                    0A   13   024B   542          BEQL    10$                      ; Branch if it is.                      [11]
   10 AD   00000000'EF   D1   024D   543          CMPL    L^DS$GA_CHKLPPC, 16(FP)  ; Is this the correct CHKLOOP.          [09]
                    0C   12   0255   544          BNEQ    RCKLOOPX                 ; Exit if not right call.               [11]
                              0257   545
   00000000'EF   10 AD   D0   0257   546 10$:     MOVL    16(FP), L^DS$GA_CHKLPPC  ; Save PC of this CHKLOOP.              [09]
            10 AD   5C   D0   025F   547          MOVL    AP, 16(FP)               ; Replace return PC with loop address.  [11]
                              0263   548
                              0263   549 RCKLOOPX:
                         04   0263   550          RET                              ; Return to the diagnostic.             [11]
```

```
0264    552          .SBTTL  DSX$InLoop Routine
0264    553 ;++
0264    554 ; FUNCTIONAL DESCRIPTION:
0264    555 ;
0264    556 ;       This routine returns successfully if the diagnostic has
0264    557 ;       encountered an error AND the loop flag is set.
0264    558 ;
0264    559 ; CALLING SEQUENCE:
0264    560 ;
0264    561 ;       DS$INLOOP()
0264    562 ;
0264    563 ; INPUT PARAMETERS:     NONE
0264    564 ;
0264    565 ; IMPLICIT INPUTS:
0264    566 ;
0264    567 ;       DS$V_ERR in DS$GL_FLAGS
0264    568 ;       DSA$V_LOOP in DSA$GL_FLAGS
0264    569 ;
0264    570 ; OUTPUT PARAMETERS:    NONE
0264    571 ;
0264    572 ; IMPLICIT OUTPUTS:     NONE
0264    573 ;
0264    574 ; COMPLETION CODES:
0264    575 ;
0264    576 ;       R0=1 if looping and error else 0
0264    577 ;
0264    578 ; SIDE EFFECTS:         NONE
0264    579 ;--
```

```
                     0000  0264   581          .ENTRY  DSX$INLOOP,^M<>
                           0266   582
          00000000'EF   16  0266   583          Jsb     L^KB_CHECK                  ; Check for TTY input.
     50   00660002 8F   D0  026C   584          MOVL    #DS$_ERROR,R0               ; CORRECT ERROR STATUS CODE             [19]
                           0273   585 ;        CLRL    R0                          ; Assume failure.                       [11]
                           0273   586          Br_If_Not_ErrFlg        10$        ; If error flag not set, branch.       [18]
00000000'EF    04   E1  0273                  BBC     #DS$V_ErrFlg, -             ; If bit not set, branch                [29]
                0F   E1  027A                    L^DS$GL_Flags, 10$                ;                                       [29]
0000FE00'EF    02   E1  027B   587          BBC     #DSA$V_LOOP, -              ; If not looping on error, branch.      [11]
                07       0282   588                    L^DSA$GL_FLAGS, 10$
     50   00660001 8F   D0  0283   589          MOVL    #DS$_NORMAL,R0              ; CORRECT NORMAL STATUS CODE            [19]
                           028A   590 ;        INCL    R0                          ; Error and LOOP on error set.          [11]
                           028A   591
                     04    028A   592 10$:     RET                                 ; Return to diagnostic.                 [11]
                           028B   593
                           028B   594          .END
```

```
$$N                             = 00000005     D
$ER                             = 00000003     D
$MODULE                           00000000 R   D   02
APC$_ABORT                      = 00000006     D
APC$_CONTINUE                   = 00000009     D
APC$_DUMP                       = 00000003     D
APC$_NOP                        = 00000000     D
APC$_START                      = 00000005     D
APC$_ZERO                       = 00000004     D
APM$_ABTDON                     = 00000006     D
APM$_DEVERR                     = 00000002     D
APM$_DONE                       = 0000000A     D
APM$_EXCEPT                     = 0000000B     D
APM$_HARDERR                    = 00000003     D
APM$_MORE                       = 00000008     D
APM$_NOMES                      = 00000000     D
APM$_PREPERR                    = 0000000C     D
APM$_PRGERR                     = 00000005     D
APM$_SOFTERR                    = 00000004     D
APM$_SPOOL                      = 00000009     D
APM$_SYSERR                     = 00000001     D
BEGIN                             ********   X   03
BIT...                          = 00000004     D
BPTCODE                         = 00000003     D
DS$AA_BPTADDR                     ********   X   03
DS$AB_BPTINST                     ********   X   03
DS$GA_CHKLPPC                     ********   X   03
DS$GA_LOOPADR                     ********   X   03
DS$GL_ERRCNT                      ********   X   03
DS$GL_FLAGS                       ********   X   03
DS$GL_LSTTEST                     ********   X   03
DS$GL_SUBTEST                     ********   X   03
DS$K_BBC                        = 0000001D   G D
DS$K_BBCC                       = 00000021   G D
DS$K_BBCCI                      = 00000023   G D
DS$K_BBCS                       = 0000001F   G D
DS$K_BBS                        = 0000001C   G D
DS$K_BBSC                       = 00000020   G D
DS$K_BBSS                       = 0000001E   G D
DS$K_BBSSI                      = 00000022   G D
DS$K_BCC_M                      = 0000001B   G D
DS$K_BCS_M                      = 0000001A   G D
DS$K_BEQLU_B                    = 00000005   G D
DS$K_BEQLU_M                    = 00000011   G D
DS$K_BEQL_B                     = 00000004   G D
DS$K_BEQL_M                     = 00000010   G D
DS$K_BERROR                     = 00000026   G D
DS$K_BGEQU_B                    = 00000007   G D
DS$K_BGEQU_M                    = 00000013   G D
DS$K_BGEQ_B                     = 00000006   G D
DS$K_BGEQ_M                     = 00000012   G D
DS$K_BGTRU_B                    = 00000009   G D
DS$K_BGTRU_M                    = 00000015   G D
DS$K_BGTR_B                     = 00000008   G D
DS$K_BGTR_M                     = 00000014   G D
DS$K_BLBC                       = 00000025   G D
DS$K_BLBS                       = 00000024   G D
DS$K_BLEQU_B                    = 00000003   G D
DS$K_BLEQU_M                    = 0000000F   G D
DS$K_BLEQ_B                     = 00000002   G D
DS$K_BLEQ_M                     = 0000000E   G D
DS$K_BLSSU_B                    = 00000001   G D
DS$K_BLSSU_M                    = 0000000D   G D
DS$K_BLSS_B                     = 00000000   G D
DS$K_BLSS_M                     = 0000000C   G D
DS$K_BNEQU_B                    = 0000000B   G D
DS$K_BNEQU_M                    = 00000017   G D
DS$K_BNEQ_B                     = 0000000A   G D
DS$K_BNEQ_M                     = 00000016   G D
DS$K_BNERROR                    = 00000027   G D
DS$K_BVC_M                      = 00000019   G D
DS$K_BVS_M                      = 00000018   G D
DS$K_DS_STARTING_LOCATION       = 00010000     D
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
DS$K_TYPE_ERRSOFT               = 00000007     D
DS$K_TYPE_ERRSUP                = 00000004     D
DS$K_TYPE_ERRSYS                = 00000005     D
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
```

```
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D
DS$K_TYPE_START_ERR             = 00000018    D
DS$K_TYPE_START_LIST            = 00000025    D
DS$K_TYPE_SUMMARY               = 0000000E    D
DS$K_TYPE_USER_PROMPT           = 00000002    D
DS$K_WARNING                    = 00000000    D
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_NI                         = 04000000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$SUMMARY                      ********    X   03
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DS$_AP_NORMAL_BREAK             = 00660150    D
DS$_ARITH                       = 006600D0    D
DS$_ASBE                        = 00660118    D
DS$_BADLINK                     = 006600F0    D
DS$_BADTYPE                     = 006600E8    D
DS$_BIIC                        = 00660120    D
DS$_CHME                        = 006600A8    D
DS$_CHMK                        = 006600E0    D
DS$_DEVNAME                     = 00660108    D
DS$_ERROR                       = 00660002    D
DS$_FHWE                        = 00660068    D
DS$_FRAGBUF                     = 00660080    D
DS$_ICBUSY                      = 006600C8    D
DS$_ICERR                       = 006600C0    D
DS$_IHWE                        = 00660060    D
DS$_ILLCHAR                     = 00660018    D
DS$_ILLPAGCNT                   = 00660078    D
DS$_ILLUNIT                     = 00660100    D
DS$_INIT_FAIL                   = 00660140    D
DS$_INSFMEM                     = 00660050    D
DS$_INVCPU                      = 00660130    D
DS$_IPL2HI                      = 006600B8    D
DS$_IVADDR                      = 00660040    D
DS$_IVVECT                      = 00660038    D
DS$_KRNLSTK                     = 00660090    D
DS$_LOGIC                       = 00660070    D
DS$_MCHK                        = 00660088    D
DS$_MEM_ALLOC_ERR               = 00660138    D
DS$_MMOFF                       = 00660058    D
DS$_NEEDUNIT                    = 006600F8    D
DS$_NODE                        = 00660128    D
DS$_NOPCS                       = 00660110    D
DS$_NORMAL                      = 00660001    D
DS$_NOSUPPORT                   = 006600B1    D
DS$_NOTALLOWMP                  = 00660148    D
DS$_NOTDON                      = 00660030    D
DS$_NOTIMP                      = 006600B0    D
DS$_NULLSTR                     = 00660010    D
DS$_OVERFLOW                    = 00660008    D
DS$_POWER                       = 00660098    D
DS$_PROGERR                     = 00660020    D
DS$_SEVERE                      = 00660004    D
DS$_TRANSL                      = 006600A0    D
DS$_TRUNCATE                    = 00660028    D
DS$_UNEXPINT                    = 006600D8    D
DS$_UNSUPPDEV                   = 00660158    D
DS$_VASFULL                     = 00660048    D
DS$_WARNING                     = 00660000    D
DSA$AL_APTMAIL                    0000FE00    D
```

```
DSA$AT_APTTXT                   0000FA00    D
DSA$GL_APTCOM                   0000FE04    D
DSA$GL_DEVLEN                   0000FE58    D
DSA$GL_ERRNO                    0000FE44    D
DSA$GL_EVENT                    0000FE48    D
DSA$GL_FLAGS                    0000FE00    D
DSA$GL_MSGTYP                   0000FE40    D
DSA$GL_PASSES                   0000FE08    D
DSA$GL_PASSNO                   0000FE54    D
DSA$GL_SECTNO                   0000FE10    D
DSA$GL_SID                      0000FE14    D
DSA$GL_SUBTNO                   0000FE4C    D
DSA$GL_TESTNO                   0000FE50    D
DSA$GL_UNITS                    0000FE0C    D
DSA$GQ_MSGPTR                   0000FE68    D
DSA$GT_DEVNAM                   0000FE5C    D
DSA$V_HALT                    = 00000000    D
DSA$V_LOOP                    = 00000002    D
DSA$V_QA                      = 0000000F    D
DSQA$K_DISPAT_AFTER_INIT      = 00000014    D
DSQA$K_DISPAT_BEFORE_INIT     = 00000013    D
DSQA$K_DISPAT_CALLTEST        = 00000015    D
DSQA$K_DISPAT_DONE_TESTS      = 00000018    D
DSQA$K_DISPAT_DSX$ENDPASS_BGN = 00000001    D
DSQA$K_DISPAT_DSX$ENDPASS_END = 00000002    D
DSQA$K_DISPAT_DSX$ENDPASS_MID = 00000000    D
DSQA$K_DISPAT_DS_CLEANUP_BGN  = 00000003    D
DSQA$K_DISPAT_DS_CLEANUP_END  = 00000004    D
DSQA$K_DUMMY_1                = 00000007    D
DSQA$K_ERROR_ERROR_BGN        = 00000006    D
DSQA$K_ERROR_ERROR_END        = 00000005    D
DSQA$K_KERNEL_INIT_CONTEXT    = 00000008    D
DSQA$K_LOOP_DSX$BGNSUB        = 00000009    D
DSQA$K_LOOP_DSX$CKLOOP        = 0000000B    D
DSQA$K_LOOP_DSX$ENDSUB        = 0000000A    D
DSQA$K_PARAM_DSX$GETADDRESS   = 0000000E    D
DSQA$K_PARAM_DSX$GETDATA      = 0000000C    D
DSQA$K_PARAM_DSX$GETLOGICAL   = 0000000F    D
DSQA$K_PARAM_DSX$GETSTRING    = 00000010    D
DSQA$K_PARAM_DSX$GETVIELD     = 0000000D    D
DSQA$K_QA_DSX$BRANCH          = 00000011    D
DSQA$K_START_BEFORE_HEADER    = 00000017    D
DSQA$K_START_VRRESTART        = 00000012    D
DSQA$K_START_VRSTART          = 00000016    D
DSX$BGNSUB                      00000015 RG D   03
DSX$CKLOOP                      00000224 RG D   03
DSX$ENDSUB                      000000B3 RG D   03
DSX$ESCAPE                      00000000 RG D   03
DSX$INLOOP                      00000264 RG D   03
DSX$PRINT                       ********   X   03
DS_CLEANUP                      ********   X   03
DS_ERRSUP                       ********   X   03
FALSE                         = 00000000     D
FMTLOOPDON                      0000004F R   D   02
FMTSEQERR                       00000005 R   D   02
INIT_CONTEXT                    ********   X   03
KB_CHECK                        ********   X   03
L$A_CCP                                   00000240    D
L$A_DEVP                                  0000021C    D
L$A_DREG                                  00000224    D
L$A_DTP                                   00000218    D
L$A_ICP                                   0000023C    D
L$A_LASTADR                               00000214    D
L$A_NAME                                  00000208    D
L$A_REPP                                  00000244    D
L$A_SECNAM                                00000250    D
L$A_STATAB                                00000248    D
L$A_TSTCNT                                00000254    D
L$L_ENVIRON                               00000204    D
L$L_ERRTYP                                0000024C    D
L$L_HEADLENGTH                            00000200    D
L$L_REV                                   0000020C    D
L$L_UNIT                                  00000220    D
L$L_UNUSED                                00000228    D
L$L_UPDATE                                00000210    D
OFF                                     = 00000000    D
ON                                      = 00000001    D
QA$AOB_CHECK_STATE                        ********  X   03
QA$K_ABORT_NOW                          = 00000007    D
QA$K_APT_PHASE_ONE                      = 0000000B    D
QA$K_APT_PHASE_TWO                      = 0000000C    D
QA$K_BAD_ADDRESS                        = 00000009    D
QA$K_CONTROL_C_CONT                     = 0000000A    D
QA$K_DEALLOCATION                       = 0000000E    D
QA$K_ERROR_PHASE_ONE                    = 00000005    D
QA$K_ERROR_PHASE_THREE                  = 00000007    D
QA$K_ERROR_PHASE_TWO                    = 00000006    D
QA$K_FAILURE                            = 00000000    D
QA$K_FIRST_BRANCH_CODE                  = 00000000    D
QA$K_FIRST_TIME                         = 00000005    D
QA$K_INIT_CONTEXT_DONE                  = 00000003    D
QA$K_LAST_BRANCH_CODE                   = 00000025    D
QA$K_LOOP_ON_SUBTEST                    = 00000003    D
QA$K_LOOP_ON_TEST                       = 00000002    D
QA$K_LOOP_TEST_DONE                     = 00000004    D
QA$K_MEMORY_MANAGE                      = 0000000D    D
QA$K_MULTIPLE_PASS                      = 00000001    D
QA$K_NODEF                              = 00000000    D
QA$K_NORMAL_START                       = 00000000    D
QA$K_NO_HEADER                          = 00000006    D
QA$K_NUMBER_OF_CHECKS                   = 0000000F    D
QA$K_NUMBER_QA_FLAGS                    = 00000008    D
QA$K_NUMB_ROUTINE_STATES                = 00000004    D
QA$K_QA_DEBUG                           = 00000001    D
QA$K_QA_DONE                            = 00000002    D
QA$K_RUN_BACKWARDS                      = 00000004    D
QA$K_SECTION                            = 00000008    D
QA$K_SUCCESS                            = 00000001    D
QA$MAIN                                   ********  X   03
QA$V_CHECK_DONE                         = 00000003    D
QA$V_DOING_CLEANUP                      = 00000002    D
QA$V_ERROR_FOUND                        = 00000001    D
QA$V_INIT_DONE                          = 00000000    D
RBGNSUBX                                  000000B2 R  D   03
```

```
RCKLOOPX                        00000263 R  D  03
RESCAPEX                        00000014 R  D  03
SIZ...                        = 00000001    D
TRUE                          = 00000001    D
VRRESTART                       ********  X   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	000000A4 (164.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	0000028B (651.)	03 (3.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000005	468 (2)	277 (1) 443 (2) 468 (2)
$ER	=00000003	473 (2)	#-282 (1) #-473 (2)
$MODULE	00000000-R	140 (1)	282 (1) 473 (2)
APC$_ABORT	=00000006	128 (1)	
APC$_CONTINUE	=00000009	128 (1)	
APC$_DUMP	=00000003	128 (1)	
APC$_NOP	=00000000	128 (1)	
APC$_START	=00000005	128 (1)	
APC$_ZERO	=00000004	128 (1)	
APM$_ABTDON	=00000006	128 (1)	
APM$_DEVERR	=00000002	128 (1)	
APM$_DONE	=0000000A	128 (1)	#-451 (2)
APM$_EXCEPT	=0000000B	128 (1)	
APM$_HARDERR	=00000003	128 (1)	
APM$_MORE	=00000008	128 (1)	
APM$_NOMES	=00000000	128 (1)	
APM$_PREPERR	=0000000C	128 (1)	
APM$_PRGERR	=00000005	128 (1)	
APM$_SOFTERR	=00000004	128 (1)	
APM$_SPOOL	=00000009	128 (1)	
APM$_SYSERR	=00000001	128 (1)	
BEGIN	00000000-XR		457 (2)
BIT...	=00000004	132 (1)	126 (1) 129 (1) 130 (1) 131 (1) 132 (1)
BPTCODE	=00000003	124 (1)	#-286 (1) #-486 (2)
DS$AA_BPTADDR	00000000-XR		#-284 (1) #-483 (2)
DS$AB_BPTINST	00000000-XR		#-285 (1) #-485 (2)
DS$GA_CHKLPPC	00000000-XR		#-540 (2) #-543 (2) #-546 (2)
DS$GA_LOOPADR	00000000-XR		#-259 (1) #-412 (2)
DS$GL_ERRCNT	00000000-XR		#-443 (2)
DS$GL_FLAGS	00000000-XR		189 (1) 247 (1) 249 (1) 281 (1) 370 (2) 373 (2) 413 (2) 424 (2) 449 (2) 472 (2) 539 (2) 586 (3)
DS$GL_LSTTEST	00000000-XR		#-382 (2)
DS$GL_SUBTEST	00000000-XR		#-380 (2)
DS$K_BBC	=0000001D	131 (1)	
DS$K_BBCC	=00000021	131 (1)	
DS$K_BBCCI	=00000023	131 (1)	
DS$K_BBCS	=0000001F	131 (1)	
DS$K_BBS	=0000001C	131 (1)	
DS$K_BBSC	=00000020	131 (1)	
DS$K_BBSS	=0000001E	131 (1)	
DS$K_BBSSI	=00000022	131 (1)	
DS$K_BCC_M	=0000001B	131 (1)	
DS$K_BCS_M	=0000001A	131 (1)	
DS$K_BEQLU_B	=00000005	131 (1)	
DS$K_BEQLU_M	=00000011	131 (1)	
DS$K_BEQL_B	=00000004	131 (1)	
DS$K_BEQL_M	=00000010	131 (1)	
DS$K_BERROR	=00000026	131 (1)	

```
DS$K_BGEQU_B                    =00000007     131     (1)
DS$K_BGEQU_M                    =00000013     131     (1)
DS$K_BGEQ_B                     =00000006     131     (1)
DS$K_BGEQ_M                     =00000012     131     (1)
DS$K_BGTRU_B                    =00000009     131     (1)
DS$K_BGTRU_M                    =00000015     131     (1)
DS$K_BGTR_B                     =00000008     131     (1)
DS$K_BGTR_M                     =00000014     131     (1)
DS$K_BLBC                       =00000025     131     (1)       131     (1)
DS$K_BLBS                       =00000024     131     (1)
DS$K_BLEQU_B                    =00000003     131     (1)
DS$K_BLEQU_M                    =0000000F     131     (1)
DS$K_BLEQ_B                     =00000002     131     (1)
DS$K_BLEQ_M                     =0000000E     131     (1)
DS$K_BLSSU_B                    =00000001     131     (1)
DS$K_BLSSU_M                    =0000000D     131     (1)
DS$K_BLSS_B                     =00000000     131     (1)       131     (1)
DS$K_BLSS_M                     =0000000C     131     (1)
DS$K_BNEQU_B                    =0000000B     131     (1)
DS$K_BNEQU_M                    =00000017     131     (1)
DS$K_BNEQ_B                     =0000000A     131     (1)
DS$K_BNEQ_M                     =00000016     131     (1)
DS$K_BNERROR                    =00000027     131     (1)
DS$K_BVC_M                      =00000019     131     (1)
DS$K_BVS_M                      =00000018     131     (1)
DS$K_DS_STARTING_LOCATION       =00010000     131     (1)
DS$K_ERROR                      =00000002     126     (1)
DS$K_NORMAL                     =00000001     126     (1)
DS$K_PRINTB                     =00000002     132     (1)
DS$K_PRINTF                     =00000001     132     (1)     #-277     (1)     #-443     (2)     #-468     (2)
DS$K_PRINTI                     =00000000     132     (1)
DS$K_PRINTX                     =00000003     132     (1)
DS$K_SEVERE                     =00000004     126     (1)
DS$K_SUBSYS                     =00000066     126     (1)       126     (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     132     (1)
DS$K_TYPE_ABORT_TEST            =00000013     132     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     132     (1)
DS$K_TYPE_COMMAND_OUT           =00000016     132     (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     132     (1)
DS$K_TYPE_DS_PROMPT             =00000001     132     (1)
DS$K_TYPE_DS_START              =0000001D     132     (1)
DS$K_TYPE_ERRDEV                =00000008     132     (1)
DS$K_TYPE_ERRHARD               =00000006     132     (1)
DS$K_TYPE_ERROR_BODY            =00000009     132     (1)
DS$K_TYPE_ERROR_END             =0000000A     132     (1)
DS$K_TYPE_ERRPREP               =0000001B     132     (1)
DS$K_TYPE_ERRSOFT               =00000007     132     (1)
DS$K_TYPE_ERRSUP                =00000004     132     (1)
DS$K_TYPE_ERRSYS                =00000005     132     (1)
DS$K_TYPE_ERR_HALT              =0000000D     132     (1)
DS$K_TYPE_EXCEPTION             =0000000C     132     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     132     (1)
DS$K_TYPE_FIRST_PASS            =00000011     132     (1)
DS$K_TYPE_GENERAL               =00000000     132     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     132     (1)
DS$K_TYPE_NO_TESTS              =00000012     132     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     132     (1)
```

```
DS$K_TYPE_PROGRAM_END           =00000010     132     (1)     #-443     (2)
DS$K_TYPE_PROGRAM_INFO          =00000017     132     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     132     (1)
DS$K_TYPE_QIO_INVADP            =00000024     132     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     132     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     132     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     132     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     132     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     132     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     132     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     132     (1)     #-277     (1)     #-468     (2)
DS$K_TYPE_START_ERR             =00000018     132     (1)
DS$K_TYPE_START_LIST            =00000025     132     (1)
DS$K_TYPE_SUMMARY               =0000000E     132     (1)
DS$K_TYPE_USER_PROMPT           =00000002     132     (1)
DS$K_WARNING                    =00000000     126     (1)
DS$M_ABRTFLG                    =00000040     129     (1)
DS$M_BADTIME                    =00100000     129     (1)
DS$M_BATCH                      =00400000     129     (1)
DS$M_BRKCLR                     =00001000     129     (1)
DS$M_BRKPT                      =00000800     129     (1)
DS$M_CHARFLG                    =00000100     129     (1)
DS$M_CMDFLG                     =00000080     129     (1)
DS$M_CTRLC                      =00000001     129     (1)
DS$M_CTRLO                      =00010000     129     (1)
DS$M_DEVFLG                     =00000200     129     (1)
DS$M_DISABLCC                   =01000000     129     (1)
DS$M_DONFLG                     =00002000     129     (1)
DS$M_ERRFLG                     =00000010     129     (1)
DS$M_EXCEPT                     =00080000     129     (1)
DS$M_EXETST                     =00040000     129     (1)
DS$M_HLTFLG                     =00000008     129     (1)
DS$M_LODFLG                     =00000002     129     (1)
DS$M_MEMMGT                     =00008000     129     (1)
DS$M_NI                         =04000000     129     (1)
DS$M_OUTPUT                     =00800000     129     (1)
DS$M_RUBFLG                     =00000020     129     (1)
DS$M_SCRIPT                     =00200000     129     (1)
DS$M_SETIMR                     =02000000     129     (1)
DS$M_STRFLG                     =00000004     129     (1)
DS$M_SUBT                       =00004000     129     (1)
DS$M_SYSFLG                     =00000400     129     (1)
DS$M_TIMED                      =02000000     129     (1)
DS$M_TIMED_OUT                  =08000000     129     (1)
DS$M_TIMRON                     =00020000     129     (1)
DS$SUMMARY                       00000000-XR                446     (2)
DS$V_ABRTFLG                    =00000006     129     (1)
DS$V_BADTIME                    =00000014     129     (1)
DS$V_BATCH                      =00000016     129     (1)
DS$V_BRKCLR                     =0000000C     129     (1)
DS$V_BRKPT                      =0000000B     129     (1)
DS$V_CHARFLG                    =00000008     129     (1)
DS$V_CMDFLG                     =00000007     129     (1)     #-280     (1)     #-425     (2)     #-471     (2)
DS$V_CTRLC                      =00000000     129     (1)
DS$V_CTRLO                      =00000010     129     (1)     #-427     (2)     #-449     (2)
DS$V_DEVFLG                     =00000009     129     (1)
DS$V_DISABLCC                   =00000018     129     (1)
```

```
DS$V_DONFLG                     =0000000D     129     (1)
DS$V_ERRFLG                     =00000004     129     (1)    #-189    (1)    #-249    (1)    #-373    (2)    #-539    (2)
                                                             #-586    (3)
DS$V_EXCEPT                     =00000013     129     (1)
DS$V_EXETST                     =00000012     129     (1)
DS$V_HLTFLG                     =00000003     129     (1)
DS$V_LODFLG                     =00000001     129     (1)
DS$V_MEMMGT                     =0000000F     129     (1)
DS$V_NI                         =0000001A     129     (1)
DS$V_OUTPUT                     =00000017     129     (1)
DS$V_RUBFLG                     =00000005     129     (1)
DS$V_SCRIPT                     =00000015     129     (1)
DS$V_SETIMR                     =00000019     129     (1)
DS$V_STRFLG                     =00000002     129     (1)
DS$V_SUBT                       =0000000E     129     (1)    #-246    (1)    #-369    (2)    #-413    (2)
DS$V_SYSFLG                     =0000000A     129     (1)
DS$V_TIMED                      =00000019     129     (1)
DS$V_TIMED_OUT                  =0000001B     129     (1)
DS$V_TIMRON                     =00000011     129     (1)
DS$_AP_NORMAL_BREAK             =00660150     126     (1)
DS$_ARITH                       =006600D0     126     (1)
DS$_ASBE                        =00660118     126     (1)
DS$_BADLINK                     =006600F0     126     (1)
DS$_BADTYPE                     =006600E8     126     (1)
DS$_BIIC                        =00660120     126     (1)
DS$_CHME                        =006600A8     126     (1)
DS$_CHMK                        =006600E0     126     (1)
DS$_DEVNAME                     =00660108     126     (1)
DS$_ERROR                       =00660002     126     (1)    #-584    (3)
DS$_FHWE                        =00660068     126     (1)
DS$_FRAGBUF                     =00660080     126     (1)
DS$_ICBUSY                      =006600C8     126     (1)
DS$_ICERR                       =006600C0     126     (1)
DS$_IHWE                        =00660060     126     (1)
DS$_ILLCHAR                     =00660018     126     (1)
DS$_ILLPAGCNT                   =00660078     126     (1)
DS$_ILLUNIT                     =00660100     126     (1)
DS$_INIT_FAIL                   =00660140     126     (1)
DS$_INSFMEM                     =00660050     126     (1)
DS$_INVCPU                      =00660130     126     (1)
DS$_IPL2HI                      =006600B8     126     (1)
DS$_IVADDR                      =00660040     126     (1)
DS$_IVVECT                      =00660038     126     (1)
DS$_KRNLSTK                     =00660090     126     (1)
DS$_LOGIC                       =00660070     126     (1)
DS$_MCHK                        =00660088     126     (1)
DS$_MEM_ALLOC_ERR               =00660138     126     (1)
DS$_MMOFF                       =00660058     126     (1)
DS$_NEEDUNIT                    =006600F8     126     (1)
DS$_NODE                        =00660128     126     (1)
DS$_NOPCS                       =00660110     126     (1)
DS$_NORMAL                      =00660001     126     (1)    #-589    (3)
DS$_NOSUPPORT                   =006600B1     126     (1)
DS$_NOTALLOWMP                  =00660148     126     (1)
DS$_NOTDON                      =00660030     126     (1)
DS$_NOTIMP                      =006600B0     126     (1)      126    (1)
DS$_NULLSTR                     =00660010     126     (1)
```

```
DS$_OVERFLOW                    =00660008     126     (1)
DS$_POWER                       =00660098     126     (1)
DS$_PROGERR                     =00660020     126     (1)
DS$_SEVERE                      =00660004     126     (1)
DS$_TRANSL                      =006600A0     126     (1)
DS$_TRUNCATE                    =00660028     126     (1)
DS$_UNEXPINT                    =006600D8     126     (1)
DS$_UNSUPPDEV                   =00660158     126     (1)
DS$_VASFULL                     =00660048     126     (1)
DS$_WARNING                     =00660000     126     (1)     126     (1)
DSA$GL_FLAGS                     0000FE00                       279     (1)     372     (2)     430     (2)     453     (2)
                                                                470     (2)     538     (2)     588     (3)
DSA$GL_MSGTYP                    0000FE40                     #-452     (2)
DSA$GL_PASSES                    0000FE08                     #-392     (2)   #-399     (2)
DSA$GL_PASSNO                    0000FE54                       391     (2)   #-443     (2)
DSA$GL_SUBTNO                    0000FE4C                     #-277     (1)   #-295     (1)   #-359     (2)   #-443     (2)
                                                              #-468     (2)   #-488     (2)
DSA$GL_TESTNO                    0000FE50                     #-262     (1)   #-277     (1)   #-288     (1)   #-357     (2)
                                                              #-443     (2)   #-468     (2)   #-487     (2)
DSA$V_HALT                      =00000000                    #-279     (1)   #-470     (2)
DSA$V_LOOP                      =00000002                    #-371     (2)   #-537     (2)   #-587     (3)
DSA$V_QA                        =0000000F                    #-430     (2)   #-453     (2)
DSQA$K_DISPAT_AFTER_INIT        =00000014     130     (1)
DSQA$K_DISPAT_BEFORE_INIT       =00000013     130     (1)
DSQA$K_DISPAT_CALLTEST          =00000015     130     (1)
DSQA$K_DISPAT_DONE_TESTS        =00000018     130     (1)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001     130     (1)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002     130     (1)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000     130     (1)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003     130     (1)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004     130     (1)
DSQA$K_DUMMY_1                  =00000007     130     (1)
DSQA$K_ERROR_ERROR_BGN          =00000006     130     (1)
DSQA$K_ERROR_ERROR_END          =00000005     130     (1)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008     130     (1)
DSQA$K_LOOP_DSX$BGNSUB          =00000009     130     (1)   #-244     (1)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B     130     (1)   #-534     (2)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A     130     (1)   #-348     (2)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E     130     (1)
DSQA$K_PARAM_DSX$GETDATA        =0000000C     130     (1)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F     130     (1)
DSQA$K_PARAM_DSX$GETSTRING      =00000010     130     (1)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D     130     (1)
DSQA$K_QA_DSX$BRANCH            =00000011     130     (1)
DSQA$K_START_BEFORE_HEADER      =00000017     130     (1)
DSQA$K_START_VRRESTART          =00000012     130     (1)
DSQA$K_START_VRSTART            =00000016     130     (1)
DSX$BGNSUB                       00000015-R   242     (1)
DSX$CKLOOP                       00000224-R   532     (2)
DSX$ENDSUB                       000000B3-R   346     (2)
DSX$ESCAPE                       00000000-R   186     (1)
DSX$INLOOP                       00000264-R   581     (3)
DSX$PRINT                        00000000-XR              277     (1)     443     (2)     468     (2)
DS_CLEANUP                       00000000-XR              447     (2)
DS_ERRSUP                        00000000-XR              282     (1)     473     (2)
FALSE                           =00000000     131     (1)
FMTLOOPDON                       0000004F-R   145     (1)     443     (2)
```

FMTSEQERR	00000005-R	142	(1)	277	(1)	468	(2)					
INIT_CONTEXT	00000000-XR			445	(2)							
KB_CHECK	00000000-XR			188	(1)	296	(1)	349	(2)	536	(2)	
				583	(3)							
OFF	=00000000	131	(1)									
ON	=00000001	131	(1)									
QA$AOB_CHECK_STATE	00000000-XR			432	(2)							
QA$K_ABORT_NOW	=00000007	131	(1)									
QA$K_APT_PHASE_ONE	=0000000B	131	(1)									
QA$K_APT_PHASE_TWO	=0000000C	131	(1)									
QA$K_BAD_ADDRESS	=00000009	131	(1)									
QA$K_CONTROL_C_CONT	=0000000A	131	(1)									
QA$K_DEALLOCATION	=0000000E	131	(1)									
QA$K_ERROR_PHASE_ONE	=00000005	131	(1)									
QA$K_ERROR_PHASE_THREE	=00000007	131	(1)									
QA$K_ERROR_PHASE_TWO	=00000006	131	(1)									
QA$K_FAILURE	=00000000	131	(1)									
QA$K_FIRST_BRANCH_CODE	=00000000	131	(1)									
QA$K_FIRST_TIME	=00000005	131	(1)									
QA$K_INIT_CONTEXT_DONE	=00000003	131	(1)									
QA$K_LAST_BRANCH_CODE	=00000025	131	(1)									
QA$K_LOOP_ON_SUBTEST	=00000003	131	(1)	#-431	(2)							
QA$K_LOOP_ON_TEST	=00000002	131	(1)									
QA$K_LOOP_TEST_DONE	=00000004	131	(1)									
QA$K_MEMORY_MANAGE	=0000000D	131	(1)									
QA$K_MULTIPLE_PASS	=00000001	131	(1)									
QA$K_NODEF	=00000000	131	(1)									
QA$K_NORMAL_START	=00000000	131	(1)									
QA$K_NO_HEADER	=00000006	131	(1)									
QA$K_NUMBER_OF_CHECKS	=0000000F	131	(1)									
QA$K_NUMBER_QA_FLAGS	=00000008	131	(1)									
QA$K_NUMB_ROUTINE_STATES	=00000004	131	(1)									
QA$K_QA_DEBUG	=00000001	131	(1)									
QA$K_QA_DONE	=00000002	131	(1)									
QA$K_RUN_BACKWARDS	=00000004	131	(1)									
QA$K_SECTION	=00000008	131	(1)									
QA$K_SUCCESS	=00000001	131	(1)									
QA$MAIN	00000000-XR			244	(1)	348	(2)	534	(2)			
QA$V_CHECK_DONE	=00000003	131	(1)									
QA$V_DOING_CLEANUP	=00000002	131	(1)									
QA$V_ERROR_FOUND	=00000001	131	(1)									
QA$V_INIT_DONE	=00000000	131	(1)									
RBGNSUBX	000000B2-R	298	(1)									
RCKLOOPX	00000263-R	549	(2)	#-538	(2)	#-539	(2)	#-544	(2)			
RESCAPEX	00000014-R	192	(1)	#-189	(1)							
SIZ...	=00000001	129	(1)	129	(1)							
TRUE	=00000001	131	(1)									
VRRESTART	00000000-XR			#-454	(2)							

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_PRINT_S	1	277 (1)	277 (1) 443 (2) 468 (2)
$DEF	1	132 (1)	
$DEFINI	1	125 (1)	125 (1) 127 (1)
$DS_DSADEF	5	125 (1)	125 (1)
$DS_DSDEF	3	126 (1)	126 (1)
$DS_HDRDEF	2	127 (1)	127 (1)
$DS_QADEF	2	131 (1)	131 (1)
$DS_SUMMARY_S	1	446 (2)	446 (2)
$DS_TYPEDEF	4	132 (1)	132 (1)
$EQU	1	132 (1)	126 (1) 130 (1) 131 (1) 132 (1)
$EQULS1	1	132 (1)	126 (1) 130 (1) 131 (1) 132 (1)
$EQULST	1	126 (1)	126 (1) 130 (1) 131 (1) 132 (1)
$GBLINI	2		126 (1) 129 (1) 130 (1) 131 (1)
			132 (1)
$PRINT	2	272 (1)	272 (1) 436 (2) 463 (2)
$PUSHADR	1	282 (1)	282 (1) 473 (2)
$VIELD	1		129 (1)
$VIELD1	1	132 (1)	129 (1)
APTDEF	1	128 (1)	128 (1)
BR_IF_ERRFLG	1	373 (2)	373 (2)
BR_IF_NOT_ERRFLG	1	189 (1)	189 (1) 539 (2) 586 (3)
BR_IF_NOT_QA	1	430 (2)	430 (2) 453 (2)
CLEAR_CTRLO	1	449 (2)	449 (2)
CLEAR_ERRFLG	1	249 (1)	249 (1)
DSFDEF	3	129 (1)	129 (1)
DSQA	2	130 (1)	130 (1)
DS_QADEFS	6	131 (1)	131 (1)
ERRSUP_S	1	282 (1)	282 (1) 473 (2)
MODNAM	1	140 (1)	140 (1)
QA_MAIN	1	244 (1)	244 (1) 348 (2) 534 (2)
SET_HALT	1	279 (1)	279 (1) 470 (2)
SET_SUBT	1	413 (2)	413 (2)

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE      REFERENCES...
------          -----      -------------
AOBLEQ          00F3       392     (2)
BBC             00E1       189     (1)     371     (2)     430     (2)     431     (2)     453     (2)     537     (2)     539     (2)
                           586     (3)     587     (3)
BBCC            00E5       369     (2)
BBCS            00E3       280     (1)     471     (2)
BBS             00E0       373     (2)
BBSC            00E4       249     (1)     449     (2)
BBSS            00E2       246     (1)     279     (1)     413     (2)     470     (2)
BEQL            0013       263     (1)     360     (2)     400     (2)     542     (2)
BNEQ            0012       358     (2)     381     (2)     383     (2)     544     (2)
BRB             0011       402     (2)     434     (2)
BRW             0031       362     (2)     417     (2)     454     (2)
CALLS           00FB       244     (1)     277     (1)     282     (1)     348     (2)     443     (2)     446     (2)     468     (2)
                           473     (2)     534     (2)
CMPL            00D1       262     (1)     357     (2)     359     (2)     380     (2)     382     (2)     543     (2)
CVTBL           0098       277     (1)     443     (2)     468     (2)
DECL            00D7       401     (2)
INSV            00F0       425     (2)     427     (2)
JSB             0016       188     (1)     296     (1)     349     (2)     445     (2)     447     (2)     457     (2)     536     (2)
                           583     (3)
MOVAL           00DE       284     (1)     391     (2)     424     (2)     482     (2)
MOVB            0090       285     (1)     286     (1)     484     (2)     486     (2)
MOVL            00D0       190     (1)     258     (1)     288     (1)     295     (1)     412     (2)     451     (2)     487     (2)
                           488     (2)     546     (2)     547     (2)     584     (3)     589     (3)
MOVZBL          009A       277     (1)     443     (2)     468     (2)
PUSHAB          009F       277     (1)     443     (2)     468     (2)
PUSHAL          00DF       282     (1)     473     (2)
PUSHL           00DD       244     (1)     277     (1)     282     (1)     348     (2)     443     (2)     468     (2)     473     (2)
                           534     (2)
RET             0004       193     (1)     299     (1)     415     (2)     489     (2)     550     (2)     592     (3)
TSTL            00D5       399     (2)     540     (2)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	140	(1)	143	(1)	146	(1)										
.END	594	(3)														
.ENDC	126	(1)	129	(1)	130	(1)	131	(1)	132	(1)	244	(1)	277	(1)	282	(1)
	348	(2)	443	(2)	468	(2)	473	(2)	534	(2)						
.ENDM	125	(1)	126	(1)	127	(1)	128	(1)	129	(1)	130	(1)	131	(1)	132	(1)
	140	(1)	189	(1)	244	(1)	249	(1)	272	(1)	277	(1)	279	(1)	282	(1)
	373	(2)	413	(2)	430	(2)	446	(2)	449	(2)						
.ENDR	126	(1)	129	(1)	130	(1)	131	(1)	132	(1)	277	(1)	443	(2)	468	(2)
.ENTRY	186	(1)	242	(1)	346	(2)	532	(2)	581	(3)						
.IDENT	5	(1)														
.IF	126	(1)	129	(1)	130	(1)	131	(1)	132	(1)	244	(1)	277	(1)	282	(1)
	348	(2)	443	(2)	468	(2)	473	(2)	534	(2)						
.IFF	126	(1)	129	(1)	130	(1)	131	(1)	132	(1)	282	(1)	473	(2)		
.IF_FALSE	244	(1)	348	(2)	534	(2)										
.IIF	126	(1)	129	(1)	130	(1)	131	(1)	132	(1)	282	(1)	473	(2)		
.IRP	126	(1)	129	(1)	130	(1)	131	(1)	132	(1)	277	(1)	443	(2)	468	(2)
.LIBRARY	113	(1)	114	(1)	115	(1)										
.LIST	125	(1)	127	(1)												
.MACRO	126	(1)	129	(1)	130	(1)	131	(1)	132	(1)						
.MDELETE	126	(1)	131	(1)												
.NLIST	125	(1)	127	(1)												
.NOCROSS	125	(1)	127	(1)												
.NOSHOW	6	(1)														
.PAGE	108	(1)	133	(1)	149	(1)	185	(1)	194	(1)	241	(1)	264	(1)	289	(1)
	345	(2)	374	(2)	38	(1)	403	(2)	418	(2)	458	(2)	490	(2)	531	(2)
	580	(3)														
.PSECT	134	(1)	151	(1)												
.RESTORE	125	(1)	127	(1)												
.SAVE	125	(1)	127	(1)												
.SBTTL	109	(1)	150	(1)	195	(1)	301	(2)	491	(2)	552	(3)				
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	15	#-190	(1)	#-262	(1)	#-277	(1)	#-288	(1)	#-295	(1)	#-357	(2)
		#-359	(2)	#-380	(2)	#-382	(2)	#-468	(2)	#-487	(2)	#-488	(2)
		#-547	(2)										
FP	12	#-190	(1)	#-258	(1)	284	(1)	#-285	(1)	#-286	(1)	#-412	(2)
		482	(2)	#-484	(2)	#-486	(2)	#-543	(2)	#-546	(2)	#-547	(2)
R0	8	#-391	(2)	#-392	(2)	#-401	(2)	#-424	(2)	426	(2)	428	(2)
		#-584	(3)	#-589	(3)								
SP	6	#-277	(1)	#-443	(2)	#-468	(2)						

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.18
Command processing	881	00:00:00.22	00:00:00.65
Pass 1	674	00:00:02.39	00:00:04.43
Symbol table sort	0	00:00:00.11	00:00:00.10
Pass 2	9	00:00:00.61	00:00:01.01
Symbol table output	0	00:00:00.05	00:00:00.12
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	6	00:00:00.29	00:00:00.51
Assembler run totals	1592	00:00:03.69	00:00:07.00

The working set limit was 3400 pages.
135589 bytes (265 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 401 non-local and 22 local symbols.
594 source lines were read in Pass 1, producing 54 object records in Pass 2.
99 pages of virtual memory were used to define 31 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	7
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	15
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	28

591 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 14 Frame C8          Sequence 2771
LOOP                        *** LOOP Loop control               11-NOV-1987 17:39:46  VAX/VMS Macro V04-00        Page  36
VAX-11 Macro Run Statistics                                      3-JAN-1985 16:52:39  LOOP.MAR;1                  (3)

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:LOOP.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DI
```

Table of contents

(1) 66 MASSBUS ADAPTER INTERRUPT DISPATCHER
(1) 152 MASSBUS ADAPTER INITIALIZATION

```
                     0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element MBAINT.MAR
                     0000     2 ; *1     6-FEB-1986 15:19:30 DS ""
                     0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element MBAINT.MAR
                     0000     4         .TITLE  MBAINT - MASSBUS ADAPTER INTERRUPT DISPATCHER
                     0000     5         .IDENT  /01/
                00000000     6         .PSECT  SEP, SHR, WRT, EXE, LONG
                     0000     7
                     0000     8 ;
                     0000     9 ;*****************************************************************************
                     0000    10 ;*                                                                           *
                     0000    11 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                     0000    12 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                     0000    13 ;*                                                                           *
                     0000    14 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                     0000    15 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                     0000    16 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                     0000    17 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                     0000    18 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                     0000    19 ;*  TRANSFERRED.                                                             *
                     0000    20 ;*                                                                           *
                     0000    21 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                     0000    22 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                     0000    23 ;*  CORPORATION.                                                             *
                     0000    24 ;*                                                                           *
                     0000    25 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                     0000    26 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                     0000    27 ;*                                                                           *
                     0000    28 ;*****************************************************************************
                     0000    29 ;
                     0000    30 ; D. N. CUTLER 30-JAN-77
                     0000    31 ;
                     0000    32 ; MASSBUS ADAPTER INTERRUPT DISPATCHER
                     0000    33 ; R. S. RIGGS  14-Jun-1980 Minor mods to accomodate SUPERVISOR
                     0000    34 ;
                     0000    35 ; MASSBUS ADAPTER INTERRUPT DISPATCHER
                     0000    36 ;
                     0000    37 ; MODIFIED BY:
                     0000    38 ;
                     0000    39 ;       V03     RLR0001         Robert L. Rappaport     15-NOV-1979
                     0000    40 ;               Changed attention summary bit clearing logic to allow
                     0000    41 ;               support of TM78 Tape Controller.  Specifically, for
                     0000    42 ;               all MASSBUS multi-device controllers (currently TM03
                     0000    43 ;               and TM78), the responsibility for clearing the attention
                     0000    44 ;               summary bit for the device has been removed from this
                     0000    45 ;               module and placed in the individual Device Drivers.
                     0000    46 ;               For MASSBUS single device controllers (DISKS) the
                     0000    47 ;               logic remains as it was.  That is the attention summary
                     0000    48 ;               bit is cleared here before dispatching to the specific Driver.
                     0000    49 ;
                     0000    50 ;
                     0000    51 ;       V02     NPK0001         N. KRONENBERG           4-NOV-1979
                     0000    52 ;               ADDED CHECK FOR CBHUNG ON MBA INTERRUPT
                     0000    53 ;
```

```
                     0000     55 ;
                     0000     56 ; MACRO LIBRARY CALLS
                     0000     57 ;
                     0000     58
                     0000     59          $ADPDEF                             ;DEFINE ADP OFFSETS
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .IIF    NB,ADP$L_CSR,   ADP$L_CSR:
           00000004  0000                         .BLKL   1
                     0004                 .IIF    NB,ADP$L_LINK,  ADP$L_LINK:
           00000008  0004                         .BLKL   1
                     0008                 .IIF    NB,ADP$W_SIZE,  ADP$W_SIZE:
           0000000A  0008                         .BLKW   1
                     000A                 .IIF    NB,ADP$B_TYPE,  ADP$B_TYPE:
           0000000B  000A                         .BLKB   1
                     000B                 .IIF    NB,ADP$B_NUMBER,        ADP$B_NUMBER:
           0000000C  000B                         .BLKB   1
                     000C                 .IIF    NB,ADP$W_TR,    ADP$W_TR:
           0000000E  000C                         .BLKW   1
                     000E                 .IIF    NB,ADP$W_ADPTYPE,       ADP$W_ADPTYPE:
           00000010  000E                         .BLKW   1
                     0010                 .IIF    NB,ADP$L_VECTOR,        ADP$L_VECTOR:
                     0010                 .IIF    NB,ADP$L_CRB,   ADP$L_CRB:
           00000014  0010                         .BLKL   1
                     0014                 .IIF    NB,ADP$C_MBAADPLEN,     ADP$C_MBAADPLEN:
                     0014                 .IIF    NB,ADP$K_MBAADPLEN,     ADP$K_MBAADPLEN:
                     0014                 .IIF    NB,ADP$C_DRADPLEN,      ADP$C_DRADPLEN:
                     0014                 .IIF    NB,ADP$K_DRADPLEN,      ADP$K_DRADPLEN:
                     0014                 .IIF    NB,ADP$L_DPQFL, ADP$L_DPQFL:
                     0014                 .IIF    NB,ADP$L_PRQQFL,        ADP$L_PRQQFL:
           00000018  0014                         .BLKL   1
                     0018                 .IIF    NB,ADP$L_DPQBL, ADP$L_DPQBL:
                     0018                 .IIF    NB,ADP$L_PRQQBL,        ADP$L_PRQQBL:
           0000001C  0018                         .BLKL   1
                     001C                 .IIF    NB,ADP$L_MRQFL, ADP$L_MRQFL:
                     001C                 .IIF    NB,ADP$L_SHB,   ADP$L_SHB:
           00000020  001C                         .BLKL   1
                     0020                 .IIF    NB,ADP$L_MRQBL, ADP$L_MRQBL:
                     0020                 .IIF    NB,ADP$B_PORT,  ADP$B_PORT:
           00000021  0020                         .BLKB   1
           00000024  0021                         .BLKB   3
                     0024                 .IIF    NB,ADP$W_DPBITMAP,      ADP$W_DPBITMAP:
           00000026  0024                         .BLKW   1
                     0026                 .IIF    NB,ADP$W_MRBITMAP,      ADP$W_MRBITMAP:
           00000064  0026                         .BLKW   31
                     0064                 .IIF    NB,ADP$L_INTD,  ADP$L_INTD:
           00000070  0064                         .BLKL   3
                     0070                 .IIF    NB,ADP$C_MPMADPLEN,     ADP$C_MPMADPLEN:
                     0070                 .IIF    NB,ADP$K_MPMADPLEN,     ADP$K_MPMADPLEN:
                     0070                 .IIF    NB,ADP$C_UBAADPLEN,     ADP$C_UBAADPLEN:
                     0070                 .IIF    NB,ADP$K_UBAADPLEN,     ADP$K_UBAADPLEN:
                00000000                 .RESTORE
                     0000     60          $DDBDEF                             ;DEFINE DDB OFFSETS
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .PSECT  $ABS$,ABS
           00000000  0070                 .=0
                     0000                 .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
           00000004  0000                         .BLKL   1
```

```
                    0004                .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
          00000008  0004                        .BLKL   1
                    0008                .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
          0000000A  0008                        .BLKW   1
                    000A                .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
          0000000B  000A                        .BLKB   1
          0000000C  000B                        .BLKB   1
                    000C                .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
          00000010  000C                        .BLKL   1
                    0010                .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
          00000013  0010                        .BLKB   3
                    0013                .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
          00000014  0013                        .BLKB   1
                    0014                .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
          00000015  0014                        .BLKB   1
          00000024  0015                        .BLKB   15
                    0024                .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
          00000025  0024                        .BLKB   1
          00000034  0025                        .BLKB   15
                    0034                .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                    0034                .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                00000000                .RESTORE
                    0000    61          $DDTDEF                         ;DEFINE DDT OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  0070                .=0
                    0000                .IIF    NB,DDT$L_START, DDT$L_START:
          00000004  0000                        .BLKL   1
                    0004                .IIF    NB,DDT$L_UNSOLINT,      DDT$L_UNSOLINT:
          00000008  0004                        .BLKL   1
                    0008                .IIF    NB,DDT$L_FDT,   DDT$L_FDT:
          0000000C  0008                        .BLKL   1
                    000C                .IIF    NB,DDT$L_CANCEL,        DDT$L_CANCEL:
          00000010  000C                        .BLKL   1
                    0010                .IIF    NB,DDT$L_REGDUMP,       DDT$L_REGDUMP:
          00000014  0010                        .BLKL   1
                    0014                .IIF    NB,DDT$W_DIAGBUF,       DDT$W_DIAGBUF:
          00000016  0014                        .BLKW   1
                    0016                .IIF    NB,DDT$W_ERRORBUF,      DDT$W_ERRORBUF:
          00000018  0016                        .BLKW   1
                    0018                .IIF    NB,DDT$L_UNITINIT,      DDT$L_UNITINIT:
          0000001C  0018                        .BLKL   1
                    001C                .IIF    NB,DDT$L_ALTSTART,      DDT$L_ALTSTART:
          00000020  001C                        .BLKL   1
                00000000                .RESTORE
                    0000    62          $MBADEF                         ;DEFINE MBA REGISTER OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  0070                .=0
                00000000                .RESTORE
                    0000    63          $IDBDEF                         ;DEFINE IDB OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  0070                .=0
                    0000                .IIF    NB,IDB$L_CSR,   IDB$L_CSR:
          00000004  0000                        .BLKL   1
                    0004                .IIF    NB,IDB$L_OWNER, IDB$L_OWNER:
```

```
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,IDB$W_SIZE,  IDB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,IDB$B_TYPE,  IDB$B_TYPE:
0000000B  000A                                  .BLKB   1
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,IDB$W_UNITS, IDB$W_UNITS:
0000000E  000C                                  .BLKW   1
00000010  000E                                  .BLKW   1
          0010                  .IIF    NB,IDB$L_ADP,   IDB$L_ADP:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,IDB$L_UCBLST,        IDB$L_UCBLST:
00000034  0014                                  .BLKL   8
          0034                  .IIF    NB,IDB$C_LENGTH,        IDB$C_LENGTH:
          0034                  .IIF    NB,IDB$K_LENGTH,        IDB$K_LENGTH:
      00000000                  .RESTORE
          0000      64          $UCBDEF                         ;DEFINE UCB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  0070                  .=0
          0000                  .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
          0000                  .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
          0004                  .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
          000C                  .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
          000C                  .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
0000000D  000C                                  .BLKB   1
00000010  000D                                  .BLKB   3
          0010                  .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
          0010                  .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
          0014                  .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
          0014                  .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016  0014                                  .BLKW   1
          0016                  .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018  0016                                  .BLKW   1
          0018                  .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
          0018                  .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
0000001A  0018                                  .BLKW   1
          001A                  .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
          001A                  .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
0000001C  001A                                  .BLKW   1
          001C                  .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
00000020  001C                                  .BLKL   1
          0020                  .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
00000024  0020                                  .BLKL   1
          0024                  .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
```

```
00000028  0024                                  .BLKL   1
          0028                  .IIF    NB,UCB$L_PID,   UCB$L_PID:
0000002C  0028                                  .BLKL   1
          002C                  .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030  002C                                  .BLKL   1
          0030                  .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034  0030                                  .BLKL   1
          0034                  .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039  0038                                  .BLKB   1
          0039                  .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A  0039                                  .BLKB   1
          003A                  .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C  003A                                  .BLKW   1
          003C                  .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
          003C                  .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
          003C                  .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D  003C                                  .BLKB   1
          003D                  .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
          003D                  .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E  003D                                  .BLKB   1
          003E                  .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
          003E                  .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F  003E                                  .BLKB   1
          003F                  .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040  003F                                  .BLKB   1
          0040                  .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                                  .BLKW   1
          004A                  .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
          004A                  .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                                  .BLKB   1
          004B                  .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                                  .BLKB   1
          004C                  .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                  .BLKW   1
          0052                  .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
          0052                  .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                  .BLKB   1
          0053                  .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                  .BLKB   1
          0054                  .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
          005A                  .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
          005C                  .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C                  .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                  .BLKL   1
          0060                  .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
```

```
00000064  0060                                  .BLKL   1
          0064                  .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
          0064                  .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                                  .BLKL   1
          0068                  .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                                  .BLKL   1
          006C                  .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                                  .BLKW   1
          006E                  .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                                  .BLKW   1
          0070                  .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                                  .BLKB   1
          0071                  .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                                  .BLKB   1
          0072                  .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                                  .BLKW   1
          0074                  .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
          0074                  .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074                  . = 0
          0000                  .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                                  .BLKW   1
00000074  0002                  . = 116
          0074                  .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                                  .BLKL   1
          0080                  .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                                  .BLKL   1
          0090                  .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
          0090                  .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090                  . = 116
          0074                  .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                                  .BLKB   1
          0075                  .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                                  .BLKB   1
          0076                  .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                                  .BLKB   1
          0077                  .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                                  .BLKB   1
          0078                  .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                                  .BLKL   1
0000007E  007C                                  .BLKW   1
          007E                  .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                                  .BLKW   1
          0080                  .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                                  .BLKL   1
00000080  0084                  . = 128
00000084  0080                                  .BLKL   1
          0084                  .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
```

```
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                                  .BLKW   1
          008A                  .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                                  .BLKW   1
          008C                  .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C                  .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                  .BLKW   1
          008E                  .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                  .BLKW   1
          0090                  .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                  .BLKW   1
          0092                  .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                  .BLKW   1
          0094                  .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                  .BLKB   1
          0095                  .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                  .BLKB   1
          0096                  .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                  .BLKW   1
00000096  0098                  . = 150
00000098  0096                                  .BLKW   1
          0098                  .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                  .BLKL   1
          00A0                  .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                                  .BLKL   1
          00A4                  .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                                  .BLKW   1
          00A6                  .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                                  .BLKB   1
000000A8  00A7                                  .BLKB   1
00000074  00A8                  . = 116
00000078  0074                                  .BLKL   1
0000007C  0078                                  .BLKL   1
00000080  007C                                  .BLKL   1
00000084  0080                                  .BLKL   1
00000088  0084                                  .BLKL   1
0000008C  0088                                  .BLKL   1
          008C                  .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                  .BLKL   1
00000094  0090                                  .BLKL   1
00000098  0094                                  .BLKL   1
0000009C  0098                                  .BLKL   1
          009C                  .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                  .BLKB   1
          009D                  .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                  .BLKB   1
          009E                  .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                  .BLKB   1
000000A0  009F                                  .BLKB   1
          00A0                  .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                  .BLKB   1
          00A1                  .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                  .BLKB   1
          00A2                  .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
```

```
000000A3  00A2                          .BLKB   1
000000A4  00A3                          .BLKB   1
          00A4          .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                          .BLKB   1
          00A5          .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                          .BLKW   1
000000A8  00A7                          .BLKB   1
          00A8          .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                          .BLKL   1
000000B0  00AC                          .BLKL   1
000000B4  00B0                          .BLKL   1
000000B8  00B4                          .BLKL   1
          00B8          .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                          .BLKL   1
          00BC          .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC          .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC          . = 116
          0074          .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                          .BLKL   1
          0078          .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                          .BLKL   1
          007C          .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                          .BLKW   1
00000080  007E                          .BLKW   1
        00000000        .RESTORE
```

```
                                0000     66          .SBTTL  MASSBUS ADAPTER INTERRUPT DISPATCHER
                                0000     67 ;+
                                0000     68 ; MBA$INT - MASSBUS ADAPTER INTERRUPT DISPATCHER
                                0000     69 ;
                                0000     70 ; THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN INTERRUPT OCCURS
                                0000     71 ; ON A MASSBUS ADAPTER. THE STATE OF THE STACK ON ENTRY IS:
                                0000     72 ;
                                0000     73 ;       00(SP) = ADDRESS OF IDB ADDRESS.
                                0000     74 ;       04(SP) = SAVED R2.
                                0000     75 ;       08(SP) = SAVED R3.
                                0000     76 ;       12(SP) = SAVED R4.
                                0000     77 ;       16(SP) = SAVED R5.
                                0000     78 ;       20(SP) = INTERRUPT PC.
                                0000     79 ;       24(SP) = INTERRUPT PSL.
                                0000     80 ;
                                0000     81 ; INTERRUPT DISPATCHING OCCURS AS FOLLOWS:
                                0000     82 ;
                                0000     83 ;       IF THE INTERRUPTING ADAPTER IS CURRENTLY OWNED AND THE OWNER UNIT
                                0000     84 ;       IS EXPECTING AN INTERRUPT, THEN THAT UNIT IS DISPATCHED FIRST. ALL
                                0000     85 ;       OTHER UNITS ARE DISPATCHED BY READING THE ATTENTION SUMMARY REG-
                                0000     86 ;       ISTER AND SCANNING FOR UNITS THAT HAVE ATTENTION SET. AS EACH UNIT
                                0000     87 ;       IS FOUND, ITS ATTENTION SUMMARY BIT IS CLEARED AND THEN A TEST IS
                                0000     88 ;       MADE TO DETERMINE IF AN INTERRUPT IS EXPECTED ON THE UNIT. IF YES,
                                0000     89 ;       THEN THE DRIVER IS CALLED AT ITS INTERRUPT RETURN ADDRESS. ELSE
                                0000     90 ;       THE DRIVER IS CALLED AT ITS UNSOLICITED INTERRUPT ADDRESS. AS EACH
                                0000     91 ;       CALL TO THE DRIVER RETURNS, THE ATTENTION SUMMARY REGISTER IS RE-
                                0000     92 ;       READ AND AN ATTEMPT IS MADE TO FIND ANOTHER UNIT TO DISPATCH. WHEN
                                0000     93 ;       NO UNITS REQUESTING ATTENTION REMAIN, THE INTERRUPT IS DISMISSED.
                                0000     94 ;-
                                0000     95
                                0000     96 MBA$INT::                                    ;MASSBUS ADAPTER INTERRUPT DISPATCHER
          53    00 BE    D0     0000     97          MOVL    @(SP),R3                    ;GET ADDRESS OF IDB
             54    63    D0     0004     98          MOVL    IDB$L_CSR(R3),R4            ;GET ADDRESS OF CONFIGURATION STATUS REGISTE
 08 A4    00800000 8F    D3     0007     99          BITL    #MBA$M_SR_CBHUNG,MBA$L_SR(R4)   ;CHECK FOR MBA HUNG
                 6D    12     000F    100          BNEQ    50$                             ;BRANCH IF HUNG
          55    04 A3    D0     0011    101          MOVL    IDB$L_OWNER(R3),R5          ;GET OWNER UNIT UCB ADDRESS
                 09    13     0015    102          BEQL    10$                         ;IF EQL NO OWNER
          52    74 A5    9A     0017    103          MOVZBL  UCB$B_SLAVE(R5),R2          ;GET OWNER SLAVE CONTROLLER NUMBER
       21 58 A5    01    E0     001B    104          BBS     #UCB$V_INT,UCB$W_STS(R5),20$ ;IF SET, INTERRUPT EXPECTED
          53    00 BE    D0     0020    105 10$:     MOVL    @(SP),R3                    ;RETRIEVE ADDRESS OF IDB
             54    63    D0     0024    106          MOVL    IDB$L_CSR(R3),R4            ;RETRIEVE MBA CONFIGURATION REGISTER ADDRESS
          08 A4    00    D2     0027    107          MCOML   #0,MBA$L_SR(R4)             ;CLEAR ALL MBA STATUS BITS
       52    0410 C4    D0     002B    108          MOVL    MBA$L_AS(R4),R2             ;READ ATTENTION SUMMARY REGISTER
 52    52    08    00    EA     0030    109          FFS     #0,#8,R2,R2                 ;FIND FIRST UNIT REQUESTING ATTENTION
                 0A    12     0035    110          BNEQ    20$                         ;IF NEQ UNIT FOUND
             5E    04    C0     0037    111          ADDL    #4,SP                       ;REMOVE IDB ADDRESS FROM STACK
             52    8E    7D     003A    112          MOVQ    (SP)+,R2                    ;RESTORE REGISTERS
             54    8E    7D     003D    113          MOVQ    (SP)+,R4                    ;
                 02    00     0040    114          REI                                 ;
       55    14 A342    D0     0041    115 20$:     MOVL    IDB$L_UCBLST(R3)[R2],R5     ;GET ADDRESS OF UCB OR INTERRUPT DISPATCHER
                2F 55    E8     0046    116          BLBS    R5,40$                      ;IF LBS INTERRUPT DISPATCHER FOR MULTI-
                         0049    117                                                  ; DEVICE CONTROLLER
 0410 C4    01    52    78     0049    118          ASHL    R2,#1,MBA$L_AS(R4)          ;CLEAR ATTENTION SUMMARY BIT
                 55    D5     004F    119          TSTL    R5                          ;SEE IF UCB DEFINED
                 CD    13     0051    120          BEQL    10$                         ;IF EQL NONE DEFINED
       09 58 A5    01    E5     0053    121          BBCC    #UCB$V_INT,UCB$W_STS(R5),30$ ;IF CLR, INTERRUPT NOT EXPECTED
          53    10 A5    7D     0058    122          MOVQ    UCB$L_FR3(R5),R3            ;RESTORE DRIVER CONTEXT
```

```
                      0C B5   16  005C   123          JSB     @UCB$L_FPC(R5)          ;CALL DRIVER AT INTERRUPT RETURN ADDRESS
                         BF   11  005F   124          BRB     10$                     ;
                53    24 A5   D0  0061   125 30$:     MOVL    UCB$L_DDB(R5),R3        ;GET ADDRESS OF DDB
                53    0C A3   D0  0065   126          MOVL    DDB$L_DDT(R3),R3        ;GET ADDRESS OF DDT
                52    04 A3   D0  0069   127          MOVL    DDT$L_UNSOLINT(R3),R2   ;CALCULATE ADDRESS OF UNSOLICITED INTERRUPT
                03 52    10   E0  006D   128          BBS     #16,R2,35$              ;**** BRANCH IF SUPERVISOR ADDR
                   52    53   C0  0071   129          ADDL    R3,R2                   ;**** MAKE ABSOLUTE ADDRESS
                         62   16  0074   130 35$:     JSB     (R2)                    ;CALL UNSOLICITED INTERRUPT ROUTINE
                         A8   11  0076   131          BRB     10$                     ;
                         7E   DC  0078   132 40$:     MOVPSL  -(SP)                   ;READ CURRENT PSL
                         75   16  007A   133          JSB     -(R5)                   ;CALL SLAVE CONTROLLER INTERRUPT DISPATCHER
                         A2   11  007C   134          BRB     10$                     ;
                              007E   135
                              007E   136 ;
                              007E   137 ; IN CASE OF CBHUNG SAVE MBA INFORMATIION FOR BUGCHECK LOG AND
                              007E   138 ; BUGCHECK.  CBHUNG IS IMPLEMENTED ONLY ON THE VAX 11/750 CPU.
                              007E   139 ; IT MEANS THAT AN ACCESS TO A REGISTER OF AN EXISTENT CONTROLLER
                              007E   140 ; FAILED TO COMPLETE IN 1.5 USEC.
                              007E   141 ;
                              007E   142
                55    04 A3   D0  007E   143 50$:     MOVL    IDB$L_OWNER(R3),R5          ;SAVE OWNER UCB IF ANY
                50    08 A4   D0  0082   144          MOVL    MBA$L_SR(R4),R0             ;SAVE MBA STATUS REGISTER,
                   51    64   D0  0086   145          MOVL    MBA$L_CSR(R4),R1            ; CONFIGURATION REGISTER,
                52    14 A4   D0  0089   146          MOVL    MBA$L_DR(R4),R2             ; DIAGNOSTIC REGISTER,
                53    10 A3   D0  008D   147          MOVL    IDB$L_ADP(R3),R3            ;GET ADP ADDRESS
                53    0C A3   3C  0091   148          MOVZWL  ADP$W_TR(R3),R3             ;SAVE NEXUS NUMBER TO IDENTIFY
                              0095   149                                              ; OFFENDING MBA
                              0095   150          BUG_CHECK MBACBHUNG,FATAL           ;FATAL ERROR
             00000000'9F      16  0095          JSB     @#BUG$CHECK
46 2C 47 4E 55 48 42 43 41 42 4D 00' 009B          .ASCIC  "MBACBHUNG,FATAL"
                        4C 41 54 41  00A7
                              0F   009B
```

```
                                00AB          .SBTTL  MASSBUS ADAPTER INITIALIZATION
                                00AB    ;+
                                00AB    ; MBA$INITIAL - MASSBUS ADAPTER INITIALIZATION
                                00AB    ;
                                00AB    ; THIS ROUTINE IS CALLED VIA A JSB INSTRUCTION AT SYSTEM STARTUP AND AFTER
                                00AB    ; A POWER RECOVERY RESTART TO ALLOW INITIALIZATION OF MASSBUS ADAPTERS.
                                00AB    ;
                                00AB    ; INPUTS:
                                00AB    ;
                                00AB    ;       R4 = CSR ADDRESS OF MASSBUS ADAPTER.
                                00AB    ;       R5 = ADDRESS OF ADAPTER IDB.
                                00AB    ;
                                00AB    ;       ALL INTERRUPTS ARE LOCKED OUT.
                                00AB    ;
                                00AB    ; OUTPUTS:
                                00AB    ;
                                00AB    ;       THE MASSBUS ADAPTER IS INITIALIZED AND INTERRUPTS ARE ENABLED.
                                00AB    ;-
                                00AB
                                00AB    MBA$INITIAL::                           ;MASSBUS ADAPTER INITIALIZATION
        04 A4    01    D0      00AB            MOVL    #MBA$M_CR_INIT,MBA$L_CR(R4) ;INITIALIZE MASSBUS ADAPTER
        04 A4    04    D0      00AF            MOVL    #MBA$M_CR_IE,MBA$L_CR(R4) ;ENABLE INTERRUPTS
                         05      00B3            RSB                             ;
                                00B4
                                00B4            .END
```

```
ADP$B_NUMBER      0000000B    D          IDB$W_UNITS       0000000C    D          UCB$L_DUETIM      0000005C    D
ADP$B_PORT        00000020    D          MBA$INITIAL       000000AB RG D  01      UCB$L_DX_BFPNT    0000009C    D
ADP$B_TYPE        0000000A    D          MBA$INT           00000000 RG D  01      UCB$L_DX_BUF      00000098    D
ADP$C_DRADPLEN    00000014    D          MBA$L_AS        = 00000410    D          UCB$L_DX_RXDB     000000A0    D
ADP$C_MBAADPLEN   00000014    D          MBA$L_CR        = 00000004    D          UCB$L_EMB         00000078    D
ADP$C_MPMADPLEN   00000070    D          MBA$L_CSR       = 00000000    D          UCB$L_FIRST       00000014    D
ADP$C_UBAADPLEN   00000070    D          MBA$L_DR        = 00000014    D          UCB$L_FPC         0000000C    D
ADP$K_DRADPLEN    00000014    D          MBA$L_SR        = 00000008    D          UCB$L_FQBL        00000004    D
ADP$K_MBAADPLEN   00000014    D          MBA$M_CR_IE     = 00000004    D          UCB$L_FQFL        00000000    D
ADP$K_MPMADPLEN   00000070    D          MBA$M_CR_INIT   = 00000001    D          UCB$L_FR3         00000010    D
ADP$K_UBAADPLEN   00000070    D          MBA$M_SR_CBHUNG = 00800000    D          UCB$L_FR4         00000014    D
ADP$L_CRB         00000010    D          SIZ...          = 00000002    D          UCB$L_IOQBL       00000044    D
ADP$L_CSR         00000000    D          UCB$B_AMOD        00000053    D          UCB$L_IOQFL       00000040    D
ADP$L_DPQBL       00000018    D          UCB$B_CEX         00000077    D          UCB$L_IRP         0000004C    D
ADP$L_DPQFL       00000014    D          UCB$B_CM1         0000004A    D          UCB$L_LINK        0000002C    D
ADP$L_INTD        00000064    D          UCB$B_CM2         0000004B    D          UCB$L_LOGADR      00000064    D
ADP$L_LINK        00000004    D          UCB$B_DEVCLASS    00000038    D          UCB$L_MAXBLOCK    00000084    D
ADP$L_MRQBL       00000020    D          UCB$B_DEVTYPE     00000039    D          UCB$L_MB_MBX      0000007C    D
ADP$L_MRQFL       0000001C    D          UCB$B_DIPL        00000052    D          UCB$L_MB_PORT     0000008C    D
ADP$L_PRQQBL      00000018    D          UCB$B_DX_SCTCNT   000000A6    D          UCB$L_MB_RAST     00000078    D
ADP$L_PRQQFL      00000014    D          UCB$B_ERTCNT      00000070    D          UCB$L_MB_SHB      00000080    D
ADP$L_SHB         0000001C    D          UCB$B_ERTMAX      00000071    D          UCB$L_MB_WAST     00000074    D
ADP$L_VECTOR      00000010    D          UCB$B_FEX         00000076    D          UCB$L_MB_WIOQBL   00000088    D
ADP$W_ADPTYPE     0000000E    D          UCB$B_FIPL        0000000B    D          UCB$L_MB_WIOQFL   00000084    D
ADP$W_DPBITMAP    00000024    D          UCB$B_LOCSRV      0000003C    D          UCB$L_MEDIA       0000008C    D
ADP$W_MRBITMAP    00000026    D          UCB$B_OFFNDX      00000094    D          UCB$L_NT_DATSSB   00000074    D
ADP$W_SIZE        00000008    D          UCB$B_OFFRTC      00000095    D          UCB$L_NT_INTSSB   00000078    D
ADP$W_TR          0000000C    D          UCB$B_REMSRV      0000003D    D          UCB$L_OPCNT       00000060    D
BUG$CHECK         ********  X  01       UCB$B_SECTORS     0000003C    D          UCB$L_OWNUIC      0000001C    D
DDB$B_ACPCLASS    00000013    D          UCB$B_SLAVE       00000074    D          UCB$L_PID         00000028    D
DDB$B_TYPE        0000000A    D          UCB$B_SPR         00000075    D          UCB$L_RQBL        00000004    D
DDB$C_LENGTH      00000034    D          UCB$B_STATE       00000052    D          UCB$L_RQFL        00000000    D
DDB$K_LENGTH      00000034    D          UCB$B_TRACKS      0000003D    D          UCB$L_SVAPTE      00000068    D
DDB$L_ACPD        00000010    D          UCB$B_TT_CRFILL   0000009D    D          UCB$L_SVPN        00000064    D
DDB$L_DDT         0000000C    D          UCB$B_TT_DECRF    000000A1    D          UCB$L_TT_DECHAR   000000A8    D
DDB$L_LINK        00000000    D          UCB$B_TT_DELFF    000000A2    D          UCB$L_TT_RDUE     0000008C    D
DDB$L_UCB         00000004    D          UCB$B_TT_DESPEE   000000A0    D          UCB$L_TT_RTIMOU   000000B8    D
DDB$T_DRVNAME     00000024    D          UCB$B_TT_DETYPE   000000A4    D          UCB$L_VCB         00000030    D
DDB$T_NAME        00000014    D          UCB$B_TT_LFFILL   0000009E    D          UCB$T_PARTNER     0000000C    D
DDB$W_SIZE        00000008    D          UCB$B_TT_SPEED    0000009C    D          UCB$V_INT       = 00000001    D
DDT$L_ALTSTART    0000001C    D          UCB$B_TYPE        0000000A    D          UCB$W_BCNT        0000006E    D
DDT$L_CANCEL      0000000C    D          UCB$B_VERTSZ      0000003F    D          UCB$W_BCR         00000096    D
DDT$L_FDT         00000008    D          UCB$C_LENGTH      00000074    D          UCB$W_BOFF        0000006C    D
DDT$L_REGDUMP     00000010    D          UCB$C_MB_LENGTH   00000090    D          UCB$W_BUFQUO      00000018    D
DDT$L_START       00000000    D          UCB$C_TT_LENGTH   000000BC    D          UCB$W_BYTESTOGO   0000003E    D
DDT$L_UNITINIT    00000018    D          UCB$K_LENGTH      00000074    D          UCB$W_CHARGE      0000004A    D
DDT$L_UNSOLINT    00000004    D          UCB$K_MB_LENGTH   00000090    D          UCB$W_CYLINDERS   0000003E    D
DDT$W_DIAGBUF     00000014    D          UCB$K_TT_LENGTH   000000BC    D          UCB$W_DA          0000008C    D
DDT$W_ERRORBUF    00000016    D          UCB$L_AMB         00000054    D          UCB$W_DC          0000008E    D
IDB$B_TYPE        0000000A    D          UCB$L_ASTQBL      00000010    D          UCB$W_DEVBUFSIZ   0000003A    D
IDB$C_LENGTH      00000034    D          UCB$L_ASTQFL      0000000C    D          UCB$W_DEVSTS      0000005A    D
IDB$K_LENGTH      00000034    D          UCB$L_CPID        0000005C    D          UCB$W_DIRSEQ      00000088    D
IDB$L_ADP         00000010    D          UCB$L_CRB         00000020    D          UCB$W_DSTADDR     00000018    D
IDB$L_CSR         00000000    D          UCB$L_DDB         00000024    D          UCB$W_DX_BCR      000000A4    D
IDB$L_OWNER       00000004    D          UCB$L_DEVCHAR     00000034    D          UCB$W_EC1         00000090    D
IDB$L_UCBLST      00000014    D          UCB$L_DEVDEPEND   0000003C    D          UCB$W_EC2         00000092    D
IDB$W_SIZE        00000008    D          UCB$L_DPC         00000080    D          UCB$W_ERRCNT      00000072    D
```

```
UCB$W_FUNC        0000007E    D
UCB$W_MB_SEED     00000000    D
UCB$W_MSGCNT      00000016    D
UCB$W_MSGMAX      00000014    D
UCB$W_NT_CHAN     0000007C    D
UCB$W_OFFSET      0000008A    D
UCB$W_REFC        00000050    D
UCB$W_SIZE        00000008    D
UCB$W_SRCADDR     0000001A    D
UCB$W_STS         00000058    D
UCB$W_TT_DESIZE   000000A5    D
UCB$W_UNIT        00000048    D
UCB$W_VPROT       0000001A    D
```

```
                                        +----------------+
                                        ! Psect synopsis !
                                        +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
SEP                             000000B4  (  180.)  01 (  1.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD    WRT NOVEC LONG
$ABS$                           000000BC  (  188.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL          VALUE        DEFINITION      REFERENCES...
------          -----        ----------      -------------
ADP$W_TR        0000000C                    #-148    (1)
BUG$CHECK       00000000-XR                   150    (1)
DDB$L_DDT       0000000C                    #-126    (1)
DDT$L_UNSOLINT  00000004                    #-127    (1)
IDB$L_ADP       00000010                    #-147    (1)
IDB$L_CSR       00000000                    #-106    (1)    #-98     (1)
IDB$L_OWNER     00000004                    #-101    (1)    #-143    (1)
IDB$L_UCBLST    00000014                    #-115    (1)
MBA$INITIAL     000000AB-R   171     (1)
MBA$INT         00000000-R   96      (1)
MBA$L_AS       =00000410                    #-108    (1)    #-118    (1)
MBA$L_CR       =00000004                    #-172    (1)    #-173    (1)
MBA$L_CSR      =00000000                    #-145    (1)
MBA$L_DR       =00000014                    #-146    (1)
MBA$L_SR       =00000008                    #-107    (1)    #-144    (1)    #-99     (1)
MBA$M_CR_IE    =00000004                    #-173    (1)
MBA$M_CR_INIT  =00000001                    #-172    (1)
MBA$M_SR_CBHUNG=00800000                    #-99     (1)
UCB$B_SLAVE     00000074                    #-103    (1)
UCB$L_DDB       00000024                    #-125    (1)
UCB$L_FPC       0000000C                      123    (1)
UCB$L_FR3       00000010                    #-122    (1)
UCB$V_INT      =00000001                    #-104    (1)    #-121    (1)
UCB$W_STS       00000058                      104    (1)      121    (1)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$ADPDEF         2           59      (1)     59      (1)
$DDBDEF         1           60      (1)     60      (1)
$DDTDEF         1           61      (1)     61      (1)
$DEFINI         1           59      (1)     59      (1)     60      (1)     61      (1)     62      (1)     63      (1)
                                            64      (1)
$IDBDEF         1           63      (1)     63      (1)
$MBADEF         5           62      (1)     62      (1)
$UCBDEF         10          64      (1)     64      (1)
BUG_CHECK       1           150     (1)     150     (1)
```

```
                                        +-------------------------+
                                        ! Opcode Cross Reference !
                                        +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDL            00C0      111     (1)     129     (1)
ASHL            0078      118     (1)
BBCC            00E5      121     (1)
BBS             00E0      104     (1)     128     (1)
BEQL            0013      102     (1)     120     (1)
BITL            00D3      99      (1)
BLBS            00E8      116     (1)
BNEQ            0012      100     (1)     110     (1)
BRB             0011      124     (1)     131     (1)     134     (1)
FFS             00EA      109     (1)
JSB             0016      123     (1)     130     (1)     133     (1)     150     (1)
MCOML           00D2      107     (1)
MOVL            00D0      101     (1)     105     (1)     106     (1)     108     (1)     115     (1)     125     (1)     126     (1)
                          127     (1)     143     (1)     144     (1)     145     (1)     146     (1)     147     (1)     172     (1)
                          173     (1)     97      (1)     98      (1)
MOVPSL          00DC      132     (1)
MOVQ            007D      112     (1)     113     (1)     122     (1)
MOVZBL          009A      103     (1)
MOVZWL          003C      148     (1)
REI             0002      114     (1)
RSB             0005      174     (1)
TSTL            00D5      119     (1)
```

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ASCIC          150     (1)
.END            176     (1)
.ENDM           150     (1)     59      (1)     60      (1)     61      (1)     62      (1)     63      (1)     64      (1)
.IDENT          5       (1)
.NOCROSS        59      (1)     60      (1)     61      (1)     62      (1)     63      (1)     64      (1)
.PAGE           151     (1)     54      (1)     65      (1)
.PSECT          6       (1)
.RESTORE        59      (1)     60      (1)     61      (1)     62      (1)     63      (1)     64      (1)
.SAVE           59      (1)     60      (1)     61      (1)     62      (1)     63      (1)     64      (1)
.SBTTL          152     (1)     66      (1)
.TITLE          4       (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	1	#-144	(1)										
R1	1	#-145	(1)										
R2	12	#-103	(1)	#-108	(1)	109	(1)	#-112	(1)	#-115	(1)	#-118	(1)
		#-127	(1)	128	(1)	#-129	(1)	130	(1)	#-146	(1)		
R3	17	#-101	(1)	#-105	(1)	#-106	(1)	#-115	(1)	#-122	(1)	#-125	(1)
		#-126	(1)	#-127	(1)	#-129	(1)	#-143	(1)	#-147	(1)	#-148	(1)
		#-97	(1)	#-98	(1)								
R4	12	#-106	(1)	#-107	(1)	#-108	(1)	#-113	(1)	#-118	(1)	#-144	(1)
		#-145	(1)	#-146	(1)	#-172	(1)	#-173	(1)	#-98	(1)	#-99	(1)
R5	12	#-101	(1)	#-103	(1)	104	(1)	#-115	(1)	#-116	(1)	#-119	(1)
		121	(1)	#-122	(1)	123	(1)	#-125	(1)	133	(1)	#-143	(1)
SP	6	#-105	(1)	#-111	(1)	#-112	(1)	#-113	(1)	#-132	(1)	#-97	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.27
Command processing	891	00:00:00.23	00:00:00.98
Pass 1	491	00:00:01.15	00:00:02.46
Symbol table sort	0	00:00:00.08	00:00:00.08
Pass 2	26	00:00:00.29	00:00:00.62
Symbol table output	2	00:00:00.03	00:00:00.03
Psect synopsis output	2	00:00:00.00	00:00:00.03
Cross-reference output	4	00:00:00.05	00:00:00.05
Assembler run totals	1440	00:00:01.86	00:00:04.52

The working set limit was 2400 pages.
61016 bytes (120 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 328 non-local and 6 local symbols.
176 source lines were read in Pass 1, producing 31 object records in Pass 2.
41 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	5
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	1
SYS$COMMON:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	12

590 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 14 Frame J9          Sequence 2791
MBAINT                    - MASSBUS ADAPTER INTERRUPT DISPATCHER   11-NOV-1987 17:39:56  VAX/VMS Macro V04-00        Page  19
VAX-11 Macro Run Statistics                                        3-JAN-1985 16:52:42  MBAINT.MAR;1                      (1)

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:MBAINT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element MCHK730.B32
;    0002  0      ! *1     6-FEB-1986 15:19:36 DS ""
;    0003  0      ! DEC/CMS REPLACEMENT HISTORY, Element MCHK730.B32
;    0004  0      %title '*** MCHK730 machine check formatter'
;    0005  0      module mchk730 (                        ! Machine check formatter for NEBULA
;    0006  0                      ident = '06-07'
;    0007  0                      ) =
;    0008  0      !
;    0009  0      ! COPYRIGHT (c) 1979, 1981, 1983
;    0010  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0011  0      !
;    0012  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0013  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0014  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0015  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0016  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0017  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0018  0      ! REMAIN IN DEC.
;    0019  0      !
;    0020  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0021  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0022  0      ! CORPORATION.
;    0023  0      !
;    0024  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0025  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0026  0
;    0027  0      !++
;    0028  0      ! FACILITY:
;    0029  0      !
;    0030  0      !       DIAGNOSTIC SUPERVISOR
;    0031  0      !
;    0032  0      ! ABSTRACT:
;    0033  0      !
;    0034  0      !       Provide for the formatting of NEBULA Machine check logout.
;    0035  0      !
;    0036  0      ! AUTHOR:
;    0037  0      !
;    0038  0      !       Roger Riggs, CREATION DATE: 20-AUG-1979
;    0039  0      !
;    0040  0      ! MODIFIED BY:
;    0041  0      !       02      - Roger Riggs, 20-Jun-1980
;    0042  0      !               Modified CHR$MCHK parameters from EXCEPT and updated
;    0043  0      !               to reflect change in the machine check logout for NEBULA.
;    0044  0      !
;    0045  0      !       03      - Dave Butenhof, 19-Nov-1980, version 6.2
;    0046  0      !               Correct signal vector offsets to skip signal vector length
;    0047  0      !               and exception code (DS$_MCHK) instead of printing them
;    0048  0      !               as part of logout.
;    0049  0      !
;    0050  0      !       04      - Dave Butenhof, 28-Apr-1981, version 6.4
;    0051  0      !               Change messages to mixed case, correct spelling errors.
;    0052  0      !
;    0053  0      !       05      - Dave Butenhof, 02-Jun-1981, version 6.4
;    0054  0      !               Change library specification for flexibility
;    0055  0      !
;    0056  0      !       06      - Jack Stansbury, 24-Mar-1982, Version 6.?
;    0057  0      !               Added Library statements for Lib and $DS. Also added typecoding.
```

```
;    0058  0     !
;    0059  0     !       07      Bob Bergazzi    May 17, 1983    Version 6.11
;    0060  0     !               Changed the order of searching libraries.
;    0061  0     !--
;    0062  0
;    0063  1     begin
;    0064  1     !
;    0065  1     ! TABLE OF CONTENTS:
;    0066  1     !
;    0067  1
;    0068  1     linkage
;    0069  1         quick = jsb;
;    0070  1
;    0071  1     forward routine
;    0072  1         chr$mchk : novalue;                     ! FORMATTER FOR MACHINE LOGOUT
;    0073  1
;    0074  1     forward routine
;    0075  1         dsr$fault_clear : quick novalue;        ! RESET MACHINE CHECK
;    0076  1
;    0077  1     !
;    0078  1     ! INCLUDE FILES:
;    0079  1     !
;    0080  1
;    0081  1     Library '$Diag';                !                                                       [07]
;    0082  1
;    0083  1     Library '$DS';                  !                                                       [07]
;    0084  1
;    0085  1     Library 'Sys$Library:Lib';      !                                                       [07]
;    0086  1
;    0087  1     !
;    0088  1     ! MACROS
;    0089  1     !
;    0090  1
;    0091  1     $DS_TypeDef;                    ! Define typecodes                                      [06]
;    0092  1
;    0093  1     !
;    0094  1     ! EXTERNAL REFERENCES:
;    0095  1     !
;    0096  1
;    0097  1     external
;    0098  1         Apsimne         :       Addressing_Mode (Long_Relative),    ! PSL MNEMONICS
;    0099  1         Modelst         :       Addressing_Mode (Long_Relative),    ! PROCESSOR MODE DECODE LIST
;    0100  1         DS$GB_TypeCode : Byte Addressing_Mode (Long_Relative);      ! The typecode number byte  [06]
;    0101  1
;    0102  1     !
;    0103  1     ! PSECT DEFINITIONS:
;    0104  1     !
;    0105  1
;    0106  1     psect
;    0107  1         plit = data ( share, addressing_mode (long_relative));
;    0108  1
;    0109  1     psect
;    0110  1         code = code ( read, execute, share);
;    0111  1
```

```
%sbttl 'CHK$MCHK'

global routine chr$mchk (signal, hdrprint, txtprint) : novalue =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Using the Machine logout information on the stack
!       and the type code format each longword of the logout area.
!
! FORMAL PRAMETERS:
!
!       SIGNAL:   Address of signal array for machine check
!       HDRPRINT: Address of print routine to print hdr's
!       TXTPRINT: Address of print routine to print text
!
!       Signal array is
!       CONDITION Value DS$_MCHK
!       COUNT   Bytes pushed by machine check
!       TYPE    Type code of machine check
!       P1      1st parameter
!       P2      2nd parameter
!       PC      PC of machine check
!       PSL     PSL at time of machine check
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION CODES:
!
!       N/A (No value is returned)
!
! SIDE EFFECTS:
!
!       The machine check error status is cleared
!--
```

```
;    0155  2          begin
;    0156  2
;    0157  2          bind
;    0158  2              t_mchk = $ascic ('!/!/?? Machine check exception through vector: 04(X)!/'),     !                [04]
;    0159  2              t_count = $ascic ('count:!_!_!_!XL(X)!/'),       !                                            [04]
;    0160  2              t_type = $ascic ('Machine check type:!_!XL(X)!_; !AC!/'),       !                             [04]
;    0161  2              t_va = $ascic ('Virtual address:!_!XL(X)!/'),    !                                            [04]
;    0162  2              t_phy = $ascic ('Physical address:!_!XL(X)!/'),  !                                            [04]
;    0163  2              t_tb = $ascic ('TB entry:!_!_!XL(X)!/'),         !                                           [04]
;    0164  2              t_pte = $ascic ('PTE address:!_!_!XL(X)!/'),     !                                           [04]
;    0165  2              t_mcsr = $ascic ('Memory CSR:!_!_!XL(X)!/'),     !                                           [04]
;    0166  2              t_int = $ascic ('Interrupt identifier:!_!XL(X)!/'),     !                                    [04]
;    0167  2              t_p1 = $ascic ('Error parameter 1:!_!XL(X)!/'),  !                                           [04]
;    0168  2              t_p2 = $ascic ('Error parameter 2:!_!XL(X)!/'),  !                                           [04]
;    0169  2              t_sub = $ascic ('Subtype:!_!_!_!XL(X)!/'),       !                                           [04]
;    0170  2              t_ecc = $ascic ('PFN/Syndrome:!_!_!XL(X)!/'),    !                                           [04]
;    0171  2              t_row = $ascic ('FPA row number!_!_!XL(X)!/'),   !                                           [04]
;    0172  2              t_unknown = $ascic ('Unknown machine check type'),      !                                    [04]
;    0173  2              list_type = uplit long(                          !
;    0174  2                          $ascic('micro code should not be here'),       !                               [04]
;    0175  2                          $ascic('translation buffer parity error'),     !                               [04]
;    0176  2                          $ascic('undefined interrupt identifier'),      !                               [04]
;    0177  2                          $ascic('illegal memory CSR'),   !                                              [04]
;    0178  2                          $ascic('fast interrupt without support'),      !                               [04]
;    0179  2                          $ascic('FPA parity error'),     !                                              [04]
;    0180  2                          $ascic('error on SPTE read'),   !                                              [04]
;    0181  2                          $ascic('uncorrectable ECC error'),      !                                      [04]
;    0182  2                          $ascic('reference to non-existant memory'),     !                              [04]
;    0183  2                          $ascic('unaligned or non-longword ref to I/O space'),   !                      [04]
;    0184  2                          $ascic('illegal I/O space address'),    !                                      [04]
;    0185  2                          $ascic('illegal UNIBUS reference'),     !                                      [04]
;    0186  2                          t_unknown,                              !
;    0187  2                          t_unknown,                              !
;    0188  2                          t_unknown,                              !
;    0189  2                          'illegal UNIBUS reference',             !                                      [04]
;    0190  2                          t_unknown,                              !
;    0191  2                          t_unknown,                              !
;    0192  2                          t_unknown,                              !
;    0193  2                          t_unknown) : vector [, long],           !
;    0194  2              list_p1 = uplit long(t_sub, t_va, t_int, t_va,      !
;    0195  2                          t_p1,t_row,t_phy,t_ecc,                 !
;    0196  2                          t_phy,t_phy,t_phy,t_phy,                !
;    0197  2                          t_p1, t_row, t_phy, t_ecc,              !
;    0198  2                          t_phy, t_phy, t_phy, t_phy,             !
;    0199  2                          t_p1, t_p1, t_p1, t_p1) : vector [, long],      !
;    0200  2              list_p2 = uplit long(t_p2, t_tb, t_p2, t_mcsr,      !
;    0201  2                          t_p2,t_p2,t_mcsr,t_p2,                  !
;    0202  2                          t_p2,t_p2,t_p2,t_p2,                    !
;    0203  2                          t_p2, t_p2, t_mcsr, t_p2,               !
;    0204  2                          t_p2, t_p2, t_p2, t_p2, t_p2,           !
;    0205  2                          t_p2, t_p2, t_p2) : vector [, long];    !
;    0206  2
;    0207  2          local
;    0208  2              index,
;    0209  2              buffer : vector [128, byte];
;    0210  2
;    0211  2          map
```

```
;   0212  2              signal : ref vector [, long];
;   0213  2
;   0214  2          DS$GB_TypeCode = DS$K_Type_Exception_Head;          ! Print the exception header message                [06]
;   0215  2          (.hdrprint) (t_mchk);
;   0216  2          DS$GB_TypeCode = DS$K_Type_Exception;               ! Print the exception text messages                 [06]
;   0217  2          (.txtprint) (t_count, .signal [2]);
;   0218  2          index = (if .signal [3] gtr 13 then 2 else .signal [3]);
;   0219  2          (.txtprint) (t_type, .signal [3], .list_type [.index]);
;   0220  2          (.txtprint) (.list_p1 [.index], .signal [4]);
;   0221  2          (.txtprint) (.list_p2 [.index], .signal [5]);
;   0222  2          DS$GB_TypeCode = 0;                                 ! Clear out the typecode byte                       [06]
;   0223  1          end;

                                                        .TITLE  MCHK730 *** MCHK730 machine check formatter
                                                        .IDENT  \06-07\

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

65 6E 69 68 63 61 4D 20 3F 3F 2F 21 2F 21 36  00000 P.AAA:  .ASCII  \6!/!/?? Machine check exception through \  ;
6F 69 74 70 65 63 78 65 20 6B 63 65 68 63 20  0000F                                                           ;
                  20 68 67 75 6F 72 68 74 20 6E  0001E                                                           ;
2F 21 29 58 28 34 30 20 3A 72 6F 74 63 65 76  00028          .ASCII  \vector: 04(X)!/\                         ;
58 21 5F 21 5F 21 5F 21 3A 74 6E 75 6F 63 14  00037 P.AAB:  .ASCII  <20>\count:!_!_!_!XL(X)!/\                  ;
                           2F 21 29 58 28 4C  00046                                                           ;
20 6B 63 65 68 63 20 65 6E 69 68 63 61 4D 24  0004C P.AAC:  .ASCII  \$Machine check type:!_!XL(X)!_; !AC!/\      ;
5F 21 29 58 28 4C 58 21 5F 21 3A 65 70 79 74  0005B                                                           ;
                        2F 21 43 41 21 20 3B  0006A                                                           ;
73 65 72 64 64 61 20 6C 61 75 74 72 69 56 1A  00071 P.AAD:  .ASCII  <26>\Virtual address:!_!XL(X)!/\           ;
   2F 21 29 58 28 4C 58 21 5F 21 3A 73 73     00080                                                           ;
65 72 64 64 61 20 6C 61 63 69 73 79 68 50 1B  0008C P.AAE:  .ASCII  <27>\Physical address:!_!XL(X)!/\         ;
   2F 21 29 58 28 4C 58 21 5F 21 3A 73 73     0009B                                                           ;
21 5F 21 5F 21 3A 79 72 74 6E 65 20 42 54 15  000A8 P.AAF:  .ASCII  <21>\TB entry:!_!_!XL(X)!/\               ;
                           2F 21 29 58 28 4C  000B7                                                           ;
5F 21 3A 73 73 65 72 64 64 61 20 45 54 50 18  000BE P.AAG:  .ASCII  <24>\PTE address:!_!_!XL(X)!/\           ;
                  2F 21 29 58 28 4C 58 21 5F 21  000CD                                                           ;
21 5F 21 3A 52 53 43 20 79 72 6F 6D 65 4D 17  000D7 P.AAH:  .ASCII  <23>\Memory CSR:!_!_!XL(X)!/\            ;
                        2F 21 29 58 28 4C 58 21 5F  000E6                                                           ;
6E 65 64 69 20 74 70 75 72 72 65 74 6E 49 1F  000EF P.AAI:  .ASCII  <31>\Interrupt identifier:!_!XL(X)!/\     ;
29 58 28 4C 58 21 5F 21 3A 72 65 69 66 69 74  000FE                                                           ;
                                       2F 21  0010D                                                           ;
65 74 65 6D 61 72 61 70 20 72 6F 72 72 45 1C  0010F P.AAJ:  .ASCII  <28>\Error parameter 1:!_!XL(X)!/\       ;
   2F 21 29 58 28 4C 58 21 5F 21 3A 31 20 72  0011E                                                           ;
65 74 65 6D 61 72 61 70 20 72 6F 72 72 45 1C  0012C P.AAK:  .ASCII  <28>\Error parameter 2:!_!XL(X)!/\       ;
   2F 21 29 58 28 4C 58 21 5F 21 3A 32 20 72  0013B                                                           ;
5F 21 5F 21 5F 21 3A 65 70 79 74 62 75 53 16  00149 P.AAL:  .ASCII  <22>\Subtype:!_!_!_!XL(X)!/\            ;
                        2F 21 29 58 28 4C 58 21  00158                                                           ;
21 3A 65 6D 6F 72 64 6E 79 53 2F 4E 46 50 19  00160 P.AAM:  .ASCII  <25>\PFN/Syndrome:!_!_!XL(X)!/\         ;
                  2F 21 29 58 28 4C 58 21 5F 21 5F  0016F                                                           ;
72 65 62 6D 75 6E 20 77 6F 72 20 41 50 46 1A  0017A P.AAN:  .ASCII  <26>\FPA row number!_!_!XL(X)!/\        ;
   2F 21 29 58 28 4C 58 21 5F 21 5F 21        00189                                                           ;
6E 69 68 63 61 6D 20 6E 77 6F 6E 6B 6E 55 1A  00195 P.AAO:  .ASCII  <26>\Unknown machine check type\         ;
   65 70 79 74 20 6B 63 65 68 63 20 65        001A4                                                           ;
6F 68 73 20 65 64 6F 63 20 6F 72 63 69 6D 1D  001B0 P.AAQ:  .ASCII  <29>\micro code should not be here\       ;
65 72 65 68 20 65 62 20 74 6F 6E 20 64 6C 75  001BF                                                           ;
75 62 20 6E 6F 69 74 61 6C 73 6E 61 72 74 1F  001CE P.AAR:  .ASCII  <31>\translation buffer parity error\     ;
72 72 65 20 79 74 69 72 61 70 20 72 65 66 66  001DD                                                           ;
```

```
                                                                72  6F  001EC
65  74  6E  69  20  64  65  6E  69  66  65  64  6E  75  1E  001EE P.AAS:  .ASCII  <30>\undefined interrupt identifier\
65  69  66  69  74  6E  65  64  69  20  74  70  75  72  72  001FD
                                                                72  0020C
79  72  6F  6D  65  6D  20  6C  61  67  65  6C  6C  69  12  0020D P.AAT:  .ASCII  <18>\illegal memory CSR\
                                                            52  53  43  20  0021C
74  70  75  72  72  65  74  6E  69  20  74  73  61  66  1E  00220 P.AAU:  .ASCII  <30>\fast interrupt without support\
72  6F  70  70  75  73  20  74  75  6F  68  74  69  77  20  0022F
                                                                74  0023E
72  72  65  20  79  74  69  72  61  70  20  41  50  46  10  0023F P.AAV:  .ASCII  <16>\FPA parity error\
                                                                72  6F  0024E
20  45  54  50  53  20  6E  6F  20  72  6F  72  72  65  12  00250 P.AAW:  .ASCII  <18>\error on SPTE read\
                                                            64  61  65  72  0025F
20  65  6C  62  61  74  63  65  72  72  6F  63  6E  75  17  00263 P.AAX:  .ASCII  <23>\uncorrectable ECC error\
                        72  6F  72  72  65  20  43  43  45  00272
6E  20  6F  74  20  65  63  6E  65  72  65  66  65  72  20  0027B P.AAY:  .ASCII  \ reference to non-existant memory\
6D  65  6D  20  74  6E  61  74  73  69  78  65  2D  6E  6F  0028A
                                                            79  72  6F  00299
6E  20  72  6F  20  64  65  6E  67  69  6C  61  6E  75  2A  0029C P.AAZ:  .ASCII  \*unaligned or non-longword ref to I/O sp\
66  65  72  20  64  72  6F  77  67  6E  6F  6C  2D  6E  6F  002AB
                        70  73  20  4F  2F  49  20  6F  74  20  002BA
                                                            65  63  61  002C4      .ASCII  \ace\
70  73  20  4F  2F  49  20  6C  61  67  65  6C  6C  69  19  002C7 P.ABA:  .ASCII  <25>\illegal I/O space address\
                    73  73  65  72  64  64  61  20  65  63  61  002D6
53  55  42  49  4E  55  20  6C  61  67  65  6C  6C  69  18  002E1 P.ABB:  .ASCII  <24>\illegal UNIBUS reference\
                        65  63  6E  65  72  65  66  65  72  20  002F0
                                                                002FA      .BLKB   2
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  002FC P.AAP:  .ADDRESS P.AAQ, P.AAR, P.AAS, P.AAT, P.AAU, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  00314              P.AAV, P.AAW, P.AAX, P.AAY, P.AAZ, P.ABA, -
                    00000000' 00000000' 00000000' 00000000'  0032C              P.ABB, T_UNKNOWN, T_UNKNOWN, T_UNKNOWN
20  53  55  42  49  4E  55  20  6C  61  67  65  6C  6C  69   00338      .ASCII  \illegal UNIBUS reference\
                        65  63  6E  65  72  65  66  65  72   00347
                    00000000' 00000000' 00000000' 00000000'  00350      .ADDRESS T_UNKNOWN, T_UNKNOWN, T_UNKNOWN, -
                                                                             T_UNKNOWN
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  00360 P.ABC:  .ADDRESS T_SUB, T_VA, T_INT, T_VA, T_P1, T_ROW, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  00378              T_PHY, T_ECC, T_PHY, T_PHY, T_PHY, T_PHY, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  00390              T_P1, T_ROW, T_PHY, T_ECC, T_PHY, T_PHY, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  003A8              T_PHY, T_PHY, T_P1, T_P1, T_P1, T_P1
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  003C0 P.ABD:  .ADDRESS T_P2, T_TB, T_P2, T_MCSR, T_P2, T_P2, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  003D8              T_MCSR, T_P2, T_P2, T_P2, T_P2, T_P2, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  003F0              T_P2, T_P2, T_MCSR, T_P2, T_P2, T_P2, -
00000000' 00000000' 00000000' 00000000' 00000000' 00000000'  00408              T_P2, T_P2, T_P2, T_P2, T_P2, T_P2

T_MCHK=          P.AAA
T_COUNT=         P.AAB
T_TYPE=          P.AAC
T_VA=            P.AAD
T_PHY=           P.AAE
T_TB=            P.AAF
T_PTE=           P.AAG
T_MCSR=          P.AAH
T_INT=           P.AAI
T_P1=            P.AAJ
T_P2=            P.AAK
T_SUB=           P.AAL
T_ECC=           P.AAM
T_ROW=           P.AAN
```

```
                                                T_UNKNOWN=          P.AAO
                                                LIST_TYPE=          P.AAP
                                                LIST_P1=            P.ABC
                                                LIST_P2=            P.ABD
                                                        .EXTRN  APSLMNE, MODELST
                                                        .EXTRN  DS$GB_TYPECODE

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                    003C 00000          .ENTRY  CHR$MCHK, Save R2,R3,R4,R5                      ; 0115
                55 00000000G   EF   9E 00002            MOVAB   DS$GB_TYPECODE, R5                              ;
                54 00000000'   EF   9E 00009            MOVAB   T_MCHK, R4                                      ;
                5E        80   AE   9E 00010            MOVAB   -128(SP), SP                                    ;
                65             0B   90 00014            MOVB    #11, DS$GB_TYPECODE                             ; 0214
                               54   DD 00017            PUSHL   R4                                              ; 0215
          08    BC             01   FB 00019            CALLS   #1, @HDRPRINT                                   ;
                65             0C   90 0001D            MOVB    #12, DS$GB_TYPECODE                             ; 0216
                53        04   AC   D0 00020            MOVL    SIGNAL, R3                                      ; 0217
                          08   A3   DD 00024            PUSHL   8(R3)                                           ;
                          37   A4   9F 00027            PUSHAB  T_COUNT                                         ;
          0C    BC             02   FB 0002A            CALLS   #2, @TXTPRINT                                   ;
                0D        0C   A3   D1 0002E            CMPL    12(R3), #13                                     ; 0218
                               05   15 00032            BLEQ    1$                                              ;
                52             02   D0 00034            MOVL    #2, INDEX                                       ;
                               04   11 00037            BRB     2$                                              ;
                52        0C   A3   D0 00039 1$:         MOVL    12(R3), INDEX                                   ;
                     02FC C442   DD   0003D 2$:         PUSHL   LIST_TYPE[INDEX]                                ; 0219
                          0C   A3   DD 00042            PUSHL   12(R3)                                          ;
                          4C   A4   9F 00045            PUSHAB  T_TYPE                                          ;
          0C    BC             03   FB 00048            CALLS   #3, @TXTPRINT                                   ;
                          10   A3   DD 0004C            PUSHL   16(R3)                                          ; 0220
                     0360 C442   DD   0004F            PUSHL   LIST_P1[INDEX]                                  ;
          0C    BC             02   FB 00054            CALLS   #2, @TXTPRINT                                   ;
                          14   A3   DD 00058            PUSHL   20(R3)                                          ; 0221
                     03C0 C442   DD   0005B            PUSHL   LIST_P2[INDEX]                                  ;
          0C    BC             02   FB 00060            CALLS   #2, @TXTPRINT                                   ;
                               65   94 00064            CLRB    DS$GB_TYPECODE                                  ; 0222
                                    04 00066            RET                                                     ; 0223

; Routine Size:  103 bytes,    Routine Base:  CODE + 0000

;   0224  1
```

```
;    0226  1     %sbttl 'DSR$FAULT_CLEAR'
;    0227  1
;    0228  1     global routine dsr$fault_clear : quick novalue =
;    0229  1
;    0230  1     !++
;    0231  1     ! FUNCTIONAL DESCRIPTION:
;    0232  1     !
;    0233  1     !       This routine is called to clear the processor dependent
;    0234  1     !       error status.
;    0235  1     !
;    0236  1     ! FORMAL PARAMETERS:
;    0237  1     !
;    0238  1     !       NONE
;    0239  1     !
;    0240  1     ! IMPLICIT INPUTS:
;    0241  1     !
;    0242  1     !       NONE
;    0243  1     !
;    0244  1     ! IMPLICIT OUTPUTS:
;    0245  1     !
;    0246  1     !       NONE
;    0247  1     !
;    0248  1     ! COMPLETION CODES:
;    0249  1     !
;    0250  1     !       NONE
;    0251  1     !
;    0252  1     ! SIDE EFFECTS:
;    0253  1     !
;    0254  1     !       Machine check status is cleared
;    0255  1     !--
```

```
;   0257  1
;   0258  2         begin
;   0259  2
;   0260  2         literal
;   0261  2             pr$_mcesr = %x'26';
;   0262  2
;   0263  2         builtin
;   0264  2             mtpr;
;   0265  2
;   0266  2         mtpr (%ref (0), pr$_mcesr);
;   0267  1         end;



                                  26          00  DA 00000 DSR$FAULT_CLEAR::
                                                                  MTPR    #0, #38                          ; 0266
                                                  05 00003        RSB                                      ; 0267

; Routine Size:  4 bytes,    Routine Base:  CODE + 0067


;   0268  1
;   0269  1     end
;   0270  1
;   0271  0     eludom



;                                  PSECT SUMMARY
;
;       Name                    Bytes                        Attributes
;
;  DATA                          1056  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                           107  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...										
$ASCIC	Macro	Lib01	158	159	160	161	162	163	164	165	166	167	168
			169	170	171	172	174	175	176	177	178	179	180
			181	182	183	184	185						
$DS_TYPEDEF	Macro	Lib01	91										
$EQULST	Macro	Lib03	91										
APSLMNE	External	98											
BUFFER	Local	209											
CHR$MCHK	GlobRout	115	72f										
DS$GB_TYPECODE	External	100	214=	216=	222=								
DS$K_PRINTB	Literal	91											
DS$K_PRINTF	Literal	91											
DS$K_PRINTI	Literal	91											
DS$K_PRINTX	Literal	91											
DS$K_TYPE_ABORT_PROGRAM	Literal	91											
DS$K_TYPE_ABORT_TEST	Literal	91											
DS$K_TYPE_COMMAND_ERR	Literal	91											
DS$K_TYPE_COMMAND_OUT	Literal	91											
DS$K_TYPE_CRD_AUTOTEST	Literal	91											
DS$K_TYPE_DS_PROMPT	Literal	91											
DS$K_TYPE_DS_START	Literal	91											
DS$K_TYPE_ERRDEV	Literal	91											
DS$K_TYPE_ERRHARD	Literal	91											
DS$K_TYPE_ERROR_BODY	Literal	91											
DS$K_TYPE_ERROR_END	Literal	91											
DS$K_TYPE_ERRPREP	Literal	91											
DS$K_TYPE_ERRSOFT	Literal	91											
DS$K_TYPE_ERRSUP	Literal	91											
DS$K_TYPE_ERRSYS	Literal	91											
DS$K_TYPE_ERR_HALT	Literal	91											
DS$K_TYPE_EXCEPTION	Literal	91	216										
DS$K_TYPE_EXCEPTION_HEAD	Literal	91	214										
DS$K_TYPE_FIRST_PASS	Literal	91											
DS$K_TYPE_GENERAL	Literal	91											
DS$K_TYPE_GENERAL_ERROR	Literal	91											
DS$K_TYPE_NO_TESTS	Literal	91											
DS$K_TYPE_PARAM_ERROR	Literal	91											
DS$K_TYPE_PROGRAM_END	Literal	91											
DS$K_TYPE_PROGRAM_INFO	Literal	91											
DS$K_TYPE_PROGRAM_START	Literal	91											
DS$K_TYPE_QIO_INVADP	Literal	91											
DS$K_TYPE_QIO_NODRIVER	Literal	91											
DS$K_TYPE_QIO_WRONGVER	Literal	91											
DS$K_TYPE_SCRIPT_ECHO	Literal	91											
DS$K_TYPE_SCRIPT_PNF	Literal	91											
DS$K_TYPE_SCRIPT_PROMPT	Literal	91											
DS$K_TYPE_SCRIPT_SKIP	Literal	91											
DS$K_TYPE_SEQUENCE_ERROR	Literal	91											
DS$K_TYPE_START_ERR	Literal	91											
DS$K_TYPE_START_LIST	Literal	91											
DS$K_TYPE_SUMMARY	Literal	91											
DS$K_TYPE_USER_PROMPT	Literal	91											
DSR$FAULT_CLEAR	GlobRout	228	75f										
GET1ST_	Macro	Lib03	91										
GET2ND_	Unbound		91										
HDRPRINT	RoutForm	115	215ac										

Symbol	Type	Defined	Referenced ...						
INDEX	Local	208	218=	219.	220.	221.			
LIST_P1	Bind	194	220.						
LIST_P2	Bind	200	221.						
LIST_TYPE	Bind	173	219.						
MODELST	External	99							
MTPR	Builtin	264	266						
NUL2ND	Macro	Lib03	91						
PR$_MCESR	Literal	261	266						
QUICK	Linkage	69	75	228					
SIGNAL	RoutForm	115	212m	217a.	218a.	219a.	220a.	221a.	
TXTPRINT	RoutForm	115	217ac	219ac	220ac	221ac			
T_COUNT	Bind	159	217a						
T_ECC	Bind	170	195a	197a					
T_INT	Bind	166	194a						
T_MCHK	Bind	158	215a						
T_MCSR	Bind	165	200a	201a	203a				
T_P1	Bind	167	195a	197a	199a				
T_P2	Bind	168	200a	201a	202a	203a	204a	205a	
T_PHY	Bind	162	195a	196a	197a	198a			
T_PTE	Bind	164							
T_ROW	Bind	171	195a	197a					
T_SUB	Bind	169	194a						
T_TB	Bind	163	200a						
T_TYPE	Bind	160	219a						
T_UNKNOWN	Bind	172	186a	187a	188a	190a	191a	192a	193a
T_VA	Bind	161	194a						

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]MCHK730.B32;1
5	Module MCHK730	
81	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
83	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
85	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
271	Eludom MCHK730	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
a	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

```
;                              Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                            940         2         0        96          00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                            675         0         0        47          00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2             20450         4         0      1101          00:00.8




;                                    COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:MCHK730.B32

; Size:          107 code + 1056 data bytes
; Run Time:         00:02.4
; Elapsed Time:     00:04.2
; Lines/CPU Min:     6919
; Lexemes/CPU-Min: 99855
; Memory Used:  105 pages
; Compilation Complete
```

Table of contents

(2) 156 ALLOCATE MEMORY AND CONDITIONALLY WAIT
(2) 273 ALLOCATE NONPAGED DYNAMIC MEMORY
(2) 338 GENERAL ALLOCATE MEMORY SUBROUTINE
(2) 376 DEALLOCATE NONPAGED DYNAMIC MEMORY
(2) 431 CHECK BLOCK PARAMETERS SUBROUTINE
(2) 447 GENERAL DEALLOCATION SUBROUTINE
(2) 489 MEMPOOL$INIT ;USER MODE MEMORY INITIALIZATION
(2) 590 RETURN SIZE OF PHYSICAL MEMORY ;[07]

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element MEMALC.MAR
0000     2 ; *2    24-MAR-1986 10:20:09 BAGGETT "REPLACE VERSION WITH ULTRIX BUFFER ALLOCATION"
0000     3 ; *1     6-FEB-1986 15:19:58 DS ""
0000     4 ; DEC/CMS REPLACEMENT HISTORY, Element MEMALC.MAR
0000     5         .TITLE  MEMALC  *** MEMALC dynamic memory allocation
0000     6         .IDENT  /07-09/                                                    ;G:2
0000     7         .NLIST  CND
0000     8
0000     9 ;
0000    10 ;*****************************************************************************
0000    11 ;*                                                                           *
0000    12 ;*  Copyright (c) 1977, 1982, 1986                                          * ;G:2
0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    14 ;*                                                                           *
0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    20 ;*  TRANSFERRED.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    24 ;*  CORPORATION.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    28 ;*                                                                           *
0000    29 ;*****************************************************************************
0000    30 ;
0000    31 ; D. N. CUTLER 3-AUG-76
0000    32 ;       V0206   LMK0003         LEN KAWELL                  02-NOV-1979
0000    33 ;               ADD RESOURCE REPORTING TO SHARED MEMORY DEALLOCATION.
0000    34 ;
0000    35 ;
0000    36 ;
0000    37 ;               N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
0000    38 ;       01      DIAGNOSTIC SUPERVISOR INTEGRATION
0000    39 ;               Roger Riggs     05-Jul-78                   (ESSAA-4.03)
0000    40 ;       02      Removed REMQUE/INSQUE instructions and made
0000    41 ;               corresponding DSBINT/ENBINT macros execute only if in S/A
0000    42 ;               in EXE$ALONONPAGED/EXE$DEANONPAGED
0000    43 ;               Roger Riggs             20-NOV-1979
0000    44 ;       03      Removed L^ from REF to DS$GL_MEMSIZE
0000    45 ;               Dave Butenhof   15-may-1980, Version 5.4
0000    46 ;       04      Create .UPD file to modify VMS V2.0 source to supervisor
0000    47 ;               environment.
0000    48 ;
0000    49 ;       05      - Jack Stansbury, 10-Nov-1981, Version 6.-
0000    50 ;               Fixed truncation errors by replacing all W^ with L^.
0000    51 ;               Added .LIBRARY statements for $DIAG and $DS.
0000    52 ;
0000    53 ;       06      - Dave Butenhof, 05-Feb-1982, version 6.6
0000    54 ;               Make EXE$GL_SPLITADR global for access by SHOW MEMORY,
0000    55 ;               and add DS$GL_LOOKASIDE to remember start of lookaside
0000    56 ;               list (if any).
0000    57 ;
```

```
0000     58 ;       07      Jim Shelhamer   01-AUG-1984
0000     59 ;               Added routine to return the size of physical memory,
0000     60 ;               DS$MEMSIZE.
0000     61 ;
0000     62 ;       08      Ted Salem       March 22, 1985
0000     63 ;               Added code to interlock EXE$ALLONONPAGED and EXE$DEALNONPAGED.
0000     64 ;                                                                              ;G:2
0000     65 ;                                                                              ;G:2
0000     66 ;       09      M. Baggett      7-March-1986                                    ;G:2
0000     67 ;               Added buffer space for ULTRIX use.                              ;G:2
0000     68 ;                                                                              ;G:2
0000     69 ;       V0205   RIH0031         RICHARD I. HUSTVEDT     08-AUG-1979
0000     70 ;               ADD ALLOCATE ROUTINE FOR JIB AND CHANGE USE OF LOOKASIDE
0000     71 ;               LIST TO SATISFY IRP-SIZE AND SMALLER REQUESTS.
0000     72 ;
0000     73 ;       V0204   KDM0034         KATHLEEN D. MORSE       22-MAY-1979
0000     74 ;               ADD ASSUMPTIONS FOR COMMON EVENT BLOCK SIZES.
0000     75 ;
0000     76 ;       V0203   LMK0002         LEN KAWELL              18-MAY-1979
0000     77 ;               ADDED DEALLOCATION OF "SPECIAL" TYPES OF NON-PAGED
0000     78 ;               POOL BLOCKS.
0000     79 ;
0000     80 ;       V0202   LMK0001         LEN KAWELL              25-FEB-1979
0000     81 ;               ADDED SHARED MEMORY POOL ALLOCATION/DEALLOCATION
0000     82 ;
0000     83 ; DYNAMIC MEMORY ALLOCATION
```

```
                        0000     85 ;
                        0000     86 ; LIBRARYS
                        0000     87 ;
                        0000     88          .Library        /Sys$Library:Lib/
                        0000     89          .LIBRARY        /$DS/           ;                                       [05]
                        0000     90          .LIBRARY        /$DIAG/         ;                                       [05]
                        0000     91
                        0000     92 ;
                        0000     93 ; MACRO LIBRARY CALLS
                        0000     94 ;
                        0000     95
                        0000     96          $DS_DSDEF                       ; define return codes                   [07]
                        0000     97          $DS_DSADEF                      ;DEFINE BITS IN DSA$GL_FLAGS
                        0000     98          $CEBDEF                         ;DEFINE COMMON EVENT BLOCKS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 FE70                .=0
                        0000                 .RESTORE
                        0000     99          $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 FE70                .=0
                        0000                 .RESTORE
                        0000    100          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 FE70                .=0
                        0000                 .RESTORE
                        0000    101          $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 FE70                .=0
                        0000                 .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
                00000004 0000                        .BLKL   1
                        0004                 .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
                00000008 0004                        .BLKL   1
                        0008                 .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
                0000000A 0008                        .BLKW   1
                        000A                 .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
                0000000B 000A                        .BLKB   1
                        000B                 .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
                0000000C 000B                        .BLKB   1
                        000C                 .IIF    NB,IRP$L_PID,   IRP$L_PID:
                00000010 000C                        .BLKL   1
                        0010                 .IIF    NB,IRP$L_AST,   IRP$L_AST:
                00000014 0010                        .BLKL   1
                        0014                 .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
                00000018 0014                        .BLKL   1
                        0018                 .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
                0000001C 0018                        .BLKL   1
                        001C                 .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
                00000020 001C                        .BLKL   1
                        0020                 .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
                00000022 0020                        .BLKW   1
                        0022                 .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
                00000023 0022                        .BLKB   1
                        0023                 .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
```

```
00000024  0023                                  .BLKB   1
          0024                  .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
00000028  0024                                  .BLKL   1
          0028                  .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
0000002A  0028                                  .BLKW   1
          002A                  .IIF    NB,IRP$W_STS,   IRP$W_STS:
0000002C  002A                                  .BLKW   1
          002C                  .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
00000030  002C                                  .BLKL   1
          0030                  .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
00000032  0030                                  .BLKW   1
          0032                  .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
00000034  0032                                  .BLKW   1
          0034                  .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
          0034                  .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
          0038                  .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
          0038                  .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
00000039  0038                                  .BLKB   1
0000003C  0039                                  .BLKB   3
          003C                  .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
          003C                  .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
          003C                  .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                                  .BLKW   1
          003E                  .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                                  .BLKW   1
          0040                  .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                                  .BLKL   1
          004C                  .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                                  .BLKL   1
00000058  0054                                  .BLKL   1
0000005C  0058                                  .BLKL   1
          005C                  .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
          005C                  .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
          0000                  .RESTORE
          0000    102           $JIBDEF                         ;DEFINE JIB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  FE70                  .=0
          0000                  .RESTORE
          0000    103           $PCBDEF                         ;DEFINE PCB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  FE70                  .=0
          0000                  .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A  0008                                  .BLKW   1
```

```
                    000A            .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B  000A                              .BLKB   1
                    000B            .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C  000B                              .BLKB   1
                    000C            .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D  000C                              .BLKB   1
                    000D            .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E  000D                              .BLKB   1
                    000E            .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010  000E                              .BLKW   1
                    0010            .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014  0010                              .BLKL   1
                    0014            .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018  0014                              .BLKL   1
                    0018            .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C  0018                              .BLKL   1
                    001C            .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020  001C                              .BLKL   1
                    0020            .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024  0020                              .BLKL   1
                    0024            .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028  0024                              .BLKL   1
                    0028            .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C  0028                              .BLKL   1
                    002C            .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E  002C                              .BLKW   1
                    002E            .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                              .BLKB   1
                    002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                              .BLKB   1
                    0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                              .BLKW   1
                    0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                              .BLKW   1
                    0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                              .BLKW   1
                    0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                              .BLKW   1
                    0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                              .BLKW   1
                    003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                              .BLKW   1
                    003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                              .BLKW   1
                    003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                              .BLKW   1
                    0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                              .BLKW   1
                    0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                              .BLKW   1
                    0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                              .BLKB   8
                    004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                    004C            .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                              .BLKL   1
                    0050            .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                              .BLKL   1
```

```
                        0054                .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
                00000058 0054                       .BLKL   1
                        0058                .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
                0000005C 0058                       .BLKL   1
                        005C                .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
                00000060 005C                       .BLKL   1
                        0060                .IIF    NB,PCB$L_PID,   PCB$L_PID:
                00000064 0060                       .BLKL   1
                        0064                .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
                00000068 0064                       .BLKL   1
                        0068                .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
                00000078 0068                       .BLKB   16
                        0078                .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
                0000007C 0078                       .BLKL   1
                        007C                .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
                00000084 007C                       .BLKQ   1
                        0084                .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
                00000088 0084                       .BLKL   1
                        0088                .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                        0088                .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
                0000008A 0088                       .BLKW   1
                        008A                .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
                0000008C 008A                       .BLKW   1
                        008C                .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                        008C                .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                        0000                .RESTORE
                        0000    104         $PQBDEF                         ;DEFINE PQB OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000    105         $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000    106         $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000    107         $SHBDEF                         ;DEFINE SHARED MEM CONTROL BLOCK
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000    108         $SHDDEF                         ;DEFINE SHARED MEM DATAPAGE
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000    109         $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 FE70               .=0
                        0000                .RESTORE
                        0000    110         $TQEDEF                         ;DEFINE TQE OFFSETS
```

```
                    0000              .SAVE   LOCAL_BLOCK
                    0000              .PSECT  $ABS$,ABS
          00000000  FE70              .=0
                    0000              .RESTORE
                    0000    111       $MP_ILDEFS                          ;DEFINE THE MP$R_ACTIVE_ROUTINES  [08]
                    0000    112 ;
                    0000    113 ; EXTERNALS:
                    0000    114 ;
                    0000    115       .EXTRN  DS$SERVICE_DISPATCHER
                    0000    116
                    0000    117 ;
                    0000    118 ; LOCAL SYMBOLS
                    0000    119 ;
                    0000    120 ; ALLOCATION GRANULARITY MASK
                    0000    121 ;
                    0000    122
          0000000F  0000    123 MASK=^XF                                    ;16 BYTE ALLOCATION GRANULARITY
                    0000    124
                00000000    125       .PSECT  WORK,   NOSHR,  NOEXE,  WRT,    LONG
                    0000    126 ;
                    0000    127 ;OWN STORAGE
                    0000    128 ;
                    0000    129
                    0000    130       .ALIGN  LONG
                    0000    131 EXE$GL_NONPAGED:
          0000000B  0000    132       .LONG   11                          ;DISABLE FORK IPL
          00000000  0004    133       .LONG   0                           ;ADDRESS OF FIRST FREE BLOCK
          00000000  0008    134       .LONG   0                           ;NO BYTES IN BLOCK
                    000C    135
                    000C    136 EXE$GL_SPLITADR::                         ;LOOKASIDE I/O PACKET LIST SPLIT ADDRESS
          00000000  000C    137       .LONG   0                           ;ADDRESS OF LOWEST IRP
                    0010    138 DS$GL_LOOKASIDE::                         ; Address of lookaside start
          00000000  0010    139       .LONG   0
                    0014    140
                    0014    141
                    0014    142 ;
                    0014    143 ;I/O PACKET LOOK ASIDE LISTHEAD
                    0014    144 ;
                    0014    145
                    0014    146 IOC$GL_IRPFL::
    00000014'00000014'  0014    147       .LONG   IOC$GL_IRPFL,IOC$GL_IRPFL   ;BACKWARD LINK
              00000018  001C    148 IOC$GL_IRPBL=IOC$GL_IRPFL+4
                    001C    149
                    001C    150
                00000000    151       .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                    0000    152       MODNAM  MEMALC
    43 4C 41 4D 45 4D 00'   0000  $MODULE:       .ASCIC  \MEMALC\
                      06    0000
                    0007    153 T_CGF:
20 54 45 47 20 54 4F 4E 4E 41 43 00'  0007    154       .ASCIC  .CANNOT GET FREE CORE.
      45 52 4F 43 20 45 45 52 46      0013
                      14    0007
```

```
                     001C   156          .SBTTL  ALLOCATE MEMORY AND CONDITIONALLY WAIT
                  00000000  157 .PSECT  CODE     SHR,    EXE,    NOWRT,  BYTE
                     0000   158
                     0000   159 ;+
                     0000   160 ; EXE$ALLOCBUF - ALLOCATE BUFFERED I/O BUFFER AND CONDITIONALLY WAIT
                     0000   161 ;
                     0000   162 ; THIS ROUTINE IS CALLED TO ALLOCATE A BUFFERED I/O BUFFER. IF SUFFICIENT
                     0000   163 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
                     0000   164 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   165 ;
                     0000   166 ; EXE$ALLOCCEB - ALLOCATE COMMON EVENT BLOCK AND CONDITIONALLY WAIT
                     0000   167 ;
                     0000   168 ; THIS ROUTINE IS CALLED TO ALLOCATE A COMMON EVENT BLOCK. IF SUFFICIENT
                     0000   169 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
                     0000   170 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   171 ;
                     0000   172 ; EXE$ALLOCJIB - ALLOCATE JOB INFORMATION BLOCK AND CONDITIONALLY WAIT
                     0000   173 ;
                     0000   174 ; THIS ROUTINE IS CALLED TO ALLOCATE A JOB INFORMATION BLOCK. IF SUFFICIENT
                     0000   175 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
                     0000   176 ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   177 ;
                     0000   178 ; EXE$ALLOCIRP - ALLOCATE I/O REQUEST PACKET AND CONDITIONALLY WAIT
                     0000   179 ;
                     0000   180 ; THIS ROUTINE IS CALLED TO ALLOCATE AN I/O PACKET. IF SUFFICIENT MEMORY
                     0000   181 ; IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
                     0000   182 ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   183 ;
                     0000   184 ; EXE$ALLOCPCB - ALLOCATE PROCESS CONTROL BLOCK AND CONDITIONALLY WAIT
                     0000   185 ;
                     0000   186 ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS CONTROL BLOCK WHEN
                     0000   187 ; CREATING A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN
                     0000   188 ; A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED DEPENDING ON THE CURRENT
                     0000   189 ; PROCESS' RESOURCE WAIT MODE.
                     0000   190 ;
                     0000   191 ; EXE$ALLOCPQB - ALLOCATE PROCESS QUOTA BLOCK AND CONDITIONALLY WAIT
                     0000   192 ;
                     0000   193 ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS QUOTA BLOCK WHEN CREATING
                     0000   194 ; A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN A RESOURCE
                     0000   195 ; WAIT STATE IS ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT
                     0000   196 ; MODE.
                     0000   197 ;
                     0000   198 ; EXE$ALLOCTQE - ALLOCATE TIME QUEUE ENTRY AND CONDITIONALLY WAIT
                     0000   199 ;
                     0000   200 ; THIS ROUTINE IS CALLED TO ALLOCATE A TIME QUEUE ENTRY. IF SUFFICIENT
                     0000   201 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
                     0000   202 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
                     0000   203 ;
                     0000   204 ; INPUTS:
                     0000   205 ;
                     0000   206 ;       R4 = NORMALLY CURRENT PROCESS PCB ADDRESS, BUT NOT REQUIRED.
                     0000   207 ;
                     0000   208 ; IF ENTRY AT EXE$ALLOCBUF, THEN
                     0000   209 ;
                     0000   210 ;       R1 = SIZE OF REQUESTED BUFFER IN BYTES.
                     0000   211 ;
                     0000   212 ; OUTPUTS:
```

```
                                0000   213 ;
                                0000   214 ;       R0 = LOW BIT CLEAR IF ALLOCATION FAILURE WITH CALLING IPL PRESERVED.
                                0000   215 ;
                                0000   216 ;               R0 = SS$_INSFMEM = INSUFFICIENT MEMORY AVAILABLE TO ALLOCATE
                                0000   217 ;                    BUFFER.
                                0000   218 ;
                                0000   219 ;       R0 = LOW BIT SET IF SUCCESSFUL ALLOCATION WITH:
                                0000   220 ;
                                0000   221 ;               R1 = SIZE OF REQUESTED BUFFER IN BYTES.
                                0000   222 ;               R2 = ADDRESS OF ALLOCATED BUFFER WITH SIZE AND TYPE FIELDS
                                0000   223 ;                    FILLED IN.
                                0000   224 ;
                                0000   225 ;               AND IPL SET TO AST DELIVERY LEVEL.
                                0000   226 ;
                                0000   227 ;       R4 = ORIGINAL R4 OR CURRENT PCB IF A WAIT OCCURRED.
                                0000   228 ;-
                                0000   229
                                0000   230                .ENABL  LSB
                                0000   231 EXE$ALLOCBUF::                                ;ALLOCATE BUFFERED I/O BUFFER
                     13   DD    0000   232                PUSHL   #DYN$C_BUFIO            ;SET DATA STRUCTURE TYPE
                     28   11    0002   233                BRB     20$                     ;
                                0004   234 EXE$ALLOCCEB::                                ;ALLOCATE COMMON EVENT BLOCK
                     04   DD    0004   235                PUSHL   #DYN$C_CEB              ;SET DATA STRUCTURE TYPE
                                0006   236                ASSUME  CEB$C_LENGTH LE IRP$C_LENGTH ;IRP SIZE PACKETS ARE ALLOCATED
                                0006   237                ASSUME  CEB$C_SLAVLNG LE IRP$C_LENGTH ;FOR CEB'S TO LIMIT FRAGMENTATION
                     20   11    0006   238                BRB     10$                     ;
                                0008   239 EXE$ALLOCJIB::                                ;ALLOCATE JOB INFORMATION BLOCK - COND WAIT
                     2F   DD    0008   240                PUSHL   #DYN$C_JIB              ;SET STRUCTURE TYPE
           51  0074  8F   3C    000A   241                MOVZWL  #JIB$C_LENGTH,R1        ;AND LENGTH OF BLOCK
                     1B   11    000F   242                BRB     20$                     ;MERGE WITH COMMON ALLOCATE CODE
                                0011   243 EXE$ALLOCIRP::                                ;ALLOCATE I/O PACKET - CONDITIONAL WAIT
                     0A   DD    0011   244                PUSHL   #DYN$C_IRP              ;SET DATA STRUCTURE TYPE
                     13   11    0013   245                BRB     10$                     ;
                                0015   246 EXE$ALLOCPCB::                                ;ALLOCATE PROCESS CONTROL BLOCK
                     0C   DD    0015   247                PUSHL   #DYN$C_PCB              ;SET DATA STRUCTURE TYPE
             51   8C  8F   9A    0017   248                MOVZBL  #PCB$C_LENGTH,R1        ;AND STRUCTURE SIZE
                     0F   11    001B   249                BRB     20$                     ;
                                001D   250 EXE$ALLOCPQB::                                ;ALLOCATE PROCESS QUOTA BLOCK
                     0D   DD    001D   251                PUSHL   #DYN$C_PQB              ;SET DATA STRUCTURE TYPE
           51  08C8  8F   3C    001F   252                MOVZWL  #PQB$C_LENGTH,R1        ;AND STRUCTURE SIZE
                     06   11    0024   253                BRB     20$                     ;
                                0026   254 EXE$ALLOCTQE::                                ;ALLOCATE TIME QUEUE ENTRY
                     0F   DD    0026   255                PUSHL   #DYN$C_TQE              ;SET DATA STRUCTURE TYPE
             51   60  8F   9A    0028   256 10$:          MOVZBL  #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SET SIZE OF BUFFER REQUIRED
                     7E   DC    002C   257 20$:          MOVPSL  -(SP)                   ;READ CURRENT PSL
                     51   DD    002E   258                PUSHL   R1                      ;SAVE REQUEST SIZE
                     2E   10    0030   259                BSBB    EXE$ALONONPAGED         ;ATTEMPT TO ALLOCATE PACKET
                     0A   BA    0032   260                POPR    #^M<R1,R3>              ;RETRIEVE REQUEST SIZE AND PREVIOUS IPL
                  0C 50   E9    0034   261                BLBC    R0,40$                  ;IF LBC NO PACKET ALLOCATED
               08 A2   51   B0    0037   262                MOVW    R1,IRP$W_SIZE(R2)       ;INSERT SIZE OF ALLOCATED BLOCK
            0A A2   04   AE   9B    003B   263                MOVZBW  4(SP),IRP$B_TYPE(R2)    ;INSERT DATA STRUCTURE TYPE
                              0040   264                                                ;AND CLEAR MISCELLANEOUS BYTE
                     8E   D5    0040   265                TSTL    (SP)+                   ;Remove Type from stack
                     05   0042   266                RSB                             ;
             50   0124 8F   3C    0043   267 40$:          MOVZWL  #SS$_INSFMEM,R0         ;SET INSUFFICIENT MEMORY
                              0048   268                ERRSUP_S        MSGADR=T_CGF    ;CANNOT GET FREE CORE
              00000000'EF   DF    0048                   PUSHAL  $MODULE
```

```
                00000007'EF   DF  004E                        PUSHAL  T_CGF
                          01  DD  0054                        PUSHL   #$ER
              00000000'EF     03  FB  0056                    CALLS   #3, DS_ERRSUP
                              00  005D  269        HALT                        ;HALT FOR NOW
                          E3  11  005E  270        BRB     40$                 ;Prevent CONTINUE
                              0060  271        .DSABL  LSB
```

```
                                0060   273          .SBTTL  ALLOCATE NONPAGED DYNAMIC MEMORY
                                0060   274 ;+
                                0060   275 ; EXE$ALONONPAGED - ALLOCATE NONPAGED DYNAMIC MEMORY
                                0060   276 ;
                                0060   277 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE NONPAGED POOL.
                                0060   278 ; IF THE BLOCK IS THE SAME SIZE AS AN I/O PACKET, AN ATTEMPT IS MADE TO ALLO-
                                0060   279 ; CATE IT FROM THE LOOKASIDE LIST.
                                0060   280 ;
                                0060   281 ; INPUTS:
                                0060   282 ;
                                0060   283 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                0060   284 ;
                                0060   285 ; OUTPUTS:
                                0060   286 ;
                                0060   287 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                0060   288 ;
                                0060   289 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                0060   290 ;
                                0060   291 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                0060   292 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                0060   293 ;-
                                0060   294
                                0060   295          .ENABL  LSB
                                0060   296 EXE$ALONONPAGED::                          ;ALLOCATE NONPAGED MEMORY
                                0060   297          $MP_SET_INTERLOCK #MP$V_IL_ALLN ; INTERLOCK ROUTINE                [08]
                     04   DD    0060                PUSHL   #MP$V_IL_ALLN
                     08   DD    0062                PUSHL   #$SRVCODE_SET_INTERLOCK
          00000000'9F     02   FB  0064             CALLS   #2, a#DS$SERVICE_DISPATCHER
                                0068   298
                51   0F   C0    0068   299          ADDL    #MASK,R1                   ;ROUND SIZE UP TO NEXT BOUNDRY
                51   0F   CA    006B   300          BICL    #MASK,R1                   ;TRUNCATE SIZE BACK TO MULTIPLE
                     4B   13    006E   301          BEQL    60$                        ;IF EQL BAD ALLOCATION REQUEST
0000FE00'EF          1C   E0    0070   302          BBS     #DSA$V_USER, -
                     06         0077   303                  DSA$GL_FLAGS,15$           ;SKIP DSBINT IF IN USER MODE
                                0078   304          DSBINT  #^X1F
                7E   12   DB    0078                MFPR    S^#PR$_IPL,-(SP)
                12   1F   DA    007B                MTPR    #^X1F,S^#PR$_IPL
                                007E   305 15$:
           51   0060 8F   B1    007E   306          CMPW    #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SIZE EQUAL TO I/O PACKET?
                     1E   1F    0083   307          BLSSU   10$                        ;IF NEQ NO
                                0085   308 ;+
                                0085   309 ;       Replaced        REMQUE  @L^IOC$GL_IRPFL,R2
                                0085   310 ;-
          52   00000014'FF     DE    0085   311          MOVAL   @L^IOC$GL_IRPFL,R2         ;FIRST ENTRY IN QUEUE
                04   A2   52   D1    008C   312          CMPL    R2,4(R2)                   ;QUEUE EMPTY?
                     11   13    0090   313          BEQL    10$                        ;YES,
                04   B2   62   D0    0092   314          MOVL    (R2),@4(R2)                ;COPY PREDECESSOR LINK
                     50   62   D0    0096   315          MOVL    (R2),R0                    ;COPY LINK TO SUCCESSOR
             04   A0   04   A2   D0    0099   316          MOVL    4(R2),4(R0)                ;COPY PREDECESOR LINK
                     50   01   D0    009E   317          MOVL    #SS$_NORMAL,R0             ;SET SUCCESSFUL COMPLETION
                     0B   11    00A1   318          BRB     40$                        ;ENABLE AND EXIT
          53   00000000'EF     9E    00A3   319 10$:     MOVAB   L^EXE$GL_NONPAGED,R3       ;GET ADDRESS OF NONPAGED MEMORY LISTHEAD
                     83   D5    00AA   320          TSTL    (R3)+                      ;Skip IPL for list
                     2D   10    00AC   321          BSBB    EXE$ALLOCATE               ;ALLOCATE BLOCK
                                00AE   322 40$:
0000FE00'EF          1C   E0    00AE   323          BBS     #DSA$V_USER, -
                     03         00B5   324                  DSA$GL_FLAGS,50$           ;SKIP ENBINT IF IN USER MODE
```

```
                                  00B9   325          ENBINT
                    12   8E   DA   00B9                        MTPR    (SP)+,S^#PR$_IPL
                                  00BC   326 50$:
                         14   11   00BC   327          BRB     70$                     ; JUMP TO CLEAR INTERLOCK AND RSB  [08]
                                  00BE   328
                                  00BE   329 60$:     BUG_CHECK BADALORQSZ            ;BAD ALLOCATION REQUEST SIZE
                 00000000'9F   16   00BE               JSB     a#BUG$CHECK
2C 5A 53 51 52 4F 4C 41 44 41 42 00'  00C4               .ASCIC  "BADALORQSZ,"
                              0B   00C4
                         50   D4   00D0   330          CLRL    R0                      ;INDICATE NO BLOCK ALLOCATED
                                  00D2   331
                                  00D2   332 70$:
                                  00D2   333          $MP_CLEAR_INTERLOCK #MP$V_IL_ALLN ; CLEAR INTERLOCK              [08]
                         04   DD   00D2                        PUSHL   #MP$V_IL_ALLN
                         09   DD   00D4                        PUSHL   #$SRVCODE_CLEAR_INTERLOCK
            00000000'9F   02   FB   00D6                        CALLS   #2, a#DS$SERVICE_DISPATCHER
                              05   00DD   334          RSB
                                  00DE   335
                                  00DE   336          .DSABL  LSB
```

```
                        00DE    338          .SBTTL  GENERAL ALLOCATE MEMORY SUBROUTINE
                        00DE    339 ;+
                        00DE    340 ; EXE$ALLOCATE - ALLOCATE MEMORY SUBROUTINE
                        00DE    341 ;
                        00DE    342 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM A POOL WHOSE ENTRIES
                        00DE    343 ; ARE MAINTAINED IN A MEMORY ORDER SORTED LIST.
                        00DE    344 ;
                        00DE    345 ; INPUTS:
                        00DE    346 ;
                        00DE    347 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                        00DE    348 ;       R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                        00DE    349 ;
                        00DE    350 ; OUTPUTS:
                        00DE    351 ;
                        00DE    352 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                        00DE    353 ;
                        00DE    354 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                        00DE    355 ;
                        00DE    356 ;               R1 = SIZE OF ALLOCATED BLOCK.
                        00DE    357 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                        00DE    358 ;-
                        00DE    359
                        00DE    360 EXE$ALLOCATE::                              ;ALLOCATE MEMORY
          50    53    D0  00DE    361          MOVL    R3,R0                      ;COPY ADDRESS OF FIRST FREE BLOCK ADDRESS
          52    50    D0  00E1    362 10$:     MOVL    R0,R2                      ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
          50    62    D0  00E4    363          MOVL    (R2),R0                    ;GET ADDRESS OF NEXT FREE BLOCK
                1C    13  00E7    364          BEQL    30$                        ;IF EQL NO MEMORY AVAILABLE
       04 A0    51    D1  00E9    365          CMPL    R1,4(R0)                   ;FREE BLOCK BIG ENOUGH?
                F2    1A  00ED    366          BGTRU   10$                        ;IF GTRU NO
                0E    13  00EF    367          BEQL    20$                        ;IF EQL FREE BLOCK IS EXACT SIZE
    53    51    50    C1  00F1    368          ADDL3   R0,R1,R3                   ;CALCULATE ADDRESS OF NEW FREE BLOCK
          83    80    D0  00F5    369          MOVL    (R0)+,(R3)+                ;COPY LINK TO NEXT FREE BLOCK
    63    60    51    C3  00F8    370          SUBL3   R1,(R0),(R3)               ;CALCULATE SIZE OF NEW FREE BLOCK
          70    73    DE  00FC    371          MOVAL   -(R3),-(R0)                ;SET LINK TO NEW FREE BLOCK
          62    60    D0  00FF    372 20$:     MOVL    (R0),(R2)                  ;COPY LINK TO NEW FREE BLOCK
          52    80    9E  0102    373          MOVAB   (R0)+,R2                   ;SET ADR OF ALLOCATED BLOCK, INDICATE SUCCES
                      05  0105    374 30$:     RSB                                ;
```

```
                                0106   376          .SBTTL  DEALLOCATE NONPAGED DYNAMIC MEMORY
                                0106   377 ;+
                                0106   378 ; EXE$DEANONPAGED - DEALLOCATE NONPAGED DYNAMIC MEMORY
                                0106   379 ;
                                0106   380 ; THIS ROUTINE IS CALLED TO DEALLOCATE A BLOCK OF MEMORY TO A NONPAGED POOL.
                                0106   381 ; IF THE BLOCK IS A SHARED MEMORY BLOCK TYPE, THE BLOCK IS DEALLOCATED TO
                                0106   382 ; THE SHARED MEMORY POOL.  OTHERWISE, THE BLOCK'S ADDRESS IS CHECKED TO SEE
                                0106   383 ; IF IT WAS ALLOCATED FROM THE I/O PACKET LOOKASIDE LIST AND IF SO,
                                0106   384 ; IT IS RETURNED TO THAT LIST.  OTHERWISE IT IS MERGED INTO THE NORMAL
                                0106   385 ; NONPAGED POOL.
                                0106   386 ;
                                0106   387 ; INPUTS:
                                0106   388 ;
                                0106   389 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                                0106   390 ;       IRP$W_SIZE(R0) = SIZE OF BLOCK TO BE DEALLOCATED.
                                0106   391 ;       IRP$B_TYPE(R0) = TYPE OF BLOCK TO BE DEALLOCATED.
                                0106   392 ;
                                0106   393 ; OUTPUTS:
                                0106   394 ;
                                0106   395 ;       THE SPECIFIED BLOCK IS RETURNED TO THE APPROPRIATE POOL.
                                0106   396 ;-
                                0106   397
                                0106   398          .ENABL  LSB
                                0106   399 EXE$DEANONPAGED::                          ;DEALLOCATE NONPAGED DYNAMIC MEMORY
                                0106   400          $MP_SET_INTERLOCK #MP$V_IL_DELN ; Set interlock bit                [08]
                     05   DD    0106                    PUSHL   #MP$V_IL_DELN
                     08   DD    0108                    PUSHL   #$SRVCODE_SET_INTERLOCK
          00000000'9F    02   FB 010A                    CALLS   #2, a#DS$SERVICE_DISPATCHER
                                0111   401
                     56   10    0111   402          BSBB    CHECKBLOCK                 ;CHECK DEALLOCATION PARAMETERS
          53  00000000'EF   9E    0113   403          MOVAB   L^EXE$GL_NONPAGED,R3       ;GET ADDRESS OF NONPAGED MEMORY LISTHEAD
                     83   D5    011A   404          TSTL    (R3)+                      ;Skip IPL for list
                     1C   E0    011C   405          BBS     #DSA$V_USER -
            06 0000FE00'EF          011E   406                  ,DSA$GL_FLAGS, 7$          ;DISABLE ONLY IF IN S/A
                                0124   407          DSBINT  #^X1F
                  7E   12   DB  0124                       MFPR    S^#PR$_IPL,-(SP)
                  12   1F   DA  0127                       MTPR    #^X1F,S^#PR$_IPL
                                012A   408 7$:
          0000000C'EF   50   D1  012A   409          CMPL    R0,L^EXE$GL_SPLITADR       ;I/O PACKET?
                     1A   1F    0131   410          BLSSU   10$                        ;IF LSSU NO
                                0133   411 ;+
                                0133   412 ;       Replaced        INSQUE  (R0),@L^IOC$GL_IRPBL
                                0133   413 ;-
          53  00000018'FF   DE  0133   414          MOVAL   @L^IOC$GL_IRPBL,R3         ;WHERE TO INSERT IT
                  60   63   D0  013A   415          MOVL    (R3),(R0)
                04 A0   63   DE  013D   416          MOVAL   (R3),4(R0)
                  63   60   DE  0141   417          MOVAL   (R0),(R3)
          00000018'EF   60   DE  0144   418          MOVAL   (R0),L^IOC$GL_IRPBL
                     02   11    014B   419          BRB     20$                        ;
                     40   10    014D   420 10$:     BSBB    EXE$DEALLOCATE             ;DEALLOCATE BLOCK
                  50   03   3C  014F   421 20$:     MOVZWL  #RSN$_NPDYNMEM,R0          ;SET NONPAGED DYNAMIC MEMORY RESOURCE NUMBER
                     1C   E0    0152   422          BBS     #DSA$V_USER -
            03 0000FE00'EF          0154   423                  ,DSA$GL_FLAGS, 30$         ;ENBINT IF IN S/A
                                015A   424          ENBINT                             ;ENABLE INTERRUPTS
                  12   8E   DA  015A                       MTPR    (SP)+,S^#PR$_IPL
                                015D   425 30$:
                                015D   426          $MP_CLEAR_INTERLOCK #MP$V_IL_DELN ; CLEAR interlock bit            [08]
```

```
                    05   DD  015D                        PUSHL   #MP$V_IL_DELN
                    09   DD  015F                        PUSHL   #$SRVCODE_CLEAR_INTERLOCK
          00000000'9F   02   FB  0161                    CALLS   #2, a#DS$SERVICE_DISPATCHER
                              0168   427
                         05   0168   428 31$:    RSB                         ;RETURN
                              0169   429          .DSABL  LSB
```

```
                                0169   431          .SBTTL  CHECK BLOCK PARAMETERS SUBROUTINE
                                0169   432 ;
                                0169   433 ; CHECKBLOCK - CHECK BLOCK PARAMETERS SUBROUTINE
                                0169   434 ;
                                0169   435
                                0169   436 CHECKBLOCK:                                    ;CHECK BLOCK PARAMETERS
                     50   0F   D3  0169   437          BITL    #MASK,R0                  ;BLOCK ALIGNED ON BOUNDRY?
                          0C   12  016C   438          BNEQ    10$                       ;IF NEQ NO - BAD DEALLOCATION
                   51   08 A0   3C  016E   439          MOVZWL  IRP$W_SIZE(R0),R1         ;GET SIZE OF BLOCK IN BYTES
                      51   0F   C0  0172   440          ADDL    #MASK,R1                  ;ROUND SIZE UP TO NEXT BOUNDRY
                      51   0F   CA  0175   441          BICL    #MASK,R1                  ;TRUNCATE SIZE BACK TO MULTIPLE
                           14   12  0178   442          BNEQ    20$                       ;IF NEQ OKAY
                                    017A   443 10$:      BUG_CHECK BADDALRQSZ              ;BAD DEALLOCATION REQUEST SIZE OR ADDRESS
                00000000'9F    16  017A                 JSB     @#BUG$CHECK
2C 5A 53 51 52 4C 41 44 44 41 42 00' 0180               .ASCIC  "BADDALRQSZ,"
                                0B  0180
                           8E   D5  018C   444          TSTL    (SP)+                     ;REMOVE RETURN FROM STACK
                                05  018E   445 20$:      RSB                               ;
```

```
                                018F    447          .SBTTL  GENERAL DEALLOCATION SUBROUTINE
                                018F    448 ;+
                                018F    449 ; EXE$DEALLOCATE - DEALLOCATION SUBROUTINE
                                018F    450 ;
                                018F    451 ; INPUTS:
                                018F    452 ;
                                018F    453 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                                018F    454 ;       R1 = SIZE OF BLOCK IN BYTES
                                018F    455 ;       R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                                018F    456 ;
                                018F    457 ; OUTPUTS:
                                018F    458 ;
                                018F    459 ;       NONE
                                018F    460 ;-
                                018F    461
                                018F    462 EXE$DEALLOCATE::                            ;DEALLOCATE BLOCK
          52    53    D0      018F    463 10$:    MOVL    R3,R2                       ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
          53    62    D0      0192    464         MOVL    (R2),R3                     ;GET ADDRESS OF NEXT FREE BLOCK
                07    13      0195    465         BEQL    20$                         ;IF EQL END OF LIST
          53    50    D1      0197    466         CMPL    R0,R3                       ;BLOCK LOGICALLY GO HERE?
                F3    1A      019A    467         BGTRU   10$                         ;IF GTRU NO
                2D    13      019C    468         BEQLU   50$                         ;IF EQLU DOUBLE DEALLOCATION
          60    53    D0      019E    469 20$:    MOVL    R3,(R0)                     ;ASSUME NO AGGLOMERATION
       7E 51    50    C1      01A1    470         ADDL3   R0,R1,-(SP)                 ;CALCULATE ADDRESS OF END OF BLOCK
          8E    53    D1      01A5    471         CMPL    R3,(SP)+                    ;END OF BLOCK EQUAL TO NEXT IN LIST?
                06    12      01A8    472         BNEQ    30$                         ;IF NEQ DO NOT AGGLOMERATE
          60    83    D0      01AA    473         MOVL    (R3)+,(R0)                  ;MOVE LINK TO BLOCK BEING RELEASED
          51    63    C0      01AD    474         ADDL    (R3),R1                     ;ACCUMULATE LENGTH OF NEW FREE BLOCK
                52    DD      01B0    475 30$:    PUSHL   R2                          ;CALCULATE ENDING ADDRESS OF PREVIOUS BLOCK
          82    50    D0      01B2    476         MOVL    R0,(R2)+                    ;ASSUME NO AGGLOMERATION
          6E    62    C0      01B5    477         ADDL    (R2),(SP)                   ;ADD LENGTH TO BLOCK BASE ADDRESS
          8E    50    D1      01B8    478         CMPL    R0,(SP)+                    ;END ADDRESS EQUAL TO BLOCK BEING RELEASED?
                09    12      01BB    479         BNEQ    40$                         ;IF NEQ DO NOT AGGLOMERATE BLOCKS
          51    62    C0      01BD    480         ADDL    (R2),R1                     ;ACCUMULATE SIZE OF NEW FREE BLOCK
          72    60    D0      01C0    481         MOVL    (R0),-(R2)                  ;MOVE LINK TO PREVIOUS FREE BLOCK
          50    52    D0      01C3    482         MOVL    R2,R0                       ;SET ADDRESS OF NEW FREE BLOCK
    04 A0 51    D0      01C6    483 40$:    MOVL    R1,4(R0)                    ;SET SIZE OF FREE BLOCK
                      05    01CA    484         RSB                                 ;
                            01CB    485 50$:    BUG_CHECK DOUBLDEALO,FATAL          ;DOUBLE DEALLOCATION OF MEMORY BLOCK
          00000000'9F   16  01CB                JSB     a#BUG$CHECK
2C 4F 4C 41 45 44 4C 42 55 4F 44 00' 01D1                .ASCIC  "DOUBLDEALO,FATAL"
                4C 41 54 41 46  01DD
                              10  01D1
                              00  01E2    486         HALT                                ;HALT FOR NOW
                      E6      11  01E3    487         BRB     50$
```

```
                        01E5   489          .SBTTL  MEMPOOL$INIT                    ;USER MODE MEMORY INITIALIZATION
                     000001E5   490 .PSECT  CODE,   SHR,    EXE,    NOWRT,  BYTE
                        01E5   491
                        01E5   492 ;++
                        01E5   493 ;FUNCTIONAL DESCRIPTION:
                        01E5   494 ;
                        01E5   495 ;       This routine allocates how must scratch memory is required and
                        01E5   496 ;       expands the program region to get it. This memory is optionally
                        01E5   497 ;       split between IRP's and other free memory
                        01E5   498 ;
                        01E5   499 ;CALLING SEQUENCING:
                        01E5   500 ;
                        01E5   501 ;       BSBW    USER$MEMINIT
                        01E5   502 ;
                        01E5   503 ;INPUT PARAMETERS:
                        01E5   504 ;
                        01E5   505 ;IMPLICIT INPUTS:
                        01E5   506 ;
                        01E5   507 ;OUTPUT PARAMETERS:     NONE
                        01E5   508 ;
                        01E5   509 ;IMPLICIT OUTPUTS:      NONE
                        01E5   510 ;
                        01E5   511 ;COMPLETION CODES:      NONE
                        01E5   512 ;
                        01E5   513 ;SIDE EFFECTS:          NONE
                        01E5   514 ;--
                        01E5   515
                        01E5   516 MEMPOOL$INIT::
                  0C    BB    01E5   517          PUSHR   #^M<R2,R3>              ; Save working registers
          50    00000000'8F   D0    01E7   518          MOVL    #SCRIPT$MINCORE,R0      ;SPACE NEEDED FOR SCRIPTS
          50    00000000'8F   C0    01EE   519          ADDL2   #QIO$MINCORE,R0         ;PLUS QIO SPACE
              51    50    0A    C7    01F5   520          DIVL3   #10,R0,R1               ;PLUS 10%
                    50    51    C0    01F9   521          ADDL2   R1,R0
              51    50    05    C5    01FC   522          MULL3   #5,R0,R1                ;IN USER MODE USE 5 TIMES MINIMUM
          50    00000000'8F   C0    0200   523          ADDL2   #ULTRIX$IOB+ULTRIX$B,R0 ; Added space for ULTRIX buf [09]     ;G:2
          0000FE00'EF   1C    E0    0207   524          BBS     #DSA$V_USER,DSA$GL_FLAGS -
                        0F          020E   525                  . 10$                  ;SKIP CALCULATION IN USER MODE
                                    020F   526
00000000'EF     00000000'EF   C3    020F   527          SUBL3   DS$GA_LASTADR,DS$GL_MEMSIZE -
                        51          021A   528                  . R1                   ;ROOM ABOVE SUPERVISOR
                    51    0A    C6    021B   529          DIVL2   #10,R1                  ;10% OF THAT
                                    021E   530 10$:
              0003 51   00    50    F1    021E   531          ACBL    R0,#0,R1,20$           ;MAX (MIN NECESSARY, AVAILABLE)
                    50    51    D0    0224   532          MOVL    R1,R0
                50    01FF  8F    A8    0227   533 20$:     BISW2   #^X1FF,R0              ;PAGE ALIGN
                          50    D6    022C   534          INCL    R0
          0000FE00'EF   1C    E0    022E   535          BBS     #DSA$V_USER,DSA$GL_FLAGS -
                        0D          0235   536                  . 30$                  ;USER ALLOCATIONS ARE DIFFERENT
                                    0236   537
          51    00000000'EF   D0    0236   538          MOVL    DS$GA_LASTADR,R1        ;SAVE START ADDRESS OF SPACE
                  50    6041  9E    023D   539          MOVAB   (R0)[R1],R0             ;HIGHEST+1 ADDRESS OF SPACE
                        23    11    0241   540          BRB     40$                     ;JOIN COMMON CODE
                                    0243   541 30$:
                    51    7E    7E    0243   542          MOVAQ   -(SP),R1                ;WHERE VMS WILL RETURN THE ADDRESSES
           50   50      F7  8F    78    0246   543          ASHL    #-9,R0,R0               ;MAKE PAGE COUNT
                                    024B   544          $EXPREG_S R0,(R1)               ;ALLOCATE THE SPACE
                        00    DD    024B                 PUSHL   #0
```

```
                        00   DD  024D                        PUSHL   #0
                        61   7F  024F                        PUSHAQ  (R1)
                        50   DD  0251                        PUSHL   R0
          00000000'GF   04   FB  0253                        CALLS   #4,G^SYS$EXPREG
                        02   BA  025A   545          POPR    #^M<R1>                 ;GET START ADDRESS
     50   8E   000001FF 8F   C9  025C   546          BISL3   #^X1FF,(SP)+,R0         ;GET END ADDRESS
                        50   D6  0264   547          INCL    R0                      ;+1
                             0266   548 40$:
                             0266   549 ;+
                             0266   550 ;       R0      Highest address allocated + 1
                             0266   551 ;       R1      Lowest address allocated
                             0266   552 ;-
          00000000'EF   51   D0  0266   553          Movl    R1, L^ULTRIX_IOB_ADD    ; Address of IOB buffer                 [09]
00000000'EF   0000'C1   9E  026D   554          Movab   ULTRIX$IOB(R1),L^ULTRIX_B_ADD ; ADDRESS OF B BUFFER              [09]
          51   0000'C1   9E  0276   555          MOVAB   ULTRIX$IOB+ULTRIX$B(R1),R1 ; RESERVED FOR ULTRIX                [09]
          51   01FF C1   DE  027B   556          MOVAL   511(R1),R1              ; Insure page alignment                 [09]
     51   000001FF 8F   CA  0280   557          BICL2   #^X1FF,R1               ; for look-aside buffer                 [09]
                             0287   558                                                                                 ;G:2
          00000010'EF   51   D0  0287   559          MovL    R1, L^Ds$GL_LookAside   ; Store start of look-aside            [06]
          00000000'EF   50   D0  028E   560          MovL    R0, L^DS$GA_LASTADR     ;PUT BACK NEW AVAILABLE ADDR
          00000004'EF   51   D0  0295   561          MOVL    R1,EXE$GL_NONPAGED+4    ;SET ADDRESS OF CHUNK
                        61   D4  029C   562          CLRL    (R1)                    ;THIS IS FIRST AND LAST CHUNK
                             029E   563
     52   00000014'EF   9E  029E   564          MOVAB   L^IOC$GL_IRPFL,R2       ;ADDR OF LIST HEAD
               62   52   D0  02A5   565          MOVL    R2,(R2)                 ;INIT LISTHEAD
            04 A2   52   D0  02A8   566          MOVL    R2,4(R2)                ;AND TAIL
53   50   00000000'8F   C3  02AC   567          SUBL3   #<SGN$C_IRPCNT*<IRP$C_LENGTH+15>&-16>,R0 -
                             02B4   568                  ,R3                     ;ADDR OF FIRST IRP, LOW LIMIT
                        11   11  02B4   569          BRB     60$                     ;COUNT FIRST
                             02B6   570 50$:
                             02B6   571 ;+
                             02B6   572 ;       Replaced        INSQUE (R0),(R2)
                             02B6   573 ;-
               53   62   D0  02B6   574          MOVL    (R2),R3                 ;Address of next
               60   53   D0  02B9   575          MOVL    R3,(R0)                 ;Make flink of new point to next
            04 A0   52   D0  02BC   576          MOVL    R2,4(R0)                ;Make blink of new point to current
               62   50   D0  02C0   577          MOVL    R0,(R2)                 ;Make flink of curr point to new
            04 A3   50   D0  02C3   578          MOVL    R0,4(R3)                ;Make blink of old point to new
                             02C7   579 60$:
FFE5 50     FFFFFFA0 8F   53  F1  02C7   580          ACBL    R3,#-<<IRP$C_LENGTH+15>&-16>,R0 -
                             02D1   581                  ,50$                    ;DECREMENT ADDR BY LENGTH AND MAKE ONE
                             02D1   582
          04 A1   50   51   C3  02D1   583          SUBL3   R1,R0,4(R1)             ;SET LENGTH OF REMAINING CHUNK
          0000000C'EF   53   D0  02D6   584          MOVL    R3,EXE$GL_SPLITADR      ;SET ADDRESS OF LOWEST IRP
                             02DD   585
                        0C   BA  02DD   586          POPR    #^M<R2,R3>              ; Restore registers
                             05  02DF   587          RSB                             ;RETURN
                             02E0   588
```

```
                                02E0          .SBTTL  RETURN SIZE OF PHYSICAL MEMORY                                  ;[07
                                02E0    ;++                                                                          [07]
                                02E0    ;FUNCTIONAL DESCRIPTION                                                      [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       This routine can be called to return the size of physical memory.    [07]
                                02E0    ;       The size returned is given in number of pages. A check is made to    [07]
                                02E0    ;       to see if VDS is ryunning standalone. If not, an error return code   [07]
                                02E0    ;       is given.                                                            [07]
                                02E0    ;                                                                            [07]
                                02E0    ;CALLING SEQUENCE                                                            [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       CALLS   #1,DS$MEMSIZE                                                [07]
                                02E0    ;                                                                            [07]
                                02E0    ;INPUT PARAMETERS:                                                           [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       4(AP) - Address of longword to receive page count                    [07]
                                02E0    ;                                                                            [07]
                                02E0    ;IMPLICIT INPUTS:                                                            [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       NONE                                                                 [07]
                                02E0    ;                                                                            [07]
                                02E0    ;OUTPUT PARAMETERS:                                                          [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       @4(AP) - Page count of physical memory                               [07]
                                02E0    ;                                                                            [07]
                                02E0    ;IMPLICIT OUPUTS:                                                            [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       NONE                                                                 [07]
                                02E0    ;                                                                            [07]
                                02E0    ;COMPLETION CODES:                                                           [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       SS$_NORMAL - Successful completion                                   [07]
                                02E0    ;       DS$_NOSUPPORT - VDS is online, no value returned                     [07]
                                02E0    ;                                                                            [07]
                                02E0    ;SIDE EFFECTS:                                                               [07]
                                02E0    ;                                                                            [07]
                                02E0    ;       NONE                                                                 [07]
                                02E0    ;--                                                                          [07]
                          0000  02E0            .ENTRY  DSX$MEMSIZE,^M<>        ;ENTRY POINT - RETURN MEMORY SIZE    [07]
                                02E2
        50   006600B1 8F   D0   02E2            MOVL    #DS$_NOSUPPORT,R0       ;Prepare to return error code        [07]
                                02E9            BR_IF_USER DS$MEMSIZE_X         ;Is VDS in user-mode?                [07]
        0000FE00'EF   1C   E0   02E9            BBS     #DSA$V_User, -                  ; If bit set, branch         [28]
                      10        02F0                      L^DSA$GL_Flags, DS$MEMSIZE_X          ;
  00000000'EF   00000200 8F   C7   02F1            DIVL3   #512,DS$GL_MEMSIZE,@4(AP)   ;Get the memory size in pages    [07]
                   04 BC        02FC
                50    01   D0   02FE            MOVL    #SS$_NORMAL,R0          ;Set normal return code              [07]
                                0301    DS$MEMSIZE_X:                                                                ;[07
                           04   0301            RET                             ;return to caller                    [07]
                                0302
                                0302
                                0302            .END
```

```
$$T1                            = 00000000     D
$ER                             = 00000002     D
$MODULE                           00000000 R   D   03
$SRVCODE_BOOTATTACHED           = 00000000     D
$SRVCODE_CLEAR_INTERLOCK        = 00000009     D
$SRVCODE_GETSYSBUF              = 00000004     D
$SRVCODE_HALTATTACHED           = 00000002     D
$SRVCODE_INTERLOCK_TEST         = 00000007     D
$SRVCODE_PPSCB                  = 0000000A     D
$SRVCODE_PRINTREV               = 00000006     D
$SRVCODE_RELSYSBUF              = 00000005     D
$SRVCODE_SET_INTERLOCK          = 00000008     D
$SRVCODE_SHOWIDLE               = 00000003     D
$SRVCODE_STARTATTACHED          = 00000001     D
BUG$CHECK                         ********   X   04
CEB$C_LENGTH                    = 00000038     D
CEB$C_SLAVLNG                   = 00000044     D
CHECKBLOCK                        00000169 R   D   04
DS$GA_LASTADR                     ********   X   04
DS$GL_LOOKASIDE                   00000010 RG  D   02
DS$GL_MEMSIZE                     ********   X   04
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$MEMSIZE_X                      00000301 R   D   04
DS$SERVICE_DISPATCHER             ********   X   00
DS$_AP_NORMAL_BREAK             = 00660150     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INIT_FAIL                   = 00660140     D
DS$_INSFMEM                     = 00660050     D
DS$_INVCPU                      = 00660130     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MEM_ALLOC_ERR               = 00660138     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTALLOWMP                  = 00660148     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_UNSUPPDEV                   = 00660158     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$V_USER                      = 0000001C     D
DSX$MEMSIZE                       000002E0 RG  D   04
DS_ERRSUP                         ********   X   04
DYN$C_BUFIO                     = 00000013     D
DYN$C_CEB                       = 00000004     D
DYN$C_IRP                       = 0000000A     D
DYN$C_JIB                       = 0000002F     D
DYN$C_PCB                       = 0000000C     D
DYN$C_PQB                       = 0000000D     D
DYN$C_TQE                       = 0000000F     D
EXE$ALLOCATE                      000000DE RG  D   04
EXE$ALLOCBUF                      00000000 RG  D   04
EXE$ALLOCCEB                      00000004 RG  D   04
EXE$ALLOCIRP                      00000011 RG  D   04
EXE$ALLOCJIB                      00000008 RG  D   04
EXE$ALLOCPCB                      00000015 RG  D   04
EXE$ALLOCPQB                      0000001D RG  D   04
EXE$ALLOCTQE                      00000026 RG  D   04
EXE$ALONONPAGED                   00000060 RG  D   04
EXE$DEALLOCATE                    0000018F RG  D   04
EXE$DEANONPAGED                   00000106 RG  D   04
```

```
EXE$GL_NONPAGED                   00000000 R  D  02        PCB$B_WEFC                    0000002E    D
EXE$GL_SPLITADR                   0000000C RG D  02        PCB$C_LENGTH                  0000008C    D
IOC$GL_IRPBL                    = 00000018 R  D  02        PCB$K_LENGTH                  0000008C    D
IOC$GL_IRPFL                      00000014 RG D  02        PCB$L_ARB                     00000084    D
IRP$B_CARCON                      00000038    D          PCB$L_ASTQBL                  00000014    D
IRP$B_EFN                         00000022    D          PCB$L_ASTQFL                  00000010    D
IRP$B_PRI                         00000023    D          PCB$L_EFC2P                   00000058    D
IRP$B_RMOD                        0000000B    D          PCB$L_EFC3P                   0000005C    D
IRP$B_TYPE                        0000000A    D          PCB$L_EFCS                    00000050    D
IRP$C_LENGTH                      0000005C    D          PCB$L_EFCU                    00000054    D
IRP$K_LENGTH                      0000005C    D          PCB$L_EFWM                    0000004C    D
IRP$L_ARB                         00000050    D          PCB$L_JIB                     00000078    D
IRP$L_AST                         00000010    D          PCB$L_OWNER                   0000001C    D
IRP$L_ASTPRM                      00000014    D          PCB$L_PHD                     00000064    D
IRP$L_DIAGBUF                     00000044    D          PCB$L_PHYPCB                  00000018    D
IRP$L_EXTEND                      0000004C    D          PCB$L_PID                     00000060    D
IRP$L_IOQBL                       00000004    D          PCB$L_PQB                     0000004C    D
IRP$L_IOQFL                       00000000    D          PCB$L_SQBL                    00000004    D
IRP$L_IOSB                        00000024    D          PCB$L_SQFL                    00000000    D
IRP$L_IOST1                       00000034    D          PCB$L_STS                     00000024    D
IRP$L_IOST2                       00000038    D          PCB$L_UIC                     00000088    D
IRP$L_MEDIA                       00000034    D          PCB$L_WSSWP                   00000020    D
IRP$L_PID                         0000000C    D          PCB$L_WTIME                   00000028    D
IRP$L_SEGVBN                      00000040    D          PCB$Q_PRIV                    0000007C    D
IRP$L_SEQNUM                      00000048    D          PCB$T_LNAME                   00000068    D
IRP$L_SVAPTE                      0000002C    D          PCB$T_TERMINAL                00000044    D
IRP$L_TT_TERM                     00000038    D          PCB$W_APTCNT                  00000030    D
IRP$L_UCB                         0000001C    D          PCB$W_ASTCNT                  00000038    D
IRP$L_WIND                        00000018    D          PCB$W_BIOCNT                  0000003A    D
IRP$Q_NT_PRVMSK                   0000003C    D          PCB$W_BIOLM                   0000003C    D
IRP$W_ABCNT                       0000003C    D          PCB$W_DIOCNT                  0000003E    D
IRP$W_BCNT                        00000032    D          PCB$W_DIOLM                   00000040    D
IRP$W_BOFF                        00000030    D          PCB$W_GPGCNT                  00000034    D
IRP$W_CHAN                        00000028    D          PCB$W_GRP                     0000008A    D
IRP$W_FUNC                        00000020    D          PCB$W_MEM                     00000088    D
IRP$W_OBCNT                       0000003E    D          PCB$W_MTXCNT                  0000000E    D
IRP$W_SIZE                        00000008    D          PCB$W_PPGCNT                  00000036    D
IRP$W_STS                         0000002A    D          PCB$W_PRCCNT                  00000042    D
IRP$W_TT_PRMPT                    0000003C    D          PCB$W_SIZE                    00000008    D
JIB$C_LENGTH                    = 00000074    D          PCB$W_STATE                   0000002C    D
MASK                            = 0000000F    D          PCB$W_TMBU                    00000032    D
MEMPOOL$INIT                      000001E5 RG D  04        PQB$C_LENGTH                = 000008C8    D
MP$V_IL_ALLN                    = 00000004    D          PR$_IPL                     = 00000012    D
MP$V_IL_CNDB                    = 00000007    D          QIO$MINCORE                   ********  X   04
MP$V_IL_CNHN                    = 00000006    D          RSN$_NPDYNMEM               = 00000003    D
MP$V_IL_DELN                    = 00000005    D          SCRIPT$MINCORE                ********  X   04
MP$V_IL_DSBK                    = 00000008    D          SGN$C_IRPCNT                  ********  X   04
MP$V_IL_ERSP                    = 00000003    D          SIZ...                      = 00000001    D
MP$V_IL_GTLN                    = 00000009    D          SS$_INSFMEM                 = 00000124    D
MP$V_IL_NUMT                    = 00000000    D          SS$_NORMAL                  = 00000001    D
MP$V_IL_PRNT                    = 00000001    D          SYS$EXPREG                    ******** GX   04
MP$V_IL_RRPT                    = 00000002    D          T_CGF                         00000007 R  D  03
PCB$B_ASTACT                      0000000C    D          ULTRIX$B                      ********  X   04
PCB$B_ASTEN                       0000000D    D          ULTRIX$IOB                    ********  X   04
PCB$B_PRI                         0000000B    D          ULTRIX_B_ADD                  ********  X   04
PCB$B_PRIB                        0000002F    D          ULTRIX_IOB_ADD                ********  X   04
PCB$B_TYPE                        0000000A    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000001C (28.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	0000001C (28.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000302 (770.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$T1	=00000000	544 (2)	544 (2)
$ER	=00000002	268 (2)	#-268 (2)
$MODULE	00000000-R	152 (2)	268 (2)
$SRVCODE_BOOTATTACHED	=00000000	426 (2)	
$SRVCODE_CLEAR_INTERLOCK	=00000009	426 (2)	#-333 (2) #-426 (2)
$SRVCODE_GETSYSBUF	=00000004	426 (2)	
$SRVCODE_HALTATTACHED	=00000002	426 (2)	
$SRVCODE_INTERLOCK_TEST	=00000007	426 (2)	
$SRVCODE_PPSCB	=0000000A	426 (2)	
$SRVCODE_PRINTREV	=00000006	426 (2)	
$SRVCODE_RELSYSBUF	=00000005	426 (2)	
$SRVCODE_SET_INTERLOCK	=00000008	426 (2)	#-297 (2) #-400 (2)
$SRVCODE_SHOWIDLE	=00000003	426 (2)	
$SRVCODE_STARTATTACHED	=00000001	426 (2)	
BIT...	=00000018	96 (2)	96 (2)
BUG$CHECK	00000000-XR		329 (2) 443 (2) 485 (2)
CEB$C_LENGTH	=00000038		236 (2)
CEB$C_SLAVLNG	=00000044		237 (2)
CHECKBLOCK	00000169-R	436 (2)	#-402 (2)
DS$GA_LASTADR	00000000-XR		#-527 (2) #-538 (2) #-560 (2)
DS$GL_LOOKASIDE	00000010-R	138 (2)	#-559 (2)
DS$GL_MEMSIZE	00000000-XR		#-527 (2) #-632 (2)
DS$K_ERROR	=00000002	96 (2)	
DS$K_NORMAL	=00000001	96 (2)	
DS$K_SEVERE	=00000004	96 (2)	
DS$K_SUBSYS	=00000066	96 (2)	96 (2)
DS$K_WARNING	=00000000	96 (2)	
DS$MEMSIZE_X	00000301-R	634 (2)	#-631 (2)
DS$SERVICE_DISPATCHER	00000000-XR		115 (2) 297 (2) 333 (2) 400 (2) 426 (2)
DS$_AP_NORMAL_BREAK	=00660150	96 (2)	
DS$_ARITH	=006600D0	96 (2)	
DS$_ASBE	=00660118	96 (2)	
DS$_BADLINK	=006600F0	96 (2)	
DS$_BADTYPE	=006600E8	96 (2)	
DS$_BIIC	=00660120	96 (2)	
DS$_CHME	=006600A8	96 (2)	
DS$_CHMK	=006600E0	96 (2)	
DS$_DEVNAME	=00660108	96 (2)	
DS$_ERROR	=00660002	96 (2)	
DS$_FHWE	=00660068	96 (2)	
DS$_FRAGBUF	=00660080	96 (2)	
DS$_ICBUSY	=006600C8	96 (2)	
DS$_ICERR	=006600C0	96 (2)	
DS$_IHWE	=00660060	96 (2)	
DS$_ILLCHAR	=00660018	96 (2)	
DS$_ILLPAGCNT	=00660078	96 (2)	
DS$_ILLUNIT	=00660100	96 (2)	
DS$_INIT_FAIL	=00660140	96 (2)	
DS$_INSFMEM	=00660050	96 (2)	

```
DS$_INVCPU                      =00660130      96     (2)
DS$_IPL2HI                      =006600B8      96     (2)
DS$_IVADDR                      =00660040      96     (2)
DS$_IVVECT                      =00660038      96     (2)
DS$_KRNLSTK                     =00660090      96     (2)
DS$_LOGIC                       =00660070      96     (2)
DS$_MCHK                        =00660088      96     (2)
DS$_MEM_ALLOC_ERR               =00660138      96     (2)
DS$_MMOFF                       =00660058      96     (2)
DS$_NEEDUNIT                    =006600F8      96     (2)
DS$_NODE                        =00660128      96     (2)
DS$_NOPCS                       =00660110      96     (2)
DS$_NORMAL                      =00660001      96     (2)
DS$_NOSUPPORT                   =006600B1      96     (2)    #-630     (2)
DS$_NOTALLOWMP                  =00660148      96     (2)
DS$_NOTDON                      =00660030      96     (2)
DS$_NOTIMP                      =006600B0      96     (2)      96      (2)
DS$_NULLSTR                     =00660010      96     (2)
DS$_OVERFLOW                    =00660008      96     (2)
DS$_POWER                       =00660098      96     (2)
DS$_PROGERR                     =00660020      96     (2)
DS$_SEVERE                      =00660004      96     (2)
DS$_TRANSL                      =006600A0      96     (2)
DS$_TRUNCATE                    =00660028      96     (2)
DS$_UNEXPINT                    =006600D8      96     (2)
DS$_UNSUPPDEV                   =00660158      96     (2)
DS$_VASFULL                     =00660048      96     (2)
DS$_WARNING                     =00660000      96     (2)       96      (2)
DSA$GL_FLAGS                     0000FE00                        303     (2)      324     (2)      406     (2)      423     (2)
                                                                 524     (2)      535     (2)      631     (2)
DSA$V_USER                      =0000001C                      #-302     (2)    #-323     (2)    #-405     (2)    #-422     (2)
                                                               #-524     (2)    #-535     (2)    #-631     (2)
DSX$MEMSIZE                      000002E0-R    628     (2)
DS_ERRSUP                        00000000-XR                     268     (2)
DYN$C_BUFIO                     =00000013                      #-232     (2)
DYN$C_CEB                       =00000004                      #-235     (2)
DYN$C_IRP                       =0000000A                      #-244     (2)
DYN$C_JIB                       =0000002F                      #-240     (2)
DYN$C_PCB                       =0000000C                      #-247     (2)
DYN$C_PQB                       =0000000D                      #-251     (2)
DYN$C_TQE                       =0000000F                      #-255     (2)
EXE$ALLOCATE                     000000DE-R    360     (2)    #-321     (2)
EXE$ALLOCBUF                     00000000-R    231     (2)
EXE$ALLOCCEB                     00000004-R    234     (2)
EXE$ALLOCIRP                     00000011-R    243     (2)
EXE$ALLOCJIB                     00000008-R    239     (2)
EXE$ALLOCPCB                     00000015-R    246     (2)
EXE$ALLOCPQB                     0000001D-R    250     (2)
EXE$ALLOCTQE                     00000026-R    254     (2)
EXE$ALONONPAGED                  00000060-R    296     (2)    #-259     (2)
EXE$DEALLOCATE                   0000018F-R    462     (2)    #-420     (2)
EXE$DEANONPAGED                  00000106-R    399     (2)
EXE$GL_NONPAGED                  00000000-R    131     (2)      319     (2)      403     (2)    #-561     (2)
EXE$GL_SPLITADR                  0000000C-R    136     (2)    #-409     (2)    #-584     (2)
IOC$GL_IRPBL                    =00000018-R    148     (2)      414     (2)    #-418     (2)
IOC$GL_IRPFL                     00000014-R    146     (2)      147     (2)      148     (2)      311     (2)      564     (2)
IRP$B_TYPE                       0000000A                       #-263     (2)
```

```
IRP$C_LENGTH                 0000005C                    236    (2)     237    (2)    #-256    (2)    #-306    (2)
                                                       #-567    (2)   #-580    (2)
IRP$W_SIZE                   00000008                  #-262    (2)   #-439    (2)
JIB$C_LENGTH                =00000074                  #-241    (2)
MASK                        =0000000F     123    (2)  #-256    (2)   #-299    (2)    #-300    (2)    #-306    (2)
                                                       #-437    (2)   #-440    (2)    #-441    (2)
MEMPOOL$INIT                 000001E5-R    516    (2)
MP$V_IL_ALLN                =00000004     111    (2)  #-297    (2)   #-333    (2)
MP$V_IL_CNDB                =00000007     111    (2)
MP$V_IL_CNHN                =00000006     111    (2)
MP$V_IL_DELN                =00000005     111    (2)  #-400    (2)   #-426    (2)
MP$V_IL_DSBK                =00000008     111    (2)
MP$V_IL_ERSP                =00000003     111    (2)
MP$V_IL_GTLN                =00000009     111    (2)
MP$V_IL_NUMT                =00000000     111    (2)
MP$V_IL_PRNT                =00000001     111    (2)
MP$V_IL_RRPT                =00000002     111    (2)
PCB$C_LENGTH                 0000008C                  #-248    (2)
PQB$C_LENGTH                =000008C8                  #-252    (2)
PR$_IPL                     =00000012                  #-304    (2)   #-325    (2)    #-407    (2)    #-424    (2)
QIO$MINCORE                  00000000-XR               #-519    (2)
RSN$_NPDYNMEM               =00000003                  #-421    (2)
SCRIPT$MINCORE               00000000-XR               #-518    (2)
SGN$C_IRPCNT                 00000000-XR               #-567    (2)
SS$_INSFMEM                 =00000124                  #-267    (2)
SS$_NORMAL                  =00000001                  #-317    (2)   #-633    (2)
SYS$EXPREG                   00000000-XR                 544    (2)
T_CGF                        00000007-R    153    (2)    268    (2)
ULTRIX$B                     00000000-XR               #-523    (2)     555    (2)
ULTRIX$IOB                   00000000-XR               #-523    (2)     554    (2)      555    (2)
ULTRIX_B_ADD                 00000000-XR               #-554    (2)
ULTRIX_IOB_ADD               00000000-XR               #-553    (2)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ASNPUSH	1	544 (2)	544 (2)
$CEBDEF	2	98 (2)	98 (2)
$DEF	1	96 (2)	
$DEFINI	1	97 (2)	100 (2) 101 (2) 102 (2) 103 (2)
			104 (2) 105 (2) 106 (2) 107 (2)
			108 (2) 109 (2) 110 (2) 97 (2)
			98 (2) 99 (2)
$DS_DSADEF	5	97 (2)	97 (2)
$DS_DSDEF	3	96 (2)	96 (2)
$DS_SRVCODE_DEF	2	426 (2)	297 (2) 333 (2) 400 (2) 426 (2)
$DYNDEF	2	99 (2)	99 (2)
$EQU	1	96 (2)	96 (2)
$EQULS1	1	96 (2)	96 (2)
$EQULST	1	96 (2)	96 (2)
$EXPREG_S	1	544 (2)	544 (2)
$GBLINI	2	96 (2)	96 (2)
$IPLDEF	1	100 (2)	100 (2)
$IRPDEF	4	101 (2)	101 (2)
$JIBDEF	3	102 (2)	102 (2)
$MP_CLEAR_INTERLOCK	1	333 (2)	333 (2) 426 (2)
$MP_ILDEFS	1	111 (2)	111 (2)
$MP_SET_INTERLOCK	1	297 (2)	297 (2) 400 (2)
$PCBDEF	4	103 (2)	103 (2)
$PQBDEF	3	104 (2)	104 (2)
$PRDEF	5	105 (2)	105 (2)
$PUSHADR	1	268 (2)	268 (2) 544 (2)
$RSNDEF	1	106 (2)	106 (2)
$SHBDEF	1	107 (2)	107 (2)
$SHDDEF	3	108 (2)	108 (2)
$SSDEF	21	109 (2)	109 (2)
$TQEDEF	2	110 (2)	110 (2)
$VIELD1	1	96 (2)	
ASSUME	1	236 (2)	236 (2) 237 (2)
BR_IF_USER	1	631 (2)	631 (2)
BUG_CHECK	1	329 (2)	329 (2) 443 (2) 485 (2)
DSBINT	1	304 (2)	304 (2) 407 (2)
ENBINT	1	325 (2)	325 (2) 424 (2)
ERRSUP_S	1	268 (2)	268 (2)
MODNAM	1	152 (2)	152 (2)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ACBL	00F1	531 (2) 580 (2)
ADDL	00C0	299 (2) 440 (2) 474 (2) 477 (2) 480 (2)
ADDL2	00C0	519 (2) 521 (2) 523 (2)
ADDL3	00C1	368 (2) 470 (2)
ASHL	0078	543 (2)
BBS	00E0	302 (2) 323 (2) 405 (2) 422 (2) 524 (2) 535 (2) 631 (2)
BEQL	0013	301 (2) 313 (2) 364 (2) 367 (2) 465 (2)
BEQLU	0013	468 (2)
BGTRU	001A	366 (2) 467 (2)
BICL	00CA	300 (2) 441 (2)
BICL2	00CA	557 (2)
BISL3	00C9	546 (2)
BISW2	00A8	533 (2)
BITL	00D3	437 (2)
BLBC	00E9	261 (2)
BLSSU	001F	307 (2) 410 (2)
BNEQ	0012	438 (2) 442 (2) 472 (2) 479 (2)
BRB	0011	233 (2) 238 (2) 242 (2) 245 (2) 249 (2) 253 (2) 270 (2)
		318 (2) 327 (2) 419 (2) 487 (2) 540 (2) 569 (2)
BSBB	0010	259 (2) 321 (2) 402 (2) 420 (2)
CALLS	00FB	268 (2) 297 (2) 333 (2) 400 (2) 426 (2) 544 (2)
CLRL	00D4	330 (2) 562 (2)
CMPL	00D1	312 (2) 365 (2) 409 (2) 466 (2) 471 (2) 478 (2)
CMPW	00B1	306 (2)
DIVL2	00C6	529 (2)
DIVL3	00C7	520 (2) 632 (2)
HALT	0000	269 (2) 486 (2)
INCL	00D6	534 (2) 547 (2)
JSB	0016	329 (2) 443 (2) 485 (2)
MFPR	00DB	304 (2) 407 (2)
MOVAB	009E	319 (2) 373 (2) 403 (2) 539 (2) 554 (2) 555 (2) 564 (2)
MOVAL	00DE	311 (2) 371 (2) 414 (2) 416 (2) 417 (2) 418 (2) 556 (2)
MOVAQ	007E	542 (2)
MOVL	00D0	314 (2) 315 (2) 316 (2) 317 (2) 361 (2) 362 (2) 363 (2)
		369 (2) 372 (2) 415 (2) 463 (2) 464 (2) 469 (2) 473 (2)
		476 (2) 481 (2) 482 (2) 483 (2) 518 (2) 532 (2) 538 (2)
		553 (2) 559 (2) 560 (2) 561 (2) 565 (2) 566 (2) 574 (2)
		575 (2) 576 (2) 577 (2) 578 (2) 584 (2) 630 (2) 633 (2)
MOVPSL	00DC	257 (2)
MOVW	00B0	262 (2)
MOVZBL	009A	248 (2) 256 (2)
MOVZBW	009B	263 (2)
MOVZWL	003C	241 (2) 252 (2) 267 (2) 421 (2) 439 (2)
MTPR	00DA	304 (2) 325 (2) 407 (2) 424 (2)
MULL3	00C5	522 (2)
POPR	00BA	260 (2) 545 (2) 586 (2)
PUSHAL	00DF	268 (2)
PUSHAQ	007F	544 (2)
PUSHL	00DD	232 (2) 235 (2) 240 (2) 244 (2) 247 (2) 251 (2) 255 (2)
		258 (2) 268 (2) 297 (2) 333 (2) 400 (2) 426 (2) 475 (2)

```
                         544  (2)
PUSHR            00BB    517  (2)
RET              0004    635  (2)
RSB              0005    266  (2)    334  (2)    374  (2)    428  (2)    445  (2)    484  (2)    587  (2)
SUBL3            00C3    370  (2)    527  (2)    567  (2)    583  (2)
TSTL             00D5    265  (2)    320  (2)    404  (2)    444  (2)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ALIGN	130	(2)														
.ASCIC	152	(2)	154	(2)	329	(2)	443	(2)	485	(2)						
.DSABL	271	(2)	336	(2)	429	(2)										
.ENABL	230	(2)	295	(2)	398	(2)										
.END	638	(2)														
.ENDC	236	(2)	237	(2)	268	(2)	304	(2)	325	(2)	407	(2)	424	(2)	544	(2)
	96	(2)														
.ENDM	100	(2)	101	(2)	102	(2)	103	(2)	104	(2)	105	(2)	106	(2)	107	(2)
	108	(2)	109	(2)	110	(2)	111	(2)	152	(2)	236	(2)	268	(2)	297	(2)
	304	(2)	325	(2)	329	(2)	333	(2)	400	(2)	426	(2)	544	(2)	631	(2)
	96	(2)	97	(2)	98	(2)	99	(2)								
.ENDR	96	(2)														
.ENTRY	628	(2)														
.EXTRN	115	(2)														
.GLOBL	544	(2)														
.IDENT	6	(1)														
.IF	236	(2)	237	(2)	268	(2)	304	(2)	325	(2)	407	(2)	424	(2)	544	(2)
	96	(2)														
.IFF	236	(2)	237	(2)	268	(2)	304	(2)	325	(2)	407	(2)	424	(2)	544	(2)
	96	(2)														
.IIF	268	(2)	96	(2)												
.IRP	96	(2)														
.LIBRARY	88	(2)	89	(2)	90	(2)										
.LIST	97	(2)														
.LONG	132	(2)	133	(2)	134	(2)	137	(2)	139	(2)	147	(2)				
.MACRO	96	(2)														
.MDELETE	297	(2)	333	(2)	400	(2)	426	(2)	96	(2)						
.NLIST	7	(1)	97	(2)												
.NOCROSS	100	(2)	101	(2)	102	(2)	103	(2)	104	(2)	105	(2)	106	(2)	107	(2)
	108	(2)	109	(2)	110	(2)	97	(2)	98	(2)	99	(2)				
.PAGE	155	(2)	272	(2)	337	(2)	375	(2)	430	(2)	446	(2)	488	(2)	589	(2)
.PSECT	125	(2)	151	(2)	157	(2)	490	(2)								
.RESTORE	100	(2)	101	(2)	102	(2)	103	(2)	104	(2)	105	(2)	106	(2)	107	(2)
	108	(2)	109	(2)	110	(2)	97	(2)	98	(2)	99	(2)				
.SAVE	100	(2)	101	(2)	102	(2)	103	(2)	104	(2)	105	(2)	106	(2)	107	(2)
	108	(2)	109	(2)	110	(2)	97	(2)	98	(2)	99	(2)				
.SBTTL	156	(2)	273	(2)	338	(2)	376	(2)	431	(2)	447	(2)	489	(2)	590	(2)
.TITLE	5	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	1	#-632 (2)
R0	60	#-261 (2) #-267 (2) #-315 (2) #-316 (2) #-317 (2) #-330 (2)
		#-361 (2) #-362 (2) #-363 (2) #-365 (2) #-368 (2) #-369 (2)
		#-370 (2) #-371 (2) #-372 (2) 373 (2) #-409 (2) #-415 (2)
		#-416 (2) 417 (2) 418 (2) #-421 (2) #-437 (2) #-439 (2)
		#-466 (2) #-469 (2) #-470 (2) #-473 (2) #-476 (2) #-478 (2)
		#-481 (2) #-482 (2) #-483 (2) #-518 (2) #-519 (2) #-520 (2)
		#-521 (2) #-522 (2) #-523 (2) #-531 (2) #-532 (2) #-533 (2)
		#-534 (2) 539 (2) #-543 (2) #-544 (2) #-546 (2) #-547 (2)
		#-560 (2) #-567 (2) #-575 (2) #-576 (2) #-577 (2) #-578 (2)
		#-580 (2) #-583 (2) #-630 (2) #-633 (2)
R1	44	#-241 (2) #-248 (2) #-252 (2) #-256 (2) #-258 (2) #-260 (2)
		#-262 (2) #-299 (2) #-300 (2) #-306 (2) #-365 (2) #-368 (2)
		#-370 (2) #-439 (2) #-440 (2) #-441 (2) #-470 (2) #-474 (2)
		#-480 (2) #-483 (2) #-520 (2) #-521 (2) #-522 (2) #-528 (2)
		#-529 (2) #-531 (2) #-532 (2) #-538 (2) 539 (2) #-542 (2)
		544 (2) #-545 (2) #-553 (2) 554 (2) 555 (2) 556 (2)
		#-557 (2) #-559 (2) #-561 (2) #-562 (2) #-583 (2)
R2	31	#-262 (2) #-263 (2) #-311 (2) #-312 (2) #-314 (2) #-315 (2)
		#-316 (2) #-362 (2) #-363 (2) #-372 (2) #-373 (2) #-463 (2)
		#-464 (2) #-475 (2) #-476 (2) #-477 (2) #-480 (2) #-481 (2)
		#-482 (2) #-517 (2) #-564 (2) #-565 (2) #-566 (2) #-574 (2)
		#-576 (2) #-577 (2) #-586 (2)
R3	29	#-260 (2) #-319 (2) #-320 (2) #-361 (2) #-368 (2) #-369 (2)
		#-370 (2) 371 (2) #-403 (2) #-404 (2) #-414 (2) #-415 (2)
		416 (2) #-417 (2) #-463 (2) #-464 (2) #-466 (2) #-469 (2)
		#-471 (2) #-473 (2) #-474 (2) #-517 (2) #-568 (2) #-574 (2)
		#-575 (2) #-578 (2) #-580 (2) #-584 (2) #-586 (2)
SP	14	#-257 (2) #-263 (2) #-265 (2) #-304 (2) #-325 (2) #-407 (2)
		#-424 (2) #-444 (2) #-470 (2) #-471 (2) #-477 (2) #-478 (2)
		542 (2) #-546 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	40	00:00:00.03	00:00:00.21
Command processing	930	00:00:00.19	00:00:01.05
Pass 1	864	00:00:03.46	00:00:06.14
Symbol table sort	0	00:00:00.32	00:00:00.32
Pass 2	6	00:00:00.87	00:00:01.53
Symbol table output	0	00:00:00.04	00:00:00.15
Psect synopsis output	0	00:00:00.00	00:00:00.00
Cross-reference output	4	00:00:00.20	00:00:00.36
Assembler run totals	1844	00:00:05.11	00:00:09.76

The working set limit was 3400 pages.
208285 bytes (407 pages) of virtual memory were used to buffer the intermediate code.

```
There were 60 pages of symbol table space allocated to hold 1159 non-local and 30 local symbols.
638 source lines were read in Pass 1, producing 83 object records in Pass 2.
137 pages of virtual memory were used to define 37 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	10
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	7
SYS$COMMON:[SYSLIB]LIB.MLB;2	9
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	37

```
1516 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:MEMALC.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 218 Libraries, Macros, and External Symbols
(1) 279 Own Storage
(1) 324 Data Storage
(1) 333 MapMem Routine - Map all of memory
(1) 590 MAPMEM$IOSPACE Setup page tables for I/O space
(3) 667 ACCESSABLE check a page for accessability
(4) 717 STACKPROT
(5) 775 MAP FREE MEMORY PROCEDURE.
(7) 867 GET A BLOCK OF VIRTUAL MEMORY.
(9) 1045 RELEASE A BLOCK OF VIRTUAL MEMORY.
(12) 1250 TURN MEMORY MANAGEMENT ON.
(13) 1302 TURN MEMORY MANAGEMENT OFF.
(14) 1357 DSR$MMENABLE Enable to disable memory management
(15) 1407 SET PROTECTION ON PAGES.
(18) 1567 CALCULATE PTE ADDRESS SUBROUTINE
(20) 1651 DSX$MAPDBGBLOCK - MAP A BLOCK OF MEMORY FOR THE DEBUGGER [23]
(21) 1692 DSX$FREEDBGSYM - FREE MEMORY MAPPED TO DEBUGGER SYMTABS
(22) 1722 MAP_DEBUGGER - MAP THE DEBUGGER'S MEMORY SPACE [23]
(23) 1747 UNMAP_DEBUGGER - UNMAP THE DEBUGGER'S MEMORY SPACE [23]

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element MEMMGT.MAR
0000     2 ; *7    11-DEC-1986 14:11:28 BAGGETT "MM FINISHED FOR RT-VAX"
0000     3 ; *6     8-AUG-1986 13:39:52 SILVER "FASTER TO CREATE PAGE TABLES"
0000     4 ; *5    24-JUL-1986 15:49:11 SILVER "<no reason given>"
0000     5 ; *4    23-JUL-1986 14:26:35 SALEM "DONE"
0000     6 ; *3     5-JUN-1986 14:17:31 BAGGETT "<no reason given>"
0000     7 ; *2     3-APR-1986 11:05:55 ANDELLA "<no reason given>"
0000     8 ; *1     6-FEB-1986 15:20:00 DS ""
0000     9 ; DEC/CMS REPLACEMENT HISTORY, Element MEMMGT.MAR
0000    10 
0000    11         .TITLE  MEMMGT  *** MEMMGT Memory management setup/control                ;G:7
0000    12         .IDENT  /10-39/                                                          ;G:7
0000    13         .NoShow Conditionals
0000    14         .DSABL  GBL
0000    15 
0000    16 ;
0000    17 ; COPYRIGHT (C) 1977, 1981, 1983, 1984, 1985, 1986                               ;G:2
0000    18 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    19 ;
0000    20 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    21 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    22 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    23 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    24 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    25 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    26 ; REMAIN IN DEC.
0000    27 ;
0000    28 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    29 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    30 ; CORPORATION.
0000    31 ;
0000    32 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    33 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    34 
0000    35 ;++
0000    36 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    37 ;
0000    38 ; ABSTRACT:
0000    39 ;
0000    40 ; ENVIRONMENT:
0000    41 ;
0000    42 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    43 ;
0000    44 ; MODIFIED BY:
0000    45 ;                KEN CHAPMAN     30-NOV-77       VERSION 02
0000    46 ;       01       DSPR #63 - FIXED $DS_GETBUF_L OVERLAYING HARDWARE P-TABLES.
0000    47 ;
0000    48 ;                NICK HOWGATE    09-JAN-78       VERSION 03 (ESSAA-3.06)
0000    49 ;       02       DSPR #41 - TURN ON MEMORY MANAGEMENT FIX.
0000    50 ;       03       ADDED $SETPRT ROUTINE.
0000    51 ;
0000    52 ;                KEN CHAPMAN     26-JAN-78       VERSION 04 (ESSAA-3.07)
0000    53 ;       04       MORE FIXES TO MEMORY MANAGEMENT PAGE TABLES.
0000    54 ;
0000    55 ;                KEN CHAPMAN     06-APR-78       VERSION 05 (ESSAA-4.01)
0000    56 ;       05       ADDED P1 AND SYSTEM BUFFER ALLOCATION CAPABILITY.
0000    57 ;                Roger Riggs     12-Jun-78       Version 06 (ESSAA-4.03)
```

```
                     0000     58 ;       06      Fixed Release of P1 or SYS space conflict of PTE's
                     0000     59 ;       07      MAPMEM cleanup and set DS$GL_MEMSIZE to bytes of physical
                     0000     60 ;               memory.
                     0000     61 ;               N. HOWGATE      14-NOV-78       VERSION 07 (ESSAA-5.01)
                     0000     62 ;       08      CLEAR MACHINE CHECKS CORRECTLY
                     0000     63 ;       09      FIX BUG IN $DS_SETPRT
                     0000     64 ;       10      Removed dependence of MAPFREE on L$L_UNITS
                     0000     65 ;
                     0000     66 ;       11      Added processor dependent I/O space mapping
                     0000     67 ;               Roger Riggs     6-Sept-1979
                     0000     68 ;       12      Added IO$GQ_PHYSICAL, used to determine where to map
                     0000     69 ;               I/O space in P1 space.
                     0000     70 ;               Roger Riggs, 24-Jan-1980,  Version 5.2
                     0000     71 ;       12      Re-arranged system page table to allow system space addresses
                     0000     72 ;               to map 1 for 1 P0 addresses in the range %X10000 to the
                     0000     73 ;               top of the P0 page tables.  References to this range will
                     0000     74 ;               reference the same addresses with or without memory management
                     0000     75 ;               enabled.
                     0000     76 ;               Roger Riggs, 5-Mar-1980, Version 5.3
                     0000     77 ;       13      Made page at APT$AT_APTTXT valid, affects APT when
                     0000     78 ;               Memory Management is on.
                     0000     79 ;               Roger Riggs, 12-Mar-1980, Version 5.3
                     0000     80 ;       14      Added byte to save state of memory managment, allows
                     0000     81 ;               Memory managment to be enabled or disabled after cleanup
                     0000     82 ;               to restore it to what the operator indicated.
                     0000     83 ;               Also if the operator enabled memory managment then the program
                     0000     84 ;               is not allowed to turn it off.  DS$GB_MM_ENB is set
                     0000     85 ;               by CLI command SET MM ON and SET MM OFF.
                     0000     86 ;               Also added routines and support for XDELTA
                     0000     87 ;               Roger Riggs, 21-May-1980, Version 5.4
                     0000     88 ;       15      Removed upper limit on amount of memory checked
                     0000     89 ;               Fix DS$EXPREG to set the protection of pages allocated to UW,
                     0000     90 ;               In DS$MMON AND DS$MMOFF return the previous state of MM as
                     0000     91 ;               SS$_WASSET OR SS$_WASCLR and return DS$_WARNING from
                     0000     92 ;               DS$MMOFF if operator set MMON and program is disallowed from
                     0000     93 ;               turning it off, Add code to flush the translation buffer
                     0000     94 ;               before turning MM ON, Set protections on stacks so
                     0000     95 ;               underflows will can be caught by MM.
                     0000     96 ;
                     0000     97 ;               Dave Butenhof   17-jul-1980
                     0000     98 ;       16      Call MAPFREE from MAPMEM to free up memory; to allow
                     0000     99 ;               GETBUF from Supervisor before program is loaded.
                     0000    100 ;               Specifically, for HELP utility.
                     0000    101 ;
                     0000    102 ;               Roger Riggs, 17-Sep-1980, Version 6.0
                     0000    103 ;       17      Changed MAPFREE to check program loaded flag to determine
                     0000    104 ;               how much of low memory to map.
                     0000    105 ;
                     0000    106 ;               Dave Butenhof, 23-feb-1981, version 6.3
                     0000    107 ;       18      Change a word-relative to longword-relative.
                     0000    108 ;       19      Fix two bugs: in EXE$SETPRT, correct return address array
                     0000    109 ;               start address--BICL3 first two operands were reversed.
                     0000    110 ;               in EXE$CNTREG, Comparison of P0 and P1/SYS PTE resulted in
                     0000    111 ;               ERRSUP if M bit was altered, and ignored V and PFN; fix the
                     0000    112 ;               compare mask.
                     0000    113 ;
                     0000    114 ;               - Dave Butenhof, 08-Jun-1981, version 6.4
```

```
0000   115 ;       20      Fix SETPRT some more.  Do a CHMK to force cpu
0000   116 ;               to KERNEL mode, so we can use MFPR.
0000   117 ;
0000   118 ;       21      - dave butenhof, 3-sep-1981, version 6.5
0000   119 ;               alter to use DS with new XDELTA
0000   120 ;
0000   121 ;       22      Jack Stansbury, Version 6.8, May 28, 1982
0000   122 ;               Add an entry point to the ExpReg routine for use by CRD.
0000   123 ;               In stand-alone, ExpReg was allocating space beginning at 200,
0000   124 ;               which is a no-no! Fix it to expand above the Supervisor (as it
0000   125 ;               does on-line).
0000   126 ;
0000   127 ;       23      Richard Brown, Version 6.8, 25-June-82
0000   128 ;               Addition of routines which map a block
0000   129 ;               of memory for the debugger to use.
0000   130 ;
0000   131 ;       24      Richard Brown, Version 6.9, 4-August-82
0000   132 ;               Addition of routine to deallocate memory assigned
0000   133 ;               to the debugger's symbol tables.
0000   134 ;
0000   135 ;       25      Bob Bergazzi    5-Aug-82        Version 6.9
0000   136 ;               Changed MAPFREE to call DSR$CheckLoad (SHOWMEM)
0000   137 ;               to see if a program is loaded - fixes "SET MM ON,
0000   138 ;               ATTACH, Translation not valid fault" bug.
0000   139 ;
0000   140 ;       26      Jack Stansbury  25-Aug-1982     Version 6.9
0000   141 ;               Added two other vaiables: DS$GL_Actual_MemSize and
0000   142 ;               DS$GL_Set_MemSize. The former is the actual size of the memory
0000   143 ;               of the machine. The latter is the size that the operator wishes
0000   144 ;               to set the memory. Also added .Library statements, and changed
0000   145 ;               the PSect from SEP to what they should be. Also changed the
0000   146 ;               insufficient memory message from a PrintF to a Print.
0000   147 ;
0000   148 ;       27      Jack Stansbury  25-Aug-1982     Version 6.9
0000   149 ;               Fixed several truncation errors.
0000   150 ;
0000   151 ;       28      Bob Bergazzi    2-Nov-1982      Version 6.9
0000   152 ;               Changed the ACCESSABLE routine so that it only copies the
0000   153 ;               first quadword of a page rather than every quadword on the
0000   154 ;               page. This results in considerable savings of startup time,
0000   155 ;               especially for NEBULA systems with large amounts of memory
0000   156 ;               (greater than 3 Meg took longer than 10 seconds, causing
0000   157 ;               APT to consider the supervisor hung).
0000   158 ;
0000   159 ;       29      Bob Bergazzi    29-Aug-1983     Version 6.13
0000   160 ;               Changed the value of the minimum memory size required from
0000   161 ;               256K to 512K bytes.
0000   162 ;
0000   163 ;       30      Bob Bergazzi    26-Dec-1984     Version 8.0
0000   164 ;               Turned interrupts off during the STACKPROT routine. If
0000   165 ;               a clock interrupt (or any other interrupt) happened
0000   166 ;               while memory management was turned on in this routine,
0000   167 ;               the console detects an interrupt stack not valid and halts,
0000   168 ;               because this routine sets protection no access before it
0000   169 ;               resets it correctly.
0000   170 ;
0000   171 ;       31      Domenic Andella 30-Jan-1985     Version 8.1
```

```
                                0000   172 ;               Modified routine EXE$EXPREG to receive additional
                                0000   173 ;               argument when called via $DS_GETBUF or $DS_GETSYSBUF.
                                0000   174 ;               Each pages PTE will be marked to indicate which service
                                0000   175 ;               was called.
                                0000   176 ;               Modified routine EXE$CNTREG to receive additional
                                0000   177 ;               argument when called via $DS_RELBUF or $DS_GETSYSBUF
                                0000   178 ;               in order to control deallocation of pages.
                                0000   179 ;
                                0000   180 ;       32      Domenic Andella 28-May-1985     Version 8.2
                                0000   181 ;               Modified routines DSX$MMON/OFF to prohibit
                                0000   182 ;               use if active mp system.
                                0000   183 ;
                                0000   184 ;       33      Ted Salem       18-September-1985  Version 8.2
                                0000   185 ;               Modified above change (#32) to prohibit
                                0000   186 ;               the change of status of memory management rather
                                0000   187 ;               than prohibit the use of DSX$MMON/OFF services.
                                0000   188 ;
                                0000   189 ;       34      Bob Bergazzi    21-NOV-85       Version 9.0
                                0000   190 ;               Changed first SOBGTR in the EXE$EXPREG routine so that a
                                0000   191 ;               GETBUF which asks for all memory wil get it. It was off
                                0000   192 ;               by one.
                                0000   193 ;                                                                       ;G:2
                                0000   194 ;       35      Domenic Andella 4-Apr-1986      Version 10.0            ;G:2
                                0000   195 ;               For DS$_RELSYSBUF service reset prot of deallocated     ;G:2
                                0000   196 ;               page to full access (UW). This is necessary for mp      ;G:2
                                0000   197 ;               systems that use the service and modify protection      ;G:2
                                0000   198 ;               codes for an Attached processors SCB and stacks.        ;G:2
                                0000   199 ;                                                                       ;G:2
                                0000   200 ;       36      M. Baggett      5-Jun-1986      Version 10.1            ;G:3
                                0000   201 ;               Do not allow set memory management on RT-VAX.           ;G:3
                                0000   202 ;                                                                       ;G:4
                                0000   203 ;       37      Ted Salem       27-June-1986    Version 10.1            ;G:4
                                0000   204 ;               Changed the method used to Map PO space for Andromeda   ;G:4
                                0000   205 ;               Also commented out change #36 for temp fix.             ;G:4
                                0000   206 ;                                                                       ;G:6
                                0000   207 ;       38      Doug Silver     8-Aug-1986      Version 10.1            ;G:6
                                0000   208 ;               Changed MAPMEM to check increments of 1 Meg until no    ;G:6
                                0000   209 ;               access and then check from last good memory access in   ;G:6
                                0000   210 ;               increments of 1 K.  This way, each full Meg will only   ;G:6
                                0000   211 ;               have to be checked 1 time for accessability instead of  ;G:6
                                0000   212 ;               1024 times.                                             ;G:6
                                0000   213 ;                                                                       ;G:7
                                0000   214 ;       39      M. Baggett      7-Oct-1986      Version 10.3            ;G:7
                                0000   215 ;               Set special case for RTUVAX memory management.          ;G:7
                                0000   216 ;--
```

```
                     0000    218          .SubTitle       Libraries, Macros, and External Symbols
                     0000    219
                     0000    220 ;
                     0000    221 ; INCLUDE FILES:
                     0000    222 ;
                     0000    223
                     0000    224          .Library        /Sys$Library:Lib/                                    ;[26
                     0000    225          .Library        /$DS/                                               ;[26
                     0000    226          .Library        /$Diag/                                             ;[26
                     0000    227
                     0000    228 ;
                     0000    229 ; EQUATED SYMBOLS:
                     0000    230 ;
                     0000    231
                     0000    232          CMKDEF
                     0000    233          DSFDEF
                     0000    234
                     0000    235          $CNTREGDEF
                     0000    236          $DS_BITDEF
                     0000    237          $DS_CFDEF
                     0000    238          $DS_DSADEF
                     0000    239          $DS_DSDEF
                     0000    240          $DS_ENVDEF
                     0000    241          $DS_GETBUF_DEF
                     0000    242          $DS_GETMEM_DEF
                     0000    243          $DS_HDRDEF
                     0000    244          $DS_RELBUF_DEF
                     0000    245          $DS_RELMEM_DEF
                     0000    246          $DS_SCBDEF
                     0000    247          $DS_TypeDef                         ; The typecodes                 [26]
                     0000    248
                     0000    249          $EXPREGDEF
                     0000    250          $IRPDEF
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .PSECT  $ABS$,ABS
                     FE70                 .=0
                     0000                 .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
00000004             0000                         .BLKL   1
                     0004                 .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
00000008             0004                         .BLKL   1
                     0008                 .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
0000000A             0008                         .BLKW   1
                     000A                 .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
0000000B             000A                         .BLKB   1
                     000B                 .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
0000000C             000B                         .BLKB   1
                     000C                 .IIF    NB,IRP$L_PID,   IRP$L_PID:
00000010             000C                         .BLKL   1
                     0010                 .IIF    NB,IRP$L_AST,   IRP$L_AST:
00000014             0010                         .BLKL   1
                     0014                 .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
00000018             0014                         .BLKL   1
                     0018                 .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
0000001C             0018                         .BLKL   1
                     001C                 .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
00000020             001C                         .BLKL   1
                     0020                 .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
```

```
00000022  0020                          .BLKW   1
          0022              .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
00000023  0022                          .BLKB   1
          0023              .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
00000024  0023                          .BLKB   1
          0024              .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
00000028  0024                          .BLKL   1
          0028              .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
0000002A  0028                          .BLKW   1
          002A              .IIF    NB,IRP$W_STS,   IRP$W_STS:
0000002C  002A                          .BLKW   1
          002C              .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
00000030  002C                          .BLKL   1
          0030              .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
00000032  0030                          .BLKW   1
          0032              .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
00000034  0032                          .BLKW   1
          0034              .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
          0034              .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
00000038  0034                          .BLKL   1
          0038              .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
          0038              .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
          0038              .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
00000039  0038                          .BLKB   1
0000003C  0039                          .BLKB   3
          003C              .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
          003C              .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
          003C              .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                          .BLKW   1
          003E              .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                          .BLKW   1
          0040              .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                          .BLKL   1
          0044              .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                          .BLKL   1
          0048              .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                          .BLKL   1
          004C              .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                          .BLKL   1
          0050              .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                          .BLKL   1
00000058  0054                          .BLKL   1
0000005C  0058                          .BLKL   1
          005C              .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
          005C              .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
          0000              .RESTORE
          0000    251       $PHDDEF
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  FE70              .=0
          0000              .RESTORE
          0000    252       $PRDEF
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  FE70              .=0
          0000              .RESTORE
          0000    253       $PRTDEF
```

```
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  FE70                .=0
                    0000                .RESTORE
                    0000    254         $PSLDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  FE70                .=0
                    0000                .RESTORE
                    0000    255         $PTEDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  FE70                .=0
                    0000                .RESTORE
                    0000    256           $VIELD  PTE,<29-9>,<<IO,,M>>
                    0000    257         $SETPRTDEF
                    0000    258         $SSDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  FE70                .=0
                    0000                .RESTORE
                    0000    259
                    0000    260
          0000007E  0000    261 K_P1PAGES=126                       ; Number of P1 pages to create
                    0000    262
                    0000    263         .EXTRN  POPT_BASE,      MAP$IOSPACE,    MEMPOOL$INIT
                    0000    264         .EXTRN  MMG$GL_SPTBASE, DS$AQ_SYSSRV,   DS$GA_LASTADR
                    0000    265         .EXTRN  SYS$UNWIND,     DS$GL_BUFCNT,   DSR$SETIMR_INI
                    0000    266         .EXTRN  DS$GQ_PHYADR,   DS_ERRSUP,      DSX$Print                       ;[26
                    0000    267         .EXTRN  IO$GQ_PHYSICAL
                    0000    268         .EXTRN  SCB_UNKINT,STACK_BASE,ISTKPTR,KSTKPTR,ESTKPTR,SSTKPTR,USTKPTR
                    0000    269         .EXTRN  DS$K_PRGSIZ,    DS$A_PRGBGN,    DS$GL_FLAGS
                    0000    270         .EXTRN  DEBUG_LO,       DEBUG_HI                                        ;[23
                    0000    271         .EXTRN  SYMTAB_HI                                                       ;[24
                    0000    272         .EXTRN  DSR$CheckLoad                                                   ;[25
                    0000    273         .EXTRN  MP$GR_BOOTED_PROCESSORS                                         ;[32
                    0000    274         .EXTRN  PR$_SID_TYPRTUV2                                                ;[36
                    0000    275         .EXTRN  DS$GB_SYSTYPE                                                   ;[37
                    0000    276         .EXTRN  PR$_SYS_TYP900                                                  ;[37
                    0000    277                                                                             ;G:4
```

```
                     0000   279          .Subtitle       Own Storage
                 00000000   280          .PSect          Work, NoShr, NoExe, Wrt, Long                          ;[26
                     0000   281
                     0000   282 ;
                     0000   283 ; OWN STORAGE:
                     0000   284 ;
                     0000   285
            00000004 0000   286 L_POLEN:          .BLKL   1       ; P0 (PHYSICAL MEMORY) LENGTH.
            00000008 0004   287 A_P1BUFPTE:       .BLKL   1       ; P0 ADDRESS OF P1 BUFFER PTE'S.
            0000000C 0008   288 A_P1BUFVA:        .BLKL   1       ; P1 BUFFER VIRTUAL ADDRESS.
            00000010 000C   289 A_SPTEND:         .BLKL   1       ; SYS PAGE TABLE END.
            00000014 0010   290 DS$GL_MEMSIZE:: .BLKL   1       ; BYTES OF PHYSICAL MEMORY
                     0014   291
                     0014   292 DS$GL_Actual_MemSize::          ; The actual size of memory                    [26]
            00000018 0014   293                     .BlkL   1   ;                                              [26]
                     0018   294
                     0018   295 DS$GL_Set_MemSize::             ; The 'SET' memory size                        [26]
            00000000 0018   296                     .Long   0   ; Initialized to zero                          [26]
                     001C   297
            0000001D 001C   298 DS$GB_MM_ENB::  .BLKB   1       ; 0 if MM OFF, 1 if ON
                     001D   299
                     001D   300 ;
                     001D   301 ; Symbols used to give initial values to X6 thru XD.
                     001D   302 ;
                     001D   303
            00000000 001D   304 PFN$AW_SWPVBN    == L_POLEN             ; X6                                   [21]
            00000004 001D   305 PFN$AL_PTE       == A_P1BUFPTE          ; X7
            00000008 001D   306 PFN$AL_BAK       == A_P1BUFVA           ; X8
            0000000C 001D   307 PFN$AW_REFCNT    == A_SPTEND            ; X9
            00000000 001D   308 PFN$AW_FLINK     == 0                   ; XA                                   [21]
            00000000 001D   309 PFN$AB_BLINK     == 0                   ; XB                                   [21]
            00000010 001D   310 PFN$AB_STATE     == DS$GL_MEMSIZE       ; XC Bytes physical memory
            00000000 001D   311 PFN$AB_TYPE      == 0                   ; XD                                   [21]
            00000000 001D   312 XDS$INIT         == 0                   ; Address of command string NOP
                     001D   313
                     001D   314 ;
                     001D   315 ; LOCAL SYMBOLS
                     001D   316 ;
                     001D   317
            00000014 001D   318 EXPREG$_PTE_OWNER_CODE = ^X14    ; Offset to extract PTE ownership code [31]
            00000014 001D   319 CNTREG$_VA             = ^X14    ; Offset to extract starting VA        [31]
            20040010 001D   320 PPA_CONSRV4            = ^X20040010 ; Highest Sys Memory Address      [37]   ;G:4
                     001D   321                                                                                 ;G:4
                     001D   322                                                                                 ;G:4
```

```
                                    001D   324          .Subtitle       Data Storage
                                00000000   325          .PSect          Data, Shr, NoExe, NoWrt, Long
                                    0000   326
                                    0000   327          MODNAM  MEMMGT
                  54 47 4D 4D 45 4D 00' 0000       $MODULE:        .ASCIC  \MEMMGT\
                                    06  0000
                                    0007   328
                                    0007   329 T_INSFMEM:
4C 55 21 20 79 6C 6E 4F 20 3F 3F 00' 0007   330          .ASCIC  "?? Only !UL pages memory present, 1024 are required.!/"        ;[29
72 6F 6D 65 6D 20 73 65 67 61 70 20  0013
31 20 2C 74 6E 65 73 65 72 70 20 79  001F
75 71 65 72 20 65 72 61 20 34 32 30  002B
                  2F 21 2E 64 65 72 69  0037
                                    36  0007
                                    003E   331
```

```
                     003E   333          .SBTTL  MapMem Routine - Map all of memory
                  00000000  334          .PSect  Code, Shr, Exe, NoWrt, Long
                     0000   335
                     0000   336 ;++
                     0000   337 ; FUNCTIONAL DESCRIPTION:
                     0000   338 ;
                     0000   339 ;       MAPS AVAILABLE MEMORY AND SETS UP MEMORY MANAGEMENT PAGE TABLES.
                     0000   340 ;
                     0000   341 ; Note:
                     0000   342 ;       ESKAZ (memory management testing) expects the length register
                     0000   343 ;       for each addressing space to exclude 1 valid PTE. As a result
                     0000   344 ;       there is a more valid PTE just beyond the length register.
                     0000   345 ;
                     0000   346 ; Note:
                     0000   347 ;       This allows a value to be inserted in the longword DS$GL_Set_MemSize
                     0000   348 ;       that will be used as the memory size, rather than the actual memory
                     0000   349 ;       size. This way, the operator can, for example, set up the memory for
                     0000   350 ;       a 256K machine in a machine that actually has 2M (or whatever). Note
                     0000   351 ;       that this user-set memory size must fulfill the following two
                     0000   352 ;       conditions:
                     0000   353 ;               1. Must be greater than 0. 0 is the default.
                     0000   354 ;               2. Must be less than or equal to the actual memory size.
                     0000   355 ;
                     0000   356 ; CALLING SEQUENCE:
                     0000   357 ;
                     0000   358 ;       BSBW    MAPMEM
                     0000   359 ;
                     0000   360 ; INPUT PARAMETERS:     NONE
                     0000   361 ;
                     0000   362 ; IMPLICIT INPUTS:      NONE
                     0000   363 ;
                     0000   364 ; OUTPUT PARAMETERS:    NONE
                     0000   365 ;
                     0000   366 ; IMPLICIT OUTPUTS:     NONE
                     0000   367 ;
                     0000   368 ; COMPLETION CODES:     NONE
                     0000   369 ;
                     0000   370 ; SIDE EFFECTS:
                     0000   371 ;
                     0000   372 ;       Plenty!
                     0000   373 ;--
```

```
                                0000   375 MAPMEM::
                                0000   376
                                0000   377 ;+
                                0000   378 ; Determine how much memory is present and create page table for it.
                                0000   379 ; The last page/PTE is outside the length register and reserved for
                                0000   380 ; ESKAZ.
                                0000   381 ;-
                     007C 8F    BB  0000   382         PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save all the registers that are used  [26]
                          53    D4  0004   383         CLRL    R3                      ; Start with 0 MEG                      [38]
                          55    D4  0006   384         CLRL    R5                      ; Start with 0 index                    [38]
          00'8F   00000000'EF   91  0008   385         CMPB    DS$GB_SYSTYPE, -        ; Is this an Andromeda?                 [37]
                                0010   386                   #PR$_SYS_TYP900         ;                                       [37]
                          31    12  0010   387         BNEQ    10$                     ; NO, then branch                       [37]
               55   00000000'EF   DE  0012   388         MOVAL   P0PT_BASE,R5            ; Base page table                       ;G:6
               56   20040010 9F   D0  0019   389         MOVL    a#PPA_CONSRV4,R6        ; ELSE, copy Highest Sys Memory Address [37]
                54    56   F7 8F   78  0020   390         ASHL    #-9,R6,R4               ; COMPUTE # OF PAGES IN THE SYSTEM      [37]
     00000018'EF   00000400 8F   D0  0025   391         MOVL    #1024,DS$GL_Set_MemSize ; Set memory size to be maped at 512K   [37]
                                0030   392 5$:                                                                                 ;G:6
                6543   6543   C2  0030   393         SUBL2   (R5)[R3],(R5)[R3]       ; CLEAR PTE                             [37]
              6543   A1800000 E3  9E  0035   394         MOVAB   PTE$M_VALID ! PTE$M_OWN ! PTE$C_UW(R3), -                       ;G:6
                                003D   395                 (R5)[R3]                ; Make page valid                       [37]
                 EF 53    54   F2  003D   396         AOBLSS  R4,R3,5$                ; Repeat for the total # of pages       [37]
                          57    11  0041   397         BRB     20$                     ; and continue                          [37]
                                0043   398 10$:                                                                               ;G:6
                   7E    53    14  78  0043   399         ASHL    #20,R3,-(SP)            ; Get 1st MEG adress                    [38]
              000002E6'EF    01  FB  0047   400         CALLS   #1,L^ACCESSABLE         ; Is it accessable?                     [38]
                       08 50    E9  004E   401         BLBC    R0,12$                  ; If not, branch                        [38]
          EA 53   00000FFF 8F   F2  0051   402         AOBLSS  #^XFFF,R3,10$           ; Repeat for all of memory              [38]
                                0059   403 12$:                                                                               ;G:6
                          53    D7  0059   404         DECL    R3                      ; Get last good MEG                     [38]
                54    53    14  78  005B   405         ASHL    #20,R3,R4               ; Get starting address                  [38]
                                005F   406                                                                                    ;G:6
                                005F   407         ;+                                                                          ;G:6
                                005F   408         ; Now determine size of remaining memory after last full MEG            ;G:6
                                005F   409         ; R3 = number of full MEGs                                              ;G:6
                                005F   410         ;-                                                                          ;G:6
                                005F   411                                                                                    ;G:6
                                005F   412 14$:                                                                                ;G:6
                56    55    09  78  005F   413         ASHL    #9,R5,R6                ; Next page address                     [38]
                7E    56    54  C1  0063   414         ADDL3   R4,R6,-(SP)             ; Finish building address               [38]
              000002E6'EF    01  FB  0067   415         CALLS   #1,L^ACCESSABLE         ; Is it accessable?                     [38]
                       08 50    E9  006E   416         BLBC    R0,16$                  ; If not, branch                        [38]
          E6 55   00000800 8F   F2  0071   417         AOBLSS  #^X800,R5,14$           ; Repeat for remaining memory           [38]
                                0079   418 16$:                                                                               ;G:6
                53    53    0B  78  0079   419         ASHL    #11,R3,R3               ; How many pages in remaining memory?   [38]
                          53    55  C0  007D   420         ADDL2   R5,R3                   ; Total number of pages?                [38]
               55   00000000'EF   DE  0080   421         MOVAL   P0PT_BASE,R5            ; Base page table                       [38]
                          54    D4  0087   422         CLRL    R4                      ; Start with PTE 0                      [38]
                                0089   423 18$:                                                                               ;G:6
                6544   6544   C2  0089   424         SUBL2   (R5)[R4],(R5)[R4]       ; Clear PTE                             [38]
              6544   A1800000 E4  9E  008E   425         MOVAB   PTE$M_VALID ! PTE$M_OWN ! PTE$C_UW(R4), -                       ;G:6
                                0096   426                 (R5)[R4]                ; Make page valid                       [38]
                 EF 54    53   F3  0096   427         AOBLEQ  R3,R4,18$               ; Do for all pages                      [38]
                                009A   428
                                009A   429         ;+
                                009A   430         ; R3 now equals the number of pages in memory (i.e., number of K Bytes  [26]
                                009A   431         ; equal to R3 / 2).                                                     [26]
```

```
                                009A   432          ;-
                                009A   433
00000014'EF   53   09   78       009A   434 20$:     AshL    #9, R3, -                       ; Use the value in R3 (^9) as the       [26]
                                00A2   435                    DS$GL_Actual_MemSize        ; ... actual memory size                [26]
     56   00000018'EF   D0       00A2   436          MovL    DS$GL_Set_MemSize, R6           ; Get the user-set memory size (pages)  [26]
                   08   13       00A9   437          Beql    22$                             ; If = 0, then branch                   [26]
                53   56   D1       00AB   438          CmpL    R6, R3                          ; Compare user-set and actual           [26]
                   03   14       00AE   439          Bgtr    22$                             ; If user-set > actual, use actual      [26]
                                00B0   440
                                00B0   441          ;+
                                00B0   442          ; Now the user-set memory size (in R6) satisfies the following:         [26]
                                00B0   443          ;       0 < R6 <= R3                                                    [26]
                                00B0   444          ; Remember that both of these quanities are number of pages.            [26]
                                00B0   445          ;-
                                00B0   446
                53   56   D0       00B0   447          MovL    R6, R3                          ; Use the user-set - not the actual     [26]
                                00B3   448
00000010'EF   53   09   78       00B3   449 22$:     ASHL    #9,R3,DS$GL_MEMSIZE             ; Set memory size                       [26]
     00000400 8F   53   D1       00BB   450          CMPL    R3,#1024                        ; 512K memory present?                  [26]
                   15   18       00C2   451          BGEQ    25$                             ; Branch if enough memory
                                00C4   452
                                00C4   453          $Print  -                               ; Print an error message                [26]
                                00C4   454                    #DS$K_Type_General_Error, -; ... a general error                [26]
                                00C4   455                    #DS$K_PrintF, -             ; ... use PrintF                        [26]
                                00C4   456                    T_InsfMem, -                ; ... the error message                 [26]
                                00C4   457                    R3                          ; ... the number of pages               [26]
                   53   DD       00C4                  PUSHL   R3
           00000007'EF   9F       00C6                  PUSHAB  T_InsfMem
                      7E   01   9A   00CC                  movzbl  #DS$K_PrintF, -(sp)
                      7E   03   98   00CF                  cvtbl   #DS$K_Type_General_Error, -(sp)
      00000000'GF   04   FB       00D2                  CALLS   #$$N+2, G^DSX$PRINT
                                00D9   458
      65   78000000 8F   CA       00D9   459 25$:     BICL    #PTE$M_PROT,(R5)                ; No access to page zero
           50   53   01   C3       00E0   460          SUBL3   #1,R3,R0                        ; Reserve last 1 PTE'S for ESKAZ
      00000000'EF   50   D0       00E4   461          MOVL    R0,L_POLEN                      ; Length in longwords of P0PT
                09   50   DA       00EB   462          MTPR    R0, #PR$_P0LR                   ; SET UP P0 LENGTH REGISTER.
                                00EE   463
            53   7F 8F   93       00EE   464 30$:     BITB    #^X7F,R3                        ; Offset page aligned
                   0C   13       00F2   465          BEQL    40$                             ; Branch if so
                 6543   63   9E   00F4   466          MOVAB   PTE$C_NA(R3),(R5)[R3]           ; Make invalid page
EE 53   000FFFFF 8F   F2       00F8   467          AOBLSS  #^XFFFFF,R3,30$                 ; Repeat until aligned
                                0100   468
            55   6543   DE       0100   469 40$:     MOVAL   (R5)[R3],R5                     ; Point to SPT base
```

```
                          0104   471
                          0104   472 ;+
                          0104   473 ; Setup system page tables
                          0104   474 ; Map the normal system service vectors to their supervisor counterparts
                          0104   475 ; but do not allow access to catch programming errors.
                          0104   476 ; Map pages from 80000800 to 80010000 N/A and invalid, available for expreg
                          0104   477 ; in system space.
                          0104   478 ; Map pages from 80010000 to 80000000+end of P0 Page table user read, kernel
                          0104   479 ; writeable. Makes system space map to physical memory.
                          0104   480 ; Map each page allocated to the P0 page tables and set the P0 Base and
                          0104   481 ; Length registers.
                          0104   482 ; Allocate 16 System PTEs for EXPREG to use in allocating SYSTEM space.
                          0104   483 ; Allocate 32 PTEs to map P1 page tables and set the P1 base and length
                          0104   484 ; registers. Also allocate 1 PTE for the space allocated by EXPREG in P1 space.
                          0104   485 ; And reserve two PTEs, one inside the length register and one outside for
                          0104   486 ; ESKAZ.
                          0104   487 ;-
                          0104   488
                          0104   489
                          0104   490 50$:
            0C    55  DA  0104   491          MTPR    R5, #PR$_SBR                ; SET SYSTEM BASE REGISTER.
 00000000'EF    55  D0  0107   492          MOVL    R5, MMG$GL_SPTBASE          ; Save pointer to system page table
                53  D4  010E   493          CLRL    R3                          ; Start with PFN 0 in system space
          51  00'8F  9A  0110   494          MOVZBL  #DS$AQ_SYSSRV a-9, R1       ; SYSTEM SERVICE PAGE NUMBER.
                      0114   495
      6543  81800000 E1  9E  0114   496 60$:   MOVAB   PTE$M_VALID ! PTE$C_NA ! PTE$M_OWN(R1), -
                      011C   497                 (R5)[R3]                    ; Set no access to  virtural 80000000
                51  D6  011C   498          INCL    R1                          ; Next page
             F2 53    04  F2  011E   499          AOBLSS  #4,R3,60$                   ; Fill 4 pages
                      0122   500
                6543  D4  0122   501 65$:   CLRL    (R5)[R3]                    ; Clear next PTE
       F5 53  00000080 8F  F2  0125   502          AOBLSS  #128,R3,65$                 ; Fill rest of page with avail SPTE's
                      012D   503
   51  00000010'EF  F7 8F  78  012D   504          ASHL    #-9,DS$GL_MEMSIZE,R1        ; Get PFN of last page + 1
       6543  F1800000 E3  9E  0136   505 70$:   MOVAB   PTE$M_VALID ! PTE$C_URKW ! PTE$M_OWN (R3), -
                      013E   506                 (R5)[R3]                    ; Map 1 for 1 valid and user readable
             F4 53    51  F2  013E   507          AOBLSS  R1,R3,70$                   ; Map up to end of P0 PT
          08  80000000'8F  DA  0142   508          MTPR    #^X80000000 ! P0PT_BASE, -
                          0149   509                  #PR$_P0BR                  ; Set P0 Base register
```

```
                                0149   511
                                0149   512 ;+
                                0149   513 ; Setup PTE's to map P1 page tables
                                0149   514 ;-
                                0149   515
          50   53   09   78   0149   516          ASHL    #9,R3,R0                ; SVA for this PFN
       50   80000000 8F   C8   014D   517          BISL    #^X80000000,R0          ; Make SVA of first P1 PTE
                                0154   518
54  00000000'EF   15   09   EF   0154   519          EXTZV   #9,#21,IO$GQ_PHYSICAL,R4; Get PFN of lowest I/O space
                                015D   520
       54   0000007F 8F   C2   015D   521          SUBL    #127,R4                 ; PFN of scratch area
            0B   54   DA   0164   522          MTPR    R4,#PR$_P1LR            ; Set length register
                 54   D7   0167   523          DECL    R4                      ; Base is really 128 pages away
                                0169   524
          51   54   04   C5   0169   525          MULL3   #4,R4,R1                ; Make byte offset
               50   51   C2   016D   526          SUBL    R1,R0                   ; byte address of P1 base
               0A   50   DA   0170   527          MTPR    R0, #PR$_P1BR           ; Set P1 base register
                                0173   528
       52   01800000 8F   D0   0173   529          MOVL    #PTE$M_OWN,R2           ; Initial PTE protection
                                017A   530
             6543   52   D0   017A   531          MOVL    R2,(R5)[R3]             ; Free SPTE for P1 scratch
                    53   D6   017E   532          INCL    R3                      ; Update free PTE pointer
                                0180   533
       50   00000000'EF   7D   0180   534          MOVQ    IO$GQ_PHYSICAL,R0       ; Get I/O space address range
               51   50   C2   0187   535          SUBL    R0,R1                   ; Length of I/O space
    50   51   0E   10   EF   018A   536          EXTZV   #16,#14,R1,R0           ; Get SPTE's required
         50   02 A340   9E   018F   537          MOVAB   2(R3)[R0],R0            ; Final PTE * 2 for ESKAZ
                                0194   538
             6543   52   D0   0194   539 80$:     MOVL    R2,(R5)[R3]             ; Set PTE protection
          F8 53   50   F2   0198   540          AOBLSS  R0,R3,80$               ; For each page of P1 page tables
                                019C   541
 0000000C'EF    6543   DE   019C   542          MOVAL   (R5)[R3], A_SPTEND      ; SAVE END OF TABLE ADDRESS.
       50   53   01   C3   01A4   543          SUBL3   #1,R3,R0                ; Decrease for base register
            0D   50   DA   01A8   544          MTPR    R0, #PR$_SLR            ; SET SYSTEM LENGTH REGISTER
                                01AB   545
          53   7F 8F   93   01AB   546 90$:     BITB    #^X7F,R3                ; End of page?
                    0B   13   01AF   547          BEQL    100$                    ; Branch if so
                  6543   D4   01B1   548          CLRL    (R5)[R3]                ; Invalid PTE and no access
EF 53    000FFFFF 8F   F3   01B4   549          AOBLEQ  #^XFFFFF,R3,90$         ; Increment and branch
                                01BC   550
          55   6543   DE   01BC   551 100$:     MOVAL   (R5)[R3],R5             ; First free
00'8F    0000FE17'EF   91   01C0   552          CMPB    DSA$GL_SID+3,#PR$_SID_TYPRTUV2 ; Is it RT-VAX                  [39]
                 37   12   01C8   553          BNEQ    120$                    ; set for RT-VAX                       [39]
               50   0B   DB   01CA   554          MFPR    #PR$_P1LR,R0            ; Get length of P1                     [39]
                    50   D7   01CD   555          DECL    R0                      ; BASE IS REALLY 128 PAGES AWAY        [39]
       50   50   04   C5   01CF   556          MULL3   #4,R0,R0                ; Make byte offset                     [39]
       50   55   50   C3   01D3   557          SUBL3   R0,R5,R0                ; Compute P1 physical base value       [39]
            0A   50   DA   01D7   558          MTPR    R0,#PR$_P1BR            ; Set P1 base register.                [39]
    08   00000000'8F   DA   01DA   559          MTPR    #P0PT_BASE,#PR$_P0BR    ; Set P0 Base register                 [39]
    50   00000000'EF   7D   01E1   560          MOVQ    IO$GQ_PHYSICAL,R0       ; Get I/O space address range          [39]
       50   51   50   C3   01E8   561          SUBL3   R0,R1,R0                ; Length of I/O space                  [39]
    50   00000200 8F   C6   01EC   562          DIVL2   #512,R0                 ; Get # of PTE's to set up             [39]
            51   54   D0   01F3   563          MOVL    R4,R1                   ; P1 address                           [39]
            52   55   D0   01F6   564          MOVL    R5,R2                   ; Address of P1 PTE table              [39]
            82   51   D0   01F9   565 110$:     MOVL    R1,(R2)+                ; PFN of P1                            [39]
                 51   D6   01FC   566          INCL    R1                      ; Next PFN                             [39]
              F8 50   F4   01FE   567          SOBGEQ  R0,110$                 ; Set up all of P1 space               [39]
```

```
                                  0201   568 120$:                                                                  ;G:7
                 51   54   09   78  0201   569          ASHL    #9,R4,R1                  ; P1 address
00000008'EF   51   40000400 8F   C9  0205   570          BISL3   #^X40000400,R1,A_P1BUFVA; Virtural address of P1 scratch area
                                  0211   571
                      51   54   D0  0211   572          MOVL    R4,R1                     ; P1 PFN to map for scratch area
              52   01800000 8F   D0  0214   573          MOVL    #PTE$M_OWN ! PTE$C_NA,R2; No access to these pages
                    50   7E 8F   9A  021B   574          MOVZBL  #K_P1PAGES,R0             ; Map K_P1PAGES pages
                                  021F   575
                         39   10  021F   576          BSBB    MAPMEM$IOSPACE            ; setup that page and the other 127
              51   00000008'EF   D0  0221   577          MOVL    A_P1BUFVA,R1              ; Get P1 virtural address
                   00000804'EF   16  0228   578          Jsb     RPTEADR                   ; Get address of PTE                [27]
              00000004'EF   51   D0  022E   579          MOVL    R1,A_P1BUFPTE             ; Address of first P1 PTE
                   00000000'EF   16  0235   580          Jsb     MAP$IOSPACE               ; Map I/O space                     [27]
              00000000'EF   55   D0  023B   581          MOVL    R5,DS$GA_LASTADR          ; Record last available
                                  0242   582
                   00000000'EF   16  0242   583          JSB     MEMPOOL$INIT              ; Initialize memory pool
                                  0248   584
                   00000321'EF   16  0248   585          Jsb     STACKPROT                 ; Set protections on stack pages    [27]
              00000392'EF   00   FB  024E   586          CALLS   #0,MAPFREE                ; Free memory
                      007C 8F   BA  0255   587          POPR    #^M<R2,R3,R4,R5,R6>       ; Restore the saved registers       [26]
                                05  0259   588          RSB
```

```
                        025A    590          .SBTTL  MAPMEM$IOSPACE  Setup page tables for I/O space
                        025A    591 ;++
                        025A    592 ; FUNCTIONAL DESCRIPTION:
                        025A    593 ;
                        025A    594 ;       This routine is called by MAP$IOSPACE to setup a PTE
                        025A    595 ;       for an I/O space address. If necessary a page is allocated for the
                        025A    596 ;       corresponding SPTE and filled with invalid no access PTE's.
                        025A    597 ;       The particular address desired is then given the specified
                        025A    598 ;       protection.
                        025A    599 ;
                        025A    600 ; CALLING SEQUENCE:
                        025A    601 ;
                        025A    602 ;       BSBW    MAPMEM$IOSPACE
                        025A    603 ;
                        025A    604 ; INPUT PARAMETERS:     NONE
                        025A    605 ;
                        025A    606 ; IMPLICIT INPUTS:
                        025A    607 ;
                        025A    608 ;       R0      Count of pages to setup
                        025A    609 ;       R1      P1 PFN to map
                        025A    610 ;       R2      PTE format protection and valid bits (pte format)
                        025A    611 ;       R5      Address of free space pointer
                        025A    612 ;
                        025A    613 ; OUTPUT PARAMETERS:    NONE
                        025A    614 ;
                        025A    615 ; IMPLICIT OUTPUTS:
                        025A    616 ;
                        025A    617 ;       R5      Possibly update free page pointer
                        025A    618 ;--
```

```
                                  025A   MAPMEM$IOSPACE::
00'8F   0000FE17'EF   91   025A          CMPB    DSA$GL_SID+3,#PR$_SID_TYPRTUV2 ; Is it RT-VAX                [39]
                  58   13   0262          BEQL    RTVAX                          ; Skip if RT-VAX                  [39]
                  18   BB   0264          PUSHR   #^M<R3,R4>                     ; Save
            53    0A   DB   0266          MFPR    #PR$_P1BR,R3                   ; P1 base register
         53    6341   DE   0269          MOVAL   (R3)[R1],R3                    ; VA of P1 PTE
   54   53    15   09   EF   026D          EXTZV   #9,#21,R3,R4                   ; PFN of SVA
            7E    0C   DB   0272          MFPR    #PR$_SBR,-(SP)                 ; Get system base register
         54    9E44   DE   0275          MOVAL   @(SP)+[R4],R4                  ; Physical address of SPTE
        23 64    1F   E0   0279          BBS     #31,(R4),20$                   ; Branch if already valid
                            027D
                  03   BB   027D          PUSHR   #^M<R0,R1>                     ; Save 2
   50   55    F7 8F   78   027F          ASHL    #-9,R5,R0                      ; PFN of free page
   64   F1800000 E0   9E   0284          MOVAB   PTE$M_VALID ! PTE$C_URKW ! PTE$M_OWN(R0),(R4)
        51    7F 8F   8A   028B          BICB    #^X7F,R1                       ; Lowest PFN this page of PTE's
        50    7F A1   9E   028F          MOVAB   127(R1),R0                     ; Final PFN
   85   01800000 E1   9E   0293   10$:    MOVAB   PTE$M_OWN ! PTE$C_NA(R1), -
                            029A                  (R5)+                          ; Fill invalid no access PTE's
        F5 51    50   F3   029A          AOBLEQ  R0,R1,10$                      ; Count and continue
                  03   BA   029E          POPR    #^M<R0,R1>                     ; Restore 2
                            02A0
   7E   64    15   00   EF   02A0   20$:    EXTZV   #0,#21,(R4),-(SP)              ; Get Physical PFN of P1 PTE
   53   17    09   8E   F0   02A5          INSV    (SP)+,#9,#23,R3                ; Now have Physical addr of PTE
   63   F8000000 8F   CA   02AA          BICL    #PTE$M_PROT!PTE$M_VALID,(R3) ; Clear protection
            63    52   C8   02B1          BISL    R2,(R3)                        ; Set protection
                            02B4
                  18   BA   02B4          POPR    #^M<R3,R4>                     ; Restore
                  51   D6   02B6          INCL    R1                             ; Update PFN
            9F    50   F5   02B8          SOBGTR  R0,MAPMEM$IOSPACE              ; Repeat until done all pages
                  05   02BB          RSB
                            02BC                                                                                ;G:7
                  08   BB   02BC   RTVAX:  PUSHR   #^M<R3>                        ; Save                            [39]
            53    0A   DB   02BE   10$:    MFPR    #PR$_P1BR,R3                   ; P1 base register                [39]
         53    6341   DE   02C1          MOVAL   (R3)[R1],R3                    ; Physical address of P1 PTE      [39]
            55    53   D1   02C5          CMPL    R3,R5                          ; Check if allocated yet          [39]
                  0A   19   02C8          BLSS    20$                            ; Skip if allocated               [39]
            55    53   D0   02CA          MOVL    R3,R5                          ; Set pointer to cover            [39]
   85   F1800000 E1   9E   02CD          MOVAB   PTE$M_VALID ! PTE$C_URKW ! PTE$M_OWN (R1), - ;                [39]
                            02D4                  (R5)+                          ;                                 ;G:7
   63   F8000000 8F   CA   02D4   20$:    BICL    #PTE$M_PROT!PTE$M_VALID,(R3) ; Clear protection                [39]
            63    52   C8   02DB          BISL2   R2,(R3)                        ; Set protection                  [39]
                            02DE                                                                                ;G:7
                  51   D6   02DE          INCL    R1                             ; Update PFN                      [39]
            DB    50   F5   02E0          SOBGTR  R0,10$                         ; Repeat until done all pages     [39]
                  08   BA   02E3          POPR    #^M<R3>                        ; Restore                         [39]
                  05   02E5          RSB                                                                          ;G:7
```

```
                                02E6    667          .SBTTL  ACCESSABLE      check a page for accessability
                                02E6    668 ;++
                                02E6    669 ; FUNCTIONAL DESCRIPTION:
                                02E6    670 ;
                                02E6    671 ;       This routine is used to determine if every word on a page
                                02E6    672 ;       can be read and written. specificially by writing each
                                02E6    673 ;       QUADWORD to itself.
                                02E6    674 ;
                                02E6    675 ; CALLING SEQUENCE:
                                02E6    676 ;
                                02E6    677 ;       ACCESSABLE(address)
                                02E6    678 ;
                                02E6    679 ; INPUT PARAMETERS:     NONE
                                02E6    680 ;
                                02E6    681 ;       address Page address to check bits 0-8 are ignored
                                02E6    682 ;
                                02E6    683 ; IMPLICIT INPUTS:      NONE
                                02E6    684 ;
                                02E6    685 ; OUTPUT PARAMETERS:    NONE
                                02E6    686 ;
                                02E6    687 ; IMPLICIT OUTPUTS:     NONE
                                02E6    688 ;
                                02E6    689 ; COMPLETION CODES:
                                02E6    690 ;
                                02E6    691 ;       SS$_NORMAL      if page ok
                                02E6    692 ;       SS$_MCHK        if machine check occured during processing
                                02E6    693 ;
                                02E6    694 ;--
                                02E6    695
                                02E6    696 ACCESSABLE:
                          0000  02E6    697          .WORD   ^M<>                    ; Save no registers
                6D    FC'AF     9E  02E8    698          MOVAB   B^30$,(FP)              ; Set condition handler
51    04 AC   000001FF  8F      CB  02EC    699          BICL3   #^X1FF,4(AP),R1         ; Address to start
                                02F5    700
                  61    61      7D  02F5    701          MOVQ    (R1),(R1)               ; Copy quadword to self                 [28]
                  50    01      D0  02F8    702          MOVL    S^#SS$_NORMAL,R0        ; Success
                                04  02FB    703          RET
                                02FC    704
                          0000  02FC    705 30$:     .WORD   ^M<>                    ; Save no registers
                50    04 AC     7D  02FE    706          MOVQ    4(AP),R0                ; Signal and mechanism pointers
      04 A0   00660088  8F      D1  0302    707          CMPL    #DS$_MCHK,4(R0)         ; Machine check?
                        0F      12  030A    708          BNEQ    40$                     ; Branch if not
          0C A1   04    A0      D0  030C    709          MOVL    4(R0),12(R1)            ; Change routine R0
                        7E      7C  0311    710          CLRQ    -(SP)                   ; No args to unwind
     00000000'EF        02      FB  0313    711          CALLS   #2,SYS$UNWIND           ; Unwind stack
                                04  031A    712          RET                             ; Return
                                031B    713
             50   0918  8F      3C  031B    714 40$:     MOVZWL  #SS$_RESIGNAL,R0        ; Let someone else have it
                                04  0320    715          RET                             ; Return
```

```
                                0321          .SBTTL  STACKPROT
                                0321  ;++
                                0321  ; FUNCTIONAL DESCRIPTION:
                                0321  ;
                                0321  ;       This routine is used to set the protections on stack pages so
                                0321  ;       that underflows of each stack will cause access violations.
                                0321  ;       The user stack underflows into the SUPER stack, the SUPER stack
                                0321  ;       underflows into the EXEC stack, the EXEC stack underflows into
                                0321  ;       the INTERRUT stack (protected KW), the KERNEL stack underflows into
                                0321  ;       SCB_UNKINT which is read only.
                                0321  ;
                                0321  ; CALLING SEQUENCE:
                                0321  ;
                                0321  ;       BSBW    STACKPROT
                                0321  ;
                                0321  ; INPUT PARAMETERS:     NONE
                                0321  ;
                                0321  ; IMPLICIT INPUTS:
                                0321  ;
                                0321  ;       The page tables and stack addresses
                                0321  ;
                                0321  ; OUTPUT PARAMETERS:    NONE
                                0321  ;
                                0321  ; IMPLICIT OUTPUTS:     NONE
                                0321  ;
                                0321  ; COMPLETION CODES:     NONE
                                0321  ;
                                0321  ; SIDE EFFECTS:         NONE
                                0321  ;--
                                0321
                                0321  STACKPROT:
                                0321          DSBINT  #31                     ; Turn off interrupts                   [30]
                7E    12   DB   0321                  MFPR    S^#PR$_IPL,-(SP)
                12    1F   DA   0324                  MTPR    #31,S^#PR$_IPL
      51   00000000'EF   DE   0327          MOVAL   P0PT_BASE,R1            ; Base P0 page table
           50   0000'8F   3C   032E          MOVZWL  #STACK_BASE@-9,R0       ; PFN of read only section
                                0333
    6140   78000000 8F   CA   0333  10$:    BICL2   #PTE$M_PROT,(R1)[R0]    ; Clear protections in all pte's
   F0 50   00000000'8F   F2   033B          AOBLSS  #USTKPTR@-9,R0,10$      ; Branch if more to clear
                                0343
           50   0000'8F   3C   0343          MOVZWL  #STACK_BASE@-9,R0       ; PFN of read only section
                                0348
              6140   00   C8   0348          BISL    #PTE$C_NA, (R1)[R0]     ; Set protection to no access
                     50   D6   034C          INCL    R0                      ; 1 page protects bottom of stack
                                034E
    6140   70000000 8F   C8   034E  30$:    BISL    #PTE$C_URKW, (R1)[R0]   ; Set protection to KERNEL Write
   F0 50   00000000'8F   F2   0356          AOBLSS  #ISTKPTR@-9,R0,30$      ; Protect KW thru INTERRUPT stack
                                035E
    6140   68000000 8F   C8   035E  40$:    BISL    #PTE$C_UREW, (R1)[R0]   ; Set protection to EXEC write
   F0 50   00000000'8F   F2   0366          AOBLSS  #ESTKPTR@-9,R0,40$      ; Protect EW thru EXEC stack
                                036E
    6140   60000000 8F   C8   036E  50$:    BISL    #PTE$C_URSW, (R1)[R0]   ; Set protection to SUPER write
   F0 50   00000000'8F   F2   0376          AOBLSS  #SSTKPTR@-9,R0,50$      ; Protect SW thru SUPER stack
                                037E
    6140   20000000 8F   C8   037E  60$:    BISL    #PTE$C_UW, (R1)[R0]     ; Set protection to USER write
   F0 50   00000000'8F   F2   0386          AOBLSS  #USTKPTR@-9,R0,60$      ; Protect UW thru USER stack
                                038E
```

```
                    038E   772        ENBINT                          ; Allow interrupts again
        12   8E   DA  038E                   MTPR    (SP)+,S^#PR$_IPL
                  05  0391   773        RSB                             ; Return
```

```
0392   775          .SBTTL  MAP FREE MEMORY PROCEDURE.
0392   776 ;**
0392   777 ; FUNCTIONAL DESCRIPTION:
0392   778 ;
0392   779 ;       MAPS ALL FREE MEMORY BEFORE STARTING A PROGRAM.
0392   780 ;
0392   781 ; CALLING SEQUENCE:
0392   782 ;
0392   783 ;       PROCEDURE CALL.
0392   784 ;
0392   785 ; INPUT PARAMETERS:
0392   786 ;
0392   787 ;       NONE
0392   788 ;
0392   789 ; IMPLICIT INPUTS:
0392   790 ;
0392   791 ;       L$L_HPTL        - LENGTH OF EACH HARDWARE PARAMETER TABLE.
0392   792 ;       L$A_LASTADR     - LAST ADDRESS OF USER PROGRAM.
0392   793 ;       DSA$GL_UNITS    - NUMBER OF UNITS CURRENTLY SELECTED.
0392   794 ;       L_POLEN         - LENGTH OF P0 PAGE TABLE.
0392   795 ;
0392   796 ; OUTPUT PARAMETERS:
0392   797 ;
0392   798 ;       NONE
0392   799 ;
0392   800 ; IMPLICIT OUTPUTS:
0392   801 ;
0392   802 ;       NONE
0392   803 ;
0392   804 ; COMPLETION CODES:
0392   805 ;
0392   806 ;       NONE
0392   807 ;
0392   808 ; SIDE EFFECTS:
0392   809 ;
0392   810 ;       EFFECTIVE $DS_RELBUF_L OF ALL USER BUFFER SPACE.
0392   811 ;
0392   812 ;--
```

```
                          003C  0392   814 .ENTRY  MAPFREE,^M<R2,R3,R4,R5>          ; ENTRY POINT
                                0394   815
          54   00000000'EF    9E  0394   816         MOVAB   DS$K_PRGSIZ+DS$A_PRGBGN,R4 ; Assume no program loaded
               00000000'EF    16  039B   817         JSB     DSR$CheckLoad             ; Find out, check program hdr      [25]
                      11 50   E9  03A1   818         BLBC    R0, 5$                    ; Branch if no program loaded      [25]
                              03A4   819
                              03A4   820         IFNORD  #4,L$A_LASTADR,5$         ; Branch if can't read it
      00000214'EF   04   00   0C  03A4                   PROBER  #0,#4,L$A_LASTADR
                         07   13  03AC                   BEQL    5$
          54   00000214'EF    D0  03AE   821         MOVL    L$A_LASTADR, R4           ; Use size of program instead
                              03B5   822 5$:
                         53   D4  03B5   823         CLRL    R3                        ; CLEAR R3.
                    52   53   09  78   03B7   824 10$:    ASHL    #9,R3,R2                  ; MAKE VIRTURAL ADDRESS THIS PAGE
                         54   52  D1   03BB   825         CMPL    R2, R4                    ; CHECK FOR END OF USER CODE.
                         20   19  03BE   826         BLSS    30$
          0000FA00 8F    52   D1  03C0   827         CMPL    R2, #DSA$AT_APTTXT        ; Bottom of APT text area?
                         09   19  03C7   828         BLSS    20$
          00000000'EF    52   D1  03C9   829         CMPL    R2, L^DS$GA_LASTADR       ; Check for start of buffers
                         0E   19  03D0   830         BLSS    30$
  00000000'EF43   81800000 8F   CA  03D2   831 20$:    BICL2   #PTE$M_VALID + PTE$M_OWN, -       ; INVALIDATE PAGE.
                              03DE   832                   P0PT_BASE[R3]
                         0C   11  03DE   833         BRB     40$
  00000000'EF43   81800000 8F   C8  03E0   834 30$:    BISL2   #PTE$M_VALID + PTE$M_OWN, -       ; VALIDATE PAGE.
                              03EC   835                   P0PT_BASE[R3]
                              03EC   836
        C3 53   00000000'EF   F2  03EC   837 40$:    AOBLSS  L^L_POLEN, R3, 10$        ; CHECK FOR ALL DONE.              [27]
                              03F4   838
                              03F4   839 ;
                              03F4   840 ; INIT P1 SPACE BUFFER.
                              03F4   841 ;
                              03F4   842
           52   7E 8F   9A  03F4   843         MOVZBL  #K_P1PAGES,R2             ; P1 PTE's to clear
      53   00000004'EF   D0  03F8   844         MOVL    A_P1BUFPTE, R3                    ; PAGE TABLE ENTRY POINTER.
                              03FF   845
                         83   D4  03FF   846 50$:    CLRL    (R3)+                     ; FREE PAGE.
                      FB 52   F5  0401   847         SOBGTR  R2,50$
                              0404   848
                              0404   849 ;
                              0404   850 ; INIT SYSTEM SPACE BUFFER.
                              0404   851 ;
                              0404   852
      53   00000000'EF   D0  0404   853         MOVL    MMG$GL_SPTBASE,R3                 ; System page table pointer.
                              040B   854
    03   63   02   17   ED  040B   855 60$:    CMPZV   #PTE$V_OWN, #PTE$S_OWN, (R3), #3  ; SUPERVISOR PAGE?
                         02   13  0410   856         BEQL    70$
                         63   D4  0412   857         CLRL    (R3)                      ; FREE PAGE.
                              0414   858
FFED 53   04   0000000C'EF   F1  0414   859 70$:    ACBL    A_SPTEND, #4, R3, 60$             ; LOOP TO END.
      50   00000000'EF   7E  041E   860         MOVAQ   DS$GL_BUFCNT,R0           ; Address of buffer counts
                         80   7C  0425   861         CLRQ    (R0)+                     ; Reset P0, P1 buffer counts
                         80   7C  0427   862         CLRQ    (R0)+                     ; RESET SYS AND SUP BUFFER CNTS.
                              0429   863
                    FEF4 CF   16  0429   864         Jsb     STACKPROT                 ; Re-protect stacks                [27]
                         04   042D   865         RET
```

```
                        042E    867          .SBTTL  GET A BLOCK OF VIRTUAL MEMORY.
                        042E    868 ;++
                        042E    869 ; FUNCTIONAL DESCRIPTION:
                        042E    870 ;
                        042E    871 ;       THIS PROCEDURE EMULATES THE STARLET SYSTEM SERVICE "EXPREG",
                        042E    872 ;       EXPAND PROGRAM REGION.
                        042E    873 ;
                        042E    874 ;       IF THIS ROUTINE IS CALLED VIA $DS_GETSYSBUF, EACH PAGES PTE<21> WILL
                        042E    875 ;       BE SET TO A ONE TO INDICATE INTERNAL VDS OWNERSHIP. THIS SPECIAL
                        042E    876 ;       MARKING WILL PROHIBIT THE PAGES FROM BEING DEALLOCATED BY AN EXTERNAL
                        042E    877 ;       USERS DIAGNOSTIC INVOKING A $DS_RELBUF.
                        042E    878 ;
                        042E    879 ;       EACH PAGE THAT IS ALLOCATED HAS ITS PTE SET AS FOLLOWS:
                        042E    880 ;
                        042E    881 ;               PTE<31>, THE VALID BIT IS SET
                        042E    882 ;               PTE<30:27>, THE PROT CODE IS SET FOR USER WRITE (FULL ACCESS)
                        042E    883 ;               PTE<24:23>, STORES THE REGION CODE
                        042E    884 ;               PTE<21>, THIS BIT IS USED TO DENOTE PAGE OWNERSHIP (1=VDS
                        042E    885 ;                       INTERNAL, 0=EXTERNAL USER)
                        042E    886 ;               PTE<20:00> PAGE FRAME NUMBER
                        042E    887 ;
                        042E    888 ;
                        042E    889 ; CALLING SEQUENCE:
                        042E    890 ;
                        042E    891 ;       STARLET PROCEDURE CALL.
                        042E    892 ;
                        042E    893 ; INPUT PARAMETERS:
                        042E    894 ;
                        042E    895 ;       EXPREG$_PAGCNT(AP) =    NUMBER OF PAGES TO ADD TO THE END OF THE PROGRAM
                        042E    896 ;                               REGION.
                        042E    897 ;       EXPREG$_RETADR(AP) =    ADDRESS OF A 2-LONGWORD ARRAY IN WHICH TO RETURN
                        042E    898 ;                               THE VIRTUAL ADDRESSES OF THE STARTING PAGE AND
                        042E    899 ;                               ENDING PAGE OF THE EXPANTION AREA.
                        042E    900 ;       EXPREG$_ACMODE =        ACCESS MODE.  NOT USED.
                        042E    901 ;       EXPREG$_REGION(AP) =    REGION CODE:
                        042E    902 ;                                       0 = PROGRAM (P0) REGION.
                        042E    903 ;                                       1 = CONTROL (P1) REGION.
                        042E    904 ;                                       2 = SYSTEM REGION.
                        042E    905 ;                                       3 = USED BY DIAGNOSTIC SUPERVISOR AND
                        042E    906 ;                                           CRD
                        042E    907 ;
                        042E    908 ;       EXPREG$_PTE_OWNER_CODE(AP) = MASK TO WRITE INTO EACH PAGES PTE
                        042E    909 ;                                       DENOTING OWNER.                 [31]
                        042E    910 ;                                       000000 = CALLED VIA $DS_GETBUF
                        042E    911 ;                                               (EXTERNAL DIAGNOSTIC USER)
                        042E    912 ;                                       200000 = CALLED VIA $DS_GETSYSBUF
                        042E    913 ;                                               (INTERNAL VDS USER)     [31]
                        042E    914 ; IMPLICIT INPUTS:
                        042E    915 ;
                        042E    916 ;       NONE
                        042E    917 ;
                        042E    918 ; OUTPUT PARAMETERS:
                        042E    919 ;
                        042E    920 ;       @EXPREG$_RETADR(AP) =   2-LONGWORD ARRAY CONTAINING THE STARTING AND
                        042E    921 ;                               ENDING ADDRESSES OF THE EXPANDED REGION.
                        042E    922 ;
                        042E    923 ; IMPLICIT OUTPUTS:
```

```
                     042E    924 ;
                     042E    925 ;       NONE
                     042E    926 ;
                     042E    927 ; COMPLETION CODES:
                     042E    928 ;
                     042E    929 ;       SS$_NORMAL =    SERVICE SUCESSFULLY COMPLETED.
                     042E    930 ;       SS$_ILLPAGCNT = PAGE COUNT < 1.
                     042E    931 ;       SS$_VASFULL =   VIRTUAL ADDRESS SPACE FULL.
                     042E    932 ;
                     042E    933 ; SIDE EFFECTS:
                     042E    934 ;
                     042E    935 ;       NONE
                     042E    936 ;
                     042E    937 ; REGISTER USAGE:
                     042E    938 ;
                     042E    939 ;       R0 =    RETURN STATUS.
                     042E    940 ;       R1 =    BEGINNING P0 PTE POINTER.
                     042E    941 ;       R2 =    P0 PTE POINTER LIMIT.
                     042E    942 ;       R3 =    COUNT / 'REGION' PTE POINTER.
                     042E    943 ;       R4 =    COUNT / 'REGION' PTE LIMIT.
                     042E    944 ;       R5 =    VIRTUAL ADDRESS.
                     042E    945 ;       R6 =    PTE DATA.
                     042E    946 ;--
```

```
                        007C  042E   948 .Entry  Crd$ExpReg,^M<R2,R3,R4,R5,R6>          ; CRD entry point                [22]
                              0430   949
     52   00000000'EF   F7 8F   78   0430   950          AshL    #-9, L^DS$GA_LastAdr, R2          ; Calculate PFN of LastAdr       [22]
        54   00000000'EF   52   C3   0439   951          SubL3   R2, L^L_POLen, R4                 ; Get remaining length of table  [22]
           52   00000000'EF42   DE   0441   952          MovAL   POPT_Base [R2], R2                ; Get new starting address       [22]
                         10   11   0449   953          Brb     ExpReg_Common                     ; Skip to the common stuff       [22]
                              044B   954
                        007C  044B   955 .ENTRY  EXE$EXPREG,^M<R2,R3,R4,R5,R6>   ; ENTRY POINT
                              044D   956
           52   00000000'EF   DE   044D   957          MOVAL   POPT_BASE, R2                     ; START OF P0 PAGE TABLES.
           54   00000000'EF   D0   0454   958          MOVL    L^L_POLEN, R4                     ; P0 PAGE TABLE LENGTH.          [27]
                              045B   959
                              045B   960 ExpReg_Common:                                        ; Common point                   [22]
                              045B   961
                   51   52   D0   045B   962 10$:     MOVL    R2, R1                            ; SAVE PAGE NUMBER IN CASE FREE.
                53   04 AC   D0   045E   963          MOVL    EXPREG$_PAGCNT(AP), R3            ; GET PAGE LIMIT.
                         08   14   0462   964          BGTR    20$                               ; BR IF > 1.
                50   00FC 8F   3C   0464   965          MOVZWL  #SS$_ILLPAGCNT, R0                ; ILLEGAL PAGE COUNT.
                       00F9   31   0469   966          BRW     EXE$EXPREG_X
                      07 54   F4   046C   967 20$:     SOBGEQ  R4, 30$                           ; COUNT PAGES.                   ;G:
                50   0244 8F   3C   046F   968          MOVZWL  #SS$_VASFULL, R0                  ; OUT OF MEMORY.
                         0A   11   0474   969          BRB     40$
                         82   D5   0476   970 30$:     TSTL    (R2)+                             ; LOOK FOR EXISTS BUT NOT VALID.
                         E1   15   0478   971          BLEQ    10$                               ; BR ON NO GOOD.
                      EF 53   F5   047A   972          SOBGTR  R3, 20$                           ; LOOP TILL ENOUGH PAGES.
                   50   01   D0   047D   973          MOVL    #SS$_NORMAL, R0                   ; NORMAL RETURN.
     55   61   FFE00000 8F   CB   0480   974 40$:     BICL3   #^CPTE$M_PFN,(R1),R5              ; ISOLATE PFN       [31]
             55   55   09   78   0488   975          ASHL    #9,R5,R5                          ; GENERATE P0 VA    [31]
           00000000'EF   55   D0   048C   976          MOVL    R5,DS$GQ_PHYADR                   ; SAVE THE PHYSICAL ADDR.
                         7E   7C   0493   977          CLRQ    -(SP)                             ; SPACE FOR VIRTUAL ADRS.
            03   00   10 AC   CF   0495   978          CASEL   EXPREG$_REGION(AP), #0, #3        ; SET UP FOR REGION.
                        000D' 049A   979 50$:     .WORD   60$ - 50$                         ; P0            (0)
                        0012' 049C   980          .WORD   70$ - 50$                         ; P1            (1)
                        0027' 049E   981          .WORD   80$ - 50$                         ; SYS           (2)
                        000D' 04A0   982          .WORD   60$ - 50$                         ; SUPERVISOR. (3)
                              04A2   983
                   50   D4   04A2   984          CLRL    R0                                ; RETURN ERROR STATUS.
                00BE   31   04A4   985          BRW     EXE$EXPREG_X
                              04A7   986
                53   51   7D   04A7   987 60$:     MOVQ    R1, R3                            ; COPY P0 PAGE TABLE POINTERS.
                         3D   11   04AA   988          BRB     110$                              ; BR TO COMMON CODE.
                              04AC   989
     53   00000004'EF   D0   04AC   990 70$:     MOVL    A_P1BUFPTE, R3                    ; P1 PAGE TABLE POINTER.
        54   01F8 C3   DE   04B3   991          MOVAL   K_P1PAGES*4(R3), R4               ; End of P1 page table
     55   00000008'EF   D0   04B8   992          MOVL    A_P1BUFVA, R5                     ; P1 BUFFER VIRTUAL ADDRESS.
                         12   11   04BF   993          BRB     100$                              ; BR TO COMMON CODE.
                              04C1   994
     53   00000000'EF   D0   04C1   995 80$:     MOVL    MMG$GL_SPTBASE,R3                 ; System page table pointer
     54   0000000C'EF   D0   04C8   996          MOVL    A_SPTEND, R4                      ; END OF SYS PAGE TABLE.
        55   01   1F   78   04CF   997          ASHL    #31, #1, R5                       ; SYSTEM VIRTUAL ADDRESS.
                              04D3   998
                         63   D5   04D3   999 100$:    TSTL    (R3)                              ; LOOK FOR VALID BIT.
                         12   18   04D5  1000          BGEQ    110$                              ; BR IF NOT VALID.
     55   00000200 8F   C0   04D7  1001          ADDL    #512, R5                          ; NEXT VIRTUAL PAGE.
FFEF 53   04   54   F1   04DE  1002          ACBL    R4, #4, R3, 100$                  ; LOOP TO END OF PT.
             6E   55   D0   04E4  1003          MOVL    R5, (SP)                          ; SAVE FIRST ADR.
                         3A   11   04E7  1004          BRB     130$
```

```
                                  04E9
                 6E   55   D0     04E9  110$:   MOVL    R5, (SP)                          ; SAVE FIRST ADR.
            56   01   1F   78     04EC          ASHL    #PTE$V_VALID, #1, R6              ; GET VALID BIT.
     56   02   17   10 AC   F0     04F0          INSV    EXPREG$_REGION(AP), -             ; INSERT REGION CODE
                                  04F6                    #PTE$V_OWN, #PTE$S_OWN, R6      ;  INTO OWN FIELD.
        56   20000000 8F   C8     04F6          BISL2   #PTE$C_UW, R6                     ; User writeable
                                  04FD
             56   14 AC   C8     04FD          BISL    EXPREG$_PTE_OWNER_CODE(AP),R6     ; Set page owner code    [31]
        61   78000000 8F   CA     0501  120$:   BICL2   #PTE$M_PROT, (R1)                 ; Clear old protection
                 61   56   C8     0508          BISL2   R6, (R1)                          ; Set valid, prot,region, and
                                  050B                                                    ; owner code.            [31]
                 83   81   D0     050B          MOVL    (R1)+, (R3)+                      ; COPY TO 'REGION' PTE.
        55   00000200 8F   C0     050E          ADDL    #512, R5                          ; NEXT PHYSICAL PAGE.
                 52   51   D1     0515          CMPL    R1, R2                            ; CHECK FOR END OF BUFFER.
                      0E   18     0518          BGEQ    140$
                 54   53   D1     051A          CMPL    R3, R4                            ; CHECK FOR END OF REGION.
                      04   18     051D          BGEQ    130$                              ; ERROR EXIT.
                      63   D5     051F          TSTL    (R3)                              ; CHECK NEXT PAGE FOR FREE.
                      DE   18     0521          BGEQ    120$                              ; LOOP TIL ENOUGH.
                                  0523
            50   0244 8F   3C     0523  130$:   MOVZWL  #SS$_VASFULL, R0                  ; VIRTUAL ADDRESS SPACE FULL.
                                  0528
         04 AE   55   01   C3     0528  140$:   SUBL3   #1, R5, 4(SP)                     ; STORE END ADDRESS.
   00000004'EF   71   09   78     052D          ASHL    #9, -(R1), DS$GQ_PHYADR+4         ; SAVE END PHYSICAL ADR.
00000004'EF   40000000 8F   CA     0535          BICL    #^X40000000,DS$GQ_PHYADR+4        ; STRIP OFF SHIFTED PTE<21>
                                  0540                                                    ; OWNER BIT IF GETSYSBUF.[31]
   00000004'EF   01FF 8F   A8     0540          BISW2   #^X1FF, DS$GQ_PHYADR+4            ; SET TO LAST BYTE ON PAGE
                   08 AC   D5     0549          TSTL    EXPREG$_RETADR(AP)                ; CHECK IF RETURN ADR REQUESED.
                      17   13     054C          BEQL    EXE$EXPREG_X                      ; SKIP IF NOT.
         04 AE   01FF 8F   A8     054E          BISW2   #^X1FF, 4(SP)                     ; SET TO HIGHEST BYTE ADDR
             10 AC   01   D1     0554          CMPL    #1,EXPREG$_REGION(AP)             ; P1 space?
                      07   12     0558          BNEQ    150$                              ; Branch if not
                 51   8E   7D     055A          MOVQ    (SP)+,R1                          ; Get low, high
                      51   DD     055D          PUSHL   R1                                ; put low in first
                      52   DD     055F          PUSHL   R2                                ; put high in second
                                  0561
             08 BC   8E   7D     0561  150$:   MOVQ    (SP)+, @EXPREG$_RETADR(AP)        ; STORE VIRTUAL ADDRESSES.
                                  0565  EXE$EXPREG_X:
                           04     0565          RET                                       ; RETURN.
```

```
0566  1045          .SBTTL  RELEASE A BLOCK OF VIRTUAL MEMORY.
0566  1046 ;++
0566  1047 ; FUNCTIONAL DESCRIPTION:
0566  1048 ;
0566  1049 ;       THIS PROCEDURE EMULATES THE STARLET SYSTEM SERVICE "CNTREG",
0566  1050 ;       CONTRACT PROGRAM REGION.
0566  1051 ;
0566  1052 ;       IF CALLED VIA $DS_RELBUF, ANY PTE WITH BIT<21> SET WILL BE BYPASSED
0566  1053 ;       AS THIS INDICATES PAGES BELONGING TO  VDS INTERNAL CODE.
0566  1054 ;
0566  1055 ;       IF CALLED VIA $DS_RELSYSBUF, PAGES WILL BE DEALLOCATED STARTING
0566  1056 ;       WITH THE VIRTUAL ADDRESS PASSED, AND SUBJECT TO PTE<21>=1.
0566  1057 ;
0566  1058 ; CALLING SEQUENCE:
0566  1059 ;
0566  1060 ;       STARLET PROCEDURE CALL.
0566  1061 ;
0566  1062 ; INPUT PARAMETERS:
0566  1063 ;
0566  1064 ;       CNTREG$_PAGCNT(AP) =    NUMBER OF PAGES TO REMOVE FROM THE END OF THE
0566  1065 ;                               PROGRAM REGION.
0566  1066 ;       CNTREG$_RETADR(AP) =    ADDRESS OF A 2-LONGWORD ARRAY IN WHICH  TO RETURN
0566  1067 ;                               THE VIRTUAL ADDRESSES OF THE STARTING PAGE AND
0566  1068 ;                               ENDING PAGE OF THE REMOVED AREA.
0566  1069 ;       CNTREG$_ACMODE(AP) =    ACCESS MODE. IGNORED.
0566  1070 ;       CNTREG$_REGION(AP) =    REGION CODE:
0566  1071 ;                                       0 = PROGRAM (P0) REGION.
0566  1072 ;                                       1 = CONTROL (P1) REGION.
0566  1073 ;                                       2 = SYSTEM REGION.
0566  1074 ;                                       3 = USED BY DIAGNOSTIC SUPERVISOR AND
0566  1075 ;                                           CRD
0566  1076 ;
0566  1077 ;       CNTREG$_VA(AP)    =     FOR DS$_RELSYSBUF, CONTAINS THE STARTING VA TO
0566  1078 ;                               DEALL FROM                                      [31]
0566  1079 ;                               FOR DS$_RELBUF, WILL CONTAIN ZERO               [31]
0566  1080 ;
0566  1081 ; IMPLICIT INPUTS:
0566  1082 ;
0566  1083 ;       NONE
0566  1084 ;
0566  1085 ; OUTPUT PARAMETERS:
0566  1086 ;
0566  1087 ;       @CNTREG$_RETADR(AP) =   2-LONGWORD ARRAY CONTAINING THE STARTING AND
0566  1088 ;                               ENDING ADDRESSES OF THE REMOVEDED REGION.
0566  1089 ;
0566  1090 ; IMPLICIT OUTPUTS:
0566  1091 ;
0566  1092 ;       NONE
0566  1093 ;
0566  1094 ; COMPLETION CODES:
0566  1095 ;
0566  1096 ;       SS$_NORMAL =    SERVICE SUCESSFULLY COMPLETED.
0566  1097 ;       SS$_ILLPAGCNT = PAGE COUNT < 1.
0566  1098 ;       SS$_PAGOWNVIO = PAGE OWNED BY MORE PRIVILAGED ACCESS MODE.
0566  1099 ;
0566  1100 ; SIDE EFFECTS:
0566  1101 ;
```

```
                     0566  1102 ;       NONE
                     0566  1103 ;
                     0566  1104 ; REGISTER USAGE:
                     0566  1105 ;
                     0566  1106 ;       R0 =    RETURN STATUS CODE.
                     0566  1107 ;       R1 =    'REGION' END PTE.
                     0566  1108 ;       R2 =    'REGION' BEGINNING PTE.
                     0566  1109 ;       R3 =    VIRTUAL ADDRESS.
                     0566  1110 ;       R4 =    PAGE COUNTER.
                     0566  1111 ;       R5 =    POPT POINTER.
                     0566  1112 ;--
```

```
                     003C  0566  1114          .ENTRY  EXE$CNTREG,^M<R2,R3,R4,R5>      ; ENTRY POINT
                           0568  1115
                  50 01 D0 0568  1116          MOVL    #SS$_NORMAL, R0                 ; START ASSUMING THE BEST!
            03 00 10 AC CF 056B  1117          CASEL   CNTREG$_REGION(AP), #0, #3      ; DISPATCH ON REGION CODE.
                           0570  1118
                  000C'    0570  1119 10$:     .WORD   15$ - 10$                       ; P0          (0)     [31]
                  0049'    0572  1120          .WORD   30$ - 10$                       ; P1          (1)
                  0063'    0574  1121          .WORD   40$ - 10$                       ; SYS         (2)
                  0031'    0576  1122          .WORD   20$ - 10$                       ; SUPERVISOR. (3)
                           0578  1123
               3D 10 AC    E8 0578  1124          BLBS    CNTREG$_REGION(AP), 30$         ; REGION INDICATOR.
                           057C  1125
                           057C  1126 ;++                                                                  [31]begin
                           057C  1127 ; Arrival here means a $DS_RELBUF request of P0 pages, or a $DS_RELSYSBUF
                           057C  1128 ; request from internal use.
                           057C  1129 ; The normal action is to set the P0PT POINTER to the address following
                           057C  1130 ; the end of the P0PT, and to have a pointer targeted to one beyond the
                           057C  1131 ; last VA of the P0PT. From here the code works its way up from the bottom
                           057C  1132 ; of the P0PT, blindly deallocated the last allocated pages of P0 space.
                           057C  1133 ;
                           057C  1134 ; The following Algorithm modifies pointers using the starting VA argument
                           057C  1135 ; supplied from a $DS_RELSYSBUF
                           057C  1136 ; service:
                           057C  1137 ;
                           057C  1138 ;       IF CNTREG$_VA(AP) = 0      ! no VA was passed
                           057C  1139 ;             THEN PROCESS AS NORMAL $DS_RELBUF P0 REQUEST
                           057C  1140 ;              ELSE TARGET P0PT PTR TO BEYOND LAST REQUESTED PAGES PTE
                           057C  1141 ;                   SET VA POINTER TO BEYOND LAST REQUESTED PAGE   [31]end
                           057C  1142 ;       ENDIF
                           057C  1143 ;--
                           057C  1144
                     14 AC D5 057C  1145 15$:    TSTL    CNTREG$_VA(AP)  ; CHECK FOR VA                          [31]
                        20 13 057F  1146         BEQL    20$             ; GO SERVICE $DS_RELBUF                 [31]
                  52 00000000'EF DE 0581  1147         MOVAL   P0PT_BASE, R2   ; POINTER TO TOP OF PAGE TABLE.         [31]
            53 04 AC 00000200 8F C5 0588  1148         MULL3   #^X200,CNTREG$_PAGCNT(AP),R3 ; NUMBER OF BYTES TO DEALLOCATE [31]
                     53 14 AC C0 0591  1149         ADDL    CNTREG$_VA(AP),R3       ; POINTER TO LAST ENDING VA +1  [31]
                  51 53 F7 8F 78 0595  1150         ASHL    #-9,R3,R1               ; GENERATE OFFSET INTO P0PT [31]
                     51 6241 DE 059A  1151         MOVAL   (R2)[R1],R1             ; P0PT PTR TO PTE BEYOND LAST REQUESTED [31]
                        004D 31 059E  1152         BRW     50$                     ; BRANCH TO COMMON      [31]
                           05A1  1153
                  51 00000000'EF D0 05A1  1154 20$:    MOVL    L_P0LEN, R1                     ; P0 PAGE COUNT.
                     53 51 09 78 05A8  1155         ASHL    #9, R1, R3              ; END VIRTUAL ADDRESS.
                  52 00000000'EF DE 05AC  1156         MOVAL   P0PT_BASE, R2           ; POINTER TO TOP OF PAGE TABLE.
                     51 6241 DE 05B3  1157         MOVAL   (R2)[R1], R1            ; POINTER TO END OF PAGE TABLE.
                        35 11 05B7  1158         BRB     50$                     ; BR TO COMMON CODE.
                           05B9  1159
                  52 00000004'EF D0 05B9  1160 30$:    MOVL    A_P1BUFPTE, R2          ; START OF P1 PTE.
                     51 01F8 C2 DE 05C0  1161         MOVAL   K_P1PAGES*4(R2), R1     ; End of P1 page table
53 00000008'EF 0000FC00 8F C1 05C5  1162         ADDL3   #K_P1PAGES@9,A_P1BUFVA,R3       ; LAST VIRTURAL ADDRESS
                        1B 11 05D1  1163         BRB     50$                     ; BR TO COMMON CODE.
                           05D3  1164
                  51 0000000C'EF D0 05D3  1165 40$:    MOVL    L^A_SPTEND, R1                  ; END OF SYS PAGE TABLE.        [27]
                  52 00000000'EF D0 05DA  1166         MOVL    MMG$GL_SPTBASE, R2              ; Start of SYS page table
                     53 01 1F 78 05E1  1167         ASHL    #31, #1, R3             ; START OF SYS VIRTUAL ADR.
                     7E 51 52 C3 05E5  1168         SUBL3   R2, R1, -(SP)           ; SPT LENGTH.
                  53 16 07 8E F0 05E9  1169         INSV    (SP)+, #9-2, #20+2, R3          ; COMPUTE LAST VIRTUAL ADDRESS.
```

```
                     53   D7  05EE  1171 50$:    DECL    R3                                      ; END OF PREVIOUS PAGE.
                     54   D4  05F0  1172          CLRL    R4                                      ; INIT PAGE COUNTER.
                                05F2  1173
                     71   D5  05F2  1174 60$:    TSTL    -(R1)                                   ; LOOK FOR VALID BIT.
                     72   18  05F4  1175          BGEQ    100$                                    ; BR IF NOT VALID.
                                05F6  1176
          10 AC   61   02   17  05F6  1177          CMPZV   #PTE$V_OWN, #PTE$S_OWN, (R1), - ; CHECK 'OWN' CODE
                                05FC  1178            CNTREG$_REGION(AP)                    ;  FOR REGION.
                     6A   12  05FC  1179          BNEQ    100$                                    ; WRONG OWNER.
                                05FE  1180
                                05FE  1181 ;++
                                05FE  1182 ; The deallocation algorithm follows:                        begin[31]
                                05FE  1183 ;
                                05FE  1184 ;       IF DS$RELBUF SERVICE    (VA = zero)
                                05FE  1185 ;          THEN
                                05FE  1186 ;               IF PTE<21> EQL 1
                                05FE  1187 ;                  THEN DO NOT DEALL, GET NEXT PTE
                                05FE  1188 ;                  ELSE CONTINUE NORMAL PROCESSING
                                05FE  1189 ;          ELSE
                                05FE  1190 ;               IF PTE<21> EQL 1
                                05FE  1191 ;                  THEN DEALLOCATE PAGE
                                05FE  1192 ;                  ELSE REPORT SUPERVISOR ERROR (ERRSUP).
                                05FE  1193 ;               ! This would be the result of an incorrect supplied VA,
                                05FE  1194 ;               ! page count, region, or blown page tables.
                                05FE  1195 ;--
                                05FE  1196
                14 AC   D5  05FE  1197          TSTL    CNTREG$_VA(AP)                  ; IS THIS A $DS_RELSYSBUF REQUEST?
                     07   12  0601  1198          BNEQ    62$                             ; YES, BRANCH
             61 61   15   E0  0603  1199          BBS     #21,(R1),100$                   ; NOT THE RIGHT PTE, GET THE NEXT ONE
                   0007   31  0607  1200          BRW     65$                             ; BRANCH FOR NORMAL DEALLOCATION
             03 61   15   E0  060A  1201 62$:     BBS     #21,(R1),65$                    ; DEALLOCATE INTERNAL_VDS PAGE
                   0022   31  060E  1202          BRW     67$                             ; BAD PAGE TABLES, POSSIBLY WRONG VA,
                              0611  1203                                                  ; PAGCNT, OR REGION!
55   61   FFF00000 8F   CB  0611  1204 65$:     BICL3   #^CPTE$M_PFN!^X100000, (R1), R5 ; ISOLATE PFN                   end[31]
  55   00000000'EF45   DE  0619  1205          MOVAL   POPT_BASE[R5], R5               ; GET P0 PTE, SAME PAGE.
            55   51   D1  0621  1206          CMPL    R1, R5                          ; CHECK FOR P0 REGION.
                 22   13  0624  1207          BEQL    80$                             ; BR IF P0.
       7E   65   61   CD  0626  1208          XORL3   (R1),(R5), -(SP)                ; GET DIFFERENCE BETWEEN PTE'S
    8E   827FFFFF 8F   D3  062A  1209          BITL    #^C<PTE$M_MODIFY ! -            ; Ignore MODIFY,                [19]
                           0631  1210              PTE$M_PROT !   -                    ; PROT,                         [19]
                           0631  1211              PTE$M_OWN>,    -                    ;  and OWN fields               [19]
                           0631  1212              (SP)+                               ; Do any useful fields differ?
                 13   13   0631  1213          BEQL    70$                             ; NO, RELEASE IT
                           0633  1214 67$:     ERRSUP_S                                ; BUM PAGE TABLES.
       00000000'EF   DF   0633                          PUSHAL  $MODULE
                     00   DD  0639                     PUSHL   #0
                     01   DD  063B                     PUSHL   #$ER
    00000000'EF      03   FB  063D                     CALLS   #3, DS_ERRSUP
                     3E   11  0644  1215          BRB     120$
                              0646  1216
                     61   D4  0646  1217 70$:     CLRL    (R1)                            ; RELEASE 'REGION' PAGE.
                              0648  1218
    65   81800000 8F   CA  0648  1219 80$:     BICL2   #PTE$M_VALID ! PTE$M_OWN,(R5)   ; RELEASE P0 PAGE.
    65   00200000 8F   CA  064F  1220          BICL    #^X200000,(R5)                  ; CLEAR PTE<21>, IN CASE SET.[31]
  65   04   1B   04   F0  0656  1221          INSV    #4,#27,#4,(R5)                  ; Set prot to full access [35] ;G:2
                     54   D5  065B  1222          TSTL    R4                              ; FIRST PAGE?
                     02   12  065D  1223          BNEQ    90$                             ; BR OF NOT.
```

```
                  53   DD  065F  1224          PUSHL   R3                                  ; SAVE VIRTUAL ADDR.
                           0661  1225
      0D 54   04 AC   F2  0661  1226 90$:     AOBLSS  CNTREG$_PAGCNT(AP), R4, 110$        ; COUNT PAGES.
                  23   11  0666  1227          BRB     130$                                ; SUCCESS EXIT.
                           0668  1228
                  54   D5  0668  1229 100$:    TSTL    R4                                  ; FIRST PAGE?
                  07   13  066A  1230          BEQL    110$                                ; SKIP IF SO.
   50   00660080 8F   D0  066C  1231          MOVL    #DS$_FRAGBUF, R0                    ; WARN CALLER.
                           0673  1232
   53   00000200 8F   C2  0673  1233 110$:    SUBL    #512, R3                            ; ADJUST VITRUAL ADR.
            52   51   D1  067A  1234          CMPL    R1, R2                              ; CHECK FOR OUT OF PAGE TABLE.
                  02   1A  067D  1235          BGTRU   115$                                ; BR IF MORE.
                  03   11  067F  1236          BRB     120$                                ; NO MORE
                FF6E   31  0681  1237 115$:    BRW     60$                                 ; GO GET MORE
                           0684  1238
   50   000001EC 8F   D0  0684  1239 120$:    MOVL    #SS$_PAGOWNVIO, R0                  ; RAN OUT OF PAGES.
                           068B  1240
              08 AC   D5  068B  1241 130$:    TSTL    CNTREG$_RETADR(AP)                  ; CHECK FOR RETURN REQUESTED.
                  0B   13  068E  1242          BEQL    EXE$CNTREG_X                        ; EXIT IF NOT.
      53   01FF 8F   AA  0690  1243          BICW2   #^X1FF, R3                          ; BEGINNING OF PAGE.
                  53   DD  0695  1244          PUSHL   R3                                  ; SAVE FIRST VIRTUAL ADR.
         08 BC   8E   7D  0697  1245          MOVQ    (SP)+, @CNTREG$_RETADR(AP)          ; RETURN VIRTUAL ADDRESSES.
                           069B  1246
                           069B  1247 EXE$CNTREG_X:
                       04  069B  1248          RET                                         ; RETURN.
```

```
                                069C  1250          .SBTTL  TURN MEMORY MANAGEMENT ON.
                                069C  1251 ;++
                                069C  1252 ; FUNCTIONAL DESCRIPTION:
                                069C  1253 ;
                                069C  1254 ;       This routine can be called by the program to enable memory management.
                                069C  1255 ;       All memory mapping, page tables and base and length registers
                                069C  1256 ;       must be setup prior to calling this routine.
                                069C  1257 ;
                                069C  1258 ; CALLING SEQUENCE:
                                069C  1259 ;
                                069C  1260 ;       NONE
                                069C  1261 ;       CALLS   #0,DS$MMON
                                069C  1262 ;
                                069C  1263 ; INPUT PARAMETERS:
                                069C  1264 ;
                                069C  1265 ;       NONE
                                069C  1266 ;
                                069C  1267 ; IMPLICIT INPUTS:
                                069C  1268 ;
                                069C  1269 ;       NONE
                                069C  1270 ;
                                069C  1271 ; OUTPUT PARAMETERS:
                                069C  1272 ;
                                069C  1273 ;       NONE
                                069C  1274 ;
                                069C  1275 ; IMPLICIT OUTPUTS:
                                069C  1276 ;
                                069C  1277 ;       NONE
                                069C  1278 ;
                                069C  1279 ; COMPLETION CODES:
                                069C  1280 ;
                                069C  1281 ;       SS$_WASSET      If was enabled before
                                069C  1282 ;       SS$_WASCLR      If was not enabled
                                069C  1283 ;       DS$_NOTALLOWMP  Service not allowed if active mp system
                                069C  1284 ;
                                069C  1285 ; SIDE EFFECTS:
                                069C  1286 ;
                                069C  1287 ;       Memory management is enabled
                                069C  1288 ;
                                069C  1289 ;--
                                069C  1290
                          0000  069C  1291 .ENTRY  DSX$MMON,^M<>                  ; ENTRY POINT
                50    01    D0  069E  1292          MOVL    #1,R0                 ; Enable memory managment
                      39    10  06A1  1293          BSBB    DSR$MMENABLE          ; Do it
           50   50    03    78  06A3  1294          ASHL    #3,R0,R0              ; Position old enabled bit to bit 3
                      50    D6  06A7  1295          INCL    R0                    ; Make it success return code
      51   00660148  8F    D1  06A9  1296          CMPL    #DS$_NOTALLOWMP,R1    ; Change of MM status in mp system? [33]
                      03    12  06B0  1297          BNEQ    DSX$MMON_X            ; No, then Normal return            [33]
                50    51    D0  06B2  1298          MOVL    R1,R0                 ; else, Return error                [33]
                            04  06B5  1299 DSX$MMON_X:
                                06B5  1300          RET
```

```
                                06B6          .SBTTL  TURN MEMORY MANAGEMENT OFF.
                                06B6  ;++
                                06B6  ; FUNCTIONAL DESCRIPTION:
                                06B6  ;
                                06B6  ;       This routine can be called to disable memory managment
                                06B6  ;       However, if the operator enabled memory management with the
                                06B6  ;       SET  MM ON command, it will not disable memory management.
                                06B6  ;
                                06B6  ; CALLING SEQUENCE:
                                06B6  ;
                                06B6  ;       CALLS   #0,DS$MMOFF
                                06B6  ;
                                06B6  ; INPUT PARAMETERS:
                                06B6  ;
                                06B6  ;       NONE
                                06B6  ;
                                06B6  ; IMPLICIT INPUTS:
                                06B6  ;
                                06B6  ;       NONE
                                06B6  ;
                                06B6  ; OUTPUT PARAMETERS:
                                06B6  ;
                                06B6  ;       NONE
                                06B6  ;
                                06B6  ; IMPLICIT OUTPUTS:
                                06B6  ;
                                06B6  ;       NONE
                                06B6  ;
                                06B6  ; COMPLETION CODES:
                                06B6  ;
                                06B6  ;       SS$_WASSET      If was enabled before
                                06B6  ;       SS$_WASCLR      If was not enabled
                                06B6  ;       DS$_WARNING     If was set by operator and you can't disable it
                                06B6  ;       DS$_NOTALLOWMP  Service not allowed if active mp system
                                06B6  ;
                                06B6  ; SIDE EFFECTS:
                                06B6  ;
                                06B6  ;       NONE
                                06B6  ;       Memory management is disabled unless the operator explicitly
                                06B6  ;       enabled it.
                                06B6  ;
                                06B6  ;--
                          0000  06B6  .ENTRY  DSX$MMOFF,^M<>                  ; ENTRY POINT
50   00660000 8F            D0  06B8          MOVL    #DS$_WARNING,R0         ; Prepare for can't disable return
     0000001C'EF            95  06BF          TSTB    DS$GB_MM_ENB            ; Was MM enabled by operator?
                14          12  06C5          BNEQ    DSX$MMOFF_X             ; Yes, Do not disable it
                13          10  06C7          BSBB    DSR$MMENABLE            ; and do it
     50    50   03          78  06C9          ASHL    #3,R0,R0                ; Position old enable bit to bit 3
                50          D6  06CD          INCL    R0                      ; Make it success return code
51   00660148 8F            D1  06CF          CMPL    #DS$_NOTALLOWMP,R1      ; Change of MM status in mp system? [33]
                03          12  06D6          BNEQ    DSX$MMOFF_X             ; No, then Normal return            [33]
          50    51          D0  06D8          MOVL    R1,R0                   ; else, Return error                [33]
                                06DB  DSX$MMOFF_X:
                            04  06DB          RET
```

```
                        06DC  1357          .SBTTL  DSR$MMENABLE    Enable to disable memory management
                        06DC  1358 ;++
                        06DC  1359 ; FUNCTION DESCRIPTION:
                        06DC  1360 ;
                        06DC  1361 ;       This routine is called to enable or disable memory managment.
                        06DC  1362 ;
                        06DC  1363 ; CALLING SEQUENCE:
                        06DC  1364 ;
                        06DC  1365 ;       BSBW    DSR$MMENABLE
                        06DC  1366 ;
                        06DC  1367 ; INPUT PARAMETERS:
                        06DC  1368 ;
                        06DC  1369 ;       R0      value to be written in to PR$_MAPEN
                        06DC  1370 ;
                        06DC  1371 ; IMPLICIT INPUTS:      NONE
                        06DC  1372 ;
                        06DC  1373 ; OUTPUT PARAMETERS:
                        06DC  1374 ;
                        06DC  1375 ;       R0      Previous contents of PR$_MAPEN
                        06DC  1376 ;       R1      Completion code : DS$_NOTALLOWMP,
                        06DC  1377 ;                                 SS$_NORMAL
                        06DC  1378 ;
                        06DC  1379 ; IMPLICIT OUTPUTS:     NONE
                        06DC  1380 ;
                        06DC  1381 ; COMPLETION CODES:     NONE
                        06DC  1382 ;
                        06DC  1383 ; SIDE EFFECTS:
                        06DC  1384 ;
                        06DC  1385 ;       Memory management is enabled or disabled.
                        06DC  1386 ;--
                        06DC  1387
                        06DC  1388 DSR$MMENABLE::
                        06DC  1389          CMK     MMENABLE                    ; Change to Kernel mode
              7E    DC  06DC                MOVPSL  -(SP)
       06 6E  1A    E0  06DE                BBS     #PSL$V_IS, (SP), 30000$
              8E    D4  06E2                CLRL    (SP)+
              03    BC  06E4                CHMK    #CMK$_MMENABLE
              06    11  06E6                BRB     30001$
       000006EF'EF  16  06E8        30000$: JSB     CMK$MMENABLE
                        06EE        30001$:
              05    06  06EE  1390          RSB                                 ; Return to caller
                        06EF  1391
                        06EF  1392 CMK$MMENABLE::
           39   00  DA  06EF  1393          MTPR    #0,#PR$_TBIA                ; Invalidate the TB before enabling MM
           7E   38  DB  06F2  1394          MFPR    #PR$_MAPEN,-(SP)            ; Get current state of memory managment
50    50   01   00  EF  06F5  1395          EXTZV   #0,#1,R0,R0                 ; Only 1 bit
      00000000'EF  D5  06FA  1396          TSTL    MP$GR_BOOTED_PROCESSORS     ; Are we an mp system?          [32][33]
           0E   13  0700  1397          Beql    10$                         ; Branch if not mp system       [32][33]
           6E   50  D1  0702  1398          CMPL    R0,(SP)                     ; Compare desired state with old[33]
           0C   13  0705  1399          Beql    15$                         ; Same state so don't change    [33]
51    00660148 8F  D0  0707  1400          MOVL    #DS$_NOTALLOWMP,R1          ; State change not allowed in mp[33]
           06   11  070E  1401          BRB     20$                         ; Continue                      [33]
           38   50  DA  0710  1402 10$:    MTPR    R0,#PR$_MAPEN               ; Set new desired state
           51   01  D0  0713  1403 15$:    MOVL    #SS$_NORMAL,R1              ; Successful status             [33]
                01  BA  0716  1404 20$:    POPR    #^M<R0>                     ; Get old state
                    02  0718  1405          REI                                 ; Return to previous mode
```

```
0719  1407          .SBTTL  SET PROTECTION ON PAGES.
0719  1408 ;++
0719  1409 ; FUNCTIONAL DESCRIPTION:
0719  1410 ; SET THE ACCESS PROTECTION ON A VIRTUAL PAGE
0719  1411 ;
0719  1412 ; CALLING SEQUENCE:
0719  1413 ;
0719  1414 ;       PROCEDURE CALL.
0719  1415 ;
0719  1416 ; INPUT PARAMETERS:
0719  1417 ;
0719  1418 ;       SETPRT$_INADR(AP) =     ARAY ADDRESS (INPUT)
0719  1419 ;       SETPRT$_RETADR(AP) =    ARAY ADDRESS (OUTPUT)
0719  1420 ;       SETPRT$_ACMODE(AP) =    ACCESS MODE (OPTIONAL)
0719  1421 ;       SETPRT$_PROT(AP) =      NEW PROTECTION (BITS <3:0>)
0719  1422 ;       SETPRT$_PRVPRT(AP) =    ADDRESS TO STORE PREVIOUS PROTECTION
0719  1423 ;
0719  1424 ; IMPLICIT INPUTS:
0719  1425 ;
0719  1426 ;       NONE
0719  1427 ;
0719  1428 ; OUTPUT PARAMETERS:
0719  1429 ;
0719  1430 ;       R0 = COMPLETION CODE
0719  1431 ;
0719  1432 ; IMPLICIT OUTPUTS:
0719  1433 ;
0719  1434 ;       NONE
0719  1435 ;
0719  1436 ; COMPLETION CODES:
0719  1437 ;
0719  1438 ;       DS$_NORMAL = SERVICE SUCCESSFULLY COMPLETED
0719  1439 ;       SS$_ACCVIO = ACCESS VIOLATION
0719  1440 ;       SS$_LENVIO = LENGTH VIOLATION
0719  1441 ;
0719  1442 ; SIDE EFFECTS:
0719  1443 ;
0719  1444 ;       NONE
0719  1445 ;
0719  1446 ;--
```

```
                          007C  0719  1448 .ENTRY  EXE$SETPRT,^M<R2,R3,R4,R5,R6>   ; ENTRY POINT
                                071B  1449
                                071B  1450         CMK     SETPRT          ; Do set protection                          [20]
                     7E   DC    071B               MOVPSL  -(SP)
            06 6E    1A   E0    071D               BBS     #PSL$V_IS, (SP), 30002$
                     8E   D4    0721               CLRL    (SP)+
                     0D   BC    0723               CHMK    #CMK$_SETPRT
                     06   11    0725               BRB     30003$
          0000072E'EF     16    0727       30002$: JSB     CMK$SETPRT
                                072D       30003$:
                          04    072D  1451         RET                     ; Return home                                [20]
                                072E  1452
                                072E  1453 CMK$SETPRT::
            50   08 AC    D0    072E  1454         MOVL    SETPRT$_RETADR(AP), R0          ; INIT THE RETURN ARAY
                     06   13    0732  1455         BEQL    5$                              ; IF THERE IS ONE
               80    00   D2    0734  1456         MCOML   #0,(R0)+                        ;
               60    00   D2    0737  1457         MCOML   #0,(R0)
                                073A  1458 5$:
            50   04 AC    D0    073A  1459         MOVL    SETPRT$_INADR(AP), R0           ; INPUT ARAY
       53   80   F7 8F    78    073E  1460         ASHL    #-9,(R0)+,R3                    ; STARTING VA PFN
       54   60   F7 8F    78    0743  1461         ASHL    #-9,(R0),R4                     ; ENDING VA PFN
          55   54    53   C3    0748  1462         SUBL3   R3,R4,R5                        ; PAGES MINUS 1
                     0A   18    074C  1463         BGEQ    10$                             ; BETTER BE ONE
     50   00660020 8F    D0    074E  1464         MOVL    #DS$_PROGERR,R0                 ; INVALID ARGUMENT
                   0031   31    0755  1465         BRW     SETPRT_XIT                      ; RETURN
                                0758  1466 10$:
            54   08 AC    D0    0758  1467         MOVL    SETPRT$_RETADR(AP), R4          ; RETURN ADDRESS ARAY
                     55   D6    075C  1468         INCL    R5                              ; NUMBER OF PAGES
                                075E  1469 20$:
     56   53   02   15   EF    075E  1470         EXTZV   #21,#2,R3,R6                    ; ADDRESS SPACE INDEX
                  72'AF   9F    0763  1471         PUSHAB  B^30$                           ; Setup return address
                                0766  1472         CASE    R6,LIMIT=#0,TYPE=L, -
                                0766  1473                 DISPLIST=<PZERO,PONE,SYSTEM,RESERVD>
          03'  00   56   CF    0766               CASEL   R6,#0,S^#<<30005$-30004$>/2>-1
                                076A       30004$:
                          0020' 076A               .SIGNED_WORD    PZERO-30004$
                          0036' 076C               .SIGNED_WORD    PONE-30004$
                          004C' 076E               .SIGNED_WORD    SYSTEM-30004$
                          0062' 0770               .SIGNED_WORD    RESERVD-30004$
                                0772       30005$:
                                0772  1474
                  14 50   E9    0772  1475 30$:    BLBC    R0,SETPRT_XIT                   ; IF ERROR EXIT
                     53   D6    0775  1476         INCL    R3                              ; INCREMENT THE PFN
                  E4 55   F5    0777  1477         SOBGTR  R5,20$                          ; DO THE NEXT PAGE
                  08 AC   D5    077A  1478         TSTL    SETPRT$_RETADR(AP)              ; IS THERE A RETURN ADDRESS ?
                     0A   13    077D  1479         BEQL    SETPRT_XIT                      ; NO -
08 BC   04 BC   000001FF 8F   CB    077F  1480         BICL3   #^X1FF, @SETPRT$_INADR(AP), -   ; YES-INDICATE THE STARTING     [19]
                                0789  1481                   @SETPRT$_RETADR(AP)           ; VIRTUAL ADDRESS
                                0789  1482 SETPRT_XIT:
                          02    0789  1483         REI                     ; Return from exception                      [20]
```

```
                        078A  1485 ;+
                        078A  1486 ; SET PROTECTION FOR P0 ADDRESS SPACE <3FFFFFFF:00000000>
                        078A  1487 ; R3 = VIRTUAL PAGE FRAME NUMBER
                        078A  1488 ; R4 = ENDING RETURN VIRTUAL ADDRESS
                        078A  1489 ;-
                        078A  1490 PZERO:
51   53   15   00   EF  078A  1491          EXTZV   #0,#21,R3,R1                    ; PFN <29:9>
          50   09   DB  078F  1492          MFPR    #PR$_P0LR,R0                    ; P0 LENGTH REGISTER
          51   50   D1  0792  1493          CMPL    R0,R1                           ; VA<29:9> GEQ P0LR ?
               06   14  0795  1494          BGTR    10$                             ; NO - LENGTH WITHIN RANGE
     50   018C 8F   3C  0797  1495          MOVZWL  #SS$_LENVIO,R0                  ; YES- LENGTH FAULT
               05   10  079C  1496          RSB                                     ; RETURN
          31   10   BB  079D  1497 10$:     BSBB    SET                             ; SET THE PROTECTION
               05   10  079F  1498          RSB                                     ; RETURN
                        07A0  1499
                        07A0  1500 ;+
                        07A0  1501 ; SET PROTECTION FOR P1 ADDRESS SPACE <7FFFFFFF:40000000>
                        07A0  1502 ; R3 = VIRTUAL PAGE FRAME NUMBER
                        07A0  1503 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                        07A0  1504 ;-
                        07A0  1505 PONE:
51   53   15   00   EF  07A0  1506          EXTZV   #0,#21,R3,R1                    ; PFN <29:9>
          50   0B   DB  07A5  1507          MFPR    #PR$_P1LR,R0                    ; P1 LENGTH REGISTER
          51   50   D1  07A8  1508          CMPL    R0,R1                           ; VA<29:9> LSS P1LR
               06   15  07AB  1509          BLEQ    10$                             ; NO - LENGTH WITHIN RANGE
     50   018C 8F   3C  07AD  1510          MOVZWL  #SS$_LENVIO,R0                  ; YES- LENGTH VIOLATION
               05   10  07B2  1511          RSB                                     ; RETURN
          1B   10   BB  07B3  1512 10$:     BSBB    SET                             ; SET PROTECTION
               05   10  07B5  1513          RSB                                     ; RETURN
                        07B6  1514
                        07B6  1515 ;+
                        07B6  1516 ; SET PROTECTION FOR SYSTEM ADDRESS SPACE <BFFFFFFF:80000000>
                        07B6  1517 ; R3 = VIRTUAL PAGE FRAME NUMBER
                        07B6  1518 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                        07B6  1519 ;-
                        07B6  1520 SYSTEM:
51   53   15   00   EF  07B6  1521          EXTZV   #0,#21,R3,R1                    ; PFN <30:9>
          50   0D   DB  07BB  1522          MFPR    #PR$_SLR,R0                     ; SYSTEM LENGTH REGISTER
          51   50   D1  07BE  1523          CMPL    R0,R1                           ; VA<30:9> GTR SLR
               06   14  07C1  1524          BGTR    10$                             ; NO - LENGTH WITHIN RANGE
     50   018C 8F   3C  07C3  1525          MOVZWL  #SS$_LENVIO,R0                  ; YES- LENGTH VIOLATION
               05   10  07C8  1526          RSB                                     ; RETURN
          05   10   BB  07C9  1527 10$:     BSBB    SET                             ; SET PROTECTION
               05   10  07CB  1528          RSB                                     ; RETURN
                        07CC  1529
                        07CC  1530 ;+
                        07CC  1531 ; SET PROTECTION FOR RESERVED ADDRESS SPACE <FFFFFFFF:C0000000>
                        07CC  1532 ; THIS FUNCTION IS ILLEGAL
                        07CC  1533 ; R3 = VIRTUAL PAGE FRAME NUMBER
                        07CC  1534 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                        07CC  1535 ;-
                        07CC  1536 RESERVD:
          50   0C   D0  07CC  1537          MOVL    #SS$_ACCVIO,R0                  ; INDICATE ACCESS VIOLATION
               05   10  07CF  1538          RSB                                     ; RETURN
                        07D0  1539
                        07D0  1540 ;+
                        07D0  1541 ; SUBROUTINE TO SET PAGE PROTECTION
```

```
                                07D0  1542 ; R3 = VIRTUAL PAGE FRAME NUMBER
                                07D0  1543 ; R4 = RETURN ENDING VIRTUAL ADDRESS
                                07D0  1544 ;-
                                07D0  1545 SET:
            51   53   09   78  07D0  1546         ASHL    #9,R3,R1                ; MAKE ADDRESS OF PAGE FROM PFN
                      2E   10  07D4  1547         BSBB    RPTEADR                 ; GET ADDRESS OF PTE
                   14 AC   D5  07D6  1548         TSTL    SETPRT$_PRVPRT(AP)      ; IS THERE A PREVIOUS
                                07D9  1549                                         ; PROTECTION ADDRESS?
                      09   13  07D9  1550         BEQL    10$                     ; NO -
        50   61    04   1B   EF  07DB  1551         EXTZV   #27, #4, (R1), R0       ; Get old protection            [20]
             14 BC   50   90  07E0  1552         MOVB    R0, @SETPRT$_PRVPRT(AP) ; Store it                      [20]
                                07E4  1553                                         ; YES- PASS TO CALLER
                                07E4  1554                                         ; AS PREVIOUS PROTECTION
                                07E4  1555 10$:
      61    04   1B   10 AC   F0  07E4  1556         INSV    SETPRT$_PROT(AP),#27,#4,(R1) ; INSERT NEW PROTECTION
            51   53   09   78  07EA  1557         ASHL    #9, R3, R1              ; FORM ADDRESS
                 3A   51   DA  07EE  1558         MTPR    R1, #PR$_TBIS           ; FLUSH TRANSLATION BUF.
              50   08 AC   D0  07F1  1559         MOVL    SETPRT$_RETADR(AP),R0   ; IS THERE A RETURN ADDRESS ?
                      09   13  07F5  1560         BEQL    20$                     ; NO -
  04 A0   51   000001FF 8F   C9  07F7  1561         BISL3   #^X1FF,R1,4(R0)         ; RETURN HIGHEST ADDRESS OF PAGE
                                0800  1562 20$:
                 50   01   D0  0800  1563         MOVL    #SS$_NORMAL,R0          ; INDICATE SUCCESS
                                05  0803  1564         RSB                             ; RETURN
                                0804  1565
```

```
                                0804  1567          .SBTTL  CALCULATE PTE ADDRESS SUBROUTINE
                                0804  1568 ;++
                                0804  1569 ; RPTEADR - CALCULATE PTE ADDRESS SUBROUTINE
                                0804  1570 ;
                                0804  1571 ; THIS SUBROUTINE IS CALLED TO CALCULATE THE ADDRESS OF A PAGE TABLE
                                0804  1572 ; ENTRY, GIVEN A VIRTUAL ADDRESS
                                0804  1573 ;
                                0804  1574 ; INPUTS:
                                0804  1575 ;
                                0804  1576 ;       R1 = VIRTUAL ADDRESS
                                0804  1577 ;
                                0804  1578 ; OUTPUTS:
                                0804  1579 ;
                                0804  1580 ;       R1 = PTE ADDRESS
                                0804  1581 ;--
                                0804  1582
                                0804  1583 RPTEADR::                                ;
          36 51    1F    E0    0804  1584          BBS     #31,R1,100$               ; IF C, P0 OR P1 ADDRESS                    ;G:7
          05 51    1E    E0    0808  1585          BBS     #30,R1,20$                ; IF S, P1 ADDRESS                          ;G:7
             7E    08    DB    080C  1586          MFPR    #PR$_P0BR,-(SP)           ; ELSE, GET P0PT SVA BASE
                   03    11    080F  1587          BRB     30$                       ;  AND PROCEED
             7E    0A    DB    0811  1588 20$:     MFPR    #PR$_P1BR,-(SP)           ; GET P1PT SVA BASE
       51    51    15    09    EF    0814  1589 30$:     EXTZV   #9,#21,R1,R1              ; PVA PFN
             51    51    02    78    0819  1590          ASHL    #2,R1,R1                  ; SPT'PxPT OFFSET
             51    8E    C0    081D  1591          ADDL2   (SP)+,R1                  ; SVA ( PHYSICAL ADD FOR RTUVAX)            ;G:7
00'8F    0000FE17'EF    91    0820  1592          CMPB    DSA$GL_SID+3,#PR$_SID_TYPRTUV2 ; Is it RT-VAX                       [39]
                   13    13    0828  1593          BEQL    50$                       ; Skip if RT-VAX                            [39]
       7E    51    09    00    EF    082A  1594          EXTZV   #0,#9,R1,-(SP)            ; SAVE PxPT OFFSET
                   0D    10    082F  1595          BSBB    100$                      ; CALCULATE SPTE ADDRESS
       51    61    15    00    EF    0831  1596          EXTZV   #0,#21,(R1),R1            ; PHYSICAL PxPT ADDRESS
             51    51    09    78    0836  1597          ASHL    #9,R1,R1                  ; SHIFT TO PFN POSITION
                   51    8E    C0    083A  1598          ADDL2   (SP)+,R1                  ; PxPTE ADDRESS
                         05    083D  1599 50$:     RSB                               ; RETURN                                    ;G:7
                               083E  1600 ;
                               083E  1601 ; CALCULATE SPTE PHYSICAL ADDRESS
                               083E  1602 ;
                               083E  1603
                   7E    0C    DB    083E  1604 100$:    MFPR    #PR$_SBR,-(SP)            ; SPT BASE
       51    51    15    09    EF    0841  1605          EXTZV   #9,#21,R1,R1              ; SVA PFN
             51    51    02    78    0846  1606          ASHL    #2,R1,R1                  ; SPT OFFSET
                   51    8E    C0    084A  1607          ADDL2   (SP)+,R1                  ; SPTE ADDRESS
                         05    084D  1608          RSB                               ; RETURN
```

```
                     084E  1610 ;**
                     084E  1611 ; FUNCTIONAL DESCRIPTION:
                     084E  1612 ;
                     084E  1613 ;       These entry points are here only to satify unneeded
                     084E  1614 ;       functionality.
                     084E  1615 ;
                     084E  1616 ;       MMG$IOLOCK      Routine to lock page(s) in memory, returns PTE addr
                     084E  1617 ;       MMG$UNLOCK      Routine to unlock page(s)
                     084E  1618 ;       INI$RDONLY      Routine to make read only parts of DS readonly
                     084E  1619 ;       INI$WRITEABLE   Routine to make DS writeable
                     084E  1620 ;
                     084E  1621 ; CALLING SEQUENCE:
                     084E  1622 ;
                     084E  1623 ;       BSBW    ???
                     084E  1624 ;
                     084E  1625 ; INPUT PARAMETERS:
                     084E  1626 ;
                     084E  1627 ;       Varied but all unused
                     084E  1628 ;
                     084E  1629 ; IMPLICIT INPUTS:      NONE
                     084E  1630 ;
                     084E  1631 ; OUTPUT PARAMETERS:
                     084E  1632 ;
                     084E  1633 ;       MMG$IOLOCK      R1 - Address of PTE for address in R0 on entry
                     084E  1634 ;
                     084E  1635 ; IMPLICIT OUTPUTS:     NONE
                     084E  1636 ;
                     084E  1637 ; SIDE EFFECTS:         NONE
                     084E  1638 ;
                     084E  1639 ; COMPLETION CODES:     SS$_NORMAL - Always
                     084E  1640 ;--
                     084E  1641
                     084E  1642 MMG$IOLOCK::
   51    50    D0    084E  1643          MOVL    R0,R1                   ; Copy starting virtural address
      B0 AF    16    0851  1644          Jsb     RPTEADR                 ; Get address of First PTE      [27]
00 51    1F    E2    0854  1645          BBSS    #31,R1,MMG$UNLOCK       ; Make it a system virtural address also
                     0858  1646
                     0858  1647 MMG$UNLOCK::
   50    01    D0    0858  1648          MOVL    #SS$_NORMAL,R0          ; Set success
                     05    085B  1649          RSB
```

```
                                085C  .SBTTL DSX$MAPDBGBLOCK - MAP A BLOCK OF MEMORY FOR THE DEBUGGER          [23]
                                085C
                                085C  ;++                                                                      [23]
                                085C  ; FUNCTIONAL DESCRIPTION                                                 [23]
                                085C  ;       This routine will map a section of memory space and add the      [23]
                                085C  ; space to the debugger's total memory allocation.  The new space is     [23]
                                085C  ; allocated just below that which has already been allocated, resulting [23]
                                085C  ; in contiguous downward growth of memory space assigned to the debug-   [23]
                                085C  ; ger.  This routine is for standalone use only.                         [23]
                                085C  ;                                                                        [23]
                                085C  ; INPUTS:                                                                [23]
                                085C  ;       4(AP) - size of requested memory block                           [23]
                                085C  ;       8(AP) - address of location to return starting address of        [23]
                                085C  ;               new buffer space.                                        [23]
                                085C  ;                                                                        [23]
                                085C  ; OUTPUTS:                                                               [23]
                                085C  ;       @8(AP) - starting addr. of new buffer space                      [23]
                                085C  ;       R0 - 1 if memory space allocated                                 [23]
                                085C  ;            0 if insufficient free memory to allow allocation           [23]
                                085C  ;--
                                085C
                          000C  085C  .ENTRY  DSX$MAPDBGBLOCK, ^M<R2,R3>;                                      [23]
                                085E
                      1C    E0  085E          BBS     #DSA$V_USER,-           ;IF USER MODE                   [23]
          43 0000FE00'EF        0860                  DSA$GL_FLAGS,1000$      ;THEN JUST RETURN               [23]
             52    04 AC    D0  0866          MOVL    4(AP),R2                ;GET REQUESTED SIZE             [23]
       52   000001FF 8F    D3  086A          BITL    #^X1FF,R2               ;IF NOT A PAGE BOUNDARY         [23]
                      0E    13  0871          BEQL    10$                     ;THEN                           [23]
       52   000001FF 8F    CA  0873          BICL    #^X1FF,R2               ;   CLEAR PAGE OFFSET           [23]
       52   00000200 8F    C0  087A          ADDL2   #^X200,R2               ;   MAKE IT ONE PAGE BIGGER     [23]
  52   00000000'EF   52    C3  0881  10$:    SUBL3   R2,DEBUG_LO,R2          ;CALC. NEW LOW BUFFER LIMIT     [23]
       53   00000000'EF    D0  0889          MOVL    DEBUG_LO,R3             ;SET TOP OF NEW AREA            [23]
                      52    DD  0890          PUSHL   R2                      ;SAVE R2 FOR LATER              [23]
            00000914'EF    16  0892          Jsb     MAP_BLK                 ;CALL ROUTINE TO DO THE MAP     [27]
                      52 8ED0  0898          POPL    R2                      ;RESTORE LOW BUFFER LIMIT       [23]
                   0B 50    E9  089B          BLBC    R0,1000$                ;IF NO ERROR FROM MAP_BLK       [23]
       00000000'EF   52    D0  089E          MOVL    R2,DEBUG_LO             ;THEN SET NEW LOW LIMIT         [23]
             08 BC   52    D0  08A5          MOVL    R2,@8(AP)               ;PASS IT TO CALLER ALSO         [23]
                      04  08A9  1000$:  RET                                  ;RETURN (R0 EITHER SET OR CLEARED)[23]
                                08AA
```

```
                                08AA  1692 .SBTTL DSX$FREEDBGSYM - FREE MEMORY MAPPED TO DEBUGGER SYMTABS
                                08AA  1693
                                08AA  1694 ;++
                                08AA  1695 ; FUNCTIONAL DESCRIPTION                                                    [24]
                                08AA  1696 ;       This routine will deallocate the memory which has been              [24]
                                08AA  1697 ; assigned to the debugger's symbol tables.                                 [24]
                                08AA  1698 ;                                                                           [24]
                                08AA  1699 ; INPUTS                                                                    [24]
                                08AA  1700 ;       none                                                                [24]
                                08AA  1701 ;                                                                           [24]
                                08AA  1702 ; OUTPUTS                                                                   [24]
                                08AA  1703 ;       none                                                                [24]
                                08AA  1704 ;                                                                           [24]
                                08AA  1705 ; IMPLICIT INPUTS                                                           [24]
                                08AA  1706 ;       DEBUG_LO - Bottom of debugger's memory allocation, i. e.,           [24]
                                08AA  1707 ;                  bottom of symbol table                                   [24]
                                08AA  1708 ;       SYMTAB_HI- Top of symbol table                                      [24]
                                08AA  1709 ;--                                                                         [24]
                                08AA  1710
                         000C   08AA  1711 .ENTRY  DSX$FREEDBGSYM, ^M<R2,R3>       ;                                 [24]
                                08AC  1712
                      1C   E0   08AC  1713         BBS     #DSA$V_USER,-          ;IF USER MODE                     [24]
                1F 0000FE00'EF        08AE  1714                 DSA$GL_FLAGS,1000$     ;THEN JUST RETURN             [24]
              52  00000000'EF   D0   08B4  1715         MOVL    DEBUG_LO,R2            ;GET BOTTOM OF TABLES         [24]
              53  00000000'EF   D0   08BB  1716         MOVL    SYMTAB_HI,R3           ;GET TOP OF TABLES            [24]
                  0000096C'EF   16   08C2  1717         Jsb     UNMAP_BLK              ;DEALLOCATE THE AREA          [27]
00000000'EF       00000000'EF   D0   08C8  1718         MOVL    SYMTAB_HI,DEBUG_LO     ;ADJUST DEBUGGER LOW ADDRESS  [24]
                        04      08D3  1719 1000$:  RET                               ;THAT'S ALL                   [24]
                                08D4  1720
```

```
                        08D4  1722 .SBTTL MAP_DEBUGGER - MAP THE DEBUGGER'S MEMORY SPACE                    [23]
                        08D4  1723
                        08D4  1724 ;++                                                                      [23]
                        08D4  1725 ; FUNCTIONAL DESCRIPTION                                                 [23]
                        08D4  1726 ;       This routine will map all memory allocated to the debugger       [23]
                        08D4  1727 ; This includes memory space allocated both for the debugger's           [23]
                        08D4  1728 ; executable code and for the buffer space assigned for symbol table     [23]
                        08D4  1729 ; construction.                                                          [23]
                        08D4  1730 ;                                                                        [23]
                        08D4  1731 ; INPUTS                                                                 [23]
                        08D4  1732 ;       NONE                                                             [23]
                        08D4  1733 ;                                                                        [23]
                        08D4  1734 ; OUTPUTS                                                                [23]
                        08D4  1735 ;       R0 = 1 if space mapped successfully                              [23]
                        08D4  1736 ;          = 0 if space already mapped (this may not be an error)        [23]
                        08D4  1737 ;--
                        08D4  1738
                        08D4  1739 MAP_DEBUGGER::                       ;                                   [23]
              0C   BB   08D4  1740         PUSHR   #^M<R2,R3>           ;SAVE REGISTERS                     [23]
      52 00000000'EF   D0   08D6  1741         MOVL    DEBUG_LO,R2          ;GET LOW LIMIT OF ALLOCATION    [23]
      53 00000000'EF   D0   08DD  1742         MOVL    DEBUG_HI,R3          ;GET HIGH LIMIT OF ALLOCATION   [23]
      00000914'EF   16   08E4  1743         Jsb     MAP_BLK              ;DO THE MAP                     [27]
              0C   BA   08EA  1744         POPR    #^M<R2,R3>           ;RESTORE REGISTERS              [23]
                   05   08EC  1745         RSB                          ;RETURN                         [23]
```

```
                                08ED  .SBTTL UNMAP_DEBUGGER - UNMAP THE DEBUGGER'S MEMORY SPACE                [23]
                                08ED
                                08ED  ;++                                                                    [23]
                                08ED  ; FUNCTIONAL DESCRIPTION                                               [23]
                                08ED  ;       This routine will unmap the pages of memory previously         [23]
                                08ED  ; allocated to the debugger.                                           [23]
                                08ED  ;                                                                      [23]
                                08ED  ; INPUTS                                                               [23]
                                08ED  ;       NONE                                                           [23]
                                08ED  ;                                                                      [23]
                                08ED  ; OUTPUTS                                                              [23]
                                08ED  ;       NONE                                                           [23]
                                08ED  ;--                                                                    [23]
                                08ED
                                08ED  UNMAP_DEBUGGER::                          ;                            [23]
                  0C    BB      08ED          PUSHR   #^M<R2,R3>                ;SAVE REGISTERS              [23]
 52   00000000'EF    D0      08EF          MOVL    DEBUG_LO,R2               ;GET LOW LIMIT OF ALLOCATION   [23]
 52   000001FF 8F    CA      08F6          BICL2   #^X1FF,R2                 ;MAKE SURE IT'S A PAGE BOUNDARY [23]
 53   00000000'EF    D0      08FD          MOVL    DEBUG_HI,R3               ;GET HIGH LIMIT OF ALLOCATION  [23]
 53   000001FF 8F    CA      0904          BICL2   #^X1FF,R3                 ;MAKE SURE IT'S A PAGE BOUNDARY [23]
      0000096C'EF    16      090B          Jsb     UNMAP_BLK                 ;DO THE MAP                    [27]
                  0C    BA      0911          POPR    #^M<R2,R3>                ;RESTORE REGISTERS           [23]
                        05      0913          RSB                               ;RETURN                      [23]
                                0914
```

```
                     0914  1772 ; ROUTINE TO MAP A SPECIFIED BLOCK OF MEMORY.  IF THE BLOCK HAS ALREADY [23]
                     0914  1773 ; BEEN MAPPED, INDICATE ERROR RETURN.                                    [23]
                     0914  1774 ;                                                                        [23]
                     0914  1775 ; INPUTS                                                                [23]
                     0914  1776 ;       R2 = BOTTOM ADDR. OF NEW BUFFER                                 [23]
                     0914  1777 ;       R3 = TOP ADDR. OF NEW BUFFER                                    [23]
                     0914  1778 ;                                                                        [23]
                     0914  1779 ; OUTPUTS                                                               [23]
                     0914  1780 ;       R0 = 0 IF SPACE MAPPED SUCCESSFULLY                             [23]
                     0914  1781 ;          = 1 IF SPACE ALREADY ALLOCATED                               [23]
                     0914  1782 ;                                                                        [23]
                     0914  1783
                     0914  1784 MAP_BLK:
            54   DD  0914  1785         PUSHL   R4                      ;SAVE A REGISTER              [23]
      54    52   D0  0916  1786         MOVL    R2,R4                   ;SAVE IT FOR LATER            [23]
                     0919  1787                                         ;NOW TEST REQUESTED BUFFER AREA [23]
                     0919  1788                                         ;TO MAKE SURE IT HASN'T ALREADY [23]
                     0919  1789                                         ;BEEN ALLOCATED TO SOMEONE ELSE [23]
                     0919  1790 100$:                                   ;REPEAT                       [23]
      51    52   D0  0919  1791         MOVL    R2,R1                   ;   GET NEXT PAGE             [23]
      FEE4  CF   16  091C  1792         Jsb     RPTEADR                 ;   GET PTE OF PAGE           [27]
   43 61    1F   E0  0920  1793         BBS     #PTE$V_VALID,(R1),1000$ ;   IF PAGE ALLOCATED, LEAVE [23]
52 00000200 8F   C0  0924  1794         ADDL2   #512,R2                 ;   ADDR. OF NEXT PAGE        [23]
      53    52   D1  092B  1795         CMPL    R2,R3                   ;   SEE IF TOP OF NEW BUFFER AREA[23]
            E9   19  092E  1796         BLSS    100$                    ;UNTIL ENTIRE NEW BUFFER CHECKED[23]
                     0930  1797                                         ;IT'S OK TO ALLOCATE THE SPACE [23]
      52    54   D0  0930  1798         MOVL    R4,R2                   ;GET BOTTOM OF BUFFER         [23]
                     0933  1799 200$:                                   ;REPEAT                       [23]
      51    52   D0  0933  1800         MOVL    R2,R1                   ;   GET NEXT PAGE             [23]
      FECA  CF   16  0936  1801         Jsb     RPTEADR                 ;   GET PTE OF PAGE           [27]
   54 01    1F   78  093A  1802         ASHL    #PTE$V_VALID,#1,R4      ;   CREATE BIT MASK - SET VALID [23]
54 02    17   00  F0  093E  1803         INSV    #0,#PTE$V_OWN,#PTE$S_OWN,R4;SET OWNER BITS TO ZERO    [23]
 54 20000000 8F   C8  0943  1804         BISL2   #PTE$C_UW,R4            ;   SET USER-WRITABLE IN MASK [23]
 61 78000000 8F   CA  094A  1805         BICL2   #PTE$M_PROT,(R1)         ;   CLEAR PROTECTION IN PTE   [23]
      61    54   C8  0951  1806         BISL2   R4,(R1)                 ;   SET BITS IN PTE           [23]
52 00000200 8F   C0  0954  1807         ADDL2   #512,R2                 ;   ADDR. OF NEXT PAGE        [23]
      53    52   D1  095B  1808         CMPL    R2,R3                   ;   SEE IF TOP OF NEW BUFFER AREA[23]
            D3   19  095E  1809         BLSS    200$                    ;UNTIL ENTIRE NEW BUFFER MAPPED [23]
      50    01   D0  0960  1810         MOVL    #1,R0                   ;SET SUCCESS                  [23]
            54 8ED0  0963  1811 900$:   POPL    R4                      ;RESTORE A REGISTER           [23]
               05  0966  1812         RSB                             ;RETURN                       [23]
      50    00   D0  0967  1813 1000$:  MOVL    #0,R0                   ;SET ERROR                    [23]
            F7   11  096A  1814         BRB     900$                    ;RETURN                       [23]
                     096C  1815
                     096C  1816
                     096C  1817 ; ROUTINE TO UNMAP A SPECIFIED BLOCK OF MEMORY.                        [23]
                     096C  1818 ;                                                                      [23]
                     096C  1819 ; INPUTS:                                                              [23]
                     096C  1820 ;       R2 = BOTTOM ADDR. OF AREA TO UNMAP                             [23]
                     096C  1821 ;       R3 = TOP ADDR. OF AREA TO UNMAP                                [23]
                     096C  1822 ;                                                                      [23]
                     096C  1823 ; OUTPUTS:                                                             [23]
                     096C  1824 ;       NONE                                                           [23]
                     096C  1825 ;                                                                      [23]
                     096C  1826
                     096C  1827 UNMAP_BLK:                           ;                                 [23]
                     096C  1828 100$:                                ;REPEAT                           [23]
```

```
                 51    52   D0  096C  1829          MOVL    R2,R1                           ;   GET A PAGE                      [23]
                 FE91  CF   16  096F  1830          Jsb     RPTEADR                         ;   GET ADDR. OF PTE                [27]
          61     81800000 8F   CA  0973  1831          BICL2   #PTE$M_VALID+PTE$M_OWN,(R1);MAKE PAGE INVALID                [23]
          52     00000200 8F   C0  097A  1832          ADDL2   #512,R2                         ;   GET NEXT PAGE                [23]
                 53    52   D1  0981  1833          CMPL    R2,R3                           ;   SEE IF TOP OF AREA YET      [23]
                       E6   19  0984  1834          BLSS    100$                            ;UNTIL TOP OF AREA REACHED      [23]
                            05  0986  1835          RSB                                     ;RETURN                         [23]
                                0987  1836
                                0987  1837 ; Referenced by XDELTA
                                0987  1838 ;
                                0987  1839 INI$RDONLY::
                                0987  1840 INI$WRITABLE::
                            05  0987  1841          RSB
                                0988  1842
                                0988  1843          .END
```

```
$$ARGS                          =  00000005     D
$$N                             =  00000002     D
$$T1                            =  00000018     D
$ER                             =  00000002     D
$MODULE                            00000000 R   D   03
ACCESSABLE                         000002E6 R   D   04
A_P1BUFPTE                         00000004 R   D   02
A_P1BUFVA                          00000008 R   D   02
A_SPTEND                           0000000C R   D   02
BIT...                          =  00000015     D
BIT0                            =  00000001     D
BIT1                            =  00000002     D
BIT10                           =  00000400     D
BIT11                           =  00000800     D
BIT12                           =  00001000     D
BIT13                           =  00002000     D
BIT14                           =  00004000     D
BIT15                           =  00008000     D
BIT16                           =  00010000     D
BIT17                           =  00020000     D
BIT18                           =  00040000     D
BIT19                           =  00080000     D
BIT2                            =  00000004     D
BIT20                           =  00100000     D
BIT21                           =  00200000     D
BIT22                           =  00400000     D
BIT23                           =  00800000     D
BIT24                           =  01000000     D
BIT25                           =  02000000     D
BIT26                           =  04000000     D
BIT27                           =  08000000     D
BIT28                           =  10000000     D
BIT29                           =  20000000     D
BIT3                            =  00000008     D
BIT30                           =  40000000     D
BIT31                           =  80000000     D
BIT4                            =  00000010     D
BIT5                            =  00000020     D
BIT6                            =  00000040     D
BIT7                            =  00000080     D
BIT8                            =  00000100     D
BIT9                            =  00000200     D
CEP_FUNCTIONAL                  =  00000000     D
CEP_REPAIR                      =  00000001     D
CF$L_AP                            00000008     D
CF$L_FP                            0000000C     D
CF$L_ONCOND                        00000000     D
CF$L_PC                            00000010     D
CF$L_REG                           00000014     D
CF$W_MASK                          00000006     D
CF$W_PSW                           00000004     D
CMK$MMENABLE                       000006EF RG  D   04
CMK$SETPRT                         0000072E RG  D   04
CMK$_APBOOT                     =  00000004     D
CMK$_ASTEXIT                    =  00000000     D
CMK$_CANTIM                     =  0000000B     D
CMK$_HIBER                      =  00000007     D
CMK$_INITSCB                    =  00000008     D
CMK$_LAST                       =  0000000E     D
CMK$_MMENABLE                   =  00000003     D
CMK$_RCONSOLE                   =  00000001     D
CMK$_REFRESH                    =  00000005     D
CMK$_SCHDWK                     =  00000006     D
CMK$_SETIMR                     =  0000000C     D
CMK$_SETIPL                     =  00000009     D
CMK$_SETPRT                     =  0000000D     D
CMK$_SGIPR                      =  0000000A     D
CMK$_TCONSOLE                   =  00000002     D
CNTREG$_ACMODE                  =  0000000C     D
CNTREG$_NARGS                   =  00000004     D
CNTREG$_PAGCNT                  =  00000004     D
CNTREG$_REGION                  =  00000010     D
CNTREG$_RETADR                  =  00000008     D
CNTREG$_VA                      =  00000014     D
CRD$EXPREG                         0000042E RG  D   04
DEBUG_HI                           ********     X   00
DEBUG_LO                           ********     X   00
DS$AQ_SYSSRV                       ********     X   00
DS$A_PRGBGN                        ********     X   00
DS$GA_LASTADR                      ********     X   00
DS$GB_MM_ENB                       0000001C RG  D   02
DS$GB_SYSTYPE                      ********     X   00
DS$GL_ACTUAL_MEMSIZE               00000014 RG  D   02
DS$GL_BUFCNT                       ********     X   00
DS$GL_FLAGS                        ********     X   00
DS$GL_MEMSIZE                      00000010 RG  D   02
DS$GL_SET_MEMSIZE                  00000018 RG  D   02
DS$GQ_PHYADR                       ********     X   00
DS$K_ERROR                      =  00000002     D
DS$K_NORMAL                     =  00000001     D
DS$K_PRGSIZ                        ********     X   00
DS$K_PRINTB                     =  00000002     D
DS$K_PRINTF                     =  00000001     D
DS$K_PRINTI                     =  00000000     D
DS$K_PRINTX                     =  00000003     D
DS$K_SEVERE                     =  00000004     D
DS$K_SUBSYS                     =  00000066     D
DS$K_TYPE_ABORT_PROGRAM         =  00000014     D
DS$K_TYPE_ABORT_TEST            =  00000013     D
DS$K_TYPE_COMMAND_ERR           =  00000015     D
DS$K_TYPE_COMMAND_OUT           =  00000016     D
DS$K_TYPE_CRD_AUTOTEST          =  0000001A     D
DS$K_TYPE_DS_PROMPT             =  00000001     D
DS$K_TYPE_DS_START              =  0000001D     D
DS$K_TYPE_ERRDEV                =  00000008     D
DS$K_TYPE_ERRHARD               =  00000006     D
DS$K_TYPE_ERROR_BODY            =  00000009     D
DS$K_TYPE_ERROR_END             =  0000000A     D
DS$K_TYPE_ERRPREP               =  0000001B     D
DS$K_TYPE_ERRSOFT               =  00000007     D
DS$K_TYPE_ERRSUP                =  00000004     D
DS$K_TYPE_ERRSYS                =  00000005     D
DS$K_TYPE_ERR_HALT              =  0000000D     D
DS$K_TYPE_EXCEPTION             =  0000000C     D
```

```
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D        DS$V_CMDFLG                     = 00000007    D
DS$K_TYPE_FIRST_PASS            = 00000011    D        DS$V_CTRLC                      = 00000000    D
DS$K_TYPE_GENERAL               = 00000000    D        DS$V_CTRLO                      = 00000010    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D        DS$V_DEVFLG                     = 00000009    D
DS$K_TYPE_NO_TESTS              = 00000012    D        DS$V_DISABLCC                   = 00000018    D
DS$K_TYPE_PARAM_ERROR           = 0000001C    D        DS$V_DONFLG                     = 0000000D    D
DS$K_TYPE_PROGRAM_END           = 00000010    D        DS$V_ERRFLG                     = 00000004    D
DS$K_TYPE_PROGRAM_INFO          = 00000017    D        DS$V_EXCEPT                     = 00000013    D
DS$K_TYPE_PROGRAM_START         = 0000000F    D        DS$V_EXETST                     = 00000012    D
DS$K_TYPE_QIO_INVADP            = 00000024    D        DS$V_HLTFLG                     = 00000003    D
DS$K_TYPE_QIO_NODRIVER          = 00000022    D        DS$V_LODFLG                     = 00000001    D
DS$K_TYPE_QIO_WRONGVER          = 00000023    D        DS$V_MEMMGT                     = 0000000F    D
DS$K_TYPE_SCRIPT_ECHO           = 00000021    D        DS$V_NI                         = 0000001A    D
DS$K_TYPE_SCRIPT_PNF            = 0000001E    D        DS$V_OUTPUT                     = 00000017    D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020    D        DS$V_RUBFLG                     = 00000005    D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F    D        DS$V_SCRIPT                     = 00000015    D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D        DS$V_SETIMR                     = 00000019    D
DS$K_TYPE_START_ERR             = 00000018    D        DS$V_STRFLG                     = 00000002    D
DS$K_TYPE_START_LIST            = 00000025    D        DS$V_SUBT                       = 0000000E    D
DS$K_TYPE_SUMMARY               = 0000000E    D        DS$V_SYSFLG                     = 0000000A    D
DS$K_TYPE_USER_PROMPT           = 00000002    D        DS$V_TIMED                      = 00000019    D
DS$K_WARNING                    = 00000000    D        DS$V_TIMED_OUT                  = 0000001B    D
DS$M_ABRTFLG                    = 00000040    D        DS$V_TIMRON                     = 00000011    D
DS$M_BADTIME                    = 00100000    D        DS$_AP_NORMAL_BREAK             = 00660150    D
DS$M_BATCH                      = 00400000    D        DS$_ARITH                       = 006600D0    D
DS$M_BRKCLR                     = 00001000    D        DS$_ASBE                        = 00660118    D
DS$M_BRKPT                      = 00000800    D        DS$_BADLINK                     = 006600F0    D
DS$M_CHARFLG                    = 00000100    D        DS$_BADTYPE                     = 006600E8    D
DS$M_CMDFLG                     = 00000080    D        DS$_BIIC                        = 00660120    D
DS$M_CTRLC                      = 00000001    D        DS$_CHME                        = 006600A8    D
DS$M_CTRLO                      = 00010000    D        DS$_CHMK                        = 006600E0    D
DS$M_DEVFLG                     = 00000200    D        DS$_DEVNAME                     = 00660108    D
DS$M_DISABLCC                   = 01000000    D        DS$_ERROR                       = 00660002    D
DS$M_DONFLG                     = 00002000    D        DS$_FHWE                        = 00660068    D
DS$M_ERRFLG                     = 00000010    D        DS$_FRAGBUF                     = 00660080    D
DS$M_EXCEPT                     = 00080000    D        DS$_ICBUSY                      = 006600C8    D
DS$M_EXETST                     = 00040000    D        DS$_ICERR                       = 006600C0    D
DS$M_HLTFLG                     = 00000008    D        DS$_IHWE                        = 00660060    D
DS$M_LODFLG                     = 00000002    D        DS$_ILLCHAR                     = 00660018    D
DS$M_MEMMGT                     = 00008000    D        DS$_ILLPAGCNT                   = 00660078    D
DS$M_NI                         = 04000000    D        DS$_ILLUNIT                     = 00660100    D
DS$M_OUTPUT                     = 00800000    D        DS$_INIT_FAIL                   = 00660140    D
DS$M_RUBFLG                     = 00000020    D        DS$_INSFMEM                     = 00660050    D
DS$M_SCRIPT                     = 00200000    D        DS$_INVCPU                      = 00660130    D
DS$M_SETIMR                     = 02000000    D        DS$_IPL2HI                      = 006600B8    D
DS$M_STRFLG                     = 00000004    D        DS$_IVADDR                      = 00660040    D
DS$M_SUBT                       = 00004000    D        DS$_IVVECT                      = 00660038    D
DS$M_SYSFLG                     = 00000400    D        DS$_KRNLSTK                     = 00660090    D
DS$M_TIMED                      = 02000000    D        DS$_LOGIC                       = 00660070    D
DS$M_TIMED_OUT                  = 08000000    D        DS$_MCHK                        = 00660088    D
DS$M_TIMRON                     = 00020000    D        DS$_MEM_ALLOC_ERR               = 00660138    D
DS$V_ABRTFLG                    = 00000006    D        DS$_MMOFF                       = 00660058    D
DS$V_BADTIME                    = 00000014    D        DS$_NEEDUNIT                    = 006600F8    D
DS$V_BATCH                      = 00000016    D        DS$_NODE                        = 00660128    D
DS$V_BRKCLR                     = 0000000C    D        DS$_NOPCS                       = 00660110    D
DS$V_BRKPT                      = 0000000B    D        DS$_NORMAL                      = 00660001    D
DS$V_CHARFLG                    = 00000008    D        DS$_NOSUPPORT                   = 006600B1    D
```

```
DS$_NOTALLOWMP                = 00660148     D
DS$_NOTDON                    = 00660030     D
DS$_NOTIMP                    = 006600B0     D
DS$_NULLSTR                   = 00660010     D
DS$_OVERFLOW                  = 00660008     D
DS$_POWER                     = 00660098     D
DS$_PROGERR                   = 00660020     D
DS$_SEVERE                    = 00660004     D
DS$_TRANSL                    = 006600A0     D
DS$_TRUNCATE                  = 00660028     D
DS$_UNEXPINT                  = 006600D8     D
DS$_UNSUPPDEV                 = 00660158     D
DS$_VASFULL                   = 00660048     D
DS$_WARNING                   = 00660000     D
DSA$AL_APTMAIL                  0000FE00     D
DSA$AT_APTTXT                   0000FA00     D
DSA$GL_APTCOM                   0000FE04     D
DSA$GL_DEVLEN                   0000FE58     D
DSA$GL_ERRNO                    0000FE44     D
DSA$GL_EVENT                    0000FE48     D
DSA$GL_FLAGS                    0000FE00     D
DSA$GL_MSGTYP                   0000FE40     D
DSA$GL_PASSES                   0000FE08     D
DSA$GL_PASSNO                   0000FE54     D
DSA$GL_SECTNO                   0000FE10     D
DSA$GL_SID                      0000FE14     D
DSA$GL_SUBTNO                   0000FE4C     D
DSA$GL_TESTNO                   0000FE50     D
DSA$GL_UNITS                    0000FE0C     D
DSA$GQ_MSGPTR                   0000FE68     D
DSA$GT_DEVNAM                   0000FE5C     D
DSA$V_USER                    = 0000001C     D
DSR$CHECKLOAD                   ********   X   00
DSR$MMENABLE                    000006DC RG  D   04
DSR$SETIMR_INI                  ********   X   00
DSX$FREEDBGSYM                  000008AA RG  D   04
DSX$MAPDBGBLOCK                 0000085C RG  D   04
DSX$MMOFF                       000006B6 RG  D   04
DSX$MMOFF_X                     000006DB R   D   04
DSX$MMON                        0000069C RG  D   04
DSX$MMON_X                      000006B5 R   D   04
DSX$PRINT                       ********   X   00
DS_ERRSUP                       ********   X   00
ENV$M_DOMAIN                  = 00000002     D
ENV$M_LEVEL                   = 00000001     D
ENV$M_SUPER                   = 000003FC     D
ENV$S_DOMAIN                  = 00000001     D
ENV$S_LEVEL                   = 00000001     D
ENV$S_SUPER                   = 00000008     D
ENV$V_DOMAIN                  = 00000001     D
ENV$V_LEVEL                   = 00000000     D
ENV$V_SUPER                   = 00000002     D
ENV$_CPU                      = 00000000     D
ENV$_FUNCTIONAL               = 00000000     D
ENV$_REPAIR                   = 00000001     D
ENV$_SUPER                    = 00000001     D
ENV$_SYSTEM                   = 00000001     D
ESTKPTR                         ********   X   00
EXE$CNTREG                      00000566 RG  D   04
EXE$CNTREG_X                    0000069B R   D   04
EXE$EXPREG                      0000044B RG  D   04
EXE$EXPREG_X                    00000565 R   D   04
EXE$SETPRT                      00000719 RG  D   04
EXPREG$_ACMODE                = 0000000C     D
EXPREG$_NARGS                 = 00000004     D
EXPREG$_PAGCNT                = 00000004     D
EXPREG$_PTE_OWNER_CODE        = 00000014     D
EXPREG$_REGION                = 00000010     D
EXPREG$_RETADR                = 00000008     D
EXPREG_COMMON                   0000045B R   D   04
GETBUF$_NARGS                 = 00000004     D
GETBUF$_PAGCNT                = 00000004     D
GETBUF$_PHYADR                = 0000000C     D
GETBUF$_REGION                = 00000010     D
GETBUF$_RETADR                = 00000008     D
GETMEM$_NARGS                 = 00000003     D
GETMEM$_PAGCNT                = 00000004     D
GETMEM$_PHYADR                = 00000008     D
GETMEM$_VRTADR                = 0000000C     D
INI$RDONLY                      00000987 RG  D   04
INI$WRITABLE                    00000987 RG  D   04
IO$GQ_PHYSICAL                  ********   X   00
IRP$B_CARCON                    00000038     D
IRP$B_EFN                       00000022     D
IRP$B_PRI                       00000023     D
IRP$B_RMOD                      0000000B     D
IRP$B_TYPE                      0000000A     D
IRP$C_LENGTH                    0000005C     D
IRP$K_LENGTH                    0000005C     D
IRP$L_ARB                       00000050     D
IRP$L_AST                       00000010     D
IRP$L_ASTPRM                    00000014     D
IRP$L_DIAGBUF                   00000044     D
IRP$L_EXTEND                    0000004C     D
IRP$L_IOQBL                     00000004     D
IRP$L_IOQFL                     00000000     D
IRP$L_IOSB                      00000024     D
IRP$L_IOST1                     00000034     D
IRP$L_IOST2                     00000038     D
IRP$L_MEDIA                     00000034     D
IRP$L_PID                       0000000C     D
IRP$L_SEGVBN                    00000040     D
IRP$L_SEQNUM                    00000048     D
IRP$L_SVAPTE                    0000002C     D
IRP$L_TT_TERM                   00000038     D
IRP$L_UCB                       0000001C     D
IRP$L_WIND                      00000018     D
IRP$Q_NT_PRVMSK                 0000003C     D
IRP$W_ABCNT                     0000003C     D
IRP$W_BCNT                      00000032     D
IRP$W_BOFF                      00000030     D
IRP$W_CHAN                      00000028     D
IRP$W_FUNC                      00000020     D
IRP$W_OBCNT                     0000003E     D
```

```
IRP$W_SIZE                      00000008    D
IRP$W_STS                       0000002A    D
IRP$W_TT_PRMPT                  0000003C    D
ISTKPTR                         ********    X   00
KSTKPTR                         ********    X   00
K_P1PAGES                     = 0000007E    D
L$A_CCP                         00000240    D
L$A_DEVP                        0000021C    D
L$A_DREG                        00000224    D
L$A_DTP                         00000218    D
L$A_ICP                         0000023C    D
L$A_LASTADR                     00000214    D
L$A_NAME                        00000208    D
L$A_REPP                        00000244    D
L$A_SECNAM                      00000250    D
L$A_STATAB                      00000248    D
L$A_TSTCNT                      00000254    D
L$L_ENVIRON                     00000204    D
L$L_ERRTYP                      0000024C    D
L$L_HEADLENGTH                  00000200    D
L$L_REV                         0000020C    D
L$L_UNIT                        00000220    D
L$L_UNUSED                      00000228    D
L$L_UPDATE                      00000210    D
L_P0LEN                         00000000 R  D   02
MAP$IOSPACE                     ********    X   00
MAPFREE                         00000392 RG D   04
MAPMEM                          00000000 RG D   04
MAPMEM$IOSPACE                  0000025A RG D   04
MAP_BLK                         00000914 R  D   04
MAP_DEBUGGER                    000008D4 RG D   04
MEMPOOL$INIT                    ********      X 00
MMG$GL_SPTBASE                  ********      X 00
MMG$IOLOCK                      0000084E RG D   04
MMG$UNLOCK                      00000858 RG D   04
MP$GR_BOOTED_PROCESSORS         ********      X 00
P0PT_BASE                       ********      X 00
PFN$AB_STATE                  = 00000010 RG D   02
PFN$AB_TYPE                   = 00000000  G D
PFN$AL_BAK                    = 00000008 RG D   02
PFN$AL_PTE                    = 00000004 RG D   02
PFN$AW_BLINK                  = 00000000  G D
PFN$AW_FLINK                  = 00000000  G D
PFN$AW_REFCNT                 = 0000000C RG D   02
PFN$AW_SWPVBN                 = 00000000 RG D   02
PONE                            000007A0 R  D   04
PPA_CONSRV4                   = 20040010    D
PR$_IPL                       = 00000012    D
PR$_MAPEN                     = 00000038    D
PR$_P0BR                      = 00000008    D
PR$_P0LR                      = 00000009    D
PR$_P1BR                      = 0000000A    D
PR$_P1LR                      = 0000000B    D
PR$_SBR                       = 0000000C    D
PR$_SID_TYPRTUV2                ********    X   00
PR$_SLR                       = 0000000D    D
PR$_SYS_TYP900                  ********    X   00
PR$_TBIA                      = 00000039    D
PR$_TBIS                      = 0000003A    D
PSL$V_IS                      = 0000001A    D
PTE$C_NA                      = 00000000    D
PTE$C_UREW                    = 68000000    D
PTE$C_URKW                    = 70000000    D
PTE$C_URSW                    = 60000000    D
PTE$C_UW                      = 20000000    D
PTE$M_IO                      = 00100000    D
PTE$M_MODIFY                  = 04000000    D
PTE$M_OWN                     = 01800000    D
PTE$M_PFN                     = 001FFFFF    D
PTE$M_PROT                    = 78000000    D
PTE$M_VALID                   = 80000000    D
PTE$S_OWN                     = 00000002    D
PTE$V_IO                      = 00000014    D
PTE$V_OWN                     = 00000017    D
PTE$V_VALID                   = 0000001F    D
PZERO                           0000078A R  D   04
RELBUF$_NARGS                 = 00000003    D
RELBUF$_PAGCNT                = 00000004    D
RELBUF$_REGION                = 0000000C    D
RELBUF$_RETADR                = 00000008    D
RELMEM$_NARGS                 = 00000003    D
RELMEM$_PAGCNT                = 00000004    D
RELMEM$_PHYADR                = 00000008    D
RELMEM$_VRTADR                = 0000000C    D
RESERVD                         000007CC R  D   04
RPTEADR                         00000804 RG D   04
RTVAX                           000002BC R  D   04
SCB$L_ACCESS                    00000020      D
SCB$L_ARITH                     00000034      D
SCB$L_BREAK                     0000002C      D
SCB$L_CHME                      00000044      D
SCB$L_CHMK                      00000040      D
SCB$L_CHMS                      00000048      D
SCB$L_CHMU                      0000004C      D
SCB$L_COMPAT                    00000030      D
SCB$L_KNLSTK                    00000008      D
SCB$L_MACHCHK                   00000004      D
SCB$L_OPCCUS                    00000014      D
SCB$L_OPCDEC                    00000010      D
SCB$L_POWER                     0000000C      D
SCB$L_RADRMOD                   0000001C      D
SCB$L_ROPRAND                   00000018      D
SCB$L_RXDB                      000000F8      D
SCB$L_SFTLVL1                   00000084      D
SCB$L_SFTLVL10                  000000A8      D
SCB$L_SFTLVL11                  000000AC      D
SCB$L_SFTLVL12                  000000B0      D
SCB$L_SFTLVL13                  000000B4      D
SCB$L_SFTLVL14                  000000B8      D
SCB$L_SFTLVL15                  000000BC      D
SCB$L_SFTLVL2                   00000088      D
SCB$L_SFTLVL3                   0000008C      D
SCB$L_SFTLVL4                   00000090      D
SCB$L_SFTLVL5                   00000094      D
```

```
SCB$L_SFTLVL6                     00000098     D
SCB$L_SFTLVL7                     0000009C     D
SCB$L_SFTLVL8                     000000A0     D
SCB$L_SFTLVL9                     000000A4     D
SCB$L_TBIT                        00000028     D
SCB$L_TIMER                       000000C0     D
SCB$L_TRANSL                      00000024     D
SCB$L_TXDB                        000000FC     D
SCB$L_VM                          00000038     D
SCB$L_ZERO                        00000000     D
SCB_UNKINT                        ********   X   00
SEP_FUNCTIONAL                  = 00000002     D
SEP_REPAIR                      = 00000003     D
SET                               000007D0 R   D   04
SETPRT$_ACMODE                  = 0000000C     D
SETPRT$_INADR                   = 00000004     D
SETPRT$_NARGS                   = 00000005     D
SETPRT$_PROT                    = 00000010     D
SETPRT$_PRVPRT                  = 00000014     D
SETPRT$_RETADR                  = 00000008     D
SETPRT_XIT                        00000789 R   D   04
SIZ...                          = 00000001     D
SS$_ACCVIO                      = 0000000C     D
SS$_ILLPAGCNT                   = 000000FC     D
SS$_LENVIO                      = 0000018C     D
SS$_NORMAL                      = 00000001     D
SS$_PAGOWNVIO                   = 000001EC     D
SS$_RESIGNAL                    = 00000918     D
SS$_VASFULL                     = 00000244     D
SSTKPTR                           ********   X   00
STACKPROT                         00000321 R   D   04
STACK_BASE                        ********   X   00
SYMTAB_HI                         ********   X   00
SYS$UNWIND                        ********   X   00
SYSTEM                            000007B6 R   D   04
T_INSFMEM                         00000007 R   D   03
UNMAP_BLK                         0000096C R   D   04
UNMAP_DEBUGGER                    000008ED RG  D   04
USTKPTR                           ********   X   00
XDS$INIT                        = 00000000   G D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000001D (29.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	0000003E (62.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000988 (2440.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                  VALUE       DEFINITION      REFERENCES...
------                  -----       ----------      -------------
$$ARGS                  =00000005   257    (1)      235    (1)      241    (1)      242    (1)      244    (1)
                                                    245    (1)      249    (1)      257    (1)
$$N                     =00000002   457    (1)      457    (1)
$$T1                    =00000018   257    (1)      235    (1)      241    (1)      242    (1)      244    (1)
                                                    245    (1)      249    (1)      257    (1)
$ER                     =00000002   1214   (11)   #-1214   (11)
$MODULE                  00000000-R 327    (1)      1214   (11)
ACCESSABLE               000002E6-R 696    (3)      400    (1)      415    (1)
A_P1BUFPTE               00000004-R 287    (1)    #-1160   (10)     305    (1)    #-579    (1)    #-844    (6)
                                                  #-990    (8)
A_P1BUFVA                00000008-R 288    (1)    #-1162   (10)     306    (1)    #-570    (1)    #-577    (1)
                                                  #-992    (8)
A_SPTEND                 0000000C-R 289    (1)    #-1165   (10)     307    (1)    #-542    (1)    #-859    (6)
                                                  #-996    (8)
BIT...                  =00000015   256    (1)      232    (1)      233    (1)      239    (1)      240    (1)
                                                    247    (1)      256    (1)
BIT0                    =00000001   236    (1)
BIT1                    =00000002   236    (1)
BIT10                   =00000400   236    (1)
BIT11                   =00000800   236    (1)
BIT12                   =00001000   236    (1)
BIT13                   =00002000   236    (1)
BIT14                   =00004000   236    (1)
BIT15                   =00008000   236    (1)
BIT16                   =00010000   236    (1)
BIT17                   =00020000   236    (1)
BIT18                   =00040000   236    (1)
BIT19                   =00080000   236    (1)
BIT2                    =00000004   236    (1)
BIT20                   =00100000   236    (1)
BIT21                   =00200000   236    (1)
BIT22                   =00400000   236    (1)
BIT23                   =00800000   236    (1)
BIT24                   =01000000   236    (1)
BIT25                   =02000000   236    (1)
BIT26                   =04000000   236    (1)
BIT27                   =08000000   236    (1)
BIT28                   =10000000   236    (1)
BIT29                   =20000000   236    (1)
BIT3                    =00000008   236    (1)
BIT30                   =40000000   236    (1)
BIT31                   =80000000   236    (1)
BIT4                    =00000010   236    (1)
BIT5                    =00000020   236    (1)
BIT6                    =00000040   236    (1)
BIT7                    =00000080   236    (1)
BIT8                    =00000100   236    (1)
BIT9                    =00000200   236    (1)
CEP_FUNCTIONAL          =00000000   240    (1)
CEP_REPAIR              =00000001   240    (1)
```

```
CMK$MMENABLE                    000006EF-R   1392  (14)     1389  (14)
CMK$SETPRT                      0000072E-R   1453  (16)     1450  (16)
CMK$_APBOOT                    =00000004     232   (1)
CMK$_ASTEXIT                   =00000000     232   (1)
CMK$_CANTIM                    =0000000B     232   (1)
CMK$_HIBER                     =00000007     232   (1)
CMK$_INITSCB                   =00000008     232   (1)
CMK$_LAST                      =0000000E     232   (1)
CMK$_MMENABLE                  =00000003     232   (1)    #-1389  (14)
CMK$_RCONSOLE                  =00000001     232   (1)
CMK$_REFRESH                   =00000005     232   (1)
CMK$_SCHDWK                    =00000006     232   (1)
CMK$_SETIMR                    =0000000C     232   (1)
CMK$_SETIPL                    =00000009     232   (1)
CMK$_SETPRT                    =0000000D     232   (1)    #-1450  (16)
CMK$_SGIPR                     =0000000A     232   (1)
CMK$_TCONSOLE                  =00000002     232   (1)
CNTREG$_ACMODE                 =0000000C     235   (1)
CNTREG$_NARGS                  =00000004     235   (1)
CNTREG$_PAGCNT                 =00000004     235   (1)    #-1148  (10)   #-1226  (11)
CNTREG$_REGION                 =00000010     235   (1)    #-1117  (10)   #-1124  (10)   #-1178  (11)
CNTREG$_RETADR                 =00000008     235   (1)    #-1241  (11)   #-1245  (11)
CNTREG$_VA                     =00000014     319   (1)    #-1145  (10)   #-1149  (10)   #-1197  (11)
CRD$EXPREG                      0000042E-R   948   (8)
DEBUG_HI                        00000000-XR                #-1742  (22)   #-1765  (23)     270   (1)
DEBUG_LO                        00000000-XR                #-1681  (20)   #-1682  (20)   #-1687  (20)   #-1715  (21)
                                                           #-1718  (21)   #-1741  (22)   #-1763  (23)     270   (1)
DS$AQ_SYSSRV                    00000000-XR                  264   (1)    #-494   (1)
DS$A_PRGBGN                     00000000-XR                  269   (1)     816   (6)
DS$GA_LASTADR                   00000000-XR                  264   (1)    #-581   (1)   #-829   (6)   #-950   (8)
DS$GB_MM_ENB                    0000001C-R   298   (1)    #-1346  (13)
DS$GB_SYSTYPE                   00000000-XR                  275   (1)    #-385   (1)
DS$GL_ACTUAL_MEMSIZE            00000014-R   292   (1)    #-435   (1)
DS$GL_BUFCNT                    00000000-XR                  265   (1)     860   (6)
DS$GL_FLAGS                     00000000-XR                  269   (1)
DS$GL_MEMSIZE                   00000010-R   290   (1)     310   (1)    #-449   (1)   #-504   (1)
DS$GL_SET_MEMSIZE               00000018-R   295   (1)    #-391   (1)    #-436   (1)
DS$GQ_PHYADR                    00000000-XR                #-1028   (8)   #-1029   (8)   #-1031   (8)     266   (1)
                                                           #-976   (8)
DS$K_ERROR                     =00000002     239   (1)
DS$K_NORMAL                    =00000001     239   (1)
DS$K_PRGSIZ                     00000000-XR                  269   (1)     816   (6)
DS$K_PRINTB                    =00000002     247   (1)
DS$K_PRINTF                    =00000001     247   (1)    #-457   (1)
DS$K_PRINTI                    =00000000     247   (1)
DS$K_PRINTX                    =00000003     247   (1)
DS$K_SEVERE                    =00000004     239   (1)
DS$K_SUBSYS                    =00000066     239   (1)    239   (1)
DS$K_TYPE_ABORT_PROGRAM        =00000014     247   (1)
DS$K_TYPE_ABORT_TEST           =00000013     247   (1)
DS$K_TYPE_COMMAND_ERR          =00000015     247   (1)
DS$K_TYPE_COMMAND_OUT          =00000016     247   (1)
DS$K_TYPE_CRD_AUTOTEST         =0000001A     247   (1)
DS$K_TYPE_DS_PROMPT            =00000001     247   (1)
DS$K_TYPE_DS_START             =0000001D     247   (1)
DS$K_TYPE_ERRDEV               =00000008     247   (1)
DS$K_TYPE_ERRHARD              =00000006     247   (1)
```

```
DS$K_TYPE_ERROR_BODY          =00000009     247    (1)
DS$K_TYPE_ERROR_END           =0000000A     247    (1)
DS$K_TYPE_ERRPREP             =0000001B     247    (1)
DS$K_TYPE_ERRSOFT             =00000007     247    (1)
DS$K_TYPE_ERRSUP              =00000004     247    (1)
DS$K_TYPE_ERRSYS              =00000005     247    (1)
DS$K_TYPE_ERR_HALT            =0000000D     247    (1)
DS$K_TYPE_EXCEPTION           =0000000C     247    (1)
DS$K_TYPE_EXCEPTION_HEAD      =0000000B     247    (1)
DS$K_TYPE_FIRST_PASS          =00000011     247    (1)
DS$K_TYPE_GENERAL             =00000000     247    (1)
DS$K_TYPE_GENERAL_ERROR       =00000003     247    (1)    #-457    (1)
DS$K_TYPE_NO_TESTS            =00000012     247    (1)
DS$K_TYPE_PARAM_ERROR         =0000001C     247    (1)
DS$K_TYPE_PROGRAM_END         =00000010     247    (1)
DS$K_TYPE_PROGRAM_INFO        =00000017     247    (1)
DS$K_TYPE_PROGRAM_START       =0000000F     247    (1)
DS$K_TYPE_QIO_INVADP          =00000024     247    (1)
DS$K_TYPE_QIO_NODRIVER        =00000022     247    (1)
DS$K_TYPE_QIO_WRONGVER        =00000023     247    (1)
DS$K_TYPE_SCRIPT_ECHO         =00000021     247    (1)
DS$K_TYPE_SCRIPT_PNF          =0000001E     247    (1)
DS$K_TYPE_SCRIPT_PROMPT       =00000020     247    (1)
DS$K_TYPE_SCRIPT_SKIP         =0000001F     247    (1)
DS$K_TYPE_SEQUENCE_ERROR      =00000019     247    (1)
DS$K_TYPE_START_ERR           =00000018     247    (1)
DS$K_TYPE_START_LIST          =00000025     247    (1)
DS$K_TYPE_SUMMARY             =0000000E     247    (1)
DS$K_TYPE_USER_PROMPT         =00000002     247    (1)
DS$K_WARNING                  =00000000     239    (1)
DS$M_ABRTFLG                  =00000040     233    (1)
DS$M_BADTIME                  =00100000     233    (1)
DS$M_BATCH                    =00400000     233    (1)
DS$M_BRKCLR                   =00001000     233    (1)
DS$M_BRKPT                    =00000800     233    (1)
DS$M_CHARFLG                  =00000100     233    (1)
DS$M_CMDFLG                   =00000080     233    (1)
DS$M_CTRLC                    =00000001     233    (1)
DS$M_CTRLO                    =00010000     233    (1)
DS$M_DEVFLG                   =00000200     233    (1)
DS$M_DISABLCC                 =01000000     233    (1)
DS$M_DONFLG                   =00002000     233    (1)
DS$M_ERRFLG                   =00000010     233    (1)
DS$M_EXCEPT                   =00080000     233    (1)
DS$M_EXETST                   =00040000     233    (1)
DS$M_HLTFLG                   =00000008     233    (1)
DS$M_LODFLG                   =00000002     233    (1)
DS$M_MEMMGT                   =00008000     233    (1)
DS$M_NI                       =04000000     233    (1)
DS$M_OUTPUT                   =00800000     233    (1)
DS$M_RUBFLG                   =00000020     233    (1)
DS$M_SCRIPT                   =00200000     233    (1)
DS$M_SETIMR                   =02000000     233    (1)
DS$M_STRFLG                   =00000004     233    (1)
DS$M_SUBT                     =00004000     233    (1)
DS$M_SYSFLG                   =00000400     233    (1)
DS$M_TIMED                    =02000000     233    (1)
```

```
DS$M_TIMED_OUT                  =08000000     233     (1)
DS$M_TIMRON                     =00020000     233     (1)
DS$V_ABRTFLG                    =00000006     233     (1)
DS$V_BADTIME                    =00000014     233     (1)
DS$V_BATCH                      =00000016     233     (1)
DS$V_BRKCLR                     =0000000C     233     (1)
DS$V_BRKPT                      =0000000B     233     (1)
DS$V_CHARFLG                    =00000008     233     (1)
DS$V_CMDFLG                     =00000007     233     (1)
DS$V_CTRLC                      =00000000     233     (1)
DS$V_CTRLO                      =00000010     233     (1)
DS$V_DEVFLG                     =00000009     233     (1)
DS$V_DISABLCC                   =00000018     233     (1)
DS$V_DONFLG                     =0000000D     233     (1)
DS$V_ERRFLG                     =00000004     233     (1)
DS$V_EXCEPT                     =00000013     233     (1)
DS$V_EXETST                     =00000012     233     (1)
DS$V_HLTFLG                     =00000003     233     (1)
DS$V_LODFLG                     =00000001     233     (1)
DS$V_MEMMGT                     =0000000F     233     (1)
DS$V_NI                         =0000001A     233     (1)
DS$V_OUTPUT                     =00000017     233     (1)
DS$V_RUBFLG                     =00000005     233     (1)
DS$V_SCRIPT                     =00000015     233     (1)
DS$V_SETIMR                     =00000019     233     (1)
DS$V_STRFLG                     =00000002     233     (1)
DS$V_SUBT                       =0000000E     233     (1)
DS$V_SYSFLG                     =0000000A     233     (1)
DS$V_TIMED                      =00000019     233     (1)
DS$V_TIMED_OUT                  =0000001B     233     (1)
DS$V_TIMRON                     =00000011     233     (1)
DS$_AP_NORMAL_BREAK             =00660150     239     (1)
DS$_ARITH                       =006600D0     239     (1)
DS$_ASBE                        =00660118     239     (1)
DS$_BADLINK                     =006600F0     239     (1)
DS$_BADTYPE                     =006600E8     239     (1)
DS$_BIIC                        =00660120     239     (1)
DS$_CHME                        =006600A8     239     (1)
DS$_CHMK                        =006600E0     239     (1)
DS$_DEVNAME                     =00660108     239     (1)
DS$_ERROR                       =00660002     239     (1)
DS$_FHWE                        =00660068     239     (1)
DS$_FRAGBUF                     =00660080     239     (1)     #-1231    (11)
DS$_ICBUSY                      =006600C8     239     (1)
DS$_ICERR                       =006600C0     239     (1)
DS$_IHWE                        =00660060     239     (1)
DS$_ILLCHAR                     =00660018     239     (1)
DS$_ILLPAGCNT                   =00660078     239     (1)
DS$_ILLUNIT                     =00660100     239     (1)
DS$_INIT_FAIL                   =00660140     239     (1)
DS$_INSFMEM                     =00660050     239     (1)
DS$_INVCPU                      =00660130     239     (1)
DS$_IPL2HI                      =006600B8     239     (1)
DS$_IVADDR                      =00660040     239     (1)
DS$_IVVECT                      =00660038     239     (1)
DS$_KRNLSTK                     =00660090     239     (1)
DS$_LOGIC                       =00660070     239     (1)
```

```
DS$_MCHK                =00660088      239     (1)     #-707     (3)
DS$_MEM_ALLOC_ERR       =00660138      239     (1)
DS$_MMOFF               =00660058      239     (1)
DS$_NEEDUNIT            =006600F8      239     (1)
DS$_NODE                =00660128      239     (1)
DS$_NOPCS               =00660110      239     (1)
DS$_NORMAL              =00660001      239     (1)
DS$_NOSUPPORT           =006600B1      239     (1)
DS$_NOTALLOWMP          =00660148      239     (1)     #-1296    (12)    #-1351    (13)    #-1400    (14)
DS$_NOTDON              =00660030      239     (1)
DS$_NOTIMP              =006600B0      239     (1)       239     (1)
DS$_NULLSTR             =00660010      239     (1)
DS$_OVERFLOW            =00660008      239     (1)
DS$_POWER               =00660098      239     (1)
DS$_PROGERR             =00660020      239     (1)     #-1464    (16)
DS$_SEVERE              =00660004      239     (1)
DS$_TRANSL              =006600A0      239     (1)
DS$_TRUNCATE            =00660028      239     (1)
DS$_UNEXPINT            =006600D8      239     (1)
DS$_UNSUPPDEV           =00660158      239     (1)
DS$_VASFULL             =00660048      239     (1)
DS$_WARNING             =00660000      239     (1)     #-1345    (13)      239     (1)
DSA$AT_APTTXT            0000FA00                      #-827     (6)
DSA$GL_FLAGS             0000FE00                        1675    (20)      1714    (21)
DSA$GL_SID               0000FE14                      #-1592    (18)    #-552     (1)     #-621     (2)
DSA$V_USER              =0000001C                      #-1674    (20)    #-1713    (21)
DSR$CHECKLOAD            00000000-XR                     272     (1)       817     (6)
DSR$MMENABLE             000006DC-R     1388    (14)    #-1293    (12)    #-1348    (13)
DSR$SETIMR_INI           00000000-XR                     265     (1)
DSX$FREEDBGSYM           000008AA-R     1711    (21)
DSX$MAPDBGBLOCK          0000085C-R     1672    (20)
DSX$MMOFF                000006B6-R     1344    (13)
DSX$MMOFF_X              000006DB-R     1354    (13)    #-1347    (13)    #-1352    (13)
DSX$MMON                 0000069C-R     1291    (12)
DSX$MMON_X               000006B5-R     1299    (12)    #-1297    (12)
DSX$PRINT                00000000-XR                     266     (1)       457     (1)
DS_ERRSUP                00000000-XR                    1214    (11)       266     (1)
ENV$M_DOMAIN            =00000002      240     (1)
ENV$M_LEVEL             =00000001      240     (1)
ENV$M_SUPER             =000003FC      240     (1)
ENV$S_DOMAIN            =00000001      240     (1)
ENV$S_LEVEL             =00000001      240     (1)
ENV$S_SUPER             =00000008      240     (1)
ENV$V_DOMAIN            =00000001      240     (1)       240     (1)
ENV$V_LEVEL             =00000000      240     (1)       240     (1)
ENV$V_SUPER             =00000002      240     (1)
ENV$_CPU                =00000000      240     (1)       240     (1)
ENV$_FUNCTIONAL         =00000000      240     (1)       240     (1)
ENV$_REPAIR             =00000001      240     (1)       240     (1)
ENV$_SUPER              =00000001      240     (1)
ENV$_SYSTEM             =00000001      240     (1)       240     (1)
ESTKPTR                  00000000-XR                     268     (1)     #-764     (4)
EXE$CNTREG               00000566-R     1114    (10)
EXE$CNTREG_X             0000069B-R     1247    (11)    #-1242    (11)
EXE$EXPREG               0000044B-R     955     (8)
EXE$EXPREG_X             00000565-R     1042    (8)     #-1033    (8)     #-966     (8)     #-985     (8)
EXE$SETPRT               00000719-R     1448    (16)
```

```
EXPREG$_ACMODE                  =0000000C     249    (1)
EXPREG$_NARGS                   =00000004     249    (1)
EXPREG$_PAGCNT                  =00000004     249    (1)     #-963    (8)
EXPREG$_PTE_OWNER_CODE          =00000014     318    (1)     #-1012   (8)
EXPREG$_REGION                  =00000010     249    (1)     #-1008   (8)     #-1035   (8)     #-978    (8)
EXPREG$_RETADR                  =00000008     249    (1)     #-1032   (8)     #-1041   (8)
EXPREG_COMMON                    0000045B-R   960    (8)     #-953    (8)
GETBUF$_NARGS                   =00000004     241    (1)
GETBUF$_PAGCNT                  =00000004     241    (1)
GETBUF$_PHYADR                  =0000000C     241    (1)
GETBUF$_REGION                  =00000010     241    (1)
GETBUF$_RETADR                  =00000008     241    (1)
GETMEM$_NARGS                   =00000003     242    (1)
GETMEM$_PAGCNT                  =00000004     242    (1)
GETMEM$_PHYADR                  =00000008     242    (1)
GETMEM$_VRTADR                  =0000000C     242    (1)
INI$RDONLY                       00000987-R   1839   (24)
INI$WRITABLE                     00000987-R   1840   (24)
IO$GQ_PHYSICAL                   00000000-XR                 267      (1)      519      (1)     #-534    (1)     #-560    (1)
ISTKPTR                          00000000-XR                 268      (1)     #-761     (4)
KSTKPTR                          00000000-XR                 268      (1)
K_P1PAGES                       =0000007E     261    (1)      1161     (10)    #-1162    (10)    #-574    (1)     #-843    (6)
                                                             991      (8)
L$A_LASTADR                      00000214                    820      (6)     #-821     (6)
L_POLEN                          00000000-R    286    (1)     #-1154   (10)      304      (1)     #-461    (1)     #-837    (6)
                                                            #-951     (8)     #-958     (8)
MAP$IOSPACE                      00000000-XR                 263      (1)      580      (1)
MAPFREE                          00000392-R    814    (6)      586      (1)
MAPMEM                           00000000-R    375    (1)
MAPMEM$IOSPACE                   0000025A-R    620    (2)     #-576    (1)     #-648     (2)
MAP_BLK                          00000914-R    1784   (24)     1684     (20)     1743     (22)
MAP_DEBUGGER                     000008D4-R    1739   (22)
MEMPOOL$INIT                     00000000-XR                 263      (1)      583      (1)
MMG$GL_SPTBASE                   00000000-XR                #-1166    (10)      264      (1)     #-492    (1)     #-853    (6)
                                                            #-995     (8)
MMG$IOLOCK                       0000084E-R    1642   (19)
MMG$UNLOCK                       00000858-R    1647   (19)    #-1645    (19)
MP$GR_BOOTED_PROCESSORS          00000000-XR                #-1396    (14)      273      (1)
POPT_BASE                        00000000-XR                 1147     (10)     1156     (10)     1205     (11)     263      (1)
                                                             388      (1)      421      (1)     #-508    (1)     #-559    (1)
                                                             749      (4)     #-832     (6)     #-835    (6)      952      (8)
                                                             957      (8)
PFN$AB_STATE                    =00000010-R   310    (1)
PFN$AB_TYPE                     =00000000     311    (1)
PFN$AL_BAK                      =00000008-R   306    (1)
PFN$AL_PTE                      =00000004-R   305    (1)
PFN$AW_BLINK                    =00000000     309    (1)
PFN$AW_FLINK                    =00000000     308    (1)
PFN$AW_REFCNT                   =0000000C-R   307    (1)
PFN$AW_SWPVBN                   =00000000-R   304    (1)
PONE                             000007A0-R   1505   (17)     1473     (16)
PPA_CONSRV4                     =20040010     320    (1)     #-389     (1)
PR$_IPL                         =00000012                   #-748     (4)     #-772     (4)
PR$_MAPEN                       =00000038                  #-1394    (14)    #-1402    (14)
PR$_P0BR                        =00000008                  #-1586    (18)     #-509    (1)     #-559    (1)
PR$_P0LR                        =00000009                  #-1492    (17)     #-462    (1)
PR$_P1BR                        =0000000A                  #-1588    (18)     #-527    (1)     #-558    (1)     #-624    (2)
```

```
                                                        #-652    (2)
PR$_P1LR                =0000000B                       #-1507   (17)   #-522    (1)    #-554    (1)
PR$_SBR                 =0000000C                       #-1604   (18)   #-491    (1)    #-627    (2)
PR$_SID_TYPRTUV2         00000000-XR                    #-1592   (18)     274    (1)    #-552    (1)    #-621    (2)
PR$_SLR                 =0000000D                       #-1522   (17)   #-544    (1)
PR$_SYS_TYP900           00000000-XR                      276    (1)    #-386    (1)
PR$_TBIA                =00000039                       #-1393   (14)
PR$_TBIS                =0000003A                       #-1558   (17)
PSL$V_IS                =0000001A       232     (1)     #-1389   (14)   #-1450   (16)
PTE$C_NA                =00000000                        466     (1)      496    (1)    #-573    (1)     636     (2)
                                                        #-757    (4)
PTE$C_UREW              =68000000                       #-763    (4)
PTE$C_URKW              =70000000                        505     (1)      633    (2)      657    (2)    #-760    (4)
PTE$C_URSW              =60000000                       #-766    (4)
PTE$C_UW                =20000000                       #-1010   (8)    #-1804   (24)     394    (1)     425     (1)
                                                        #-769    (4)
PTE$M_IO                =00100000       256     (1)
PTE$M_MODIFY            =04000000                       #-1209   (11)
PTE$M_OWN               =01800000                       #-1211   (11)   #-1219   (11)   #-1831   (24)     394     (1)
                                                         425     (1)      496    (1)      505    (1)    #-529    (1)
                                                        #-573    (1)      633    (2)      636    (2)      657    (2)
                                                        #-831    (6)    #-834    (6)
PTE$M_PFN               =001FFFFF                       #-1204   (11)   #-974    (8)
PTE$M_PROT              =78000000                       #-1013   (8)    #-1210   (11)   #-1805   (24)   #-459    (1)
                                                        #-643    (2)    #-659    (2)    #-752    (4)
PTE$M_VALID             =80000000                       #-1219   (11)   #-1831   (24)     394    (1)     425     (1)
                                                         496     (1)      505    (1)      633    (2)    #-643    (2)
                                                         657     (2)    #-659    (2)    #-831    (6)    #-834    (6)
PTE$S_OWN               =00000002                       #-1009   (8)    #-1177   (11)   #-1803   (24)   #-855    (6)
PTE$V_IO                =00000014       256     (1)
PTE$V_OWN               =00000017                       #-1009   (8)    #-1177   (11)   #-1803   (24)   #-855    (6)
PTE$V_VALID             =0000001F                       #-1007   (8)    #-1793   (24)   #-1802   (24)
PZERO                    0000078A-R     1490    (17)     1473    (16)
RELBUF$_NARGS           =00000003       244     (1)
RELBUF$_PAGCNT          =00000004       244     (1)
RELBUF$_REGION          =0000000C       244     (1)
RELBUF$_RETADR          =00000008       244     (1)
RELMEM$_NARGS           =00000003       245     (1)
RELMEM$_PAGCNT          =00000004       245     (1)
RELMEM$_PHYADR          =00000008       245     (1)
RELMEM$_VRTADR          =0000000C       245     (1)
RESERVD                  000007CC-R     1536    (17)     1473    (16)
RPTEADR                  00000804-R     1583    (18)   #-1547    (17)     1644    (19)     1792    (24)     1801    (24)
                                                         1830    (24)      578    (1)
RTVAX                    000002BC-R     651     (2)    #-622     (2)
SCB_UNKINT               00000000-XR                      268     (1)
SEP_FUNCTIONAL          =00000002       240     (1)
SEP_REPAIR              =00000003       240     (1)
SET                      000007D0-R     1545    (17)   #-1497    (17)   #-1512   (17)   #-1527   (17)
SETPRT$_ACMODE          =0000000C       257     (1)
SETPRT$_INADR           =00000004       257     (1)    #-1459    (16)   #-1480   (16)
SETPRT$_NARGS           =00000005       257     (1)
SETPRT$_PROT            =00000010       257     (1)    #-1556    (17)
SETPRT$_PRVPRT          =00000014       257     (1)    #-1548    (17)   #-1552   (17)
SETPRT$_RETADR          =00000008       257     (1)    #-1454    (16)   #-1467   (16)   #-1478   (16)   #-1481   (16)
                                                       #-1559    (17)
SETPRT_XIT               00000789-R     1482    (16)   #-1465    (16)   #-1475   (16)   #-1479   (16)
```

```
SIZ...                  =00000001     256    (1)      233    (1)      240    (1)      256    (1)
SS$_ACCVIO              =0000000C                    #-1537  (17)
SS$_ILLPAGCNT           =000000FC                    #-965   (8)
SS$_LENVIO              =0000018C                    #-1495  (17)   #-1510  (17)   #-1525  (17)
SS$_NORMAL              =00000001                    #-1116  (10)   #-1403  (14)   #-1563  (17)   #-1648  (19)
                                                     #-702   (3)    #-973   (8)
SS$_PAGOWNVIO           =000001EC                    #-1239  (11)
SS$_RESIGNAL            =00000918                    #-714   (3)
SS$_VASFULL             =00000244                    #-1025  (8)    #-968   (8)
SSTKPTR                  00000000-XR                   268   (1)    #-767   (4)
STACKPROT                00000321-R    747    (4)      585   (1)     864    (6)
STACK_BASE               00000000-XR                   268   (1)    #-750   (4)    #-755   (4)
SYMTAB_HI                00000000-XR                  #-1716  (21)   #-1718  (21)    271    (1)
SYS$UNWIND               00000000-XR                   265   (1)     711    (3)
SYSTEM                   000007B6-R    1520   (17)     1473  (16)
T_INSFMEM                00000007-R    329    (1)      457   (1)
UNMAP_BLK                0000096C-R    1827   (24)     1717  (21)    1767   (23)
UNMAP_DEBUGGER           000008ED-R    1761   (23)
USTKPTR                  00000000-XR                   268   (1)    #-753   (4)    #-770   (4)
XDS$INIT                =00000000     312    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CNTREGDEF	1	235 (1)	235 (1)
$D1_PRINT_S	1	457 (1)	457 (1)
$DEF	1	247 (1)	
$DEFINI	1	237 (1)	237 (1) 238 (1) 243 (1) 246 (1) 250 (1)
			251 (1) 252 (1) 253 (1) 254 (1) 255 (1)
			258 (1)
$DS_BITDEF	2	236 (1)	236 (1)
$DS_CFDEF	1	237 (1)	237 (1)
$DS_DSADEF	5	238 (1)	238 (1)
$DS_DSDEF	3	239 (1)	239 (1)
$DS_ENVDEF	2	240 (1)	240 (1)
$DS_GETBUF_DEF	1	241 (1)	241 (1)
$DS_GETMEM_DEF	1	242 (1)	242 (1)
$DS_HDRDEF	2	243 (1)	243 (1)
$DS_RELBUF_DEF	1	244 (1)	244 (1)
$DS_RELMEM_DEF	1	245 (1)	245 (1)
$DS_SCBDEF	3	246 (1)	246 (1)
$DS_TYPEDEF	4	247 (1)	247 (1)
$EQU	1	247 (1)	232 (1) 236 (1) 239 (1) 240 (1) 247 (1)
$EQULS1	1	247 (1)	232 (1) 239 (1) 240 (1) 247 (1)
$EQULST	1	232 (1)	232 (1) 239 (1) 240 (1) 247 (1)
$EXPREGDEF	1	249 (1)	249 (1)
$GBLINI	2	232 (1)	232 (1) 233 (1) 236 (1) 239 (1) 240 (1)
			247 (1)
$IRPDEF	4	250 (1)	250 (1)
$OFFDEF	1	235 (1)	235 (1) 241 (1) 242 (1) 244 (1) 245 (1)
			249 (1) 257 (1)
$PHDDEF	6	251 (1)	251 (1)
$PRDEF	5	252 (1)	252 (1)
$PRINT	2	454 (1)	454 (1)
$PRTDEF	1	253 (1)	253 (1)
$PSLDEF	2	254 (1)	254 (1)
$PTEDEF	3	255 (1)	255 (1)
$PUSHADR	1	1214 (11)	1214 (11)
$SETPRTDEF	1	257 (1)	257 (1)
$SSDEF	21	258 (1)	258 (1)
$VIELD	1	233 (1)	233 (1) 240 (1) 256 (1)
$VIELD1	1	247 (1)	233 (1) 240 (1) 256 (1)
CASE	1	1472 (16)	1472 (16)
CMK	1	1389 (14)	1389 (14) 1450 (16)
CMKDEF	1	232 (1)	232 (1)
DSBINT	1	748 (4)	748 (4)
DSFDEF	3	233 (1)	233 (1)
ENBINT	1	772 (4)	772 (4)
ERRSUP_S	1	1214 (11)	1214 (11)
IFNORD	1	820 (6)	820 (6)
MODNAM	1	327 (1)	327 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ACBL	00F1	1002	(8)	859	(6)										
ADDL	00C0	1001	(8)	1017	(8)	1149	(10)								
ADDL2	00C0	1591	(18)	1598	(18)	1607	(18)	1680	(20)	1794	(24)	1807	(24)	1832	(24)
		420	(1)												
ADDL3	00C1	1162	(10)	414	(1)										
AOBLEQ	00F3	427	(1)	549	(1)	638	(2)								
AOBLSS	00F2	1226	(11)	396	(1)	402	(1)	417	(1)	467	(1)	499	(1)	502	(1)
		507	(1)	540	(1)	753	(4)	761	(4)	764	(4)	767	(4)	770	(4)
		837	(6)												
ASHL	0078	1007	(8)	1028	(8)	1150	(10)	1155	(10)	1167	(10)	1294	(12)	1349	(13)
		1460	(16)	1461	(16)	1546	(17)	1557	(17)	1590	(18)	1597	(18)	1606	(18)
		1802	(24)	390	(1)	399	(1)	405	(1)	413	(1)	419	(1)	434	(1)
		449	(1)	504	(1)	516	(1)	569	(1)	632	(2)	824	(6)	950	(8)
		975	(8)	997	(8)										
BBS	00E0	1199	(11)	1201	(11)	1389	(14)	1450	(16)	1584	(18)	1585	(18)	1674	(20)
		1713	(21)	1793	(24)	629	(2)								
BBSS	00E2	1645	(19)												
BEQL	0013	1033	(8)	1146	(10)	1207	(11)	1213	(11)	1230	(11)	1242	(11)	1397	(14)
		1399	(14)	1455	(16)	1479	(16)	1550	(17)	1560	(17)	1593	(18)	1678	(20)
		437	(1)	465	(1)	547	(1)	622	(2)	820	(6)	856	(6)		
BGEQ	0018	1000	(8)	1019	(8)	1021	(8)	1023	(8)	1175	(11)	1463	(16)	451	(1)
BGTR	0014	1494	(17)	1524	(17)	439	(1)	964	(8)						
BGTRU	001A	1235	(11)												
BICB	008A	634	(2)												
BICL	00CA	1029	(8)	1220	(11)	1679	(20)	459	(1)	643	(2)	659	(2)		
BICL2	00CA	1013	(8)	1219	(11)	1764	(23)	1766	(23)	1805	(24)	1831	(24)	752	(4)
		831	(6)												
BICL3	00CB	1204	(11)	1480	(16)	699	(3)	974	(8)						
BICW2	00AA	1243	(11)												
BISL	00C8	1012	(8)	517	(1)	644	(2)	757	(4)	760	(4)	763	(4)	766	(4)
		769	(4)												
BISL2	00C8	1010	(8)	1014	(8)	1804	(24)	1806	(24)	660	(2)	834	(6)		
BISL3	00C9	1561	(17)	570	(1)										
BISW2	00A8	1031	(8)	1034	(8)										
BITB	0093	464	(1)	546	(1)										
BITL	00D3	1209	(11)	1677	(20)										
BLBC	00E9	1475	(16)	1686	(20)	401	(1)	416	(1)	818	(6)				
BLBS	00E8	1124	(10)												
BLEQ	0015	1509	(17)	971	(8)										
BLSS	0019	1796	(24)	1809	(24)	1834	(24)	655	(2)	826	(6)	828	(6)	830	(6)
BNEQ	0012	1036	(8)	1179	(11)	1198	(11)	1223	(11)	1297	(12)	1347	(13)	1352	(13)
		387	(1)	553	(1)	708	(3)								
BRB	0011	1004	(8)	1158	(10)	1163	(10)	1215	(11)	1227	(11)	1236	(11)	1389	(14)
		1401	(14)	1450	(16)	1587	(18)	1814	(24)	397	(1)	833	(6)	953	(8)
		969	(8)	988	(8)	993	(8)								
BRW	0031	1152	(10)	1200	(11)	1202	(11)	1237	(11)	1465	(16)	966	(8)	985	(8)
BSBB	0010	1293	(12)	1348	(13)	1497	(17)	1512	(17)	1527	(17)	1547	(17)	1595	(18)
		576	(1)												
CALLS	00FB	1214	(11)	400	(1)	415	(1)	457	(1)	586	(1)	711	(3)		
CASEL	00CF	1117	(10)	1473	(16)	978	(8)								

```
CHMK            00BC    1389  (14)  1450  (16)
CLRL            00D4    1172  (11)  1217  (11)  1389  (14)  1450  (16)  383   (1)   384   (1)   422   (1)
                        493   (1)   501   (1)   548   (1)   823   (6)   846   (6)   857   (6)   984   (8)
CLRQ            007C    710   (3)   861   (6)   862   (6)   977   (8)
CMPB            0091    1592  (18)  385   (1)   552   (1)   621   (2)
CMPL            00D1    1018  (8)   1020  (8)   1035  (8)   1206  (11)  1234  (11)  1296  (12)  1351  (13)
                        1398  (14)  1493  (17)  1508  (17)  1523  (17)  1795  (24)  1808  (24)  1833  (24)
                        438   (1)   450   (1)   654   (2)   707   (3)   825   (6)   827   (6)   829   (6)
CMPZV           00ED    1177  (11)  855   (6)
CVTBL           0098    457   (1)
DECL            00D7    1171  (11)  404   (1)   523   (1)   555   (1)
DIVL2           00C6    562   (1)
EXTZV           00EF    1395  (14)  1470  (16)  1491  (17)  1506  (17)  1521  (17)  1551  (17)  1589  (18)
                        1594  (18)  1596  (18)  1605  (18)  519   (1)   536   (1)   626   (2)   641   (2)
INCL            00D6    1295  (12)  1350  (13)  1468  (16)  1476  (16)  498   (1)   532   (1)   566   (1)
                        647   (2)   662   (2)   758   (4)
INSV            00F0    1008  (8)   1169  (10)  1221  (11)  1556  (17)  1803  (24)  642   (2)
JSB             0016    1389  (14)  1450  (16)  1644  (19)  1684  (20)  1717  (21)  1743  (22)  1767  (23)
                        1792  (24)  1801  (24)  1830  (24)  578   (1)   580   (1)   583   (1)   585   (1)
                        817   (6)   864   (6)
MCOML           00D2    1456  (16)  1457  (16)
MFPR            00DB    1394  (14)  1492  (17)  1507  (17)  1522  (17)  1586  (18)  1588  (18)  1604  (18)
                        554   (1)   624   (2)   627   (2)   652   (2)   748   (4)
MOVAB           009E    394   (1)   425   (1)   466   (1)   496   (1)   505   (1)   537   (1)   633   (2)
                        635   (2)   636   (2)   657   (2)   698   (3)   816   (6)
MOVAL           00DE    1147  (10)  1151  (10)  1156  (10)  1157  (10)  1161  (10)  1205  (11)  388   (1)
                        421   (1)   469   (1)   542   (1)   551   (1)   625   (2)   628   (2)   653   (2)
                        749   (4)   952   (8)   957   (8)   991   (8)
MOVAQ           007E    860   (6)
MOVB            0090    1552  (17)
MOVL            00D0    1003  (8)   1006  (8)   1016  (8)   1116  (10)  1154  (10)  1160  (10)  1165  (10)
                        1166  (10)  1231  (11)  1239  (11)  1292  (12)  1298  (12)  1345  (13)  1353  (13)
                        1400  (14)  1403  (14)  1454  (16)  1459  (16)  1464  (16)  1467  (16)  1537  (17)
                        1559  (17)  1563  (17)  1643  (19)  1648  (19)  1676  (20)  1682  (20)  1687  (20)
                        1688  (20)  1715  (21)  1716  (21)  1718  (21)  1741  (22)  1742  (22)  1763  (23)
                        1765  (23)  1786  (24)  1791  (24)  1798  (24)  1800  (24)  1810  (24)  1813  (24)
                        1829  (24)  389   (1)   391   (1)   436   (1)   447   (1)   461   (1)   492   (1)
                        529   (1)   531   (1)   539   (1)   563   (1)   564   (1)   565   (1)   572   (1)
                        573   (1)   577   (1)   579   (1)   581   (1)   656   (2)   702   (3)   709   (3)
                        821   (6)   844   (6)   853   (6)   958   (8)   962   (8)   963   (8)   973   (8)
                        976   (8)   990   (8)   992   (8)   995   (8)   996   (8)
MOVPSL          00DC    1389  (14)  1450  (16)
MOVQ            007D    1037  (8)   1041  (8)   1245  (11)  534   (1)   560   (1)   701   (3)   706   (3)
                        987   (8)
MOVZBL          009A    457   (1)   494   (1)   574   (1)   843   (6)
MOVZWL          003C    1025  (8)   1495  (17)  1510  (17)  1525  (17)  714   (3)   750   (4)   755   (4)
                        965   (8)   968   (8)
MTPR            00DA    1393  (14)  1402  (14)  1558  (17)  462   (1)   491   (1)   508   (1)   522   (1)
                        527   (1)   544   (1)   558   (1)   559   (1)   748   (4)   772   (4)
MULL3           00C5    1148  (10)  525   (1)   556   (1)
POPL            8ED0    1685  (20)  1811  (24)
POPR            00BA    1404  (14)  1744  (22)  1768  (23)  587   (1)   639   (2)   646   (2)   664   (2)
PROBER          000C    820   (6)
PUSHAB          009F    1471  (16)  457   (1)
PUSHAL          00DF    1214  (11)
PUSHL           00DD    1038  (8)   1039  (8)   1214  (11)  1224  (11)  1244  (11)  1683  (20)  1785  (24)
                        457   (1)
```

PUSHR	00BB	1740	(22)	1762	(23)	382	(1)	623	(2)	631	(2)	651	(2)		
REI	0002	1405	(14)	1483	(16)										
RET	0004	1043	(8)	1248	(11)	1300	(12)	1355	(13)	1451	(16)	1689	(20)	1719	(21)
		703	(3)	712	(3)	715	(3)	865	(6)						
RSB	0005	1390	(14)	1496	(17)	1498	(17)	1511	(17)	1513	(17)	1526	(17)	1528	(17)
		1538	(17)	1564	(17)	1599	(18)	1608	(18)	1649	(19)	1745	(22)	1769	(23)
		1812	(24)	1835	(24)	1841	(24)	588	(1)	649	(2)	665	(2)	773	(4)
SOBGEQ	00F4	567	(1)	967	(8)										
SOBGTR	00F5	1477	(16)	648	(2)	663	(2)	847	(6)	972	(8)				
SUBL	00C2	1233	(11)	521	(1)	526	(1)	535	(1)						
SUBL2	00C2	393	(1)	424	(1)										
SUBL3	00C3	1027	(8)	1168	(10)	1462	(16)	1681	(20)	460	(1)	543	(1)	557	(1)
		561	(1)	951	(8)										
TSTB	0095	1346	(13)												
TSTL	00D5	1022	(8)	1032	(8)	1145	(10)	1174	(11)	1197	(11)	1222	(11)	1229	(11)
		1241	(11)	1396	(14)	1478	(16)	1548	(17)	970	(8)	999	(8)		
XORL3	00CD	1208	(11)												

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	327	(1)	330	(1)												
.BLKB	298	(1)														
.BLKL	286	(1)	287	(1)	288	(1)	289	(1)	290	(1)	293	(1)				
.DSABL	14	(1)														
.END	1843	(24)														
.ENDC	1214	(11)	232	(1)	233	(1)	236	(1)	239	(1)	240	(1)	247	(1)	256	(1)
	457	(1)	748	(4)	772	(4)										
.ENDM	1214	(11)	1389	(14)	1472	(16)	232	(1)	233	(1)	235	(1)	236	(1)	237	(1)
	238	(1)	239	(1)	240	(1)	241	(1)	242	(1)	243	(1)	244	(1)	245	(1)
	246	(1)	247	(1)	249	(1)	250	(1)	251	(1)	252	(1)	253	(1)	254	(1)
	255	(1)	257	(1)	258	(1)	327	(1)	454	(1)	457	(1)	748	(4)	772	(4)
	820	(6)														
.ENDR	1473	(16)	232	(1)	233	(1)	239	(1)	240	(1)	247	(1)	256	(1)	457	(1)
.ENTRY	1114	(10)	1291	(12)	1344	(13)	1448	(16)	1672	(20)	1711	(21)	814	(6)	948	(8)
	955	(8)														
.EXTRN	263	(1)	264	(1)	265	(1)	266	(1)	267	(1)	268	(1)	269	(1)	270	(1)
	271	(1)	272	(1)	273	(1)	274	(1)	275	(1)	276	(1)				
.IDENT	12	(1)														
.IF	1214	(11)	232	(1)	233	(1)	236	(1)	239	(1)	240	(1)	247	(1)	256	(1)
	457	(1)	748	(4)	772	(4)										
.IFF	1214	(11)	232	(1)	233	(1)	236	(1)	239	(1)	240	(1)	247	(1)	748	(4)
	772	(4)														
.IIF	1214	(11)	232	(1)	233	(1)	236	(1)	239	(1)	240	(1)	247	(1)	256	(1)
.IRP	1473	(16)	232	(1)	233	(1)	235	(1)	239	(1)	240	(1)	241	(1)	242	(1)
	244	(1)	245	(1)	247	(1)	249	(1)	256	(1)	257	(1)	457	(1)		
.LIBRARY	224	(1)	225	(1)	226	(1)										
.LIST	235	(1)	237	(1)	238	(1)	241	(1)	242	(1)	243	(1)	244	(1)	245	(1)
	246	(1)	249	(1)	257	(1)										
.LONG	296	(1)														
.MACRO	232	(1)	233	(1)	236	(1)	239	(1)	240	(1)	247	(1)				
.MDELETE	232	(1)	236	(1)	239	(1)	240	(1)	241	(1)	242	(1)	244	(1)	245	(1)
.MLIST	235	(1)	237	(1)	238	(1)	241	(1)	242	(1)	243	(1)	244	(1)	245	(1)
	246	(1)	249	(1)	257	(1)										
.NOCROSS	237	(1)	238	(1)	243	(1)	246	(1)	250	(1)	251	(1)	252	(1)	253	(1)
	254	(1)	255	(1)	258	(1)										
.NOSHOW	13	(1)														
.PAGE	217	(1)	278	(1)	323	(1)	332	(1)	374	(1)	470	(1)	510	(1)	589	(1)
.PSECT	280	(1)	325	(1)	334	(1)										
.RESTORE	237	(1)	238	(1)	243	(1)	246	(1)	250	(1)	251	(1)	252	(1)	253	(1)
	254	(1)	255	(1)	258	(1)										
.SAVE	237	(1)	238	(1)	243	(1)	246	(1)	250	(1)	251	(1)	252	(1)	253	(1)
	254	(1)	255	(1)	258	(1)										
.SBTTL	1045	(9)	1250	(12)	1302	(13)	1357	(14)	1407	(15)	1567	(18)	1651	(20)	1692	(21)
	1722	(22)	1747	(23)	333	(1)	590	(1)	667	(3)	717	(4)	775	(5)	867	(7)
.SIGNED_WORD	1473	(16)														
.SUBTITLE	218	(1)	279	(1)	324	(1)										
.TITLE	11	(1)														
.WORD	1119	(10)	1120	(10)	1121	(10)	1122	(10)	697	(3)	705	(3)	979	(8)	980	(8)
	981	(8)	982	(8)												

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	31	#-1008	(8)	#-1012	(8)	#-1032	(8)	#-1035	(8)	#-1041	(8)	#-1117	(10)
		#-1124	(10)	#-1145	(10)	#-1148	(10)	#-1149	(10)	#-1178	(11)	#-1197	(11)
		#-1226	(11)	#-1241	(11)	#-1245	(11)	#-1454	(16)	#-1459	(16)	#-1467	(16)
		#-1478	(16)	#-1480	(16)	#-1481	(16)	#-1548	(17)	#-1552	(17)	#-1556	(17)
		#-1559	(17)	#-1676	(20)	#-1688	(20)	#-699	(3)	#-706	(3)	#-963	(8)
		#-978	(8)										
FP	1	#-698	(3)										
R0	112	#-1025	(8)	#-1116	(10)	#-1231	(11)	#-1239	(11)	#-1292	(12)	#-1294	(12)
		#-1295	(12)	#-1298	(12)	#-1345	(13)	#-1349	(13)	#-1350	(13)	#-1353	(13)
		1395	(14)	#-1398	(14)	#-1402	(14)	#-1404	(14)	#-1454	(16)	#-1456	(16)
		#-1457	(16)	#-1459	(16)	#-1460	(16)	#-1461	(16)	#-1464	(16)	#-1475	(16)
		#-1492	(17)	#-1493	(17)	#-1495	(17)	#-1507	(17)	#-1508	(17)	#-1510	(17)
		#-1522	(17)	#-1523	(17)	#-1525	(17)	#-1537	(17)	#-1551	(17)	#-1552	(17)
		#-1559	(17)	#-1561	(17)	#-1563	(17)	#-1643	(19)	#-1648	(19)	#-1686	(20)
		#-1810	(24)	#-1813	(24)	#-401	(1)	#-416	(1)	#-460	(1)	#-461	(1)
		#-462	(1)	#-516	(1)	#-517	(1)	#-526	(1)	#-527	(1)	#-534	(1)
		#-535	(1)	#-536	(1)	537	(1)	#-540	(1)	#-543	(1)	#-544	(1)
		#-554	(1)	#-555	(1)	#-556	(1)	#-557	(1)	#-558	(1)	#-560	(1)
		#-561	(1)	#-562	(1)	#-567	(1)	#-574	(1)	#-631	(2)	#-632	(2)
		633	(2)	#-635	(2)	#-638	(2)	#-639	(2)	#-648	(2)	#-663	(2)
		#-702	(3)	#-706	(3)	#-707	(3)	#-709	(3)	#-714	(3)	#-750	(4)
		#-752	(4)	#-753	(4)	#-755	(4)	#-757	(4)	#-758	(4)	#-760	(4)
		#-761	(4)	#-763	(4)	#-764	(4)	#-766	(4)	#-767	(4)	#-769	(4)
		#-770	(4)	#-818	(6)	#-860	(6)	#-861	(6)	#-862	(6)	#-965	(8)
		#-968	(8)	#-973	(8)	#-984	(8)						
R1	114	#-1013	(8)	#-1014	(8)	#-1016	(8)	#-1018	(8)	#-1028	(8)	#-1037	(8)
		#-1038	(8)	#-1150	(10)	1151	(10)	#-1154	(10)	#-1155	(10)	1157	(10)
		#-1161	(10)	#-1165	(10)	#-1168	(10)	#-1174	(11)	1177	(11)	1199	(11)
		1201	(11)	#-1204	(11)	#-1206	(11)	#-1208	(11)	#-1217	(11)	#-1234	(11)
		#-1296	(12)	#-1298	(12)	#-1351	(13)	#-1353	(13)	#-1400	(14)	#-1403	(14)
		#-1491	(17)	#-1493	(17)	#-1506	(17)	#-1508	(17)	#-1521	(17)	#-1523	(17)
		#-1546	(17)	1551	(17)	1556	(17)	#-1557	(17)	#-1558	(17)	#-1561	(17)
		1584	(18)	1585	(18)	1589	(18)	#-1590	(18)	#-1591	(18)	1594	(18)
		1596	(18)	#-1597	(18)	#-1598	(18)	1605	(18)	#-1606	(18)	#-1607	(18)
		#-1643	(19)	1645	(19)	#-1791	(24)	1793	(24)	#-1800	(24)	#-1805	(24)
		#-1806	(24)	#-1829	(24)	#-1831	(24)	#-494	(1)	496	(1)	#-498	(1)
		#-504	(1)	#-507	(1)	#-525	(1)	#-526	(1)	#-535	(1)	536	(1)
		#-561	(1)	#-563	(1)	#-565	(1)	#-566	(1)	#-569	(1)	#-570	(1)
		#-572	(1)	#-577	(1)	#-579	(1)	625	(2)	#-631	(2)	#-634	(2)
		635	(2)	636	(2)	#-638	(2)	#-639	(2)	#-647	(2)	653	(2)
		657	(2)	#-662	(2)	#-699	(3)	#-701	(3)	#-709	(3)	#-749	(4)
		#-752	(4)	#-757	(4)	#-760	(4)	#-763	(4)	#-766	(4)	#-769	(4)
		#-962	(8)	#-974	(8)	#-987	(8)						
R2	70	#-1018	(8)	#-1039	(8)	1114	(10)	#-1147	(10)	1151	(10)	#-1156	(10)
		1157	(10)	#-1160	(10)	1161	(10)	#-1166	(10)	#-1168	(10)	#-1234	(11)
		1448	(16)	1672	(20)	#-1676	(20)	#-1677	(20)	#-1679	(20)	#-1680	(20)
		#-1681	(20)	#-1683	(20)	#-1685	(20)	#-1687	(20)	#-1688	(20)	1711	(21)
		#-1715	(21)	#-1740	(22)	#-1741	(22)	#-1744	(22)	#-1762	(23)	#-1763	(23)
		#-1764	(23)	#-1768	(23)	#-1786	(24)	#-1791	(24)	#-1794	(24)	#-1795	(24)
		#-1798	(24)	#-1800	(24)	#-1807	(24)	#-1808	(24)	#-1829	(24)	#-1832	(24)

```
                 #-1833  (24)    #-382   (1)     #-529   (1)     #-531   (1)     #-539   (1)     #-564   (1)
                 #-565   (1)     #-573   (1)     #-587   (1)     #-644   (2)     #-660   (2)       814   (6)
                 #-824   (6)     #-825   (6)     #-827   (6)     #-829   (6)     #-843   (6)     #-847   (6)
                   948   (8)       950   (8)     #-951   (8)       952   (8)       955   (8)     #-957   (8)
                 #-962   (8)     #-970   (8)
R3       128     #-1002  (8)     #-1016  (8)     #-1020  (8)     #-1022  (8)      1114   (10)    #-1148  (10)
                 #-1149  (10)    #-1150  (10)    #-1155  (10)    #-1162  (10)    #-1167  (10)      1169  (10)
                 #-1171  (11)    #-1224  (11)    #-1233  (11)    #-1243  (11)    #-1244  (11)      1448  (16)
                 #-1460  (16)    #-1462  (16)      1470  (16)    #-1476  (16)      1491  (17)      1506  (17)
                   1521  (17)    #-1546  (17)    #-1557  (17)      1672  (20)    #-1682  (20)      1711  (21)
                 #-1716  (21)    #-1740  (22)    #-1742  (22)    #-1744  (22)    #-1762  (23)    #-1765  (23)
                 #-1766  (23)    #-1768  (23)    #-1795  (24)    #-1808  (24)    #-1833  (24)    #-382   (1)
                 #-383   (1)     #-393   (1)       394   (1)     #-395   (1)     #-396   (1)     #-399   (1)
                 #-402   (1)     #-404   (1)     #-405   (1)     #-419   (1)     #-420   (1)     #-427   (1)
                 #-434   (1)     #-438   (1)     #-447   (1)     #-449   (1)     #-450   (1)     #-457   (1)
                 #-460   (1)     #-464   (1)       466   (1)     #-467   (1)       469   (1)     #-493   (1)
                 #-497   (1)     #-499   (1)     #-501   (1)     #-502   (1)       505   (1)     #-506   (1)
                 #-507   (1)     #-516   (1)     #-531   (1)     #-532   (1)       537   (1)     #-539   (1)
                 #-540   (1)       542   (1)     #-543   (1)     #-546   (1)     #-548   (1)     #-549   (1)
                   551   (1)     #-587   (1)     #-623   (2)     #-624   (2)       625   (2)       626   (2)
                   642   (2)     #-643   (2)     #-644   (2)     #-646   (2)     #-651   (2)     #-652   (2)
                   653   (2)     #-654   (2)     #-656   (2)     #-659   (2)     #-660   (2)     #-664   (2)
                   814   (6)     #-823   (6)     #-824   (6)     #-832   (6)     #-835   (6)     #-837   (6)
                 #-844   (6)     #-846   (6)     #-853   (6)       855   (6)     #-857   (6)     #-859   (6)
                   948   (8)       955   (8)     #-963   (8)     #-972   (8)     #-987   (8)     #-990   (8)
                   991   (8)     #-995   (8)     #-999   (8)
R4       58      #-1002  (8)     #-1020  (8)      1114   (10)    #-1172  (11)    #-1222  (11)    #-1226  (11)
                 #-1229  (11)      1448  (16)    #-1461  (16)    #-1462  (16)    #-1467  (16)    #-1785  (24)
                 #-1786  (24)    #-1798  (24)    #-1802  (24)      1803  (24)    #-1804  (24)    #-1806  (24)
                 #-1811  (24)    #-382   (1)     #-390   (1)     #-396   (1)     #-405   (1)     #-414   (1)
                 #-422   (1)     #-424   (1)       425   (1)     #-426   (1)     #-427   (1)     #-519   (1)
                 #-521   (1)     #-522   (1)     #-523   (1)     #-525   (1)     #-563   (1)     #-569   (1)
                 #-572   (1)     #-587   (1)     #-623   (2)     #-626   (2)       628   (2)       629   (2)
                 #-633   (2)       641   (2)     #-646   (2)       814   (6)     #-816   (6)     #-821   (6)
                 #-825   (6)       948   (8)     #-951   (8)       955   (8)     #-958   (8)     #-967   (8)
                 #-991   (8)     #-996   (8)
R5       64      #-1001  (8)     #-1003  (8)     #-1006  (8)     #-1017  (8)     #-1027  (8)      1114   (10)
                 #-1204  (11)      1205  (11)    #-1206  (11)    #-1208  (11)    #-1219  (11)    #-1220  (11)
                   1221  (11)      1448  (16)    #-1462  (16)    #-1468  (16)    #-1477  (16)    #-382   (1)
                 #-384   (1)     #-388   (1)     #-393   (1)     #-395   (1)     #-413   (1)     #-417   (1)
                 #-420   (1)     #-421   (1)     #-424   (1)     #-426   (1)     #-459   (1)     #-466   (1)
                   469   (1)     #-491   (1)     #-492   (1)     #-497   (1)     #-501   (1)     #-506   (1)
                 #-531   (1)     #-539   (1)       542   (1)     #-548   (1)       551   (1)     #-557   (1)
                 #-564   (1)     #-581   (1)     #-587   (1)     #-632   (2)     #-637   (2)     #-654   (2)
                 #-656   (2)     #-658   (2)       814   (6)       948   (8)       955   (8)     #-974   (8)
                 #-975   (8)     #-976   (8)     #-992   (8)     #-997   (8)
R6       19      #-1007  (8)       1009  (8)     #-1010  (8)     #-1012  (8)     #-1014  (8)      1448  (16)
                 #-1470  (16)    #-1473  (16)    #-382   (1)     #-389   (1)     #-390   (1)     #-413   (1)
                 #-414   (1)     #-436   (1)     #-438   (1)     #-447   (1)     #-587   (1)       948   (8)
                   955   (8)
SP       38      #-1003  (8)     #-1006  (8)     #-1027  (8)     #-1034  (8)     #-1037  (8)     #-1041  (8)
                 #-1168  (10)    #-1169  (10)    #-1208  (11)    #-1212  (11)    #-1245  (11)    #-1389  (14)
                 #-1394  (14)    #-1398  (14)    #-1450  (16)    #-1586  (18)    #-1588  (18)    #-1591  (18)
                 #-1594  (18)    #-1598  (18)    #-1604  (18)    #-1607  (18)    #-399   (1)     #-414   (1)
                 #-457   (1)     #-627   (2)       628   (2)     #-641   (2)     #-642   (2)     #-710   (3)
                 #-748   (4)     #-772   (4)     #-977   (8)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.20
Command processing	886	00:00:00.28	00:00:00.73
Pass 1	1198	00:00:04.69	00:00:06.86
Symbol table sort	0	00:00:00.40	00:00:00.39
Pass 2	75	00:00:01.35	00:00:02.54
Symbol table output	3	00:00:00.10	00:00:00.26
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	22	00:00:00.56	00:00:00.94
Assembler run totals	2210	00:00:07.42	00:00:11.93

The working set limit was 3400 pages.
243189 bytes (475 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1302 non-local and 101 local symbols.
1843 source lines were read in Pass 1, producing 113 object records in Pass 2.
142 pages of virtual memory were used to define 37 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	15
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	6
SYS$COMMON:[SYSLIB]LIB.MLB;2	5
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	40

1437 GETS were required to define 40 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:MEMMGT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element MPDUMMY.B32
;    0002  0      ! *2    12-NOV-1987 10:52:41 GEARIN "FIXED EMPTY EXPRESSION PROBLEM"
;    0003  0      ! *1     6-FEB-1986 15:20:45 DS ""
;    0004  0      ! DEC/CMS REPLACEMENT HISTORY, Element MPDUMMY.B32
;    0005  0      %TITLE '*** - MPDUMMY - ENTRY POINTS FOR DUMMY MULTI-PROC ROUTINES'
;    0006  0      MODULE MPDUMMY(                     !
;    0007  0                     IDENT = '8.0',
;    0008  0                     OPTLEVEL = 3,
;    0009  0                     ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE,
;    0010  0                     NONEXTERNAL = LONG_RELATIVE)
;    0011  0                     ) =
;    0012  0
;    0013  0      !++
;    0014  0      !                           COPYRIGHT (c) 1985 BY
;    0015  0      !       DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS
;    0016  0      !
;    0017  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE AND MAY BE USED AND COPIED
;    0018  0      ! ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
;    0019  0      ! INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
;    0020  0      ! COPIES THEREOF MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY
;    0021  0      ! OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE  SOFTWARE  IS HEREBY
;    0022  0      ! TRANSFERRED.
;    0023  0      !
;    0024  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0025  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0026  0      ! CORPORATION.
;    0027  0      !
;    0028  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0029  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0030  0      !-
;    0031  0
;    0032  0
;    0033  0
;    0034  0      !++
;    0035  0      ! Facility:
;    0036  0      !
;    0037  0      !       Diagnostic Supervisor
;    0038  0      !
;    0039  0      ! Abstract:
;    0040  0      !
;    0041  0      !       This module contains dummy multi-processing routines/entry points
;    0042  0      !       to handle calls from non-multi-processing supervisors.
;    0043  0      !
;    0044  0      ! Environment:
;    0045  0      !
;    0046  0      !
;    0047  0      !
;    0048  0      ! Author:
;    0049  0      !
;    0050  0      !       Domenic Andella,        6-Dec-1984
;    0051  0      !
;    0052  0      ! Creation Date:
;    0053  0      !
;    0054  0      !        6-Dec-1984
;    0055  0      !
;    0056  0      ! Version:
;    0057  0      !
```

```
;   0058  0     !       8.0
;   0059  0     !
;   0060  0     ! Modified By:
;   0061  0     !
;   0062  0     !--
```

```
;    0064  0     %SBTTL ' Declarations'
;    0065  1     BEGIN
;    0066  1
;    0067  1     %SBTTL  'Libraries'
;    0068  1     LIBRARY '$Diag';
;    0069  1
;    0070  1     LIBRARY '$DS';
;    0071  1
;    0072  1     LIBRARY 'Sys$Library:Lib';
```

```
;    0074  1      GLOBAL
;    0075  1        MP$GL_CONDITION_FLAGS : BITVECTOR [32] INITIAL (0);
;    0076  1
;    0077  1
;    0078  1      ! THIS SYMBOL IS TEMPORARILY COMMENTED OUT UNTIL ALL MP
;    0079  1      ! MODULES ARE INCLUDED ARE ADDED TO MP OPTIONS LIST. IN
;    0080  1      ! ADDITION, SYMBOL MUST BE REMOVED FROM KERNEL.
;    0081  1
;    0082  1       ! MP$GR_BOOTED_PROCESSORS : BITVECTOR [32] INITIAL (0);
;    0083  1
;    0084  1      ! Dummy Routines
;    0085  1      GLOBAL ROUTINE DSX$_INTERLOCK_TEST(LOCAL_TEST,GLOBAL_TEST)=
;    0086  2      BEGIN
;    0087  2        RETURN (1);                                                    !G:2
;    0088  1      END;


                                                    .TITLE  MPDUMMY *** - MPDUMMY - ENTRY POINTS FOR DUMMY
;                                                           MULTI-PR
                                                    .IDENT  \8.0\

                                                    .PSECT  $GLOBAL$,NOEXE,2

                                    00000000  00000 MP$GL_CONDITION_FLAGS::
                                                    .LONG   0                                       ;


                                                    .PSECT  $CODE$,NOWRT,2

                                              0000 00000    .ENTRY  DSX$_INTERLOCK_TEST, Save nothing       ; 0085
                                        50    01   D0 00002  MOVL   #1, R0                                  ; 0087
                                              04   00005    RET                                             ; 0088

; Routine Size:  6 bytes,    Routine Base:  $CODE$ + 0000


;    0089  1
;    0090  1      GLOBAL ROUTINE DSX$BOOTATTACHED(UNIT,SCBADDR)=
;    0091  2      BEGIN
;    0092  2        RETURN (1);                                                  !G:2
;    0093  1      END;



                                              0000 00000    .ENTRY  DSX$BOOTATTACHED, Save nothing          ; 0090
                                        50    01   D0 00002  MOVL   #1, R0                                  ; 0092
                                              04   00005    RET                                             ; 0093

; Routine Size:  6 bytes,    Routine Base:  $CODE$ + 0006


;    0094  1
;    0095  1      GLOBAL ROUTINE DSX$STARTATTACHED(UNIT,START_ADDR)=
;    0096  2      BEGIN
```

```
;   0097  2       RETURN (1);                                                                    !G:2
;   0098  1     END;



                                                0000 00000        .ENTRY  DSX$STARTATTACHED, Save nothing                  ; 0095
                                   50        01  D0 00002        MOVL    #1, R0                                           ; 0097
                                                04 00005        RET                                                      ; 0098

; Routine Size:  6 bytes,    Routine Base:  $CODE$ + 000C


;   0099  1
;   0100  1     GLOBAL ROUTINE DSX$HALTATTACHED (UNIT) =
;   0101  2     BEGIN
;   0102  2       RETURN (1);                                                                    !G:2
;   0103  1     END;



                                                0000 00000        .ENTRY  DSX$HALTATTACHED, Save nothing                   ; 0100
                                   50        01  D0 00002        MOVL    #1, R0                                           ; 0102
                                                04 00005        RET                                                      ; 0103

; Routine Size:  6 bytes,    Routine Base:  $CODE$ + 0012


;   0104  1
;   0105  1     GLOBAL ROUTINE DSX$SHOWIDLE (BITMAP) =
;   0106  2     BEGIN
;   0107  2       RETURN (1);                                                                    !G:2
;   0108  1     END;



                                                0000 00000        .ENTRY  DSX$SHOWIDLE, Save nothing                       ; 0105
                                   50        01  D0 00002        MOVL    #1, R0                                           ; 0107
                                                04 00005        RET                                                      ; 0108

; Routine Size:  6 bytes,    Routine Base:  $CODE$ + 0018


;   0109  1
;   0110  1     GLOBAL ROUTINE MP$_DEALL_MEMORY : NOVALUE =
;   0111  2     BEGIN
;   0112  2       RETURN (1);                                                                    !G:2
;   0113  1     END;
```

```
                                                0000 00000          .ENTRY  MP$_DEALL_MEMORY, Save nothing                  ; 0110
                                                  04 00002          RET                                                     ; 0113

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 001E


;   0114  1
;   0115  1     GLOBAL ROUTINE MP$_GET_PROCESSOR_ID (THIS_PROCESSOR_ID) =
;   0116  2     BEGIN
;   0117  2      .THIS_PROCESSOR_ID = 1; ! For non mp systems, 1 will always be returned for id.
;   0118  1     END;
; INFO#212        L1:0117
; Null expression appears in value-required context




                                                0000 00000          .ENTRY  MP$_GET_PROCESSOR_ID, Save nothing              ; 0115
                          04   BC               01  D0 00002          MOVL    #1, @THIS_PROCESSOR_ID                          ; 0117
                                               50  D4 00006          CLRL    R0                                              ; 0118
                                                  04 00008          RET                                                     ;

; Routine Size:  9 bytes,    Routine Base:  $CODE$ + 0021


;   0119  1
;   0120  1     GLOBAL ROUTINE MP$_STOP_EVERYTHING : NOVALUE =
;   0121  2     BEGIN
;   0122  2       RETURN (1);                                                                                   !G:2
;   0123  1     END;




                                                0000 00000          .ENTRY  MP$_STOP_EVERYTHING, Save nothing               ; 0120
                                                  04 00002          RET                                                     ; 0123

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 002A


;   0124  1
;   0125  1     GLOBAL ROUTINE MP$_ATTACHED_CPU_CHECK : NOVALUE =
;   0126  2     BEGIN
;   0127  2       RETURN (1);                                                                                   !G:2
;   0128  1     END;




                                                0000 00000          .ENTRY  MP$_ATTACHED_CPU_CHECK, Save nothing            ; 0125
                                                  04 00002          RET                                                     ; 0128

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 002D
```

```
;   0129  1
;   0130  1     GLOBAL ROUTINE MP$_PRIMARY_CPU_CHECK : NOVALUE =
;   0131  2     BEGIN
;   0132  2       RETURN (1);                                                                    !G:2
;   0133  1     END;



                                                0000 00000        .ENTRY  MP$_PRIMARY_CPU_CHECK, Save nothing          ; 0130
                                                  04 00002        RET                                                  ; 0133

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 0030


;   0134  1
;   0135  1     GLOBAL ROUTINE MP$_RESUME_ATTACHED_PROCESSORS : NOVALUE =
;   0136  2     BEGIN
;   0137  2       RETURN (1);                                                                    !G:2
;   0138  1     END;



                                                0000 00000        .ENTRY  -                                            ; 0135
                                                                          MP$_RESUME_ATTACHED_PROCESSORS, Save nothin-;
                                                                          g                                            ;
                                                  04 00002        RET                                                  ; 0138

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 0033


;   0139  1
;   0140  1     GLOBAL ROUTINE DSX$_SET_INTERLOCK: NOVALUE =
;   0141  2     BEGIN
;   0142  2       RETURN (1);                                                                    !G:2
;   0143  1     END;



                                                0000 00000        .ENTRY  DSX$_SET_INTERLOCK, Save nothing             ; 0140
                                                  04 00002        RET                                                  ; 0143

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 0036


;   0144  1
;   0145  1     GLOBAL ROUTINE DSX$_CLEAR_INTERLOCK: NOVALUE =
;   0146  2     BEGIN
;   0147  2       RETURN (1);                                                                    !G:2
;   0148  1     END;
```

```
                                                0000 00000          .ENTRY  DSX$_CLEAR_INTERLOCK, Save nothing          ; 0145
                                                  04 00002          RET                                                 ; 0148

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 0039

;   0149  1
;   0150  1     END
;   0151  0     ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $GLOBAL$                          4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                           60  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...
BITMAP	RoutForm	105	
DSX$BOOTATTACHED	GlobRout	90	
DSX$HALTATTACHED	GlobRout	100	
DSX$SHOWIDLE	GlobRout	105	
DSX$STARTATTACHED	GlobRout	95	
DSX$_CLEAR_INTERLOCK	GlobRout	145	
DSX$_INTERLOCK_TEST	GlobRout	85	
DSX$_SET_INTERLOCK	GlobRout	140	
GLOBAL_TEST	RoutForm	85	
LOCAL_TEST	RoutForm	85	
MP$GL_CONDITION_FLAGS	Global	75	75=
MP$_ATTACHED_CPU_CHECK	GlobRout	125	
MP$_DEALL_MEMORY	GlobRout	110	
MP$_GET_PROCESSOR_ID	GlobRout	115	
MP$_PRIMARY_CPU_CHECK	GlobRout	130	
MP$_RESUME_ATTACHED_PROCESSORS	GlobRout	135	
MP$_STOP_EVERYTHING	GlobRout	120	
SCBADDR	RoutForm	90	
START_ADDR	RoutForm	95	
THIS_PROCESSOR_ID	RoutForm	115	117a=
UNIT	RoutForm	90	
	RoutForm	95	
	RoutForm	100	

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]MPDUMMY.B32;2
6	Module MPDUMMY	
68	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
70	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
72	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
151	Eludom MPDUMMY	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
@	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

; Library Statistics
;

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	0	0	96	00:00.0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	0	0	47	00:00.0
SYS$COMMON:[SYSLIB]LIB.L32;2	20450	0	0	1101	00:00.9

; Information: 1
; Warnings: 0
; Errors: 0

; COMMAND QUALIFIERS

; BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DSRW:/LIST=DSRW: DSRW:MPDUMMY.B32

; Size: 60 code + 4 data bytes
; Run Time: 00:01.5
; Elapsed Time: 00:03.9
; Lines/CPU Min: 5921
; Lexemes/CPU-Min: 16078
; Memory Used: 51 pages
; Compilation Complete

Table of contents

(2) 97 DECLARATIONS
(3) 246 TU81 load device initialization
(4) 392 TU81 load driver QIO

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element MUBTDRIVR.MAR
0000     2 ; *6    15-MAY-1986 15:30:24 MUSE "<no reason given>"
0000     3 ; *5    15-MAY-1986 15:27:15 MUSE "<no reason given>"
0000     4 ; *4    15-MAY-1986 14:10:27 MUSE "<no reason given>"
0000     5 ; *3    13-MAY-1986 15:30:03 MUSE "<no reason given>"
0000     6 ; *2    13-MAY-1986 15:17:55 MUSE "<no reason given>"
0000     7 ; *1     6-FEB-1986 15:20:47 DS ""
0000     8 ; DEC/CMS REPLACEMENT HISTORY, Element MUBTDRIVR.MAR
0000     9         .TITLE  MUBTDRIVR - TU81 LOAD DRIVER
0000    10         .IDENT  '06-12'
0000    11         .DISABLE GBL
0000    12 ;
0000    13 ;*****************************************************************************
0000    14 ;*                                                                           *
0000    15 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1983, 1984 BY                           *
0000    16 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    17 ;*  ALL RIGHTS RESERVED.                                                     *
0000    18 ;*                                                                           *
0000    19 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    20 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    21 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    22 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    23 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    24 ;*  TRANSFERRED.                                                             *
0000    25 ;*                                                                           *
0000    26 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    27 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    28 ;*  CORPORATION.                                                             *
0000    29 ;*                                                                           *
0000    30 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    31 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    32 ;*                                                                           *
0000    33 ;*                                                                           *
0000    34 ;*****************************************************************************
0000    35 ;
0000    36
0000    37 ;++
0000    38 ; FACILITY:     VDS
0000    39 ;
0000    40 ; ABSTRACT:
0000    41 ;       This module contains the load driver for the
0000    42 ;       TU81 magtape.
0000    43 ;
0000    44 ; ENVIRONMENT: kernel mode
0000    45 ;
0000    46 ; AUTHOR:  Marion Baggett
0000    47 ;
0000    48 ; MODIFIED BY:
0000    49 ;
0000    50 ;
0000    51 ;       01      M. Baggett      VDS 6.10
0000    52 ;               Change error reporting for command execution failure.
0000    53 ;
0000    54 ;       02      M. Baggett      VDS 6.10
0000    55 ;               Change ID field in message buffer to tape value.
0000    56 ;
0000    57 ;       03      M. Baggett      VDS 6.11
```

```
0000     58 ;                Interchanged the offsets names in the skip record and skip file
0000     59 ;                sections for marks to skip.
0000     60 ;
0000     61 ;        04      M. Baggett      VDS 6.11 Jan-29-1983
0000     62 ;                Change the skip file section to return a normal code on
0000     63 ;                detection of tape marks.
0000     64 ;
0000     65 ;        05      M. Baggett      VDS 6.11 Feb-23-1983
0000     66 ;                Changed symbol BTD$K_TU to BTD$K_MU for TU81.
0000     67 ;
0000     68 ;        06      M. Baggett      VDS 6.11 June-29-1983
0000     69 ;                Added GET UNIT STATUS to supply values for ONLINE command.
0000     70 ;
0000     71 ;        07      BOB BERGAZZI    6.12    July 18, 1983
0000     72 ;                Changed b^VALID to W^VALID to resolve truncation errors.
0000     73 ;
0000     74 ;        08      M. Baggett      VDS 6.13 Oct 24, 1983
0000     75 ;                Correct displacement error in ONLINE command packet.
0000     76 ;                Set CLEAR SERIOUS EXCEPTION bit in several commands.
0000     77 ;
0000     78 ;        09      M. BAGGETT      VDS 6.14 Jan 5, 1984
0000     79 ;                Definition of the internal processors registers are temporarily
0000     80 ;                defined in this module.
0000     81 ;
0000     82 ;        10      M. BAGGETT      VDS 7.1 4-OCT-1984
0000     83 ;                Change skipfile routine to skip records until tape mark.
0000     84 ;
0000     85 ;        11      M. BAGGETT      VDS 7.1 25-OCT-1984
0000     86 ;                MSCP definitions in library are not defined in program.
0000     87 ;
0000     88 ;        12      M. BAGGETT      VDS 7.1 2-NOV-1984
0000     89 ;                Only init device if error or at BOT. Remove
0000     90 ;                change [10].
0000     91 ;                                                                        ;G:2
0000     92 ;        13      R.E.Muse        13-MAY-1986     VERSION 10.1            ;G:2
0000     93 ;                installed the TIMEDWAIT macro in place of reference's   ;G:2
0000     94 ;                to TODR.                                                ;G:2
0000     95 ;--
```

```
                        0000     97          .SBTTL  DECLARATIONS                                                    ;G:2
                        0000     98                                                                                  ;G:2
                        0000     99  ;+                                                                              ;G:2
                        0000    100  ;       external labels                                                         ;G:2
                        0000    101  ;-                                                                              ;G:2
                        0000    102                                                                                  ;G:2
                        0000    103  .extrn  EXE$GL_UBDELAY,EXE$GL_TENUSEC                                           ;G:2
                        0000    104                                                                                  ;G:2
                        0000    105  ;
                        0000    106  ; INCLUDE FILES:
                        0000    107  ;
                        0000    108
                        0000    109          $BTDDEF                                 ; Boot device types
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    110          $IODEF                                  ; I/O function codes
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    111          $MSCPDEF                                ; MSCP definitions
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    112          $PTEDEF                                 ; Page table entries
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    113          $RPBDEF                                 ; RPB offsets
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    114          $SSDEF                                  ; Status codes
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    115          $UBADEF                                 ; UBA definitions
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    116          $UBIDEF                                 ; 11/750 UBA definitions
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    117          $VADEF                                  ; Virtual addresses
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .RESTORE
                        0000    118
                        0000    119  ;
                        0000    120  ; EQUATED SYMBOLS:
                        0000    121  ;
000000B1                0000    122          BTD$K_TM        = 177   ; TEMP
000000B2                0000    123          BTD$K_TS        = 178   ; TEMP
000000B3                0000    124          BTD$K_TF        = 179   ; TEMP
000000B4                0000    125          BTD$K_MU        = 180   ; TEMP                                          [05]
                        0000    126
00000000                0000    127          TUIP    = 0
00000002                0000    128          TUSA    = 2
00000001                0000    129          GO      = 1
00008000                0000    130          OWN     = 1@15
0000000B                0000    131          S1      = 11
0000000E                0000    132          S4      = 14
000001EE                0000    133          UMR     = 494                   ; Last-1 UNIBUS Mapping Register
                        0000    134          .MACRO  $PRDEF,$GBL
                        0000    135
```

```
0000   136          $DEFINI PR,$GBL
0000   137
0000   138
0000   139  $EQU    PR$_KSP 0
0000   140  $EQU    PR$_ESP 1
0000   141  $EQU    PR$_SSP 2
0000   142  $EQU    PR$_USP 3
0000   143  $EQU    PR$_ISP 4
0000   144  $EQU    PR$_P0BR          8
0000   145  $EQU    PR$_P0LR          9
0000   146  $EQU    PR$_P1BR          10
0000   147  $EQU    PR$_P1LR          11
0000   148  $EQU    PR$_SBR 12
0000   149  $EQU    PR$_SLR 13
0000   150  $EQU    PR$_PCBB          16
0000   151  $EQU    PR$_SCBB          17
0000   152  $EQU    PR$_IPL 18
0000   153  $EQU    PR$_ASTLVL        19
0000   154  $EQU    PR$_SIRR          20
0000   155  $EQU    PR$_SISR          21
0000   156  $EQU    PR$_ICCS          24
0000   157  $EQU    PR$_NICR          25
0000   158  $EQU    PR$_ICR 26
0000   159  $EQU    PR$_TODR          27
0000   160  $EQU    PR$_RXCS          32
0000   161  $EQU    PR$_RXDB          33
0000   162  $EQU    PR$_TXCS          34
0000   163  $EQU    PR$_TXDB          35
0000   164  $EQU    PR$_ACCS          40
0000   165  $EQU    PR$_ACCR          41
0000   166  $EQU    PR$_MAPEN         56
0000   167  $EQU    PR$_TBIA          57
0000   168  $EQU    PR$_TBIS          58
0000   169  $EQU    PR$_PME 61
0000   170  $EQU    PR$_SID 62
0000   171  $EQU    PR$_TBCHK         63
0000   172  $EQU    PR$V_SID_SN       0
0000   173  $EQU    PR$S_SID_SN       12
0000   174  $EQU    PR$V_SID_PL       12
0000   175  $EQU    PR$S_SID_PL       3
0000   176  $EQU    PR$V_SID_ECO      15
0000   177  $EQU    PR$S_SID_ECO      9
0000   178  $EQU    PR$V_SID_TYPE     24
0000   179  $EQU    PR$S_SID_TYPE     8
0000   180  $EQU    PR$_SID_TYP780    1
0000   181  $EQU    PR$_SID_TYP750    2
0000   182  $EQU    PR$_SID_TYP730    3
0000   183  $EQU    PR$_SID_TYP7VV    4
0000   184  $EQU    PR$_SID_TYPMAX    4
0000   185  $EQU    PR$_WCSA          44
0000   186  $EQU    PR$_WCSD          45
0000   187  $EQU    PR$_SBIFS         48
0000   188  $EQU    PR$_SBIS          49
0000   189  $EQU    PR$_SBISC         50
0000   190  $EQU    PR$_SBIMT         51
0000   191  $EQU    PR$_SBIER         52
0000   192  $EQU    PR$_SBITA         53
```

```
                     0000   193  $EQU    PR$_SBIQC       54
                     0000   194  $EQU    PR$_CMIERR      23
                     0000   195  $EQU    PR$_CSRS        28
                     0000   196  $EQU    PR$_CSRD        29
                     0000   197  $EQU    PR$_CSTS        30
                     0000   198  $EQU    PR$_CSTD        31
                     0000   199  $EQU    PR$_TBDR        36
                     0000   200  $EQU    PR$_CADR        37
                     0000   201  $EQU    PR$_MCESR       38
                     0000   202  $EQU    PR$_CAER        39
                     0000   203  $EQU    PR$_UBRESET     55
                     0000   204  $EQU    PR$_PAMACC      64
                     0000   205  $EQU    PR$_PAMLOC      65
                     0000   206  $EQU    PR$_CSWP        66
                     0000   207  $EQU    PR$_CRBT        67
                     0000   208  $EQU    PR$_MCTL1       68
                     0000   209  $EQU    PR$_MCTL2       69
                     0000   210  $EQU    PR$_MGEN        70
                     0000   211  $EQU    PR$_MTBER       71
                     0000   212  $EQU    PR$_MEAR        72
                     0000   213  $EQU    PR$_MDCR1       73
                     0000   214  $EQU    PR$_MEDR        74
                     0000   215  $EQU    PR$_MECCR       75
                     0000   216
                     0000   217          $DEFEND PR,$GBL,DEF
                     0000   218
                     0000   219          .ENDM   $PRDEF
                     0000   220          $PRDEF                          ; Processor registers
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .RESTORE
                     0000   221
                     0000   222 ;
                     0000   223 ; OWN STORAGE:
                     0000   224 ;
                0000000A  0000   225 MSCP$K_OP_SETUC=10
                     0000   226 ;
                     0000   227 ; Boot driver table entry
                     0000   228 ;
                     0000   229
                00000000  0000   230 .Psect  Bootdrivr_2,    page    ; Define psect allocation
                     0000   231
                     0000   232          $BOOT_DRIVER    DEVTYPE = BTD$K_MU,-    ; Device type (TU81)          [05]
                     0000   233                          SIZE = TU_DRVSIZ,-      ; Driver size
                     0000   234                          ADDR = START,-          ; Driver starting address
                     0000   235                          ENTRY = TU_DRIVER       ; Driver entry point
                00000000                .PSECT  BOOTDRIVR_4
                    FFFF  0000          .WORD   -1
                    00B4  0002          .WORD   BTD$K_MU
                00000000  0004          .LONG   0
                000003E3' 0008          .LONG   TU_DRVSIZ
                00000000' 000C          .LONG   START-$TABLE
                00000280' 0010          .LONG   TU_DRIVER-START
                00000000' 0014          .LONG   -START
                00000000                .PSECT  BOOTDRIVR_2
                     0000   236 ;                          UNIT_INIT = TU_INIT,-   ; Driver unit init entry
                     0000   237 ;                          DRIVRNAME = TUPDRVNAME,-; Driver tape name
                     0000   238 ;                          AUXDRNAME = PRTDRVNAME  ; Driver port name
```

```
                        0000   239
                        0000   240 START:
                        0000   241 ;TUPDRVNAME:
                        0000   242 ;       .ASCIC  /TUDRIVER.EXE/          ; Tape class driver filename
                        0000   243 ;PRTDRVNAME:
                        0000   244 ;       .ASCIC  /PTDRIVER.EXE/          ; Port driver filename
```

```
                                0000   246          .SBTTL  TU81 load device initialization
                                0000   247
                                0000   248 ;++
                                0000   249 ;
                                0000   250 ; Inputs:
                                0000   251 ;
                                0000   252 ;       R9 --> RPB
                                0000   253 ;
                                0000   254 ; Outputs:
                                0000   255 ;
                                0000   256 ;       R0 - status code
                                0000   257 ;
                                0000   258 ;--
                                0000   259
                         01FC   0000   260 .Entry TU_INIT, ^M<R2,R3,R4,R5,R6,R7,R8>
                                0002   261
                                0002   262          .ENABLE LSB
                                0002   263
                50   38   DB   0002   264          MFPR    #PR$_MAPEN, R0          ; Get the mapping status
                                0005   265 ;
                                0005   266 ; Set up a UNIBUS mapping register(s) to cover the ring and buffers.
                                0005   267 ; To make things easy, we will grab the last two register (494 & 495).
                                0005   268 ; These registers are necessary since the ring area will be accessed by
                                0005   269 ; the controller which is a UNIBUS device that does not do any mapping.
                                0005   270 ;
          56   0003DC00 8F   D0   0005   271          MOVL    #<UMRa9>, R6            ; Set up a constant
               52   0200'CF   9E   000C   272          MOVAB   W^INTTBL, R2            ; Get the address of the ring
     51   52   00000098'EF   CB   0011   273          BICL3   BYTE_OFF, R2, R1        ; Get the byte offset in page          ;G:3
           02 A2   51   56   C9   0019   274          BISL3   R6, R1, 2(R2)           ; Set in the UNIBUS addr
                 02 A2   10'   C0   001E   275          ADDL    S^#<RING-INTTBL>,2(R2)  ; Make it the ring address
          52   52   15   09   EF   0022   276          EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R2 ; Get the page frame
                                0027   277          ASSUME  RPB$L_ADPVIR EQ RPB$L_ADPPHY+4
               53   5C A940   D0   0027   278          MOVL    RPB$L_ADPPHY(R9)[R0],R3 ; Get correct pointer to TU81 reg
                                002C   279          ASSUME  RPB$L_CSRVIR EQ RPB$L_CSRPHY+4
               57   54 A940   D0   002C   280          MOVL    RPB$L_CSRPHY(R9)[R0],R7 ; Get correct address of device CSR
                      0C 50   E9   0031   281          BLBC    R0,20$                  ; If clr, then physical
               52   50 B942   D0   0034   282          MOVL    @RPB$L_SVASPT(R9)[R2],R2; Virtual, get physical
          52   FFE00000 8F   CA   0039   283          BICL    #^C<PTE$M_PFN>,R2       ; Now a physical PFN
               54   0FB8 C3   DE   0040   284 20$:     MOVAL   UBA$L_MAP+<UMR*4>(R3),R4; Get the last two UMR's
          84   52   010A'CF   C9   0045   285          BISL3   W^VALID,R2,(R4)+        ; Set as valid w/PFN                   [07]
                         52   D6   004B   286          INCL    R2                      ; Set next page just in case
          64   52   010A'CF   C9   004D   287          BISL3   W^VALID,R2,(R4)         ; Set as valid w/PFN                   [07]
                                0053   288 ;
                                0053   289 ; Now go thru the ridiculously complicated startup sequence.  This is a
                                0053   290 ; fugue in four parts.
                                0053   291 ;
               58   02   D0   0053   292          MOVL    #2,R8                   ; Make two tries at this
               53   0200'CF   9E   0056   293 RETRY:  MOVAB   W^INTTBL, R3
                    52   0B   D0   005B   294          MOVL    #S1, R2                 ; Step flag                            ;G:4
                         67   B4   005E   295          CLRW    TUIP(R7)                ; Poke the controller's CSR
                                0060   296 ;
                                0060   297 ; Wait 10 seconds.                                                              ;G:2
                                0060   298 ;
                                0060   299 TIME:                                                                          ;G:2
                                0060   300          TIMEDWAIT TIME=#1000*1000,-             ; wait for 10 seconds   [13]  ;G:2
                                0060   301                    INS1    = <MOVW TUSA(R7),R4>,-  ; check the setup register    ;G:2
                                0060   302                    INS2    = <BLSS 25$>,-          ; bit 15 set ?                 ;G:2
```

```
                                0060   303                  INS3    = <BBS  R2,R4,25$>,-    ; done with this step ?       ;G:2
                                0060   304                  DONELBL = 25$                   ; label for exiting wait loop ;G:2
                     50   01    3C   0060                  MOVZWL  #SS$_NORMAL,R0
51   00000000'GF    000F4240 8F   C5   0063                  MULL3   #1000*1000,G^EXE$GL_TENUSEC,R1
                              7E   D4   006F                  CLRL    -(SP)
                                0071        30000$:
                      54   02 A7   B0   0071                  MOVW TUSA(R7),R4
                                13   19   0075                  BLSS    25$
                      0F 54     52   E0   0077                  BBS     R2,R4,25$
                6E    00000000'GF   D0   007B                  MOVL    G^EXE$GL_UBDELAY,(SP)
                           FD 6E    F5   0082        30001$:   SOBGTR  (SP),30001$
                           E9 51    F5   0085                  SOBGTR  R1,30000$
                              50    D4   0088                  CLRL    R0
                                008A        25$:
                              8E    D5   008A                  TSTL    (SP)+
                           0D 50    E8   008C   305            BLBS    R0,26$                  ; br if not timed out         ;G:2
                                008F   306                                                                                  ;G:2
                           C4 58    F5   008F   307 ERROR:     SOBGTR  R8,RETRY                ; Try once again
                     50    0054 8F   3C   0092   308            MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                              04   0097   309                  RET
                                0098   310 BYTE_OFF:
                          FFFFFE00   0098   311                .LONG   ^C<^X1FF>               ; Mask for byte offset in page
                                009C   312
                           54   B5   009C   313 26$:     TSTW    R4                      ; check status register for error [13] ;G:2
                           EF   19   009E   314            BLSS    ERROR                   ; br if error                 [13]    ;G:
                                00A0   315                                                                                   ;G:2
                      02 A7     83   B0   00A0   316 30$:     MOVW    (R3)+,TUSA(R7)          ; Send the controller the next step
                      B8 52     0E   F3   00A4   317            AOBLEQ  #S4,R2,TIME             ; Set for next step           ;G:5
                                00A8   318 ;
                                00A8   319 ; Initialization complete.  Write the packet address in the ring.
                                00A8   320 ;
                   51   0250'CF   9E   00A8   321                MOVAB   W^RSPPKT, R1            ; Get the address of response packet
                     51    E8 AF   CA   00AD   322                BICL    BYTE_OFF, R1
          0210'CF    51     56   C9   00B1   323                BISL3   R6, R1, W^RD            ; Set byte offset in ring desc
                   51   021C'CF   9E   00B7   324                MOVAB   W^CMDPKT, R1            ; Get the address of command packet
                   55     51   D0   00BC   325                MOVL    R1,R5                   ; Save pointer
                     51    D6 AF   CA   00BF   326                BICL    BYTE_OFF, R1
          0214'CF    51     56   C9   00C3   327                BISL3   R6, R1,W^CD             ; Set byte offset in ring desc
                                00C9   328
                                00C9   329
                                00C9   330 ;
                                00C9   331 ; Now bring the device on-line
                                00C9   332 ;
                      85     01   D0   00C9   333                MOVL    #1,(R5)+                ; Set command ref number
                        85    64 A9   9A   00CC   334                MOVZBL  RPB$W_UNIT(R9),(R5)+    ; Put unit number in cmd packet field
                85   20000009 8F   D0   00D0   335                MOVL    #<MSCP$M_MD_CLSEX@16>!MSCP$K_OP_ONLIN,(R5)+ ; opcode online     [08]
                      85       D5   00D7   336                TSTL    (R5)+                   ; Increment register          [06]
                                00D9   337                                                   ; Set characteristics         [06]
                      85       7C   00D9   338                CLRQ    (R5)+                   ; Clear reserved area
                      85       D4   00DB   339                CLRL    (R5)+                   ; Set default device dependent parameter[06]
                         0286  30   00DD   340                BSBW    EXECUTE                 ; Send it out
                                00E0   341 ;
                                00E0   342 ; Get unit characteristics
                                00E0   343 ;
                   55   021C'CF   9E   00E0   344                MOVAB   W^CMDPKT, R5            ; Get the address of command packet     [12]
                      85     01   D0   00E5   345                MOVL    #1,(R5)+                ; Set command ref number                [06]
                        85    64 A9   9A   00E8   346                MOVZBL  RPB$W_UNIT(R9),(R5)+    ; Put unit number in cmd packet field   [06]
```

```
                      65   03    9A  00EC   347          MOVZBL  #MSCP$K_OP_GTUNT,(R5)   ; Set opcode to get unit status          [06]
                         0274    30  00EF   348          BSBW    EXECUTE                   ; Send it out                            [06]
                                     00F2   349
    0000010F'EF   0000025E'EF    9B  00F2   350          MOVZBW  RSPPKT+MSCP$W_UNT_FLGS, UNIT_FLAGS     ; SAVE UNIT FLAGS       [12]
    00000113'EF   00000270'EF    D0  00FD   351          MOVL    RSPPKT+MSCP$W_FORMAT, FORMAT           ; SAVE FORMAT           [12]
                                 04  0108   352          RET
                                     0109   353
                                     0109   354
                                     0109   355          .DISABLE LSB
                          0000010A   0109   356          .=<.+1>&-2
                          80000000   010A   357 VALID:   .LONG   ^X80000000              ; SIgn bit set
                                00   010E   358 INIT:    .byte   0                     ; init flag                              [12]
                                     010F   359 UNIT_FLAGS:
                          00000000   010F   360          .LONG   0                     ; UNIT FLAGS                             [12]
                                     0113   361 FORMAT:
                          00000000   0113   362          .LONG   0                     ; SPEED AND FORMAT                       [12]
                                     0117   363 DEVICE_CSR:
                          0000011B   0117   364          .BLKL   1                     ; Device csr from last initialization    [12]
                                     011B   365 DEVICE_UNIT:
                          0000011D   011B   366          .BLKW   1                     ; Device unit number from last init      [12]
                                     011D   367 ;
                                     011D   368 ; RINGS
                                     011D   369 ;
                                     011D   370 ;
                                     011D   371 .Align  Page
                                     0200   372
                              8000   0200   373 INTTBL: .WORD   OWN                 ; Step 1 pattern
                          00000000   0202   374          .LONG   0                 ; Step 2 & 3 pattern
                              0001   0206   375          .WORD   GO
                                     0208   376
                         0000 0000   0208   377          .WORD   0,0               ; Reserved
                              0000   020C   378 CMDINT:  .WORD   0                 ; Command status word
                              0000   020E   379 RSPINT:  .WORD   0                 ; Response status word
                                     0210   380 RING:
                          00000000   0210   381 RD:      .LONG   0                 ; UNIBUS address of response ring
                          00000000   0214   382 CD:      .LONG   0                 ; UNIBUS address of command ring
                                     0218   383 ;
                              0030   0218   384          .WORD   48                ; Length of message
                              0101   021A   385          .WORD   ^X101             ; Tape ID = 1                                [02]
                              0001   021C   386 CMDPKT:  .WORD   1
                          0000024C   021E   387          .BLKW   23                ; Full envelope
                                     024C   388 ;
                          00000250   024C   389          .BLKW   2
                          00000280   0250   390 RSPPKT:  .BLKW   24
```

```
                     0280   392          .SBTTL  TU81 load driver QIO
                     0280   393
                     0280   394 ;++
                     0280   395 ;
                     0280   396 ; Inputs:
                     0280   397 ;
                     0280   398 ;       R3      - base address of adapter's register space
                     0280   399 ;       R5      - lbn for current piece of transfer
                     0280   400 ;       R6      - contains 0
                     0280   401 ;       R7      - address of the device's CSR
                     0280   402 ;       R8      - size of transfer in bytes
                     0280   403 ;       R9      - address of the RPB
                     0280   404 ;       R10     - starting address of transfer (byte offset in first
                     0280   405 ;                 page ORed with starting map register number)
                     0280   406 ;       R11     - LBN at start of transfer
                     0280   407 ;
                     0280   408 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                     0280   409 ;       SIZE(AP)- Size of transfer in bytes
                     0280   410 ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
                     0280   411 ;       BUF(AP) - Address of buffer
                     0280   412 ;
                     0280   413 ; Implicit inputs:
                     0280   414 ;
                     0280   415 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                     0280   416 ;
                     0280   417 ; Outputs:
                     0280   418 ;
                     0280   419 ;       R0 - status code
                     0280   420 ;
                     0280   421 ;               SS$_NORMAL      - successful transfer
                     0280   422 ;               SS$_CTRLERR     - fatal controller error
                     0280   423 ;               SS$_TAPEPOSLOST - tape position lost
                     0280   424 ;               SS$_DATAOVERUN  - record too long
                     0280   425 ;               SS$_ILLIOFUNC   - illegal I/O function
                     0280   426 ;               SS$_CTRLERR     - controller error
                     0280   427 ;               SS$_ENDOFFILE   - tape mark was read
                     0280   428 ;               SS$_DEVOFFLINE  - device is offline
                     0280   429 ;
                     0280   430 ;       R1 -    Count of bytes transferred for READ operations.
                     0280   431 ;
                     0280   432 ;       R3 - must be preserved
                     0280   433 ;
                     0280   434 ;
                     0280   435 ;--
00000004             0280   436 BUF   = 4
00000008             0280   437 SIZE  = 8
0000000C             0280   438 LBN   = 12
00000010             0280   439 FUNC  = 16
00000014             0280   440 MODE  = 20
                     0280   441
                     0280   442 TU_DRIVER:                                ; TU81 device driver.
                     0280   443
                     0280   444
                     0280   445 ; Translate the I/O function code into a device-dependent function
                     0280   446 ; code for this tape.
                     0280   447 ;
04 AE    D4          0280   448         CLRL    4(SP)                    ; Clear retries count ********          [06]
```

```
                21   10 AC   D1  0283   449          CmpL    Func(AP), #IO$_ReadLBlk ; Read logical block?
                       64   13  0287   450          BEql    E_Read                  ; Execute read function
                24   10 AC   D1  0289   451          CmpL    Func(AP), #IO$_Rewind   ; Rewind?
                       4A   13  028D   452          BEql    E_Rewind                ; Do a rewind
                25   10 AC   D1  028F   453          CmpL    Func(AP), #IO$_SkipFile ; Skip past eof
                       77   13  0293   454          BEql    E_SkipFile              ; Do it
                26   10 AC   D1  0295   455          CmpL    Func(AP), #IO$_SkipRecord; Skip records?
                       03   12  0299   456          BNEQ    10$                     ; Branch to check next function          [08]
                     009F   31  029B   457          BRW     E_SkipRecord            ; Do it                                  [08]
                04   10 AC   D1  029E   458 10$:     CmpL    Func(AP), #IO$_DrvClr   ; Is this drive clear?
                       30   12  02A2   459          Bneq    20$                     ; Yes, all done, exit
              FE6E CF  57   D1  02A4   460          CMPL    R7,DEVICE_CSR           ; IS THIS THE SAME CSR AS LAST TIME ?    [12]
                       0F   12  02A9   461          BNEQ    15$                     ; GO RE-INIT IF NOT                      [12]
              FE6B CF  53   B1  02AB   462          CMPW    R3,DEVICE_UNIT          ; IS THIS THE SAME UNIT AS LAST TIME ?   [12]
                       08   12  02B0   463          BNEQ    15$                     ; GO RE-INIT IF NOT                      [12]
                   50  01   D0  02B2   464          MOVL    #SS$_NORMAL,R0          ; ASSUME NO ERROR                        [12]
                1E FE55 CF   E8  02B5   465          BLBS    W^INIT,RETURN           ; EXIT IF NO NEED TO INIT                [12]
                                02BA   466 ;
                                02BA   467 ; Start out by initing the TU81
                                02BA   468 ;
              FD41 CF  00   FB  02BA   469 15$:     CallS   #0, TU_Init             ; Call init code
                     16 50   E9  02BF   470          BLBC    R0,RETURN               ; EXIT ON ERROR                          [12]
                FE50 CF  57   D0  02C2   471          MOVL    R7,DEVICE_CSR           ; SAVE CSR FOR THIS DEVICE               [12]
             FE4E CF   64 A9   B0  02C7   472          MOVW    RPB$W_UNIT(R9),DEVICE_UNIT ; SAVE UNIT # FOR THIS DEVICE        [12]
                FE3C CF  01   90  02CD   473          MOVB    #1,W^INIT               ; SET FLAG                               [12]
                                02D2   474 ;
                       04   11  02D2   475          Brb     Return                  ; Exit
                50   F4 8F   9A  02D4   476 20$:     MovZBL  #SS$_IllIOFunc, R0      ; illegal I/O function
                                02D8   477 Return:
                                05  02D8   478          RSB
                                02D9   479 ;+
                                02D9   480 ; Rewind slave drive, wait for completion
                                02D9   481 ;-
                                02D9   482 E_REWIND:
                     25   90  02D9   483          MOVB    #MSCP$K_OP_REPOS,-      ; Set function reposition
                FF46 CF          02DB   484                  CMDPKT+MSCP$B_OPCODE
                2002 8F  B0  02DE   485          MOVW    #MSCP$M_MD_CLSEX!MSCP$M_MD_REWND,- ; Set to rewind/clear serious exc
                FF41 CF          02E2   486                  CMDPKT+MSCP$W_MODIFIER  ;
                FF3F CF  7C  02E5   487          CLRQ    CMDPKT+MSCP$L_REC_CNT   ; Clear records/tape marks to avoid errors.
                   007A  30  02E9   488          BSBW    EXECUTE                 ; Execute function.
                                05  02EC   489          RSB                     ; Exit.
                                02ED   490 ;+
                                02ED   491 ; Read block of data
                                02ED   492 ;-
                                02ED   493 E_READ:
                     21   90  02ED   494          MOVB    #MSCP$K_OP_READ,-       ; Set function to read
                FF32 CF          02EF   495                  CMDPKT+MSCP$B_OPCODE    ;
FF2D CF         2000 8F  B0  02F2   496          MOVW    #MSCP$M_MD_CLSEX,CMDPKT+MSCP$W_MODIFIER ; Clear serious except  [08]
     FF2A CF    58   D0  02F9   497          MOVL    R8,CMDPKT+MSCP$L_BYTE_CNT ; Set byte count.
     FF29 CF    5A   D0  02FE   498          MOVL    R10,CMDPKT+MSCP$B_BUFFER ; Set buffer address.
                   0060  30  0303   499          BSBW    EXECUTE                 ; Execute function.
     51    FF52 CF     D0  0306   500          MOVL    RSPPKT+MSCP$L_BYTE_CNT, R1 ; Get # of bytes transferred         [12]
                                05  030B   501          RSB                     ; EXIT
                                030C   502 ;
                                030C   503 ;
                                030C   504 ;+
                                030C   505 ; Skip to end of file
```

```
                        030C   506 ; P1 is negative if backwards else forwards
                        030C   507 ;-
                        030C   508 E_SKIPFILE:
                  25    90   030C   509          MOVB    #MSCP$K_OP_REPOS,-      ; Set function to reposition.
              FF13 CF          030E   510                  CMDPKT+MSCP$B_OPCODE
              2080 8F    B0    0311   511          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_CLSEX,-  ; Detect LEOT and          [08]
              FF0E CF          0315   512                                  CMDPKT+MSCP$W_MODIFIER  ; Clear serious except  [08]
           50   04 AC    D0    0318   513          MOVL    BUF(AP),R0              ; Count of MARKS to skip.               [12]
                0A    14    031C   514          BGTR    10$                     ; Branch if positive.
              50   50    CE    031E   515          MNEGL   R0,R0                   ; Make count positive.                  [12]
              2088 8F    B0    0321   516          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_REVRS!MSCP$M_MD_CLSEX,-; Detect LEOT [08]
              FEFE CF          0325   517                  CMDPKT+MSCP$W_MODIFIER  ; Reverse/clear serious except  [08]
              FEFC CF    D4    0328   518 10$:     CLRL    CMDPKT+MSCP$L_REC_CNT   ; CLEAR record FIELD            [12]
        FEFB CF   50    D0    032C   519          MOVL    R0,CMDPKT+MSCP$L_TMGP_CNT       ; SET field             [12]
                0032    30    0331   520          BSBW    EXECUTE                 ; Execute function.
                0E   51    D1    0334   521          CMPL    R1, #MSCP$K_ST_TAPEM    ; Was tape mark detected        [04]
                   03    12    0337   522          BNEQ    20$                     ;                               [12]
              50   01    9A    0339   523          MOVZBL  #SS$_NORMAL, R0         ; Change to normal if EOF       [04]
                         05    033C   524 20$:     RSB                             ; Exit.
                        033D   525 ;
                        033D   526 ;+
                        033D   527 ; Skip to end of record
                        033D   528 ; P1 is negative if backward else forwards
                        033D   529 ;-
                        033D   530 E_SKIPRECORD:
                  25    90   033D   531          MOVB    #MSCP$K_OP_REPOS,-      ; Set function to reposition.
              FEE2 CF          033F   532                  CMDPKT+MSCP$B_OPCODE
              2080 8F    B0    0342   533          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_CLSEX,-  ; Detect LEOT          [08]
              FEDD CF          0346   534                                  CMDPKT+MSCP$W_MODIFIER  ; Clear serious except  [08]
           50   04 AC    D0    0349   535          MOVL    BUF(AP),R0              ; Count of RECORDS to skip.
                0A    14    034D   536          BGTR    10$                     ; Branch if positive.
              50   50    CE    034F   537          MNEGL   R0,R0                   ; Make count positive.
              2088 8F    B0    0352   538          MOVW    #MSCP$M_MD_DLEOT!MSCP$M_MD_REVRS!MSCP$M_MD_CLSEX,-; Detect LEOT [08]
              FECD CF          0356   539                  CMDPKT+MSCP$W_MODIFIER  ;  Reverse/clear serious except [08]
        FECA CF   50    D0    0359   540 10$:     MOVL    R0,CMDPKT+MSCP$L_REC_CNT ; Set RECORD to skip.                  [03]
              FECA CF    D4    035E   541          CLRL    CMDPKT+MSCP$L_TMGP_CNT  ; Clear to prevent errors       [08]
                 0001    30    0362   542          BSBW    EXECUTE                 ; Execute function.
                         05    0365   543          RSB                             ; Exit.
                        0366   544
                        0366   545
                        0366   546 ;+
                        0366   547 ; Execute all functions.
                        0366   548 ;-
                        0366   549 EXECUTE:
           54   02 A7    B0    0366   550          MOVW    TUSA(R7),R4             ; Controller offline?
                6B    12    036A   551          BNEQ    50$                     ; Yep, error out
FEA3 CF    8000 8F    A8    036C   552          BISW    #OWN, CD+2              ; Set controller ownership
FE98 CF    8000 8F    A8    0373   553          BISW    #OWN, RD+2              ; Ditto
              54   67    B0    037A   554          MOVW    TUIP(R7),R4             ; Tell controller to go.
           54   02 A7    B0    037D   555          MOVW    TUSA(R7),R4             ; Any problems?
                54    12    0381   556          BNEQ    50$                     ; Yes
           54   02 A7    B0    0383   557 30$:     MovW    TUSa(R7), R4            ; Any problems?
                4E    12    0387   558          BNeq    50$                     ; Yes
              FE85 CF    B5    0389   559          TSTW    RD+2                    ; Any response back?
                F4    19    038D   560          BLSS    30$                     ; No, spin until there is
                        038F   561
                        038F   562 ;
```

```
                                 038F    563 ; Return with  status code.
                                 038F    564 ;
      51   FEC5 CF    05    00    EF  038F    565 40$:    EXTZV   #0, #5, RSPPKT+MSCP$W_STATUS,R1 ; Get return status.          [04]
           50   0084 8F    3C    0396    566         MovZWL  #SS$_DevOffLine, R0     ; Assume offline
                03   51    D1    039B    567         CmpL    R1, #MSCP$K_ST_OFFLN    ; Off line
                     42    13    039E    568         BEql    60$                     ;
           50   0870 8F    3C    03A0    569         MovZWL  #SS$_EndOfFile, R0      ; Assume end of file
                0E   51    D1    03A5    570         CmpL    R1, #MSCP$K_ST_TAPEM    ; Tape mark
                     38    13    03A8    571         BEql    60$                     ;
                13   51    D1    03AA    572         CmpL    R1, #MSCP$K_ST_LED      ; LEOT Detected                           [08]
                     33    13    03AD    573         BEQL    60$                     ;                                         [08]
                50   01    D0    03AF    574         MovL    #SS$_Normal, R0         ; Assume OK
                00   51    D1    03B2    575         CmpL    R1, #MSCP$K_ST_SUCC     ; Success
                     2B    13    03B5    576         BEql    60$                     ;
                04   51    D1    03B7    577         Cmpl    R1,#MSCP$K_ST_AVLBL     ; Unit available                          [08]
                     26    13    03BA    578         Beql    60$                     ;                                         [08]
                0D   51    D1    03BC    579         CmpL    R1, #MSCP$K_ST_BOT      ; At BOT
                     21    13    03BF    580         BEql    60$                     ;
           50   0838 8F    3C    03C1    581         MOVZWL  #SS$_DataoveRun, R0     ; Assume record too large
                10   51    D1    03C6    582         CmpL    R1, #MSCP$K_ST_RDTRN    ; Data truncation
                     17    13    03C9    583         BEql    60$                     ;
      50   00000224 8F    D0    03CB    584         MOVL    #SS$_TAPEPOSLOST, R0    ; Assume tape position lost.
                11   51    D1    03D2    585         CmpL    R1, #MSCP$K_ST_PLOST    ; Tape position lost                      [08]
                     0B    13    03D5    586         BEql    60$                     ;                                         [08]
      50   00000054 8F    D0    03D7    587 50$:    MovL    #SS$_CtrlErr, R0        ; Other codes are failure
                FD2C CF    94    03DE    588         CLRB    W^INIT                  ; MUST RE-INIT NEXT TIME                  [12]
                     05    03E2    589 60$:    Rsb                                 ; Return
                   000003E3    03E3    590 TU_DRVSIZ=.-START
                               03E3    591
                               03E3    592         .END
```

```
$TABLE                          = 00000000 R   D  03
BTD$K_MU                        = 000000B4     D
BTD$K_TF                        = 000000B3     D
BTD$K_TM                        = 000000B1     D
BTD$K_TS                        = 000000B2     D
BUF                             = 00000004     D
BYTE_OFF                          00000098 R   D  02
CD                                00000214 R   D  02
CMDINT                            0000020C R   D  02
CMDPKT                            0000021C R   D  02
DEVICE_CSR                        00000117 R   D  02
DEVICE_UNIT                       0000011B R   D  02
ERROR                             0000008F R   D  02
EXE$GL_TENUSEC                    ********   X   00
EXE$GL_UBDELAY                    ********   X   00
EXECUTE                           00000366 R   D  02
E_READ                            000002ED R   D  02
E_REWIND                          000002D9 R   D  02
E_SKIPFILE                        0000030C R   D  02
E_SKIPRECORD                      0000033D R   D  02
FORMAT                            00000113 R   D  02
FUNC                            = 00000010     D
GO                              = 00000001     D
INIT                              0000010E R   D  02
INTTBL                            00000200 R   D  02
IO$_DRVCLR                      = 00000004     D
IO$_READLBLK                    = 00000021     D
IO$_REWIND                      = 00000024     D
IO$_SKIPFILE                    = 00000025     D
IO$_SKIPRECORD                  = 00000026     D
LBN                             = 0000000C     D
MODE                            = 00000014     D
MSCP$B_BUFFER                   = 00000010     D
MSCP$B_OPCODE                   = 00000008     D
MSCP$K_OP_GTUNT                 = 00000003     D
MSCP$K_OP_ONLIN                 = 00000009     D
MSCP$K_OP_READ                  = 00000021     D
MSCP$K_OP_REPOS                 = 00000025     D
MSCP$K_OP_SETUC                 = 0000000A     D
MSCP$K_ST_AVLBL                 = 00000004     D
MSCP$K_ST_BOT                   = 0000000D     D
MSCP$K_ST_LED                   = 00000013     D
MSCP$K_ST_OFFLN                 = 00000003     D
MSCP$K_ST_PLOST                 = 00000011     D
MSCP$K_ST_RDTRN                 = 00000010     D
MSCP$K_ST_SUCC                  = 00000000     D
MSCP$K_ST_TAPEM                 = 0000000E     D
MSCP$L_BYTE_CNT                 = 0000000C     D
MSCP$L_REC_CNT                  = 0000000C     D
MSCP$L_TMGP_CNT                 = 00000010     D
MSCP$M_MD_CLSEX                 = 00002000     D
MSCP$M_MD_DLEOT                 = 00000080     D
MSCP$M_MD_REVRS                 = 00000008     D
MSCP$M_MD_REWND                 = 00000002     D
MSCP$W_FORMAT                   = 00000020     D
MSCP$W_MODIFIER                 = 0000000A     D
MSCP$W_STATUS                   = 0000000A     D
MSCP$W_UNT_FLGS                 = 0000000E     D
OWN                             = 00008000     D
PR$_MAPEN                       = 00000038     D
PTE$M_PFN                       = 001FFFFF     D
RD                                00000210 R   D  02
RETRY                             00000056 R   D  02
RETURN                            000002D8 R   D  02
RING                              00000210 R   D  02
RPB$L_ADPPHY                    = 0000005C     D
RPB$L_ADPVIR                    = 00000060     D
RPB$L_CSRPHY                    = 00000054     D
RPB$L_CSRVIR                    = 00000058     D
RPB$L_SVASPT                    = 00000050     D
RPB$W_UNIT                      = 00000064     D
RSPINT                            0000020E R   D  02
RSPPKT                            00000250 R   D  02
S1                              = 0000000B     D
S4                              = 0000000E     D
SIZE                            = 00000008     D
SS$_CTRLERR                     = 00000054     D
SS$_DATAOVERUN                  = 00000838     D
SS$_DEVOFFLINE                  = 00000084     D
SS$_ENDOFFILE                   = 00000870     D
SS$_ILLIOFUNC                   = 000000F4     D
SS$_NORMAL                      = 00000001     D
SS$_TAPEPOSLOST                 = 00000224     D
START                             00000000 R   D  02
TIME                              00000060 R   D  02
TUIP                            = 00000000     D
TUSA                            = 00000002     D
TU_DRIVER                         00000280 R   D  02
TU_DRVSIZ                       = 000003E3     D
TU_INIT                           00000000 RG  D  02
UBA$L_MAP                       = 00000800     D
UMR                             = 000001EE     D
UNIT_FLAGS                        0000010F R   D  02
VA$S_VPN                        = 00000015     D
VA$V_VPN                        = 00000009     D
VALID                             0000010A R   D  02
```

```
                                    +-----------------+
                                    ! Psect synopsis !
                                    +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
BOOTDRIVR_2                     000003E3  (  995.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC PAGE
BOOTDRIVR_4                     00000018  (   24.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL                          VALUE        DEFINITION      REFERENCES...
------                          -----        ----------      -------------
$TABLE                         =00000000-R   235    (2)      235    (2)
BTD$K_MU                       =000000B4     125    (2)      235    (2)
BTD$K_TF                       =000000B3     124    (2)
BTD$K_TM                       =000000B1     122    (2)
BTD$K_TS                       =000000B2     123    (2)
BUF                            =00000004     436    (4)    #-513    (4)    #-535    (4)
BYTE_OFF                        00000098-R   310    (3)    #-273    (3)    #-322    (3)    #-326    (3)
CD                              00000214-R   382    (3)    #-327    (3)    #-552    (4)
CMDINT                          0000020C-R   378    (3)
CMDPKT                          0000021C-R   386    (3)      324    (3)      344    (3)    #-484    (4)    #-486    (4)
                                                           #-487    (4)    #-495    (4)    #-496    (4)    #-497    (4)
                                                           #-498    (4)    #-510    (4)    #-512    (4)    #-517    (4)
                                                           #-518    (4)    #-519    (4)    #-532    (4)    #-534    (4)
                                                           #-539    (4)    #-540    (4)    #-541    (4)
DEVICE_CSR                      00000117-R   363    (3)    #-460    (4)    #-471    (4)
DEVICE_UNIT                     0000011B-R   365    (3)    #-462    (4)    #-472    (4)
ERROR                           0000008F-R   307    (3)    #-314    (3)
EXE$GL_TENUSEC                  00000000-XR                 103    (2)    #-304    (3)
EXE$GL_UBDELAY                  00000000-XR                 103    (2)    #-304    (3)
EXECUTE                         00000366-R   549    (4)    #-340    (3)    #-348    (3)    #-488    (4)    #-499    (4)
                                                           #-520    (4)    #-542    (4)
E_READ                          000002ED-R   493    (4)    #-450    (4)
E_REWIND                        000002D9-R   482    (4)    #-452    (4)
E_SKIPFILE                      0000030C-R   508    (4)    #-454    (4)
E_SKIPRECORD                    0000033D-R   530    (4)    #-457    (4)
FORMAT                          00000113-R   361    (3)    #-351    (3)
FUNC                           =00000010     439    (4)    #-449    (4)    #-451    (4)    #-453    (4)    #-455    (4)
                                                           #-458    (4)
GO                             =00000001     129    (2)      375    (3)
INIT                            0000010E-R   358    (3)    #-465    (4)    #-473    (4)    #-588    (4)
INTTBL                          00000200-R   373    (3)      272    (3)    #-275    (3)      293    (3)
IO$_DRVCLR                     =00000004                   #-458    (4)
IO$_READLBLK                   =00000021                   #-449    (4)
IO$_REWIND                     =00000024                   #-451    (4)
IO$_SKIPFILE                   =00000025                   #-453    (4)
IO$_SKIPRECORD                 =00000026                   #-455    (4)
LBN                            =0000000C     438    (4)
MODE                           =00000014     440    (4)
MSCP$B_BUFFER                  =00000010                   #-498    (4)
MSCP$B_OPCODE                  =00000008                   #-484    (4)    #-495    (4)    #-510    (4)    #-532    (4)
MSCP$K_OP_GTUNT                =00000003                   #-347    (3)
MSCP$K_OP_ONLIN                =00000009                   #-335    (3)
MSCP$K_OP_READ                 =00000021                   #-494    (4)
MSCP$K_OP_REPOS                =00000025                   #-483    (4)    #-509    (4)    #-531    (4)
MSCP$K_OP_SETUC                =0000000A     225    (2)
MSCP$K_ST_AVLBL                =00000004                   #-577    (4)
MSCP$K_ST_BOT                  =0000000D                   #-579    (4)
MSCP$K_ST_LED                  =00000013                   #-572    (4)
MSCP$K_ST_OFFLN                =00000003                   #-567    (4)
MSCP$K_ST_PLOST                =00000011                   #-585    (4)
```

```
MSCP$K_ST_RDTRN                 =00000010                       #-582    (4)
MSCP$K_ST_SUCC                  =00000000                       #-575    (4)
MSCP$K_ST_TAPEM                 =0000000E                       #-521    (4)     #-570    (4)
MSCP$L_BYTE_CNT                 =0000000C                       #-497    (4)     #-500    (4)
MSCP$L_REC_CNT                  =0000000C                       #-487    (4)     #-518    (4)     #-540    (4)
MSCP$L_TMGP_CNT                 =00000010                       #-519    (4)     #-541    (4)
MSCP$M_MD_CLSEX                 =00002000                       #-335    (3)     #-485    (4)     #-496    (4)     #-511    (4)
                                                                #-516    (4)     #-533    (4)     #-538    (4)
MSCP$M_MD_DLEOT                 =00000080                       #-511    (4)     #-516    (4)     #-533    (4)     #-538    (4)
MSCP$M_MD_REVRS                 =00000008                       #-516    (4)     #-538    (4)
MSCP$M_MD_REWND                 =00000002                       #-485    (4)
MSCP$W_FORMAT                   =00000020                       #-351    (3)
MSCP$W_MODIFIER                 =0000000A                       #-486    (4)     #-496    (4)     #-512    (4)     #-517    (4)
                                                                #-534    (4)     #-539    (4)
MSCP$W_STATUS                   =0000000A                         565    (4)
MSCP$W_UNT_FLGS                 =0000000E                       #-350    (3)
OWN                             =00008000       130     (2)       373    (3)     #-552    (4)     #-553    (4)
PR$_MAPEN                       =00000038                       #-264    (3)
PTE$M_PFN                       =001FFFFF                       #-283    (3)
RD                               00000210-R     381     (3)     #-323    (3)     #-553    (4)     #-559    (4)
RETRY                            00000056-R     293     (3)     #-307    (3)
RETURN                           000002D8-R     477     (4)     #-465    (4)     #-470    (4)     #-475    (4)
RING                             00000210-R     380     (3)     #-275    (3)
RPB$L_ADPPHY                    =0000005C                         277    (3)     #-278    (3)
RPB$L_ADPVIR                    =00000060                         277    (3)
RPB$L_CSRPHY                    =00000054                         279    (3)     #-280    (3)
RPB$L_CSRVIR                    =00000058                         279    (3)
RPB$L_SVASPT                    =00000050                       #-282    (3)
RPB$W_UNIT                      =00000064                       #-334    (3)     #-346    (3)     #-472    (4)
RSPINT                           0000020E-R     379     (3)
RSPPKT                           00000250-R     390     (3)       321    (3)     #-350    (3)     #-351    (3)     #-500    (4)
                                                                  565    (4)
S1                              =0000000B       131     (2)     #-294    (3)
S4                              =0000000E       132     (2)     #-317    (3)
SIZE                            =00000008       437     (4)
SS$_CTRLERR                     =00000054                       #-308    (3)     #-587    (4)
SS$_DATAOVERUN                  =00000838                       #-581    (4)
SS$_DEVOFFLINE                  =00000084                       #-566    (4)
SS$_ENDOFFILE                   =00000870                       #-569    (4)
SS$_ILLIOFUNC                   =000000F4                       #-476    (4)
SS$_NORMAL                      =00000001                       #-304    (3)     #-464    (4)     #-523    (4)     #-574    (4)
SS$_TAPEPOSLOST                 =00000224                       #-584    (4)
START                            00000000-R     240     (2)       235    (2)       590    (4)
TIME                             00000060-R     299     (3)     #-317    (3)
TUIP                            =00000000       127     (2)     #-295    (3)     #-554    (4)
TUSA                            =00000002       128     (2)     #-304    (3)     #-316    (3)     #-550    (4)     #-555    (4)
                                                                #-557    (4)
TU_DRIVER                        00000280-R     442     (4)       235    (2)
TU_DRVSIZ                       =000003E3       590     (4)       235    (2)
TU_INIT                          00000000-R     260     (3)       469    (4)
UBA$L_MAP                       =00000800                         284    (3)
UMR                             =000001EE       133     (2)     #-271    (3)       284    (3)
UNIT_FLAGS                       0000010F-R     359     (3)     #-350    (3)
VA$S_VPN                        =00000015                       #-276    (3)
VA$V_VPN                        =00000009                       #-276    (3)
VALID                            0000010A-R     357     (3)     #-285    (3)     #-287    (3)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO            SIZE         DEFINITION      REFERENCES...
-----            ----         ----------      -------------
$BOOT_DRIVER     1            232    (2)      232    (2)
$BTDDEF          2            109    (2)      109    (2)
$DEFINI          1            109    (2)      109    (2)      110    (2)      111    (2)      112    (2)      113    (2)
                                             114    (2)      115    (2)      116    (2)      117    (2)      220    (2)
$IODEF           15           110    (2)      110    (2)
$MSCPDEF         25           111    (2)      111    (2)
$PRDEF           4            134    (2)      220    (2)
$PTEDEF          3            112    (2)      112    (2)
$RPBDEF          5            113    (2)      113    (2)
$SSDEF           21           114    (2)      114    (2)
$UBADEF          6            115    (2)      115    (2)
$UBIDEF          2            116    (2)      116    (2)
$VADEF           1            117    (2)      117    (2)
ASSUME           1            277    (3)      277    (3)      279    (3)
TIMEDWAIT        1            300    (3)      300    (3)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	275	(3)												
AOBLEQ	00F3	317	(3)												
BBS	00E0	304	(3)												
BEQL	0013	450	(4)	452	(4)	454	(4)	568	(4)	571	(4)	573	(4)	576	(4)
		578	(4)	580	(4)	583	(4)	586	(4)						
BGTR	0014	514	(4)	536	(4)										
BICL	00CA	283	(3)	322	(3)	326	(3)								
BICL3	00CB	273	(3)												
BISL3	00C9	274	(3)	285	(3)	287	(3)	323	(3)	327	(3)				
BISW	00A8	552	(4)	553	(4)										
BLBC	00E9	281	(3)	470	(4)										
BLBS	00E8	305	(3)	465	(4)										
BLSS	0019	304	(3)	314	(3)	560	(4)								
BNEQ	0012	456	(4)	459	(4)	461	(4)	463	(4)	522	(4)	551	(4)	556	(4)
		558	(4)												
BRB	0011	475	(4)												
BRW	0031	457	(4)												
BSBW	0030	340	(3)	348	(3)	488	(4)	499	(4)	520	(4)	542	(4)		
CALLS	00FB	469	(4)												
CLRB	0094	588	(4)												
CLRL	00D4	304	(3)	339	(3)	448	(4)	518	(4)	541	(4)				
CLRQ	007C	338	(3)	487	(4)										
CLRW	00B4	295	(3)												
CMPL	00D1	449	(4)	451	(4)	453	(4)	455	(4)	458	(4)	460	(4)	521	(4)
		567	(4)	570	(4)	572	(4)	575	(4)	577	(4)	579	(4)	582	(4)
		585	(4)												
CMPW	00B1	462	(4)												
EXTZV	00EF	276	(3)	565	(4)										
INCL	00D6	286	(3)												
MFPR	00DB	264	(3)												
MNEGL	00CE	515	(4)	537	(4)										
MOVAB	009E	272	(3)	293	(3)	321	(3)	324	(3)	344	(3)				
MOVAL	00DE	284	(3)												
MOVB	0090	473	(4)	483	(4)	494	(4)	509	(4)	531	(4)				
MOVL	00D0	271	(3)	278	(3)	280	(3)	282	(3)	292	(3)	294	(3)	304	(3)
		325	(3)	333	(3)	335	(3)	345	(3)	351	(3)	464	(4)	471	(4)
		497	(4)	498	(4)	500	(4)	513	(4)	519	(4)	535	(4)	540	(4)
		574	(4)	584	(4)	587	(4)								
MOVW	00B0	304	(3)	316	(3)	472	(4)	485	(4)	496	(4)	511	(4)	516	(4)
		533	(4)	538	(4)	550	(4)	554	(4)	555	(4)	557	(4)		
MOVZBL	009A	334	(3)	346	(3)	347	(3)	476	(4)	523	(4)				
MOVZBW	009B	350	(3)												
MOVZWL	003C	304	(3)	308	(3)	566	(4)	569	(4)	581	(4)				
MULL3	00C5	304	(3)												
RET	0004	309	(3)	352	(3)										
RSB	0005	478	(4)	489	(4)	501	(4)	524	(4)	543	(4)	589	(4)		
SOBGTR	00F5	304	(3)	307	(3)										
TSTL	00D5	304	(3)	336	(3)										
TSTW	00B5	313	(3)	559	(4)										

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ALIGN	371	(3)														
.BLKL	364	(3)														
.BLKW	366	(3)	387	(3)	389	(3)	390	(3)								
.BYTE	358	(3)														
.DISABLE	11	(1)	355	(3)												
.ENABLE	262	(3)														
.END	592	(4)														
.ENDC	235	(2)	277	(3)	279	(3)	304	(3)								
.ENDM	109	(2)	110	(2)	111	(2)	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)
	117	(2)	219	(2)	232	(2)	277	(3)	300	(3)						
.ENTRY	260	(3)														
.EXTRN	103	(2)														
.IDENT	10	(1)														
.IF	235	(2)	277	(3)	279	(3)	304	(3)								
.IFF	235	(2)	277	(3)	279	(3)										
.LONG	235	(2)	311	(3)	357	(3)	360	(3)	362	(3)	374	(3)	381	(3)	382	(3)
.MACRO	134	(2)														
.NLIST	304	(3)														
.NOCROSS	109	(2)	110	(2)	111	(2)	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)
	117	(2)	220	(2)												
.PSECT	230	(2)	235	(2)												
.RESTORE	109	(2)	110	(2)	111	(2)	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)
	117	(2)	220	(2)												
.SAVE	109	(2)	110	(2)	111	(2)	112	(2)	113	(2)	114	(2)	115	(2)	116	(2)
	117	(2)	220	(2)												
.SBTTL	246	(3)	392	(4)	97	(2)										
.TITLE	9	(1)														
.WORD	235	(2)	373	(3)	375	(3)	377	(3)	378	(3)	379	(3)	384	(3)	385	(3)
	386	(3)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	7	#-449	(4)	#-451	(4)	#-453	(4)	#-455	(4)	#-458	(4)	#-513	(4)
		#-535	(4)										
R0	26	#-264	(3)	#-278	(3)	#-280	(3)	#-281	(3)	#-304	(3)	#-305	(3)
		#-308	(3)	#-464	(4)	#-470	(4)	#-476	(4)	#-513	(4)	#-515	(4)
		#-519	(4)	#-523	(4)	#-535	(4)	#-537	(4)	#-540	(4)	#-566	(4)
		#-569	(4)	#-574	(4)	#-581	(4)	#-584	(4)	#-587	(4)		
R1	22	#-273	(3)	#-274	(3)	#-304	(3)	#-321	(3)	#-322	(3)	#-323	(3)
		#-324	(3)	#-325	(3)	#-326	(3)	#-327	(3)	#-500	(4)	#-521	(4)
		#-565	(4)	#-567	(4)	#-570	(4)	#-572	(4)	#-575	(4)	#-577	(4)
		#-579	(4)	#-582	(4)	#-585	(4)						
R10	1	#-498	(4)										
R2	16	260	(3)	#-272	(3)	#-273	(3)	#-274	(3)	#-275	(3)	276	(3)
		#-282	(3)	#-283	(3)	#-285	(3)	#-286	(3)	#-287	(3)	#-294	(3)
		#-304	(3)	#-317	(3)								
R3	6	260	(3)	#-278	(3)	284	(3)	#-293	(3)	#-316	(3)	#-462	(4)
R4	11	260	(3)	#-284	(3)	#-285	(3)	#-287	(3)	#-304	(3)	#-313	(3)
		#-550	(4)	#-554	(4)	#-555	(4)	#-557	(4)				
R5	12	260	(3)	#-325	(3)	#-333	(3)	#-334	(3)	#-335	(3)	#-336	(3)
		#-338	(3)	#-339	(3)	#-344	(3)	#-345	(3)	#-346	(3)	#-347	(3)
R6	5	260	(3)	#-271	(3)	#-274	(3)	#-323	(3)	#-327	(3)		
R7	11	260	(3)	#-280	(3)	#-295	(3)	#-304	(3)	#-316	(3)	#-460	(4)
		#-471	(4)	#-550	(4)	#-554	(4)	#-555	(4)	#-557	(4)		
R8	4	260	(3)	#-292	(3)	#-307	(3)	#-497	(4)				
R9	6	#-278	(3)	#-280	(3)	#-282	(3)	#-334	(3)	#-346	(3)	#-472	(4)
SP	5	#-304	(3)	#-448	(4)								

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.18
Command processing	877	00:00:00.20	00:00:00.68
Pass 1	680	00:00:03.15	00:00:04.83
Symbol table sort	0	00:00:00.48	00:00:00.47
Pass 2	6	00:00:00.95	00:00:02.00
Symbol table output	0	00:00:00.01	00:00:00.06
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	4	00:00:00.14	00:00:00.26
Assembler run totals	1589	00:00:04.97	00:00:08.49

The working set limit was 3400 pages.
172785 bytes (338 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1690 non-local and 16 local symbols.
592 source lines were read in Pass 1, producing 109 object records in Pass 2.
57 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	8
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	15

1715 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:MUBTDRIVR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

```
0     ! DEC/CMS REPLACEMENT HISTORY, Element ODS.B32
0     ! *1     6-FEB-1986 15:20:49 DS ""
0     ! DEC/CMS REPLACEMENT HISTORY, Element ODS.B32
0     %title '*** ODS Disk RMS routines'
0     module ODS (                                  ! RMS$OPEN, RMS$GET, RMS$READ, RMS$CLOSE services
0                     ident = '01-04'
0                     ) =
0     !
0     ! Copyright (c) 1980, 1982
0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0     !
0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0     ! REMAIN IN DEC.
0     !
0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0     ! CORPORATION.
0     !
0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0
0     !++
0     ! FACILITY: Diagnostic Supervisor
0     !
0     ! ABSTRACT:
0     !
0     !       This module supports RMS functions on ODS media.
0     !
0     ! AUTHOR:
0     !       Dave Butenhof,          19-aug-1980
0     !
0     ! MODIFIED BY:
0     !
0     !               Dave Butenhof, 08-oct-1980, version 6.1
0     !       1       Specify "long_relative" access to BOO$QIO routine
0     !               Dave Butenhof, 6-feb-1981, version 6.3
0     !       2       Fix so full record length if returned in RAB$L_STV on
0     !               RMS$_RTB errors.
0     !               - Dave Butenhof, 02-Jun-1981, version 6.4
0     !       03      Alter library spec for flexibility
0     !
0     !       04      - Dave Butenhof, 13-Jan-1982, version 6.6
0     !               Make some minor fixes like variable attributes.  Also,
0     !               support ODS-1 by taking RSIZE field of file header if
0     !               MAXREC is 0 (as it is for ODS-1 fixed length).
0     !--
0
```

```
;    0054  0     %sbttl 'declarations'
;    0055  1     begin
;    0056  1     !
;    0057  1     ! include files:
;    0058  1     !
;    0059  1
;    0060  1     library '$diag';                !                                                    [03]
;    0061  1
;    0062  1     library '$ds';                  !                                                    [03]
;    0063  1
;    0064  1     library 'sys$library:lib.l32';
;    0065  1
;    0066  1     !
;    0067  1     ! local declarations
;    0068  1     !
;    0069  1
;    0070  1     linkage
;    0071  1         jsb_ods = jsb (register = 0) : global (block = 6, vbn = 7, offset = 8, cblock = 9, rab = 10);
;    0072  1
;    0073  1     !
;    0074  1     ! table of contents:
;    0075  1     !
;    0076  1
;    0077  1     forward routine
;    0078  1         ods$advance : jsb_ods novalue,          ! Advance RFA
;    0079  1         ods$access : jsb_ods,                   ! Access a VBN
;    0080  1         ods$open : call_rms,                    ! RMS$OPEN emulator--open file
;    0081  1         ods$get : call_rms,                     ! RMS$GET emulator--deblock record
;    0082  1         ods$read : call_rms;                    ! RMS$READ emulator--read file block(s)
;    0083  1
;    0084  1     !
;    0085  1     ! Local definitions
;    0086  1     !
;    0087  1     !
;    0088  1     ! External references
;    0089  1     !
;    0090  1
;    0091  1     external routine
;    0092  1         rms$alloc_cache : jsb_cblock,       ! Allocate cache blocks
;    0093  1         fil$openfile,                       ! 'Open' a file
;    0094  1         fil$readvbn,                        ! Read a virtual block
;    0095  1         boo$qio : addressing_mode (long_relative),  ! standalone 'level-3' QIO
;    0096  1         exe$alononpaged : jsb_alloc addressing_mode (long_relative);        ! Allocate static buffer space
;    0097  1
;    0098  1     !
;    0099  1     ! psect definitions:
;    0100  1     !
;    0101  1
;    0102  1     psect
;    0103  1         plit = data ( share, addressing_mode (long_relative));
;    0104  1
;    0105  1     psect
;    0106  1         own = work ( read, write);
;    0107  1
;    0108  1     psect
;    0109  1         global = work ( read, write);
;    0110  1
```

```
;   0111  1     psect
;   0112  1         code = code (share);
;   0113  1
```

```
;    0115  1     %sbttl 'Advance RFA'
;    0116  1     routine ods$advance (len) : jsb_ods novalue =
;    0117  1
;    0118  1     !++
;    0119  1     ! Functional description:
;    0120  1     !       Increment the internal 'RFA' (offset and vbn parts) by the
;    0121  1     !       specified number of bytes, considering round-off and block
;    0122  1     !       overlap.
;    0123  1     !
;    0124  1     ! Inputs:
;    0125  1     !       len                     number of bytes to increment by
;    0126  1     !
;    0127  1     ! Implicit inputs:
;    0128  1     !       R7                      vbn
;    0129  1     !       R8                      offset in vbn
;    0130  1     !
;    0131  1     ! Outputs:
;    0132  1     !       none
;    0133  1     !
;    0134  1     ! Implicit outputs:
;    0135  1     !       R7                      new vbn
;    0136  1     !       R8                      new byte offset in vbn
;    0137  1     !
;    0138  1     ! Side effects:
;    0139  1     !       none
;    0140  1     !
;    0141  1     ! Return codes:
;    0142  1     !       none
;    0143  1     !--
;    0144  1
;    0145  2         begin
;    0146  2
;    0147  2         external register
;    0148  2             vbn = 7,                            ! Virtual block number
;    0149  2             offset = 8;                         ! Byte offset in VBN
;    0150  2
;    0151  2         local
;    0152  2             bite;
;    0153  2
;    0154  2         bite = (.offset + .len + 1) and not 1;      ! Add length and round up
;    0155  2         offset = .bite and 511;                     ! Truncate to offset in page
;    0156  3         vbn = .vbn + (.bite/512)                    ! Calculate new page
;    0157  1         end;                                        ! of ods$advance

                                                    .TITLE  ODS *** ODS Disk RMS routines
                                                    .IDENT  \01-04\

                                                    .EXTRN  RMS$ALLOC_CACHE
                                                    .EXTRN  FIL$OPENFILE, FIL$READVBN
                                                    .EXTRN  BOO$QIO, EXE$ALONONPAGED

                                                    .PSECT  CODE,NOWRT,  SHR,2

                            50      01 A048  9E 00000 ODS$ADVANCE:
                                                    MOVAB   1(LEN)[OFFSET], R0                  ; 0154
                            50           01  8A 00005        BICB2   #1, BITE                    ;
```

```
                58              50              09              00  EF 00008        EXTZV   #0, #9, BITE, OFFSET                    ; 0155
                                                50 00000200     8F  C6 0000D        DIVL2   #512, R0                                ; 0156
                                                57              50  C0 00014        ADDL2   R0, VBN                                 ;
                                                                    05 00017        RSB                                             ; 0157

; Routine Size:  24 bytes,    Routine Base:  CODE + 0000
```

```
;    0159  1     %sbttl 'Access file block'
;    0160  1     routine ods$access : jsb_ods =
;    0161  1
;    0162  1     !++
;    0163  1     ! Functional description:
;    0164  1     !       If the desired VBN is not currently cached, update cache to
;    0165  1     !       contain it.
;    0166  1     !
;    0167  1     ! Inputs:
;    0168  1     !       none
;    0169  1     !
;    0170  1     ! Implicit inputs:
;    0171  1     !       R7                      VBN to cache/access
;    0172  1     !       R9                      pointer to CBLOCK control structure
;    0173  1     !       R10                     pointer to RAB
;    0174  1     !
;    0175  1     ! Outputs:
;    0176  1     !       none
;    0177  1     !
;    0178  1     ! Implicit outputs:
;    0179  1     !       R6                      pointer to 'memory image' of file block
;    0180  1     !
;    0181  1     ! Side effects:
;    0182  1     !       may alter cache control structure of CBLOCK
;    0183  1     !
;    0184  1     ! Return codes:
;    0185  1     !       RMS$_NORMAL             OK
;    0186  1     !       RMS$_RFA                random RFA is illegal
;    0187  1     !       RMS$_EOF                VBN is past end of file
;    0188  1     !       RMS$_RER                File read error
;    0189  1     !--
;    0190  1
;    0191  2         begin
;    0192  2
;    0193  2         external register
;    0194  2             block = 6,
;    0195  2             vbn = 7,
;    0196  2             cblock = 9 : ref bblock,
;    0197  2             rab = 10 : ref bblock;
;    0198  2
;    0199  2         local
;    0200  2             read_blocks;                            ! Blocks to EOF
;    0201  2
;    0202  2         bind
;    0203  2             cache = cblock [ctl$l_cache1] : vector; ! Map for convenience
;    0204  2
;    0205  2         read_blocks = .cblock [ctl$l_eofvbn] - .vbn;
;    0206  2
;    0207  2         if .read_blocks lss 0                       ! If no more blocks
;    0208  3             or (.read_blocks eql 0                  !  or at EOF block
;    0209  3             and .cblock [ctl$w_offbyte] eql 0)      !  and no bytes there
;    0210  2         then
;    0211  2             return
;    0212  2
;    0213  2                 if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof;      ! Check for illegal vbn
;    0214  2
;    0215  2         if .vbn lss .cblock [ctl$l_lvbn]            ! If VBN isn't
```

```
;   0216  2             or .vbn geq .cblock [ctl$l_hvbn]          !  cached now
;   0217  2         then
;   0218  3             begin                                     ! then cache it
;   0219  3
;   0220  3             if .cblock [ctl$w_cache_size] eql 0       ! If no cache allocated
;   0221  3             then
;   0222  4                 begin
;   0223  4
;   0224  4                 local
;   0225  4                     stat;
;   0226  4
;   0227  4                 stat = rms$alloc_cache (1);           ! Allocate cache
;   0228  4
;   0229  4                 if not .stat
;   0230  4                 then
;   0231  4                     return rms$_dme                   ! Error if can't
;   0232  3                 end;
;   0233  3
;   0234  3             cblock [ctl$l_lvbn] = 0;                  ! Clear start vbn
;   0235  3             cblock [ctl$l_hvbn] = 0;                  ! Clear high vbn
;   0236  3             read_blocks = min (2,                     ! 3 blocks or to EOF
;   0237  3                 (.read_blocks*512 + .cblock [ctl$w_offbyte] + 511)/512 - 1);
;   0238  3
;   0239  3             incr i from 0 to .read_blocks do          ! read cache buffers
;   0240  4                 begin
;   0241  4
;   0242  4                 local
;   0243  4                     stat;
;   0244  4
;   0245  5                 stat = (if .cblock [ctl$l_startlbn] neq 0   ! If contiguous
;   0246  5                 then boo$qio (.cache [.i], 512,       ! do direct QIO
;   0247  5                         .vbn + .cblock [ctl$l_startlbn] + .i - 1, io$_readlblk, 0, .cblock [ctl$l_rpb]) else
;   0248  5                     fil$readvbn (.cblock [ctl$l_rpb], .vbn + .i,    ! Translate VBN to
;   0249  4                         .cache [.i], .cblock [ctl$l_filhdr]));      !  LBN and read
;   0250  4
;   0251  4                 if not .stat                          ! If it failed
;   0252  4                 then
;   0253  5                     begin
;   0254  5                     rab [rab$l_stv] = .stat;          ! Load QIO return
;   0255  5                     return rms$_rer
;   0256  5                     end
;   0257  5
;   0258  3                 end;
;   0259  3
;   0260  3             cblock [ctl$l_lvbn] = .vbn;               ! Set new low vbn
;   0261  3             cblock [ctl$l_hvbn] = .vbn + .read_blocks + 1   ! Set new high vbn
;   0262  2             end;
;   0263  2
;   0264  2         block = .cache [.vbn - .cblock [ctl$l_lvbn]];       ! Get buffer pointer
;   0265  2         1                                             ! Return success
;   0266  1         end;                                          ! of ods$access
```

```
                                                                 PUSHR   #^M<R2,R3,R4>                                    ; 0160
                                 54       2C   A9  9E 00002      MOVAB   44(CBLOCK), R4                                  ; 0203
                          53     44   A9       57  C3 00006      SUBL3   VBN, 68(CBLOCK), READ_BLOCKS                    ; 0205
                                              07  19 0000B      BLSS    1$                                              ; 0207
                                              1D  12 0000D      BNEQ    3$                                              ; 0208
                                          42  A9  B5 0000F      TSTW    66(CBLOCK)                                      ; 0209
                                              18  12 00012      BNEQ    3$                                              ;
                                 02       1E   AA  91 00014 1$:  CMPB    30(RAB), #2                                     ; 0213
                                              09  12 00018      BNEQ    2$                                              ;
                                 50 0001865C   8F  D0 0001A      MOVL    #99932, R0                                      ;
                                              2D  11 00021      BRB     5$                                              ;
                                 50 0001827A   8F  D0 00023 2$:  MOVL    #98938, R0                                      ;
                                              24  11 0002A      BRB     5$                                              ;
                                 28   A9           57  D1 0002C 3$:  CMPL    VBN, 40(CBLOCK)                         ; 0215
                                              09  19 00030      BLSS    4$                                              ;
                                 24   A9           57  D1 00032      CMPL    VBN, 36(CBLOCK)                         ; 0216
                                              03  18 00036      BGEQ    4$                                              ;
                                            009D  31 00038      BRW     13$                                             ;
                                          0A  A9  B5 0003B 4$:  TSTW    10(CBLOCK)                                      ; 0220
                                              12  12 0003E      BNEQ    6$                                              ;
                                 50           01  D0 00040      MOVL    #1, R0                                          ; 0227
                                           0000G  30 00043      BSBW    RMS$ALLOC_CACHE                                 ;
                                 09           50  E8 00046      BLBS    STAT, 6$                                        ; 0229
                                 50 000184D4   8F  D0 00049      MOVL    #99540, R0                                      ; 0231
                                              76  11 00050 5$:  BRB     11$                                             ;
                                          24  A9  7C 00052 6$:  CLRQ    36(CBLOCK)                                      ; 0235
                          50             53       09  78 00055      ASHL    #9, READ_BLOCKS, R0                     ; 0237
                                 51       42   A9  3C 00059      MOVZWL  66(CBLOCK), R1                                  ;
                                 50     01FF C140  9E 0005D      MOVAB   511(R1)[R0], R0                                 ;
                                 50 00000200   8F  C6 00063      DIVL2   #512, R0                                        ;
                                              50  D7 0006A      DECL    R0                                              ;
                                 02           50  D1 0006C      CMPL    R0, #2                                  ; 0236
                                              03  15 0006F      BLEQ    7$                                              ;
                                 50           02  D0 00071      MOVL    #2, R0                                          ;
                                 53           50  D0 00074 7$:  MOVL    R0, READ_BLOCKS                                 ;
                                 52           01  CE 00077      MNEGL   #1, I                                   ; 0247
                                              4E  11 0007A      BRB     12$                                             ;
                          51     52           02  78 0007C 8$:  ASHL    #2, I, R1                               ; 0246
                                          50  A9  D5 00080      TSTL    80(CBLOCK)                              ; 0245
                                              22  13 00083      BEQL    9$                                              ;
                                          38  A9  DD 00085      PUSHL   56(CBLOCK)                              ; 0247
                                 7E           21  7D 00088      MOVQ    #33, -(SP)                              ; 0246
                          50     57       50  A9  C1 0008B      ADDL3   80(CBLOCK), VBN, R0                     ; 0247
                                        FF A240  9F 00090      PUSHAB  -1(I)[R0]                                       ;
                                 7E     0200   8F  3C 00094      MOVZWL  #512, -(SP)                             ; 0246
                                           6144  9F 00099      PUSHAB  (R1)[R4]                                        ;
                                              9E  DD 0009C      PUSHL   @(SP)+                                  ;
                    00000000G    EF           06  FB 0009E      CALLS   #6, BOO$QIO                                     ;
                                              13  11 000A5      BRB     10$                                             ;
                                          54  A9  DD 000A7 9$:  PUSHL   84(CBLOCK)                              ; 0249
                                           6144  9F 000AA      PUSHAB  (R1)[R4]                                        ;
                                              9E  DD 000AD      PUSHL   @(SP)+                                  ;
                                           6247  9F 000AF      PUSHAB  (I)[VBN]                                ; 0248
                                          38  A9  DD 000B2      PUSHL   56(CBLOCK)                                      ;
                          0000G        CF           04  FB 000B5      CALLS   #4, FIL$READVBN                                 ;
                                 0D           50  E8 000BA 10$:  BLBS    STAT, 12$                               ; 0251
                                 0C   AA          50  D0 000BD      MOVL    STAT, 12(RAB)                           ; 0254
```

```
                                50 0001C0F4    8F  D0 000C1          MOVL    #114932, R0                                     ; 0255
                                               1A  11 000C8 11$:     BRB     14$                                             ;
                     AE              52        53  F3 000CA 12$:     AOBLEQ  READ_BLOCKS, I, 8$                              ; 0253
                           28        A9        57  D0 000CE          MOVL    VBN, 40(CBLOCK)                                 ; 0260
                           24        A9     01 A347  9E 000D2          MOVAB   1(READ_BLOCKS)[VBN], 36(CBLOCK)                 ; 0261
                     50              57     28   A9  C3 000D8 13$:     SUBL3   40(CBLOCK), VBN, R0                             ; 0264
                                     56      6440  D0 000DD          MOVL    (R4)[R0], BLOCK                                 ;
                                     50        01  D0 000E1          MOVL    #1, R0                                          ; 0266
                                               1C  BA 000E4 14$:     POPR    #^M<R2,R3,R4>                                   ;
                                                   05 000E6          RSB                                                     ;

; Routine Size:  231 bytes,    Routine Base:  CODE + 0018
```

```
;    0268  1      %sbttl 'ODS OPEN service'
;    0269  1
;    0270  1      global routine ods$open : call_rms =
;    0271  1
;    0272  1      !++
;    0273  1      ! Functional description:
;    0274  1      !       Access a file on ODS files-11 disk media.  Return file parameters
;    0275  1      !       in the FAB and internal control block associated with the file
;    0276  1      !
;    0277  1      ! Inputs:
;    0278  1      !       none
;    0279  1      !
;    0280  1      ! Implicit inputs:
;    0281  1      !       R11     points to the FAB
;    0282  1      !       R9      points to the CBLOCK (internal RMS control block)
;    0283  1      !
;    0284  1      ! Outputs:
;    0285  1      !       none
;    0286  1      !
;    0287  1      ! Implicit outputs:
;    0288  1      !       Several fields in the FAB and in the static file control block
;    0289  1      !       are altered to describe the file.  Also, if an XAB (XABFHC is
;    0290  1      !       the only XAB currently supported) exists, it will be filled in.
;    0291  1      !
;    0292  1      ! Return status
;    0293  1      !       RMS$_DME        insufficient memory for buffer allocation
;    0294  1      !       RMS$_FNF        file not found
;    0295  1      !       RMS$_FNM        Bad file name
;    0296  1      !       RMS$_RER        QIO error on file access
;    0297  1      !       RMS$_NORMAL     success
;    0298  1      !--
;    0299  1
;    0300  2          begin
;    0301  2
;    0302  2          local
;    0303  2              ndxhdr : bblock [512],                  ! Buffer for index header
;    0304  2              filhdr : ref bblock,                    ! Convenient pointer
;    0305  2              status : vector [2],                    ! BOO$QIO status return
;    0306  2              stat;
;    0307  2
;    0308  2          external register
;    0309  2              cblock = 9 : ref bblock,                ! Pointer to control block
;    0310  2              rab = 10,
;    0311  2              fab = 11 : ref bblock;                  ! Define the FAB
;    0312  2
;    0313  3          begin                                       ! Allocate file header space
;    0314  3
;    0315  3          global register
;    0316  3              size = 1,
;    0317  3              addr = 2;
;    0318  3
;    0319  3          size = 512;                                 ! Size of file header
;    0320  3
;    0321  3          if not exe$alononpaged () then return rms$_dme;
;    0322  3
;    0323  3          filhdr = cblock [ctl$l_filhdr] = .addr      ! Remember address
;    0324  2          end;
```

```
;    0325  2          stat = fil$openfile (.cblock [ctl$l_rpb], cblock [ctl$q_filename], ndxhdr, .filhdr, status);
;    0326  2                                                    ! Access the file
;    0327  2
;    0328  2          if not .stat                              ! Error
;    0329  2          then
;    0330  3              return (selectone .stat of
;    0331  3                      set
;    0332  3                      [ss$_nosuchfile] : rms$_fnf;
;    0333  3                      [ss$_badfilename] : rms$_fnm;
;    0334  3                      [ss$_filestruct, ss$_badfilehdr, ss$_badchksum] : (fab [fab$l_stv] = .stat; rms$_acc);
;    0335  4                      [otherwise] : (fab [fab$l_stv] = .stat; rms$_rer)
;    0336  2                      tes);
;    0337  2
;    0338  2          if .filhdr [fh2$b_struclev] eql 2         ! If structure level 2                                    [04]
;    0339  2          then
;    0340  2              cblock [ctl$v_ods2] = 1;              ! Then set flag bit
;    0341  2
;    0342  2          cblock [ctl$l_get] = ods$get;             ! Set GET routine address
;    0343  2          cblock [ctl$l_read] = ods$read;           ! Set READ routine address
;    0344  2          cblock [ctl$l_startlbn] = .status [0];    ! First LBN (if contiguous)
;    0345  3          begin                                     ! Load file attributes
;    0346  3
;    0347  3          builtin
;    0348  3              rot;
;    0349  3
;    0350  3          bind
;    0351  3              fileatt = (if .cblock [ctl$v_ods2] then filhdr [fh2$w_recattr] else filhdr [fh1$w_recattr]) : bblock;
;    0352  3
;    0353  3          fab [fab$b_rat] = .fileatt [fat$b_rattrib]; ! Copy record attributes
;    0354  3                                                    ! Get EOF block, which is
;    0355  3          cblock [ctl$l_eofvbn] =                   ! The EOF block VBN is
;    0356  3          rot (.fileatt [fat$l_efblk], 16);         !  stored sideways
;    0357  3          cblock [ctl$w_offbyte] = .fileatt [fat$w_ffbyte];   ! First free byte
;    0358  3          fab [fab$b_org] = .fileatt [fat$v_fileorg]; ! File organization
;    0359  3          fab [fab$b_rfm] = .fileatt [fat$v_rtype];   ! Record format
;    0360  3          fab [fab$w_mrs] = .fileatt [fat$w_maxrec];  ! Max. rec size
;    0361  3
;    0362  3          If .Fab [Fab$W_Mrs] Eql 0                 ! If no MAXREC                                            [04]
;    0363  3          Then                                      !                                                         [04]
;    0364  3              Fab [Fab$W_Mrs] = .FileAtt [Fat$W_RSize];       ! Get the record size                         [04]
;    0365  3
;    0366  3          fab [fab$b_fsz] = .fileatt [fat$b_vfcsize]  ! Variable/fixed header size
;    0367  2          end;
;    0368  2          rms$_normal
;    0369  1          end;                                      ! of ODS$OPEN


                                    01FC 00000          .ENTRY  ODS$OPEN, Save R2,R3,R4,R5,R6,R7,R8          ; 0270
                5E      FDF8    CE  9E 00002          MOVAB   -520(SP), SP                                 ;
                51       0200   8F  3C 00007          MOVZWL  #512, SIZE                                   ; 0319
                   00000000G    EF  16 0000C          JSB     EXE$ALONONPAGED                              ; 0321
                08              50  E8 00012          BLBS    R0, 1$                                       ;
                50 000184D4     8F  D0 00015          MOVL    #99540, R0                                   ;
                                    04 0001C          RET                                                  ;
```

```
                    54   A9              52  D0 0001D 1$:    MOVL    ADDR, 84(CBLOCK)                    ; 0323
                                   4004  8F  BB 00021        PUSHR   #^M<R2,SP>                          ; 0325
                                     10  AE  9F 00025        PUSHAB  NDXHDR                              ;
                                     48  A9  9F 00028        PUSHAB  72(CBLOCK)                          ;
                                     38  A9  DD 0002B        PUSHL   56(CBLOCK)                          ;
                  0000G  CF              05  FB 0002E        CALLS   #5, FIL$OPENFILE                    ;
                         55              50  E8 00033        BLBS    STAT, 6$                            ; 0328
              00000910  8F              50  D1 00036        CMPL    STAT, #2320                         ; 0332
                                        08  12 0003D        BNEQ    2$                                  ;
                         50 00018292    8F  D0 0003F        MOVL    #98962, R0                          ;
                                            04 00046        RET                                         ;
              00000818  8F              50  D1 00047 2$:    CMPL    STAT, #2072                         ; 0333
                                        08  12 0004E        BNEQ    3$                                  ;
                         50 0001852C    8F  D0 00050        MOVL    #99628, R0                          ;
                                            04 00057        RET                                         ;
              00000808  8F              50  D1 00058 3$:    CMPL    STAT, #2056                         ; 0334
                                        12  13 0005F        BEQL    4$                                  ;
              00000810  8F              50  D1 00061        CMPL    STAT, #2064                         ;
                                        09  13 00068        BEQL    4$                                  ;
              000008C0  8F              50  D1 0006A        CMPL    STAT, #2240                         ;
                                        0C  12 00071        BNEQ    5$                                  ;
                    0C   AB              50  D0 00073 4$:    MOVL    STAT, 12(FAB)                       ;
                         50 0001C002    8F  D0 00077        MOVL    #114690, R0                         ;
                                            04 0007E        RET                                         ;
                    0C   AB              50  D0 0007F 5$:    MOVL    STAT, 12(FAB)                       ; 0335
                         50 0001C0F4    8F  D0 00083        MOVL    #114932, R0                         ;
                                            04 0008A        RET                                         ; 0330
                         02       07    A2  91 0008B 6$:    CMPB    7(FILHDR), #2                       ; 0338
                                        04  12 0008F        BNEQ    7$                                  ;
                    01   A9              04  88 00091        BISB2   #4, 1(CBLOCK)                       ; 0340
                    0C   A9     0000V    CF  9E 00095 7$:    MOVAB   ODS$GET, 12(CBLOCK)                 ; 0342
                    10   A9     0000V    CF  9E 0009B        MOVAB   ODS$READ, 16(CBLOCK)                ; 0343
                    50   A9              6E  D0 000A1        MOVL    STATUS, 80(CBLOCK)                  ; 0344
              08    01   A9              02  E1 000A5        BBC     #2, 1(CBLOCK), 8$                   ; 0351
                         52              14  C0 000AA        ADDL2   #20, R2                             ;
                         50              52  D0 000AD        MOVL    R2, R0                              ;
                                        04  11 000B0        BRB     9$                                  ;
                         50       0E    A2  9E 000B2 8$:    MOVAB   14(R2), R0                          ;
                    1E   AB       01    A0  90 000B6 9$:    MOVB    1(R0), 30(FAB)                      ; 0353
          44   A9    08   A0              10  9C 000BB        ROTL    #16, 8(R0), 68(CBLOCK)              ; 0356
                    42   A9       0C    A0  B0 000C1        MOVW    12(R0), 66(CBLOCK)                  ; 0357
     51         60        04              04  EF 000C6        EXTZV   #4, #4, (R0), R1                    ; 0358
                    1D   AB              51  90 000CB        MOVB    R1, 29(FAB)                         ;
     51         60        04              00  EF 000CF        EXTZV   #0, #4, (R0), R1                    ; 0359
                    1F   AB              51  90 000D4        MOVB    R1, 31(FAB)                         ;
                    36   AB       10    A0  B0 000D8        MOVW    16(R0), 54(FAB)                     ; 0360
                                        05  12 000DD        BNEQ    10$                                 ; 0362
                    36   AB       02    A0  B0 000DF        MOVW    2(R0), 54(FAB)                      ; 0364
                    3F   AB       0F    A0  90 000E4 10$:   MOVB    15(R0), 63(FAB)                     ; 0366
                         50 00010001    8F  D0 000E9        MOVL    #65537, R0                          ; 0369
                                            04 000F0        RET                                         ;

; Routine Size:  241 bytes,    Routine Base:  CODE + 00FF

;   0370  1
```

```
;    0372  1      %sbttl 'ODS$GET routine'
;    0373  1
;    0374  1      global routine ods$get : call_rms =
;    0375  1
;    0376  1      !++
;    0377  1      ! Functional description:
;    0378  1      !       De-block and return a record from an ODS file.
;    0379  1      !
;    0380  1      ! Inputs:
;    0381  1      !       none
;    0382  1      !
;    0383  1      ! Implicit inputs:
;    0384  1      !       R9                      CBLOCK pointer
;    0385  1      !       R10                     RAB pointer
;    0386  1      !       R11                     FAB pointer
;    0387  1      !       several fields in the above data structures
;    0388  1      !
;    0389  1      ! Outputs:
;    0390  1      !       none
;    0391  1      !
;    0392  1      ! Implicit outputs:
;    0393  1      !       RAB [rab$l_stv]         Set to error status of failing system service
;    0394  1      !       RAB [rab$l_rbf]         Set to user buffer
;    0395  1      !       RAB [rab$w_rsz]         Size of record copied to user buffer
;    0396  1      !
;    0397  1      ! Side effects:
;    0398  1      !       CBLOCK [ctl$w_rfa]      advanced one record if access was not
;    0399  1      !                               random-by-RFA
;    0400  1      !       CBLOCK                  cache control and cache buffers may be
;    0401  1      !                               altered if part or all of the record is
;    0402  1      !                               not within a block currently cached.
;    0403  1      !
;    0404  1      ! Return codes:
;    0405  1      !       RMS$_NORMAL             if OK
;    0406  1      !       RMS$_RER                if system read error (QIO)
;    0407  1      !       RMS$_RFA                if random RFA is beyond EOF
;    0408  1      !       RMS$_EOF                if RFA is beyond end of file
;    0409  1      !       RMS$_IRC                Illegal record
;    0410  1      !       RMS$_RTB                Record is too big for user buffer (truncated)
;    0411  1      !--
;    0412  1
;    0413  2          begin
;    0414  2
;    0415  2          external register
;    0416  2              cblock = 9 : ref bblock,                ! Pointer to CBLOCK
;    0417  2              rab = 10 : ref bblock,                  ! Pointer to RAB
;    0418  2              fab = 11 : ref bblock;                  ! Pointer to FAB
;    0419  2
;    0420  2          global register
;    0421  2              block = 6 : ref vector [, byte],        ! Pointer to cache buffer
;    0422  2              vbn = 7,                                ! Current VBN in file
;    0423  2              offset = 8;                             ! Current offset in VBN
;    0424  2
;    0425  2          local
;    0426  2              f_size : word,                          ! Full size of record                  [04]
;    0427  2              size : word,                            ! Size of record                       [04]
;    0428  2              lenlim : word,                          ! Size of user buffer                  [04]
```

```
;    0429  2              ptr;                                    ! Address in user buffer
;    0430  2
;    0431  2          vbn = .cblock [ctl$l_vbn];                  ! Extract vbn from next RFA
;    0432  2          offset = .cblock [ctl$w_byte];              ! Extract byte offset
;    0433  2
;    0434  2          do
;    0435  3              begin                                   ! Loop until find real record
;    0436  3
;    0437  3              if .rab [rab$b_rac] neq rab$c_rfa       ! If not random access
;    0438  3              then
;    0439  3                  ods$advance (.cblock [ctl$w_rec_size]);     ! Else advance to next
;    0440  3
;    0441  3              cblock [ctl$l_vbn] = .vbn;              ! Update CBLOCK vbn
;    0442  3              cblock [ctl$w_byte] = .offset;          !  and it's offset
;    0443  3
;    0444  3              if .vbn gtr .cblock [ctl$l_eofvbn] or .vbn eql .cblock [ctl$l_eofvbn] and .offset geq .cblock [
;    0445  3                      ctl$w_offbyte]                  ! If past file EOF
;    0446  3              then
;    0447  3                  return rms$_eof;                    ! can't read past EOF
;    0448  3
;    0449  4              (
;    0450  4
;    0451  4              local
;    0452  4                  stat;
;    0453  4
;    0454  4              stat = ods$access ();                   ! Load the block in cache
;    0455  3              if not .stat then return .stat);
;    0456  3              cblock [ctl$w_rec_size] =               ! Set size of current record
;    0457  4              (if .fab [fab$b_rfm] neq fab$c_fix      ! If not fixed record format
;    0458  4              then
;    0459  5                  begin
;    0460  5                  size = .(block [.offset])<0, 16>;   ! then fetch record size
;    0461  5                  ods$advance (2);                    ! Skip over the size
;    0462  6                  (
;    0463  6
;    0464  6                  local
;    0465  6                      stat;
;    0466  6
;    0467  6                  stat = ods$access ();               ! and be sure (new) VBN is there
;    0468  5                  if not .stat then return .stat);
;    0469  5                  .size + 2
;    0470  5                  end
;    0471  4              else size = .fab [fab$w_mrs])           ! else get fixed size from FAB
;    0472  3              end
;    0473  3          until (if .size eql %x'FFFF'                ! Means non-spanned 'skip to next block'
;    0474  3              then
;    0475  4                  begin
;    0476  4                  vbn = .vbn + 1;
;    0477  4                  cblock [ctl$w_rec_size] = offset = 0;
;    0478  4                  0
;    0479  4                  end
;    0480  2              else 1);
;    0481  2
;    0482  2          if .fab [fab$b_rfm] eql fab$c_vfc           ! If we have header record
;    0483  2          then
;    0484  3              begin
;    0485  3
```

```
;   0486  3             if .size lss .fab [fab$b_fsz]
;   0487  3             then
;   0488  4                 (rab [rab$l_stv] = .vbn;            ! Store VBN of illegal record
;   0489  4                 return rms$_irc                     ! Illegal record
;   0490  3                 );
;   0491  3
;   0492  3             if .rab [rab$l_rhb] neq 0               ! If there's a place to put it
;   0493  3             then
;   0494  3                 ch$move (.fab [fab$b_fsz], block [.offset], .rab [rab$l_rhb]);      !  copy it over
;   0495  3
;   0496  3             size = .size - .fab [fab$b_fsz];        ! Decrease record size by header length
;   0497  3             ods$advance (.fab [fab$b_fsz])          ! and advance over it anyway
;   0498  2             end;
;   0499  2
;   0500  2         ptr = rab [rab$l_rbf] = .rab [rab$l_ubf];   ! Point to user buffer
;   0501  2         lenlim = .rab [rab$w_usz];                  ! Size limit
;   0502  2         rab [rab$w_rsz] = 0;                        ! Init output size
;   0503  2         f_size = .size;                             ! Store size, since SIZE moves
;   0504  2
;   0505  2         while 1 do                                  ! Copy over the record
;   0506  3             begin
;   0507  3
;   0508  3             local
;   0509  3                 length;
;   0510  3
;   0511  4             (
;   0512  4
;   0513  4             local
;   0514  4                 stat;
;   0515  4
;   0516  4             stat = ods$access ();                   ! Be sure VBN is there
;   0517  3             if not .stat then return .stat);
;   0518  3             length = min (.size, (512 - .offset), .lenlim); ! Size to copy
;   0519  3             ptr = ch$move (.length, block [.offset], .ptr); ! Copy (part of) record
;   0520  3             rab [rab$w_rsz] = .rab [rab$w_rsz] + .length;   ! Update actual length so far
;   0521  3             size = .size - .length;                 ! Rest of record
;   0522  3
;   0523  3             if .size eql 0 then return rms$_normal; ! Return if done
;   0524  3
;   0525  3             lenlim = .lenlim - .length;             ! Buffer size left
;   0526  3
;   0527  3             if .lenlim eql 0                        ! If no space left
;   0528  3             then
;   0529  4                 begin
;   0530  4                 rab [rab$l_stv] = .f_size;          ! Return true size
;   0531  4                 return rms$_rtb
;   0532  3                 end;
;   0533  3
;   0534  3             ods$advance (.length)                   ! Skip on to next block
;   0535  2             end;
;   0536  2
;   0537  2         rms$_normal
;   0538  1         end;
```

```
                                                  01FC 00000          .ENTRY  ODS$GET, Save R2,R3,R4,R5,R6,R7,R8          ; 0374
                                 5E            10  C2 00002          SUBL2   #16, SP                                     ;
                                 57        18  A9  D0 00005          MOVL    24(CBLOCK), VBN                             ; 0431
                                 58        1C  A9  3C 00009          MOVZWL  28(CBLOCK), OFFSET                          ; 0432
                                 02        1E  AA  91 0000D 1$:      CMPB    30(RAB), #2                                 ; 0437
                                               07  13 00011          BEQL    2$                                          ;
                                 50        1E  A9  3C 00013          MOVZWL  30(CBLOCK), R0                              ; 0439
                                            FDF6  30 00017          BSBW    ODS$ADVANCE                                 ;
                           18    A9            57  D0 0001A 2$:      MOVL    VBN, 24(CBLOCK)                             ; 0441
                           1C    A9            58  B0 0001E          MOVW    OFFSET, 28(CBLOCK)                          ; 0442
                           44    A9            57  D1 00022          CMPL    VBN, 68(CBLOCK)                             ; 0444
                                               0A  14 00026          BGTR    3$                                          ;
                                               10  12 00028          BNEQ    4$                                          ;
    58       42    A9            10            00  ED 0002A          CMPZV   #0, #16, 66(CBLOCK), OFFSET                 ;
                                               08  14 00030          BGTR    4$                                          ;
                                 50 0001827A   8F  D0 00032 3$:      MOVL    #98938, R0                                  ; 0447
                                                   04 00039          RET                                                 ;
                                            FDEB  30 0003A 4$:      BSBW    ODS$ACCESS                                  ; 0454
                                 15            50  E9 0003D          BLBC    STAT, 5$                                    ; 0455
                                 01        1F  AB  91 00040          CMPB    31(FAB), #1                                 ; 0457
                                               1A  13 00044          BEQL    6$                                          ;
                                            6846  9F 00046          PUSHAB  (OFFSET)[BLOCK]                             ; 0460
                                 6E            9E  B0 00049          MOVW    @(SP)+, SIZE                                ;
                                 50            02  D0 0004C          MOVL    #2, R0                                      ; 0461
                                            FDBE  30 0004F          BSBW    ODS$ADVANCE                                 ;
                                            FDD3  30 00052          BSBW    ODS$ACCESS                                  ; 0467
                                 75            50  E9 00055 5$:      BLBC    STAT, 13$                                   ; 0468
                                 50            6E  3C 00058          MOVZWL  SIZE, R0                                    ; 0469
                                 50            02  C0 0005B          ADDL2   #2, R0                                      ;
                                               07  11 0005E          BRB     7$                                          ;
                                 50        36  AB  3C 00060 6$:      MOVZWL  54(FAB), R0                                 ; 0471
                                 6E            50  B0 00064          MOVW    R0, SIZE                                    ;
                           1E    A9            50  B0 00067 7$:      MOVW    R0, 30(CBLOCK)                              ; 0457
                         FFFF    8F            6E  B1 0006B          CMPW    SIZE, #65535                                ; 0473
                                               09  12 00070          BNEQ    8$                                          ;
                                               57  D6 00072          INCL    VBN                                         ; 0476
                                               58  D4 00074          CLRL    OFFSET                                      ; 0477
                                         1E    A9  B4 00076          CLRW    30(CBLOCK)                                  ;
                                               92  11 00079          BRB     1$                                          ;
                                 03        1F  AB  91 0007B 8$:      CMPB    31(FAB), #3                                 ; 0482
                                               32  12 0007F          BNEQ    11$                                         ;
                                 50        3F  AB  9A 00081          MOVZBL  63(FAB), R0                                 ; 0486
                                 6E            50  B1 00085          CMPW    R0, SIZE                                    ;
                                               0C  1B 00088          BLEQU   9$                                          ;
                           0C    AA            57  D0 0008A          MOVL    VBN, 12(RAB)                                ; 0488
                                 50 0001857C   8F  D0 0008E          MOVL    #99708, R0                                  ; 0489
                                                   04 00095          RET                                                 ;
                                         2C    AA  D5 00096 9$:      TSTL    44(RAB)                                     ; 0492
                                               0A  13 00099          BEQL    10$                                         ;
                                 50        3F  AB  9A 0009B          MOVZBL  63(FAB), R0                                 ; 0494
    2C    BA                6846            50  28 0009F          MOVC3   R0, (OFFSET)[BLOCK], @44(RAB)                ;
                                 50        3F  AB  9A 000A5 10$:     MOVZBL  63(FAB), R0                                 ; 0496
                                 6E            50  A2 000A9          SUBW2   R0, SIZE                                    ;
                                 50        3F  AB  9A 000AC          MOVZBL  63(FAB), R0                                 ; 0497
                                            FD5D  30 000B0          BSBW    ODS$ADVANCE                                 ;
                                 50        24  AA  D0 000B3 11$:     MOVL    36(RAB), R0                                 ; 0500
```

```
                                28   AA          50  D0 000B7          MOVL    R0, 40(RAB)                                  ;
                                     53          50  D0 000BB          MOVL    R0, PTR                                      ;
                                08   AE      20  AA  B0 000BE          MOVW    32(RAB), LENLIM                              ; 0501
                                             22  AA  B4 000C3          CLRW    34(RAB)                                      ; 0502
                                0C   AE          6E  B0 000C6          MOVW    SIZE, F_SIZE                                 ; 0503
                                             FD5B  30 000CA 12$:      BSBW    ODS$ACCESS                                   ; 0516
                                     58          50  E9 000CD 13$:      BLBC    STAT, 18$                                    ; 0517
                     51 00000200     8F          58  C3 000D0          SUBL3   OFFSET, #512, R1                             ; 0518
                                     50          6E  3C 000D8          MOVZWL  SIZE, R0                                     ;
                                     51          50  D1 000DB          CMPL    R0, R1                                       ;
                                                 03  15 000DE          BLEQ    14$                                          ;
                                     50          51  D0 000E0          MOVL    R1, R0                                       ;
        50       08   AE             10          00  ED 000E3 14$:      CMPZV   #0, #16, LENLIM, R0                          ;
                                                 04  18 000E9          BGEQ    15$                                          ;
                                     50      08  AE  3C 000EB          MOVZWL  LENLIM, R0                                   ;
                                04   AE          50  D0 000EF 15$:      MOVL    R0, LENGTH                                   ;
                     63           6846      04  AE  28 000F3          MOVC3   LENGTH, (OFFSET)[BLOCK], (PTR)               ; 0519
                                22   AA      04  AE  A0 000F9          ADDW2   LENGTH, 34(RAB)                              ; 0520
                                     6E      04  AE  A2 000FE          SUBW2   LENGTH, SIZE                                 ; 0521
                                                 1D  13 00102          BEQL    17$                                          ; 0523
                                08   AE      04  AE  A2 00104          SUBW2   LENGTH, LENLIM                               ; 0525
                                                 0D  12 00109          BNEQ    16$                                          ; 0527
                                0C   AA      0C  AE  3C 0010B          MOVZWL  F_SIZE, 12(RAB)                              ; 0530
                                     50 000181A8  8F  D0 00110          MOVL    #98728, R0                                   ; 0531
                                                     04 00117          RET                                                  ;
                                     50      04  AE  D0 00118 16$:      MOVL    LENGTH, R0                                   ; 0534
                                             FCF1  30 0011C          BSBW    ODS$ADVANCE                                  ;
                                                 A9  11 0011F          BRB     12$                                          ;
                                     50 00010001  8F  D0 00121 17$:      MOVL    #65537, R0                                   ; 0538
                                                     04 00128 18$:      RET                                                  ;

; Routine Size:  297 bytes,    Routine Base:  CODE + 01F0

;   0539  1
```

```
;    0541  1      %sbttl 'ODS$READ routine'
;    0542  1
;    0543  1      global routine ods$read : call_rms =
;    0544  1
;    0545  1      !++
;    0546  1      ! Functional description:
;    0547  1      !       Read virtual blocks of file
;    0548  1      !
;    0549  1      ! Inputs:
;    0550  1      !       none
;    0551  1      !
;    0552  1      ! Implicit inputs:
;    0553  1      !       R9                      CBLOCK pointer
;    0554  1      !       R10                     RAB pointer
;    0555  1      !       various fields in the above-listed data structures
;    0556  1      !
;    0557  1      ! Outputs:
;    0558  1      !       none
;    0559  1      !
;    0560  1      ! Implicit outputs:
;    0561  1      !       RAB [rab$l_rbf]         points to buffer
;    0562  1      !       RAB [rab$w_rsz]         size of buffer
;    0563  1      !
;    0564  1      ! Side Effects:
;    0565  1      !       CBLOCK [ctl$l_nbp]      points to next VBN for sequential READ.
;    0566  1      !
;    0567  1      ! Return codes:
;    0568  1      !       RMS$_NORMAL             OK
;    0569  1      !       RMS$_RER                Read error
;    0570  1      !--
;    0571  1
;    0572  2          begin
;    0573  2
;    0574  2          external register
;    0575  2              cblock = 9 : ref bblock,                    ! Map the control block
;    0576  2              rab = 10 : ref bblock;                      ! Map the RAB
;    0577  2
;    0578  2          local
;    0579  2              size,
;    0580  2              next;
;    0581  2
;    0582  2          next = .cblock [ctl$l_nbp];                     ! Block to start at
;    0583  2          size = rab [rab$w_rsz] = min (.rab [rab$w_usz],     ! Determine actual size to
;    0584  2              (.cblock [ctl$l_eofvbn] - .next)*512 +      !  read--either given size
;    0585  2              .cblock [ctl$w_offbyte]);                   !  or to end of file if less
;    0586  2          cblock [ctl$l_nbp] = .next + (.size + 511)/512;     ! Point to next full block
;    0587  2
;    0588  2          if .cblock [ctl$l_startlbn] neq 0               ! If file is contiguous
;    0589  2          then
;    0590  3              begin
;    0591  3
;    0592  3              local
;    0593  3                  stat;
;    0594  3
;    0595  3              stat = boo$qio (rab [rab$l_rbf] = .rab [rab$l_ubf], .size, .cblock [ctl$l_startlbn] + .next - 1,
;    0596  3                  io$_readlblk, 0, .cblock [ctl$l_rpb]);      ! Read all at once
;    0597  3
```

```
;   0598  3             if not .stat                          ! If error, scram
;   0599  3             then
;   0600  4                 (rab [rab$l_stv] = .stat; return rms$_rer)
;   0601  4
;   0602  3             end
;   0603  2         else                                      ! Non-contiguous, read it
;   0604  3             begin                                 !  block by block
;   0605  3
;   0606  3             local
;   0607  3                 ptr;
;   0608  3
;   0609  3             ptr = rab [rab$l_rbf] = .rab [rab$l_ubf];       ! Start of user buffer
;   0610  3
;   0611  3             while .size gtr 0 do
;   0612  4                 begin
;   0613  4
;   0614  4                 global register
;   0615  4                     block = 6,                    ! Registers for ACCESS
;   0616  4                     vbn = 7,
;   0617  4                     offset = 8;
;   0618  4
;   0619  4                 local
;   0620  4                     rs,
;   0621  4                     stat;
;   0622  4
;   0623  4                 vbn = .next;                      ! Next block to read
;   0624  4                 rs = min (.size, 512);            ! Read a block or less
;   0625  4                 stat = ods$access ();             ! Access next block
;   0626  4
;   0627  4                 if not .stat                      ! Scram if error
;   0628  4                 then
;   0629  4                     (rab [rab$l_stv] = .stat; return rms$_rer);
;   0630  4
;   0631  4                 ptr = ch$move (.rs, .block, .ptr);  ! Copy to user buffer
;   0632  4                 next = .next + 1;                 ! Advance block number
;   0633  4                 size = .size - .rs                ! Count size of read
;   0634  4                 end
;   0635  4
;   0636  2             end;
;   0637  2
;   0638  2         rms$_normal
;   0639  1         end;                                      ! Of ods$read
```

```
                                        09FC 00000        .ENTRY  ODS$READ, Save R2,R3,R4,R5,R6,R7,R8,R11     ; 0543
                        5E          04  C2 00002        SUBL2   #4, SP                                      ;
                                04  A9  DD 00005        PUSHL   4(CBLOCK)                                   ; 0582
        51      44      A9          6E  C3 00008        SUBL3   NEXT, 68(CBLOCK), R1                        ; 0584
        51              51          09  78 0000D        ASHL    #9, R1, R1                                  ;
                        50      42  A9  3C 00011        MOVZWL  66(CBLOCK), R0                              ; 0585
                        51          50  C0 00015        ADDL2   R0, R1                                      ;
                        50      20  AA  3C 00018        MOVZWL  32(RAB), R0                                 ; 0584
                        51          50  D1 0001C        CMPL    R0, R1                                      ;
                                    03  15 0001F        BLEQ    1$                                          ;
```

```
                                                
```

```
;
;  CODE                          972  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...											
ADDR	GlobReg	317	323.											
BBLOCK	Structur	Lib02	196	197	303	304	309	311	351	416	417	418	575	576
BITE	Local	152	154=	155.	156.									
BLOCK	ExtReg	194	264=											
	GlobReg	421	460a.	494a.	519a.									
	GlobReg	615	631a.											
BOO$QIO	ExtRout	95	246c	595c										
CACHE	Bind	203	246.	249.	264.									
CALL_RMS	Linkage	Lib02	80	81	82	270	374	543						
CBLOCK	ExtReg	196	203aa	205a.	209a.	215a.	216a.	220a.	234a=	235a=	237a.	245a.	247a.	248a.
			249a.	260a=	261a=	264a.								
	ExtReg	309	323a=	325a.	325aa	340a=	342a=	343a=	344a=	351a.	355a=	357a=		
	ExtReg	416	431a.	432a.	439a.	441a=	442a=	444a.	456a=	477a=				
	ExtReg	575	582a.	584a.	585a.	586a=	588a.	595a.	596a.					
CTL$L_CACHE1	Macro	Lib02	203											
CTL$L_EOFVBN	Macro	Lib02	205	355	444	584								
CTL$L_FILHDR	Macro	Lib02	249	323										
CTL$L_GET	Macro	Lib02	342											
CTL$L_HVBN	Macro	Lib02	216	235	261									
CTL$L_LVBN	Macro	Lib02	215	234	260	264								
CTL$L_NBP	Macro	Lib02	582	586										
CTL$L_READ	Macro	Lib02	343											
CTL$L_RPB	Macro	Lib02	247	248	325	596								
CTL$L_STARTLBN	Macro	Lib02	245	247	344	588	595							
CTL$L_VBN	Macro	Lib02	431	441										
CTL$Q_FILENAME	Macro	Lib02	325											
CTL$V_ODS2	Macro	Lib02	340	351										
CTL$W_BYTE	Macro	Lib02	432	442										
CTL$W_CACHE_SIZE	Macro	Lib02	220											
CTL$W_OFFBYTE	Macro	Lib02	209	237	357	445	585							
CTL$W_REC_SIZE	Macro	Lib02	439	456	477									
EXE$ALONONPAGED	ExtRout	96	321c											
FAB	ExtReg	311	334a=	335a=	353a=	358a=	359a=	360a=	362a.	364a=	366a=			
	ExtReg	418	457a.	471a.	482a.	486a.	494a.	496a.	497a.					
FAB$B_FSZ	Macro	Lib03	366	486	494	496	497							
FAB$B_ORG	Macro	Lib03	358											
FAB$B_RAT	Macro	Lib03	353											
FAB$B_RFM	Macro	Lib03	359	457	482									
FAB$C_FIX	Literal	Lib03	457											
FAB$C_VFC	Literal	Lib03	482											
FAB$L_STV	Macro	Lib03	334	335										
FAB$W_MRS	Macro	Lib03	360	362	364	471								
FAT$B_RATTRIB	Macro	Lib03	353											
FAT$B_VFCSIZE	Macro	Lib03	366											
FAT$L_EFBLK	Macro	Lib03	356											
FAT$V_FILEORG	Macro	Lib03	358											
FAT$V_RTYPE	Macro	Lib03	359											
FAT$W_FFBYTE	Macro	Lib03	357											

Symbol	Type	Defined	Referenced ...
FAT$W_MAXREC	Macro	Lib03	360
FAT$W_RSIZE	Macro	Lib03	364
FH1$W_RECATTR	Macro	Lib03	351
FH2$B_STRUCLEV	Macro	Lib03	338
FH2$W_RECATTR	Macro	Lib03	351
FIL$OPENFILE	ExtRout	93	325c
FIL$READVBN	ExtRout	94	248c
FILEATT	Bind	351	353. 356. 357. 358. 359. 360. 364. 366.
FILHDR	Local	304	323= 325. 338a. 351aa
F_SIZE	Local	426	503= 530.
I	Local	239	246. 247. 248. 249.
IO$_READLBLK	Literal	Lib03	247 596
JSB_ALLOC	Linkage	Lib02	96
JSB_CBLOCK	Linkage	Lib02	92
JSB_ODS	Linkage	71	78 79 116 160
LEN	RoutForm	116	154.
LENGTH	Local	509	518= 519. 520. 521. 525. 534.
LENLIM	Local	428	501= 518. 525= 525. 527.
NDXHDR	Local	303	325a
NEXT	Local	580	582= 584. 586. 595. 623. 632= 632.
ODS$ACCESS	Routine	160	79f 454c 467c 516c 625c
ODS$ADVANCE	Routine	116	78f 439c 461c 497c 534c
ODS$GET	GlobRout	374	81f 342a
ODS$OPEN	GlobRout	270	80f
ODS$READ	GlobRout	543	82f 343a
OFFSET	ExtReg	149	154. 155=
	GlobReg	423	432= 442. 444. 460. 477= 494. 518. 519.
	GlobReg	617	
PTR	Local	429	500= 519= 519a=
	Local	607	609= 631= 631a=
RAB	ExtReg	197	213a. 254a=
	ExtReg	310	
	ExtReg	417	437a. 488a= 492a. 494a= 500a= 500a. 501a. 502a= 520a= 520a. 530a=
	ExtReg	576	583a= 583a. 595a= 595a. 600a= 609a= 609a. 629a=
RAB$B_RAC	Macro	Lib03	213 437
RAB$C_RFA	Literal	Lib03	213 437
RAB$L_RBF	Macro	Lib03	500 595 609
RAB$L_RHB	Macro	Lib03	492 494
RAB$L_STV	Macro	Lib03	254 488 530 600 629
RAB$L_UBF	Macro	Lib03	500 595 609
RAB$W_RSZ	Macro	Lib03	502 520 583
RAB$W_USZ	Macro	Lib03	501 583
READ_BLOCKS	Local	200	205= 207. 208. 236= 237. 239. 261.
RMS$ALLOC_CACHE	ExtRout	92	227c
RMS$_ACC	Literal	Lib03	334
RMS$_DME	Literal	Lib03	231 321
RMS$_EOF	Literal	Lib03	213 447
RMS$_FNF	Literal	Lib03	332
RMS$_FNM	Literal	Lib03	333
RMS$_IRC	Literal	Lib03	489
RMS$_NORMAL	Literal	Lib03	368 523 537 638
RMS$_RER	Literal	Lib03	255 335 600 629
RMS$_RFA	Literal	Lib03	213
RMS$_RTB	Literal	Lib03	531
ROT	Builtin	348	356

```
Symbol              Type  Defined  Referenced ...
---------------- -------- ----- ----------------
RS                Local      620   624=    631.    633.
SIZE              GlobReg    316   319=
                  Local      427   460=    469.    471=    473.    486.    496=    496.    503.    518.    521=    521.    523.
                  Local      579   583=    586.    595.    611.    624.    633=    633.
SS$_BADCHKSUM     Literal  Lib03   334
SS$_BADFILEHDR    Literal  Lib03   334
SS$_BADFILENAME   Literal  Lib03   333
SS$_FILESTRUCT    Literal  Lib03   334
SS$_NOSUCHFILE    Literal  Lib03   332
STAT              Local      225   227=    229.
                  Local      243   245=    251.    254.
                  Local      306   325=    328.    330.    334.    335.
                  Local      452   454=    455.
                  Local      465   467=    468.
                  Local      514   516=    517.
                  Local      593   595=    598.    600.
                  Local      621   625=    627.    629.
STATUS            Local      305   325a    344.
VBN               ExtReg     148   156=    156.
                  ExtReg     195   205.    215.    216.    247.    248.    260.    261.    264.
                  GlobReg    422   431=    441.    444.    476=    476.    488.
                  GlobReg    616   623=


                CROSS REFERENCE MAP

Line #      Event          File ...
-----  ---------------  ---------
    1  Source (start)   DISK$DS:[DS.LOCAL.RELEASE.WORK]ODS.B32;1
    5  Module ODS
   60  Library #1       DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
   62  Library #2       DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
   64  Library #3       SYS$COMMON:[SYSLIB]LIB.L32;2
  643  Eludom ODS


        KEY TO REFERENCE TYPE FLAGS

            .   Fetch
            =   Store
            c   Routine call
            a   Address use
            @   Indirect use
            e   External, external routine, or external literal declaration
            f   Forward or forward routine declaration
            h   Condition handler enabling
            m   Map declaration
            u   Undeclare declaration


;                           Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
```

```
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                          940          0          0         96          00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                          675         21          3         47          00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2             20450         43          0       1101          00:00.8




;                                       COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:ODS.B32

; Size:         972 code + 0 data bytes
; Run Time:         00:04.3
; Elapsed Time:     00:07.9
; Lines/CPU Min:     8930
; Lexemes/CPU-Min: 69152
; Memory Used:  163 pages
; Compilation Complete
```

Table of contents


```
(1)     126     Macro invocations
(1)     219     SHELL OF STAND ALONE SUPERVISOR MEMORY.
(2)     292     DSP$CONFIG_ADP  Build P-table for adapter
(5)     414     MAP$IOSPACE     Validate mapping for adapters
(7)     467     FETCH_LONG      Retreive longword from possibly bad address
(8)     512     PURGE DATAPATH
(10)    555     UBA$INITIAL     UNIBUS ADAPTER INITIALIZATION
(11)    583     IO$TESTCSR_x    Test CSR for existence
(12)    632     IOGEN$CONN_VEC
(13)    659     QIOCLEAN_CPU
(14)    681     ONLY_SCB        CPU specific SCB setup
(15)    692     HDW_CLOCK_SET   CPU specific clock setup
(16)    703     ONLY_ATTACH                             ; [18]
(17)    714     SCBVECTOR_xxx   Otherwise undefined SCB vectors
(17)    767     DSR$ONLY_CONSOLE_CONFIG  console device configuration for load path
(18)    824     ONLY_CPU        CPU specific setup                      [23]  ;G:2
```

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element ONLY730.MAR
                        0000     2 ; *3    11-NOV-1987 08:38:12 GEARIN "ADDED SUPPORT FOR EXTENDED P-TABLE"
                        0000     3 ; *2    10-APR-1986 17:24:16 SHELHAMER "<no reason given>"
                        0000     4 ; *1     6-FEB-1986 15:20:57 DS ""
                        0000     5 ; DEC/CMS REPLACEMENT HISTORY, Element ONLY730.MAR
                        0000     6         .TITLE  ONLY730 *** ONLY730 NEBULA specific routines
                        0000     7         .IDENT /10.10-3/                                                ;G:3
                        0000     8         .LIST   MEB
                        0000     9         .NLIST  CND
                        0000    10         .DSABL  GBL
                        0000    11
                        0000    12 ;
                        0000    13 ; Copyright (c) 1979, 1983, 1984, 1985, 1986, 1987                      ;G:3
                        0000    14 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000    15 ;
                        0000    16 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    17 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    18 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    19 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    20 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    21 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    22 ; REMAIN IN DEC.
                        0000    23 ;
                        0000    24 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    25 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    26 ; CORPORATION.
                        0000    27 ;
                        0000    28 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    29 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    30
                        0000    31 ;++
                        0000    32 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
                        0000    33 ;
                        0000    34 ; ABSTRACT:
                        0000    35 ;
                        0000    36 ; ENVIRONMENT:
                        0000    37 ;
                        0000    38 ; AUTHOR:       Roger Riggs     20-FEB-79       VERSION 01.
                        0000    39 ;
                        0000    40 ; MODIFIED BY:
                        0000    41 ;
                        0000    42 ;                   Roger Riggs, 24-Jan-1989, Version 5.2
                        0000    43 ;           02      Added definition of value DS$GK_USOVR, for
                        0000    44 ;                   WAITUS clock overhead.
                        0000    45 ;                   Roger Riggs, 15-Feb-1980, Version 5.2
                        0000    46 ;           03      Fixed QIOCLEAN_CPU to correctly restore the SCB
                        0000    47 ;                   Dave Butenhof, 07-May-1981, version 6.4
                        0000    48 ;           04      Fix PURGDATAP routine to set correct address for UBA map.
                        0000    49 ;           05      ATTACH DW730, not DW750.
                        0000    50 ;
                        0000    51 ;           06      - Dave Butenhof, 23-Nov-1981, version 6.5
                        0000    52 ;                   Add .LIBRARY declarations, change SEP Psect to Code and
                        0000    53 ;                   Work.
                        0000    54 ;
                        0000    55 ;           07      - Dave Butenhof, 09-Mar-1982, version 6.6
                        0000    56 ;                   Fix truncation errors.
                        0000    57 ;
```

```
0000     58 ;       08      Bob Bergazzi    April 26, 1983  Version 6.11
0000     59 ;               Added ONLY_SCB routine for VENUS. Does nothing for NEBULA.
0000     60 ;
0000     61 ;       09      Bob Bergazzi    May 16, 1983    Version 6.11
0000     62 ;               Changed the order of the .LIB statements.
0000     63 ;
0000     64 ;       10      Bob Bergazzi 11-16-83   Version 6.14
0000     65 ;               Changed MODNAM from ONLY750 to ONY730.
0000     66 ;
0000     67 ;       11      Bob Bergazzi    Jan. 25, 1984   Version 6.14
0000     68 ;               Added ONLY_CLOCK_INIT routine, which reads the TODR.
0000     69 ;               We no longer want to MFPR the TODR in CLK$RE_INIT
0000     70 ;               because not all cpus read the TODR with an MFPR.
0000     71 ;               Also, TODR is now defined by $PR730DEF as PR730$_TODR.
0000     72 ;
0000     73 ;       12      Bob Bergazzi    May 9, 1984     Version 7.0
0000     74 ;               Added UBINTSZ in order to remove code from QIO.
0000     75 ;
0000     76 ;       13      Bob Bergazzi    Aug. 27, 1984   Version 7.1
0000     77 ;               Removed ONLY_CLOCK_INIT routine.
0000     78 ;
0000     79 ;       14      Bob Bergazzi    Aug. 29, 1984   Version 7.1
0000     80 ;               Added HDW_CLOCK_SET routine. Does nothing for 11/730.
0000     81 ;
0000     82 ;       15      Bob Bergazzi    Oct. 9, 1984    Version 7.1
0000     83 ;               Added EXE$GL_TENUSEC and EXE$GL_UBDELAY values for use by
0000     84 ;               bootdrivrs that use the TIMEDWAIT macro (PABTDRIVR).
0000     85 ;
0000     86 ;       16      RE Muse         Nov.7,1984      version 7.1
0000     87 ;               This psect was originally in the kernel module and now
0000     88 ;               becomes machine specific to accomodate the differences
0000     89 ;               for scb and unexpected handlers page allocation.
0000     90 ;
0000     91 ;       17      Domenic Andella 9-Apr-85        Version 8.2
0000     92 ;               Establish global location to store the primary cpu
0000     93 ;               identifier number for use by multi-processing in
0000     94 ;               routine MAP$IOSPACE.
0000     95 ;
0000     96 ;       18      R.E. Muse       9 May 85        Version 8.2
0000     97 ;
0000     98 ;               incorporate the only_attach routine
0000     99 ;
0000    100 ;       19      Bob Bergazzi    7-June-1985     Version 8.3
0000    101 ;               Change EXE$GL_TENUSEC and EXE$GL_UBDELAY to
0000    102 ;               ONLY$GL_TENUSEC and ONLY$GL_UBDELAY because the former
0000    103 ;               are set from the latter at boot time, and the former
0000    104 ;               symbols are now kept in the BOOTDRIVR module.
0000    105 ;               This is so the TIMEDWAIT macro in some bootdrivr modules
0000    106 ;               will work.
0000    107 ;
0000    108 ;       20      M. Baggett      21-June-1985    Version 8.3
0000    109 ;               Truncation error.
0000    110 ;
0000    111 ;       21      Ted Salem       21-November-1985 Version 9.2
0000    112 ;               Added DSR$ONLY_CONSOLE_CONFIG to all ONLY modules.
0000    113 ;
0000    114 ;       22      Bob Bergazzi    7-Jan-1986      Version 9.1
```

```
                                0000   115 ;               Added DEF$Q_DEV, DEF$Q_DIR, SYS$SYSROOT, and SYS$DISK. These
                                0000   116 ;               were removed from the LOAD module so that the default directory
                                0000   117 ;               could be system specific.
                                0000   118 ;                                                                          ;G:2
                                0000   119 ;         23    Jim Shelhamer   10-Apr-1986     Version 10.0               ;G:2
                                0000   120 ;               Added ONLY_CPU section.                                     ;G:2
                                0000   121 ;                                                                          ;G:3
                                0000   122 ;               David Gearin    23-Oct-1987     Version 10.10              ;G:3
                                0000   123 ;               Added code for extended p-table format.                    ;G:3
                                0000   124 ;--
```

```
                     0000   126 .subtitle       Macro invocations
                     0000   127
                     0000   128 .library        /sys$library:lib/       ;                       [09]
                     0000   129 .library        /$DS/                   ;                       [09]
                     0000   130 .library        /$DIAG/                 ;                       [09]
                     0000   131
                     0000   132          .NLIST  ME,MEB
                     0000   133         $PHDDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .RESTORE
                     0000   134         $JIBDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .RESTORE
                     0000   135         $ARBDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .RESTORE
                     0000   136         $UCBDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                     0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004             0000                        .BLKL   1
                     0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                     0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008             0004                        .BLKL   1
                     0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
0000000A             0008                        .BLKW   1
                     000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
0000000B             000A                        .BLKB   1
                     000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
0000000C             000B                        .BLKB   1
                     000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                     000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                     000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
0000000D             000C                        .BLKB   1
00000010             000D                        .BLKB   3
                     0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                     0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
00000014             0010                        .BLKL   1
                     0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                     0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                     0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
00000016             0014                        .BLKW   1
                     0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
00000018             0016                        .BLKW   1
                     0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                     0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
0000001A             0018                        .BLKW   1
                     001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                     001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
0000001C             001A                        .BLKW   1
                     001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
00000020             001C                        .BLKL   1
                     0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
00000024             0020                        .BLKL   1
                     0024                .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
00000028             0024                        .BLKL   1
                     0028                .IIF    NB,UCB$L_PID,   UCB$L_PID:
```

```
0000002C  0028                                  .BLKL   1
          002C                  .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
00000030  002C                                  .BLKL   1
          0030                  .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
00000034  0030                                  .BLKL   1
          0034                  .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
00000039  0038                                  .BLKB   1
          0039                  .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
0000003A  0039                                  .BLKB   1
          003A                  .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
0000003C  003A                                  .BLKW   1
          003C                  .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
          003C                  .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
          003C                  .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
0000003D  003C                                  .BLKB   1
          003D                  .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
          003D                  .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
0000003E  003D                                  .BLKB   1
          003E                  .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
          003E                  .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
0000003F  003E                                  .BLKB   1
          003F                  .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
00000040  003F                                  .BLKB   1
          0040                  .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                                  .BLKW   1
          004A                  .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
          004A                  .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                                  .BLKB   1
          004B                  .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                                  .BLKB   1
          004C                  .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                  .BLKW   1
          0052                  .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
          0052                  .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                  .BLKB   1
          0053                  .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                  .BLKB   1
          0054                  .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
          005A                  .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
          005C                  .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C                  .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                  .BLKL   1
          0060                  .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                                  .BLKL   1
          0064                  .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
```

```
                    0064                .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068            0064                        .BLKL   1
                    0068                .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C            0068                        .BLKL   1
                    006C                .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E            006C                        .BLKW   1
                    006E                .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070            006E                        .BLKW   1
                    0070                .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071            0070                        .BLKB   1
                    0071                .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072            0071                        .BLKB   1
                    0072                .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074            0072                        .BLKW   1
                    0074                .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                    0074                .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000            0074                . = 0
                    0000                .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002            0000                        .BLKW   1
00000074            0002                . = 116
                    0074                .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078            0074                        .BLKL   1
                    0078                .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C            0078                        .BLKL   1
                    007C                .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080            007C                        .BLKL   1
                    0080                .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084            0080                        .BLKL   1
                    0084                .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088            0084                        .BLKL   1
                    0088                .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C            0088                        .BLKL   1
                    008C                .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090            008C                        .BLKL   1
                    0090                .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                    0090                .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074            0090                . = 116
                    0074                .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075            0074                        .BLKB   1
                    0075                .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076            0075                        .BLKB   1
                    0076                .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077            0076                        .BLKB   1
                    0077                .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078            0077                        .BLKB   1
                    0078                .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C            0078                        .BLKL   1
0000007E            007C                        .BLKW   1
                    007E                .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080            007E                        .BLKW   1
                    0080                .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084            0080                        .BLKL   1
00000080            0084                . = 128
00000084            0080                        .BLKL   1
                    0084                .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088            0084                        .BLKL   1
                    0088                .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
```

```
0000008A  0088                          .BLKW   1
          008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                          .BLKW   1
          008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                          .BLKW   1
          008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                          .BLKW   1
          0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                          .BLKW   1
          0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                          .BLKW   1
          0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                          .BLKB   1
          0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                          .BLKB   1
          0096          .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                          .BLKW   1
00000096  0098          . = 150
00000098  0096                          .BLKW   1
          0098          .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                          .BLKL   1
          009C          .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                          .BLKL   1
          00A0          .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                          .BLKL   1
          00A4          .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                          .BLKW   1
          00A6          .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                          .BLKB   1
000000A8  00A7                          .BLKB   1
00000074  00A8          . = 116
00000078  0074                          .BLKL   1
0000007C  0078                          .BLKL   1
00000080  007C                          .BLKL   1
00000084  0080                          .BLKL   1
00000088  0084                          .BLKL   1
0000008C  0088                          .BLKL   1
          008C          .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                          .BLKL   1
00000094  0090                          .BLKL   1
00000098  0094                          .BLKL   1
0000009C  0098                          .BLKL   1
          009C          .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                          .BLKB   1
          009D          .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                          .BLKB   1
          009E          .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                          .BLKB   1
000000A0  009F                          .BLKB   1
          00A0          .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                          .BLKB   1
          00A1          .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                          .BLKB   1
          00A2          .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                          .BLKB   1
000000A4  00A3                          .BLKB   1
```

```
                    00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5            00A4                        .BLKB   1
                    00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7            00A5                        .BLKW   1
000000A8            00A7                        .BLKB   1
                    00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC            00A8                        .BLKL   1
000000B0            00AC                        .BLKL   1
000000B4            00B0                        .BLKL   1
000000B8            00B4                        .BLKL   1
                    00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC            00B8                        .BLKL   1
                    00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                    00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074            00BC                . = 116
                    0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078            0074                        .BLKL   1
                    0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C            0078                        .BLKL   1
                    007C                .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E            007C                        .BLKW   1
00000080            007E                        .BLKW   1
                    0000                .RESTORE
                    0000    137         $CRBDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
00000000            00BC                .=0
                    0000                .IIF    NB,CRB$L_WQFL,  CRB$L_WQFL:
00000004            0000                        .BLKL   1
                    0004                .IIF    NB,CRB$L_WQBL,  CRB$L_WQBL:
00000008            0004                        .BLKL   1
                    0008                .IIF    NB,CRB$W_SIZE,  CRB$W_SIZE:
0000000A            0008                        .BLKW   1
                    000A                .IIF    NB,CRB$B_TYPE,  CRB$B_TYPE:
0000000B            000A                        .BLKB   1
0000000C            000B                        .BLKB   1
                    000C                .IIF    NB,CRB$W_REFC,  CRB$W_REFC:
0000000E            000C                        .BLKW   1
                    000E                .IIF    NB,CRB$B_MASK,  CRB$B_MASK:
0000000F            000E                        .BLKB   1
00000010            000F                        .BLKB   1
                    0010                .IIF    NB,CRB$L_LINK,  CRB$L_LINK:
00000014            0010                        .BLKL   1
                    0014                .IIF    NB,CRB$L_INTD,  CRB$L_INTD:
00000038            0014                        .BLKL   9
                    0038                .IIF    NB,CRB$C_LENGTH,        CRB$C_LENGTH:
                    0038                .IIF    NB,CRB$K_LENGTH,        CRB$K_LENGTH:
                    0038                .IIF    NB,CRB$L_INTD2, CRB$L_INTD2:
0000005C            0038                        .BLKL   9
                    0000                .RESTORE
                    0000    138         $ADPDEF
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
00000000            00BC                .=0
                    0000                .IIF    NB,ADP$L_CSR,   ADP$L_CSR:
00000004            0000                        .BLKL   1
                    0004                .IIF    NB,ADP$L_LINK,  ADP$L_LINK:
```

```
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,ADP$W_SIZE,  ADP$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,ADP$B_TYPE,  ADP$B_TYPE:
0000000B  000A                                  .BLKB   1
          000B                  .IIF    NB,ADP$B_NUMBER,        ADP$B_NUMBER:
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,ADP$W_TR,    ADP$W_TR:
0000000E  000C                                  .BLKW   1
          000E                  .IIF    NB,ADP$W_ADPTYPE,       ADP$W_ADPTYPE:
00000010  000E                                  .BLKW   1
          0010                  .IIF    NB,ADP$L_VECTOR,        ADP$L_VECTOR:
          0010                  .IIF    NB,ADP$L_CRB,   ADP$L_CRB:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,ADP$C_MBAADPLEN,     ADP$C_MBAADPLEN:
          0014                  .IIF    NB,ADP$K_MBAADPLEN,     ADP$K_MBAADPLEN:
          0014                  .IIF    NB,ADP$C_DRADPLEN,      ADP$C_DRADPLEN:
          0014                  .IIF    NB,ADP$K_DRADPLEN,      ADP$K_DRADPLEN:
          0014                  .IIF    NB,ADP$L_DPQFL, ADP$L_DPQFL:
          0014                  .IIF    NB,ADP$L_PRQQFL,        ADP$L_PRQQFL:
00000018  0014                                  .BLKL   1
          0018                  .IIF    NB,ADP$L_DPQBL, ADP$L_DPQBL:
          0018                  .IIF    NB,ADP$L_PRQQBL,        ADP$L_PRQQBL:
0000001C  0018                                  .BLKL   1
          001C                  .IIF    NB,ADP$L_MRQFL, ADP$L_MRQFL:
          001C                  .IIF    NB,ADP$L_SHB,   ADP$L_SHB:
00000020  001C                                  .BLKL   1
          0020                  .IIF    NB,ADP$L_MRQBL, ADP$L_MRQBL:
          0020                  .IIF    NB,ADP$B_PORT,  ADP$B_PORT:
00000021  0020                                  .BLKB   1
00000024  0021                                  .BLKB   3
          0024                  .IIF    NB,ADP$W_DPBITMAP,      ADP$W_DPBITMAP:
00000026  0024                                  .BLKW   1
          0026                  .IIF    NB,ADP$W_MRBITMAP,      ADP$W_MRBITMAP:
00000064  0026                                  .BLKW   31
          0064                  .IIF    NB,ADP$L_INTD,  ADP$L_INTD:
00000070  0064                                  .BLKL   3
          0070                  .IIF    NB,ADP$C_MPMADPLEN,     ADP$C_MPMADPLEN:
          0070                  .IIF    NB,ADP$K_MPMADPLEN,     ADP$K_MPMADPLEN:
          0070                  .IIF    NB,ADP$C_UBAADPLEN,     ADP$C_UBAADPLEN:
          0070                  .IIF    NB,ADP$K_UBAADPLEN,     ADP$K_UBAADPLEN:
          0000                  .RESTORE
          0000    139           $VECDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .IIF    NB,VEC$Q_DISPATCH,      VEC$Q_DISPATCH:
00000008  0000                                  .BLKQ   1
          0008                  .IIF    NB,VEC$L_IDB,   VEC$L_IDB:
0000000C  0008                                  .BLKL   1
          000C                  .IIF    NB,VEC$L_INITIAL,       VEC$L_INITIAL:
00000010  000C                                  .BLKL   1
          0010                  .IIF    NB,VEC$W_MAPREG,        VEC$W_MAPREG:
00000012  0010                                  .BLKW   1
          0012                  .IIF    NB,VEC$B_NUMREG,        VEC$B_NUMREG:
00000013  0012                                  .BLKB   1
          0013                  .IIF    NB,VEC$B_DATAPATH,      VEC$B_DATAPATH:
```

```
00000014  0013                                  .BLKB   1
          0014                  .IIF    NB,VEC$L_ADP,   VEC$L_ADP:
00000018  0014                                  .BLKL   1
          0018                  .IIF    NB,VEC$L_UNITINIT,      VEC$L_UNITINIT:
0000001C  0018                                  .BLKL   1
          001C                  .IIF    NB,VEC$L_START, VEC$L_START:
00000020  001C                                  .BLKL   1
          0020                  .IIF    NB,VEC$L_UNITDISC,      VEC$L_UNITDISC:
00000024  0020                                  .BLKL   1
          0024                  .IIF    NB,VEC$C_LENGTH,        VEC$C_LENGTH:
          0024                  .IIF    NB,VEC$K_LENGTH,        VEC$K_LENGTH:
          0000                  .RESTORE
          0000    140           $DYNDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .RESTORE
          0000    141           $PSLDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .RESTORE
          0000    142           $PTEDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .RESTORE
          0000    143           $VIELD  PTE,<29-9>,<<IO,,M>>
          0000    144           $PRTDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .RESTORE
          0000    145           $PRDEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .RESTORE
          0000    146           $PR730DEF                       ; TODR now here for VMS v4      [11]
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .RESTORE
          0000    147           $DS_HPODEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
00000008  0000                  .IIF    NB,.BLKQ,       .BLKQ   1
          0008                  .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
0000000A  0008                  .IIF    NB,.BLKW,       .BLKW   1
          000A                  .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
0000000B  000A                  .IIF    NB,.BLKB,       .BLKB   1
          000B                  .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
0000000C  000B                  .IIF    NB,.BLKB,       .BLKB   1
          000C                  .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
00000018  000C                  .IIF    NB,.BLKB,       .BLKB   12
```

```
                     0018                .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
0000001C             0018                .IIF    NB,..BLKL,      .BLKL   1
                     001C                .IIF    NB,HP$A_DVA,    HP$A_DVA:
00000020             001C                .IIF    NB,..BLKL,      .BLKL   1
                     0020                .IIF    NB,HP$A_LINK,   HP$A_LINK:
00000024             0020                .IIF    NB,..BLKL,      .BLKL   1
                     0024                .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
00000026             0024                .IIF    NB,..BLKW,      .BLKW   1
                     0026                .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
00000032             0026                .IIF    NB,..BLKB,      .BLKB   12
                     0032                .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                     0000                .RESTORE
                     0000    148         $DS_HPEODEF                                             ;G:3
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             00BC                .=0
                     0000                .IIF    NB,HPE$T_DEVICE,        HPE$T_DEVICE:
00000014             0000                .IIF    NB,..BLKL,      .BLKL   5
                     0014                .IIF    NB,HPE$W_EXT_DRIVE,     HPE$W_EXT_DRIVE:
00000016             0014                .IIF    NB,..BLKW,      .BLKW   1
                     0016                .IIF    NB,HPE$A_EPB,   HPE$A_EPB:
0000001A             0016                .IIF    NB,..BLKL,      .BLKL   1
                     0000                .RESTORE
                     0000    149         $DS_DW730_DEF                   ;               [05]
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000032             00BC                .=HP$A_DEPENDENT
                     0032                .IIF    NB,DW730$K_LEN, DW730$K_LEN:
                     0000                .RESTORE
                     0000    150         $DS_DSDEF
                     0000    151         $UBIDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             00BC                .=0
                     0000                .RESTORE
                     0000    152
00000001             0000    153 SS$_NORMAL=1
00000001             0000    154 IS=1
                     0000    155
                     0000    156 ;
                     0000    157 ; Clock overhead factor
                     0000    158 ;
00000200             0000    159 DS$GK_USOVR     == ^X200                ; Overflow factor for 11/730
                     0000    160 ;
                     0000    161 ; I/O SPACE DEFINITION
                     0000    162 ;
                     0000    163
40000000             0000    164 IOC$K_IOSPACE == ^X40000000
03000000             0000    165 DS$GK_SID       ==      3@24            ; Pseudo SID for 11/730
                     0000    166
                     0000    167         .EXTRN  BUG$CHECK,      CMK$RCONSOLE, CMK$TCONSOLE,     DS$WAITUS
                     0000    168         .EXTRN  DS_ERRSUP
                     0000    169         .EXTRN  EXE_UNEXPINT
                     0000    170         .EXTRN  EXE$ALONONPAGED, EXE$DEANONPAGED,       FETCH_STORE
                     0000    171         .EXTRN  INS$PTABLE,     IOC$GL_ADPLIST
                     0000    172         .EXTRN  MAPMEM$IOSPACE
                     0000    173         .EXTRN  QIO$DEALLOCATE
```

```
                                  0000          .EXTRN  SCB_IMAGE
                                  0000          .EXTRN  SS$_CONTINUE,   SS$_CONTROLC,   SS$_CTRLERR,    SS$_DATACHECK
                                  0000          .EXTRN  SS$_RESIGNAL
                                  0000          .EXTRN  SYS$SETPRT,     SYS$UNWIND
                                  0000          .EXTRN  TST$MCHK
                                  0000          .EXTRN  UBINT_DISP,HPE$C_EPB_SIZE                                    ;G:3
                                  0000
                              00000000  .Psect  Work, Noshr, Noexe, Wrt, Long
                                  0000
                                  0000  LOCAL_STORE:
                        00000000  0000          .LONG   0
                                  0004
                                  0004  ;                                                                  begin [22]
                                  0004  SYS$SYSROOT::
59 53 24 53 59 53 0000000C'010E0000' 0004          .ASCID  "SYS$SYSROOT"           ; Device name to equate
                   54 4F 4F 52 53  0012
                                  0017  SYS$DISK::
49 44 24 53 59 53 0000001F'010E0000' 0017          .ASCID  "SYS$DISK"              ; default Device name to equate
                            4B 53  0025
                                  0027  DEF$Q_DEV::
                0000002F'00000000  0027          .LONG   0,10$                   ; Null default device
                        0000004D  002F  10$:    .BLKB   30                      ; Storage for default device
                                  004D  DEF$Q_DIR::
41 4D 53 59 53 5B 00000055'010E0000' 004D          .ASCID  "[SYSMAINT]"            ; Default load device
                      5D 54 4E 49  005B
                        000000CE  005F          .BLKB   <39*3>+4-10             ; Extra space for load device spec
                                  00CE
                                  00CE          .ALIGN  LONG                    ; VMS REQUIREMENT
                                  00D0  ;                                                                  end [22]
                              00000000  .Psect  Data, Shr, Noexe, Nowrt, Long
                                  0000          MODNAM  ONLY730
          30 33 37 59 4C 4E 4F 00' 0000      $MODULE:        .ASCIC  \ONLY730\
                              07  0000
                                  0008  IO$GQ_PHYSICAL::
                01000000 00F00000  0008          .LONG   ^XF00000,^X1000000
                                  0010
                                  0010  ; Set up location to store the primary cpu identifier number which      [17]
                                  0010  ; is needed for multi-processing;                                       [17]
                                  0010
                              00  0010  DS$GB_PRIMARY_CPU_IDENTIFIER::  .BYTE   0 ;                            [17]
                                  0011
                                  0011  ;
                                  0011  ; CPU specific values for the TIMEDWAIT macro                          [15]
                                  0011  ;
                                  0011  ONLY$GL_TENUSEC::                         ;                              [19]
                        00000001  0011          .LONG   1                       ;                              [15]
                                  0015  ONLY$GL_UBDELAY::                         ;                              [19]
                        00000001  0015          .LONG   1                       ;                              [15]
                                  0019
                                  0019  ;                                                                  begin [16]
                                  0019  .SBTTL  SHELL OF STAND ALONE SUPERVISOR MEMORY.
                              00000000  .PSECT  _LAST, PAGE
                                  0000
                                  0000  AX_BUFF::
                        00000000  0000          $$$ = .
                                  0000  ;
                                  0000  ; PROCESS CONTROL BLOCK (HARDWARE PCB).
```

```
                     0000   226 ;
                     0000   227
00000000             0000   228 PHD_BASE        == $$$
00000078             0000   229 PCB_BASE        == $$$+PHD$L_PCB
00000200             0000   230         $$$ = $$$ + 512                 ; 1 Page for PHD
                     0000   231
                     0000   232 ;
                     0000   233 ; Job Information Block and Access Rights Block
                     0000   234 ;
                     0000   235
00000200             0000   236 DS$AX_JIB        == $$$
00000274             0000   237 DS$AX_ARB        == $$$ + JIB$C_LENGTH
                     0000   238
                     0000   239 ;
                     0000   240 ; PROCESS CONTROL BLOCK (SOFTWARE PCB).
                     0000   241 ;
                     0000   242
000002EC             0000   243 DS$AX_SOFTPCB    == $$$ + JIB$C_LENGTH + ARB$C_LENGTH
                     0000   244
                     0000   245 ;
                     0000   246 ; PROCESS CONTROL BLOCK (SOFTWARE PCB).
                     0000   247 ;
                     0000   248
00000400             0000   249         $$$ = $$$ + 512                 ; 1 Page for JIB, ARB, and PCB
                     0000   250
                     0000   251 ;
                     0000   252 ; SYSTEM CONTROL BLOCK (SCB).
                     0000   253 ;
                     0000   254
00000400             0000   255 SCB_BASE         == $$$
00000800             0000   256         $$$ = $$$ + <2*512>             ; MAX 2 pages for SCB
00000800             0000   257 SCB_UNKINT==$$$
00000C00             0000   258         $$$ = $$$ + <2*512>             ; MAX 2 pages for unexpected handlers
                     0000   259
                     0000   260 ;
                     0000   261 ; STACK AREAS.
                     0000   262 ;
                     0000   263 ;       Stacks are ordered so that when a mode overflows it's stack
                     0000   264 ;       it will get an access violation because the stack below it
                     0000   265 ;       is protected by memory management. The interrupt stack is
                     0000   266 ;       above the kernel stack so that kernel stack overflows will not
                     0000   267 ;       fall into the interrupt stack.  The pages below the kernel stack
                     0000   268 ;       are NO ACCESS.  See MEMMGT for setup.
                     0000   269 ;--
                     0000   270
00000C00             0000   271 STACK_BASE       == $$$
00000E00             0000   272         $$$ = $$$ + <1*512>             ; 1 pages for no-access
                     0000   273
00002200             0000   274         $$$ = $$$ + <10*512>            ; 10 pages for KERNEL stack
00002200             0000   275 KSTKPTR == $$$
                     0000   276
00002A00             0000   277         $$$ = $$$ + <4*512>             ; 4 pages for INTERRUPT stack
00002A00             0000   278 ISTKPTR == $$$
                     0000   279
00003000             0000   280         $$$ = $$$ + <3*512>             ; 3 pages for EXECUTIVE stack
00003000             0000   281 ESTKPTR == $$$
                     0000   282
```

```
                        00003600  0000   283          $$$ = $$$ + <3*512>             ; 3 pages for SUPERVISOR stack
                        00003600  0000   284 SSTKPTR == $$$
                                  0000   285
                        00003C00  0000   286          $$$ = $$$ + <3*512>             ; 3 pages for USER stack
                        00003C00  0000   287 USTKPTR == $$$
                                  0000   288
                        00003C00  0000   289 P0PT_BASE       == $$$
                                  0000   290 ;                                                          end [16]
```

```
                        0000   292          .SBTTL  DSP$CONFIG_ADP  Build P-table for adapter
                        0000   293 ;++
                        0000   294 ; FUNCTIONAL DESCRIPTION:
                        0000   295 ;
                        0000   296 ;       This routine builds and inserts a P-table for
                        0000   297 ;       the channel/adapter addressed.
                        0000   298 ;
                        0000   299 ; CALLING SEQUENCE:
                        0000   300 ;
                        0000   301 ;       DSP$CONFIG_ADP(Adapter_address,PT_address,type_address)
                        0000   302 ;
                        0000   303 ; INPUT PARAMETERS:
                        0000   304 ;
                        0000   305 ;       4(AP)   Address of configuration status register of adapter
                        0000   306 ;
                        0000   307 ; IMPLICIT INPUTS:      NONE
                        0000   308 ;
                        0000   309 ; OUTPUT PARAMETERS:
                        0000   310 ;
                        0000   311 ;       8(AP)   Address to store address of P-table created
                        0000   312 ;       12(AP)  Address to store type of adapter, 0=MBA, 1=UBA
                        0000   313 ;
                        0000   314 ; IMPLICIT OUTPUTS:
                        0000   315 ;
                        0000   316 ;       P-table for Adapter
                        0000   317 ;
                        0000   318 ; COMPLETION CODES:
                        0000   319 ;
                        0000   320 ;       SS$_NORMAL
                        0000   321 ;       DS$_MACHCK
                        0000   322 ;
                        0000   323 ; SIDE EFFECTS:
                        0000   324 ;
                        0000   325 ;       The adapter specified is pinged to determine
                        0000   326 ;       interrupt level and interrupt vector.
                        0000   327 ;
                        0000   328 ; REGISTER USAGE:
                        0000   329 ;
                        0000   330 ;       R4      Address of current PT
                        0000   331 ;       R5      Address of channel
                        0000   332 ;--
                        0000   333          .NLIST  ME,MEB
                        0000   334          $OFFSET 0,NEGATIVE, < -
                        0000   335                  <L_VECTOR,4>, -
                        0000   336                  <L_IPL,4>, -
                        0000   337                  <L_CONFIG,4>, -
                        0000   338                  <L_LOCAL,0>>
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT $ABS$ ABS
00000000                00BC                .=0
FFFFFFFC                0000                .BLKB   -4
                        FFFC        L_VECTOR:
FFFFFFF8                FFFC                .BLKB   -4
                        FFF8        L_IPL:
FFFFFFF4                FFF8                .BLKB   -4
                        FFF4        L_CONFIG:
                        FFF4        L_LOCAL:
```

```
                        00000000                .RESTORE
                            0000    339
                            0000    340         .LIST   MEB
```

```
                                00000000   342 .Psect  Code, Shr, Exe, Nowrt, Long
                                    0000   343
                                019C 0000   344 .ENTRY  DSP$CONFIG_ADP,^M<R2,R3,R4,R7,R8>                                ;G:3
                                    0002   345
                                    0002   346 ;       MOVAB   W^HANDLR,(FP)                   ; Exception handler
                      5E   F4 AD    9E 0002   347         MOVAB   L_LOCAL(FP),SP                ; Allocate space for local storage
              00000000'EF    5D    D0 0006   348         MOVL    FP,L^LOCAL_STORE              ; And save address for interrupts       [07]
                        FC AD    D4 000D   349         CLRL    L_VECTOR(FP)                  ; Clear vector
                        F8 AD    D4 0010   350         CLRL    L_IPL(FP)                     ; Clear IPL
                                    0013   351
        54   04 AC   40000000 8F    C9 0013   352         BISL3   #IOC$K_IOSPACE,4(AP),R4 ; Base adapter
                         F4 AD    54 D0 001C   353         MOVL    R4,L_CONFIG(FP)               ; Store Configuration register addr
                            50    64 D0 0020   354         MOVL    (R4),R0                       ; Get Configuration register
              50   50    05    03 EF 0023   355         EXTZV   #3,#5,R0,R0                   ; Get adapter type
                                    0028   356         CASE    R0,LIMIT=#5,DISPLIST=<40$>; UBI only
                    00'  05    50 AF 0028             CASEW   R0,#5,S^#<<30001$-30000$>/2>-1
                                    002C       30000$:
                                001B' 002C             .SIGNED_WORD    40$-30000$
                                    002E       30001$:
                                    002E   357         ERRSUP_S                              ; Unknown adapter type
                     00000000'EF    DF 002E                 PUSHAL  $MODULE
                                00  DD 0034                 PUSHL   #0
                                01  DD 0036                 PUSHL   #$ER
                     00000000'EF    03 FB 0038             CALLS   #3, DS_ERRSUP
                     50   00660002 8F    D0 003F   358         MOVL    #DS$_ERROR,R0                 ; Error
                                    04 0046   359         RET
                                    0047   360
                      0C BC    01    D0 0047   361 40$:    MOVL    #1,@12(AP)                    ; Set adapter type to UBA
                         51    32 D0 004B   362         MOVL    #DW730$K_LEN,R1               ; Set length of PT                       [05]
                   51   00000000'8F    C0 004E   363         ADDL2   #HPE$C_EPB_SIZE,R1            ; Add size of EPB                       ;G:3
                              FFA8' 30 0055   364         BSBW    EXE$ALONONPAGED               ; Allocate space for PT
                                 04 50 E8 0058   365         BLBS    R0,60$                        ; Branch if successful
                                5E    08 C0 005B   366         ADDL    #8,SP                         ; Reset stack
                                    04 005E   367 50$:    RET                                   ; Return on error
                                    005F   368
                1C A2   40FC0000 8F    D0 005F   369 60$:    MOVL    #^X40FC0000,HP$A_DVA(R2); Set device virtual address
                         08 A2    51 B0 0067   370         MOVW    R1,HP$W_SIZE(R2)              ; Set length of PT
                      58   52    51 C1 006B   371         ADDL3   R1,R2,R8                      ; End of extended ptable block          ;G:3
                      57   58    04 C3 006F   372         SUBL3   #4,R8,R7                      ; Address to store HPE$A_EPB            ;G:3
                67   58   00000000'8F    C3 0073   373         SUBL3   #HPE$C_EPB_SIZE,R8,(R7) ; Store address                         ;G:3
                            58    67 D0 007B   374         MOVL    (R7),R8                       ; Address of EPB in R8                  ;G:3
                         0B A2    00 90 007E   375         MOVB    #0,HP$B_DRIVE(R2)             ; Drive #                               ;G:3
                         14 A8    00 B0 0082   376         MOVW    #0,HPE$W_EXT_DRIVE(R8)        ; Drive # in EPB                        ;G:3
                0C A2   3057445F 8F    D0 0086   377         MOVL    #^A"_DW0",HP$T_DEVICE(R2); Set device name
                   26 A2    4405 8F    B0 008E   378         MOVW    #^X4405,HP$T_TYPE(R2)         ; Set length and "D" of "DW750"
                28 A2   30333757 8F    D0 0094   379         MOVL    #^A"W730",HP$T_TYPE+2(R2); Set rest of "W730"                   [05]
                   24 A2    0200 8F    B0 009C   380         MOVW    #^X200,HP$W_VECTOR(R2)        ; Set SCB base for UBIO
                                    00A2   381
                            62    04 D0 00A2   382 70$:    MOVL    #4,HP$Q_DEVICE(R2)            ; Set length of device name
                      04 A2    0C A2 9E 00A5   383         MOVAB   HP$T_DEVICE(R2), -
                                    00AA   384                 HP$Q_DEVICE+4(R2)             ; Set address of device name
                         18 A2    54 D0 00AA   385         MOVL    R4,HP$A_DEVICE(R2)            ; Set address of device
                                    00AE   386
                            20 A2    D4 00AE   387         CLRL    HP$A_LINK(R2)                 ; Clear address of link PT
                         08 BC    52 D0 00B1   388         MOVL    R2,@8(AP)                     ; Store address of PT just made
                   00000000'EF    62 FA 00B5   389         CALLG   (R2),INS$PTABLE               ; Insert this in the table
                              0D 50 E8 00BC   390         BLBS    R0,CONFIG_EXIT                ; Exit complete
```

```
                                00BF   391
                                00BF   392 CONFIG_RELEASE:
                     50   52    D0  00BF   393          MOVL    R2,R0                   ; Copy address of buffer
                        FF3B'   30  00C2   394          BSBW    EXE$DEANONPAGED         ; Deallocate buffer
                50   00660002 8F   D0  00C5   395          MOVL    #DS$_ERROR,R0           ; Flag error
                                00CC   396 CONFIG_EXIT:
                                04  00CC   397          RET
```

```
                        0004  00CD   399 HANDLR: .WORD   ^M<R2>                    ; Unexpected exception routine
      52   00000000'EF    D0  00CF   400         MOVL    L^LOCAL_STORE,R2          ; Base stack                          [07]
           51    04 BC    DE  00D6   401         MOVAL   @4(AP),R1                 ; Point to signal arguments
         50    0000'8F    3C  00DA   402         MOVZWL  #SS$_RESIGNAL,R0          ; Assume not ours
                                00DF   403
   04 A1   006600D8 8F    D1  00DF   404         CMPL    #DS$_UNEXPINT,4(R1)       ; Unexpected interrupt?
                     14    12  00E7   405         BNEQ    10$                       ; Branch if not
        FC A2    08 A1    D0  00E9   406         MOVL    8(R1),L_VECTOR(R2)        ; Store interrupt vector
           F8 A2    12    DB  00EE   407         MFPR    #PR$_IPL,L_IPL(R2)        ; Save current IPL
           50    F4 A2    D0  00F2   408         MOVL    L_CONFIG(R2),R0           ; Get Configuration register address
      50   00000000'8F    D0  00F6   409         MOVL    #SS$_CONTINUE,R0          ; Continue from exception
                                00FD   410
                           04  00FD   411 10$:    RET
                                00FE   412
```

```
                        .SBTTL  MAP$IOSPACE     Validate mapping for adapters
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This is routine is called from MAPMEM (MEMMGT) to
                ;       validate those P1 space address that map thru to physical
                ;       I/O space... bit 29=1
                ;       For each slot, 16 pages are mapped.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    MAP$IOSPACE
                ;
                ; INPUT PARAMETERS:     NONE
                ;
                ;       R5      Address of P1 page tables
                ;
                ; IMPLICIT INPUTS:      NONE
                ;
                ; OUTPUT PARAMETERS:    NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       Updates P1 page tables
                ;
                ; SIDE EFFECTS:         NONE
                ;--
```

```
                                  00FE   442 MAP$IOSPACE::
                        18   BB   00FE   443          PUSHR   #^M<R3,R4>                  ; Save
              00000010'EF    94   0100   444          CLRB    DS$GB_PRIMARY_CPU_IDENTIFIER    ; =s zero for non-mp systems    [17]
        54    00F00000 8F    D0   0106   445          MOVL    #^XF00000,R4                ; Lowest slot address
                        7E   DF   010D   446 10$:     PUSHAL  -(SP)                       ; Address to store longword
                        10   BB   010F   447          PUSHR   #^MR4                       ; Address to fetch
        00000152'EF     02   FB   0111   448          CALLS   #2,FETCH_LONG               ; Check it out
                        08   BA   0118   449          POPR    #^MR3                       ; Get contents
                     12 50   E9   011A   450          BLBC    R0,50$                      ; Branch if nothing there
                                  011D   451
         51    54    F7 8F   78   011D   452          ASHL    #-9,R4,R1                   ; PFN of slot
        52    A0000000 8F    D0   0122   453          MOVL    #PTE$M_VALID!PTE$C_UW,R2;PROTECTION
                  50    10   D0   0129   454          MOVL    #16,R0                      ; Map 16 pages
                     FED1'   30   012C   455          BSBW    MAPMEM$IOSPACE              ; Call to setup PTE's
                                  012F   456
54   00002000 8F   00F7FFFF 8F   F1   012F   457 50$:     ACBL    #^XF7FFFF,#^X2000,R4,10$; Do each slot 0-63
                            FFD0   013B
                                  013D   458
              51    7FF0 8F   3C   013D   459          MOVZWL  #^XFFE000a-9,R1             ; PFN of unibus ^O760000
        52    A0000000 8F    D0   0142   460          MOVL    #PTE$M_VALID!PTE$C_UW,R2;PROTECTION
                  50    10   D0   0149   461          MOVL    #16,R0                      ; Map 16 pages
                     FEB1'   30   014C   462          BSBW    MAPMEM$IOSPACE              ; Call to setup PTE's
                                  014F   463
                        18   BA   014F   464          POPR    #^M<R3,R4>
                             05   0151   465          RSB
```

```
                                0152   467          .SBTTL  FETCH_LONG      Retreive longword from possibly bad address
                                0152   468 ;++
                                0152   469 ; FUNCTIONAL DESCRIPTION:
                                0152   470 ;
                                0152   471 ;       This routine is used to fetch a longword from the designated location
                                0152   472 ;       It handles machine checks if they occur and return the indicated status
                                0152   473 ;
                                0152   474 ; CALLING SEQUENCE:
                                0152   475 ;
                                0152   476 ;       FETCH_LONG(ADDRESS,RESULT)
                                0152   477 ;
                                0152   478 ; INPUT PARAMETERS:
                                0152   479 ;
                                0152   480 ;       ADDRESS Address of Longword to fetch
                                0152   481 ;
                                0152   482 ; IMPLICIT INPUTS:      NONE
                                0152   483 ;
                                0152   484 ; OUTPUT PARAMETERS:
                                0152   485 ;
                                0152   486 ;       RESULT  Address to store the fetched longword
                                0152   487 ;
                                0152   488 ; IMPLICIT OUTPUTS:     NONE
                                0152   489 ;
                                0152   490 ;--
                                0152   491
                        0000    0152   492 .ENTRY  FETCH_LONG,^M<>
                                0154   493
          6D    61'AF    9E    0154   494          MOVAB   B^20$,(FP)              ; Set exception handler
       08 BC    04 BC    D0    0158   495          MOVL    @4(AP),@8(AP)           ; Fetch
             50    01    D0    015D   496          MOVL    S^#SS$_NORMAL,R0        ; Success
                         04    0160   497          RET
                               0161   498
                        0000   0161   499 20$:     .WORD   ^M<>                    ; Save no registers
          50    04 AC    7D    0163   500          MOVQ    4(AP),R0                ; Get signal and mechanism pointers
    04 A0    00660088 8F    D1    0167   501          CMPL    #DS$_MCHK,4(R0)         ; Machine check?
                      0F    12    016F   502          BNEQ    30$                     ; Branch if not
       0C A1    04 A0    D0    0171   503          MOVL    4(R0),12(R1)            ; Change return code
                      7E    7C    0176   504          CLRQ    -(SP)                   ; Null args to unwind
    00000000'EF       02    FB    0178   505          CALLS   #2,SYS$UNWIND           ; Unwind the stack
                         04    017F   506          RET                             ; And return
                               0180   507
          50    0000'8F    3C    0180   508 30$:     MOVZWL  #SS$_RESIGNAL,R0        ; Let someone else have it
                         04    0185   509          RET                             ; Return
                               0186   510
```

```
                     0186   512          .SBTTL  PURGE DATAPATH
                     0186   513 ;+
                     0186   514 ; IOC$PURGDATAP - PURGE DATAPATH
                     0186   515 ;
                     0186   516 ; THIS ROUTINE PURGES THE CALLER'S BUFFERED DATAPATH, AND CLEARS ANY
                     0186   517 ; DATAPATH ERRORS.  IF THERE WAS A DATAPATH ERROR, THIS FACT IS
                     0186   518 ; RETURNED TO THE CALLER.
                     0186   519 ;
                     0186   520 ; INPUTS:
                     0186   521 ;
                     0186   522 ;       R5 = UCB ADDRESS
                     0186   523 ;
                     0186   524 ; OUTPUTS:
                     0186   525 ;
                     0186   526 ;       R0-R3 ALTERED
                     0186   527 ;       OTHER REGISTERS PRESERVED
                     0186   528 ;       R0 LOW BIT CLEAR/SET IF TRANSMISSION ERROR/SUCCESS
                     0186   529 ;       R1 = DPR CONTENTS AFTER PURGE (FOR REGISTER DUMP BY CALLER)
                     0186   530 ;       R2 = ADDRESS OF START OF ADAPTER MAP REGISTERS (FOR REG DUMP BY CALLER)
                     0186   531 ;       R3 = CRB ADDRESS
                     0186   532 ;-
                     0186   533
                     0186   534
                     0186   535
                     0186   536 IOC$PURGDATAP::
     53    20 A5   D0  0186   537          MOVL    UCB$L_CRB(R5),R3        ;CRB ADDRESS
     52    28 B3   D0  018A   538          MOVL    @CRB$L_INTD+VEC$L_ADP(R3),R2 ;GET START OF ADAPTER
52   00000800 8F   C0  018E   539          ADDL2   #^X800, R2              ; Offset address to MAP regs.          [04]
        50    01   7D  0195   540          MOVQ    #1,R0                   ;R0=1, R1= 0
                   05  0198   541          RSB                             ;RETURN
```

```
                                0199   543 ;++
                                0199   544 ; 11/750 SPECIFIC UBI INITIALIZATION
                                0199   545 ; AND ADAPTER BLOCK BUILD
                                0199   546 ;--
                                0199   547 UBADP_CPU::
00000600'EF   DE                0199   548          MOVAL   L^SCB_BASE+^X200,-      ; UB0                                  [06]
      10 A2                     019F   549                  ADP$L_VECTOR(R2)        ; VECTOR SPACE
 24 A2   0E   B0                01A1   550          MOVW    #^XE,ADP$W_DPBITMAP(R2) ; MARK DATAPATHS 1-3 AVAILABLE
              05                01A5   551          RSB                             ; RETURN TO COMMON CODE
                                01A6   552
                       000000B4 01A6   553 UBINTSZ==^XB4                           ; Used by QIO                          [12]
```

```
                    01A6   555          .SBTTL  UBA$INITIAL     UNIBUS ADAPTER INITIALIZATION
                    01A6   556 ;+
                    01A6   557 ; UBA$INITIAL - UNIBUS ADAPTER INITIALIZATION
                    01A6   558 ;
                    01A6   559 ; THIS ROUTINE IS CALLED VIA A JSB INSTRUCTION AT SYSTEM STARTUP AND AFTER
                    01A6   560 ; A POWER RECOVERY RESTART TO ALLOW INITIALIZATION OF UNIBUS ADAPTERS.
                    01A6   561 ;
                    01A6   562 ; INPUTS:
                    01A6   563 ;
                    01A6   564 ;       R4 = ADDRESS OF UNIBUS ADAPTER CONFIGURATION STATUS REGISTER.
                    01A6   565 ;
                    01A6   566 ;       ALL INTERRUPTS ARE LOCKED OUT.
                    01A6   567 ;
                    01A6   568 ; OUTPUTS:
                    01A6   569 ;
                    01A6   570 ;       THE UNIBUS ADAPTER IS INITIALIZED AND INTERRUPTS ARE ENABLED.
                    01A6   571 ;-
                    01A6   572
                    01A6   573 UBA$INITIAL::                           ;UNIBUS ADAPTER INITIALIZATION
                05  01A6   574          RSB                           ;
                    01A7   575 ;
                    01A7   576 ; IGNORE UNEXPECTED UNIBUS INTERRUPTS
                    01A7   577 ;
                    01A7   578
                    01A7   579          .ALIGN  LONG
                    01A8   580 UBA$UNEXINT::                           ; UNEXPECTED INTERRUPT CODE
                02  01A8   581          REI                           ; AND RETURN
```

```
                                01A9   583          .SBTTL  IO$TESTCSR_x    Test CSR for existence
                                01A9   584 ;++
                                01A9   585 ;
                                01A9   586 ; IO$TESTCSR_L - TEST A LONGWORD FOR EXISTENCE
                                01A9   587 ; IO$TESTCSR_W - TEST A WORD FOR EXISTENCE IN I/O SPACE
                                01A9   588 ;
                                01A9   589 ; THIS TEST IS CPU DEPENDENT. THE FOLLOWING CPU'S ARE SUPPORTED:
                                01A9   590 ;
                                01A9   591 ;       11/780 - TEST CSR AND CHECK RESULT IN THE UBA STATUS REGISTER.
                                01A9   592 ;       11/750 - NOM-EXISTENT CSR IS REPORTED VIA MACHINE CHECK AS A
                                01A9   593 ;     & 11/730   NON-EXISTENT MEMORY REFERENCE. CONNECT A TEMPORARY
                                01A9   594 ;                MACHINE CHECK HANDLER, TEST THE CSR, AND RESTORE THE
                                01A9   595 ;                ORIGINAL MACHINE CHECK HANDLER.
                                01A9   596 ;
                                01A9   597 ; INPUTS:
                                01A9   598 ;
                                01A9   599 ;       R0 = CSR ADDRESS
                                01A9   600 ;       R6 = ADAPTER CONFIGURATION REGISTER
                                01A9   601 ;
                                01A9   602 ; OUTPUTS:
                                01A9   603 ;
                                01A9   604 ;       R0 = LOW BIT SET/CLEAR FOR EXISTENT/NONEX CSR
                                01A9   605 ;            OTHER REGISTERS ARE PRESERVED.
                                01A9   606 ;
                                01A9   607 ;--
                                01A9   608 IO$TESTCSR_L::
                     02     BB  01A9   609          PUSHR   #^M<R1>                 ; Save a register
           51 00000404'EF     DE  01AB   610          MOVAL   SCB_BASE+4,R1           ; Base SCB
                     61     DD  01B2   611          PUSHL   (R1)                    ; Save previous machine check handler
           61 00000000'EF     DE  01B4   612          MOVAL   TST$MCHK,(R1)           ; Set machine check handler
                     60     D5  01BB   613          TSTL    (R0)                    ; Test for existence
                     61   8ED0  01BD   614          POPL    (R1)                    ; Restore previous MCHK handler
                     50     DC  01C0   615          MOVPSL  R0                      ; C-bit set indicates MCHK
                50   50     D2  01C2   616          MCOML   R0,R0                   ; C-bit clear indicates ok
                     02     BA  01C5   617          POPR    #^M<R1>                 ; Restore
                     05     RSB 01C7   618          RSB                             ; Return
                                01C8   619
                                01C8   620 IO$TESTCSR_W::
                     02     BB  01C8   621          PUSHR   #^M<R1>                 ; Save a register
           51 00000404'EF     DE  01CA   622          MOVAL   SCB_BASE+4,R1           ; Base SCB
                     61     DD  01D1   623          PUSHL   (R1)                    ; Save previous machine check handler
           61 00000000'EF     DE  01D3   624          MOVAL   TST$MCHK,(R1)           ; Set machine check handler
                     60     B5  01DA   625          TSTW    (R0)                    ; Test for existence
                     61   8ED0  01DC   626          POPL    (R1)                    ; Restore previous MCHK handler
                     50     DC  01DF   627          MOVPSL  R0                      ; C-bit set indicates MCHK
                50   50     D2  01E1   628          MCOML   R0,R0                   ; C-bit clear indicates ok
                     02     BA  01E4   629          POPR    #^M<R1>                 ; Restore
                     05     RSB 01E6   630          RSB                             ; Return
```

```
                                01E7   632          .SBTTL  IOGEN$CONN_VEC
                                01E7   633 ;++
                                01E7   634 ;
                                01E7   635 ; IOGEN$CONN_VEC - CONNECT A UNIBUS INTERRUPT DISPATCHER TO A VECTOR
                                01E7   636 ;
                                01E7   637 ; THIS SUBROUTINE IS CPU-DEPENDENT. THE FOLLOWING CPU'S ARE SUPPORTED:
                                01E7   638 ;
                                01E7   639 ;       11/780 - CONNECT VEC$Q_DISPATCH+2 (JSB a#) TO VECTOR
                                01E7   640 ;       11/750 - CONNECT VEC$Q_DISPATCH (PUSHR) TO VECTOR
                                01E7   641 ;
                                01E7   642 ; FOR ALL CPU'S, PUSHR #^M<R2,R3,R4,R5> IN THE INTERRUPT DISPATCH BLOCK
                                01E7   643 ; IS CHANGED TO PUSHR #^M<R0,R1,R2,R3,R4,R5>.
                                01E7   644 ;
                                01E7   645 ; INPUTS:
                                01E7   646 ;
                                01E7   647 ;       R0 = ADDRESS OF VECTOR TO CONNECT
                                01E7   648 ;       R4 = ADDRESS OF INTERRUPT DISPATCH BLOCK IN CRB
                                01E7   649 ;
                                01E7   650 ; OUTPUTS:
                                01E7   651 ;
                                01E7   652 ;       ALL REGISTERS PRESERVED
                                01E7   653 ;--
                                01E7   654 IOGEN$CONN_VEC::
     64   00000000'EF    B0    01E7   655          MOVW    UBINT_DISP,VEC$Q_DISPATCH(R4)   ; CHANGE PUSHR TO PUSH R0-R5
          60    64       9E    01EE   656          MOVAB   VEC$Q_DISPATCH(R4),(R0)         ; CONNECT PUSHR TO VECTOR
                         05    01F1   657          RSB                                     ; RETURN
```

```
                                01F2   659          .SBTTL  QIOCLEAN_CPU
                                01F2   660 ;
                                01F2   661 ; CPU SPECIFIC QIO CLEANUP
                                01F2   662 ;       R4 = SCB_IMAGE
                                01F2   663 ;       R5 = SCB_BASE
                                01F2   664 ;
                                01F2   665 QIOCLEAN_CPU::
                      33   BB   01F2   666          PUSHR   #^M<R0,R1,R4,R5>        ; SAVE R0,R1,R4,R5
              50    0080 8F   3C   01F4   667          MOVZWL  #<512/4>,R0             ; Start with SCB page 2, offset ^X200
      6540    00000800'EF40   DE   01F9   668 10$:     MOVAL   SCB_UNKINT[R0],(R5)[R0] ; Re-fill SCB with pointer to UNK_INT
       EF 50    00000180 8F   F2   0202   669          AOBLSS  #3*<512/4>,R0,10$       ; Fill 2 SCB vector pages
          53    00000000'EF   DE   020A   670          MOVAL   L^IOC$GL_ADPLIST,R3     ; POINT TO THE ADAPTER HEADER           [20]
                              0211   671 20$:
                   50   63   D0   0211   672          MOVL    (R3),R0                 ; POINT TO THE FIRST ADAPTER BLOCK
                        09   13   0214   673          BEQL    30$                     ; FINISH IF EMPTY
                    63   04 A0   D0   0216   674          MOVL    ADP$L_LINK(R0),(R3)     ; LINK TO THE NEXT
                       FDE3'   30   021A   675          BSBW    QIO$DEALLOCATE          ; RELEASE THE ADAPTER CONTROL BLOCK
                         F2    11   021D   676          BRB     20$                     ; RELEASE THEM ALL
                              021F   677 30$:
                      33   BA   021F   678          POPR    #^M<R0,R1,R4,R5>        ; RESTORE R0,R1,R4,R5
                           05   0221   679          RSB                             ; Return
```

```
                     0222   681          .SBTTL  ONLY_SCB        CPU specific SCB setup
                     0222   682 ;++                                                          [08]
                     0222   683 ; FUNCTIONAL DESCRIPTION:
                     0222   684 ;
                     0222   685 ;       Does nothing for the case of NEBULA.
                     0222   686 ;
                     0222   687 ;--                                                          [08]
                     0222   688
                     0222   689 ONLY_SCB::                              ;                  [08]
                  05 0222   690          RSB                           ;                  [08]
```

```
                0223          .SBTTL  HDW_CLOCK_SET   CPU specific clock setup
                0223  ;++                                                                    [14]
                0223  ; FUNCTIONAL DESCRIPTION:
                0223  ;
                0223  ;       Does nothing for the case of NEBULA.
                0223  ;
                0223  ;--                                                                    [14]
                0223
                0223  HDW_CLOCK_SET::                          ;                             [14]
          05    0223          RSB                              ;                             [14]
```

```
        0224          .SBTTL  ONLY_ATTACH                                    ; [18]
        0224    ;++
        0224    ; FUNCTIONAL DESCRIPTION:
        0224    ;
        0224    ;       Does nothing for the case of NEBULA.
        0224    ;
        0224    ;--
        0224
        0224    ONLY_ATTACH::                                          ;
   05   0224            RSB                                            ;
```

```
                     0225   714          .SBTTL  SCBVECTOR_xxx   Otherwise undefined SCB vectors
                     0225   715 ;++
                     0225   716 ; FUNCTIONAL DESCRIPTION:
                     0225   717 ;
                     0225   718 ;       These interrupt entry points declare unexpected exceptions
                     0225   719 ;       for all unused SCB vectors.
                     0225   720 ;
                     0225   721 ;--
                     0225   722          .LIST   MEB,MC
                     0225   723
                     0225   724          .MACRO  SPAREVECTOR     OFFSET
                     0225   725          .ALIGN  LONG
                     0225   726 SCBVECTOR_'OFFSET'==.+IS
                     0225   727          BSBB    COMMON_SCB                      ; Enter common handler
                     0225   728          .WORD   ^X'OFFSET'
                     0225   729          .ENDM
                     0225   730                                                                  ;G:3
                     0225   731          SPAREVECTOR     000
                     0225                .ALIGN  LONG
            7A   10  0228                BSBB    COMMON_SCB                      ; Enter common handler
              0000   022A                .WORD   ^X000
                     022C   732          SPAREVECTOR     038
            76   10  022C                BSBB    COMMON_SCB                      ; Enter common handler
              0038   022E                .WORD   ^X038
                     0230   733          SPAREVECTOR     03C
            72   10  0230                BSBB    COMMON_SCB                      ; Enter common handler
              003C   0232                .WORD   ^X03C
                     0234   734          SPAREVECTOR     050
            6E   10  0234                BSBB    COMMON_SCB                      ; Enter common handler
              0050   0236                .WORD   ^X050
                     0238   735          SPAREVECTOR     054
            6A   10  0238                BSBB    COMMON_SCB                      ; Enter common handler
              0054   023A                .WORD   ^X054
                     023C   736          SPAREVECTOR     058
            66   10  023C                BSBB    COMMON_SCB                      ; Enter common handler
              0058   023E                .WORD   ^X058
                     0240   737          SPAREVECTOR     05C
            62   10  0240                BSBB    COMMON_SCB                      ; Enter common handler
              005C   0242                .WORD   ^X05C
                     0244   738          SPAREVECTOR     060
            5E   10  0244                BSBB    COMMON_SCB                      ; Enter common handler
              0060   0246                .WORD   ^X060
                     0248   739          SPAREVECTOR     064
            5A   10  0248                BSBB    COMMON_SCB                      ; Enter common handler
              0064   024A                .WORD   ^X064
                     024C   740          SPAREVECTOR     068
            56   10  024C                BSBB    COMMON_SCB                      ; Enter common handler
              0068   024E                .WORD   ^X068
                     0250   741          SPAREVECTOR     06C
            52   10  0250                BSBB    COMMON_SCB                      ; Enter common handler
              006C   0252                .WORD   ^X06C
                     0254   742          SPAREVECTOR     070
            4E   10  0254                BSBB    COMMON_SCB                      ; Enter common handler
              0070   0256                .WORD   ^X070
                     0258   743          SPAREVECTOR     074
            4A   10  0258                BSBB    COMMON_SCB                      ; Enter common handler
              0074   025A                .WORD   ^X074
```

```
                    025C   744          SPAREVECTOR     078
          46   10   025C                BSBB    COMMON_SCB                  ; Enter common handler
             0078   025E                .WORD   ^X078
                    0260   745          SPAREVECTOR     07C
          42   10   0260                BSBB    COMMON_SCB                  ; Enter common handler
             007C   0262                .WORD   ^X07C
                    0264   746          SPAREVECTOR     080
          3E   10   0264                BSBB    COMMON_SCB                  ; Enter common handler
             0080   0266                .WORD   ^X080
                    0268   747          SPAREVECTOR     0C4
          3A   10   0268                BSBB    COMMON_SCB                  ; Enter common handler
             00C4   026A                .WORD   ^X0C4
                    026C   748          SPAREVECTOR     0C8
          36   10   026C                BSBB    COMMON_SCB                  ; Enter common handler
             00C8   026E                .WORD   ^X0C8
                    0270   749          SPAREVECTOR     0CC
          32   10   0270                BSBB    COMMON_SCB                  ; Enter common handler
             00CC   0272                .WORD   ^X0CC
                    0274   750          SPAREVECTOR     0D0
          2E   10   0274                BSBB    COMMON_SCB                  ; Enter common handler
             00D0   0276                .WORD   ^X0D0
                    0278   751          SPAREVECTOR     0D4
          2A   10   0278                BSBB    COMMON_SCB                  ; Enter common handler
             00D4   027A                .WORD   ^X0D4
                    027C   752          SPAREVECTOR     0D8
          26   10   027C                BSBB    COMMON_SCB                  ; Enter common handler
             00D8   027E                .WORD   ^X0D8
                    0280   753          SPAREVECTOR     0DC
          22   10   0280                BSBB    COMMON_SCB                  ; Enter common handler
             00DC   0282                .WORD   ^X0DC
                    0284   754          SPAREVECTOR     0E0
          1E   10   0284                BSBB    COMMON_SCB                  ; Enter common handler
             00E0   0286                .WORD   ^X0E0
                    0288   755          SPAREVECTOR     0E4
          1A   10   0288                BSBB    COMMON_SCB                  ; Enter common handler
             00E4   028A                .WORD   ^X0E4
                    028C   756          SPAREVECTOR     0E8
          16   10   028C                BSBB    COMMON_SCB                  ; Enter common handler
             00E8   028E                .WORD   ^X0E8
                    0290   757          SPAREVECTOR     0EC
          12   10   0290                BSBB    COMMON_SCB                  ; Enter common handler
             00EC   0292                .WORD   ^X0EC
                    0294   758          SPAREVECTOR     0F0
          0E   10   0294                BSBB    COMMON_SCB                  ; Enter common handler
             00F0   0296                .WORD   ^X0F0
                    0298   759          SPAREVECTOR     0F4
          0A   10   0298                BSBB    COMMON_SCB                  ; Enter common handler
             00F4   029A                .WORD   ^X0F4
                    029C   760          SPAREVECTOR     0F8
          06   10   029C                BSBB    COMMON_SCB                  ; Enter common handler
             00F8   029E                .WORD   ^X0F8
                    02A0   761          SPAREVECTOR     0FC
          02   10   02A0                BSBB    COMMON_SCB                  ; Enter common handler
             00FC   02A2                .WORD   ^X0FC
                    02A4   762 COMMON_SCB:
   6E   00 BE   3C   02A4   763          MOVZWL  @(SP),(SP)              ; Get vector offset
         FD55'   31   02A8   764          BRW     EXE_UNEXPINT            ; Generate unexpected interrupt
```

```
                                  02AB                                                  ; Build the console ptable
                                  02AB
                                  02AB          .SBTTL  DSR$ONLY_CONSOLE_CONFIG  console device configuration for load path
                                  02AB  ;++
                                  02AB  ; FUNCTIONAL DESCRIPTION:
                                  02AB  ;
                                  02AB  ;       This subroutine sets up the P-tables for the console load device(s).
                                  02AB  ;
                                  02AB  ; CALLING SEQUENCE:
                                  02AB  ;
                                  02AB  ;       JSB     DSR$ONLY_CONSOLE_CONFIG
                                  02AB  ;
                                  02AB  ; INPUT PARAMETERS:     NONE
                                  02AB  ;
                                  02AB  ; IMPLICIT INPUTS:      NONE
                                  02AB  ;
                                  02AB  ; OUTPUT PARAMETERS:    NONE
                                  02AB  ;
                                  02AB  ; IMPLICIT OUTPUTS:
                                  02AB  ;
                                  02AB  ;       P-talbles for each device in the console load path.
                                  02AB  ;
                                  02AB  ; SIDE EFFECTS:
                                  02AB  ;
                                  02AB  ;       R0, R1, and R2 are destroyed.
                                  02AB  ;
                                  02AB  ;--
                                  02AB  DSR$ONLY_CONSOLE_CONFIG::                                                  ;G:3
                                  02AB                                                  ; Build console P-table for [21] ;G:
                                  02AB                                                  ; this particular processor   ;G:3
                 51    32    3C   02AB          MovZWL  #Hp$A_Dependent, R1             ; Length of Ptable to make
        51   00000000'8F    C0   02AE          ADDL2   #HPE$C_EPB_SIZE,R1              ; Add size of EPB             ;G:3
             00000000'EF    16   02B5          Jsb     L^Exe$AloNonPaged               ; Allocate it
                    63 50    E9   02BB          BlbC    R0, 10$                         ; Skip setup if couldn't do it
                       3E    BB   02BE          PushR   #^M<R1,R2,R3,R4,R5>             ; Save volatile registers
 62   51   00   6E    00    2C   02C0          MovC5   #0, (Sp), #0, R1, (R2)          ; Clear the Ptable
                       3E    BA   02C6          PopR    #^M<R1,R2,R3,R4,R5>             ; Restore registers
            58   52    51    C1   02C8          ADDL3   R1,R2,R8                        ; End of extended ptable block ;G:3
            57   58    04    C3   02CC          SUBL3   #4,R8,R7                        ; Address to store HPE$A_EPB  ;G:3
 67   58   00000000'8F    C3   02D0          SUBL3   #HPE$C_EPB_SIZE,R8,(R7)         ; Store address               ;G:3
                 58    67    D0   02D8          MOVL    (R7),R8                         ; Address of EPB in R8        ;G:3
              08 A2    51    B0   02DB          MovW    R1, Hp$W_Size(R2)               ; Set size field
              0B A2    00    90   02DF          Movb    #0,HP$B_DRIVE(R2)               ; Drive #                     ;G:3
              14 A8    00    B0   02E3          MOVW    #0,HPE$W_EXT_DRIVE(R8)          ; Drive #                     ;G:3
                 62    05    D0   02E7          MovL    #5, Hp$Q_Device(R2)             ; Set length of descriptor
              50   0C A2    9E   02EA          MovAB   Hp$T_Device(R2), R0             ; Set address of "_CSA0"
              04 A2    60    9E   02EE          MovAB   (R0), Hp$Q_Device+4(R2)         ; Set address of string
        80   4153435F 8F    D0   02F2          MovL    #^A"_CSA", (R0)+                ; Set first 4 chars of name
                 60    30    90   02F9          MovB    #^A"0", (R0)                    ; Set last character
              50   26 A2    9E   02FC          MovAB   Hp$T_Type(R2), R0               ; Set address of type string
        80   6E6F6307 8F    D0   0300          MovL    #7!<<^A"con">@8>, (R0)+         ; Set length/"con"
        60   656C6F73 8F    D0   0307          MovL    #^A"sole", (R0)                 ; Set rest of name
        00000000'EF    62    FA   030E          CallG   (R2), L^Ins$Ptable              ; Insert the Ptable in list
                    09 50    E8   0315          BlbS    R0, 10$                         ; Continue if it worked
                 50    52    D0   0318          MovL    R2, R0                          ; Else, copy length to R0
             00000000'EF    16   031B          Jsb     L^Exe$DeaNonPaged               ; .. and deallocate the buffer
                                  0321  10$:
```

```
                    05  0321    822         RSB                                    ; RETURN
```

```
                     0322   824          .SBTTL  ONLY_CPU        CPU specific setup                    [23]  ;G:2
                     0322   825 ;++                                                                            ;G:3
                     0322   826 ; FUNCTIONAL DESCRIPTION:                                                     ;G:2
                     0322   827 ;                                                                             ;G:2
                     0322   828 ;       Does nothing for the case of NEBULA.                                  ;G:2
                     0322   829 ;                                                                             ;G:2
                     0322   830 ;--                                                                           ;G:3
                     0322   831                                                                             ;G:2
                     0322   832 ONLY_CPU::                          ;                                      [23]  ;G:2
                  05 0322   833          RSB                       ;                                      [23]  ;G:2
                     0323   834          .END
```

```
$$$                             = 00003C00 R   D  04
$ER                             = 00000002     D
$MODULE                           00000000 R   D  03
ADP$B_NUMBER                      0000000B     D
ADP$B_PORT                        00000020     D
ADP$B_TYPE                        0000000A     D
ADP$C_DRADPLEN                    00000014     D
ADP$C_MBAADPLEN                   00000014     D
ADP$C_MPMADPLEN                   00000070     D
ADP$C_UBAADPLEN                   00000070     D
ADP$K_DRADPLEN                    00000014     D
ADP$K_MBAADPLEN                   00000014     D
ADP$K_MPMADPLEN                   00000070     D
ADP$K_UBAADPLEN                   00000070     D
ADP$L_CRB                         00000010     D
ADP$L_CSR                         00000000     D
ADP$L_DPQBL                       00000018     D
ADP$L_DPQFL                       00000014     D
ADP$L_INTD                        00000064     D
ADP$L_LINK                        00000004     D
ADP$L_MRQBL                       00000020     D
ADP$L_MRQFL                       0000001C     D
ADP$L_PRQQBL                      00000018     D
ADP$L_PRQQFL                      00000014     D
ADP$L_SHB                         0000001C     D
ADP$L_VECTOR                      00000010     D
ADP$W_ADPTYPE                     0000000E     D
ADP$W_DPBITMAP                    00000024     D
ADP$W_MRBITMAP                    00000026     D
ADP$W_SIZE                        00000008     D
ADP$W_TR                          0000000C     D
ARB$C_LENGTH                    = 00000078     D
AX_BUFF                           00000000 RG  D  04
BIT...                          = 00660160     D
BUG$CHECK                         ********   X  00
CMK$RCONSOLE                      ********   X  00
CMK$TCONSOLE                      ********   X  00
COMMON_SCB                        000002A4 R   D  05
CONFIG_EXIT                       000000CC R   D  05
CONFIG_RELEASE                    000000BF R   D  05
CRB$B_MASK                        0000000E     D
CRB$B_TYPE                        0000000A     D
CRB$C_LENGTH                      00000038     D
CRB$K_LENGTH                      00000038     D
CRB$L_INTD                        00000014     D
CRB$L_INTD2                       00000038     D
CRB$L_LINK                        00000010     D
CRB$L_WQBL                        00000004     D
CRB$L_WQFL                        00000000     D
CRB$W_REFC                        0000000C     D
CRB$W_SIZE                        00000008     D
DEF$Q_DEV                         00000027 RG  D  02
DEF$Q_DIR                         0000004D RG  D  02
DIR...                          = FFFFFFFF     D
DS$AX_ARB                       = 00000274 RG  D  04
DS$AX_JIB                       = 00000200 RG  D  04
DS$AX_SOFTPCB                   = 000002EC RG  D  04

DS$GB_PRIMARY_CPU_IDENTIFIER      00000010 RG  D  03
DS$GK_SID                       = 03000000   G  D
DS$GK_USOVR                     = 00000200   G  D
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$WAITUS                         ********   X  00
DS$_AP_NORMAL_BREAK             = 00660150     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INIT_FAIL                   = 00660140     D
DS$_INSFMEM                     = 00660050     D
DS$_INVCPU                      = 00660130     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MEM_ALLOC_ERR               = 00660138     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTALLOWMP                  = 00660148     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_UNSUPPDEV                   = 00660158     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
```

```
DSP$CONFIG_ADP                  00000000 RG D  05
DSR$ONLY_CONSOLE_CONFIG         000002AB RG D  05
DS_ERRSUP                       ********   X   00
DW730$K_LEN                     00000032     D
ESTKPTR                       = 00003000 RG D  04
EXE$ALONONPAGED                 ********   X   00
EXE$DEANONPAGED                 ********   X   00
EXE_UNEXPINT                    ********   X   00
FETCH_LONG                      00000152 RG D  05
FETCH_STORE                     ********   X   00
HANDLR                          000000CD R   D  05
HDW_CLOCK_SET                   00000223 RG D  05
HP$A_DEPENDENT                  00000032     D
HP$A_DEVICE                     00000018     D
HP$A_DVA                        0000001C     D
HP$A_LINK                       00000020     D
HP$B_DRIVE                      0000000B     D
HP$B_FLAGS                      0000000A     D
HP$Q_DEVICE                     00000000     D
HP$T_DEVICE                     0000000C     D
HP$T_TYPE                       00000026     D
HP$W_SIZE                       00000008     D
HP$W_VECTOR                     00000024     D
HPE$A_EPB                       00000016     D
HPE$C_EPB_SIZE                  ********   X   00
HPE$T_DEVICE                    00000000     D
HPE$W_EXT_DRIVE                 00000014     D
INS$PTABLE                      ********   X   00
IO$GQ_PHYSICAL                  00000008 RG D  03
IO$TESTCSR_L                    000001A9 RG D  05
IO$TESTCSR_W                    000001C8 RG D  05
IOC$GL_ADPLIST                  ********   X   00
IOC$K_IOSPACE                 = 40000000   G D
IOC$PURGDATAP                   00000186 RG D  05
IOGEN$CONN_VEC                  000001E7 RG D  05
IS                            = 00000001     D
ISTKPTR                       = 00002A00 RG D  04
JIB$C_LENGTH                  = 00000074     D
KSTKPTR                       = 00002200 RG D  04
LOCAL_STORE                     00000000 R   D  02
L_CONFIG                        FFFFFFF4     D
L_IPL                           FFFFFFF8     D
L_LOCAL                         FFFFFFF4     D
L_VECTOR                        FFFFFFFC     D
MAP$IOSPACE                     000000FE RG D  05
MAPMEM$IOSPACE                  ********   X   00
ONLY$GL_TENUSEC                 00000011 RG D  03
ONLY$GL_UBDELAY                 00000015 RG D  03
ONLY_ATTACH                     00000224 RG D  05
ONLY_CPU                        00000322 RG D  05
ONLY_SCB                        00000222 RG D  05
POPT_BASE                     = 00003C00 RG D  04
PCB_BASE                      = 00000078 RG D  04
PHD$L_PCB                     = 00000078     D
PHD_BASE                      = 00000000 RG D  04
PR$_IPL                       = 00000012     D
PTE$C_UW                      = 20000000     D
PTE$M_IO                      = 00100000     D
PTE$M_VALID                   = 80000000     D
PTE$V_IO                      = 00000014     D
QIO$DEALLOCATE                  ********   X   00
QIOCLEAN_CPU                    000001F2 RG D  05
SAVABS...                     = FFFFFFF4     D
SCBVECTOR_000                 = 00000229 RG D  05
SCBVECTOR_038                 = 0000022D RG D  05
SCBVECTOR_03C                 = 00000231 RG D  05
SCBVECTOR_050                 = 00000235 RG D  05
SCBVECTOR_054                 = 00000239 RG D  05
SCBVECTOR_058                 = 0000023D RG D  05
SCBVECTOR_05C                 = 00000241 RG D  05
SCBVECTOR_060                 = 00000245 RG D  05
SCBVECTOR_064                 = 00000249 RG D  05
SCBVECTOR_068                 = 0000024D RG D  05
SCBVECTOR_06C                 = 00000251 RG D  05
SCBVECTOR_070                 = 00000255 RG D  05
SCBVECTOR_074                 = 00000259 RG D  05
SCBVECTOR_078                 = 0000025D RG D  05
SCBVECTOR_07C                 = 00000261 RG D  05
SCBVECTOR_080                 = 00000265 RG D  05
SCBVECTOR_0C4                 = 00000269 RG D  05
SCBVECTOR_0C8                 = 0000026D RG D  05
SCBVECTOR_0CC                 = 00000271 RG D  05
SCBVECTOR_0D0                 = 00000275 RG D  05
SCBVECTOR_0D4                 = 00000279 RG D  05
SCBVECTOR_0D8                 = 0000027D RG D  05
SCBVECTOR_0DC                 = 00000281 RG D  05
SCBVECTOR_0E0                 = 00000285 RG D  05
SCBVECTOR_0E4                 = 00000289 RG D  05
SCBVECTOR_0E8                 = 0000028D RG D  05
SCBVECTOR_0EC                 = 00000291 RG D  05
SCBVECTOR_0F0                 = 00000295 RG D  05
SCBVECTOR_0F4                 = 00000299 RG D  05
SCBVECTOR_0F8                 = 0000029D RG D  05
SCBVECTOR_0FC                 = 000002A1 RG D  05
SCB_BASE                      = 00000400 RG D  04
SCB_IMAGE                       ********   X   00
SCB_UNKINT                    = 00000800 RG D  04
SIZ...                        = 00000001     D
SS$_CONTINUE                    ********   X   00
SS$_CONTROLC                    ********   X   00
SS$_CTRLERR                     ********   X   00
SS$_DATACHECK                   ********   X   00
SS$_NORMAL                    = 00000001     D
SS$_RESIGNAL                    ********   X   00
SSTKPTR                       = 00003600 RG D  04
STACK_BASE                    = 00000C00 RG D  04
SYS$DISK                        00000017 RG D  02
SYS$SETPRT                      ********   X   00
SYS$SYSROOT                     00000004 RG D  02
SYS$UNWIND                      ********   X   00
TST$MCHK                        ********   X   00
UBA$INITIAL                     000001A6 RG D  05
UBA$UNEXINT                     000001A8 RG D  05
UBADP_CPU                       00000199 RG D  05
```

UBINTSZ	= 000000B4	G	D	UCB$L_FR4	00000014		D
UBINT_DISP	********	X	00	UCB$L_IOQBL	00000044		D
UCB$B_AMOD	00000053		D	UCB$L_IOQFL	00000040		D
UCB$B_CEX	00000077		D	UCB$L_IRP	0000004C		D
UCB$B_CM1	0000004A		D	UCB$L_LINK	0000002C		D
UCB$B_CM2	0000004B		D	UCB$L_LOGADR	00000064		D
UCB$B_DEVCLASS	00000038		D	UCB$L_MAXBLOCK	00000084		D
UCB$B_DEVTYPE	00000039		D	UCB$L_MB_MBX	0000007C		D
UCB$B_DIPL	00000052		D	UCB$L_MB_PORT	0000008C		D
UCB$B_DX_SCTCNT	000000A6		D	UCB$L_MB_RAST	00000078		D
UCB$B_ERTCNT	00000070		D	UCB$L_MB_SHB	00000080		D
UCB$B_ERTMAX	00000071		D	UCB$L_MB_WAST	00000074		D
UCB$B_FEX	00000076		D	UCB$L_MB_WIOQBL	00000088		D
UCB$B_FIPL	0000000B		D	UCB$L_MB_WIOQFL	00000084		D
UCB$B_LOCSRV	0000003C		D	UCB$L_MEDIA	0000008C		D
UCB$B_OFFNDX	00000094		D	UCB$L_NT_DATSSB	00000074		D
UCB$B_OFFRTC	00000095		D	UCB$L_NT_INTSSB	00000078		D
UCB$B_REMSRV	0000003D		D	UCB$L_OPCNT	00000060		D
UCB$B_SECTORS	0000003C		D	UCB$L_OWNUIC	0000001C		D
UCB$B_SLAVE	00000074		D	UCB$L_PID	00000028		D
UCB$B_SPR	00000075		D	UCB$L_RQBL	00000004		D
UCB$B_STATE	00000052		D	UCB$L_RQFL	00000000		D
UCB$B_TRACKS	0000003D		D	UCB$L_SVAPTE	00000068		D
UCB$B_TT_CRFILL	0000009D		D	UCB$L_SVPN	00000064		D
UCB$B_TT_DECRF	000000A1		D	UCB$L_TT_DECHAR	000000A8		D
UCB$B_TT_DELFF	000000A2		D	UCB$L_TT_RDUE	0000008C		D
UCB$B_TT_DESPEE	000000A0		D	UCB$L_TT_RTIMOU	000000B8		D
UCB$B_TT_DETYPE	000000A4		D	UCB$L_VCB	00000030		D
UCB$B_TT_LFFILL	0000009E		D	UCB$T_PARTNER	0000000C		D
UCB$B_TT_SPEED	0000009C		D	UCB$W_BCNT	0000006E		D
UCB$B_TYPE	0000000A		D	UCB$W_BCR	00000096		D
UCB$B_VERTSZ	0000003F		D	UCB$W_BOFF	0000006C		D
UCB$C_LENGTH	00000074		D	UCB$W_BUFQUO	00000018		D
UCB$C_MB_LENGTH	00000090		D	UCB$W_BYTESTOGO	0000003E		D
UCB$C_TT_LENGTH	000000BC		D	UCB$W_CHARGE	0000004A		D
UCB$K_LENGTH	00000074		D	UCB$W_CYLINDERS	0000003E		D
UCB$K_MB_LENGTH	00000090		D	UCB$W_DA	0000008C		D
UCB$K_TT_LENGTH	000000BC		D	UCB$W_DC	0000008E		D
UCB$L_AMB	00000054		D	UCB$W_DEVBUFSIZ	0000003A		D
UCB$L_ASTQBL	00000010		D	UCB$W_DEVSTS	0000005A		D
UCB$L_ASTQFL	0000000C		D	UCB$W_DIRSEQ	00000088		D
UCB$L_CPID	0000005C		D	UCB$W_DSTADDR	00000018		D
UCB$L_CRB	00000020		D	UCB$W_DX_BCR	000000A4		D
UCB$L_DDB	00000024		D	UCB$W_EC1	00000090		D
UCB$L_DEVCHAR	00000034		D	UCB$W_EC2	00000092		D
UCB$L_DEVDEPEND	0000003C		D	UCB$W_ERRCNT	00000072		D
UCB$L_DPC	00000080		D	UCB$W_FUNC	0000007E		D
UCB$L_DUETIM	0000005C		D	UCB$W_MB_SEED	00000000		D
UCB$L_DX_BFPNT	0000009C		D	UCB$W_MSGCNT	00000016		D
UCB$L_DX_BUF	00000098		D	UCB$W_MSGMAX	00000014		D
UCB$L_DX_RXDB	000000A0		D	UCB$W_NT_CHAN	0000007C		D
UCB$L_EMB	00000078		D	UCB$W_OFFSET	0000008A		D
UCB$L_FIRST	00000014		D	UCB$W_REFC	00000050		D
UCB$L_FPC	0000000C		D	UCB$W_SIZE	00000008		D
UCB$L_FQBL	00000004		D	UCB$W_SRCADDR	0000001A		D
UCB$L_FQFL	00000000		D	UCB$W_STS	00000058		D
UCB$L_FR3	00000010		D	UCB$W_TT_DESIZE	000000A5		D

```
UCB$W_UNIT                      00000048    D
UCB$W_VPROT                     0000001A    D
USTKPTR                       = 00003C00 RG D  04
VEC$B_DATAPATH                  00000013    D
VEC$B_NUMREG                    00000012    D
VEC$C_LENGTH                    00000024    D
VEC$K_LENGTH                    00000024    D
VEC$L_ADP                       00000014    D
VEC$L_IDB                       00000008    D
VEC$L_INITIAL                   0000000C    D
VEC$L_START                     0000001C    D
VEC$L_UNITDISC                  00000020    D
VEC$L_UNITINIT                  00000018    D
VEC$Q_DISPATCH                  00000000    D
VEC$W_MAPREG                    00000010    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	000000D0 (208.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000019 (25.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
_LAST	00000000 (0.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
CODE	00000323 (803.)	05 (5.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00003C00-R	286 (1)	228 (1) 229 (1) 230 (1) 236 (1)
			237 (1) 243 (1) 249 (1) 255 (1)
			256 (1) 257 (1) 258 (1) 271 (1)
			272 (1) 274 (1) 275 (1) 277 (1)
			278 (1) 280 (1) 281 (1) 283 (1)
			284 (1) 286 (1) 287 (1) 289 (1)
$ER	=00000002	357 (3)	#-357 (3)
$MODULE	00000000-R	201 (1)	357 (3)
ADP$L_LINK	00000004		#-674 (13)
ADP$L_VECTOR	00000010		#-549 (9)
ADP$W_DPBITMAP	00000024		#-550 (9)
ARB$C_LENGTH	=00000078		243 (1)
AX_BUFF	00000000-R	222 (1)	
BIT...	=00660160	150 (1)	143 (1) 150 (1)
BUG$CHECK	00000000-XR		167 (1)
CMK$RCONSOLE	00000000-XR		167 (1)
CMK$TCONSOLE	00000000-XR		167 (1)
COMMON_SCB	000002A4-R	762 (17)	#-731 (17) #-732 (17) #-733 (17) #-734 (17)
			#-735 (17) #-736 (17) #-737 (17) #-738 (17)
			#-739 (17) #-740 (17) #-741 (17) #-742 (17)
			#-743 (17) #-744 (17) #-745 (17) #-746 (17)
			#-747 (17) #-748 (17) #-749 (17) #-750 (17)
			#-751 (17) #-752 (17) #-753 (17) #-754 (17)
			#-755 (17) #-756 (17) #-757 (17) #-758 (17)
			#-759 (17) #-760 (17) #-761 (17)
CONFIG_EXIT	000000CC-R	396 (3)	#-390 (3)
CONFIG_RELEASE	000000BF-R	392 (3)	
CRB$L_INTD	00000014		#-538 (8)
DEF$Q_DEV	00000027-R	191 (1)	
DEF$Q_DIR	0000004D-R	194 (1)	
DIR...	=FFFFFFFF	338 (2)	338 (2)
DS$AX_ARB	=00000274-R	237 (1)	
DS$AX_JIB	=00000200-R	236 (1)	
DS$AX_SOFTPCB	=000002EC-R	243 (1)	
DS$GB_PRIMARY_CPU_IDENTIFIER	00000010-R	208 (1)	#-444 (6)
DS$GK_SID	=03000000	165 (1)	
DS$GK_USOVR	=00000200	159 (1)	
DS$K_ERROR	=00000002	150 (1)	
DS$K_NORMAL	=00000001	150 (1)	
DS$K_SEVERE	=00000004	150 (1)	
DS$K_SUBSYS	=00000066	150 (1)	150 (1)
DS$K_WARNING	=00000000	150 (1)	
DS$WAITUS	00000000-XR		167 (1)
DS$_AP_NORMAL_BREAK	=00660150	150 (1)	
DS$_ARITH	=006600D0	150 (1)	
DS$_ASBE	=00660118	150 (1)	
DS$_BADLINK	=006600F0	150 (1)	
DS$_BADTYPE	=006600E8	150 (1)	
DS$_BIIC	=00660120	150 (1)	
DS$_CHME	=006600A8	150 (1)	

```
DS$_CHMK                        =006600E0      150     (1)
DS$_DEVNAME                     =00660108      150     (1)
DS$_ERROR                       =00660002      150     (1)     #-358    (3)     #-395    (3)
DS$_FHWE                        =00660068      150     (1)
DS$_FRAGBUF                     =00660080      150     (1)
DS$_ICBUSY                      =006600C8      150     (1)
DS$_ICERR                       =006600C0      150     (1)
DS$_IHWE                        =00660060      150     (1)
DS$_ILLCHAR                     =00660018      150     (1)
DS$_ILLPAGCNT                   =00660078      150     (1)
DS$_ILLUNIT                     =00660100      150     (1)
DS$_INIT_FAIL                   =00660140      150     (1)
DS$_INSFMEM                     =00660050      150     (1)
DS$_INVCPU                      =00660130      150     (1)
DS$_IPL2HI                      =006600B8      150     (1)
DS$_IVADDR                      =00660040      150     (1)
DS$_IVVECT                      =00660038      150     (1)
DS$_KRNLSTK                     =00660090      150     (1)
DS$_LOGIC                       =00660070      150     (1)
DS$_MCHK                        =00660088      150     (1)     #-501    (7)
DS$_MEM_ALLOC_ERR               =00660138      150     (1)
DS$_MMOFF                       =00660058      150     (1)
DS$_NEEDUNIT                    =006600F8      150     (1)
DS$_NODE                        =00660128      150     (1)
DS$_NOPCS                       =00660110      150     (1)
DS$_NORMAL                      =00660001      150     (1)
DS$_NOSUPPORT                   =006600B1      150     (1)
DS$_NOTALLOWMP                  =00660148      150     (1)
DS$_NOTDON                      =00660030      150     (1)
DS$_NOTIMP                      =006600B0      150     (1)       150    (1)
DS$_NULLSTR                     =00660010      150     (1)
DS$_OVERFLOW                    =00660008      150     (1)
DS$_POWER                       =00660098      150     (1)
DS$_PROGERR                     =00660020      150     (1)
DS$_SEVERE                      =00660004      150     (1)
DS$_TRANSL                      =006600A0      150     (1)
DS$_TRUNCATE                    =00660028      150     (1)
DS$_UNEXPINT                    =006600D8      150     (1)     #-404    (4)
DS$_UNSUPPDEV                   =00660158      150     (1)
DS$_VASFULL                     =00660048      150     (1)
DS$_WARNING                     =00660000      150     (1)       150    (1)
DSP$CONFIG_ADP                   00000000-R    344     (3)
DSR$ONLY_CONSOLE_CONFIG          000002AB-R    792     (17)
DS_ERRSUP                        00000000-XR                     168    (1)       357    (3)
DW730$K_LEN                      00000032                      #-362    (3)
ESTKPTR                         =00003000-R    281     (1)
EXE$ALONONPAGED                  00000000-XR                     170    (1)     #-364    (3)       797    (17)
EXE$DEANONPAGED                  00000000-XR                     170    (1)     #-394    (3)       820    (17)
EXE_UNEXPINT                     00000000-XR                     169    (1)     #-764    (17)
FETCH_LONG                       00000152-R    492     (7)       448    (6)
FETCH_STORE                      00000000-XR                     170    (1)
HANDLR                           000000CD-R    399     (4)
HDW_CLOCK_SET                    00000223-R    700     (15)
HP$A_DEPENDENT                   00000032                      #-795    (17)
HP$A_DEVICE                      00000018                      #-385    (3)
HP$A_DVA                         0000001C                      #-369    (3)
HP$A_LINK                        00000020                      #-387    (3)
```

```
HP$B_DRIVE                      0000000B                        #-375    (3)     #-807    (17)
HP$Q_DEVICE                     00000000                        #-382    (3)     #-384    (3)     #-809    (17)    #-811    (17)
HP$T_DEVICE                     0000000C                        #-377    (3)       383    (3)       810    (17)
HP$T_TYPE                       00000026                        #-378    (3)     #-379    (3)       814    (17)
HP$W_SIZE                       00000008                        #-370    (3)     #-806    (17)
HP$W_VECTOR                     00000024                        #-380    (3)
HPE$C_EPB_SIZE                  00000000-XR                       179    (1)     #-363    (3)     #-373    (3)     #-796    (17)
                                                                #-804    (17)
HPE$W_EXT_DRIVE                 00000014                        #-376    (3)     #-808    (17)
INS$PTABLE                      00000000-XR                       171    (1)       389    (3)       817    (17)
IO$GQ_PHYSICAL                  00000008-R      202     (1)
IO$TESTCSR_L                    000001A9-R      608     (11)
IO$TESTCSR_W                    000001C8-R      620     (11)
IOC$GL_ADPLIST                  00000000-XR                       171    (1)       670    (13)
IOC$K_IOSPACE                   =40000000       164     (1)     #-352    (3)
IOC$PURGDATAP                    00000186-R     536     (8)
IOGEN$CONN_VEC                   000001E7-R     654     (12)
IS                              =00000001       154     (1)       731    (17)      732    (17)      733    (17)      734    (17)
                                                                  735    (17)      736    (17)      737    (17)      738    (17)
                                                                  739    (17)      740    (17)      741    (17)      742    (17)
                                                                  743    (17)      744    (17)      745    (17)      746    (17)
                                                                  747    (17)      748    (17)      749    (17)      750    (17)
                                                                  751    (17)      752    (17)      753    (17)      754    (17)
                                                                  755    (17)      756    (17)      757    (17)      758    (17)
                                                                  759    (17)      760    (17)      761    (17)
ISTKPTR                         =00002A00-R     278     (1)
JIB$C_LENGTH                    =00000074                         237    (1)       243    (1)
KSTKPTR                         =00002200-R     275     (1)
LOCAL_STORE                      00000000-R     183     (1)     #-348    (3)     #-400    (4)
L_CONFIG                         FFFFFFF4       338     (2)     #-353    (3)     #-408    (4)
L_IPL                            FFFFFFF8       338     (2)     #-350    (3)     #-407    (4)
L_LOCAL                          FFFFFFF4       338     (2)       347    (3)
L_VECTOR                         FFFFFFFC       338     (2)     #-349    (3)     #-406    (4)
MAP$IOSPACE                      000000FE-R     442     (6)
MAPMEM$IOSPACE                  00000000-XR                       172    (1)     #-455    (6)     #-462    (6)
ONLY$GL_TENUSEC                 00000011-R      213     (1)
ONLY$GL_UBDELAY                 00000015-R      215     (1)
ONLY_ATTACH                     00000224-R      711     (16)
ONLY_CPU                        00000322-R      832     (18)
ONLY_SCB                        00000222-R      689     (14)
POPT_BASE                       =00003C00-R     289     (1)
PCB_BASE                        =00000078-R     229     (1)
PHD$L_PCB                       =00000078                         229    (1)
PHD_BASE                        =00000000-R     228     (1)
PR$_IPL                         =00000012                       #-407    (4)
PTE$C_UW                        =20000000                       #-453    (6)     #-460    (6)
PTE$M_IO                        =00100000       143     (1)
PTE$M_VALID                     =80000000                       #-453    (6)     #-460    (6)
PTE$V_IO                        =00000014       143     (1)
QIO$DEALLOCATE                   00000000-XR                      173    (1)     #-675    (13)
QIOCLEAN_CPU                     000001F2-R     665     (13)
SAVABS...                       =FFFFFFF4       338     (2)
SCBVECTOR_000                   =00000229-R     731     (17)
SCBVECTOR_038                   =0000022D-R     732     (17)
SCBVECTOR_03C                   =00000231-R     733     (17)
SCBVECTOR_050                   =00000235-R     734     (17)
SCBVECTOR_054                   =00000239-R     735     (17)
```

```
SCBVECTOR_058                   =0000023D-R   736     (17)
SCBVECTOR_05C                   =00000241-R   737     (17)
SCBVECTOR_060                   =00000245-R   738     (17)
SCBVECTOR_064                   =00000249-R   739     (17)
SCBVECTOR_068                   =0000024D-R   740     (17)
SCBVECTOR_06C                   =00000251-R   741     (17)
SCBVECTOR_070                   =00000255-R   742     (17)
SCBVECTOR_074                   =00000259-R   743     (17)
SCBVECTOR_078                   =0000025D-R   744     (17)
SCBVECTOR_07C                   =00000261-R   745     (17)
SCBVECTOR_080                   =00000265-R   746     (17)
SCBVECTOR_0C4                   =00000269-R   747     (17)
SCBVECTOR_0C8                   =0000026D-R   748     (17)
SCBVECTOR_0CC                   =00000271-R   749     (17)
SCBVECTOR_0D0                   =00000275-R   750     (17)
SCBVECTOR_0D4                   =00000279-R   751     (17)
SCBVECTOR_0D8                   =0000027D-R   752     (17)
SCBVECTOR_0DC                   =00000281-R   753     (17)
SCBVECTOR_0E0                   =00000285-R   754     (17)
SCBVECTOR_0E4                   =00000289-R   755     (17)
SCBVECTOR_0E8                   =0000028D-R   756     (17)
SCBVECTOR_0EC                   =00000291-R   757     (17)
SCBVECTOR_0F0                   =00000295-R   758     (17)
SCBVECTOR_0F4                   =00000299-R   759     (17)
SCBVECTOR_0F8                   =0000029D-R   760     (17)
SCBVECTOR_0FC                   =000002A1-R   761     (17)
SCB_BASE                        =00000400-R   255     (1)     548     (9)     610     (11)     622     (11)
SCB_IMAGE                        00000000-XR                  174     (1)
SCB_UNKINT                      =00000800-R   257     (1)     668     (13)
SIZ...                          =00000001     143     (1)     143     (1)
SS$_CONTINUE                     00000000-XR                  175     (1)     #-409     (4)
SS$_CONTROLC                     00000000-XR                  175     (1)
SS$_CTRLERR                      00000000-XR                  175     (1)
SS$_DATACHECK                    00000000-XR                  175     (1)
SS$_NORMAL                      =00000001     153     (1)     #-496     (7)
SS$_RESIGNAL                     00000000-XR                  176     (1)     #-402     (4)     #-508     (7)
SSTKPTR                         =00003600-R   284     (1)
STACK_BASE                      =00000C00-R   271     (1)
SYS$DISK                         00000017-R   189     (1)
SYS$SETPRT                       00000000-XR                  177     (1)
SYS$SYSROOT                      00000004-R   187     (1)
SYS$UNWIND                       00000000-XR                  177     (1)     505     (7)
TST$MCHK                         00000000-XR                  178     (1)     612     (11)     624     (11)
UBA$INITIAL                      000001A6-R   573     (10)
UBA$UNEXINT                      000001A8-R   580     (10)
UBADP_CPU                        00000199-R   547     (9)
UBINTSZ                         =000000B4     553     (9)
UBINT_DISP                       00000000-XR                  179     (1)     #-655     (12)
UCB$L_CRB                        00000020                   #-537     (8)
USTKPTR                         =00003C00-R   287     (1)
VEC$L_ADP                        00000014                   #-538     (8)
VEC$Q_DISPATCH                   00000000                   #-655     (12)     656     (12)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ADPDEF	2	138 (1)	138 (1)
$ARBDEF	1	135 (1)	135 (1)
$CRBDEF	1	137 (1)	137 (1)
$DEF	1	150 (1)	
$DEFINI	1	133 (1)	133 (1) 134 (1) 135 (1) 136 (1) 137 (1)
			138 (1) 139 (1) 140 (1) 141 (1) 142 (1)
			144 (1) 145 (1) 146 (1) 147 (1) 148 (1)
			149 (1) 151 (1)
$DS_DSDEF	3	150 (1)	150 (1)
$DS_DW730_DEF	1	149 (1)	149 (1)
$DS_HPEODEF	1	148 (1)	148 (1)
$DS_HPODEF	2	147 (1)	147 (1)
$DYNDEF	2	140 (1)	140 (1)
$EQU	1	150 (1)	150 (1)
$EQULS1	1	150 (1)	150 (1)
$EQULST	1		150 (1)
$GBLINI	2		150 (1)
$JIBDEF	3	134 (1)	134 (1)
$OFFSET	2	334 (2)	334 (2)
$OFFST1	1	338 (2)	338 (2)
$PHDDEF	6	133 (1)	133 (1)
$PR730DEF	1	146 (1)	146 (1)
$PRDEF	5	145 (1)	145 (1)
$PRTDEF	1	144 (1)	144 (1)
$PSLDEF	2	141 (1)	141 (1)
$PTEDEF	3	142 (1)	142 (1)
$PUSHADR	1	357 (3)	357 (3)
$UBIDEF	2	151 (1)	151 (1)
$UCBDEF	10	136 (1)	136 (1)
$VECDEF	2	139 (1)	139 (1)
$VIELD	1		143 (1)
$VIELD1	1	150 (1)	143 (1)
CASE	1	356 (3)	356 (3)
ERRSUP_S	1	357 (3)	357 (3)
MODNAM	1	201 (1)	201 (1)
SPAREVECTOR	1	724 (17)	731 (17) 732 (17) 733 (17) 734 (17) 735 (17)
			736 (17) 737 (17) 738 (17) 739 (17) 740 (17)
			741 (17) 742 (17) 743 (17) 744 (17) 745 (17)
			746 (17) 747 (17) 748 (17) 749 (17) 750 (17)
			751 (17) 752 (17) 753 (17) 754 (17) 755 (17)
			756 (17) 757 (17) 758 (17) 759 (17) 760 (17)
			761 (17)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ACBL	00F1	457	(6)												
ADDL	00C0	366	(3)												
ADDL2	00C0	363	(3)	539	(8)	796	(17)								
ADDL3	00C1	371	(3)	802	(17)										
AOBLSS	00F2	669	(13)												
ASHL	0078	452	(6)												
BEQL	0013	673	(13)												
BISL3	00C9	352	(3)												
BLBC	00E9	450	(6)	798	(17)										
BLBS	00E8	365	(3)	390	(3)	818	(17)								
BNEQ	0012	405	(4)	502	(7)										
BRB	0011	676	(13)												
BRW	0031	764	(17)												
BSBB	0010	731	(17)	732	(17)	733	(17)	734	(17)	735	(17)	736	(17)	737	(17)
		738	(17)	739	(17)	740	(17)	741	(17)	742	(17)	743	(17)	744	(17)
		745	(17)	746	(17)	747	(17)	748	(17)	749	(17)	750	(17)	751	(17)
		752	(17)	753	(17)	754	(17)	755	(17)	756	(17)	757	(17)	758	(17)
		759	(17)	760	(17)	761	(17)								
BSBW	0030	364	(3)	394	(3)	455	(6)	462	(6)	675	(13)				
CALLG	00FA	389	(3)	817	(17)										
CALLS	00FB	357	(3)	448	(6)	505	(7)								
CASEW	00AF	356	(3)												
CLRB	0094	444	(6)												
CLRL	00D4	349	(3)	350	(3)	387	(3)								
CLRQ	007C	504	(7)												
CMPL	00D1	404	(4)	501	(7)										
EXTZV	00EF	355	(3)												
JSB	0016	797	(17)	820	(17)										
MCOML	00D2	616	(11)	628	(11)										
MFPR	00DB	407	(4)												
MOVAB	009E	347	(3)	383	(3)	494	(7)	656	(12)	810	(17)	811	(17)	814	(17)
MOVAL	00DE	401	(4)	548	(9)	610	(11)	612	(11)	622	(11)	624	(11)	668	(13)
		670	(13)												
MOVB	0090	375	(3)	807	(17)	813	(17)								
MOVC5	002C	800	(17)												
MOVL	00D0	348	(3)	353	(3)	354	(3)	358	(3)	361	(3)	362	(3)	369	(3)
		374	(3)	377	(3)	379	(3)	382	(3)	385	(3)	388	(3)	393	(3)
		395	(3)	400	(4)	406	(4)	408	(4)	409	(4)	445	(6)	453	(6)
		454	(6)	460	(6)	461	(6)	495	(7)	496	(7)	503	(7)	537	(8)
		538	(8)	672	(13)	674	(13)	805	(17)	809	(17)	812	(17)	815	(17)
		816	(17)	819	(17)										
MOVPSL	00DC	615	(11)	627	(11)										
MOVQ	007D	500	(7)	540	(8)										
MOVW	00B0	370	(3)	376	(3)	378	(3)	380	(3)	550	(9)	655	(12)	806	(17)
		808	(17)												
MOVZWL	003C	402	(4)	459	(6)	508	(7)	667	(13)	763	(17)	795	(17)		
POPL	8ED0	614	(11)	626	(11)										
POPR	00BA	449	(6)	464	(6)	617	(11)	629	(11)	678	(13)	801	(17)		
PUSHAL	00DF	357	(3)	446	(6)										
PUSHL	00DD	357	(3)	611	(11)	623	(11)								

PUSHR	00BB	443	(6)	447	(6)	609	(11)	621	(11)	666	(13)	799	(17)			
REI	0002	581	(10)													
RET	0004	359	(3)	367	(3)	397	(3)	411	(4)	497	(7)	506	(7)	509	(7)	
RSB	0005	465	(6)	541	(8)	551	(9)	574	(10)	618	(11)	630	(11)	657	(12)	
		679	(13)	690	(14)	701	(15)	712	(16)	822	(17)	833	(18)			
SUBL3	00C3	372	(3)	373	(3)	803	(17)	804	(17)							
TSTL	00D5	613	(11)													
TSTW	00B5	625	(11)													

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ALIGN	198	(1)	579	(10)	731	(17)	732	(17)	733	(17)	734	(17)	735	(17)	736	(17)
	737	(17)	738	(17)	739	(17)	740	(17)	741	(17)	742	(17)	743	(17)	744	(17)
	745	(17)	746	(17)	747	(17)	748	(17)	749	(17)	750	(17)	751	(17)	752	(17)
	753	(17)	754	(17)	755	(17)	756	(17)	757	(17)	758	(17)	759	(17)	760	(17)
	761	(17)														
.ASCIC	201	(1)														
.ASCID	188	(1)	190	(1)	195	(1)										
.BLKB	193	(1)	196	(1)	338	(2)										
.BYTE	208	(1)														
.DSABL	10	(1)														
.END	834	(18)														
.ENDC	143	(1)	150	(1)	338	(2)	357	(3)								
.ENDM	133	(1)	134	(1)	135	(1)	136	(1)	137	(1)	138	(1)	139	(1)	140	(1)
	141	(1)	142	(1)	144	(1)	145	(1)	146	(1)	147	(1)	148	(1)	149	(1)
	150	(1)	151	(1)	201	(1)	334	(2)	338	(2)	356	(3)	357	(3)	729	(17)
.ENDR	143	(1)	150	(1)	338	(2)	356	(3)								
.ENTRY	344	(3)	492	(7)												
.EXTRN	167	(1)	168	(1)	169	(1)	170	(1)	171	(1)	172	(1)	173	(1)	174	(1)
	175	(1)	176	(1)	177	(1)	178	(1)	179	(1)						
.IDENT	7	(1)														
.IF	143	(1)	150	(1)	338	(2)	357	(3)								
.IFF	150	(1)	338	(2)	357	(3)										
.IIF	143	(1)	150	(1)	357	(3)										
.IRP	143	(1)	150	(1)	338	(2)	356	(3)								
.LIBRARY	128	(1)	129	(1)	130	(1)										
.LIST	338	(2)	340	(2)	722	(17)	8	(1)								
.LONG	184	(1)	192	(1)	203	(1)	214	(1)	216	(1)						
.MACRO	150	(1)	724	(17)												
.MDELETE	150	(1)														
.NLIST	132	(1)	333	(2)	338	(2)	9	(1)								
.NOCROSS	133	(1)	134	(1)	135	(1)	136	(1)	137	(1)	138	(1)	139	(1)	140	(1)
	141	(1)	142	(1)	144	(1)	145	(1)	146	(1)	147	(1)	148	(1)	149	(1)
	151	(1)														
.PAGE	125	(1)														
.PSECT	181	(1)	200	(1)	220	(1)	338	(2)	342	(3)						
.RESTORE	133	(1)	134	(1)	135	(1)	136	(1)	137	(1)	138	(1)	139	(1)	140	(1)
	141	(1)	142	(1)	144	(1)	145	(1)	146	(1)	147	(1)	148	(1)	149	(1)
	151	(1)	338	(2)												
.SAVE	133	(1)	134	(1)	135	(1)	136	(1)	137	(1)	138	(1)	139	(1)	140	(1)
	141	(1)	142	(1)	144	(1)	145	(1)	146	(1)	147	(1)	148	(1)	149	(1)
	151	(1)	338	(2)												
.SBTTL	219	(1)	292	(2)	414	(5)	467	(7)	512	(8)	555	(10)	583	(11)	632	(12)
	659	(13)	681	(14)	692	(15)	703	(16)	714	(17)	767	(17)	824	(18)		
.SIGNED_WORD	356	(3)														
.SUBTITLE	126	(1)														
.TITLE	6	(1)														
.WORD	399	(4)	499	(7)	731	(17)	732	(17)	733	(17)	734	(17)	735	(17)	736	(17)
	737	(17)	738	(17)	739	(17)	740	(17)	741	(17)	742	(17)	743	(17)	744	(17)
	745	(17)	746	(17)	747	(17)	748	(17)	749	(17)	750	(17)	751	(17)	752	(17)
	753	(17)	754	(17)	755	(17)	756	(17)	757	(17)	758	(17)	759	(17)	760	(17)

761 (17)

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	7	#-352 (3) #-361 (3) #-388 (3) 401 (4) #-495 (7) #-500 (7)
FP	6	347 (3) #-348 (3) #-349 (3) #-350 (3) #-353 (3) #-494 (7)
R0	48	#-354 (3) 355 (3) #-356 (3) #-358 (3) #-365 (3) #-390 (3)
		#-393 (3) #-395 (3) #-402 (4) #-408 (4) #-409 (4) #-450 (6)
		#-454 (6) #-461 (6) #-496 (7) #-500 (7) #-501 (7) #-503 (7)
		#-508 (7) #-540 (8) #-613 (11) #-615 (11) #-616 (11) #-625 (11)
		#-627 (11) #-628 (11) #-656 (12) #-666 (13) #-667 (13) 668 (13)
		#-669 (13) #-672 (13) #-674 (13) #-678 (13) #-798 (17) #-810 (17)
		811 (17) #-812 (17) #-813 (17) #-814 (17) #-815 (17) #-816 (17)
		#-818 (17) #-819 (17)
R1	31	#-362 (3) #-363 (3) #-370 (3) #-371 (3) #-401 (4) #-404 (4)
		#-406 (4) #-452 (6) #-459 (6) #-503 (7) #-609 (11) #-610 (11)
		#-611 (11) #-612 (11) #-614 (11) #-617 (11) #-621 (11) #-622 (11)
		#-623 (11) #-624 (11) #-626 (11) #-629 (11) #-666 (13) #-678 (13)
		#-795 (17) #-796 (17) #-799 (17) #-800 (17) #-801 (17) #-802 (17)
		#-806 (17)
R2	40	344 (3) #-369 (3) #-370 (3) #-371 (3) #-375 (3) #-377 (3)
		#-378 (3) #-379 (3) #-380 (3) #-382 (3) 383 (3) #-384 (3)
		#-385 (3) #-387 (3) #-388 (3) 389 (3) #-393 (3) 399 (4)
		#-400 (4) #-406 (4) #-407 (4) #-408 (4) #-453 (6) #-460 (6)
		#-538 (8) #-539 (8) #-549 (9) #-550 (9) #-799 (17) 800 (17)
		#-801 (17) #-802 (17) #-806 (17) #-807 (17) #-809 (17) 810 (17)
		#-811 (17) 814 (17) 817 (17) #-819 (17)
R3	11	344 (3) #-443 (6) #-449 (6) #-464 (6) #-537 (8) #-538 (8)
		#-670 (13) #-672 (13) #-674 (13) #-799 (17) #-801 (17)
R4	17	344 (3) #-352 (3) #-353 (3) #-354 (3) #-385 (3) #-443 (6)
		#-445 (6) #-447 (6) #-452 (6) #-457 (6) #-464 (6) #-655 (12)
		656 (12) #-666 (13) #-678 (13) #-799 (17) #-801 (17)
R5	6	#-537 (8) #-666 (13) #-668 (13) #-678 (13) #-799 (17) #-801 (17)
R7	7	344 (3) #-372 (3) #-373 (3) #-374 (3) #-803 (17) #-804 (17)
		#-805 (17)
R8	11	344 (3) #-371 (3) #-372 (3) #-373 (3) #-374 (3) #-376 (3)
		#-802 (17) #-803 (17) #-804 (17) #-805 (17) #-808 (17)
SP	7	#-347 (3) #-366 (3) 446 (6) #-504 (7) #-763 (17) 800 (17)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.04	00:00:00.28
Command processing	903	00:00:00.23	00:00:00.98
Pass 1	1007	00:00:03.57	00:00:07.01
Symbol table sort	0	00:00:00.25	00:00:00.24
Pass 2	55	00:00:00.87	00:00:02.12
Symbol table output	2	00:00:00.04	00:00:00.23
Psect synopsis output	2	00:00:00.02	00:00:00.01
Cross-reference output	6	00:00:00.28	00:00:00.46
Assembler run totals	2010	00:00:05.30	00:00:11.33

```
The working set limit was 2900 pages.
186545 bytes (365 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 936 non-local and 16 local symbols.
834 source lines were read in Pass 1, producing 77 object records in Pass 2.
144 pages of virtual memory were used to define 36 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	10
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	2
SYS$COMMON:[SYSLIB]LIB.MLB;2	5
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	29

```
1212 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:ONLY730.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W
```

Table of contents

(2)	118	DECLARATIONS	
(3)	188	DSX$GETDATA	NUMERIC DATA INPUT ROUTINE
(5)	281	DSX$GETVIELD	BIT FIELD INPUT ROUTINE
(7)	366	DSX$GETADDRESS	ADDRESS INPUT ROUTINE
(9)	444	DSX$GETLOGICAL	LOGICAL RESPONSE ROUTINE
(11)	565	DSX$GETSTRING	ASCII STRING INPUT ROUTINE
(13)	751	RESPONSE	PARAMETER RETRIEVAL SUBROUTINE

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element PARAM.MAR
                        0000     2 ; *4    19-OCT-1987 08:53:13 MEIER "RECOMPILE"
                        0000     3 ; *3    25-JUN-1987 15:40:05 MEIER "CHANGES TO DS"
                        0000     4 ; *2    25-JUN-1987 12:00:25 MEIER "FIXED BR_IF_MP, A BEQL WAS MISSING."
                        0000     5 ; *1     6-FEB-1986 15:21:28 DS ""
                        0000     6 ; DEC/CMS REPLACEMENT HISTORY, Element PARAM.MAR
                        0000     7         .TITLE  PARAM   *** PARAM Parameter input routines
                        0000     8         .IDENT  /9.1-17/
                        0000     9         .NoShow Conditionals
                        0000    10
                        0000    11         .DSABL  GBL
                        0000    12
                        0000    13 ;
                        0000    14 ; COPYRIGHT  1977, 1980, 1981, 1982, 1985, 1986, 1987                    ;G:3
                        0000    15 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        0000    16 ;
                        0000    17 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        0000    18 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                        0000    19 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                        0000    20 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                        0000    21 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                        0000    22 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                        0000    23 ; REMAIN IN DEC.
                        0000    24 ;
                        0000    25 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                        0000    26 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                        0000    27 ; CORPORATION.
                        0000    28 ;
                        0000    29 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                        0000    30 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        0000    31
                        0000    32 ;++
                        0000    33 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
                        0000    34 ;
                        0000    35 ; ABSTRACT: This version of PARAM was created solely for
                        0000    36 ;           use in Scorpio and Nautilus to support multiprocessing.
                        0000    37 ;
                        0000    38 ; ENVIRONMENT:
                        0000    39 ;
                        0000    40 ; AUTHOR:       TOM SOUTTER     10-NOV-77       VERSION 01
                        0000    41 ;
                        0000    42 ; MODIFIED BY:
                        0000    43 ;                KEN CHAPMAN     03-FEB-78       VERSION 02 (ESSAA-3.07).
                        0000    44 ;       01       CHANGED UNIT NUMBER CHECK IN $GPHARD FROM SIGNED TO MAGNITUDE.
                        0000    45 ;                Roger Riggs     15-JUL-78
                        0000    46 ;       02       Rewrite to shorten and let GETLINE do the prompting
                        0000    47 ;                so that in script mode the prompts come out correctly.
                        0000    48 ;                Fix entry points to conform to naming conventions.
                        0000    49 ;                Also use .ENTRY ... ENTRY fixed to use .VECTOR
                        0000    50 ;                Also removed MOVC5 in GETSTRING
                        0000    51 ;                Roger Riggs     13-March-1978
                        0000    52 ;       03       Removed DS_GETPARAM, unnecessary since P-code was removed
                        0000    53 ;                Roger Riggs, 28-May-1980, Version 5.4
                        0000    54 ;       04       Fixed to correctly return 0 in R1 if default answer was
                        0000    55 ;                given to DS$ASKSTR. Also fixed size of field in ASKDATA
                        0000    56 ;                Roger RIggs, 19-Jun-1980, Version 5.5
                        0000    57 ;       05       Fixed default value when prompt was disabled
```

```
0000     58 ;              Roger Riggs, 16-Oct-1980
0000     59 ;        06    Fixed bug that failed to give default values on prompt
0000     60 ;              in ASKDATA, ASKVLD, ASKADR.
0000     61 ;
0000     62 ;        07    - Jack Stansbury, 22-Oct-1981, Version 6.-
0000     63 ;              Changed the .PSECT CODE statement on line 117 that was
0000     64 ;              commented out (for some strange reason) to .PSECT CODE.
0000     65 ;              Also took out the $DS_HDRDEF statement since none of the
0000     66 ;              diagnostic program header symbols are used here!
0000     67 ;              Also fixed some truncation errors.
0000     68 ;              Also changed some messages to mixed case.
0000     69 ;              Also added calls to the QA_MAIN macro and DSQA for QA.
0000     70 ;              Also added .LIBRARY statements for $DS and $DIAG.
0000     71 ;
0000     72 ;        08    - Jack Stansbury, 30-Nov-1981, Version 6.5
0000     73 ;              Added some more code for the QA routines.
0000     74 ;
0000     75 ;        09    - Jack Stansbury, 5-Dec-1981, Version 6.6
0000     76 ;              Changed the entry masks on each of the .ENTRY lines so
0000     77 ;              that ALL the registers are saved (rather than only those
0000     78 ;              used). This is so that the QA dump routine can find all
0000     79 ;              the registers in the call frame set up for these
0000     80 ;              ASK routines.
0000     81 ;
0000     82 ;        10    Marion Baggett, 31-Mar-1982, Version 6.7
0000     83 ;              Changed all $DS_PRINT type statements to $PRINT form.
0000     84 ;
0000     85 ;        11    Marion Baggett, 7-Apr-1982, Version 6.7
0000     86 ;              Added macro $DS_TYPEDEF definition.
0000     87 ;
0000     88 ;        12    Amy Iorio, 25-June-1985, Version 9.0
0000     89 ;              Added $DS> to ASK macro prompts.
0000     90 ;
0000     91 ;        13    Domenic Andella 26-June-1985    9.0
0000     92 ;              "Yes" transfer address in routine
0000     93 ;              DSX$GETLOGICAL must be altered to
0000     94 ;              load the saved FP's pc instead of
0000     95 ;              the currect FP's to compensate
0000     96 ;              for additional call frame generated
0000     97 ;              as a result of interlocking code.
0000     98 ;
0000     99 ;        14    Amy Iorio,      22-July-1985    9.0
0000    100 ;              Took $DS> change out.
0000    101 ;
0000    102 ;        15    Domenic Andella 7-August-1985   9.0
0000    103 ;              Comment out modification 13 until
0000    104 ;              interlocking code is reinstated.
0000    105 ;
0000    106 ;        16    Domenic Andella 30-August-1985  9.1
0000    107 ;              Reinstate edit 13.
0000    108 ;
0000    109 ;        17    Domenic Andella 17-Dec-1985     9.1
0000    110 ;              Modify routine DSX$GETLOGICAL such that if cpu
0000    111 ;              is a Scorpio or Nautilus, frame pointer will be
0000    112 ;              altered to accomodate interlocking.
0000    113 ;
0000    114 ;        18    Domenic Andella 31-Dec-1985     9.1
```

```
                        0000   115 ;                Add extrn symbol ref for DS$GK_SID
                        0000   116 ;--
```

```
                        0000   118          .SBTTL  DECLARATIONS
                        0000   119 ;
                        0000   120 ; INCLUDE FILES:
                        0000   121 ;
                        0000   122          .LIBRARY        /SYS$LIBRARY:LIB/                       ;               [10]
                        0000   123          .LIBRARY        /$DS/                                   ;               [07]
                        0000   124          .LIBRARY        /$DIAG/                                 ;               [07]
                        0000   125
                        0000   126 ;
                        0000   127 ;       ENTERNAL SYMBOLS
                        0000   128 ;
                        0000   129          .EXTRN  DS$ABORT,       DSX$PRINT                       ;               [10]
                        0000   130          .EXTRN  QA$AOB_Flags,   QA$Main                         ;               [08]
                        0000   131          .EXTRN  DS$K_STRBUF,    DS$GL_BUFLEN,   DS$GT_STRBUF
                        0000   132          .EXTRN  DS_GETLINE,     SYS$FAO,        SCAN$NUMERIC,   SCAN$SPACES
                        0000   133          .EXTRN  DS$GK_SID                                               ;[18]
                        0000   134 ;
                        0000   135 ; MACROS:
                        0000   136 ;
                        0000   137 ;
                        0000   138 ; EQUATED SYMBOLS:
                        0000   139 ;
                00000001 0000   140 SS$_NORMAL=1
                        0000   141          $DS_DSDEF                       ; DIAGNOSTIC SUPERVISOR ERROR CODES
                        0000   142          $DS_DSADEF                      ; CONTROL FLAG DEFINITIONS
                        0000   143          $DS_PARDEF                      ; PARAMETER ARG DEFINITIONS
                        0000   144          DSQA                            ; QA definitions                [07]
                        0000   145          DS_QADEFS                       ; More QA defs....              [08]
                        0000   146          $DS_TYPEDEF                     ; DEFINE TYPE CODES             [11]
                0000003F 0000   147          QUESTN = 63                     ; ASCII QUESTION MARK
                        0000   148
                        0000   149          .PSECT  WORK, NoSHR, NoEXE, WRT, BYTE
                        0000   150
                        0000   151 ;
                        0000   152 ; OWN STORAGE:
                        0000   153 ;
                        0000   154
                        0000   155 PRNT_PRMT::
                00000000 0000   156          .LONG   0
                        0004   157
                        0000   158          .PSECT  DATA, SHR, NOEXE, NOWRT, BYTE
                        0000   159
                        0000   160 INDEX_RADIX:
0000004B'0000003A'00000029'00000018' 0000   161          .LONG   10$,20$,30$,40$,50$,60$
                00000077'00000061' 0010
4C 53 21 2D 4C 53 21 2B 21 5B 20 00' 0018   162 10$:     .ASCIC  " [!+!SL-!SL(D)] "
                     20 5D 29 44 28  0024
                                 10  0018
4C 4F 21 2D 4C 4F 21 2B 21 5B 20 00' 0029   163 20$:     .ASCIC  " [!+!OL-!OL(O)] "
                     20 5D 29 4F 28  0035
                                 10  0029
4C 58 21 2D 4C 58 21 2B 21 5B 20 00' 003A   164 30$:     .ASCIC  " [!+!XL-!XL(X)] "
                     20 5D 29 58 28  0046
                                 10  003A
53 21 20 2C 29 4C 53 21 28 5B 20 00' 004B   165 40$:     .ASCIC  " [(!SL), !SL-!SL(D)] "
      20 5D 29 44 28 4C 53 21 2D 4C  0057
                                 15  004B
```

```
4F 21 20 2C 29 4C 4F 21 28 5B 20 00' 0061    166 50$:    .ASCIC  " [(!OL), !OL-!OL(0)] "
      20 5D 29 4F 28 4C 4F 21 2D 4C  006D
                                 15  0061
58 21 20 2C 29 4C 58 21 28 5B 20 00' 0077    167 60$:    .ASCIC  " [(!XL), !XL-!XL(X)] "
      20 5D 29 58 28 4C 58 21 2D 4C  0083
                                 15  0077
                                     008D    168
                                     008D    169
                                     008D    170 FMT_NODEFAULT:
73 69 20 65 72 65 68 54 20 3F 3F 00' 008D    171         .ASCIC  .?? There is no default for this parameter!/.   ;               [07]
20 74 6C 75 61 66 65 64 20 6F 6E 20  0099
72 61 70 20 73 69 68 74 20 72 6F 66  00A5
         2F 21 72 65 74 65 6D 61     00B1
                                 2B  008D
                                     00B9    172 FMT_INVALID:
20 64 69 6C 61 76 6E 49 20 3F 3F 00' 00B9    173         .ASCIC  .?? Invalid response!/.                         ;               [07]
      2F 21 65 73 6E 6F 70 73 65 72  00C5
                                 15  00B9
                                     00CF    174 FMT_PROGERR:
20 72 74 67 20 77 6F 4C 20 3F 3F 00' 00CF    175         .ASCIC  .?? Low gtr high at PC=!XL!/.                   ;               [07]
21 3D 43 50 20 74 61 20 68 67 69 68  00DB
               2F 21 4C 58           00E7
                                 1B  00CF
                                     00EB    176 FMT_RANGE:
                                     00EB    177 FMT_OVERFLOW:
72 20 66 6F 20 74 75 4F 20 3F 3F 00' 00EB    178         .ASCIC  .?? Out of range or overflow!/.                 ;               [07]
72 65 76 6F 20 72 6F 20 65 67 6E 61  00F7
               2F 21 77 6F 6C 66     0103
                                 1D  00EB
                                     0109    179 FMT_ILLCHAR:
20 6C 61 67 65 6C 6C 49 20 3F 3F 00' 0109    180         .ASCIC  "?? Illegal character!/"                        ;               [07]
   2F 21 72 65 74 63 61 72 61 68 63  0115
                                 16  0109
                                     0120    181 FMT_DEFYES:
6F 4E 20 2C 29 73 65 59 28 5B 20 00' 0120    182         .ASCIC  " [(Yes), No] "                                 ;               [07]
                              20 5D  012C
                                 0D  0120
                                     012E    183 FMT_DEFNO:
73 65 59 20 2C 29 6F 4E 28 5B 20 00' 012E    184         .ASCIC  " [(No), Yes] "                                 ;               [07]
                              20 5D  013A
                                 0D  012E
                                     013C    185 FMT_NULL:
                                 00' 013C    186         .ASCIC  ""
                                 00  013C
```

```
                     013D   188          .SBTTL  DSX$GETDATA     NUMERIC DATA INPUT ROUTINE
                 00000000   189          .PSECT  Code, Exe, Shr, NoWrt, Long              ;  [07]
                     0000   190 ;++
                     0000   191 ; FUNCTIONAL DESCRIPTION:
                     0000   192 ;
                     0000   193 ;       DISPLAYS A PROMPT MESSAGE AND ACCEPTS AN ASCII NUMERIC
                     0000   194 ;       STRING WHICH IS CONVERTED, TRUNCATED AND STORED INTO
                     0000   195 ;       A CALLER SPECIFIED VARIABLE FIELD (MASK)
                     0000   196 ;
                     0000   197 ; CALLING SEQUENCE:
                     0000   198 ;
                     0000   199 ;       CALLG   ARGLST, a#DS$GETDATA
                     0000   200 ;         OR
                     0000   201 ;       CALLS   #9, a#DS$GETDATA
                     0000   202 ;
                     0000   203 ; INPUT PARAMETERS:
                     0000   204 ;
                     0000   205 ;        4(AP) ADDRESS OF COUNTED ASCII PROMPT MESSAGE
                     0000   206 ;        8(AP) OFFSET OR ADDRESS FOR DATA STORAGE
                     0000   207 ;       12(AP) RADIX FOR ASCII TO BINARY CONVERSION
                     0000   208 ;       16(AP) BIT POSITION MASK FOR FIELD INSERTION
                     0000   209 ;       20(AP) LOW LIMIT (MAY BE OFFSET OR ADDRESS)
                     0000   210 ;       24(AP) HIGH LIMIT (MAY BE OFFSET OR ADDRESS)
                     0000   211 ;       28(AP) DEFAULT VALUE (MAY BE OFFSET OR ADDRESS)
                     0000   212 ;       32(AP) DEVICE TYPE MASK
                     0000   213 ;       36(AP) EXCEPTION MASK
                     0000   214 ;
                     0000   215 ; IMPLICIT INPUTS:      NONE
                     0000   216 ;
                     0000   217 ; OUTPUT PARAMETERS:
                     0000   218 ;
                     0000   219 ;       NONE
                     0000   220 ;
                     0000   221 ; IMPLICIT OUTPUTS:
                     0000   222 ;
                     0000   223 ;       NONE
                     0000   224 ;
                     0000   225 ; COMPLETION CODES:
                     0000   226 ;
                     0000   227 ;       SS$_NORMAL
                     0000   228 ;       DS$_TRUNCATE
                     0000   229 ;       DS$_PROGERR
                     0000   230 ;
                     0000   231 ; SIDE EFFECTS:
                     0000   232 ;
                     0000   233 ;       R1 = CONVERTED BINARY VALUE (NOT TRUNCATED)
                     0000   234 ;
                     0000   235 ;--
```

```
                                 0FFC  0000  237 .ENTRY  DSX$GETDATA,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;                    [09]
                                       0002  238
                            13    E1  0002  239          BBC     #DSA$V_ASK_PROMPT,-      ; Flag to print ASK> prompt
                07 0000FE00'EF        0004  240                  L^DSA$GL_FLAGS,5$
            00000000'EF     01    D0  000A  241          MOVL    #1,PRNT_PRMT             ; Flag to getline to print prompt for $DSASK
                                       0011  242
        50   24 AC   01     01    EE  0011  243 5$:      EXTV    #PAR$V_NODEF, #1, -      ; Extract the NoDef flag                [08]
                                       0017  244                    36(AP), R0             ;                                       [08]
   00000000'EF   01   00     50    F0  0017  245          INSV    R0, S^#QA$K_NoDef, -     ; Store the NoDef flag in the           [08]
                                       0020  246                     #1, L^QA$AOB_Flags    ; QA$AOB_Flags bit vector for QA        [08]
                                       0020  247          QA_MAIN Param_DSX$GetData        ; Call QA$MAIN                          [07]
                            0C    DD  0020                PUSHL   #DSQA$K_Param_DSX$GetData
            00000000'9F     01    FB  0022                CALLS   #1, a#QA$MAIN
                                       0029  248
            50   00660020 8F      D0  0029  249          MOVL    #DS$_PROGERR,R0          ; ASSUME PROGRAMMER ERROR
                      6C    09    D1  0030  250          CMPL    #9,(AP)                  ; CHECK ARG LIST VALIDITY
                            4F    12  0033  251          BNEQ    GETDATA_X                ; INCORRECT NUMBER OF ARGS
                        10 AC     D5  0035  252          TSTL    16(AP)                   ; ANY BITS IN FIELD?
                            4A    13  0038  253          BEQL    GETDATA_X                ; ERROR, NULL FIELD
                                       003A  254
                            01    DD  003A  255          PUSHL   #1                       ; Flag signed comparisons
                        24 AC     DD  003C  256          PUSHL   36(AP)                   ; Flags
                        1C AC     DD  003F  257          PUSHL   28(AP)                   ; Default value
                  7E    14 AC     7D  0042  258          MOVQ    20(AP),-(SP)             ; Low and High
                        0C AC     DD  0046  259          PUSHL   12(AP)                   ; Conversion radix
                        04 AC     DD  0049  260          PUSHL   4(AP)                    ; Prompt string
            000003FB'EF     07    FB  004C  261          CALLS   #7,RESPONSE              ; GET RESPONSE FROM OPERATOR
                        2E 50     E9  0053  262          BLBC    R0,GETDATA_X             ; EXIT IF ERROR
                                       0056  263
        53   10 AC   20     00    EA  0056  264          FFS     #0,#32,16(AP),R3         ; GET POSITION OF FIELD
                54   20     53    C3  005C  265          SUBL3   R3,#32,R4                ; DETERMINE REMAINING # OF BITS
        54   10 AC   54     53    EB  0060  266          FFC     R3,R4,16(AP),R4          ; GET SIZE OF FIELD
                     54     53    C2  0066  267          SUBL    R3,R4                    ; Make field size
                                       0069  268
        08 BC   54   53     51    F0  0069  269          INSV    R1,R3,R4,a8(AP)          ; STORE THE PARAMETER
        53   08 BC   54     53    EF  006F  270          EXTZV   R3,R4,a8(AP),R3          ; GET STORED FIELD
                                       0075  271
                     50     01    D0  0075  272          MOVL    #SS$_NORMAL,R0           ; ASSUME IT'S OK
                     53     51    D1  0078  273          CMPL    R1,R3                    ; INSERTED SAME AS EXTRACTED?
                            07    13  007B  274          BEQL    GETDATA_X                ; BRANCH IF NOT TRUNCATED
                                       007D  275
            50   00660028 8F      D0  007D  276          MOVL    #DS$_TRUNCATE,R0         ; YES, INDICATE TRUNCATED
                                       0084  277
                                       0084  278 GETDATA_X:
                                 04    0084  279          RET                              ; RETURN
```

```
                     0085   281          .SBTTL  DSX$GETVIELD    BIT FIELD INPUT ROUTINE
                     0085   282 ;++
                     0085   283 ; FUNCTIONAL DESCRIPTION:
                     0085   284 ;
                     0085   285 ;       DISPLAYS A PROMPT MESSAGE AND ACCEPTS AN ASCII NUMERIC
                     0085   286 ;       STRING WHICH IS CONVERTED, TRUNCATED AND STORED INTO
                     0085   287 ;       A CALLER SPECIFIED VARIABLE FIELD (POSITION & SIZE)
                     0085   288 ;
                     0085   289 ; CALLING SEQUENCE:
                     0085   290 ;
                     0085   291 ;       CALLG   ARGLST, a#DS$GETVIELD
                     0085   292 ;         OR
                     0085   293 ;       CALLS   #10, a#DS$GETVIELD
                     0085   294 ;
                     0085   295 ; INPUT PARAMETERS:
                     0085   296 ;
                     0085   297 ;        4(AP) ADDRESS OF COUNTED ASCII PROMPT MESSAGE
                     0085   298 ;        8(AP) OFFSET OR ADDRESS FOR DATA STORAGE
                     0085   299 ;       12(AP) RADIX FOR ASCII TO BINARY CONVERSION
                     0085   300 ;       16(AP) BIT POSITION FOR FIELD INSERTION
                     0085   301 ;       20(AP) FIELD SIZE FOR FIELD INSERTION
                     0085   302 ;       24(AP) LOW LIMIT (MAY BE OFFSET OR ADDRESS)
                     0085   303 ;       28(AP) HIGH LIMIT (MAY BE OFFSET OR ADDRESS)
                     0085   304 ;       32(AP) DEFAULT VALUE (MAY BE OFFSET OR ADDRESS)
                     0085   305 ;       36(AP) DEVICE TYPE MASK
                     0085   306 ;       40(AP) EXCEPTION MASK
                     0085   307 ;
                     0085   308 ; IMPLICIT INPUTS:      NONE
                     0085   309 ;
                     0085   310 ; OUTPUT PARAMETERS:
                     0085   311 ;
                     0085   312 ;       NONE
                     0085   313 ;
                     0085   314 ; IMPLICIT OUTPUTS:
                     0085   315 ;
                     0085   316 ;       NONE
                     0085   317 ;
                     0085   318 ; COMPLETION CODES:
                     0085   319 ;
                     0085   320 ;       SS$_NORMAL
                     0085   321 ;       DS$_TRUNCATE
                     0085   322 ;       DS$_PROGERR
                     0085   323 ;
                     0085   324 ; SIDE EFFECTS:
                     0085   325 ;
                     0085   326 ;       R1 = CONVERTED BINARY VALUE (NOT TRUNCATED)
                     0085   327 ;
                     0085   328 ;--
```

```
                                  0FFC  0085   330 .ENTRY  DSX$GETVIELD,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>        ;                [09]
                                        0087   331
                             13    E1  0087   332          BBC     #DSA$V_ASK_PROMPT,-      ; Flag to print ASK> prompt
                07 0000FE00'EF         0089   333                  L^DSA$GL_FLAGS,5$
              00000000'EF    01    D0  008F   334          MOVL    #1,PRNT_PRMT             ; Flag to getline to print prompt for $DSASK
                                        0096   335
        50    28 AC    01    01    EE  0096   336 5$:      EXTV    #PAR$V_NODEF, #1, -      ; Extract the NoDef flag                [08]
                                        009C   337                    40(AP), R0             ;                                       [08]
   00000000'EF    01    00    50    F0  009C   338          INSV    R0, S^#QA$K_NoDef, -     ; Store the NoDef flag in the           [08]
                                        00A5   339                      #1, L^QA$AOB_Flags   ; QA$AOB_Flags bit vector for QA        [08]
                                        00A5   340          QA_MAIN Param_DSX$GetVield       ; Call QA$Main                          [07]
                             0D    DD  00A5               PUSHL   #DSQA$K_Param_DSX$GetVield
              00000000'9F    01    FB  00A7               CALLS   #1, a#QA$MAIN
                                        00AE   341
              50   00660020 8F    D0  00AE   342          MOVL    #DS$_PROGERR,R0          ; ASSUME ERROR
                        6C    0A    D1  00B5   343          CMPL    #10,(AP)                 ; CHECK ARG LIST VALIDITY
                             3B    12  00B8   344          BNEQ    GETVIELD_X               ; INCORRECT NUMBER OF ARGS
                                        00BA   345
                             01    DD  00BA   346          PUSHL   #1                       ; Do signed comparisons
                          28 AC    DD  00BC   347          PUSHL   40(AP)                   ; Flags
                          20 AC    DD  00BF   348          PUSHL   32(AP)                   ; Default
                     7E   18 AC    7D  00C2   349          MOVQ    24(AP),-(SP)             ; Low and high
                          0C AC    DD  00C6   350          PUSHL   12(AP)                   ; Conversion radix
                          04 AC    DD  00C9   351          PUSHL   4(AP)                    ; Prompt string
              000003FB'EF    07    FB  00CC   352          CALLS   #7,RESPONSE              ; GET RESPONSE FROM OPERATOR
                          1F 50    E9  00D3   353          BLBC    R0,GETVIELD_X            ; EXIT IF ERROR
                                        00D6   354
   08 BC    14 AC    10 AC    51    F0  00D6   355          INSV    R1,16(AP),20(AP),a8(AP) ; STORE THE PARAMETER
   53    08 BC    14 AC    10 AC    EF  00DE   356          EXTZV   16(AP),20(AP),a8(AP),R3 ; GET FIELD JUST STORED
                                        00E6   357
                        50    01    D0  00E6   358          MOVL    #SS$_NORMAL,R0           ; ASSUME IT WORKED
                        53    51    D1  00E9   359          CMPL    R1,R3                    ; SAME?
                             07    13  00EC   360          BEQL    GETVIELD_X               ; NO, NONE FOUND
              50   00660028 8F    D0  00EE   361          MOVL    #DS$_TRUNCATE,R0         ; NO, INDICATE TRUNCATED
                                        00F5   362
                                        00F5   363 GETVIELD_X:
                             04         00F5   364          RET                              ; RETURN
```

```
                        .SBTTL  DSX$GETADDRESS  ADDRESS INPUT ROUTINE
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       DISPLAYS A PROMPT MESSAGE AND ACCEPTS AN ASCII NUMERIC
                ;       ADDRESS STRING WHICH IS CONVERTED AND STORED INTO A
                ;       CALLER SPECIFIED LONGWORD
                ;
                ; CALLING SEQUENCE:
                ;
                ;       CALLG   ARGLST, a#DS$GETADDRESS
                ;         OR
                ;       CALLS   #8, a#DS$GETADDRESS
                ;
                ; INPUT PARAMETERS:
                ;
                ;        4(AP) ADDRESS OF COUNTED ASCII PROMPT MESSAGE
                ;        8(AP) OFFSET OR ADDRESS FOR DATA STORAGE
                ;       12(AP) RADIX FOR ASCII TO BINARY CONVERSION
                ;       16(AP) LOW LIMIT (MAY BE OFFSET OR ADDRESS)
                ;       20(AP) HIGH LIMIT (MAY BE OFFSET OR ADDRESS)
                ;       24(AP) DEFAULT VALUE (MAY BE OFFSET OR ADDRESS)
                ;       28(AP) DEVICE TYPE MASK
                ;       32(AP) EXCEPTION MASK
                ;
                ; IMPLICIT INPUTS:      NONE
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;
                ;       SS$_NORMAL
                ;       DS$_TRUNCATE
                ;       DS$_PROGERR
                ;
                ; SIDE EFFECTS:
                ;
                ;       R1 = CONVERTED BINARY ADDRESS
                ;
                ;--
```

```
                          0FFC  00F6   413 .ENTRY  DSX$GETADDRESS,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ;              [09]
                                00F8   414
                       13   E1  00F8   415         BBC     #DSA$V_ASK_PROMPT,-      ; Flag to print ASK> prompt
                 07 0000FE00'EF        00FA   416                 L^DSA$GL_FLAGS,5$
               00000000'EF   01   D0  0100   417         MOVL    #1,PRNT_PRMT             ; Flag to getline to print prompt for $DSASK
                                0107   418
        50   20 AC   01   01   EE  0107   419 5$:      EXTV    #PAR$V_NODEF, #1, -      ; Extract the NoDef flag                [08]
                                010D   420                   32(AP), R0               ;                                       [08]
   00000000'EF   01   00   50   F0  010D   421         INSV    R0, S^#QA$K_NoDef, -     ; Store the NoDef flag in the           [08]
                                0116   422                    #1, L^QA$AOB_Flags     ; QA$AOB_Flags bit vector for QA        [08]
                                0116   423         QA_MAIN Param_DSX$GetAddress      ; Call QA$Main                          [07]
                       0E   DD  0116           PUSHL   #DSQA$K_Param_DSX$GetAddress
               00000000'9F   01   FB  0118           CALLS   #1, a#QA$MAIN
                                011F   424
               50   00660020 8F   D0  011F   425         MOVL    #DS$_PROGERR,R0          ; ASSUME BAD CALL
                          6C   08   D1  0126   426         CMPL    #8,(AP)                  ; CHECK ARG LIST VALIDITY
                               CA   12  0129   427         BNEQ    GETVIELD_X               ; INCORRECT NUMBER OF ARGS
                                012B   428
                               00   DD  012B   429         PUSHL   #0                       ; Use unsigned comparisons
                             20 AC   DD  012D   430         PUSHL   32(AP)                   ; Flags
                             18 AC   DD  0130   431         PUSHL   24(AP)                   ; Default
                          7E   10 AC   7D  0133   432         MOVQ    16(AP),-(SP)             ; Low and high
                             0C AC   DD  0137   433         PUSHL   12(AP)                   ; Radix
                             04 AC   DD  013A   434         PUSHL   4(AP)                    ; prompt string
               000003FB'EF   07   FB  013D   435         CALLS   #7,RESPONSE              ; GET RESPONSE FROM OPERATOR
                          07   50   E9  0144   436         BLBC    R0,GETADDRESS_X          ; EXIT IF ERROR
                                0147   437
                    08 BC   51   D0  0147   438         MOVL    R1,a8(AP)                ; STORE THE ADDRESS
                       50   01   D0  014B   439         MOVL    #SS$_NORMAL,R0           ; INDICATE SUCCESS
                                014E   440
                                014E   441 GETADDRESS_X:                                ;
                          04    014E   442         RET                                  ; RETURN
```

```
                        444          .SBTTL  DSX$GETLOGICAL  LOGICAL RESPONSE ROUTINE
                        445 ;++
                        446 ; FUNCTIONAL DESCRIPTION:
                        447 ;
                        448 ;       DISPLAYS A PROMPT MESSAGE AND ACCEPTS AN ASCII "YES" OR
                        449 ;       "NO" STRING WHICH IS CONVERTED TO A SINGLE BIT (FLAG) AND
                        450 ;       STORED INTO A CALLER SPECIFIED BIT WITHIN A BYTE
                        451 ;
                        452 ; CALLING SEQUENCE:
                        453 ;
                        454 ;       CALLG   ARGLST, a#DS$GETLOGICAL
                        455 ;         OR
                        456 ;       CALLS   #7, a#DS$GETLOGICAL
                        457 ;
                        458 ; INPUT PARAMETERS:
                        459 ;
                        460 ;        4(AP) ADDRESS OF COUNTED ASCII PROMPT MESSAGE
                        461 ;        8(AP) OFFSET OR ADDRESS FOR DATA STORAGE
                        462 ;       12(AP) BIT POSITION WITHIN BYTE
                        463 ;       16(AP) TRANSFER ADDRESS IF YES
                        464 ;       20(AP) TRANSFER ADDRESS IF NO
                        465 ;       24(AP) DEFAULT VALUE (0=NO, 1=YES)
                        466 ;       28(AP) DEVICE TYPE MASK
                        467 ;
                        468 ; IMPLICIT INPUTS:      NONE
                        469 ;
                        470 ; OUTPUT PARAMETERS:
                        471 ;
                        472 ;       R1 = TRANSFER ADDRESS
                        473 ;
                        474 ; IMPLICIT OUTPUTS:
                        475 ;
                        476 ;       PC = TRANSFER ADDRESS   ; IF CALLED FROM PROGRAM AND TRANSFER
                        477 ;                               ; ADDRESS IS NOT EQUAL TO ZERO
                        478 ;
                        479 ; COMPLETION CODES:
                        480 ;
                        481 ;       SS$_NORMAL
                        482 ;       DS$_PROGERR
                        483 ;
                        484 ; SIDE EFFECTS:         NONE
                        485 ;--
```

```
                        0FFC  014F  487 .ENTRY  DSX$GETLOGICAL,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ;                [09]
                              0151  488
                   13    E1   0151  489         BBC     #DSA$V_ASK_PROMPT,-      ; Flag to print ASK> prompt
        07 0000FE00'EF        0153  490                 L^DSA$GL_FLAGS,3$
     00000000'EF   01    D0   0159  491         MOVL    #1,PRNT_PRMT             ; Flag to getline to print prompt for $DSASK
                              0160  492
     00000000'EF   00    E5   0160  493 3$:     BBCC    S^#QA$K_NoDef, -         ; Clear the no-default flag since the   [08]
                   00         0167  494                   L^QA$AOB_Flags, 5$     ; default is automatically given        [08]
                              0168  495 5$:                                      ;
                              0168  496         QA_MAIN Param_DSX$GetLogical     ; Call QA$Main                          [07]
                   0F    DD   0168              PUSHL   #DSQA$K_Param_DSX$GetLogical
     00000000'9F   01    FB   016A              CALLS   #1, @#QA$MAIN
                              0171  497
  0000FE00'EF   01   0D    EF   0171  498         EXTZV   #DSA$V_PROMPT, #1, -
                   7E         0179  499                   L^DSA$GL_FLAGS, -(SP) ; SAVE STATE OF PROMPT FLAG             [07]
     50   00660020 8F    D0   017A  500         MOVL    #DS$_PROGERR,R0          ; ASSUME POSITION  GTR 7...ERROR
             6C    07    D1   0181  501         CMPL    #7,(AP)                  ; CHECK ARG LIST VALIDITY
                   06    12   0184  502         BNEQ    10$                      ; INCORRECT NUMBER OF ARGS
          07   0C AC     D1   0186  503         CMPL    12(AP),#7                ; LOOK AT POSITION VARIABLE
                   08    1B   018A  504         BLEQU   RETRY_LOGICAL            ; BRANCH IF IN A BYTE
                              018C  505 10$:
     50   00660020 8F    D0   018C  506         MOVL    #DS$_PROGERR,R0          ; PROGRAMMER ERROR
                         04   0193  507         RET
                              0194  508
                              0194  509 RETRY_LOGICAL:
          0000013C'EF    9F   0194  510         PUSHAB  L^FMT_NULL               ; Address of empty prompt string        [07]
     0000FE00'EF   0D    E3   019A  511         BBCS    #DSA$V_PROMPT, L^DSA$GL_FLAGS -        ;                        [07]
                   12         01A1  512                 , 10$                    ; NO PROMPT                             [07]
     6E   00000120'EF    9E   01A2  513         MOVAB   L^FMT_DEFYES,(SP)        ; ASSUME DEFAULT IS YES                 [07]
          07 18 AC       E8   01A9  514         BLBS    24(AP), 10$              ; BRANCH IF DEFAULT IS YES              [07]
     6E   0000012E'EF    9E   01AD  515         MOVAB   L^FMT_DEFNO,(SP)         ; DEFAULT IS NO                         [07]
                              01B4  516 10$:
             04 AC       DD   01B4  517         PUSHL   4(AP)                    ; Address of user prompt
                   04    DD   01B7  518         PUSHL   #4                       ; Up to 4 characters
          00000000'EF    DF   01B9  519         PUSHAL  L^DS$GL_BUFLEN           ; Address of longword to get length     [07]
          00000000'EF    9F   01BF  520         PUSHAB  L^DS$GT_STRBUF           ; Address of buffer                     [07]
     00000000'EF   05    FB   01C5  521         CALLS   #5,L^DS_GETLINE          ; Get a line with dual prompt           [07]
                              01CC  522                                          ;                                       [07]
          50   18 AC     D0   01CC  523         MOVL    24(AP),R0                ; GET DEFAULT RESPONSE                  [07]
          00000000'EF    D5   01D0  524         TSTL    L^DS$GL_BUFLEN           ; DEFAULT RESPONSE?                     [07]
                   30    13   01D6  525         BEQL    80$                      ; YES, USE IT
                              01D8  526
             50    00    D0   01D8  527         MOVL    #PAR$_NO,R0              ; ASSUME NEGATIVE RESPONSE
  00000000'EF   4E 8F    91   01DB  528         CMPB    #^A"N",L^DS$GT_STRBUF    ; DID HE TYPE 'N........                [07]
                   23    13   01E3  529         BEQL    80$                      ; YES, USE IT
             50    01    D0   01E5  530         MOVL    #PAR$_YES,R0             ; ASSUME POSITIVE
  00000000'EF   59 8F    91   01E8  531         CMPB    #^A"Y",L^DS$GT_STRBUF    ; DID HE TYPE 'Y........                [07]
                   16    13   01F0  532         BEQL    80$                      ; YES, USE IT
                              01F2  533
                              01F2  534         $PRINT  #DS$K_TYPE_PARAM_ERROR,- ; PROVIDE A HELP MESSAGE                [10]
                              01F2  535                 #DS$K_PRINTF,-
                              01F2  536                 L^FMT_INVALID
          000000B9'EF    9F   01F2                     PUSHAB  L^FMT_INVALID
             7E    01    9A   01F8                     movzbl  #DS$K_PRINTF, -(sp)
             7E    1C    98   01FB                     cvtbl   #DS$K_TYPE_PARAM_ERROR, -(sp)
     00000000'GF   03    FB   01FE                     CALLS   #$$N+2, G^DSX$PRINT
                FF8C     31   0205  537         BRW     RETRY_LOGICAL            ; AND TRY AGAIN
```

```
                                0208    538
      08 BC    01    0C AC    50    F0  0208    539 80$:    INSV    R0,12(AP),#1,a8(AP)     ; STORE THE PARAMETER
                                0020F   540
                     51    10 AC    D0  020F    541         MOVL    16(AP),R1               ; USE YES TRANSFER
                           04 50    E8  0213    542         BLBS    R0,120$                 ; IF SET USE YES TRANSFER
                     51    14 AC    D0  0216    543         MOVL    20(AP),R1               ; GET NO TRANSFER
                           51       D5  021A    544 120$:   TSTL    R1                      ; LOOK AT TRANSFER POINT
                           43       13  021C    545         BEQL    130$                    ; FORGET IT IF ZERO
                                    021E    546
                                    021E    547         BR_IF_MP 125$                       ; Alter fp for interlocking code [17]
  05000000 8F    00000000'8F       D1  021E                CMPL    #DS$GK_SID,#^X05000000          ; Branch if Scorpio [61]
                              2E    13  0229                BEQL    125$                            ; Yes, branch
  06000000 8F    00000000'8F       D1  022B                CMPL    #DS$GK_SID,#^X06000000          ; Branch if Nautilus [61]
                              21    13  0236                BEQL    125$                            ; Yes, branch
  11000000 8F    00000000'8F       D1  0238                CMPL    #DS$GK_SID,#^X11000000          ; Branch if RRR [69]          ;G:20
                              14    13  0243                BEQL    125$                                                          ;G:18
  0A000000 8F    00000000'8F       D1  0245                CMPL    #DS$GK_SID,#^X0A000000          ; Branch if LLL [69]          ;G:20
                              07    13  0250                BEQL    125$                                                          ;G:22
                                    0252    548
                     10 AD    51    D0  0252    549         MOVL    R1,16(FP)               ; Alter return                  [17]
                           0008    31  0256    550         BRW     130$                    ; NON-MP SYSTEM                 [17]
                                    0259    551
                                    0259    552 ; The tranfer address must be loaded into the saved frame pointers old pc. [17]
                                    0259    553 ; This is necessary since the current stack frame was built as a result    [17]
                                    0259    554 ; of interlocking routines.                                                [17]
                                    0259    555
                     50    0C AD    D0  0259    556 125$:   MOVL    12(FP),R0               ; GET SAVED FRAME POINTER       [17]
                     10 A0    51    D0  025D    557         MOVL    R1,16(R0)               ; ALTER RET ADR OF SAVED FP'S pc [17]
                        50    01    D0  0261    558 130$:   MOVL    #SS$_NORMAL,R0          ; INDICATE SUCCESS
                                    0264    559
                                    0264    560 GETLOGICAL_X:
  0000FE00'EF    01    0D    FC AD    F0  0264    561         INSV    -4(FP), #DSA$V_PROMPT, -
                                    026E    562                 #1, L^DSA$GL_FLAGS      ; RESTORE PROMPT FLAG           [07]
                                 04 026E    563         RET                             ; RETURN
```

```
                        026F    565          .SBTTL  DSX$GETSTRING   ASCII STRING INPUT ROUTINE
                        026F    566 ;++
                        026F    567 ; FUNCTIONAL DESCRIPTION:
                        026F    568 ;
                        026F    569 ;       DISPLAYS A PROMPT MESSAGE AND ACCEPTS AN ASCII STRING RESPONSE
                        026F    570 ;       WHICH IS VALIDATED AGAINST A LIST OF STRINGS BEFORE BEING
                        026F    571 ;       RETURNED TO THE CALLER SPECIFIED BUFFER
                        026F    572 ;
                        026F    573 ; CALLING SEQUENCE:
                        026F    574 ;
                        026F    575 ;       CALLG   ARGLST, a#DS$GETSTRING
                        026F    576 ;         OR
                        026F    577 ;       CALLS   #6, a#DS$GETSTRING
                        026F    578 ;
                        026F    579 ; INPUT PARAMETERS:
                        026F    580 ;
                        026F    581 ;        4(AP) ADDRESS OF COUNTED ASCII PROMPT MESSAGE
                        026F    582 ;        8(AP) OFFSET OR ADDRESS FOR STRING STORAGE
                        026F    583 ;       12(AP) SIZE OF RETURN STRING BUFFER
                        026F    584 ;       16(AP) ADDRESS OF COUNTED LIST OF COUNTED ASCII STRING POINTERS
                        026F    585 ;       20(AP) ADDRESS OF DEFAULT COUNTED ASCII STRING
                        026F    586 ;       24(AP) DEVICE TYPE MASK
                        026F    587 ;
                        026F    588 ; IMPLICIT INPUTS:      NONE
                        026F    589 ;
                        026F    590 ; OUTPUT PARAMETERS:
                        026F    591 ;
                        026F    592 ;       R1 = INDEX OF MATCHED STRING IN VALIDATION TABLE IF SUCCESSFUL
                        026F    593 ;          = 0 IF DEFAULT STRING WAS SELECTED
                        026F    594 ;
                        026F    595 ;       R1 = UNPREDICTABLE ON ERROR STATUS
                        026F    596 ;
                        026F    597 ; IMPLICIT OUTPUTS:
                        026F    598 ;
                        026F    599 ;       NONE
                        026F    600 ;
                        026F    601 ; COMPLETION CODES:
                        026F    602 ;
                        026F    603 ;       SS$_NORMAL
                        026F    604 ;       DS$_TRUNCATE
                        026F    605 ;       DS$_PROGERR
                        026F    606 ;
                        026F    607 ; SIDE EFFECTS:
                        026F    608 ;
                        026F    609 ;       NONE
                        026F    610 ;
                        026F    611 ;--
```

```
                        0FFC  026F  .ENTRY  DSX$GETSTRING,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ;                [09]
                              0271
                   13    E1   0271          BBC     #DSA$V_ASK_PROMPT,-      ; Flag to print ASK> prompt
        07 0000FE00'EF        0273                  L^DSA$GL_FLAGS,5$
     00000000'EF   01    D0   0279          MOVL    #1,PRNT_PRMT             ; Flag to getline to print prompt for $DSASK
                              0280
                14 AC    D5   0280  5$:     TSTL    20(AP)                   ; Check to see if a default string      [08]
                   0A    12   0283          BNEQ    10$                      ; If default was given, branch          [08]
     00000000'EF   00    E3   0285          BBCS    S^#QA$K_NoDef, -         ; Set NoDef flag: no default was given  [08]
                   0A         028C                    L^QA$AOB_Flags, 20$    ;                                       [08]
                   08    11   028D          BRB     20$                      ;                                       [08]
                              028F
     00000000'EF   00    E5   028F  10$:    BBCC    S^#QA$K_NoDef, -         ; Clear NoDef flag: default was given   [08]
                   00         0296                    L^QA$AOB_Flags, 20$    ;                                       [08]
                              0297
                              0297  20$:                                    ;                                       [08]
                              0297          QA_MAIN Param_DSX$GetString      ; Call QA$Main                          [07]
                   10    DD   0297          PUSHL   #DSQA$K_Param_DSX$GetString
     00000000'9F   01    FB   0299          CALLS   #1, a#QA$MAIN
                              02A0
 0000FE00'EF   01   0D    EF   02A0          EXTZV   #DSA$V_PROMPT,#1, -
                   7E         02A8                  L^DSA$GL_FLAGS ,-(SP)    ; SAVE CURRENT PROMPT FLAG              [07]
              6C   06    D1   02A9          CMPL    #6,(AP)                  ; CHECK ARG LIST VALIDITY
                   0A    13   02AC          BEQL    RETRY_STRING             ; OK, PROCEED
     50   00660020 8F    D0   02AE          MOVL    #DS$_PROGERR,R0          ; INSUFFICIENT ARGS
                 0138    31   02B5          BRW     GETSTRING_X
                              02B8
                              02B8  RETRY_STRING:
        5E   FC AD       DE   02B8          MOVAL   -4(FP),SP                ; SPACE FOR LOCAL STORAGE
                              02BC  ;+
                              02BC  ;       BUILD A PROMPT MESSAGE ON THE STACK FROM THE VALIDATION
                              02BC  ;       LIST, DEFAULT, AND PROMPT TEXT
                              02BC  ;-
     56   0000013C'EF    9E   02BC          MOVAB   L^FMT_NULL,R6            ; Address of information prompt         [07]
              57   5E    D0   02C3          MOVL    SP,R7                    ; SAVE HIGHEST ADDR OF STRING           [07]
     0000FE00'EF   0D    E3   02C6          BBCS    #DSA$V_PROMPT,L^DSA$GL_FLAGS -          ;                       [07]
                   5F         02CD                  , 70$                    ; IF NOT PROMPTING SKIP
        53   10 AC       D0   02CE          MOVL    16(AP),R3                ; GET ADDRESS OF COUNTED LIST
                   59    13   02D2          BEQL    70$                      ; NONE, SKIP
              52   63    D0   02D4          MOVL    (R3),R2                  ; GET COUNT
        7E   205D 8F     B0   02D7          MOVW    #^A"] ",-(SP)            ; TRAILING ]
                   05    11   02DC          BRB     20$                      ; SKIP INITIAL ", "
                              02DE  10$:
        7E   202C 8F     B0   02DE          MOVW    #^A", ",-(SP)            ; IF NOT FIRST PRECEED WITH ", "
                              02E3  20$:
           50   6342     D0   02E3          MOVL    (R3)[R2],R0              ; GET ADDRESS OF COUNTED STRING
                   14    13   02E7          BEQL    35$                      ; IGNORE NULL ENTRIES
        14 AC      50    D1   02E9          CMPL    R0,20(AP)                ; IS THIS THE DEFAULT?
                   0E    13   02ED          BEQL    35$                      ; YES, SKIP IT
              51   60    9A   02EF          MOVZBL  (R0),R1                  ; GET LENGTH
                   09    13   02F2          BEQL    35$                      ; NULL SKIP COPY
           7E   6041     90   02F4  30$:    MOVB    (R0)[R1],-(SP)           ; COPY BYTE
              F9 51      F5   02F8          SOBGTR  R1,30$                   ; COPY STRING
                   02    11   02FB          BRB     40$                      ; NEXT!
                   8E    B5   02FD  35$:    TSTW    (SP)+                    ; REMOVE EXTRA ", "
              DC 52      F5   02FF  40$:    SOBGTR  R2,10$                   ; COPY REST OF VALID STRINGS
                              0302
```

```
                50   14 AC    D0  0302   668          MOVL    20(AP),R0                       ; GET ADDRESS OF DEFAULT
                        17    13  0306   669          BEQL    60$                             ; NONE, FINISH IT
                   51   60    9A  0308   670          MOVZBL  (R0),R1                         ; GET LENGTH
                        12    13  030B   671          BEQL    60$                             ; NULL, SKIP COPY
               7E   202C 8F    B0  030D   672          MOVW    #^A", ",-(SP)                   ; SEPARATE FROM REST OF OPTIONS
                   7E   29    90  0312   673          MOVB    #^A")",-(SP)
                  7E   6041   90  0315   674 50$:      MOVB    (R0)[R1],-(SP)                  ; COPY BYTE
                     F9 51    F5  0319   675          SOBGTR  R1,50$                          ; COPY STRING
                   7E   28    90  031C   676          MOVB    #^A"(",-(SP)
                               031F   677 60$:
               7E   5B20 8F    B0  031F   678          MOVW    #^A" [",-(SP)                   ; FINISH DEFAULT AND OPTION LIST
                   57   5E    C2  0324   679          SUBL2   SP,R7                           ; LENGTH OF PROMPT STRING
                   7E   57    90  0327   680          MOVB    R7,-(SP)                        ; Make ASCIC prompt string
                   56   5E    D0  032A   681          MOVL    SP,R6                           ; Supercede prompt address
                               032D   682 70$:
                        56    DD  032D   683          PUSHL   R6                              ; Put Address of infor prompt on stack
                     04 AC    DD  032F   684          PUSHL   4(AP)                           ; Address of user prompt
            7E   0C AC    01    C3  0332   685          SUBL3   #1,12(AP),-(SP)                 ; GET LENGTH AVAILABLE FOR STRING
                00000000'EF   DF  0337   686          PUSHAL  L^DS$GL_BUFLEN                  ; Longword for returned length          [07]
                 52   08 AC    D0  033D   687          MOVL    8(AP),R2                        ; GET ADDRESS OF BUFFER
                      01 A2    9F  0341   688          PUSHAB  1(R2)                           ; Push address of buffer
          00000000'EF   05    FB  0344   689          CALLS   #5,L^DS_GETLINE                 ; Get a line of input                   [07]
                               034B   690
          62   00000000'EF    90  034B   691          MOVB    L^DS$GL_BUFLEN,(R2)             ; MAKE IT ASCIC                         [07]
                        31    12  0352   692          BNEQ    99$                             ; NOT DEFAULTED
                               0354   693
                50   14 AC    D0  0354   694          MOVL    20(AP),R0                               ; GET ADDRESS OF DEFAULT
                        1E    12  0358   695          BNEQ    94$                                     ; GOT ONE, USE IT
                               035A   696
                               035A   697          $PRINT  #DS$K_TYPE_PARAM_ERROR,-                ;  TELL OPERATOR                [10]
                               035A   698                  #DS$K_PRINTF,-
                               035A   699                  L^FMT_NODEFAULT
                0000008D'EF    9F  035A                    PUSHAB  L^FMT_NODEFAULT
                   7E   01    9A  0360                    movzbl  #DS$K_PRINTF, -(sp)
                   7E   1C    98  0363                    cvtbl   #DS$K_TYPE_PARAM_ERROR, -(sp)
          00000000'GF   03    FB  0366                    CALLS   #$$N+2, G^DSX$PRINT
          0000FE00'EF   0C    E1  036D   700          BBC     #DSA$V_OPER,L^DSA$GL_FLAGS -    ;                               [07]
                        74        0374   701                  , GETSTRING_E                   ; IF OPERATOR ABSENT, TAKE ERROR EXI
                     FF40   31    0375   702          BRW     RETRY_STRING                    ; TRY AGAIN
                               0378   703
                   53   60    9A  0378   704 94$:      MOVZBL  (R0),R3                 ; GET LENGTH
                   82   80    90  037B   705 95$:      MOVB    (R0)+,(R2)+             ; COPY DEFAULT TO INPUT BUFFER
                     FA 53    F4  037E   706          SOBGEQ  R3,95$                  ; COPY STRING
                        54    D4  0381   707          CLRL    R4                      ; Clear index to indicate default
                        39    11  0383   708          BRB     140$                    ; Set returns and exit
                               0385   709
                53   10 AC    D0  0385   710 99$:      MOVL    16(AP),R3               ; GET STRING LIST ADDRESS
                        33    13  0389   711          BEQL    140$                    ; IF NONE, JUST FINISH
                   54   63    D0  038B   712          MOVL    (R3),R4                 ; GET COUNT OF VALID STRINGS
                        2E    13  038E   713          BEQL    140$                    ; IF NONE, JUST FINISH
                 51   6344    D0  0390   714 100$:     MOVL    (R3)[R4],R1             ; GET STRING ADDRESS
                50   08 AC    D0  0394   715          MOVL    8(AP),R0                ; GET ADDRESS OF STRING
                   52   60    9A  0398   716          MOVZBL  (R0),R2                 ; GET LENGTH OF STRING INPUT
                   81   80    91  039B   717 110$:     CMPB    (R0)+,(R1)+             ; MATCH?
                        05    12  039E   718          BNEQ    120$                    ; NO, GO SET UP NEXT STRING
                     F8 52    F4  03A0   719          SOBGEQ  R2,110$                 ; COMPARE WHOLE STRING
                        19    11  03A3   720          BRB     140$                    ; COMPLETE MATCH USE IT
```

```
                E8 54    F5   03A5    721 120$:   SOBGTR  R4,100$                    ; COUNT AND CONTINUE
                          03A8    722
                          03A8    723         $PRINT  #DS$K_TYPE_PARAM_ERROR,-   ; FAILURE, NO MATCH FOUND              [10]
                          03A8    724                 #DS$K_PRINTF,-
                          03A8    725                 L^FMT_INVALID              ;
      000000B9'EF   9F    03A8                        PUSHAB  L^FMT_INVALID
            7E    01    03AE                        movzbl  #DS$K_PRINTF, -(sp)
            7E    1C    03B1                        cvtbl   #DS$K_TYPE_PARAM_ERROR, -(sp)
00000000'GF   03    FB    03B4                        CALLS   #$$N+2, G^DSX$PRINT
              FEFA  31    03BB    726         BRW     RETRY_STRING               ; GO TRY AGAIN
                          03BE    727 140$:
          51  54    D0    03BE    728         MOVL    R4,R1                      ; SAVE "INDEX" FOR 'R1' RETURN
       53   08 AC   D0    03C1    729         MOVL    8(AP),R3                   ; GET DESTINATION ADDRESS
50   00660028 8F    D0    03C5    730         MOVL    #DS$_TRUNCATE,R0           ; ASSUME TRUNCATION ERROR
          52  63    9A    03CC    731         MOVZBL  (R3),R2                    ; GET LENGTH OF STRING
 54   0C AC   01    C3    03CF    732         SUBL3   #1,12(AP),R4               ; LENGTH OF SPACE AVAILABLE FOR TEXT
          54  52    D1    03D4    733         CMPL    R2,R4                      ; DATA VS. SPACE AVAIL
              17    14    03D7    734         BGTR    GETSTRING_X                ; TOO MUCH DATA, EXIT
              09    13    03D9    735         BEQL    160$                       ; FIT EXACTLY NO NEED TO CLEAR, EXIT
              52    D6    03DB    736         INCL    R2                         ; Plus 1 to skip length byte
            6342    94    03DD    737 150$:   CLRB    (R3)[R2]                   ; Clear byte
       F9 52  54    F3    03E0    738         AOBLEQ  R4,R2,150$                 ; CLEAR UNTIL END OF BUFFER
                          03E4    739
          50  01    D0    03E4    740 160$:   MOVL    #SS$_NORMAL,R0             ; INDICATE SUCCESS
              07    11    03E7    741         BRB     GETSTRING_X
                          03E9    742
                          03E9    743 GETSTRING_E:
                          03E9    744         $DS_ABORT                          ; No operator abort the program
00000000'9F   6C    FA    03E9                        CALLG   (AP), a#DS$ABORT
                          03F0    745
                          03F0    746 GETSTRING_X:                               ;
0000FE00'EF   01    0D    FC AD   F0  03F0    747         INSV    -4(FP),#DSA$V_PROMPT, -
                          03FA    748                 #1, L^DSA$GL_FLAGS         ; RESTORE PROMPT FLAG                   [07]
                    04    03FA    749         RET                                ; RETURN
```

```
                        03FB   751          .SBTTL  RESPONSE        PARAMETER RETRIEVAL SUBROUTINE
                        03FB   752 ;++
                        03FB   753 ; FUNCTIONAL DESCRIPTION:
                        03FB   754 ;
                        03FB   755 ;       DISPLAYS A CALLER SPECIFIED PROMPT MESSAGE, DEFAULT AND
                        03FB   756 ;       RANGE VALUES IF REQUESTED BY THE OPERATOR, THEN ACCEPTS
                        03FB   757 ;       AN ASCII RESPONSE AND CONVERTS IT TO BINARY
                        03FB   758 ;
                        03FB   759 ; CALLING SEQUENCE:
                        03FB   760 ;
                        03FB   761 ;       RESPONSE(prompt,radix,low,high,default,flags)
                        03FB   762 ;
                        03FB   763 ; INPUT PARAMETERS:
                        03FB   764 ;
                        03FB   765 ;       4(AP)   Address of prompt string
                        03FB   766 ;       8(AP)   conversion radix
                        03FB   767 ;       12(AP)  low limit argument
                        03FB   768 ;       16(AP)  high limit argument
                        03FB   769 ;       20(AP)  default argument
                        03FB   770 ;       24(AP)  flags argument
                        03FB   771 ;       28(AP)  1 if signed comparisons, 0 if unsigned
                        03FB   772 ;
                        03FB   773 ; IMPLICIT INPUTS:      NONE
                        03FB   774 ;
                        03FB   775 ; OUTPUT PARAMETERS:
                        03FB   776 ;
                        03FB   777 ;       R1 = BINARY PARAMETER   IF SUCCESSFUL
                        03FB   778 ;
                        03FB   779 ; IMPLICIT OUTPUTS:     NONE
                        03FB   780 ;
                        03FB   781 ; COMPLETION CODES:
                        03FB   782 ;
                        03FB   783 ;       SS$_NORMAL
                        03FB   784 ;       DS$_PROGERR
                        03FB   785 ;
                        03FB   786 ; SIDE EFFECTS:          NONE
                        03FB   787 ;--
                        03FB   788 $OFFSET 0,NEGATIVE,<-
                        03FB   789          <OLDPROMPT,4>, -         ; Saved prompt flag
                        03FB   790          <BUFFER,80>, -           ; Buffer for prompt string
                        03FB   791          <DESC,8>, -              ; Descriptor of string
                        03FB   792          <LOCAL,0>>               ; Length of local storage
                        03FB                .SAVE   LOCAL_BLOCK
                        03FB                .PSECT $ABS$ ABS
                00000000 FE70               .=0
                FFFFFFFC 0000               .BLKB   -4
                         FFFC           OLDPROMPT:
                FFFFFFAC FFFC               .BLKB   -80
                         FFAC           BUFFER:
                FFFFFFA4 FFAC               .BLKB   -8
                         FFA4           DESC:
                         FFA4           LOCAL:
                      000003FB              .RESTORE
                        03FB   793 $OFFSET 0,POSITIVE,<-    ; Argument list
                        03FB   794          <CNT,4>,-                ; Argument
                        03FB   795          <PROMPT,4>, -            ; Address of ASCIC prompt string
                        03FB   796          <RADIX,4>, -             ; Radix for conversion
```

```
                     03FB   797          <LOW,4>, -                ; Low limit on response
                     03FB   798          <HIGH,4>, -               ; High limit on response
                     03FB   799          <DEF,4>, -                ; Default value
                     03FB   800          <FLAGS,4>, -              ; Flags, refer to PAR$_DEF
                     03FB   801          <SIGNED,4>>               ; 0 if unsigned, 1 if signed
                     03FB                .SAVE   LOCAL_BLOCK
                     03FB                .PSECT $ABS$ ABS
           00000000  FFFC                .=0
                     0000          CNT:
           00000004  0000                .BLKB   4
                     0004          PROMPT:
           00000008  0004                .BLKB   4
                     0008          RADIX:
           0000000C  0008                .BLKB   4
                     000C          LOW:
           00000010  000C                .BLKB   4
                     0010          HIGH:
           00000014  0010                .BLKB   4
                     0014          DEF:
           00000018  0014                .BLKB   4
                     0018          FLAGS:
           0000001C  0018                .BLKB   4
                     001C          SIGNED:
           00000020  001C                .BLKB   4
                 000003FB                .RESTORE
```

```
                                  03FB   RESPONSE:
                            01FC  03FB           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8>
                                  03FD
                 5E   A4 AD   9E  03FD           MOVAB   LOCAL(FP),SP              ; allocate space for local storage
                                  0401
 FC AD   0000FE00'EF   01   0D   EF  0401           EXTZV   #DSA$V_PROMPT,#1,L^DSA$GL_FLAGS,  -     ;                    [07]
                                  040B                   OLDPROMPT(FP)             ; SAVE STATE OF PROMPT FLAG
                                  040B   RETRY_RESP:
                 5E   A4 AD   9E  040B           MOVAB   LOCAL(FP),SP              ; Restore stack pointer
                                  040F
                 53   0C AC   D0  040F           MOVL    LOW(AP),R3                ; GET LOW LIMIT
              03 18 AC   02   E1  0413           BBC     #PAR$V_ATLO,FLAGS(AP),10$; BRANCH IF DIRECT ADDRESS
                    53   63   D0  0418           MOVL    (R3),R3                   ; GET LOW LIMIT
                                  041B   10$:
                 54   10 AC   D0  041B           MOVL    HIGH(AP),R4               ; GET HIGH LIMIT
              03 18 AC   03   E1  041F           BBC     #PAR$V_ATHI,FLAGS(AP),20$; BRANCH IF DIRECT ADDRESS
                    54   64   D0  0424           MOVL    (R4),R4                   ; GET HIGH LIMIT
                                  0427   20$:
           50   00660020 8F   D0  0427           MOVL    #DS$_PROGERR,R0           ; Prepare for programmer error
                    54   53   D1  042E           CMPL    R3,R4                     ; Compare low and high
                 05 1C AC   E9  0431           BLBC    SIGNED(AP),25$            ; Branch if unsigned comparison
                       08   15  0435           BLEQ    29$                       ; Branch if valid range
                     012F   31  0437           BRW     RESPONSE_X                ; Branch since invalid
                       03   1B  043A   25$:    BLEQU   29$                       ; Branch if valid range
                     012A   31  043C           BRW     RESPONSE_X                ; Branch since invalid
                                  043F
              0C 18 AC   01   E0  043F   29$:    BBS     #PAR$V_NODEF,FLAGS(AP),30$; IF NO DEFAULT SKIP
                 55   14 AC   D0  0444           MOVL    DEF(AP),R5                ; Get Value of default
              03 18 AC   04   E1  0448           BBC     #PAR$V_ATDEF,FLAGS(AP),30$; BRANCH IF DIRECT ADDRESS
                    55   65   D0  044D           MOVL    (R5),R5                   ; Get default value
                                  0450   30$:
                    AC AD   94  0450           CLRB    BUFFER(FP)                ; Set prompt to null
          0000FE00'EF   0D   E3  0453           BBCS    #DSA$V_PROMPT, L^DSA$GL_FLAGS -        ;                    [07]
                       4C         045A                   , 80$                     ; IF SHOULD NOT PROMPT, SKIP IT
                                  045B                                             ; SETS BIT SO ERRORS WILL PRINT RANGE
                                  045B   ;+
                                  045B   ;       R3=LOW LIMIT, R4=HIGH LIMIT, R5=ADDR DEFAULT
                                  045B   ;-
                       50   D4  045B           CLRL    R0                        ; CLEAR
                 08 AC   0A   D1  045D           CMPL    #10,RADIX(AP)             ; DECIMAL RADIX CONVERSION?
                       06   13  0461           BEQL    50$                       ; YES, USE IT
                       02   14  0463           BGTR    40$                       ; MUST BE OCTAL 8 IS LESS THAN 10
                       50   D6  0465           INCL    R0
                       50   D6  0467   40$:    INCL    R0
                                  0469   50$:                                     ; R0=0 DECIMAL, 1=OCTAL, 2=HEX
              03 18 AC   01   E0  0469           BBS     #PAR$V_NODEF,FLAGS(AP),60$; Branch if no default value present
                    50   03   C0  046E           ADDL2   #3,R0                     ; R0=3 DECIMAL, 4=OCTAL, 5=HEX
                                  0471   60$:
                    AD AD   9F  0471           PUSHAB  BUFFER+1(FP)              ; MAKE DESCRIPTOR OF OUTPUT BUFFER
                 7E   4F 8F   9A  0474           MOVZBL  #79,-(SP)                 ; LENGTH
                       7E   D4  0478           CLRL    -(SP)                     ; SPACE FOR RETURNED LENGTH
        51   00000000'EF40   D0  047A           MOVL    INDEX_RADIX[R0],R1        ; GET ADDRESS OF PROPER DIRECTIVE
                    50   81   9A  0482           MOVZBL  (R1)+,R0                  ; GET LENGTH
                       03   BB  0485           PUSHR   #^M<R0,R1>                ; MAKE CONTROL STRING DESCRIPTOR
                                  0487
                       18   BB  0487           PUSHR   #^M<R4,R3>                ; LOW VALUE AND HIGH VALUE
                       20   BB  0489           PUSHR   #^MR5                     ; DEFAULT
```

```
                    18 AE   7F   048B    860          PUSHAQ  24(SP)                         ; ADDRESS OF OUTPUT BUFFER DESCRIPTOR
                    18 AE   DF   048E    861          PUSHAL  24(SP)                         ; ADDRESS OF LENGTH LONGWORD
                    14 AE   7F   0491    862          PUSHAQ  20(SP)                         ; ADDRESS OF CONTROL STRING DESCRIPTOR
          00000000'9F   08   FB   0494    863          CALLS   #8,a#SYS$FAO                   ; EDIT AND CLEAN STACK TO RETURNED LEN
                              049B    864
                        04   BA   049B    865          POPR    #^M<R2>                        ; GET LENGTH OF EDITED STRING
                     5E   08   C0   049D    866          ADDL2   #8,SP                          ; REMOVE OUTPUT BUFFER DESCRIPTOR
                     5E   08   C0   04A0    867          ADDL2   #8,SP                          ; REMOVE OUTPUT BUFFER DESCRIPTOR
                  AC AD   52   90   04A3    868          MOVB    R2,BUFFER(FP)                  ; Make ascic info prompt
                              04A7    869 80$:
                     AC AD   9F   04A7    870          PUSHAB  BUFFER(FP)                     ; Stash info prompt address in param list
                              04AA    871
                     04 AC   DD   04AA    872          PUSHL   PROMPT(AP)                     ; User prompt
                              04AD    873
                00000000'8F   DD   04AD    874          PUSHL   #DS$K_STRBUF                   ; Length of buffer for input
                00000000'EF   DF   04B3    875          PUSHAL  L^DS$GL_BUFLEN                 ; Longword to get length of input       [07]
                00000000'EF   9F   04B9    876          PUSHAB  L^DS$GT_STRBUF                 ; Buffer for input line                 [07]
          00000000'EF   05   FB   04BF    877          CALLS   #5,L^DS_GETLINE                ; Get a line of input                   [07]
                        51   D5   04C6    878          TSTL    R1                             ; ANYTHING THERE?
                        44   13   04C8    879          BEQL    DEFAULT                        ; NO, USE DEFAULT
                        38   BB   04CA    880          PUSHR   #^M<R3,R4,R5>                  ; SAVE LOW,HIGH,DEFAULT ADDR
          55    00000000'EF   9E   04CC    881          MOVAB   L^DS$GT_STRBUF,R5              ; ADDRESS OF INPUT                      [07]
          54    00000000'EF   D0   04D3    882          MOVL    L^DS$GL_BUFLEN,R4              ; GET LENGTH OF INPUT STRING            [07]
                     FB23'    30   04DA    883          BSBW    SCAN$SPACES                    ; Remove leading spaces
                53   08 AC   D0   04DD    884          MOVL    RADIX(AP),R3                   ; SET RADIX FOR CONVERSION
                     FB1C'    30   04E1    885          BSBW    SCAN$NUMERIC                   ; CONVERT A NUMBER
                        38   BA   04E4    886          POPR    #^M<R3,R4,R5>                  ; RESTORE ARGS
                        53   13   04E6    887          BEQL    CVTERR                         ; CONVERSION ERROR
                              04E8    888
                     51   50   D0   04E8    889          MOVL    R0,R1                          ; COPY VALUE
          52    000000EB'EF   9E   04EB    890          MOVAB   L^FMT_RANGE,R2                 ; PREPARE FOR RANGE ERROR               [07]
                     53   51   D1   04F2    891          CMPL    R1,R3                          ; CHECK W/ LOW LIMIT
                  04 1C AC   E9   04F5    892          BLBC    SIGNED(AP),90$                 ; Branch if unsigned comparison
                        59   19   04F9    893          BLSS    ERR_TEXT                       ; TYPE TEXT AND RETRY
                        02   11   04FB    894          BRB     100$                           ; Branch to upper limit check
                        55   1F   04FD    895 90$:     BLSSU   ERR_TEXT                       ; TYPE TEXT AND RETRY
                     54   51   D1   04FF    896 100$:    CMPL    R1,R4                          ; CHECK W/ LOW LIMIT
                  04 1C AC   E9   0502    897          BLBC    SIGNED(AP),110$                ; Branch if unsigned comparison
                        4C   14   0506    898          BGTR    ERR_TEXT                       ; TYPE TEXT AND RETRY
                        5C   11   0508    899          BRB     RESPONSE_OK                    ; EXIT
                        48   1A   050A    900 110$:    BGTRU   ERR_TEXT                       ; Type text and retry
                        58   11   050C    901          BRB     RESPONSE_OK                    ; Exit
                              050E    902
                              050E    903 DEFAULT:
                     51   55   D0   050E    904          MOVL    R5,R1                          ; ASSUME DEFAULT
                  18 AC   01   E1   0511    905          BBC     #PAR$V_NODEF, FLAGS(AP) -
                        50         0515    906                  , RESPONSE_OK                  ; IF DEFAULT ALLOWED, EXIT
                              0516    907
                              0516    908          $PRINT  #DS$K_TYPE_PARAM_ERROR,-       ; TEXT OF MESSAGE                       [10]
                              0516    909                  #DS$K_PRINTF,-
                              0516    910                  FMT_NODEFAULT                  ;
                0000008D'EF   9F   0516                    PUSHAB  FMT_NODEFAULT
                     7E   01   9A   051C                    movzbl  #DS$K_PRINTF, -(sp)
                     7E   1C   98   051F                    cvtbl   #DS$K_TYPE_PARAM_ERROR, -(sp)
          00000000'GF   03   FB   0522                    CALLS   #$$N+2, G^DSX$PRINT
          0000FE00'EF   0C   E1   0529    911          BBC     #DSA$V_OPER,L^DSA$GL_FLAGS -   ;                               [07]
                        03         0530    912                  , 10$                          ; IF NO OPERATOR PRESENT...ABORT
```

```
                FED7   31  0531   913          BRW     RETRY_RESP                ; TRY AGAIN
                       0534   914 10$:     $DS_ABORT
00000000'9F      6C    FA  0534                CALLG   (AP), a#DS$ABORT
                       053B   915
                       053B   916 CVTERR:
52   00000109'EF   9E  053B   917          MOVAB   FMT_ILLCHAR,R2            ; ASSUME ERROR WAS ILLEGAL CHARACTER
00660018 8F      50   D1  0542   918          CMPL    R0,#DS$_ILLCHAR           ; WAS IT?
                 09   13  0549   919          BEQL    ERR_TEXT                  ; TYPE ERROR AND TRY AGAIN
52   000000EB'EF   9E  054B   920          MOVAB   FMT_OVERFLOW,R2           ; ASSUME WAS OVERFLOW
                 00   11  0552   921          BRB     ERR_TEXT                  ; TYPE ERROR AND TRY AGAIN
                       0554   922 ERR_TEXT:
                       0554   923          $PRINT  #DS$K_TYPE_PARAM_ERROR,-          ; TYPE ERROR TEXT                [10]
                       0554   924                  #DS$K_PRINTF,-
                       0554   925                  (R2)            ;
                 62    9F  0554                PUSHAB  (R2)
              7E   01    9A  0556                movzbl  #DS$K_PRINTF, -(sp)
              7E   1C    98  0559                cvtbl   #DS$K_TYPE_PARAM_ERROR, -(sp)
00000000'GF      03    FB  055C                CALLS   #$$N+2, G^DSX$PRINT
                FEA5   31  0563   926          BRW     RETRY_RESP                ; AND TRY AGAIN
                       0566   927
                       0566   928 RESPONSE_OK:
              50   01    D0  0566   929          MOVL    S^#SS$_NORMAL,R0          ; It worked.
                       0569   930 RESPONSE_X:
       0D   FC AD    F0  0569   931          INSV    OLDPROMPT(FP), #DSA$V_PROMPT -
0000FE00'EF      01        056D   932                  ,#1, L^DSA$GL_FLAGS       ; RESTORE PROMPT FLAG            [07]
                 04   11  0573   933          RET                               ; RETURN
                       0574   934
                       0574   935          .END
```

```
$$N                             = 00000001     D
BIT...                          = 00000004     D
BUFFER                            FFFFFFAC     D
CNT                               00000000     D
CVTERR                            0000053B R   D   04
DEF                               00000014     D
DEFAULT                           0000050E R   D   04
DESC                              FFFFFFA4     D
DIR...                          = 00000001     D
DS$ABORT                          ********     X   00
DS$GK_SID                         ********     X   00
DS$GL_BUFLEN                      ********     X   00
DS$GT_STRBUF                      ********     X   00
DS$K_BBC                        = 0000001D   G D
DS$K_BBCC                       = 00000021   G D
DS$K_BBCCI                      = 00000023   G D
DS$K_BBCS                       = 0000001F   G D
DS$K_BBS                        = 0000001C   G D
DS$K_BBSC                       = 00000020   G D
DS$K_BBSS                       = 0000001E   G D
DS$K_BBSSI                      = 00000022   G D
DS$K_BCC_M                      = 0000001B   G D
DS$K_BCS_M                      = 0000001A   G D
DS$K_BEQLU_B                    = 00000005   G D
DS$K_BEQLU_M                    = 00000011   G D
DS$K_BEQL_B                     = 00000004   G D
DS$K_BEQL_M                     = 00000010   G D
DS$K_BERROR                     = 00000026   G D
DS$K_BGEQU_B                    = 00000007   G D
DS$K_BGEQU_M                    = 00000013   G D
DS$K_BGEQ_B                     = 00000006   G D
DS$K_BGEQ_M                     = 00000012   G D
DS$K_BGTRU_B                    = 00000009   G D
DS$K_BGTRU_M                    = 00000015   G D
DS$K_BGTR_B                     = 00000008   G D
DS$K_BGTR_M                     = 00000014   G D
DS$K_BLBC                       = 00000025   G D
DS$K_BLBS                       = 00000024   G D
DS$K_BLEQU_B                    = 00000003   G D
DS$K_BLEQU_M                    = 0000000F   G D
DS$K_BLEQ_B                     = 00000002   G D
DS$K_BLEQ_M                     = 0000000E   G D
DS$K_BLSSU_B                    = 00000001   G D
DS$K_BLSSU_M                    = 0000000D   G D
DS$K_BLSS_B                     = 00000000   G D
DS$K_BLSS_M                     = 0000000C   G D
DS$K_BNEQU_B                    = 0000000B   G D
DS$K_BNEQU_M                    = 00000017   G D
DS$K_BNEQ_B                     = 0000000A   G D
DS$K_BNEQ_M                     = 00000016   G D
DS$K_BNERROR                    = 00000027   G D
DS$K_BVC_M                      = 00000019   G D
DS$K_BVS_M                      = 00000018   G D
DS$K_DS_STARTING_LOCATION       = 00010000     D
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_STRBUF                       ********     X   00
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
DS$K_TYPE_ERRDEV                = 00000008     D
DS$K_TYPE_ERRHARD               = 00000006     D
DS$K_TYPE_ERROR_BODY            = 00000009     D
DS$K_TYPE_ERROR_END             = 0000000A     D
DS$K_TYPE_ERRPREP               = 0000001B     D
DS$K_TYPE_ERRSOFT               = 00000007     D
DS$K_TYPE_ERRSUP                = 00000004     D
DS$K_TYPE_ERRSYS                = 00000005     D
DS$K_TYPE_ERR_HALT              = 0000000D     D
DS$K_TYPE_EXCEPTION             = 0000000C     D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B     D
DS$K_TYPE_FIRST_PASS            = 00000011     D
DS$K_TYPE_GENERAL               = 00000000     D
DS$K_TYPE_GENERAL_ERROR         = 00000003     D
DS$K_TYPE_NO_TESTS              = 00000012     D
DS$K_TYPE_PARAM_ERROR           = 0000001C     D
DS$K_TYPE_PROGRAM_END           = 00000010     D
DS$K_TYPE_PROGRAM_INFO          = 00000017     D
DS$K_TYPE_PROGRAM_START         = 0000000F     D
DS$K_TYPE_QIO_INVADP            = 00000024     D
DS$K_TYPE_QIO_NODRIVER          = 00000022     D
DS$K_TYPE_QIO_WRONGVER          = 00000023     D
DS$K_TYPE_SCRIPT_ECHO           = 00000021     D
DS$K_TYPE_SCRIPT_PNF            = 0000001E     D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020     D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F     D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019     D
DS$K_TYPE_START_ERR             = 00000018     D
DS$K_TYPE_START_LIST            = 00000025     D
DS$K_TYPE_SUMMARY               = 0000000E     D
DS$K_TYPE_USER_PROMPT           = 00000002     D
DS$K_WARNING                    = 00000000     D
DS$_AP_NORMAL_BREAK             = 00660150     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
DS$_CHMK                        = 006600E0     D
DS$_DEVNAME                     = 00660108     D
DS$_ERROR                       = 00660002     D
DS$_FHWE                        = 00660068     D
DS$_FRAGBUF                     = 00660080     D
```

```
DS$_ICBUSY                      = 006600C8     D
DS$_ICERR                       = 006600C0     D
DS$_IHWE                        = 00660060     D
DS$_ILLCHAR                     = 00660018     D
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INIT_FAIL                   = 00660140     D
DS$_INSFMEM                     = 00660050     D
DS$_INVCPU                      = 00660130     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MEM_ALLOC_ERR               = 00660138     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTALLOWMP                  = 00660148     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_UNSUPPDEV                   = 00660158     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$V_ASK_PROMPT                = 00000013     D
DSA$V_OPER                      = 0000000C     D
DSA$V_PROMPT                    = 0000000D     D
DSQA$K_DISPAT_AFTER_INIT        = 00000014     D
DSQA$K_DISPAT_BEFORE_INIT       = 00000013     D
DSQA$K_DISPAT_CALLTEST          = 00000015     D
DSQA$K_DISPAT_DONE_TESTS        = 00000018     D
DSQA$K_DISPAT_DSX$ENDPASS_BGN   = 00000001     D
DSQA$K_DISPAT_DSX$ENDPASS_END   = 00000002     D
DSQA$K_DISPAT_DSX$ENDPASS_MID   = 00000000     D
DSQA$K_DISPAT_DS_CLEANUP_BGN    = 00000003     D
DSQA$K_DISPAT_DS_CLEANUP_END    = 00000004     D
DSQA$K_DUMMY_1                  = 00000007     D
DSQA$K_ERROR_ERROR_BGN          = 00000006     D
DSQA$K_ERROR_ERROR_END          = 00000005     D
DSQA$K_KERNEL_INIT_CONTEXT      = 00000008     D
DSQA$K_LOOP_DSX$BGNSUB          = 00000009     D
DSQA$K_LOOP_DSX$CKLOOP          = 0000000B     D
DSQA$K_LOOP_DSX$ENDSUB          = 0000000A     D
DSQA$K_PARAM_DSX$GETADDRESS     = 0000000E     D
DSQA$K_PARAM_DSX$GETDATA        = 0000000C     D
DSQA$K_PARAM_DSX$GETLOGICAL     = 0000000F     D
DSQA$K_PARAM_DSX$GETSTRING      = 00000010     D
DSQA$K_PARAM_DSX$GETVIELD       = 0000000D     D
DSQA$K_QA_DSX$BRANCH            = 00000011     D
DSQA$K_START_BEFORE_HEADER      = 00000017     D
DSQA$K_START_VRRESTART          = 00000012     D
DSQA$K_START_VRSTART            = 00000016     D
DSX$GETADDRESS                    000000F6 RG  D  04
DSX$GETDATA                       00000000 RG  D  04
DSX$GETLOGICAL                    0000014F RG  D  04
DSX$GETSTRING                     0000026F RG  D  04
DSX$GETVIELD                      00000085 RG  D  04
DSX$PRINT                         ******** **  X  00
DS_GETLINE                        ******** **  X  00
ERR_TEXT                          00000554 R   D  04
FALSE                           = 00000000     D
FLAGS                             00000018     D
FMT_DEFNO                         0000012E R   D  03
FMT_DEFYES                        00000120 R   D  03
FMT_ILLCHAR                       00000109 R   D  03
FMT_INVALID                       000000B9 R   D  03
FMT_NODEFAULT                     0000008D R   D  03
FMT_NULL                          0000013C R   D  03
FMT_OVERFLOW                      000000EB R   D  03
FMT_PROGERR                       000000CF R   D  03
FMT_RANGE                         000000EB R   D  03
GETADDRESS_X                      0000014E R   D  04
GETDATA_X                         00000084 R   D  04
GETLOGICAL_X                      00000264 R   D  04
GETSTRING_E                       000003E9 R   D  04
GETSTRING_X                       000003F0 R   D  04
GETVIELD_X                        000000F5 R   D  04
HIGH                              00000010     D
INDEX_RADIX                       00000000 R   D  03
LOCAL                             FFFFFFA4     D
LOW                               0000000C     D
OFF                             = 00000000     D
OLDPROMPT                         FFFFFFFC     D
ON                              = 00000001     D
PAR$M_ATDEF                     = 00000010     D
```

```
PAR$M_ATHI                      = 00000008     D
PAR$M_ATLO                      = 00000004     D
PAR$M_NODEF                     = 00000002     D
PAR$V_ATDEF                     = 00000004     D
PAR$V_ATHI                      = 00000003     D
PAR$V_ATLO                      = 00000002     D
PAR$V_NODEF                     = 00000001     D
PAR$_BIN                        = 00000002     D
PAR$_DEC                        = 0000000A     D
PAR$_HEX                        = 00000010     D
PAR$_NO                         = 00000000     D
PAR$_OCT                        = 00000008     D
PAR$_YES                        = 00000001     D
PRNT_PRMT                         00000000 RG  D   02
PROMPT                            00000004     D
QA$AOB_FLAGS                      ********   X   00
QA$K_ABORT_NOW                  = 00000007     D
QA$K_APT_PHASE_ONE              = 0000000B     D
QA$K_APT_PHASE_TWO              = 0000000C     D
QA$K_BAD_ADDRESS                = 00000009     D
QA$K_CONTROL_C_CONT             = 0000000A     D
QA$K_DEALLOCATION               = 0000000E     D
QA$K_ERROR_PHASE_ONE            = 00000005     D
QA$K_ERROR_PHASE_THREE          = 00000007     D
QA$K_ERROR_PHASE_TWO            = 00000006     D
QA$K_FAILURE                    = 00000000     D
QA$K_FIRST_BRANCH_CODE          = 00000000     D
QA$K_FIRST_TIME                 = 00000005     D
QA$K_INIT_CONTEXT_DONE          = 00000003     D
QA$K_LAST_BRANCH_CODE           = 00000025     D
QA$K_LOOP_ON_SUBTEST            = 00000003     D
QA$K_LOOP_ON_TEST               = 00000002     D
QA$K_LOOP_TEST_DONE             = 00000004     D
QA$K_MEMORY_MANAGE              = 0000000D     D
QA$K_MULTIPLE_PASS              = 00000001     D
QA$K_NODEF                      = 00000000     D
QA$K_NORMAL_START               = 00000000     D
QA$K_NO_HEADER                  = 00000006     D
QA$K_NUMBER_OF_CHECKS           = 0000000F     D
QA$K_NUMBER_QA_FLAGS            = 00000008     D
QA$K_NUMB_ROUTINE_STATES        = 00000004     D
QA$K_QA_DEBUG                   = 00000001     D
QA$K_QA_DONE                    = 00000002     D
QA$K_RUN_BACKWARDS              = 00000004     D
QA$K_SECTION                    = 00000008     D
QA$K_SUCCESS                    = 00000001     D
QA$MAIN                           ********   X   00
QA$V_CHECK_DONE                 = 00000003     D
QA$V_DOING_CLEANUP              = 00000002     D
QA$V_ERROR_FOUND                = 00000001     D
QA$V_INIT_DONE                  = 00000000     D
QUESTN                          = 0000003F     D
RADIX                             00000008     D
RESPONSE                          000003FB R   D   04
RESPONSE_OK                       00000566 R   D   04
RESPONSE_X                        00000569 R   D   04
RETRY_LOGICAL                     00000194 R   D   04
RETRY_RESP                        0000040B R   D   04
RETRY_STRING                      000002B8 R   D   04
SAVABS...                       = 00000020     D
SCAN$NUMERIC                      ********   X   00
SCAN$SPACES                       ********   X   00
SIGNED                            0000001C     D
SIZ...                          = 00000001     D
SS$_NORMAL                      = 00000001     D
SYS$FAO                           ********   X   00
TRUE                            = 00000001     D
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000004 (4.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
DATA	0000013D (317.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000574 (1396.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000001	925 (14)	#-536 (10) #-699 (12) #-725 (12) #-910 (14) #-925 (14)
BIT...	=00000004	146 (2)	141 (2) 143 (2) 144 (2) 145 (2) 146 (2)
BUFFER	FFFFFFAC	792 (13)	#-834 (14) 851 (14) #-868 (14) 870 (14)
CNT	00000000	801 (13)	
CVTERR	0000053B-R	916 (14)	#-887 (14)
DEF	00000014	801 (13)	#-830 (14)
DEFAULT	0000050E-R	903 (14)	#-879 (14)
DESC	FFFFFFA4	792 (13)	
DIR...	=00000001	801 (13)	792 (13) 801 (13)
DS$ABORT	00000000-XR		129 (2) 744 (12) 914 (14)
DS$GK_SID	00000000-XR		133 (2) #-547 (10)
DS$GL_BUFLEN	00000000-XR		131 (2) 519 (10) #-524 (10) 686 (12) #-691 (12) 875 (14) #-882 (14)
DS$GT_STRBUF	00000000-XR		131 (2) 520 (10) #-528 (10) #-531 (10) 876 (14) 881 (14)
DS$K_BBC	=0000001D	145 (2)	
DS$K_BBCC	=00000021	145 (2)	
DS$K_BBCCI	=00000023	145 (2)	
DS$K_BBCS	=0000001F	145 (2)	
DS$K_BBS	=0000001C	145 (2)	
DS$K_BBSC	=00000020	145 (2)	
DS$K_BBSS	=0000001E	145 (2)	
DS$K_BBSSI	=00000022	145 (2)	
DS$K_BCC_M	=0000001B	145 (2)	
DS$K_BCS_M	=0000001A	145 (2)	
DS$K_BEQLU_B	=00000005	145 (2)	
DS$K_BEQLU_M	=00000011	145 (2)	
DS$K_BEQL_B	=00000004	145 (2)	
DS$K_BEQL_M	=00000010	145 (2)	
DS$K_BERROR	=00000026	145 (2)	
DS$K_BGEQU_B	=00000007	145 (2)	
DS$K_BGEQU_M	=00000013	145 (2)	
DS$K_BGEQ_B	=00000006	145 (2)	
DS$K_BGEQ_M	=00000012	145 (2)	
DS$K_BGTRU_B	=00000009	145 (2)	
DS$K_BGTRU_M	=00000015	145 (2)	
DS$K_BGTR_B	=00000008	145 (2)	
DS$K_BGTR_M	=00000014	145 (2)	
DS$K_BLBC	=00000025	145 (2)	145 (2)
DS$K_BLBS	=00000024	145 (2)	
DS$K_BLEQU_B	=00000003	145 (2)	
DS$K_BLEQU_M	=0000000F	145 (2)	
DS$K_BLEQ_B	=00000002	145 (2)	
DS$K_BLEQ_M	=0000000E	145 (2)	
DS$K_BLSSU_B	=00000001	145 (2)	
DS$K_BLSSU_M	=0000000D	145 (2)	
DS$K_BLSS_B	=00000000	145 (2)	145 (2)
DS$K_BLSS_M	=0000000C	145 (2)	

```
DS$K_BNEQU_B                    =0000000B     145     (2)
DS$K_BNEQU_M                    =00000017     145     (2)
DS$K_BNEQ_B                     =0000000A     145     (2)
DS$K_BNEQ_M                     =00000016     145     (2)
DS$K_BNERROR                    =00000027     145     (2)
DS$K_BVC_M                      =00000019     145     (2)
DS$K_BVS_M                      =00000018     145     (2)
DS$K_DS_STARTING_LOCATION       =00010000     145     (2)
DS$K_ERROR                      =00000002     141     (2)
DS$K_NORMAL                     =00000001     141     (2)
DS$K_PRINTB                     =00000002     146     (2)
DS$K_PRINTF                     =00000001     146     (2)   #-536    (10)   #-699    (12)   #-725    (12)   #-910    (14)
                                                            #-925    (14)
DS$K_PRINTI                     =00000000     146     (2)
DS$K_PRINTX                     =00000003     146     (2)
DS$K_SEVERE                     =00000004     141     (2)
DS$K_STRBUF                      00000000-XR                  131     (2)    #-874    (14)
DS$K_SUBSYS                     =00000066     141     (2)   141     (2)
DS$K_TYPE_ABORT_PROGRAM         =00000014     146     (2)
DS$K_TYPE_ABORT_TEST            =00000013     146     (2)
DS$K_TYPE_COMMAND_ERR           =00000015     146     (2)
DS$K_TYPE_COMMAND_OUT           =00000016     146     (2)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     146     (2)
DS$K_TYPE_DS_PROMPT             =00000001     146     (2)
DS$K_TYPE_DS_START              =0000001D     146     (2)
DS$K_TYPE_ERRDEV                =00000008     146     (2)
DS$K_TYPE_ERRHARD               =00000006     146     (2)
DS$K_TYPE_ERROR_BODY            =00000009     146     (2)
DS$K_TYPE_ERROR_END             =0000000A     146     (2)
DS$K_TYPE_ERRPREP               =0000001B     146     (2)
DS$K_TYPE_ERRSOFT               =00000007     146     (2)
DS$K_TYPE_ERRSUP                =00000004     146     (2)
DS$K_TYPE_ERRSYS                =00000005     146     (2)
DS$K_TYPE_ERR_HALT              =0000000D     146     (2)
DS$K_TYPE_EXCEPTION             =0000000C     146     (2)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     146     (2)
DS$K_TYPE_FIRST_PASS            =00000011     146     (2)
DS$K_TYPE_GENERAL               =00000000     146     (2)
DS$K_TYPE_GENERAL_ERROR         =00000003     146     (2)
DS$K_TYPE_NO_TESTS              =00000012     146     (2)
DS$K_TYPE_PARAM_ERROR           =0000001C     146     (2)   #-536    (10)   #-699    (12)   #-725    (12)   #-910    (14)
                                                            #-925    (14)
DS$K_TYPE_PROGRAM_END           =00000010     146     (2)
DS$K_TYPE_PROGRAM_INFO          =00000017     146     (2)
DS$K_TYPE_PROGRAM_START         =0000000F     146     (2)
DS$K_TYPE_QIO_INVADP            =00000024     146     (2)
DS$K_TYPE_QIO_NODRIVER          =00000022     146     (2)
DS$K_TYPE_QIO_WRONGVER          =00000023     146     (2)
DS$K_TYPE_SCRIPT_ECHO           =00000021     146     (2)
DS$K_TYPE_SCRIPT_PNF            =0000001E     146     (2)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     146     (2)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     146     (2)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     146     (2)
DS$K_TYPE_START_ERR             =00000018     146     (2)
DS$K_TYPE_START_LIST            =00000025     146     (2)
DS$K_TYPE_SUMMARY               =0000000E     146     (2)
DS$K_TYPE_USER_PROMPT           =00000002     146     (2)
```

```
DS$K_WARNING                    =00000000      141     (2)
DS$_AP_NORMAL_BREAK             =00660150      141     (2)
DS$_ARITH                       =006600D0      141     (2)
DS$_ASBE                        =00660118      141     (2)
DS$_BADLINK                     =006600F0      141     (2)
DS$_BADTYPE                     =006600E8      141     (2)
DS$_BIIC                        =00660120      141     (2)
DS$_CHME                        =006600A8      141     (2)
DS$_CHMK                        =006600E0      141     (2)
DS$_DEVNAME                     =00660108      141     (2)
DS$_ERROR                       =00660002      141     (2)
DS$_FHWE                        =00660068      141     (2)
DS$_FRAGBUF                     =00660080      141     (2)
DS$_ICBUSY                      =006600C8      141     (2)
DS$_ICERR                       =006600C0      141     (2)
DS$_IHWE                        =00660060      141     (2)
DS$_ILLCHAR                     =00660018      141     (2)     #-918     (14)
DS$_ILLPAGCNT                   =00660078      141     (2)
DS$_ILLUNIT                     =00660100      141     (2)
DS$_INIT_FAIL                   =00660140      141     (2)
DS$_INSFMEM                     =00660050      141     (2)
DS$_INVCPU                      =00660130      141     (2)
DS$_IPL2HI                      =006600B8      141     (2)
DS$_IVADDR                      =00660040      141     (2)
DS$_IVVECT                      =00660038      141     (2)
DS$_KRNLSTK                     =00660090      141     (2)
DS$_LOGIC                       =00660070      141     (2)
DS$_MCHK                        =00660088      141     (2)
DS$_MEM_ALLOC_ERR               =00660138      141     (2)
DS$_MMOFF                       =00660058      141     (2)
DS$_NEEDUNIT                    =006600F8      141     (2)
DS$_NODE                        =00660128      141     (2)
DS$_NOPCS                       =00660110      141     (2)
DS$_NORMAL                      =00660001      141     (2)
DS$_NOSUPPORT                   =006600B1      141     (2)
DS$_NOTALLOWMP                  =00660148      141     (2)
DS$_NOTDON                      =00660030      141     (2)
DS$_NOTIMP                      =006600B0      141     (2)       141     (2)
DS$_NULLSTR                     =00660010      141     (2)
DS$_OVERFLOW                    =00660008      141     (2)
DS$_POWER                       =00660098      141     (2)
DS$_PROGERR                     =00660020      141     (2)     #-249     (4)     #-342     (6)     #-425     (8)     #-500     (10)
                                                               #-506     (10)    #-635     (12)    #-821     (14)
DS$_SEVERE                      =00660004      141     (2)
DS$_TRANSL                      =006600A0      141     (2)
DS$_TRUNCATE                    =00660028      141     (2)     #-276     (4)     #-361     (6)     #-730     (12)
DS$_UNEXPINT                    =006600D8      141     (2)
DS$_UNSUPPDEV                   =00660158      141     (2)
DS$_VASFULL                     =00660048      141     (2)
DS$_WARNING                     =00660000      141     (2)       141     (2)
DSA$GL_FLAGS                     0000FE00                        240     (4)       333     (6)       416     (8)       490     (10)
                                                                 499     (10)      511     (10)      562     (10)      616     (12)
                                                                 632     (12)      646     (12)      700     (12)      748     (12)
                                                                 808     (14)      835     (14)      911     (14)      932     (14)
DSA$V_ASK_PROMPT                =00000013                       #-239     (4)     #-332     (6)     #-415     (8)     #-489     (10)
                                                               #-615     (12)
DSA$V_OPER                      =0000000C                      #-700     (12)    #-911     (14)
```

```
DSA$V_PROMPT                    =0000000D                       #-498   (10)   #-511   (10)   #-561   (10)   #-631   (12)
                                                                #-646   (12)   #-747   (12)   #-808   (14)   #-835   (14)
                                                                #-931   (14)
DSQA$K_DISPAT_AFTER_INIT        =00000014       144     (2)
DSQA$K_DISPAT_BEFORE_INIT       =00000013       144     (2)
DSQA$K_DISPAT_CALLTEST          =00000015       144     (2)
DSQA$K_DISPAT_DONE_TESTS        =00000018       144     (2)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001       144     (2)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002       144     (2)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000       144     (2)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003       144     (2)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004       144     (2)
DSQA$K_DUMMY_1                  =00000007       144     (2)
DSQA$K_ERROR_ERROR_BGN          =00000006       144     (2)
DSQA$K_ERROR_ERROR_END          =00000005       144     (2)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008       144     (2)
DSQA$K_LOOP_DSX$BGNSUB          =00000009       144     (2)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B       144     (2)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A       144     (2)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E       144     (2)     #-423   (8)
DSQA$K_PARAM_DSX$GETDATA        =0000000C       144     (2)     #-247   (4)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F       144     (2)     #-496   (10)
DSQA$K_PARAM_DSX$GETSTRING      =00000010       144     (2)     #-629   (12)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D       144     (2)     #-340   (6)
DSQA$K_QA_DSX$BRANCH            =00000011       144     (2)
DSQA$K_START_BEFORE_HEADER      =00000017       144     (2)
DSQA$K_START_VRRESTART          =00000012       144     (2)
DSQA$K_START_VRSTART            =00000016       144     (2)
DSX$GETADDRESS                   000000F6-R     413     (8)
DSX$GETDATA                      00000000-R     237     (4)
DSX$GETLOGICAL                   0000014F-R     487     (10)
DSX$GETSTRING                    0000026F-R     613     (12)
DSX$GETVIELD                     00000085-R     330     (6)
DSX$PRINT                        00000000-XR                     129     (2)     536     (10)    699     (12)    725     (12)
                                                                910     (14)    925     (14)
DS_GETLINE                       00000000-XR                     132     (2)     521     (10)    689     (12)    877     (14)
ERR_TEXT                         00000554-R     922     (14)    #-893   (14)    #-895   (14)    #-898   (14)    #-900   (14)
                                                                #-919   (14)    #-921   (14)
FALSE                           =00000000       145     (2)
FLAGS                            00000018       801     (13)    814     (14)    818     (14)    829     (14)    831     (14)
                                                                848     (14)    905     (14)
FMT_DEFNO                        0000012E-R     183     (2)     515     (10)
FMT_DEFYES                       00000120-R     181     (2)     513     (10)
FMT_ILLCHAR                      00000109-R     179     (2)     917     (14)
FMT_INVALID                      000000B9-R     172     (2)     536     (10)    725     (12)
FMT_NODEFAULT                    0000008D-R     170     (2)     699     (12)    910     (14)
FMT_NULL                         0000013C-R     185     (2)     510     (10)    644     (12)
FMT_OVERFLOW                     000000EB-R     177     (2)     920     (14)
FMT_PROGERR                      000000CF-R     174     (2)
FMT_RANGE                        000000EB-R     176     (2)     890     (14)
GETADDRESS_X                     0000014E-R     441     (8)     #-436   (8)
GETDATA_X                        00000084-R     278     (4)     #-251   (4)     #-253   (4)     #-262   (4)     #-274   (4)
GETLOGICAL_X                     00000264-R     560     (10)
GETSTRING_E                      000003E9-R     743     (12)    #-701   (12)
GETSTRING_X                      000003F0-R     746     (12)    #-636   (12)    #-734   (12)    #-741   (12)
GETVIELD_X                       000000F5-R     363     (6)     #-344   (6)     #-353   (6)     #-360   (6)     #-427   (8)
HIGH                             00000010       801     (13)    #-817   (14)
```

```
INDEX_RADIX                     00000000-R    160    (2)     #-854    (14)
LOCAL                           FFFFFFA4      792    (13)      806    (14)      811    (14)
LOW                             0000000C      801    (13)    #-813    (14)
OFF                            =00000000      145    (2)
OLDPROMPT                       FFFFFFFC      792    (13)    #-809    (14)    #-931    (14)
ON                             =00000001      145    (2)
PAR$M_ATDEF                    =00000010      143    (2)
PAR$M_ATHI                     =00000008      143    (2)
PAR$M_ATLO                     =00000004      143    (2)
PAR$M_NODEF                    =00000002      143    (2)
PAR$V_ATDEF                    =00000004      143    (2)    #-831    (14)
PAR$V_ATHI                     =00000003      143    (2)    #-818    (14)
PAR$V_ATLO                     =00000002      143    (2)    #-814    (14)
PAR$V_NODEF                    =00000001      143    (2)    #-243    (4)     #-336    (6)     #-419    (8)     #-829    (14)
                                                             #-848    (14)    #-905    (14)
PAR$_BIN                       =00000002      143    (2)
PAR$_DEC                       =0000000A      143    (2)
PAR$_HEX                       =00000010      143    (2)
PAR$_NO                        =00000000      143    (2)    #-527    (10)
PAR$_OCT                       =00000008      143    (2)
PAR$_YES                       =00000001      143    (2)    #-530    (10)
PRNT_PRMT                       00000000-R    155    (2)    #-241    (4)     #-334    (6)     #-417    (8)     #-491    (10)
                                                             #-617    (12)
PROMPT                          00000004      801    (13)   #-872    (14)
QA$AOB_FLAGS                    00000000-XR                    130    (2)      246    (4)      339    (6)      422    (8)
                                                               494    (10)     622    (12)     626    (12)
QA$K_ABORT_NOW                 =00000007      145    (2)
QA$K_APT_PHASE_ONE             =0000000B      145    (2)
QA$K_APT_PHASE_TWO             =0000000C      145    (2)
QA$K_BAD_ADDRESS               =00000009      145    (2)
QA$K_CONTROL_C_CONT            =0000000A      145    (2)
QA$K_DEALLOCATION              =0000000E      145    (2)
QA$K_ERROR_PHASE_ONE           =00000005      145    (2)
QA$K_ERROR_PHASE_THREE         =00000007      145    (2)
QA$K_ERROR_PHASE_TWO           =00000006      145    (2)
QA$K_FAILURE                   =00000000      145    (2)
QA$K_FIRST_BRANCH_CODE         =00000000      145    (2)
QA$K_FIRST_TIME                =00000005      145    (2)
QA$K_INIT_CONTEXT_DONE         =00000003      145    (2)
QA$K_LAST_BRANCH_CODE          =00000025      145    (2)
QA$K_LOOP_ON_SUBTEST           =00000003      145    (2)
QA$K_LOOP_ON_TEST              =00000002      145    (2)
QA$K_LOOP_TEST_DONE            =00000004      145    (2)
QA$K_MEMORY_MANAGE             =0000000D      145    (2)
QA$K_MULTIPLE_PASS             =00000001      145    (2)
QA$K_NODEF                     =00000000      145    (2)    #-245    (4)     #-338    (6)     #-421    (8)     #-493    (10)
                                                             #-621    (12)    #-625    (12)
QA$K_NORMAL_START              =00000000      145    (2)
QA$K_NO_HEADER                 =00000006      145    (2)
QA$K_NUMBER_OF_CHECKS          =0000000F      145    (2)
QA$K_NUMBER_QA_FLAGS           =00000008      145    (2)
QA$K_NUMB_ROUTINE_STATES       =00000004      145    (2)
QA$K_QA_DEBUG                  =00000001      145    (2)
QA$K_QA_DONE                   =00000002      145    (2)
QA$K_RUN_BACKWARDS             =00000004      145    (2)
QA$K_SECTION                   =00000008      145    (2)
QA$K_SUCCESS                   =00000001      145    (2)
```

```
QA$MAIN                      00000000-XR                  130    (2)      247    (4)      340    (6)      423    (8)
                                                          496    (10)     629    (12)
QA$V_CHECK_DONE             =00000003     145    (2)
QA$V_DOING_CLEANUP          =00000002     145    (2)
QA$V_ERROR_FOUND            =00000001     145    (2)
QA$V_INIT_DONE              =00000000     145    (2)
QUESTN                      =0000003F     147    (2)
RADIX                        00000008     801    (13)   #-842    (14)   #-884    (14)
RESPONSE                     000003FB-R   803    (14)     261    (4)      352    (6)      435    (8)
RESPONSE_OK                  00000566-R   928    (14)   #-899    (14)   #-901    (14)   #-906    (14)
RESPONSE_X                   00000569-R   930    (14)   #-825    (14)   #-827    (14)
RETRY_LOGICAL                00000194-R   509    (10)   #-504    (10)   #-537    (10)
RETRY_RESP                   0000040B-R   810    (14)     913    (14)   #-926    (14)
RETRY_STRING                 000002B8-R   638    (12)   #-634    (12)   #-702    (12)   #-726    (12)
SAVABS...                   =00000020     801    (13)
SCAN$NUMERIC                 00000000-XR                  132    (2)    #-885    (14)
SCAN$SPACES                  00000000-XR                  132    (2)    #-883    (14)
SIGNED                       0000001C     801    (13)   #-823    (14)   #-892    (14)   #-897    (14)
SIZ...                      =00000001     143    (2)      143    (2)
SS$_NORMAL                  =00000001     140    (2)    #-272    (4)    #-358    (6)    #-439    (8)    #-558    (10)
                                                       #-740    (12)   #-929    (14)
SYS$FAO                      00000000-XR                  132    (2)      863    (14)
TRUE                        =00000001     145    (2)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO            SIZE         DEFINITION       REFERENCES...
-----            ----         ----------       -------------
$D1_ABPROGRAM     1            744    (12)      744    (12)      914    (14)
$D1_PRINT_S       1            536    (10)      536    (10)      699    (12)      725    (12)      910    (14)      925    (14)
$DEF              1            146    (2)
$DEFINI           1            142    (2)       142    (2)
$DS_ABORT         1            744    (12)      744    (12)      914    (14)
$DS_DSADEF        5            142    (2)       142    (2)
$DS_DSDEF         3            141    (2)       141    (2)
$DS_PARDEF        1            143    (2)       143    (2)
$DS_QADEF         2            145    (2)       145    (2)
$DS_TYPEDEF       4            146    (2)       146    (2)
$EQU              1            146    (2)       141    (2)       143    (2)       144    (2)       145    (2)       146    (2)
$EQULS1           1            146    (2)       141    (2)       144    (2)       145    (2)       146    (2)
$EQULST           1            141    (2)       141    (2)       144    (2)       145    (2)       146    (2)
$GBLINI           2            141    (2)       141    (2)       143    (2)       144    (2)       145    (2)       146    (2)
$OFFSET           2            788    (13)      788    (13)      793    (13)
$OFFST1           1            792    (13)      792    (13)      801    (13)
$PRINT            2            534    (10)      534    (10)      697    (12)      723    (12)      908    (14)      923    (14)
$VIELD            1                             143    (2)
$VIELD1           1            146    (2)       143    (2)
BR_IF_MP          1            547    (10)      547    (10)
DSQA              2            144    (2)       144    (2)
DS_QADEFS         6            145    (2)       145    (2)
QA_MAIN           1            247    (4)       247    (4)       340    (6)       423    (8)       496    (10)      629    (12)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL2	00C0	849 (14) 866 (14) 867 (14)
AOBLEQ	00F3	738 (12)
BBC	00E1	239 (4) 332 (6) 415 (8) 489 (10) 615 (12) 700 (12) 814 (14) 818 (14) 831 (14) 905 (14) 911 (14)
BBCC	00E5	493 (10) 625 (12)
BBCS	00E3	511 (10) 621 (12) 646 (12) 835 (14)
BBS	00E0	829 (14) 848 (14)
BEQL	0013	253 (4) 274 (4) 360 (6) 525 (10) 529 (10) 532 (10) 545 (10) 547 (10) 634 (12) 649 (12) 657 (12) 659 (12) 661 (12) 669 (12) 671 (12) 711 (12) 713 (12) 735 (12) 843 (14) 879 (14) 887 (14) 919 (14)
BGTR	0014	734 (12) 844 (14) 898 (14)
BGTRU	001A	900 (14)
BLBC	00E9	262 (4) 353 (6) 436 (8) 823 (14) 892 (14) 897 (14)
BLBS	00E8	514 (10) 542 (10)
BLEQ	0015	824 (14)
BLEQU	001B	504 (10) 826 (14)
BLSS	0019	893 (14)
BLSSU	001F	895 (14)
BNEQ	0012	251 (4) 344 (6) 427 (8) 502 (10) 620 (12) 692 (12) 695 (12) 718 (12)
BRB	0011	623 (12) 652 (12) 664 (12) 708 (12) 720 (12) 741 (12) 894 (14) 899 (14) 901 (14) 921 (14)
BRW	0031	537 (10) 550 (10) 636 (12) 702 (12) 726 (12) 825 (14) 827 (14) 913 (14) 926 (14)
BSBW	0030	883 (14) 885 (14)
CALLG	00FA	744 (12) 914 (14)
CALLS	00FB	247 (4) 261 (4) 340 (6) 352 (6) 423 (8) 435 (8) 496 (10) 521 (10) 536 (10) 629 (12) 689 (12) 699 (12) 725 (12) 863 (14) 877 (14) 910 (14) 925 (14)
CLRB	0094	737 (12) 834 (14)
CLRL	00D4	707 (12) 841 (14) 853 (14)
CMPB	0091	528 (10) 531 (10) 717 (12)
CMPL	00D1	250 (4) 273 (4) 343 (6) 359 (6) 426 (8) 501 (10) 503 (10) 547 (10) 633 (12) 658 (12) 733 (12) 822 (14) 842 (14) 891 (14) 896 (14) 918 (14)
CVTBL	0098	536 (10) 699 (12) 725 (12) 910 (14) 925 (14)
EXTV	00EE	243 (4) 336 (6) 419 (8)
EXTZV	00EF	270 (4) 356 (6) 498 (10) 631 (12) 808 (14)
FFC	00EB	266 (4)
FFS	00EA	264 (4)
INCL	00D6	736 (12) 845 (14) 846 (14)
INSV	00F0	245 (4) 269 (4) 338 (6) 355 (6) 421 (8) 539 (10) 561 (10) 747 (12) 931 (14)
MOVAB	009E	513 (10) 515 (10) 644 (12) 806 (14) 811 (14) 881 (14) 890 (14) 917 (14) 920 (14)
MOVAL	00DE	639 (12)
MOVB	0090	662 (12) 673 (12) 674 (12) 676 (12) 680 (12) 691 (12) 705 (12) 868 (14)
MOVL	00D0	241 (4) 249 (4) 272 (4) 276 (4) 334 (6) 342 (6) 358 (6)

		361	(6)	417	(8)	425	(8)	438	(8)	439	(8)	491	(10)	500	(10)
		506	(10)	523	(10)	527	(10)	530	(10)	541	(10)	543	(10)	549	(10)
		556	(10)	557	(10)	558	(10)	617	(12)	635	(12)	645	(12)	648	(12)
		650	(12)	656	(12)	668	(12)	681	(12)	687	(12)	694	(12)	710	(12)
		712	(12)	714	(12)	715	(12)	728	(12)	729	(12)	730	(12)	740	(12)
		813	(14)	815	(14)	817	(14)	819	(14)	821	(14)	830	(14)	832	(14)
		854	(14)	882	(14)	884	(14)	889	(14)	904	(14)	929	(14)		
MOVQ	007D	258	(4)	349	(6)	432	(8)								
MOVW	00B0	651	(12)	654	(12)	672	(12)	678	(12)						
MOVZBL	009A	536	(10)	660	(12)	670	(12)	699	(12)	704	(12)	716	(12)	725	(12)
		731	(12)	852	(14)	855	(14)	910	(14)	925	(14)				
POPR	00BA	865	(14)	886	(14)										
PUSHAB	009F	510	(10)	520	(10)	536	(10)	688	(12)	699	(12)	725	(12)	851	(14)
		870	(14)	876	(14)	910	(14)	925	(14)						
PUSHAL	00DF	519	(10)	686	(12)	861	(14)	875	(14)						
PUSHAQ	007F	860	(14)	862	(14)										
PUSHL	00DD	247	(4)	255	(4)	256	(4)	257	(4)	259	(4)	260	(4)	340	(6)
		346	(6)	347	(6)	348	(6)	350	(6)	351	(6)	423	(8)	429	(8)
		430	(8)	431	(8)	433	(8)	434	(8)	496	(10)	517	(10)	518	(10)
		629	(12)	683	(12)	684	(12)	872	(14)	874	(14)				
PUSHR	00BB	856	(14)	858	(14)	859	(14)	880	(14)						
RET	0004	279	(4)	364	(6)	442	(8)	507	(10)	563	(10)	749	(12)	933	(14)
SOBGEQ	00F4	706	(12)	719	(12)										
SOBGTR	00F5	663	(12)	666	(12)	675	(12)	721	(12)						
SUBL	00C2	267	(4)												
SUBL2	00C2	679	(12)												
SUBL3	00C3	265	(4)	685	(12)	732	(12)								
TSTL	00D5	252	(4)	524	(10)	544	(10)	619	(12)	878	(14)				
TSTW	00B5	665	(12)												

Directives Cross Reference

DIRECTIVE	REFERENCES...
.ASCIC	162 (2) 163 (2) 164 (2) 165 (2) 166 (2) 167 (2) 171 (2) 173 (2) 175 (2) 178 (2) 180 (2) 182 (2) 184 (2) 186 (2)
.BLKB	792 (13) 801 (13)
.DSABL	11 (1)
.END	935 (14)
.ENDC	141 (2) 143 (2) 144 (2) 145 (2) 146 (2) 247 (4) 340 (6) 423 (8) 496 (10) 536 (10) 629 (12) 699 (12) 725 (12) 792 (13) 801 (13) 910 (14) 925 (14)
.ENDM	141 (2) 142 (2) 143 (2) 144 (2) 145 (2) 146 (2) 247 (4) 534 (10) 536 (10) 547 (10) 744 (12) 788 (13) 792 (13)
.ENDR	141 (2) 143 (2) 144 (2) 145 (2) 146 (2) 536 (10) 699 (12) 725 (12) 792 (13) 801 (13) 910 (14) 925 (14)
.ENTRY	237 (4) 330 (6) 413 (8) 487 (10) 613 (12)
.EXTRN	129 (2) 130 (2) 131 (2) 132 (2) 133 (2)
.IDENT	8 (1)
.IF	141 (2) 143 (2) 144 (2) 145 (2) 146 (2) 247 (4) 340 (6) 423 (8) 496 (10) 536 (10) 629 (12) 699 (12) 725 (12) 792 (13) 801 (13) 910 (14) 925 (14)
.IFF	141 (2) 143 (2) 144 (2) 145 (2) 146 (2) 792 (13) 801 (13)
.IF_FALSE	247 (4) 340 (6) 423 (8) 496 (10) 629 (12)
.IIF	141 (2) 143 (2) 144 (2) 145 (2) 146 (2)
.IRP	141 (2) 143 (2) 144 (2) 145 (2) 146 (2) 536 (10) 699 (12) 725 (12) 792 (13) 801 (13) 910 (14) 925 (14)
.LIBRARY	122 (2) 123 (2) 124 (2)
.LIST	142 (2) 792 (13) 801 (13)
.LONG	156 (2) 161 (2)
.MACRO	141 (2) 143 (2) 144 (2) 145 (2) 146 (2)
.MDELETE	141 (2) 143 (2) 145 (2)
.NLIST	142 (2) 792 (13) 801 (13)
.NOCROSS	142 (2)
.NOSHOW	9 (1)
.PSECT	149 (2) 158 (2) 189 (3) 792 (13) 801 (13)
.RESTORE	142 (2) 792 (13) 801 (13)
.SAVE	142 (2) 792 (13) 801 (13)
.SBTTL	118 (2) 188 (3) 281 (5) 366 (7) 444 (9) 565 (11) 751 (13)
.TITLE	7 (1)
.WORD	804 (14)

```
                                  +---------------------------+
                                  ! Register Cross Reference !
                                  +---------------------------+

REGISTER  NO. REFERENCES     REFERENCES...
--------  --------------     -------------
AP                 72          244    (4)    #-250    (4)    #-252    (4)    #-256    (4)    #-257    (4)    #-258    (4)
                             #-259    (4)    #-260    (4)      264    (4)      266    (4)      269    (4)      270    (4)
                               337    (6)    #-343    (6)    #-347    (6)    #-348    (6)    #-349    (6)    #-350    (6)
                             #-351    (6)    #-355    (6)    #-356    (6)      420    (8)    #-426    (8)    #-430    (8)
                             #-431    (8)    #-432    (8)    #-433    (8)    #-434    (8)    #-438    (8)    #-501    (10)
                             #-503    (10)   #-514    (10)   #-517    (10)   #-523    (10)   #-539    (10)   #-541    (10)
                             #-543    (10)   #-619    (12)   #-633    (12)   #-648    (12)   #-658    (12)   #-668    (12)
                             #-684    (12)   #-685    (12)   #-687    (12)   #-694    (12)   #-710    (12)   #-715    (12)
                             #-729    (12)   #-732    (12)     744    (12)   #-813    (14)     814    (14)   #-817    (14)
                               818    (14)   #-823    (14)     829    (14)   #-830    (14)     831    (14)   #-842    (14)
                               848    (14)   #-872    (14)   #-884    (14)   #-892    (14)   #-897    (14)     905    (14)
                               914    (14)
FP                 13        #-549    (10)   #-556    (10)   #-561    (10)     639    (12)   #-747    (12)     806    (14)
                             #-809    (14)     811    (14)   #-834    (14)     851    (14)   #-868    (14)     870    (14)
                             #-931    (14)
R0                 54        #-244    (4)    #-245    (4)    #-249    (4)    #-262    (4)    #-272    (4)    #-276    (4)
                             #-337    (6)    #-338    (6)    #-342    (6)    #-353    (6)    #-358    (6)    #-361    (6)
                             #-420    (8)    #-421    (8)    #-425    (8)    #-436    (8)    #-439    (8)    #-500    (10)
                             #-506    (10)   #-523    (10)   #-527    (10)   #-530    (10)   #-539    (10)   #-542    (10)
                             #-556    (10)   #-557    (10)   #-558    (10)   #-635    (12)   #-656    (12)   #-658    (12)
                             #-660    (12)   #-662    (12)   #-668    (12)   #-670    (12)   #-674    (12)   #-694    (12)
                             #-704    (12)   #-705    (12)   #-715    (12)   #-716    (12)   #-717    (12)   #-730    (12)
                             #-740    (12)   #-821    (14)   #-841    (14)   #-845    (14)   #-846    (14)   #-849    (14)
                             #-854    (14)   #-855    (14)   #-856    (14)   #-889    (14)   #-918    (14)   #-929    (14)
R1                 27        #-269    (4)    #-273    (4)    #-355    (6)    #-359    (6)    #-438    (8)    #-541    (10)
                             #-543    (10)   #-544    (10)   #-549    (10)   #-557    (10)   #-660    (12)   #-662    (12)
                             #-663    (12)   #-670    (12)   #-674    (12)   #-675    (12)   #-714    (12)   #-717    (12)
                             #-728    (12)   #-854    (14)   #-855    (14)   #-856    (14)   #-878    (14)   #-889    (14)
                             #-891    (14)   #-896    (14)   #-904    (14)
R10                 5          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)
R11                 5          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)
R2                 26          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)   #-650    (12)
                             #-656    (12)   #-666    (12)   #-687    (12)     688    (12)   #-691    (12)   #-705    (12)
                             #-716    (12)   #-719    (12)   #-731    (12)   #-733    (12)   #-736    (12)   #-737    (12)
                             #-738    (12)     804    (14)   #-865    (14)   #-868    (14)   #-890    (14)   #-917    (14)
                             #-920    (14)     925    (14)
R3                 36          237    (4)    #-264    (4)    #-265    (4)    #-266    (4)    #-267    (4)    #-269    (4)
                             #-270    (4)    #-273    (4)      330    (6)    #-356    (6)    #-359    (6)      413    (8)
                               487    (10)     613    (12)   #-648    (12)   #-650    (12)   #-656    (12)   #-704    (12)
                             #-706    (12)   #-710    (12)   #-712    (12)   #-714    (12)   #-729    (12)   #-731    (12)
                             #-737    (12)     804    (14)   #-813    (14)   #-815    (14)   #-822    (14)   #-858    (14)
                             #-880    (14)   #-884    (14)   #-886    (14)   #-891    (14)
R4                 29          237    (4)    #-265    (4)    #-266    (4)    #-267    (4)    #-269    (4)    #-270    (4)
                               330    (6)      413    (8)      487    (10)     613    (12)   #-707    (12)   #-712    (12)
                             #-714    (12)   #-721    (12)   #-728    (12)   #-732    (12)   #-733    (12)   #-738    (12)
                               804    (14)   #-817    (14)   #-819    (14)   #-822    (14)   #-858    (14)   #-880    (14)
                             #-882    (14)   #-886    (14)   #-896    (14)
R5                 14          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)     804    (14)
                             #-830    (14)   #-832    (14)   #-859    (14)   #-880    (14)   #-881    (14)   #-886    (14)
                             #-904    (14)
```

```
R6                 9          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)   #-644    (12)
                            #-681    (12)   #-683    (12)     804    (14)
R7                 9          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)   #-645    (12)
                            #-679    (12)   #-680    (12)     804    (14)
R8                 6          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)     804    (14)
R9                 5          237    (4)      330    (6)      413    (8)      487    (10)     613    (12)
SP                41         #-258    (4)    #-349    (6)    #-432    (8)    #-499    (10)   #-513    (10)   #-515    (10)
                            #-536    (10)   #-632    (12)   #-639    (12)   #-645    (12)   #-651    (12)   #-654    (12)
                            #-662    (12)   #-665    (12)   #-672    (12)   #-673    (12)   #-674    (12)   #-676    (12)
                            #-678    (12)   #-679    (12)   #-680    (12)   #-681    (12)   #-685    (12)   #-699    (12)
                            #-725    (12)   #-806    (14)   #-811    (14)   #-852    (14)   #-853    (14)     860    (14)
                              861    (14)     862    (14)   #-866    (14)   #-867    (14)   #-910    (14)   #-925    (14)

                                          +-------------------------+
                                          ! Performance indicators !
                                          +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     25    00:00:00.01    00:00:00.20
Command processing                890    00:00:00.22    00:00:00.66
Pass 1                            842    00:00:02.73    00:00:04.63
Symbol table sort                   0    00:00:00.10    00:00:00.09
Pass 2                             28    00:00:00.75    00:00:01.54
Symbol table output                 2    00:00:00.04    00:00:00.08
Psect synopsis output               2    00:00:00.00    00:00:00.00
Cross-reference output              7    00:00:00.36    00:00:00.63
Assembler run totals             1798    00:00:04.22    00:00:07.84

The working set limit was 2900 pages.
137253 bytes (269 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 349 non-local and 50 local symbols.
935 source lines were read in Pass 1, producing 56 object records in Pass 2.
86 pages of virtual memory were used to define 22 macros.

                                          +---------------------------+
                                          ! Macro library statistics !
                                          +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              8
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                5
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       7
TOTALS (all libraries)                                20

533 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:PARAM.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

Table of contents

(1) 120 DECLARATIONS
(2) 157 ASCII STRING PARSE ROUTINE.
(7) 526 SCAN$SYMBOL Scan symbol
(9) 623 SCAN$COMPARE Compare strings
(11) 686 SCAN$NUMERIC Scan number
(13) 830 SCAN$SEARCHLIST Search list for unique string
(15) 907 SCAN$DEVICE Scan a device name
(16) 984 Breakup File name into parts

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element PARSE.MAR
0000     2 ; *1     6-FEB-1986 15:21:31 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element PARSE.MAR
0000     4         .TITLE  PARSE   ASCII STRING PARSE ROUTINE.
0000     5         .IDENT  /07-20/
0000     6         .NoShow Conditionals
0000     7
0000     8 ;
0000     9 ; Copyright (c) 1977, 1982, 1983, 1985
0000    10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    11 ;
0000    12 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    13 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    14 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    15 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    16 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    17 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    18 ; REMAIN IN DEC.
0000    19 ;
0000    20 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    21 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    22 ; CORPORATION.
0000    23 ;
0000    24 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    26
0000    27 ;++
0000    28 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;
0000    32 ; ENVIRONMENT:
0000    33 ;
0000    34 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    35 ;
0000    36 ; MODIFIED BY:
0000    37 ;                KEN CHAPMAN     02-FEB-78       VERSION 02 (ESSAA-3.07).
0000    38 ;        01      ADDED QUADWORD DATA CAPABILITY.
0000    39 ;        02      ADDED STRING MATCH FUNCTIO.
0000    40 ;                Roger Riggs     6-NOV-1978
0000    41 ;        03      ADDED KEYWORD MATCH FUNCTION AND MADE SOME SUBROUTINES
0000    42 ;                USEFUL FROM OTHER PARTS OF THE SUPERVISOR.
0000    43 ;        04      Modified keyword construct.
0000    44 ;                Dave Butenhof, 13-feb-1981, version 6.3
0000    45 ;        05      Add parse table command code for SYMBOL type, so CLI can
0000    46 ;                recognise $ and _ in section names.
0000    47 ;
0000    48 ;        06      - Jack Stansbury, 28-Oct-1981, Version 6.-
0000    49 ;                Fixed a few truncation errors. Also added .LIBRARY statements
0000    50 ;                for $DS and $DIAG.
0000    51 ;
0000    52 ;        07      - Dave Butenhof, 10-Nov-1981, version 6.-
0000    53 ;                Add new PARSE codes, CLI$K_COMMA and CLI$K_VALSEP.  The
0000    54 ;                former traverses the sequence "<blank>,<blank>", while the
0000    55 ;                latter traverses "<blank>(=|:)<blank>".  These will make
0000    56 ;                CLI simpler.  Added .LIBRARY for LIB, since I added CASE
0000    57 ;                macro useage
```

```
0000     58 ;
0000     59 ;       08      - Dave Butenhof, 11-Nov-1981, version 6.-
0000     60 ;               Add another PARSE code, CLI$K_EOL to check for end of
0000     61 ;               line (considering comments).
0000     62 ;
0000     63 ;       09      - Dave Butenhof, 11-Nov-1981, version 6.-
0000     64 ;               Again, make things easier for CLI by adding extra (internal
0000     65 ;               only) entry point for PARSE.  DSX$SUPERPARSE won't stop at
0000     66 ;               physical end of line; it'll keep parsing the null string until
0000     67 ;               told to stop.
0000     68 ;
0000     69 ;       10      - Dave Butenhof, 13-Nov-1981, version 6.5
0000     70 ;               One more time...add CLI$K_CALL and CLI$K_RETURN functions
0000     71 ;               for CLI convenience.  The parse table is just too complex.
0000     72 ;               Also, CLI$K_BIFS to test subparse return code
0000     73 ;
0000     74 ;       11      - Dave Butenhof, 03-Feb-1982, Version 6.6
0000     75 ;               Add Scan$File to allow wildcards (* and %) in file names.
0000     76 ;               This is for DIRECTORY...CLI will allow them in RUN, etc;
0000     77 ;               but it won't work too well.  Also, add a macro version of
0000     78 ;               BREAKUP_FILE (for merging default file specs) here...it's
0000     79 ;               much more code efficient than the bliss version.
0000     80 ;
0000     81 ;       12      Jack Stansbury, 15-Dec-1982, Version 6.10
0000     82 ;               Fixed the code in TrvEol to fix a bug where a CLI$K_EOL was
0000     83 ;               being done twice, and causing an error when a comment
0000     84 ;               followed a DS command. The comment character was being
0000     85 ;               skipped, but not the rest of the line after the '!'. Thus,
0000     86 ;               the second time the TrvEol was called, R5 would point to the
0000     87 ;               character after the '!'. Now I set R4 = 0 (the number of
0000     88 ;               characters remaining in the line) and set R5 to point to the
0000     89 ;               real end of the line (the NEWLINE character).
0000     90 ;
0000     91 ;       13      Bob Bergazzi    May 16,1983     Version 6.11
0000     92 ;               Changed the order of the .LIB statements.
0000     93 ;
0000     94 ;       14      M. Baggett      Nov 16,1983     Version 6.13
0000     95 ;               Allow device name of the form ' name$aaann' to be used as load
0000     96 ;               devices.
0000     97 ;
0000     98 ;       15      RE MUSE         MAY 8,1984      Version 7.0
0000     99 ;               Allow device names of the form 'node$ggan' to be used as load
0000    100 ;               devices
0000    101 ;
0000    102 ;       16      M. Baggett      14-Nov-1984     Version 7.1
0000    103 ;               Allow device names of the form '$m$ggan
0000    104 ;
0000    105 ;       17      M. Baggett      19-FEB-1985     Version 8.1
0000    106 ;               Support ULTRIX file structure.
0000    107 ;
0000    108 ;       18      M. Baggett      22-May-1985     Version 8.2
0000    109 ;               Change ULTRIX directory string parsing.
0000    110 ;
0000    111 ;       19      M. Baggett      1-Jun-1985      Version 9.0
0000    112 ;               Change Ultrix filename parsing.
0000    113 ;
0000    114 ;       20      M. Baggett      24-Aug-1985     VDS 9.0
```

```
                        0000   115 ;
                        0000   116 ;
                        0000   117 ;
                        0000   118 ;--
```

Allow commands and qualifiers to be lower case if in ULTRIX mode.
If called from CLI module the commands and qualifiers are convert
to upper case for comparison.

```
                     0000    120          .SBTTL  DECLARATIONS
                     0000    121 ;
                     0000    122 ; INCLUDE FILES:
                     0000    123 ;
                     0000    124          .LIBRARY        /SYS$LIBRARY:LIB/                ;                      [13]
                     0000    125          .LIBRARY        /$DS/                            ;                      [13]
                     0000    126          .LIBRARY        /$DIAG/                          ;                      [13]
                     0000    127 ;
                     0000    128 ; MACROS:
                     0000    129 ;
                     0000    130
                     0000    131 ;
                     0000    132 ; EQUATED SYMBOLS:
                     0000    133 ;
                     0000    134          CLIDEF
                     0000    135          $ds_clidef
                     0000    136          $DS_DSDEF
                     0000    137          $DS_PARSE_DEF
                     0000    138          DSFDEF
                     0000    139          $DS_HPODEF
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .PSECT  $ABS$,ABS
            00000000 044C                 .=0
                     0000                 .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
            00000008 0000                 .IIF    NB,..BLKQ,      .BLKQ   1
                     0008                 .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
            0000000A 0008                 .IIF    NB,..BLKW,      .BLKW   1
                     000A                 .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
            0000000B 000A                 .IIF    NB,..BLKB,      .BLKB   1
                     000B                 .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
            0000000C 000B                 .IIF    NB,..BLKB,      .BLKB   1
                     000C                 .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
            00000018 000C                 .IIF    NB,..BLKB,      .BLKB   12
                     0018                 .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
            0000001C 0018                 .IIF    NB,..BLKL,      .BLKL   1
                     001C                 .IIF    NB,HP$A_DVA,    HP$A_DVA:
            00000020 001C                 .IIF    NB,..BLKL,      .BLKL   1
                     0020                 .IIF    NB,HP$A_LINK,   HP$A_LINK:
            00000024 0020                 .IIF    NB,..BLKL,      .BLKL   1
                     0024                 .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
            00000026 0024                 .IIF    NB,..BLKW,      .BLKW   1
                     0026                 .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
            00000032 0026                 .IIF    NB,..BLKB,      .BLKB   12
                     0032                 .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                     0000                 .RESTORE
            00000002 0000    140 V_BIT=2
            00000001 0000    141 SS$_NORMAL=1
            00000009 0000    142 TAB =    ^X09                                ; TAB CHARACTER                    [20]
            00000020 0000    143 SPACE = ^X20                                 ; SPACE CHARACTER                  [20]
                     0000    144 ;
                     0000    145 ; OWN STORAGE:
                     0000    146 ;
                00000000     147          .Psect  Work, Noshr, Noexe, Wrt, Long   ;                               [10]
                     0000    148 Save_call:                                   ;                                  [10]
            00000000 0000    149          .long   0                           ; Address for RETURN               [10]
                     0004    150 Save_status:                                 ; Status of CALLed subparse        [10]
            00000000 0004    151          .long   0                           ;                                  [10]
```

```
                0008   152 Ultrix_filename:                                  ;                                  [19]
      00000058  0008   153         .blkb   80                                ; use for filename must be shifted
                0058   154                                                   ; to insert '/'
            00000000   155         .PSECT  Code, Shr, Exe, Nowrt, Byte
```

```
0000    157          .SBTTL  ASCII STRING PARSE ROUTINE.
0000    158 ;++
0000    159 ; FUNCTIONAL DESCRIPTION:
0000    160 ;
0000    161 ;       THIS ROUTINE TAKES AN ASCII STRING INPUT AND PARSES IT USING A
0000    162 ;       CALLER SUPPLIED PARSE TREE.  UPON A MATCH IN THE TREE, IT
0000    163 ;       DISPATCHES TO A CALLER SUPPLIED ACTION ROUTINE.  UPON A MISMATCH,
0000    164 ;       IT BRANCHS TO ANOTHER POINT IN THE PARSE TREE.
0000    165 ;
0000    166 ; CALLING SEQUENCE:
0000    167 ;
0000    168 ;       DIAGNOSTIC PROCEDURE CALL.
0000    169 ;
0000    170 ; INPUT PARAMETERS:
0000    171 ;
0000    172 ;       PARSE$_BUFADR(AP)       = INPUT COUNTED STRING ADDRESS.
0000    173 ;       PARSE$_TREE(AP)         = PARSE TREE ENTRY POINT.
0000    174 ;       PARSE$_ACTION(AP)       = SUCCESS ACTION SUBROUTINE ADDRESS.
0000    175 ;
0000    176 ; IMPLICIT INPUTS:
0000    177 ;
0000    178 ;       NONE
0000    179 ;
0000    180 ; OUTPUT PARAMETERS:
0000    181 ;
0000    182 ;       NONE
0000    183 ;
0000    184 ; IMPLICIT OUTPUTS:
0000    185 ;
0000    186 ;       RESULTS OF CALLER'S ACTION SUBROUTINE.
0000    187 ;
0000    188 ; COMPLETION CODES:
0000    189 ;
0000    190 ;       SS$_NORMAL      = STRING SUCCESSFULLY PARSED.
0000    191 ;       DS$_ERROR       = ERROR MATCH CODE ENCOUNTERED IN PARSE TREE.
0000    192 ;
0000    193 ; SIDE EFFECTS:
0000    194 ;
0000    195 ;       NONE
0000    196 ;
0000    197 ; REGISTER USAGE:
0000    198 ;
0000    199 ;       R0      = SCRATCH AND COMPLETION CODES.
0000    200 ;       R1      = SCRATCH.
0000    201 ;       R2      = SCRATCH
0000    202 ;       R3      = RADIX AND BIT MASK POINTER
0000    203 ;       R4      = LENGTH OF REMAINING INPUT STRING
0000    204 ;       R5      = ADDRESS OF REMAINING INPUT STRING
0000    205 ;       R7      = PARSE TREE POINTER.
0000    206 ;--
```

```
                  0FFC  0000  .entry  dsx$superparse,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ;                [09]
        56   01    D0  0002          movl    #1, r6                  ; Don't exit on end of line          [09]
             04    11  0005          brb     parse_common            ; Join common code                   [09]
                  0FFC  0007  .ENTRY  DSX$PARSE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
             56    D4  0009          clrl    r6                      ; Exit on end of line                [09]
                       000B  parse_common:                           ;                                    [09]
     54   04 BC    7D  000B          MOVQ    @PARSE$_BUFADR(AP),R4   ; GET LENGTH, ADDR OF INPUT STRING
        54   54    3C  000F          MOVZWL  R4, R4                  ; LENGTH.
  00000000'EF      D4  0012          clrl    Save_call               ; Clear saved call address           [10]
     57   08 AC    D0  0018          MOVL    PARSE$_TREE(AP), R7     ; TREE POINTER.
                       001C  TRAVERSE:
          04 56    E8  001C          blbs    r6, 10$                 ; Don't check for EOL if set         [09]
             54    D5  001F          TSTL    R4                      ; ANY CHARACTERS LEFT IN INPUT STREAM
             42    13  0021          BEQL    EXIT                    ; BRANCH IF NOT
          F6 AF    DF  0023  10$:    PUSHAL  TRAVERSE                ; FAKE A SUBROUTINE CALL.            [09]
                       0026          case    src=(r7), type=b, -     ; Case on the dispatch codes         [07]
                       0026                   limit=#cli$k_error, -  ;                                    [07]
                       0026                   displist=< -           ; ...dispatch list                   [07]
                       0026                  trverr, -               ; 0 = Error.                         [07]
                       0026                  trvexit, -              ; 1 = Exit.                          [07]
                       0026                  trvbr, -                ; 2 = Branch.                        [07]
                       0026                  trvbif, -               ; 3 = Conditional branch.            [07]
                       0026                  trvspace, -             ; 4 = Separator characters.          [07]
                       0026                  trvnum, -               ; 5 = Number string.                 [07]
                       0026                  trvalpha, -             ; 6 = Alpha string.                  [07]
                       0026                  trvalnum, -             ; 7 = Alphanumeric string.           [07]
                       0026                  trvoct, -               ; 8 = Octal number string.           [07]
                       0026                  trvhex, -               ; 9 = Hex number string.             [07]
                       0026                  trvdec, -               ; 10 = Decimal number string.        [07]
                       0026                  trvstring, -            ; 11 = Unique string.                [07]
                       0026                  trvkeyword, -           ; 12 = Keyword.                      [07]
                       0026                  trvsymbol, -            ; 13 = Symbol.                       [07]
                       0026                  trvcomma, -             ; 14 = <blank>comma<blank>.          [07]
                       0026                  trvvalue, -             ; 15 = <blank>(=|:)<blank>.          [07]
                       0026                  trvslash, -             ; 16 = <blank>slash<blank>.          [07]
                       0026                  trveol, -               ; 17 = <blank>(!|<eol>).             [08]
                       0026                  trvcall, -              ; 18 = Call subtree parse.           [10]
                       0026                  trvreturn, -            ; 19 = Return from subtree.          [10]
                       0026                  trvbifs, -              ; 20 = Test subparse status.         [10]
                       0026                  trvfile -               ; 21 = Filename (incl. '%' and '*')  [11]
                       0026                  >                       ;                                    [07]
15'  80 8F   67    8F  0026          CASEb   (r7),#cli$k_error,S^#<<30001$-30000$>/2>-1
                       002B  30000$:
                  009C' 002B          .SIGNED_WORD    trverr-30000$
                  0038' 002D          .SIGNED_WORD    trvexit-30000$
                  006C' 002F          .SIGNED_WORD    trvbr-30000$
                  00B3' 0031          .SIGNED_WORD    trvbif-30000$
                  00C2' 0033          .SIGNED_WORD    trvspace-30000$
                  00D6' 0035          .SIGNED_WORD    trvnum-30000$
                  00E2' 0037          .SIGNED_WORD    trvalpha-30000$
                  00E7' 0039          .SIGNED_WORD    trvalnum-30000$
                  00C7' 003B          .SIGNED_WORD    trvoct-30000$
                  00CC' 003D          .SIGNED_WORD    trvhex-30000$
                  00D1' 003F          .SIGNED_WORD    trvdec-30000$
                  00EC' 0041          .SIGNED_WORD    trvstring-30000$
                  013B' 0043          .SIGNED_WORD    trvkeyword-30000$
                  016C' 0045          .SIGNED_WORD    trvsymbol-30000$
```

```
                     0177' 0047                .SIGNED_WORD    trvcomma-30000$
                     0180' 0049                .SIGNED_WORD    trvvalue-30000$
                     017B' 004B                .SIGNED_WORD    trvslash-30000$
                     01C0' 004D                .SIGNED_WORD    trveol-30000$
                     005E' 004F                .SIGNED_WORD    trvcall-30000$
                     003E' 0051                .SIGNED_WORD    trvreturn-30000$
                     00A7' 0053                .SIGNED_WORD    trvbifs-30000$
                     0171' 0055                .SIGNED_WORD    trvfile-30000$
                           0057          30001$:
                           0057    249
                           0057    250 ;
                           0057    251 ; NOT A SPECIAL CODE.
                           0057    252 ;
                65   67   91    0057    253         CMPB    (R7),(R5)               ; TREE CHAR MATCH INPUT STRING?
                     3D   12    005A    254         BNEQ    BRANCH                  ; BRANCH IF NOT
                     55   D6    005C    255         INCL    R5                      ; UPDATE ADDRESS
                     54   D7    005E    256         DECL    R4                      ; COUNT THIS CHARACTER
                   0085   31    0060    257         BRW     DO_ACTION               ; TO ACTION AND UPDATE TREE
                           0063    258 ;
                           0063    259 ; NORMAL EXIT CODE
                           0063    260 ;
                           0063    261 TRVEXIT:
                     3C   10    0063    262         BSBB    ACTION                  ; EXECUTE CALLER ACTION ROUTINE.
                           0065    263 EXIT:
                50   01   D0    0065    264         MOVL    #SS$_NORMAL, R0         ; SET NORMAL RETURN.
                           0068    265 DSX$PARSE_X:
                     04    0068    266         RET                             ; RETURN.
```

```
                        0069   268 ;
                        0069   269 ; Return special code.  Note: the PARSE call mechanism is not recursive;
                        0069   270 ; only one level of subtree can be used.
                        0069   271 ;
                        0069   272 trvreturn:                                      ;                                   [10]
                  36    10    0069   273         bsbb    action                   ; Call action routine               [10]
00000004'EF       50    D0    006B   274         movl    r0, Save_status          ; Save Call status for BIFS test    [10]
50    00000000'EF   D0    0072   275         movl    Save_call, r0            ; Restore saved value               [10]
                  09    12    0079   276         bneq    10$                      ; Branch if saved address not 0     [10]
50    00660002    8F    D0    007B   277         movl    #ds$_error, r0           ; If was 0, error                   [10]
                  E4    11    0082   278         brb     dsx$parse_x              ; Exit                              [10]
      57    50    04    C1    0084   279 10$:    addl3   #4, r0, r7               ; Advance to next                   [10]
                  05    0088   280         rsb                              ; Do next                           [10]
                        0089   281
                        0089   282 ;
                        0089   283 ; Call special code.  Note: this is not recursive.  Only one level of call
                        0089   284 ; may be used.  The action routine will be called before the subtree parse.
                        0089   285 ;
                        0089   286 trvcall:                                       ;                                   [10]
00000004'EF       01    D0    0089   287         Movl    #1, Save_status          ; Assume successful, if not specified [10]
00000000'EF       57    D0    0090   288         movl    r7, Save_call            ; Save address for return           [10]
                        0097   289 ;       brb     trvbr                    ; Fall through to TRVBR code        [10]
                        0097   290 ;
                        0097   291 ; BRANCH SPECIAL CODE.
                        0097   292 ;
                        0097   293 TRVBR:
                  08    10    0097   294         BSBB    ACTION                   ; EXECUTE CALLER ACTION ROUTINE.
                        0099   295 ;       BRB     BRANCH                   ; FALL INTO BRANCH
                        0099   296 ;
                        0099   297 ; HERE TO TAKE BRANCH IN TREE
                        0099   298 ;
                        0099   299 BRANCH:
      50    02    A7    32    0099   300         CVTWL   2(R7), R0                ; GET DISPLACEMENT.
            57    50    C0    009D   301         ADDL2   R0, R7                   ; UPDATE TREE POINTER.
                        05    00A0   302         RSB                              ; CONTINUE TO TRAVERSE TREE.
                        00A1   303 ;
                        00A1   304 ; DISPATCH TO CALLER'S ACTION ROUTINE.
                        00A1   305 ;
      5A    50    7D    00A1   306 ACTION: MOVQ    R0,R10                   ; COPY QUAD NUMBER TO R10-R11
      58    55    D0    00A4   307         MOVL    R5,R8                    ; COPY ADDRESS OF REMAINING STRING
      59    54    D0    00A7   308         MOVL    R4,R9                    ; COPY LENGTH OF REMAINING STRING
      50    01    A7    9A    00AA   309         MOVZBL  1(R7), R0                ; GET ACTION CODE.
52    00000000'EF       9E    00AE   310         MOVAB   L^DS$GL_CLIBASE,R2       ; BASE CLI DATA BASE                [06]
            0080    8F    BB    00B5   311         PUSHR   #^M<R7>                  ; SAVE OUR REGS
              0C    BC    16    00B9   312         JSB     @PARSE$_ACTION(AP)       ; GO TO ACTION ROUTINE.
            0080    8F    BA    00BC   313         POPR    #^M<R7>                  ; RESTORE OUR REGS
            54    59    D0    00C0   314         MOVL    R9,R4                    ; PUT CHARACTER COUNT BACK
            55    58    D0    00C3   315         MOVL    R8,R5                    ; PUT CHARACTER POINT BACK
                        05    00C6   316         RSB
                        00C7   317 ;
                        00C7   318 ; ERROR EXIT CODE.
                        00C7   319 ;
                        00C7   320 TRVERR:
                  D8    10    00C7   321         BSBB    ACTION                   ; EXECUTE CALLER ACTION ROUTINE.
50    00660002    8F    D0    00C9   322         MOVL    #DS$_ERROR, R0           ; SET ERROR RETURN CODE.
                  96    11    00D0   323         BRB     DSX$PARSE_X              ; GO EXIT.
                        00D2   324 ;
```

```
                        00D2   ; BIFS conditional branch code.  This is identical to BIF, except
                        00D2   ; that it tests the saved return code from a CALL subparse instead
                        00D2   ; of R0.
                        00D2   ;
                        00D2   TRVBIFS:                                  ;                                  [10]
              CD   10   00D2           Bsbb    Action                    ; Call action routine              [10]
BE 00000004'EF   E8   00D4           Blbs    Save_status, Branch       ; Branch if OK                     [10]
              87   D5   00DB           Tstl    (r7)+                     ; Otherwise advance to next        [10]
                   05   00DD           Rsb                               ; Continue to traverse tree        [10]
                        00DE
                        00DE   ;
                        00DE   ; BIF CONDITIONAL BRANCH SPECIAL CODE.
                        00DE   ;
                        00DE   TRVBIF:
              C1   10   00DE           BSBB    ACTION                    ; EXECUTE CALLER ACTION ROUTINE.
           B6 50   E8   00E0           BLBS    R0, BRANCH                ; BRANCH IF SATISFIED.
              87   D5   00E3           TSTL    (R7)+                     ; UPDATE TREE POINTER.
                   05   00E5           RSB                               ; CONTINUE TO TRAVERSE TREE.
```

```
                                00E6   BRANCH_IF_NONE:
                      B1    13  00E6           BEQL    BRANCH                  ; Branch if Length was zero
                                00E8   DO_ACTION:
                      B7    10  00E8           BSBB    ACTION                  ; CALL ACTION ROUTINE
                      87    D5  00EA           TSTL    (R7)+                   ; UPDATE TREE POINTER
                            05  00EC           RSB                             ; CONTINUE TRAVERSE
                                00ED   ;
                                00ED   ; SPACE SPECIAL CODE.
                                00ED   ;
                                00ED   TRVSPACE:
                    0162    30  00ED           BSBW    SCAN$SPACE              ; CHECK FOR SPACES
                      F4    11  00F0           BRB     BRANCH_IF_NONE          ; TAKE BRANCH IF NO SPACES OR TABS
                                00F2   ;
                                00F2   ; NUMERIC VALUE CODE.
                                00F2   ;
                                00F2   TRVOCT:
                  53    08  D0  00F2           MOVL    #8, R3                  ; SET RADIX TO OCTAL.
                      11    11  00F5           BRB     TRVNUMA
                                00F7   TRVHEX:
                  53    10  D0  00F7           MOVL    #16, R3                 ; SET RADIX TO HEXIDECIMAL.
                      0C    11  00FA           BRB     TRVNUMA
                                00FC   TRVDEC:
                  53    0A  D0  00FC           MOVL    #10, R3                 ; SET RADIX TO DECIMAL.
                      07    11  00FF           BRB     TRVNUMA
                                0101   TRVNUM:
          53  00000000'EF    D0  0101           MOVL    L^DS$GL_RADIX, R3       ; GET DEFAULT RADIX.                [06]
                                0108   TRVNUMA:
                    01CB    30  0108           BSBW    SCAN$NUMERIC            ; SCAN SOMETHING NUMERIC
                      D9    11  010B           BRB     BRANCH_IF_NONE          ; TAKE BRANCH IF NOTHING FOUND
                                010D   ;
                                010D   ; ALPHA CHECK.
                                010D   ;
                                010D   TRVALPHA:
                    010C    30  010D           BSBW    SCAN$ALPHA              ; CHECK FOR ALPHA                   [07]
                      D4    11  0110           BRB     BRANCH_IF_NONE          ; DO ACTION IF FOUND ELSE BRANCH
                                0112   ;
                                0112   ; ALPHA-NUMERIC CHECK.
                                0112   ;
                                0112   TRVALNUM:
                    0119    30  0112           BSBW    SCAN$ALPHANUM           ; SPAN ALPHA-NUMERIC                [07]
                      CF    11  0115           BRB     BRANCH_IF_NONE          ; DO ACTION IF FOUND ELSE BRANCH
                                0117   ;
                                0117   ; UNIQUE STRING MATCH.
                                0117   ;
                                0117   .enabl  lsb                             ; Allow access to BRW BRANCH        [10]
                                0117   TRVSTRING:
              53    04 A7   9E  0117           MOVAB   4(R7), R3               ; GET POINTER TO ASCIC.
                  52    83  9A  011B           MOVZBL  (R3)+, R2               ; GET ASCII COUNT.
                                011E
                    34 56   E9  011E           BLBC    R6,130$                 ; Ignore if not called from CLI module [20]
28 00000000'EF  00000000'8F   E1  0121           BBC     #ULTRIX_V_MODE,ULTRIX_FLAGS,130$  ; BRANCH IF NOT ULTRIX    [20]
                      51    D4  012D           CLRL    R1                      ; CLEAR INDEX                       [20]
                  50    54  D0  012F           MOVL    R4,R0                   ; LENGTH OF TOTAL COMMAND STRING    [20]
                  52    54  D1  0132           CMPL    R4,R2                   ; CHECK LENGTH AGAINST PATTERN TO MATCH [20]
                      03    15  0135           BLEQ    110$                    ; RETAIN SHORTER LENGTH             [20]
                  50    52  D0  0137           MOVL    R2,R0                   ;                                   [20]
                20    6541  91  013A   110$:   CMPB    (R5)[R1],#SPACE         ; EXIT IF SPACE                     [20]
```

```
                      15   13  013E   401          BEQL    130$                    ; END OF SECTION TO COMPARE                [20]
                09  6541   91  0140   402          CMPB    (R5)[R1],#TAB           ; EXIT IF TAB                              [20]
                    0F   13  0144   403          BEQL    130$                    ; END OF SECTION TO COMPARE                [20]
             04 6541   06  E1  0146   404          BBC     #6,(R5)[R1],120$        ; SKIP IF NOT ALPHA CHAR                   [20]
                6541   20  8A  014B   405          BICB2   #32,(R5)[R1]            ; FORCE Character TO UPPER CASE            [20]
         FFE5 51   01   50  9D  014F   406 120$:    ACBB    R0,#1,R1,110$           ; CHECK TOTAL SECTION                      [20]
                                0155   407
                    0129   30  0155   408 130$:    BSBW    SCAN$COMPARE            ; COMPARE
                      37   19  0158   409          BLSS    20$                     ; TAKE BRANCH IF NO MATCH                  [10]
                    FF44   30  015A   410          BSBW    ACTION                  ; GO DO CALLER ACTION.
                      87   D5  015D   411          TSTL    (R7)+                   ; UPDATE TREE POINTER.
                50    87   9A  015F   412          MOVZBL  (R7)+, R0               ; GET ASCIC COUNT.
                57    50   C0  0162   413          ADDL2   R0, R7                  ; PAST THE ASCII.
                           05  0165   414          RSB                             ; CONTINUE TO TRAVERSE TREE.
```

```
                        0166   416 ;
                        0166   417 ; KEYWORD MATCH ROUTINE
                        0166   418 ;
                        0166   419 TRVKEYWORD:
          00A1   30     0166   420          BSBW    SCAN$SYMBOL                ; SCAN A SYMBOL
            26   13     0169   421          BEQL    20$                        ; IF NO SYMBOL, TAKE BRANCH
       7E   50   D0     016B   422          MOVL    R0,-(SP)                   ; SAVE STRING LENGTH
    53   06 A7   32     016E   423          CVTWL   6(R7),R3                   ; Get offset to keyword table
       53   57   C0     0172   424          ADDL    R7,R3                      ; Make absolute address of table
          01FA   30     0175   425          BSBW    SCAN$SEARCHLIST            ; LOCATE SYMBOL IN TABLE
            1A   14     0178   426          BGTR    30$                        ; Exit if ambigious
            0F   19     017A   427          BLSS    10$                        ; IF NOT FOUND TAKE BRANCH
    04 A7   50   B1     017C   428          CMPW    R0,4(R7)                   ; CORRECT KEYWORD
            09   12     0180   429          BNEQ    10$                        ; NO, TAKE BRANCH
          FF1C   30     0182   430          BSBW    ACTION                     ; DO ACTION ROUTINE
       57   08   C0     0185   431          ADDL2   #8,R7                      ; UPDATE TREE POINTER
            8E   D5     0188   432          TSTL    (SP)+                      ; REMOVE SYMBOL LENGTH
                 05     018A   433          RSB                                ; NEXT TREE JUNCTURE
       54   6E   C0     018B   434 10$:      ADDL    (SP),R4                    ; ADJUST INPUT STRING LENGTH BY SYMBOL
       55   8E   C2     018E   435          SUBL    (SP)+,R5                   ; ADJUST INPUT STRING ADDR BY SYMBOL
          FF05   31     0191   436 20$:      BRW     BRANCH                     ; TAKE FAILURE BRANCH
          FECE   31     0194   437 30$:      BRW     EXIT                       ; TAKE EXIT IF AMBIGIOUS
                        0197   438 .dsabl   lsb                                ;                                          [10]
                        0197   439 ;
                        0197   440 ; Symbol match routine
                        0197   441 ;
                        0197   442 TRVSYMBOL:
            71   10     0197   443          BSBB    SCAN$SYMBOL                ; Search for symbol
          FF4A   31     0199   444          BRW     BRANCH_IF_NONE             ; Handle tree decision point
                        019C   445 ;
                        019C   446 ; Filename match routine
                        019C   447 ;
                        019C   448 TrvFile:                                     ;                                          [11]
          00A1   30     019C   449          BsbW    Scan$File                  ; Scan over filename                       [11]
          FF44   31     019F   450          BrW     Branch_If_None             ; Handle tree decision point               [11]
                        01A2   451 ;
                        01A2   452 ; Traverse a string consisting of blanks and a comma.  If the comma is not
                        01A2   453 ; found, the input string is unaffected: if the comma is found, then the
                        01A2   454 ; comma and all preceding and following blanks are skipped.
                        01A2   455 ;
                        01A2   456 trvcomma:                                    ;                                          [07]
            53   D4     01A2   457          clrl    r3                         ; Set up ',' code                          [07]
            08   11     01A4   458          brb     trv_comma_value            ; Use common routine                       [07]
                        01A6   459
                        01A6   460 ;
                        01A6   461 ; Traverse a string consisting of blanks and a slash.  If the slash is not
                        01A6   462 ; found, the input string is unaffected: if the slash is found, then the
                        01A6   463 ; slash and all preceding and following blanks are skipped
                        01A6   464 ;
                        01A6   465 trvslash:                                    ;                                          [07]
    53      01   D0     01A6   466          movl    #1, r3                     ; Set up '/' code                          [07]
            03   11     01A9   467          brb     trv_comma_value            ; Use common routine                       [07]
                        01AB   468
                        01AB   469 ;
                        01AB   470 ; Traverse a string consisting of blanks and either an '=' or a ':'.  This
                        01AB   471 ; is primarily for qualifier values.  The code from TRV_COMMA_VALUE on is
                        01AB   472 ; common between this function and the TRVCOMMA function.
```

```
                        01AB   ;
                        01AB   trvvalue:
        53   02   D0    01AB           movl    #2, r3                  ; Set up '=' code                       [07]
                        01AE
                        01AE   trv_comma_value:
             38   BB    01AE           pushr   #^M<r3,r4,r5>           ; Save current descriptor               [07]
           009F   30    01B0           bsbw    scan$space              ; Skip any spaces we might find         [07]
             50   D4    01B3           clrl    r0                      ; Assume didn't find anything           [07]
             54   D5    01B5           tstl    r4                      ; Any characters remaining?             [07]
             0E   15    01B7           bleq    10$                     ; If not, can't be either               [07]
             54   D7    01B9           decl    r4                      ; Anticipate finding it                 [07]
                        01BB           case    src=(sp), type=b, -     ; Case on type code                     [07]
                        01BB                    limit=#0, displist=< - ;                                        [07]
                        01BB                   10$, -                  ; 0 is ','                              [07]
                        01BB                   20$, -                  ; 1 is '/'                              [07]
                        01BB                   30$ -                   ; 2 is '=' or ':'                       [07]
                        01BB                   >                       ;                                       [07]
  02'  00   6E   8F     01BB           CASEb   (sp),#0,S^#<<30003$-30002$>/2>-1
                        01BF       30002$:
                0008'   01BF           .SIGNED_WORD    10$-30002$
                000D'   01C1           .SIGNED_WORD    20$-30002$
                0012'   01C3           .SIGNED_WORD    30$-30002$
                        01C5       30003$:
             1D   11    01C5           brb     60$                     ; No go                                 [07]
        2C   85   91    01C7   10$:    cmpb    (r5)+, #^A","           ; Compare next character to ','         [07]
             0E   11    01CA           brb     40$                     ; Check it out                          [07]
        2F   85   91    01CC   20$:    cmpb    (r5)+, #^A"/"           ; Is it a '/'?                          [07]
             09   11    01CF           brb     40$                     ; Check it out                          [07]
        3D   85   91    01D1   30$:    cmpb    (r5)+, #^A"="           ; Is is an '='?                         [07]
             06   13    01D4           beql    50$                     ; Yep, OK                               [07]
     3A   FF A5   91    01D6           cmpb    -1(r5), #^A":"          ; Otherwise, is it an ':'?              [07]
             08   12    01DA   40$:    bneq    60$                     ; If not, failure                       [07]
             74   10    01DC   50$:    bsbb    scan$space              ; Skip trailing spaces, too             [07]
     04 AE   54   7D    01DE           movq    r4, 4(sp)               ; Replace the saved descriptor          [07]
             50   D6    01E2           incl    r0                      ; We found something                    [07]
             38   BA    01E4   60$:    popr    #^M<r3,r4,r5>           ; Restore registers                     [07]
             50   D5    01E6           tstl    r0                      ; Check if we found anything            [07]
           FEFB   31    01E8           brw     branch_if_none          ; And finish up command                 [07]
                        01EB
                        01EB   ;
                        01EB   ; TRVEOL
                        01EB   ; Skip spaces and tabs; if next character is an '!' or if there are no
                        01EB   ; more characters, then success; else failure (take MISS branch).
                        01EB   ;
             30   BB    01EB   TrvEol: pushr   #^M<r4,r5>              ; Save the input descriptor             [08]
             63   10    01ED           bsbb    scan$space              ; Skip spaces if any are there          [08]
             54   D5    01EF           tstl    r4                      ; Check remaining length                [08]
             0C   13    01F1           beql    10$                     ; Branch to check                       [08]
             54   D7    01F3           decl    r4                      ; Downcount for "!"                     [08]
        21   85   91    01F5           cmpb    (r5)+, #^A"!"           ; Otherwise, check for comment          [08]
             05   13    01F8           beql    10$                     ; OK, do action routine                 [08]
             30   BA    01FA           popr    #^M<r4,r5>              ; Restore registers                     [08]
           FE9A   31    01FC           brw     branch                  ; Take MISS branch                      [08]
                        01FF
  55   55   54   C1    01FF   10$:    AddL3   R4,R5,R5                ; Point R5 to the <CR> character        [12]
             54   D4    0203           ClrL    R4                      ; No more characters in the line        [12]
             03   BA    0205           popr    #^M<r0,r1>              ; Remove saved registers from stack     [08]
```

```
                        FEDE   31  0207   524        brw     do_action               ; Do the action routine                 [08]
```

```
                        020A    526          .SBTTL  SCAN$SYMBOL     Scan symbol
                        020A    527 ;++
                        020A    528 ; FUNCTIONAL DESCRIPTION:
                        020A    529 ;
                        020A    530 ;       SCAN$SYMBOL     Skips characters of the set (A-Z, a-z, 0-9, $, _) .
                        020A    531 ;       SCAN$SPACE      Skips spaces and tabs.
                        020A    532 ;       SCAN$ALPHA      Skips characters of the set (A-Z,a-z).
                        020A    533 ;       SCAN$ALPHANUM   Skips characters of the set (A-z,a-z,0-9).
                        020A    534 ;       SCAN$SPANC      Skips characters of the set specified by R3
                        020A    535 ;
                        020A    536 ; CALLING SEQUENCE:
                        020A    537 ;
                        020A    538 ;       BSBW    SCAN$SYMBOL
                        020A    539 ;       BSBW    SCAN$ALPHA
                        020A    540 ;       BSBW    SCAN$ALPHANUM
                        020A    541 ;       BSBW    SCAN$SPACE
                        020A    542 ;       BSBW    SCAN$SPANC
                        020A    543 ;
                        020A    544 ; INPUT PARAMETERS:
                        020A    545 ;
                        020A    546 ;       R3      Address of bit mask for valid characters (SCAN$SPANC only)
                        020A    547 ;       R4      Length of input string
                        020A    548 ;       R5      Address of input string
                        020A    549 ;
                        020A    550 ; IMPLICIT INPUTS:      NONE
                        020A    551 ;
                        020A    552 ; OUTPUT PARAMETERS:
                        020A    553 ;
                        020A    554 ;       R0      Length of symbol
                        020A    555 ;       R1      Address of symbol
                        020A    556 ;       R3      Address of the bit mask used for validity
                        020A    557 ;       R4      Updated length of input string
                        020A    558 ;       R5      Updated address of input string
                        020A    559 ;
                        020A    560 ; IMPLICIT OUTPUTS:     NONE
                        020A    561 ;
                        020A    562 ; COMPLETION CODES:     N/A
                        020A    563 ;
                        020A    564 ; SIDE EFFECTS:         NONE
                        020A    565 ;
                        020A    566 ; REGISTER USAGE:
                        020A    567 ;
                        020A    568 ;       R0      Length of symbol
                        020A    569 ;       R1      Address of symbol
                        020A    570 ;       R3      Address of bit mask used for validity
                        020A    571 ;       R4      Current remaining length of input string
                        020A    572 ;       R5      Current address of input string
                        020A    573 ;
                        020A    574 ; CONDITION CODES:
                        020A    575 ;
                        020A    576 ;       Z-bit   Set if length of symbol=0, else set
                        020A    577 ;       N-bit   Zero
                        020A    578 ;--
```

```
                                020A   580          .ENABL  LSB
                                020A   581 SCAN$SYMBOL::
                          58 10 020A   582          BSBB    10$                     ; Address Mask and branch to common
                                020C   583 DS$GT_VSYMBOL::
07FFFFFE 87FFFFFE 03FF0010 00000000 020C   584          .LONG   ^X00000000,^X03FF0010,^X87FFFFFE,^X07FFFFFE
                                021C   585
                                021C   586 SCAN$ALPHA::
                          46 10 021C   587          BSBB    10$                     ; Address Mask and branch to common
                                021E   588 DS$GT_VALPHA::
07FFFFFE 07FFFFFE 00000000 00000000 021E   589          .LONG   ^X00000000,^X00000000,^X07FFFFFE,^X07FFFFFE
                                022E   590
                                022E   591 SCAN$ALPHANUM::
                          34 10 022E   592          BSBB    10$                     ; Address Mask and branch to common
                                0230   593 DS$GT_VALPHANUM::
07FFFFFE 07FFFFFE 03FF0000 00000000 0230   594          .LONG   ^X00000000,^X03FF0000,^X07FFFFFE,^X07FFFFFE
                                0240   595
                                0240   596 Scan$File::
                          22 10 0240   597          BsbB    10$                     ; Address Mask and branch to common
                                0242   598 DS$GT_VFILE::
07FFFFFE 87FFFFFE 03FF0430 00000000 0242   599          .Long   ^X00000000,^X03FF0430,^X87FFFFFE,^X07FFFFFE
                                0252   600
                                0252   601 SCAN$SPACES::
                                0252   602 SCAN$SPACE::
                          10 10 0252   603          BSBB    10$                     ; Address Mask and branch to common
                                0254   604 DS$GT_VSPACE::
00000000 00000000 00000001 00000200 0254   605          .LONG   ^X00000200,^X00000001,^X00000000,^X00000000
                                0264   606
                          08 BA 0264   607 10$:     POPR    #^M<R3>                 ; Get address of mask to R3
                                0266   608                                          ; Fall into SCAN$SPANC
                                0266   609          .DSABL  LSB
                                0266   610
                                0266   611 SCAN$SPANC::
                       54 54 3C 0266   612          MOVZWL  R4,R4                   ; Trim to just length
                       51 55 D0 0269   613          MOVL    R5,R1                   ; Copy start of string
                          09 11 026C   614          BRB     20$                     ; Start with count of byte
                       50 65 9A 026E   615 10$:     MOVZBL  (R5),R0                 ; Fetch a character
                    05 63 50 E1 0271   616          BBC     R0,(R3),30$             ; Branch if not valid
                       55 D6 0275   617          INCL    R5                      ; Update address
                          F4 54 F4 0277   618 20$:     SOBGEQ  R4,10$                  ; Decrement count and branch if more
                       54 D6 027A   619 30$:     INCL    R4                      ; Fix count of characters remaining
                 50 55 51 C3 027C   620          SUBL3   R1,R5,R0                ; Calculate the length just scanned
                          05 0280   621 40$:     RSB                             ; Return
```

```
                        623          .SBTTL  SCAN$COMPARE    Compare strings
0281    624 ;++
0281    625 ; FUNCTIONAL DESCRIPTION:
0281    626 ;
0281    627 ;       This routine performs the same function as the CMPC3
0281    628 ;       machine instruction but does not use the instructions.
0281    629 ;
0281    630 ; CALLING SEQUENCE:
0281    631 ;
0281    632 ;       BSBW    SCAN$COMPARE
0281    633 ;
0281    634 ; INPUT PARAMETERS:
0281    635 ;
0281    636 ;       R2      Length of string 1
0281    637 ;       R3      Address of string1
0281    638 ;       R4      Length of string 2
0281    639 ;       R5      Address of string 2
0281    640 ;
0281    641 ; IMPLICIT INPUTS:      NONE
0281    642 ;
0281    643 ; OUTPUT PARAMETERS:
0281    644 ;
0281    645 ;       R0      Length of matching section
0281    646 ;       R1      Address of start of string1
0281    647 ;       R2      Length of uncompared string 1
0281    648 ;       R3      Address of Uncompared byte in string 1
0281    649 ;       R4      Length of Uncompared string 2
0281    650 ;       R5      Address of uncompared byte in string 2
0281    651 ;
0281    652 ; IMPLICIT OUTPUTS:      NONE
0281    653 ;
0281    654 ; SIDE EFFECTS:          NONE
0281    655 ;
0281    656 ; COMPLETION CODES:      N/A
0281    657 ;
0281    658 ; CONDITION CODES:
0281    659 ;
0281    660 ;           Z-bit   Set if Exact match, either string1 or string exhausted
0281    661 ;           N-bit   Set if NO characters matched
0281    662 ;           C-bit   unpredictable
0281    663 ;           V-bit   unpredictable
0281    664 ;--
```

```
                    0281   666 SCAN$COMPARE::
     54   54   3C   0281   667          MOVZWL  R4,R4                  ; Trim length
          50   D4   0284   668          CLRL    R0                     ; Clear length of matched string
          52   D5   0286   669          TSTL    R2                     ; Any in string1?
          15   13   0288   670          BEQL    20$                    ; Branch if none
          54   D5   028A   671 10$:     TSTL    R4                     ; Any in string2?
          11   13   028C   672          BEQL    20$                    ; Branch if string2 exhausted
     65   63   91   028E   673          CMPB    (R3),(R5)              ; Compare a byte
          0C   12   0291   674          BNEQ    20$                    ; Branch if no match
          50   D6   0293   675          INCL    R0                     ; Count character in match
          53   D6   0295   676          INCL    R3                     ; Update string1 address
          55   D6   0297   677          INCL    R5                     ; Update string2 address
                    0299   678
          54   D7   0299   679          DECL    R4                     ; Count down characters in string2
          52   D7   029B   680          DECL    R2                     ; Count down characters in string1
          EB   14   029D   681          BGTR    10$                    ; Branch if string1 NOT exhausted
                    029F   682 20$:
     01   50   D1   029F   683          CMPL    R0,#1                  ; Set condition codes from match length
               05   02A2   684          RSB
```

```
                        02A3    686          .SBTTL  SCAN$NUMERIC    Scan number
                        02A3    687 ;++
                        02A3    688 ; FUNCTIONAL DESCRIPTION:
                        02A3    689 ;
                        02A3    690 ;       This routine can be called to scan and convert a ascii
                        02A3    691 ;       number to a signed, quadword value:
                        02A3    692 ;       The input form is: [+!-] [% B!b!D!d!O!o!X!x] [+!-]<digit-list>
                        02A3    693 ;       If the radix override is not present then the input radix is used for
                        02A3    694 ;       conversion.
                        02A3    695 ;
                        02A3    696 ; CALLING SEQUENCE:
                        02A3    697 ;
                        02A3    698 ;       BSBW    SCAN$NUMERIC
                        02A3    699 ;
                        02A3    700 ; INPUT PARAMETERS:
                        02A3    701 ;
                        02A3    702 ;       R3      Radix for conversion
                        02A3    703 ;       R4      Length of input string
                        02A3    704 ;       R5      Address of input string
                        02A3    705 ;
                        02A3    706 ; IMPLICIT INPUTS:      NONE
                        02A3    707 ;
                        02A3    708 ; OUTPUT PARAMETERS:
                        02A3    709 ;
                        02A3    710 ;       R0/R1   Signed Quadword value, both zero if no conversion
                        02A3    711 ;       R3      Updated conversion radix
                        02A3    712 ;       R4      Updated length of string
                        02A3    713 ;       R5      Updated address of string
                        02A3    714 ;
                        02A3    715 ; CONDITION CODES:
                        02A3    716 ;
                        02A3    717 ;       Z-bit   Set if No digits scanned
                        02A3    718 ;       N-Bit   Unpredictable
                        02A3    719 ;       C-bit   Unpredictable
                        02A3    720 ;       V-bit   Unpredictable
                        02A3    721 ;
                        02A3    722 ; IMPLICIT OUTPUTS:     NONE
                        02A3    723 ;
                        02A3    724 ; COMPLETION CODES:     N/A
                        02A3    725 ;
                        02A3    726 ; SIDE EFFECTS:         NONE
                        02A3    727 ;
                        02A3    728 ; REGISTER USAGE:
                        02A3    729 ;
                        02A3    730 ;       R0/R1   Signed quadword value
                        02A3    731 ;       R2      Current character
                        02A3    732 ;       R3      Radix for conversion
                        02A3    733 ;       R4      Length remaining in string
                        02A3    734 ;       R5      Address of next avail character
                        02A3    735 ;--
                        02A3    736
                        02A3    737 DS$GT_VHEX::
0000007E 0000007E 03FF0000 00000000
                        02A3    738          .LONG   0,^X03FF0000,^X7E,^X7E
                        02B3    739 BASE_LIST:
08 0A 02 00 78 6F 64 62 58 4F 44 42
                  10 08 0A 02 10
                        02B3    740          .ASCII  "BDOXbdox"<0><2><10><8><16><2><10><8><16>
                        02BF
```

```
                                02C4   SCAN$DECIMAL::
                53   0A   D0   02C4           MOVL    #10,R3                  ; Set radix
                     0D   11   02C7           BRB     SCAN$NUMERIC
                                02C9
                                02C9   SCAN$OCTAL::
                53   08   D0   02C9           MOVL    #8,R3                   ; Set radix
                     08   11   02CC           BRB     SCAN$NUMERIC
                                02CE
                                02CE   SCAN$BINARY::
                53   02   D0   02CE           MOVL    #2,R3                   ; Set radix
                     03   11   02D1           BRB     SCAN$NUMERIC
                                02D3
                                02D3   SCAN$HEX::
                53   10   D0   02D3           MOVL    #16,R3                  ; Set radix
                                02D6
                                02D6   SCAN$NUMERIC::
                54   54   3C   02D6           MOVZWL  R4,R4                   ; Trim length
                     30   BB   02D9           PUSHR   #^M<R4,R5>              ; Save in case have to backup
                                02DB
                     5D   10   02DB           BSBB    CHECK_SIGN              ; Check for sign value
                     50   DD   02DD           PUSHL   R0                      ; Save sign, 0=positive,1=negative
                     70   10   02DF           BSBB    CHECK_RADIX             ; Set R3 from possible radix change
                     57   10   02E1           BSBB    CHECK_SIGN              ; Check for sign value again
                6E   50   C8   02E3           BISL    R0,(SP)                 ; Set negative if sign encountered
                                02E6
                                02E6   CHECK_DIGIT:
                     55   DD   02E6           PUSHL   R5                      ; Save current position
                     50   7C   02E8           CLRQ    R0                      ; Clear accumulator
                     2D   11   02EA           BRB     70$                     ; Start by counting characters
                52   65   9A   02EC   40$:    MOVZBL  (R5),R2                 ; Get the character
           28 B0 AF   52   E1   02EF           BBC     R2,DS$GT_VHEX,80$       ; Finish if not valid digit
              03 52   06   E1   02F4           BBC     #6,R2,50$               ; Branch if not A-F
                52   09   C0   02F8           ADDL    #9,R2                   ; Offset "A" to %XA
              52   F0 8F   8A   02FB   50$:    BICB    #^C15,R2                ; Clear extra bits
                53   52   D1   02FF           CMPL    R2,R3                   ; Compare with radix
                     18   18   0302           BGEQ    80$                     ; Assume end of number
                     55   D6   0304           INCL    R5                      ; Update string address
           7E   51   53   C5   0306           MULL3   R3,R1,-(SP)             ; Radix * high order
                     50   D5   030A           TSTL    R0                      ; If low order sign bit set
                     03   18   030C           BGEQ    60$                     ; Branch if not
                6E   53   C0   030E           ADDL    R3,(SP)                 ; Compinsate for unsiged arith
      50   52   50   53   7A   0311   60$:    EMUL    R3,R0,R2,R0             ; Radix * old number + new digit
                51   8E   C0   0316           ADDL    (SP)+,R1                ; Include extra high part
                   D0 54   F4   0319   70$:    SOBGEQ  R4,40$                  ; Branch if another character present
                     54   D6   031C   80$:    INCL    R4                      ; Restore count
                     04   BA   031E           POPR    #^MR2                   ; Get position when started digits
                   0B 8E   E9   0320           BLBC    (SP)+,90$               ; Branch if No sign or "+" sign
                50   50   D2   0323           MCOML   R0,R0                   ; Complement Low order
                51   51   D2   0326           MCOML   R1,R1                   ; Complement High order
                     50   D6   0329           INCL    R0                      ; Inc Low order = Negated
                51   00   D8   032B           ADWC    #0,R1                   ; Inc High order with Carry from low
                5E   08   C0   032E   90$:    ADDL    #8,SP                   ; Remove R4/R5 from stack
                55   52   D1   0331           CMPL    R2,R5                   ; Set Z-bit from number of digits
                          05   0334           RSB                             ; Return to caller
                                0335   INVALID:
                     50   7C   0335           CLRQ    R0                      ; Clear R0/R1, clear V-bit
                     30   BA   0337           POPR    #^M<R4,R5>              ; Restore registers
```

```
                     05  0339    799          RSB                             ; Return to caller
                         033A    800
                         033A    801 CHECK_SIGN:
                 50  D4  033A    802          CLRL    R0                      ; Assume positive
                 54  D5  033C    803          TSTL    R4                      ; Check for sign character
                 10  15  033E    804          BLEQ    40$                     ; Branch if no sign, therefore no digits
              65 2B  91  0340    805          CMPB    #^A"+",(R5)             ; Positive?
                 07  13  0343    806          BEQL    30$                     ; Yes, use and skip
              65 2D  91  0345    807          CMPB    #^A"-",(R5)             ; Negative?
                 06  12  0348    808          BNEQ    40$                     ; No, Not sign
                 50  D7  034A    809          DECL    R0                      ; Mark negative
                         034C    810 30$:
                 55  D6  034C    811          INCL    R5                      ; Remove character
                 54  D7  034E    812          DECL    R4                      ; Remove character
                         0350    813 40$:
                     05  0350    814          RSB                             ; Return
                         0351    815
                         0351    816 CHECK_RADIX:
              02 54  D1  0351    817          CMPL    R4,#2                   ; Enough chars for radix mod?
                 1B  19  0354    818          BLSS    30$                     ; No, skip check
              65 25  91  0356    819          CMPB    #^A"%",(R5)             ; Start of radix modifier?
                 16  12  0359    820          BNEQ    30$                     ; No, Finish and exit
           52 FF54 CF  9E  035B    821          MOVAB   BASE_LIST,R2            ; Address of valid base list
            01 A5  82  91  0360    822 10$:     CMPB    (R2)+,1(R5)             ; match?
                 FA  1F  0364    823          BLSSU   10$                     ; Continue unless end of table
                 09  12  0366    824          BNEQ    30$                     ; Return invalid if no match here
           53  08 A2  9A  0368    825 20$:     MOVZBL  8(R2),R3                ; Fetch modified base
              54 02  C2  036C    826          SUBL2   #2,R4                   ; Decrement by characters used
                 85  B5  036F    827          TSTW    (R5)+                   ; Pass over characters used
                     05  0371    828 30$:     RSB                             ; Return
```

```
0372    830          .SBTTL  SCAN$SEARCHLIST Search list for unique string
0372    831 ;++
0372    832 ; FUNCTIONAL DESCRIPTION:
0372    833 ;
0372    834 ;       This routine can be used to validate a keyword in a list of the form:
0372    835 ;       .WORD   count,offset-item1,offset-item2.....
0372    836 ;       The offsets are self relative, i.e. ITEM-.
0372    837 ;
0372    838 ; CALLING SEQUENCE:
0372    839 ;
0372    840 ;       BSBW    SCAN$SEARCHLIST
0372    841 ;
0372    842 ; INPUT PARAMETERS:
0372    843 ;
0372    844 ;       R0      Length of string to be searched for
0372    845 ;       R1      Address of string to be search for
0372    846 ;       R3      Address of Keyword table
0372    847 ;
0372    848 ; IMPLICIT INPUTS:      NONE
0372    849 ;
0372    850 ; OUTPUT PARAMETERS:
0372    851 ;
0372    852 ;       R0      Index of first string that matched
0372    853 ;       R1      Address of ASCIC string that matched
0372    854 ;       R2-R5   Preserved across call
0372    855 ;
0372    856 ; IMPLICIT OUTPUTS:     NONE
0372    857 ;
0372    858 ; COMPLETION CODES:     N/A
0372    859 ;
0372    860 ; CONDITION CODES:
0372    861 ;
0372    862 ;       Z-Bit   Set if exactly 1 match
0372    863 ;       N-bit   Set if no matchs
0372    864 ;       V-bit   zero
0372    865 ;       C-bit   zero
0372    866 ;
0372    867 ; SIDE EFFECTS:         NONE
0372    868 ;--
```

```
                                0372   870 SCAN$SEARCHLIST::
          50    50    3C   0372   871          MOVZWL  R0,R0                      ; Trim length
                3F    BB   0375   872          PUSHR   #^M<R0,R1,R2,R3,R4,R5>     ; Save registers
                7E    7C   0377   873          CLRQ    -(SP)                      ; Space for Index 0(SP), Address 4(SP)
                7E    D4   0379   874          CLRL    -(SP)                      ; Count of matches
                                037B   875
          52    83    3C   037B   876          MOVZWL  (R3)+,R2                   ; Get count of strings to match
       53    6342    3E   037E   877          MOVAW   (R3)[R2],R3                ; Point past last word
                2E    11   0382   878          BRB     60$                        ; Start by counting
                                0384   879 10$:
          55    73    32   0384   880          CVTWL   -(R3),R5                   ; Get offset to ASCIC string
                29    13   0387   881          BEQL    60$                        ; Skip it if offset=0
          55    53    C0   0389   882          ADDL    R3,R5                      ; Make it absolute
          54    85    9A   038C   883          MOVZBL  (R5)+,R4                   ; Fetch length of ASCIC string
                21    13   038F   884          BEQL    60$                        ; Branch if null string, try next
                                0391   885 20$:
       50    0C AE    7D   0391   886          MOVQ    12(SP),R0                  ; Restore original len,addr
          54    50    D1   0395   887          CMPL    R0,R4                      ; compare
                18    14   0398   888          BGTR    60$                        ; Branch if input longer than string
                05    11   039A   889          BRB     40$                        ; Start by counting character
                                039C   890 30$:
          85    81    91   039C   891          CMPB    (R1)+,(R5)+                ; Compare a byte
                11    12   039F   892          BNEQ    60$                        ; Mismatch, try next string
             F8 50    F4   03A1   893 40$:     SOBGEQ  R0,30$                     ; Branch if more characters in input
                6E    D6   03A4   894          INCL    (SP)                       ; Count this match
       04 AE    52    D0   03A6   895          MOVL    R2,4(SP)                   ; Save index of string match
          55    63    32   03AA   896          CVTWL   (R3),R5                    ; Get offset to ASCIC
   08 AE  55    53    C1   03AD   897          ADDL3   R3,R5,8(SP)                ; Save address of match entry
                                03B2   898 60$:
             CF 52    F4   03B2   899          SOBGEQ  R2,10$                     ; If not last string, do next
                                03B5   900
          50    8E    D0   03B5   901          MOVL    (SP)+,R0                   ; Get count of matches
          6E    8E    7D   03B8   902          MOVQ    (SP)+,(SP)                 ; Remove saved R0/R1
          01    50    D1   03BB   903          CMPL    R0,#1                      ; set condition codes from # of matchs
                3F    BA   03BE   904          POPR    #^M<R0,R1,R2,R3,R4,R5>     ; index, address, restore R2,R3,R4,R5
                      05   03C0   905          RSB                                ; Return
```

```
                        907          .SBTTL  SCAN$DEVICE     Scan a device name
03C1    908 ;++
03C1    909 ; FUNCTIONAL DESCRIPTION:
03C1    910 ;
03C1    911 ;       This routine can be used to scan a device name of the form:
03C1    912 ;       ggan[:] , gg is generic device name a is single letter A-z,
03C1    913 ;       and n is a decimal number in the range 0-255.
03C1    914 ;
03C1    915 ; CALLING SEQUENCE:
03C1    916 ;
03C1    917 ;       BSBW    SCAN$DEVICE
03C1    918 ;
03C1    919 ; INPUT PARAMETERS:
03C1    920 ;
03C1    921 ;       R4      Length of input string to scan
03C1    922 ;
03C1    923 ;       R5      Address of input string to be scanned
03C1    924 ;
03C1    925 ; IMPLICIT INPUTS:      NONE
03C1    926 ;
03C1    927 ; OUTPUT PARAMETERS:
03C1    928 ;
03C1    929 ;       R0      <0,16> length of name scanned, <16,16> unit number
03C1    930 ;       R1      Address of device name
03C1    931 ;       R4      Updated input string length
03C1    932 ;       R5      Updated input string address
03C1    933 ;
03C1    934 ; COMPLETION CODES:
03C1    935 ;
03C1    936 ; CONDITION CODES:
03C1    937 ;
03C1    938 ;       Z-bit   Set if no device present
03C1    939 ;
03C1    940 ; IMPLICIT OUTPUTS:     NONE
03C1    941 ;
03C1    942 ; SIDE EFFECTS:         NONE
03C1    943 ;
03C1    944 ;--
```

```
                                03C1   SCAN$DEVICE::
          54    54    3C    03C1          MOVZWL  R4,R4                  ; Trim length
                30    BB    03C4          PUSHR   #^M<R4,R5>             ; Save input pointers
                30    BB    03C6          PUSHR   #^M<R4,R5>             ; Save input pointers
                55    D6    03C8          INCL    R5                     ; Point past the first          [16]
                54    D7    03CA          DECL    R4                     ; Character                     [16]
             FE5F    30    03CC          BSBW    SCAN$ALPHANUM          ; check name                    [15]
          24    65    91    03CF          CMPB    (R5),#^A"$"            ; is this a long device name ?  [15]
                08    12    03D2          BNEQ    5$                     ; if not then continue          [15]
                55    D6    03D4          INCL    R5                     ; else skip '$'                 [15]
                54    D7    03D6          DECL    R4                     ; skip character                [15]
                8E    7C    03D8          CLRQ    (SP)+                  ; reset stack                   [15]
                02    11    03DA          BRB     7$                     ; go check generic              [15]
                            03DC   5$:                                   ;                               [15]
                30    BA    03DC          POPR    #^M<R4,R5>             ; restore original registers    [15]
                            03DE   7$:                                   ;                               [15]
             FE3B    30    03DE          BSBW    SCAN$ALPHA             ; Get an alpha field
                23    13    03E1          BEQL    40$                    ; Branch if not present
             FEDE    30    03E3          BSBW    SCAN$DECIMAL           ; Scan a decimal number
                03    12    03E6          BNEQ    10$                    ; Branch if number scanned
          50    01    CE    03E8          MNEGL   #1,R0                  ; Force sign bit on
   6E   55    04 AE    C3    03EB   10$:   SUBL3   4(SP),R5,(SP)          ; Length of name is new position- old
       02 AE    50    B0    03F0          MOVW    R0,2(SP)               ; Save unit number in top of length
                54    D7    03F4          DECL    R4                     ; Subtract for length of ":"
                07    19    03F6          BLSS    20$                    ; Branch if no character there
          3A    85    91    03F8          CMPB    (R5)+,#^A":"           ; Check for optional ":"
                04    13    03FB          BEQL    30$                    ; Branch if present
                55    D7    03FD          DECL    R5                     ; Fix address
                54    D6    03FF   20$:   INCL    R4                     ; Fix length
                03    BA    0401   30$:   POPR    #^M<R0,R1>             ; Get length, address of device name
                50    B5    0403          TSTW    R0                     ; Set condition codes, from length
                      05    0405          RSB                            ; Return to caller
                            0406   40$:
                50    7C    0406          CLRQ    R0                     ; Clear length, address
                30    BA    0408          POPR    #^M<R4,R5>             ; restore pointers
                      05    040A          RSB                            ; Return
                            040B
```

```
                                040B  .Subtitle       Breakup File name into parts
                                040B
                                040B  ;++
                                040B  ; Function:
                                040B  ;       Break up fields of a filename spec so that they can be merged by
                                040B  ;       repeated calls to this routine.  If any of the filespec fields are
                                040B  ;       present, the descriptor for that field is overlayed over the
                                040B  ;       appropriate field of the output file block
                                040B  ;
                                040B  ; Inputs:
                                040B  ;       04(Ap)  Length of filename string
                                040B  ;       08(Ap)  Address of filename string
                                040B  ;       12(Ap)  Address of file block:
                                040B  ;
                                040B  ; Output:
                                040B  ;       12(Ap)  Address of file block:
                                040B  ;                       (00) :  Descriptor of device part (incl. ':')
                                040B  ;                       (08) :  Descriptor of directory part (incl. '[' & ']')
                                040B  ;                       (16) :  Descriptor of filename part
                                040B  ;                       (24) :  Descriptor of file type part (incl. '.')
                                040B  ;                       (32) :  Descriptor of version part (incl. ';')
                                040B  ;
                                040B  ;       ALTERNATE ULTRIX FILE STRUCTURE in file block:
                                040B  ;
                                040B  ;                       (00) :  Descriptor of device part (incl. ':')
                                040B  ;                       (08) :  Descriptor of partition (incl '(' and ')')
                                040B  ;                       (16) :  Descriptor for directory (includes imbeded '/' )
                                040B  ;                       (24) :  Descriptor for file name.(only if none of the above
                                040B  ;                                                       was not specified)
                                040B  ;--
                                040B
                                040B  OverLay:                                          ; Utility routine                [11]
     52    54    50    C3       040B          SubL3   R0, R4, R2                        ; Calculate length of field      [11]
                52    D6       040F          IncL    R2                                ; .. include terminator          [11]
          53    55    D0       0411          MovL    R5, R3                            ; Field starts at 'pivot'        [11]
        6647    52    7D       0414          MovQ    R2, (R6)[R7]                      ; Move to correct descriptor     [11]
     55    51    01    C1       0418          AddL3   #1, R1, R5                        ; Update string pointer          [11]
     54    50    01    C3       041C          SubL3   #1, R0, R4                        ; .. and length                  [11]
                      05       0420          Rsb                                       ; And return                     [11]
                               0421
                    00FC       0421  .Entry BreakUp_File, ^M<R2, R3, R4, R5, R6, R7> ; Breakup_File routine           [11]
     54    04 AC    D0       0423          MovL    04(Ap), R4                        ; Get length of filespec         [11]
                59    15       0427          BLeq    10$                               ; Skip ahead (exit)              [11]
     55    08 AC    D0       0429          MovL    08(Ap), R5                        ; Get address                    [11]
     56    0C AC    D0       042D          MovL    12(Ap), R6                        ; Get address of file block      [11]
                7E    94       0431          CLRB    -(SP)                             ; Clear a flag for ODS           [19]
 65    54    5B 8F    3A       0433          LOCC    #^A"[", R4, (R5)                  ; Find how to parse string       [19]
                1A    12       0438          BNEQ    8$                                ; Branch to treat as ODS         [19]
     65    54    3C    3A       043A          LOCC    #^A"<", R4, (R5)                  ; Find how to parse string       [19]
                14    12       043E          BNEQ    8$                                ; Branch to treat as ODS         [19]
     65    54    28    3A       0440          LOCC    #^A"(", R4, (R5)                  ; Find how to parse string       [19]
                0C    12       0444          BNEQ    7$                                ; Branch to treat as ULTRIX      [19]
      00000000'8F    E1       0446          BBC     #ULTRIX_V_MODE,-                  ; As a last resort use current   [19]
  02 00000000'EF              044C                  L^ULTRIX_FLAGS,8$                 ; State for parsing              [19]
                6E    96       0452  7$:     INCB    (SP)                              ; Flag as ULTRIX                 [19]
          7E    01    90       0454  8$:     movb    #1,-(sp)                          ; Flag to assume ULTRIX filename[19]
     65    54    3A    3A       0457          LocC    #^A":", R4, (R5)                  ; Scan for device part           [11]
```

```
                     25   13   045B  1041          BEql    10$                                     ; Branch if none                        [11]
                     6E   94   045D  1042          clrb    (sp)                                    ; Signal, don't treat as file           [19]
                     57   D4   045F  1043          ClrL    R7                                      ; Index                                 [11]
                     A8   10   0461  1044          BsbB    OverLay                                 ; Update pointers/etc.                  [11]
          00000000'8F     E1   0463  1045          BBC     #ULTRIX_V_MODE,-                        ; Skip if not ULTRIX                    [20]
       13 00000000'EF          0469  1046                  L^ULTRIX_FLAGS,10$                      ;                                       [20]
                     52   D7   046F  1047          decl    R2                                      ; Adjust length                         [20]
        09 6342      06   E1   0471  1048 88$:     BBC     #6,(R3)[R2],9$                          ; SKIP IF NOT ALPHA CHAR                [20]
        04 6342      05   E1   0476  1049          BBC     #5,(R3)[R2],9$                          ; SKIP IF UPPER CASE. DON'T WRITE IN
                                0478  1050                                                          ; WRITE PROTECTED [20]
           6342      20   8A   047B  1051          BICB2   #32,(R3)[R2]                            ; FORCE Device name TO UPPER CASE[20
              EF 52    F4   047F  1052 9$:         SOBGEQ  R2,88$                                  ; CHECK TOTAL SECTION                   [20]
                                0482  1053
                     54   D5   0482  1054 10$:     TstL    R4                                      ; More String?                          [11]
                     03   14   0484  1055          BGTR    15$                                     ;                                       [17]
                  00C1    31   0486  1056          BRW     BreakUp_File_X                          ; If not, all done                      [11]
                50   65   9A   0489  1057 15$:     MovZBL  (R5), R0                                ; Get next character                    [11]
        47 01 AE     00   E1   048C  1058          BBC     #0,1(SP),60$                            ; Branch if ODS                         [19]
                28   50   91   0491  1059          CmpB    R0, #^A"("                              ; If it's a partition start             [19]
                     13   12   0494  1060          Bneq    40$                                     ; .. character, then                    [19]
                     50   D6   0496  1061 20$:     INCL    R0                                      ; Closing ')'                           [17]
             65      54   50   3A   0498  1062          LOCC    R0,R4,(R5)                          ; Find closing character                [17]
                     03   12   049C  1063          BNEQ    30$                                     ; Branch if found                       [17]
                  00A9    31   049E  1064          BRW     BreakUp_File_X                          ; If not, exit                          [17]
                     6E   94   04A1  1065 30$:     clrb    (sp)                                    ; Signal, don't treat as file           [19]
                50   01   D0   04A3  1066          MovL    #1, R7                                  ; Setup index                           [11]
                  FF62    30   04A6  1067          BSBW    OVERLAY                                 ; Update pointer/move data              [17]
             65      54   2F   3A   04A9  1068 40$:     LOCC    #^A"/",R4,(R5)                      ; Check if "directory" spec             [19]
                     07   13   04AD  1069          BEQL    50$                                     ; Branch if not "directory"             [19]
                10 A6     54   7D   04AF  1070 45$:     MOVq    R4,16(R6)                           ; Let remainder be directory            [17]
                        0094    31   04B3  1071          Brw     BreakUp_File_X                      ; exit                                  [19]
                     6E   95   04B6  1072 50$:     TSTB    (SP)                                    ; Check if file or directory            [19]
                     F5   13   04B8  1073          beql    45$                                     ; Branch if treat as directory          [19]
             18 A6     54   01   C1   04BA  1074          ADDL3   #1,R4,24(R6)                       ; Let it be file name.                  [19]
        00000009'EF    65   54   28   04BF  1075          MOVC3   R4,(R5),Ultrix_filename+1           ; Shift over the string                 [19]
           00000008'EF  2F   90   04C7  1076          MOVB    #^A"/", Ultrix_filename                 ; Put delimiter in front                [19]
         1C A6   00000008'EF  9E   04CE  1077          MOVAB   Ultrix_filename,28(R6)              ; Set address of shifted string         [19]
                     72   11   04D6  1078          Brb     BreakUp_File_X                          ; exit                                  [19]
              5B 8F  50   91   04D8  1079 60$:     CmpB    R0, #^A"["                              ; If it's a directory start             [11]
                     05   13   04DC  1080          BEql    80$                                     ; .. character, then                    [11]
                3C   50   91   04DE  1081          CmpB    R0, #^A"<"                              ; .. use it                             [11]
                     11   12   04E1  1082          BNeq    100$                                    ; ..                                    [11]
                50   02   80   04E3  1083 80$:     AddB2   #2, R0                                  ; Form closing character                [11]
             65      54   50   3A   04E6  1084          LocC    R0, R4, (R5)                        ; Is there a corresponding              [11]
                     02   12   04EA  1085          BNeq    90$                                     ; .. closing char (']' or '>')?         [11]
                     5C   11   04EC  1086          Brb     BreakUp_File_X                          ; If not, exit                          [11]
                57   01   D0   04EE  1087 90$:     MovL    #1, R7                                  ; Setup index                           [11]
                  FF17    30   04F1  1088          BsbW    OverLay                                 ; If it's OK, use it                    [11]
             65      54   2E   3A   04F4  1089 100$:    LocC    #^A".", R4, (R5)                    ; Is there a file type?                 [11]
                55   51   D1   04F8  1090          CmpL    R1, R5                                  ; If yes, and no name, go               [11]
                     24   13   04FB  1091          BEql    120$                                    ;                                       [11]
                     50   D5   04FD  1092          TstL    R0                                      ; Was it found at all?                  [11]
                     09   12   04FF  1093          BNeq    110$                                    ; If so, get file name                  [11]
             65      54   3B   3A   0501  1094          LocC    #^A";", R4, (R5)                    ; Was there a version?                  [11]
                55   51   D1   0505  1095          CmpL    R1, R5                                  ; If it's right here,                   [11]
                     38   13   0508  1096          BEql    140$                                    ; .. use it as filename term.           [11]
                                050A  1097 110$:                                                    ; R1 points past end of name            [11]
```

```
                52   51   55   C3   050A  1098          SubL3   R5, R1, R2                      ; Get name length                [11]
                     53   55   D0   050E  1099          MovL    R5, R3                          ; Address of start of name       [11]
                  10 A6   52   7D   0511  1100          MovQ    R2, 16(R6)                      ; Move to output                 [11]
                     54   50   7D   0515  1101          MovQ    R0, R4                          ; Skip to terminator             [11]
                          54   D5   0518  1102          TstL    R4                              ; Any file left?                 [11]
                          2E   15   051A  1103          BLeq    BreakUp_File_X                  ; If not, exit now               [11]
                     65   2E   91   051C  1104          CmpB    #^A".", (R5)                    ; Is there a type?               [11]
                          21   12   051F  1105          BNeq    140$                            ; If not, check version          [11]
                               0521  1106 120$:                                                 ; Next character is '.'; type    [11]
                          54   D7   0521  1107          DecL    R4                              ; Skip past '.' to check for     [11]
                          25   19   0523  1108          BLss    BreakUp_File_X                  ; Exit if no more                [11]
                          55   D6   0525  1109          IncL    R5                              ; .. '.' or ';' for version      [11]
                65   54   3B   3A   0527  1110          LocC    #^A";", R4, (R5)                ; Is there a version?            [11]
                          04   12   052B  1111          BNeq    130$                            ; Yep, go off                    [11]
                65   54   2E   3A   052D  1112          LocC    #^A".", R4, (R5)                ; Alternate syntax for version   [11]
                               0531  1113                                                       ; R0, R1 at version or end       [11]
                52   51   55   C3   0531  1114 130$:     SubL3   R5, R1, R2                      ; Calculate length of            [11]
                          52   D6   0535  1115          IncL    R2                              ; .. the file type string        [11]
                53   55   01   C3   0537  1116          SubL3   #1, R5, R3                      ; And it's address               [11]
                  18 A6   52   7D   053B  1117          MovQ    R2, 24(R6)                      ; And move it to output          [11]
                     54   50   7D   053F  1118          MovQ    R0, R4                          ; Update to terminator/end       [11]
                          54   D5   0542  1119 140$:     TstL    R4                              ; Is there more file?            [11]
                          04   15   0544  1120          BLeq    BreakUp_File_X                  ; If not, exit                   [11]
                  20 A6   54   7D   0546  1121          MovQ    R4, 32(R6)                      ; Use rest of string as version  [11]
                               054A  1122 BreakUp_File_X:                                       ; Exit routine                   [11]
                     50   01   D0   054A  1123          MovL    #1, R0                          ; Set up status return           [11]
                          04   054D  1124          Ret                                     ; And then return, of course    [11]
                               054E  1125
                               054E  1126          .END
```

```
$$ARGS                          = 00000004     D
$$T1                            = 00000014     D
ACTION                            000000A1 R   D  03
BASE_LIST                         000002B3 R   D  03
BRANCH                            00000099 R   D  03
BRANCH_IF_NONE                    000000E6 R   D  03
BREAKUP_FILE                      00000421 RG  D  03
BREAKUP_FILE_X                    0000054A R   D  03
CHECK_DIGIT                       000002E6 R   D  03
CHECK_RADIX                       00000351 R   D  03
CHECK_SIGN                        0000033A R   D  03
CLI$K_ALNUM                     = 00000087     D
CLI$K_ALPHA                     = 00000086     D
CLI$K_BIF                       = 00000083     D
CLI$K_BIFS                      = 00000094     D
CLI$K_BR                        = 00000082     D
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_CALL                      = 00000092     D
CLI$K_COMMA                     = 0000008E     D
CLI$K_DEC                       = 0000008A     D
CLI$K_EOL                       = 00000091     D
CLI$K_ERROR                     = 00000080     D
CLI$K_EXIT                      = 00000081     D
CLI$K_FILE                      = 00000095     D
CLI$K_HEX                       = 00000089     D
CLI$K_KEYWORD                   = 0000008C     D
CLI$K_NUM                       = 00000085     D
CLI$K_OCT                       = 00000088     D
CLI$K_RETURN                    = 00000093     D
CLI$K_SIZE                        0000044C     D
CLI$K_SLASH                     = 00000090     D
CLI$K_SPACE                     = 00000084     D
CLI$K_STRING                    = 0000008B     D
CLI$K_SYMBOL                    = 0000008D     D
CLI$K_VALUE                     = 0000008F     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_FLAGS2                      00000444     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEMP_BASE                   00000448     D
CLI$L_TEST                        00000020     D
CLI$M_START_OR_RUN              = 00000008     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BASE_QUAL                 = 00000002     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NEXT                      = 00000001     D
CLI$V_NOALLOC                   = 00000000     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_START_OR_RUN              = 00000003     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
DO_ACTION                         000000E8 R   D  03
DS$GL_CLIBASE                     ********     X  03
DS$GL_RADIX                       ********     X  03
DS$GT_VALPHA                      0000021E RG  D  03
DS$GT_VALPHANUM                   00000230 RG  D  03
DS$GT_VFILE                       00000242 RG  D  03
DS$GT_VHEX                        000002A3 RG  D  03
DS$GT_VSPACE                      00000254 RG  D  03
DS$GT_VSYMBOL                     0000020C RG  D  03
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
```

```
DS$M_DONFLG                     = 00002000    D     DS$_FHWE                        = 00660068    D
DS$M_ERRFLG                     = 00000010    D     DS$_FRAGBUF                     = 00660080    D
DS$M_EXCEPT                     = 00080000    D     DS$_ICBUSY                      = 006600C8    D
DS$M_EXETST                     = 00040000    D     DS$_ICERR                       = 006600C0    D
DS$M_HLTFLG                     = 00000008    D     DS$_IHWE                        = 00660060    D
DS$M_LODFLG                     = 00000002    D     DS$_ILLCHAR                     = 00660018    D
DS$M_MEMMGT                     = 00008000    D     DS$_ILLPAGCNT                   = 00660078    D
DS$M_NI                         = 04000000    D     DS$_ILLUNIT                     = 00660100    D
DS$M_OUTPUT                     = 00800000    D     DS$_INIT_FAIL                   = 00660140    D
DS$M_RUBFLG                     = 00000020    D     DS$_INSFMEM                     = 00660050    D
DS$M_SCRIPT                     = 00200000    D     DS$_INVCPU                      = 00660130    D
DS$M_SETIMR                     = 02000000    D     DS$_IPL2HI                      = 006600B8    D
DS$M_STRFLG                     = 00000004    D     DS$_IVADDR                      = 00660040    D
DS$M_SUBT                       = 00004000    D     DS$_IVVECT                      = 00660038    D
DS$M_SYSFLG                     = 00000400    D     DS$_KRNLSTK                     = 00660090    D
DS$M_TIMED                      = 02000000    D     DS$_LOGIC                       = 00660070    D
DS$M_TIMED_OUT                  = 08000000    D     DS$_MCHK                        = 00660088    D
DS$M_TIMRON                     = 00020000    D     DS$_MEM_ALLOC_ERR               = 00660138    D
DS$V_ABRTFLG                    = 00000006    D     DS$_MMOFF                       = 00660058    D
DS$V_BADTIME                    = 00000014    D     DS$_NEEDUNIT                    = 006600F8    D
DS$V_BATCH                      = 00000016    D     DS$_NODE                        = 00660128    D
DS$V_BRKCLR                     = 0000000C    D     DS$_NOPCS                       = 00660110    D
DS$V_BRKPT                      = 0000000B    D     DS$_NORMAL                      = 00660001    D
DS$V_CHARFLG                    = 00000008    D     DS$_NOSUPPORT                   = 006600B1    D
DS$V_CMDFLG                     = 00000007    D     DS$_NOTALLOWMP                  = 00660148    D
DS$V_CTRLC                      = 00000000    D     DS$_NOTDON                      = 00660030    D
DS$V_CTRLO                      = 00000010    D     DS$_NOTIMP                      = 006600B0    D
DS$V_DEVFLG                     = 00000009    D     DS$_NULLSTR                     = 00660010    D
DS$V_DISABLCC                   = 00000018    D     DS$_OVERFLOW                    = 00660008    D
DS$V_DONFLG                     = 0000000D    D     DS$_POWER                       = 00660098    D
DS$V_ERRFLG                     = 00000004    D     DS$_PROGERR                     = 00660020    D
DS$V_EXCEPT                     = 00000013    D     DS$_SEVERE                      = 00660004    D
DS$V_EXETST                     = 00000012    D     DS$_TRANSL                      = 006600A0    D
DS$V_HLTFLG                     = 00000003    D     DS$_TRUNCATE                    = 00660028    D
DS$V_LODFLG                     = 00000001    D     DS$_UNEXPINT                    = 006600D8    D
DS$V_MEMMGT                     = 0000000F    D     DS$_UNSUPPDEV                   = 00660158    D
DS$V_NI                         = 0000001A    D     DS$_VASFULL                     = 00660048    D
DS$V_OUTPUT                     = 00000017    D     DS$_WARNING                     = 00660000    D
DS$V_RUBFLG                     = 00000005    D     DSX$PARSE                       00000007 RG   D   03
DS$V_SCRIPT                     = 00000015    D     DSX$PARSE_X                     00000068 R    D   03
DS$V_SETIMR                     = 00000019    D     DSX$SUPERPARSE                  00000000 RG   D   03
DS$V_STRFLG                     = 00000002    D     EXIT                            00000065 R    D   03
DS$V_SUBT                       = 0000000E    D     HP$A_DEPENDENT                  00000032      D
DS$V_SYSFLG                     = 0000000A    D     HP$A_DEVICE                     00000018      D
DS$V_TIMED                      = 00000019    D     HP$A_DVA                        0000001C      D
DS$V_TIMED_OUT                  = 0000001B    D     HP$A_LINK                       00000020      D
DS$V_TIMRON                     = 00000011    D     HP$B_DRIVE                      0000000B      D
DS$_AP_NORMAL_BREAK             = 00660150    D     HP$B_FLAGS                      0000000A      D
DS$_ARITH                       = 006600D0    D     HP$Q_DEVICE                     00000000      D
DS$_ASBE                        = 00660118    D     HP$T_DEVICE                     0000000C      D
DS$_BADLINK                     = 006600F0    D     HP$T_TYPE                       00000026      D
DS$_BADTYPE                     = 006600E8    D     HP$W_SIZE                       00000008      D
DS$_BIIC                        = 00660120    D     HP$W_VECTOR                     00000024      D
DS$_CHME                        = 006600A8    D     INVALID                         00000335 R    D   03
DS$_CHMK                        = 006600E0    D     OVERLAY                         0000040B R    D   03
DS$_DEVNAME                     = 00660108    D     PARSE$_ACTION                   = 0000000C    D
DS$_ERROR                       = 00660002    D     PARSE$_BUFADR                   = 00000004    D
```

```
PARSE$_FLAG                     = 00000010     D
PARSE$_NARGS                    = 00000004     D
PARSE$_TREE                     = 00000008     D
PARSE_COMMON                      0000000B R   D   03
SAVE_CALL                         00000000 R   D   02
SAVE_STATUS                       00000004 R   D   02
SCAN$ALPHA                        0000021C RG  D   03
SCAN$ALPHANUM                     0000022E RG  D   03
SCAN$BINARY                       000002CE RG  D   03
SCAN$COMPARE                      00000281 RG  D   03
SCAN$DECIMAL                      000002C4 RG  D   03
SCAN$DEVICE                       000003C1 RG  D   03
SCAN$FILE                         00000240 RG  D   03
SCAN$HEX                          000002D3 RG  D   03
SCAN$NUMERIC                      000002D6 RG  D   03
SCAN$OCTAL                        000002C9 RG  D   03
SCAN$SEARCHLIST                   00000372 RG  D   03
SCAN$SPACE                        00000252 RG  D   03
SCAN$SPACES                       00000252 RG  D   03
SCAN$SPANC                        00000266 RG  D   03
SCAN$SYMBOL                       0000020A RG  D   03
SIZ...                          = 00000001     D
SPACE                           = 00000020     D
SS$_NORMAL                      = 00000001     D
TAB                             = 00000009     D
TRAVERSE                          0000001C R   D   03
TRVALNUM                          00000112 R   D   03
TRVALPHA                          0000010D R   D   03
TRVBIF                            000000DE R   D   03
TRVBIFS                           000000D2 R   D   03
TRVBR                             00000097 R   D   03
TRVCALL                           00000089 R   D   03
TRVCOMMA                          000001A2 R   D   03
TRVDEC                            000000FC R   D   03
TRVEOL                            000001EB R   D   03
TRVERR                            000000C7 R   D   03
TRVEXIT                           00000063 R   D   03
TRVFILE                           0000019C R   D   03
TRVHEX                            000000F7 R   D   03
TRVKEYWORD                        00000166 R   D   03
TRVNUM                            00000101 R   D   03
TRVNUMA                           00000108 R   D   03
TRVOCT                            000000F2 R   D   03
TRVRETURN                         00000069 R   D   03
TRVSLASH                          000001A6 R   D   03
TRVSPACE                          000000ED R   D   03
TRVSTRING                         00000117 R   D   03
TRVSYMBOL                         00000197 R   D   03
TRVVALUE                          000001AB R   D   03
TRV_COMMA_VALUE                   000001AE R   D   03
ULTRIX_FILENAME                   00000008 R   D   02
ULTRIX_FLAGS                      ********     X   03
ULTRIX_V_MODE                     ********     X   03
V_BIT                           = 00000002     D
```

```
                                    +----------------+
                                    ! Psect synopsis !
                                    +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           0000044C  ( 1100.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
WORK                            00000058  (    88.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
CODE                            0000054E  ( 1358.)  03 (  3.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC BYTE
```

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                      VALUE        DEFINITION      REFERENCES...
------                      -----        ----------      -------------
$$ARGS                     =00000004     137    (1)      137    (1)
$$T1                       =00000014     137    (1)      137    (1)
ACTION                      000000A1-R   306    (4)    #-262    (3)    #-273    (4)    #-294    (4)    #-321    (4)
                                                        #-330    (4)    #-339    (4)    #-347    (5)    #-410    (5)
                                                        #-430    (6)
BASE_LIST                   000002B3-R   739    (11)     821    (12)
BIT...                     =00000003     138    (1)      135    (1)      136    (1)      138    (1)
BRANCH                      00000099-R   299    (4)    #-254    (3)    #-331    (4)    #-340    (4)    #-345    (5)
                                                        #-436    (6)    #-519    (6)
BRANCH_IF_NONE              000000E6-R   344    (5)    #-355    (5)    #-372    (5)    #-378    (5)    #-384    (5)
                                                        #-444    (6)    #-450    (6)    #-504    (6)
BREAKUP_FILE                00000421-R   1024   (16)
BREAKUP_FILE_X              0000054A-R   1122   (16)   #-1056   (16)   #-1064   (16)   #-1071   (16)   #-1078   (16)
                                                        #-1086   (16)   #-1103   (16)   #-1108   (16)   #-1120   (16)
CHECK_DIGIT                 000002E6-R   767    (12)
CHECK_RADIX                 00000351-R   816    (12)   #-763    (12)
CHECK_SIGN                  0000033A-R   801    (12)   #-761    (12)   #-764    (12)
CLI$K_ALNUM                =00000087     135    (1)
CLI$K_ALPHA                =00000086     135    (1)
CLI$K_BIF                  =00000083     135    (1)
CLI$K_BIFS                 =00000094     135    (1)
CLI$K_BR                   =00000082     135    (1)
CLI$K_BUFSIZ               =00000100     134    (1)      134    (1)
CLI$K_CALL                 =00000092     135    (1)
CLI$K_COMMA                =0000008E     135    (1)
CLI$K_DEC                  =0000008A     135    (1)
CLI$K_EOL                  =00000091     135    (1)
CLI$K_ERROR                =00000080     135    (1)    #-248    (3)
CLI$K_EXIT                 =00000081     135    (1)
CLI$K_FILE                 =00000095     135    (1)
CLI$K_HEX                  =00000089     135    (1)
CLI$K_KEYWORD              =0000008C     135    (1)
CLI$K_NUM                  =00000085     135    (1)
CLI$K_OCT                  =00000088     135    (1)
CLI$K_RETURN               =00000093     135    (1)
CLI$K_SIZE                  0000044C     134    (1)
CLI$K_SLASH                =00000090     135    (1)
CLI$K_SPACE                =00000084     135    (1)
CLI$K_STRING               =0000008B     135    (1)
CLI$K_SYMBOL               =0000008D     135    (1)
CLI$K_VALUE                =0000008F     135    (1)
CLI$L_ADDRESS               00000018     134    (1)
CLI$L_COMMAND               00000004     134    (1)
CLI$L_DATA                  0000001C     134    (1)
CLI$L_FLAGS                 00000000     134    (1)
CLI$L_FLAGS2                00000444     134    (1)
CLI$L_LAST                  00000024     134    (1)
CLI$L_NEXT                  00000030     134    (1)
CLI$L_PASS                  0000002C     134    (1)
CLI$L_SUBT                  00000028     134    (1)
```

```
CLI$L_TEMP_BASE                  00000448     134     (1)
CLI$L_TEST                       00000020     134     (1)
CLI$M_START_OR_RUN              =00000008     134     (1)
CLI$Q_BUFQWD                     00000034     134     (1)
CLI$Q_FILE                       00000008     134     (1)
CLI$Q_SECTION                    00000010     134     (1)
CLI$Q_TIME                       0000043C     134     (1)
CLI$T_BUFFER                     0000003C     134     (1)
CLI$V_ADAPTER                   =00000018     134     (1)
CLI$V_ADR                       =0000000B     134     (1)
CLI$V_ASCII                     =00000013     134     (1)
CLI$V_BASE_QUAL                 =00000002     134     (1)
CLI$V_BREAK                     =0000000A     134     (1)
CLI$V_BRIEF                     =0000001B     134     (1)
CLI$V_BYTE                      =0000000D     134     (1)
CLI$V_CLEAR                     =00000002     134     (1)
CLI$V_DEC                       =00000010     134     (1)
CLI$V_DEFAULT                   =0000000C     134     (1)
CLI$V_DEPOSIT                   =00000019     134     (1)
CLI$V_EVENT                     =00000008     134     (1)
CLI$V_EXAM                      =00000005     134     (1)
CLI$V_FLAGS                     =00000009     134     (1)
CLI$V_HEX                       =00000012     134     (1)
CLI$V_KERNEL                    =00000017     134     (1)
CLI$V_LOAD                      =00000006     134     (1)
CLI$V_LONG                      =0000000F     134     (1)
CLI$V_NEXT                      =00000001     134     (1)
CLI$V_NOALLOC                   =00000000     134     (1)
CLI$V_NOTNUF                    =00000001     134     (1)
CLI$V_OCT                       =00000011     134     (1)
CLI$V_PREG                      =0000001A     134     (1)
CLI$V_QA                        =00000007     134     (1)
CLI$V_QACKLOOPLOOPS             =0000001C     134     (1)
CLI$V_QAERRORPRINTS             =0000001B     134     (1)
CLI$V_QAMULTIPLEPASS            =0000001F     134     (1)
CLI$V_QASUBTESTLOOPS            =0000001E     134     (1)
CLI$V_QATESTLOOPS               =0000001D     134     (1)
CLI$V_REG                       =00000014     134     (1)
CLI$V_REQUIRED                  =00000000     134     (1)
CLI$V_RUN                       =00000015     134     (1)
CLI$V_SET                       =00000003     134     (1)
CLI$V_SHOW                      =00000004     134     (1)
CLI$V_START_OR_RUN              =00000003     134     (1)       134     (1)
CLI$V_VALSEC                    =00000016     134     (1)
CLI$V_WORD                      =0000000E     134     (1)
DO_ACTION                        000000E8-R   346     (5)     #-257     (3)     #-524     (6)
DS$GL_CLIBASE                    00000000-XR                    310     (4)
DS$GL_RADIX                      00000000-XR                  #-369     (5)
DS$GT_VALPHA                     0000021E-R   588     (8)
DS$GT_VALPHANUM                  00000230-R   593     (8)
DS$GT_VFILE                      00000242-R   598     (8)
DS$GT_VHEX                       000002A3-R   737     (11)      772     (12)
DS$GT_VSPACE                     00000254-R   604     (8)
DS$GT_VSYMBOL                    0000020C-R   583     (8)
DS$K_ERROR                      =00000002     136     (1)
DS$K_NORMAL                     =00000001     136     (1)
DS$K_SEVERE                     =00000004     136     (1)
```

```
DS$K_SUBSYS                     =00000066     136    (1)      136    (1)
DS$K_WARNING                    =00000000     136    (1)
DS$M_ABRTFLG                    =00000040     138    (1)
DS$M_BADTIME                    =00100000     138    (1)
DS$M_BATCH                      =00400000     138    (1)
DS$M_BRKCLR                     =00001000     138    (1)
DS$M_BRKPT                      =00000800     138    (1)
DS$M_CHARFLG                    =00000100     138    (1)
DS$M_CMDFLG                     =00000080     138    (1)
DS$M_CTRLC                      =00000001     138    (1)
DS$M_CTRLO                      =00010000     138    (1)
DS$M_DEVFLG                     =00000200     138    (1)
DS$M_DISABLCC                   =01000000     138    (1)
DS$M_DONFLG                     =00002000     138    (1)
DS$M_ERRFLG                     =00000010     138    (1)
DS$M_EXCEPT                     =00080000     138    (1)
DS$M_EXETST                     =00040000     138    (1)
DS$M_HLTFLG                     =00000008     138    (1)
DS$M_LODFLG                     =00000002     138    (1)
DS$M_MEMMGT                     =00008000     138    (1)
DS$M_NI                         =04000000     138    (1)
DS$M_OUTPUT                     =00800000     138    (1)
DS$M_RUBFLG                     =00000020     138    (1)
DS$M_SCRIPT                     =00200000     138    (1)
DS$M_SETIMR                     =02000000     138    (1)
DS$M_STRFLG                     =00000004     138    (1)
DS$M_SUBT                       =00004000     138    (1)
DS$M_SYSFLG                     =00000400     138    (1)
DS$M_TIMED                      =02000000     138    (1)
DS$M_TIMED_OUT                  =08000000     138    (1)
DS$M_TIMRON                     =00020000     138    (1)
DS$V_ABRTFLG                    =00000006     138    (1)
DS$V_BADTIME                    =00000014     138    (1)
DS$V_BATCH                      =00000016     138    (1)
DS$V_BRKCLR                     =0000000C     138    (1)
DS$V_BRKPT                      =0000000B     138    (1)
DS$V_CHARFLG                    =00000008     138    (1)
DS$V_CMDFLG                     =00000007     138    (1)
DS$V_CTRLC                      =00000000     138    (1)
DS$V_CTRLO                      =00000010     138    (1)
DS$V_DEVFLG                     =00000009     138    (1)
DS$V_DISABLCC                   =00000018     138    (1)
DS$V_DONFLG                     =0000000D     138    (1)
DS$V_ERRFLG                     =00000004     138    (1)
DS$V_EXCEPT                     =00000013     138    (1)
DS$V_EXETST                     =00000012     138    (1)
DS$V_HLTFLG                     =00000003     138    (1)
DS$V_LODFLG                     =00000001     138    (1)
DS$V_MEMMGT                     =0000000F     138    (1)
DS$V_NI                         =0000001A     138    (1)
DS$V_OUTPUT                     =00000017     138    (1)
DS$V_RUBFLG                     =00000005     138    (1)
DS$V_SCRIPT                     =00000015     138    (1)
DS$V_SETIMR                     =00000019     138    (1)
DS$V_STRFLG                     =00000002     138    (1)
DS$V_SUBT                       =0000000E     138    (1)
DS$V_SYSFLG                     =0000000A     138    (1)
```

```
DS$V_TIMED                      =00000019      138     (1)
DS$V_TIMED_OUT                  =0000001B      138     (1)
DS$V_TIMRON                     =00000011      138     (1)
DS$_AP_NORMAL_BREAK             =00660150      136     (1)
DS$_ARITH                       =006600D0      136     (1)
DS$_ASBE                        =00660118      136     (1)
DS$_BADLINK                     =006600F0      136     (1)
DS$_BADTYPE                     =006600E8      136     (1)
DS$_BIIC                        =00660120      136     (1)
DS$_CHME                        =006600A8      136     (1)
DS$_CHMK                        =006600E0      136     (1)
DS$_DEVNAME                     =00660108      136     (1)
DS$_ERROR                       =00660002      136     (1)    #-277    (4)    #-322    (4)
DS$_FHWE                        =00660068      136     (1)
DS$_FRAGBUF                     =00660080      136     (1)
DS$_ICBUSY                      =006600C8      136     (1)
DS$_ICERR                       =006600C0      136     (1)
DS$_IHWE                        =00660060      136     (1)
DS$_ILLCHAR                     =00660018      136     (1)
DS$_ILLPAGCNT                   =00660078      136     (1)
DS$_ILLUNIT                     =00660100      136     (1)
DS$_INIT_FAIL                   =00660140      136     (1)
DS$_INSFMEM                     =00660050      136     (1)
DS$_INVCPU                      =00660130      136     (1)
DS$_IPL2HI                      =006600B8      136     (1)
DS$_IVADDR                      =00660040      136     (1)
DS$_IVVECT                      =00660038      136     (1)
DS$_KRNLSTK                     =00660090      136     (1)
DS$_LOGIC                       =00660070      136     (1)
DS$_MCHK                        =00660088      136     (1)
DS$_MEM_ALLOC_ERR               =00660138      136     (1)
DS$_MMOFF                       =00660058      136     (1)
DS$_NEEDUNIT                    =006600F8      136     (1)
DS$_NODE                        =00660128      136     (1)
DS$_NOPCS                       =00660110      136     (1)
DS$_NORMAL                      =00660001      136     (1)
DS$_NOSUPPORT                   =006600B1      136     (1)
DS$_NOTALLOWMP                  =00660148      136     (1)
DS$_NOTDON                      =00660030      136     (1)
DS$_NOTIMP                      =006600B0      136     (1)      136    (1)
DS$_NULLSTR                     =00660010      136     (1)
DS$_OVERFLOW                    =00660008      136     (1)
DS$_POWER                       =00660098      136     (1)
DS$_PROGERR                     =00660020      136     (1)
DS$_SEVERE                      =00660004      136     (1)
DS$_TRANSL                      =006600A0      136     (1)
DS$_TRUNCATE                    =00660028      136     (1)
DS$_UNEXPINT                    =006600D8      136     (1)
DS$_UNSUPPDEV                   =00660158      136     (1)
DS$_VASFULL                     =00660048      136     (1)
DS$_WARNING                     =00660000      136     (1)      136    (1)
DSX$PARSE                        00000007-R    211     (3)
DSX$PARSE_X                      00000068-R    265     (3)    #-278    (4)    #-323    (4)
DSX$SUPERPARSE                   00000000-R    208     (3)
EXIT                             00000065-R    263     (3)    #-221    (3)    #-437    (6)
INVALID                          00000335-R    796     (12)
OVERLAY                          0000040B-R    1015    (16)   #-1044   (16)   #-1067   (16)   #-1088   (16)
```

```
PARSE$_ACTION                   =0000000C     137     (1)       312     (4)
PARSE$_BUFADR                   =00000004     137     (1)    #-214      (3)
PARSE$_FLAG                     =00000010     137     (1)
PARSE$_NARGS                    =00000004     137     (1)
PARSE$_TREE                     =00000008     137     (1)    #-217      (3)
PARSE_COMMON                     0000000B-R   213     (3)    #-210      (3)
SAVE_CALL                        00000000-R   148     (1)    #-216      (3)    #-275     (4)    #-288     (4)
SAVE_STATUS                      00000004-R   150     (1)    #-274      (4)    #-287     (4)    #-331     (4)
SCAN$ALPHA                       0000021C-R   586     (8)    #-377      (5)    #-962     (16)
SCAN$ALPHANUM                    0000022E-R   591     (8)    #-383      (5)    #-952     (16)
SCAN$BINARY                      000002CE-R   750     (12)
SCAN$COMPARE                     00000281-R   666     (10)   #-408      (5)
SCAN$DECIMAL                     000002C4-R   742     (12)   #-964      (16)
SCAN$DEVICE                      000003C1-R   946     (16)
SCAN$FILE                        00000240-R   596     (8)    #-449      (6)
SCAN$HEX                         000002D3-R   754     (12)
SCAN$NUMERIC                     000002D6-R   757     (12)   #-371      (5)    #-744     (12)   #-748     (12)   #-752     (12)
SCAN$OCTAL                       000002C9-R   746     (12)
SCAN$SEARCHLIST                  00000372-R   870     (14)   #-425      (6)
SCAN$SPACE                       00000252-R   602     (8)    #-354      (5)    #-479     (6)    #-499     (6)    #-512     (6)
SCAN$SPACES                      00000252-R   601     (8)
SCAN$SPANC                       00000266-R   611     (8)
SCAN$SYMBOL                      0000020A-R   581     (8)    #-420      (6)    #-443     (6)
SIZ...                          =00000001     138     (1)       138     (1)
SPACE                           =00000020     143     (1)    #-400      (5)
SS$_NORMAL                      =00000001     141     (1)    #-264      (3)
TAB                             =00000009     142     (1)    #-402      (5)
TRAVERSE                         0000001C-R   218     (3)       222     (3)
TRVALNUM                         00000112-R   382     (5)       248     (3)
TRVALPHA                         0000010D-R   376     (5)       248     (3)
TRVBIF                           000000DE-R   338     (4)       248     (3)
TRVBIFS                          000000D2-R   329     (4)       248     (3)
TRVBR                            00000097-R   293     (4)       248     (3)
TRVCALL                          00000089-R   286     (4)       248     (3)
TRVCOMMA                         000001A2-R   456     (6)       248     (3)
TRVDEC                           000000FC-R   365     (5)       248     (3)
TRVEOL                           000001EB-R   511     (6)       248     (3)
TRVERR                           000000C7-R   320     (4)       248     (3)
TRVEXIT                          00000063-R   261     (3)       248     (3)
TRVFILE                          0000019C-R   448     (6)       248     (3)
TRVHEX                           000000F7-R   362     (5)       248     (3)
TRVKEYWORD                       00000166-R   419     (6)       248     (3)
TRVNUM                           00000101-R   368     (5)       248     (3)
TRVNUMA                          00000108-R   370     (5)    #-361      (5)    #-364     (5)    #-367     (5)
TRVOCT                           000000F2-R   359     (5)       248     (3)
TRVRETURN                        00000069-R   272     (4)       248     (3)
TRVSLASH                         000001A6-R   465     (6)       248     (3)
TRVSPACE                         000000ED-R   353     (5)       248     (3)
TRVSTRING                        00000117-R   389     (5)       248     (3)
TRVSYMBOL                        00000197-R   442     (6)       248     (3)
TRVVALUE                         000001AB-R   474     (6)       248     (3)
TRV_COMMA_VALUE                  000001AE-R   477     (6)    #-458      (6)    #-467     (6)
ULTRIX_FILENAME                  00000008-R   152     (1)      1075     (16)   #-1076    (16)      1077    (16)
ULTRIX_FLAGS                     00000000-XR            (1)      1037     (16)     1046    (16)       394    (5)
ULTRIX_V_MODE                    00000000-XR            (1)   #-1036      (16)   #-1045    (16)   #-394     (5)
V_BIT                           =00000002     140     (1)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO            SIZE         DEFINITION      REFERENCES...
-----            ----         ----------      -------------
$DEF             1            138    (1)
$DEFINI          1            139    (1)      139    (1)
$DS_CLIDEF       1            135    (1)      135    (1)
$DS_DSDEF        3            136    (1)      136    (1)
$DS_HPODEF       2            139    (1)      139    (1)
$DS_PARSE_DEF    1            137    (1)      137    (1)
$EQU             1            138    (1)      135    (1)      136    (1)
$EQULS1          1            136    (1)      135    (1)      136    (1)
$EQULST          1            135    (1)      135    (1)      136    (1)
$GBLINI          2            135    (1)      135    (1)      136    (1)      138    (1)
$OFFDEF          1            137    (1)      137    (1)
$VIELD           1            138    (1)      138    (1)
$VIELD1          1            138    (1)      138    (1)
CASE             1            223    (3)      223    (3)      484    (6)
CLIDEF           4            134    (1)      134    (1)
DSFDEF           3            138    (1)      138    (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ACBB	009D	406	(5)												
ADDB2	0080	1083	(16)												
ADDL	00C0	424	(6)	434	(6)	774	(12)	782	(12)	784	(12)	793	(12)	882	(14)
ADDL2	00C0	301	(4)	413	(5)	431	(6)								
ADDL3	00C1	1020	(16)	1074	(16)	279	(4)	521	(6)	897	(14)				
ADWC	00D8	792	(12)												
BBC	00E1	1036	(16)	1045	(16)	1048	(16)	1049	(16)	1058	(16)	394	(5)	404	(5)
		616	(8)	772	(12)	773	(12)								
BEQL	0013	1041	(16)	1069	(16)	1073	(16)	1080	(16)	1091	(16)	1096	(16)	221	(3)
		345	(5)	401	(5)	403	(5)	421	(6)	496	(6)	514	(6)	517	(6)
		670	(10)	672	(10)	806	(12)	881	(14)	884	(14)	963	(16)	972	(16)
BGEQ	0018	777	(12)	781	(12)										
BGTR	0014	1055	(16)	426	(6)	681	(10)	888	(14)						
BICB	008A	775	(12)												
BICB2	008A	1051	(16)	405	(5)										
BISL	00C8	765	(12)												
BLBC	00E9	393	(5)	788	(12)										
BLBS	00E8	219	(3)	331	(4)	340	(4)								
BLEQ	0015	1026	(16)	1103	(16)	1120	(16)	398	(5)	482	(6)	804	(12)		
BLSS	0019	1108	(16)	409	(5)	427	(6)	818	(12)	970	(16)				
BLSSU	001F	823	(12)												
BNEQ	0012	1031	(16)	1033	(16)	1035	(16)	1060	(16)	1063	(16)	1082	(16)	1085	(16)
		1093	(16)	1105	(16)	1111	(16)	254	(3)	276	(4)	429	(6)	498	(6)
		674	(10)	808	(12)	820	(12)	824	(12)	892	(14)	954	(16)	965	(16)
BRB	0011	1078	(16)	1086	(16)	210	(3)	278	(4)	323	(4)	355	(5)	361	(5)
		364	(5)	367	(5)	372	(5)	378	(5)	384	(5)	458	(6)	467	(6)
		490	(6)	492	(6)	494	(6)	614	(8)	744	(12)	748	(12)	752	(12)
		770	(12)	878	(14)	889	(14)	958	(16)						
BRW	0031	1056	(16)	1064	(16)	1071	(16)	257	(3)	436	(6)	437	(6)	444	(6)
		450	(6)	504	(6)	519	(6)	524	(6)						
BSBB	0010	1044	(16)	262	(3)	273	(4)	294	(4)	321	(4)	330	(4)	339	(4)
		347	(5)	443	(6)	499	(6)	512	(6)	582	(8)	587	(8)	592	(8)
		597	(8)	603	(8)	761	(12)	763	(12)	764	(12)				
BSBW	0030	1067	(16)	1088	(16)	354	(5)	371	(5)	377	(5)	383	(5)	408	(5)
		410	(5)	420	(6)	425	(6)	430	(6)	449	(6)	479	(6)	952	(16)
		962	(16)	964	(16)										
CASEB	008F	248	(3)	489	(6)										
CLRB	0094	1029	(16)	1042	(16)	1065	(16)								
CLRL	00D4	1043	(16)	212	(3)	216	(3)	395	(5)	457	(6)	480	(6)	522	(6)
		668	(10)	802	(12)	874	(14)								
CLRQ	007C	769	(12)	797	(12)	873	(14)	957	(16)	979	(16)				
CMPB	0091	1059	(16)	1079	(16)	1081	(16)	1104	(16)	253	(3)	400	(5)	402	(5)
		491	(6)	493	(6)	495	(6)	497	(6)	516	(6)	673	(10)	805	(12)
		807	(12)	819	(12)	822	(12)	891	(14)	953	(16)	971	(16)		
CMPL	00D1	1090	(16)	1095	(16)	397	(5)	683	(10)	776	(12)	794	(12)	817	(12)
		887	(14)	903	(14)										
CMPW	00B1	428	(6)												
CVTWL	0032	300	(4)	423	(6)	880	(14)	896	(14)						
DECL	00D7	1047	(16)	1107	(16)	256	(3)	483	(6)	515	(6)	679	(10)	680	(10)
		809	(12)	812	(12)	951	(16)	956	(16)	969	(16)	973	(16)		

```
EMUL            007A    783 (12)
INCB            0096   1038 (16)
INCL            00D6   1017 (16)  1061 (16)  1109 (16)  1115 (16)   255 (3)   501 (6)   617 (8)
                        619 (8)    675 (10)   676 (10)   677 (10)   778 (12)  786 (12)  791 (12)
                        811 (12)   894 (14)   950 (16)   955 (16)   974 (16)
JSB             0016    312 (4)
LOCC            003A   1030 (16)  1032 (16)  1034 (16)  1040 (16)  1062 (16)  1068 (16)  1084 (16)
                       1089 (16)  1094 (16)  1110 (16)  1112 (16)
MCOML           00D2    789 (12)   790 (12)
MNEGL           00CE    966 (16)
MOVAB           009E   1077 (16)   310 (4)    390 (5)    821 (12)
MOVAW           003E    877 (14)
MOVB            0090   1039 (16)  1076 (16)
MOVC3           0028   1075 (16)
MOVL            00D0   1018 (16)  1025 (16)  1027 (16)  1028 (16)  1066 (16)  1087 (16)  1099 (16)
                       1123 (16)   209 (3)    217 (3)    264 (3)    274 (4)    275 (4)    277 (4)
                        287 (4)    288 (4)    307 (4)    308 (4)    314 (4)    315 (4)    322 (4)
                        360 (5)    363 (5)    366 (5)    369 (5)    396 (5)    399 (5)    422 (6)
                        466 (6)    475 (6)    613 (8)    743 (12)   747 (12)   751 (12)   755 (12)
                        895 (14)   901 (14)
MOVQ            007D   1019 (16)  1070 (16)  1100 (16)  1101 (16)  1117 (16)  1118 (16)  1121 (16)
                        214 (3)    306 (4)    500 (6)    886 (14)   902 (14)
MOVW            00B0    968 (16)
MOVZBL          009A   1057 (16)   309 (4)    391 (5)    412 (5)    615 (8)    771 (12)   825 (12)
                        883 (14)
MOVZWL          003C    215 (3)    612 (8)    667 (10)   758 (12)   871 (14)   876 (14)   947 (16)
MULL3           00C5    779 (12)
POPR            00BA    313 (4)    502 (6)    518 (6)    523 (6)    607 (8)    787 (12)   798 (12)
                        904 (14)   960 (16)   975 (16)   980 (16)
PUSHAL          00DF    222 (3)
PUSHL           00DD    762 (12)   768 (12)
PUSHR           00BB    311 (4)    478 (6)    511 (6)    759 (12)   872 (14)   948 (16)   949 (16)
RET             0004   1124 (16)   266 (3)
RSB             0005   1022 (16)   280 (4)    302 (4)    316 (4)    333 (4)    342 (4)    349 (5)
                        414 (5)    433 (6)    621 (8)    684 (10)   795 (12)   799 (12)   814 (12)
                        828 (12)   905 (14)   977 (16)   981 (16)
SOBGEQ          00F4   1052 (16)   618 (8)    785 (12)   893 (14)   899 (14)
SUBL            00C2    435 (6)
SUBL2           00C2    826 (12)
SUBL3           00C3   1016 (16)  1021 (16)  1098 (16)  1114 (16)  1116 (16)   620 (8)    967 (16)
TSTB            0095   1072 (16)
TSTL            00D5   1054 (16)  1092 (16)  1102 (16)  1119 (16)   220 (3)   332 (4)    341 (4)
                        348 (5)    411 (5)    432 (6)    481 (6)    503 (6)   513 (6)    669 (10)
                        671 (10)   780 (12)   803 (12)
TSTW            00B5    827 (12)   976 (16)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCII	740	(11)														
.BLKB	153	(1)														
.BLKL	134	(1)														
.DSABL	438	(6)	609	(8)												
.ENABL	388	(5)	580	(8)												
.END	1126	(16)														
.ENDC	135	(1)	136	(1)	138	(1)										
.ENDM	134	(1)	135	(1)	136	(1)	137	(1)	138	(1)	139	(1)	223	(3)		
.ENDR	135	(1)	136	(1)	138	(1)	248	(3)	489	(6)						
.ENTRY	1024	(16)	208	(3)	211	(3)										
.IDENT	5	(1)														
.IF	135	(1)	136	(1)	138	(1)										
.IFF	135	(1)	136	(1)	138	(1)										
.IIF	135	(1)	136	(1)	138	(1)										
.IRP	135	(1)	136	(1)	137	(1)	138	(1)	248	(3)	489	(6)				
.LIBRARY	124	(1)	125	(1)	126	(1)										
.LIST	134	(1)	137	(1)												
.LONG	149	(1)	151	(1)	584	(8)	589	(8)	594	(8)	599	(8)	605	(8)	738	(11)
.MACRO	135	(1)	136	(1)	138	(1)										
.MDELETE	135	(1)	136	(1)	137	(1)										
.NLIST	134	(1)	137	(1)												
.NOCROSS	139	(1)														
.NOSHOW	6	(1)														
.PAGE	119	(1)	983	(16)												
.PSECT	134	(1)	147	(1)	155	(1)										
.RESTORE	134	(1)	139	(1)												
.SAVE	134	(1)	139	(1)												
.SBTTL	120	(1)	157	(2)	526	(7)	623	(9)	686	(11)	830	(13)	907	(15)		
.SIGNED_WORD	248	(3)	489	(6)												
.SUBTITLE	984	(16)														
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	6	#-1025	(16)	#-1027	(16)	#-1028	(16)	#-214	(3)	#-217	(3)	312	(4)
R0	68	#-1016	(16)	#-1021	(16)	#-1057	(16)	#-1059	(16)	#-1061	(16)	#-1062	(16)
		#-1079	(16)	#-1081	(16)	#-1083	(16)	#-1084	(16)	#-1092	(16)	#-1101	(16)
		#-1118	(16)	#-1123	(16)	#-264	(3)	#-274	(4)	#-275	(4)	#-277	(4)
		#-279	(4)	#-300	(4)	#-301	(4)	#-306	(4)	#-309	(4)	#-322	(4)
		#-340	(4)	#-396	(5)	#-399	(5)	#-406	(5)	#-412	(5)	#-413	(5)
		#-422	(6)	#-428	(6)	#-480	(6)	#-501	(6)	#-503	(6)	#-523	(6)
		#-615	(8)	#-616	(8)	#-620	(8)	#-668	(10)	#-675	(10)	#-683	(10)
		#-762	(12)	#-765	(12)	#-769	(12)	#-780	(12)	#-783	(12)	#-789	(12)
		#-791	(12)	#-797	(12)	#-802	(12)	#-809	(12)	#-871	(14)	#-872	(14)
		#-886	(14)	#-887	(14)	#-893	(14)	#-901	(14)	#-903	(14)	#-904	(14)
		#-966	(16)	#-968	(16)	#-975	(16)	#-976	(16)	#-979	(16)		
R1	23	#-1020	(16)	#-1090	(16)	#-1095	(16)	#-1098	(16)	#-1114	(16)	#-395	(5)
		#-400	(5)	#-402	(5)	404	(5)	#-405	(5)	#-406	(5)	#-523	(6)
		#-613	(8)	#-620	(8)	#-779	(12)	#-784	(12)	#-790	(12)	#-792	(12)
		#-872	(14)	#-891	(14)	#-904	(14)	#-975	(16)				
R10	3	208	(3)	211	(3)	#-306	(4)						
R11	2	208	(3)	211	(3)								
R2	40	#-1016	(16)	#-1017	(16)	#-1019	(16)	1024	(16)	#-1047	(16)	1048	(16)
		1049	(16)	#-1051	(16)	#-1052	(16)	#-1098	(16)	#-1100	(16)	#-1114	(16)
		#-1115	(16)	#-1117	(16)	208	(3)	211	(3)	#-310	(4)	#-391	(5)
		#-397	(5)	#-399	(5)	#-669	(10)	#-680	(10)	#-771	(12)	#-772	(12)
		773	(12)	#-774	(12)	#-775	(12)	#-776	(12)	#-783	(12)	#-787	(12)
		#-794	(12)	#-821	(12)	#-822	(12)	#-825	(12)	#-872	(14)	#-876	(14)
		877	(14)	#-895	(14)	#-899	(14)	#-904	(14)				
R3	44	#-1018	(16)	1024	(16)	1048	(16)	1049	(16)	#-1051	(16)	#-1099	(16)
		#-1116	(16)	208	(3)	211	(3)	#-360	(5)	#-363	(5)	#-366	(5)
		#-369	(5)	#-390	(5)	#-391	(5)	#-423	(6)	#-424	(6)	#-457	(6)
		#-466	(6)	#-475	(6)	#-478	(6)	#-502	(6)	#-607	(8)	616	(8)
		#-673	(10)	#-676	(10)	#-743	(12)	#-747	(12)	#-751	(12)	#-755	(12)
		#-776	(12)	#-779	(12)	#-782	(12)	#-783	(12)	#-825	(12)	#-872	(14)
		#-876	(14)	877	(14)	#-880	(14)	#-882	(14)	#-896	(14)	#-897	(14)
		#-904	(14)										
R4	80	#-1016	(16)	#-1021	(16)	1024	(16)	#-1025	(16)	#-1030	(16)	#-1032	(16)
		#-1034	(16)	#-1040	(16)	#-1054	(16)	#-1062	(16)	#-1068	(16)	#-1070	(16)
		#-1074	(16)	#-1075	(16)	#-1084	(16)	#-1089	(16)	#-1094	(16)	#-1101	(16)
		#-1102	(16)	#-1107	(16)	#-1110	(16)	#-1112	(16)	#-1118	(16)	#-1119	(16)
		#-1121	(16)	208	(3)	211	(3)	#-214	(3)	#-215	(3)	#-220	(3)
		#-256	(3)	#-308	(4)	#-314	(4)	#-396	(5)	#-397	(5)	#-434	(6)
		#-478	(6)	#-481	(6)	#-483	(6)	#-500	(6)	#-502	(6)	#-511	(6)
		#-513	(6)	#-515	(6)	#-518	(6)	#-521	(6)	#-522	(6)	#-612	(8)
		#-618	(8)	#-619	(8)	#-667	(10)	#-671	(10)	#-679	(10)	#-758	(12)
		#-759	(12)	#-785	(12)	#-786	(12)	#-798	(12)	#-803	(12)	#-812	(12)
		#-817	(12)	#-826	(12)	#-872	(14)	#-883	(14)	#-887	(14)	#-904	(14)
		#-947	(16)	#-948	(16)	#-949	(16)	#-951	(16)	#-956	(16)	#-960	(16)
		#-969	(16)	#-974	(16)	#-980	(16)						
R5	83	#-1018	(16)	#-1020	(16)	1024	(16)	#-1027	(16)	1030	(16)	1032	(16)
		1034	(16)	1040	(16)	#-1057	(16)	1062	(16)	1068	(16)	1075	(16)
		1084	(16)	1089	(16)	#-1090	(16)	1094	(16)	#-1095	(16)	#-1098	(16)
		#-1099	(16)	#-1104	(16)	#-1109	(16)	1110	(16)	1112	(16)	#-1114	(16)

```
                          #-1116   (16)    208    (3)    211    (3)   #-253    (3)   #-255    (3)   #-307    (4)
                          #-315    (4)   #-400    (5)   #-402    (5)    404     (5)   #-405    (5)   #-435    (6)
                          #-478    (6)   #-491    (6)   #-493    (6)   #-495    (6)   #-497    (6)   #-502    (6)
                          #-511    (6)   #-516    (6)   #-518    (6)   #-521    (6)   #-613    (8)   #-615    (8)
                          #-617    (8)   #-620    (8)   #-673    (10)  #-677    (10)  #-759    (12)  #-768    (12)
                          #-771    (12)  #-778    (12)  #-794    (12)  #-798    (12)  #-805    (12)  #-807    (12)
                          #-811    (12)  #-819    (12)  #-822    (12)  #-827    (12)  #-872    (14)  #-880    (14)
                          #-882    (14)  #-883    (14)  #-891    (14)  #-896    (14)  #-897    (14)  #-904    (14)
                          #-948    (16)  #-949    (16)  #-950    (16)  #-953    (16)  #-955    (16)  #-960    (16)
                          #-967    (16)  #-971    (16)  #-973    (16)  #-980    (16)
R6              15        #-1019   (16)   1024    (16)  #-1028   (16)  #-1070   (16)  #-1074   (16)  #-1077   (16)
                          #-1100   (16)  #-1117   (16)  #-1121   (16)   208     (3)   #-209    (3)    211     (3)
                          #-212    (3)   #-219    (3)   #-393    (5)
R7              28        #-1019   (16)   1024    (16)  #-1043   (16)  #-1066   (16)  #-1087   (16)   208     (3)
                           211     (3)   #-217    (3)   #-248    (3)   #-253    (3)   #-279    (4)   #-288    (4)
                          #-300    (4)   #-301    (4)   #-309    (4)   #-311    (4)   #-313    (4)   #-332    (4)
                          #-341    (4)   #-348    (5)    390     (5)   #-411    (5)   #-412    (5)   #-413    (5)
                          #-423    (6)   #-424    (6)   #-428    (6)   #-431    (6)
R8               4         208     (3)    211    (3)   #-307    (4)   #-315    (4)
R9               4         208     (3)    211    (3)   #-308    (4)   #-314    (4)
SP              32        #-1029   (16)  #-1038   (16)  #-1039   (16)  #-1042   (16)   1058    (16)  #-1065   (16)
                          #-1072   (16)  #-422    (6)   #-432    (6)   #-434    (6)   #-435    (6)   #-489    (6)
                          #-500    (6)   #-765    (12)  #-779    (12)  #-782    (12)  #-784    (12)  #-788    (12)
                          #-793    (12)  #-873    (14)  #-874    (14)  #-886    (14)  #-894    (14)  #-895    (14)
                          #-897    (14)  #-901    (14)  #-902    (14)  #-957    (16)  #-967    (16)  #-968    (16)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     22    00:00:00.01    00:00:00.20
Command processing                868    00:00:00.21    00:00:00.75
Pass 1                            528    00:00:01.75    00:00:03.33
Symbol table sort                   0    00:00:00.10    00:00:00.09
Pass 2                              7    00:00:00.59    00:00:01.22
Symbol table output                 0    00:00:00.05    00:00:00.11
Psect synopsis output               0    00:00:00.01    00:00:00.01
Cross-reference output              4    00:00:00.35    00:00:00.61
Assembler run totals             1429    00:00:03.07    00:00:06.32

The working set limit was 2900 pages.
74202 bytes (145 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 289 non-local and 67 local symbols.
1126 source lines were read in Pass 1, producing 41 object records in Pass 2.
49 pages of virtual memory were used to define 15 macros.
```

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              5
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                2
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       6
TOTALS (all libraries)                              13

313 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:PARSE.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element PCSLOAD.B32
;    0002  0      ! *1     6-FEB-1986 15:21:42 DS ""
;    0003  0      ! DEC/CMS REPLACEMENT HISTORY, Element PCSLOAD.B32
;    0004  0      %TITLE 'PCSLOAD Module'
;    0005  0      MODULE PCSLOAD (                        ! Dummy module for supervisors
;    0006  0                      IDENT = '06-03'         ! that have no patch control store
;    0007  0                      ) =
;    0008  0      !++
;    0009  0      ! COPYRIGHT (c) 1980, 1981, 1983, 1984
;    0010  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0011  0      !
;    0012  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0013  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0014  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0015  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0016  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0017  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0018  0      ! REMAIN IN DEC.
;    0019  0      !
;    0020  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0021  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0022  0      ! CORPORATION.
;    0023  0      !
;    0024  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0025  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0026  0      !-
```

```
;    0028  0      !++
;    0029  0      ! Facility:
;    0030  0      !
;    0031  0      !       Diagnostic Supervisor
;    0032  0      !
;    0033  0      ! Abstract:
;    0034  0      !
;    0035  0      !       This module is used by those diagnostic supervisors that have
;    0036  0      !       no patch control store, namely, ESSAA, and ENSAA.
;    0037  0      !       The routines in this module do nothing except return status.
;    0038  0      !
;    0039  0      ! Author:
;    0040  0      !
;    0041  0      !       Bob Bergazzi
;    0042  0      !
;    0043  0      ! Creation Date:
;    0044  0      !
;    0045  0      !       22-Nov-1982
;    0046  0      !
;    0047  0      ! Version:
;    0048  0      !
;    0049  0      !       6.10
;    0050  0      !
;    0051  0      ! Modified By:
;    0052  0      !
;    0053  0      !       01      Bob Bergazzi    May 17, 1983    Version 6.11
;    0054  0      !               Changed the order of searching libraries.
;    0055  0      !
;    0056  0      !       02      Bob Bergazzi    Oct. 21, 1983   Version 6.13
;    0057  0      !               Added INITPCS command routine, DSV$INITPCS.
;    0058  0      !
;    0059  0      !       03      Bob Bergazzi    Mar. 1, 1984    Version 7.0
;    0060  0      !               Moved ECSAA functionality to PCSLOAD750.
;    0061  0      !--
```

```
;    0063  0     %SBTTL  'Libraries'
;    0064  1     BEGIN                           ! Begin the Module_Name module
;    0065  1
;    0066  1     LIBRARY
;    0067  1             '$Diag';                ! Use Diag.L32 first
;    0068  1
;    0069  1     LIBRARY
;    0070  1             '$DS';                  ! Use Ds.L32 second
;    0071  1
;    0072  1     LIBRARY
;    0073  1             '$CRD';                 ! Use CRD.L32 third
;    0074  1
;    0075  1     LIBRARY
;    0076  1             'Sys$Library:Lib';      ! Use Lib as last resort
```

```
;    0078  1     %SBTTL  'Table of Contents'
;    0079  1
;    0080  1     forward routine
;    0081  1         dsx$loadpcs,                    ! Interface to calling routine
;    0082  1         dsv$initpcs : novalue;          ! Init the PCS                  [02]
```

```
;    0084  1      %SBTTL  'Psect Definitions'
;    0085  1
;    0086  1      PSECT
;    0087  1          PLIT    = Data (NOEXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0088  1
;    0089  1      PSECT
;    0090  1          CODE    = Code (  EXECUTE,   SHARE, NOWRITE, ADDRESSING_MODE (WORD_RELATIVE));
;    0091  1
;    0092  1      PSECT
;    0093  1          OWN     = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
;    0094  1
;    0095  1      PSECT
;    0096  1          GLOBAL  = Work (NOEXECUTE, NOSHARE,   WRITE, ADDRESSING_MODE (LONG_RELATIVE));
```

```
;    0098  1     %SBTTL  'PCSLOAD-Static Storage'
;    0099  1
;    0100  1             ModNam ('PCSLOAD');             ! Module name for ErrSup macro
```

```
;    0102  1      %SBTTL 'DSX$LOADPCS Routine'
;    0103  1
;    0104  1      global routine dsx$loadpcs =
;    0105  1
;    0106  1      !++
;    0107  1      ! Functional Description:
;    0108  1      !
;    0109  1      !       This routine does nothing but return status.
;    0110  1      !
;    0111  1      ! Formal Input Parameters:
;    0112  1      !
;    0113  1      !       None
;    0114  1      !
;    0115  1      ! Formal Output Parameters:
;    0116  1      !
;    0117  1      !       None
;    0118  1      !
;    0119  1      ! Side Effects:
;    0120  1      !
;    0121  1      !       None
;    0122  1      !
;    0123  1      ! Completion Codes:
;    0124  1      !
;    0125  1      !       ds$_nopcs       - PCS not available on this cpu.
;    0126  1      !--
```

```
;   0128  2     begin                                   ! routine dsx$loadpcs
;   0129  2     return ds$_nopcs;
;   0130  1     end;                                    ! routine dsx$loadpcs

                                                                .TITLE  PCSLOAD PCSLOAD Module
                                                                .IDENT  \06-03\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                    44  41  4F  4C  53  43  50  07  00000 P.AAA:  .ASCII  <7>\PCSLOAD\                                  ;

                                                        $MODULE=            P.AAA


                                                                .PSECT  CODE,NOWRT,  SHR,2

                                            0000 00000          .ENTRY  DSX$LOADPCS, Save nothing                       ; 0104
                        50 00660110   8F  D0 00002              MOVL    #6684944, R0                                    ; 0129
                                            04 00009              RET                                                     ; 0130

; Routine Size:  10 bytes,    Routine Base:  CODE + 0000
```

```
%SBTTL 'DSV$INITPCS Routine'                                    !                    begin   [02]

global routine dsv$initpcs : novalue =

!++
! Functional Description:
!
!       This routine is called by CLI in response to an INITPCS command. The command is not valid for
!       supervisors which use this module.
!
! Formal Input Parameters:
!
!       None
!
! Formal Output Parameters:
!
!       None
!
! Side Effects:
!
!       None
!
! Completion Codes:
!
!       None
!--
```

```
;   0159  2     begin                               ! routine dsv$initpcs
;   0160  2     $printi ($ascic (%string ('?? Command not valid for this supervisor!/')));
;   0161  1     end;                                ! routine dsv$initpcs



                                                    .PSECT  DATA,NOWRT,NOEXE,  SHR,2

74  6F  6E  20  64  6E  61  6D  6D  6F  43  20  3F  3F  2A  00008 P.AAB:  .ASCII  \*?? Command not valid for this superviso\  ;
73  69  68  74  20  72  6F  66  20  64  69  6C  61  76  20  00017                                                            ;
                    6F  73  69  76  72  65  70  75  73  20  00026                                                            ;
                                                2F  21  72  00030       .ASCII  \r!/\                                        ;

                                                    .EXTRN  DSX$PRINTI

                                                    .PSECT  CODE,NOWRT,  SHR,2

                                              0000 00000            .ENTRY  DSV$INITPCS, Save nothing                      ; 0134
                                 00000000'  EF  9F 00002            PUSHAB  P.AAB                                          ; 0160
                        00000000G  9F       01  FB 00008            CALLS   #1, @#DSX$PRINTI                               ;
                                            04 0000F               RET                                                    ; 0161

; Routine Size:  16 bytes,    Routine Base:  CODE + 000A

;   0162  1
;   0163  1     end                                 ! end of pcsload module
;   0164  0     eludom
```

```
;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DATA                           51  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                           26  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...
$ASCIC	Macro	Lib01	160
$ER	Comptime	100	
$MODULE	Bind	100	
$PRINTI	Macro	Lib02	160
DS$_NOPCS	Literal	Lib01	129
DSV$INITPCS	GlobRout	134	82f
DSX$LOADPCS	GlobRout	104	81f
DSX$PRINTI	ExtRout	160	160c
MODNAM	Macro	Lib02	100

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]PCSLOAD.B32;1
5	Module PCSLOAD	
67	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
70	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
73	Library #3	DISK$DS:[DS.LOCAL.LIBRARY]CRD.L32;4
76	Library #4	SYS$COMMON:[SYSLIB]LIB.L32;2
164	Eludom PCSLOAD	

KEY TO REFERENCE TYPE FLAGS

```
.   Fetch
=   Store
c   Routine call
a   Address use
a   Indirect use
e   External, external routine, or external literal declaration
f   Forward or forward routine declaration
h   Condition handler enabling
m   Map declaration
u   Undeclare declaration
```

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	2	0	96	00:00.0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	2	0	47	00:00.0
DISK$DS:[DS.LOCAL.LIBRARY]CRD.L32;4	491	0	0	63	00:00.0
SYS$COMMON:[SYSLIB]LIB.L32;2	20450	0	0	1101	00:01.0

```
;                                       COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:PCSLOAD.B32

; Size:         26 code + 51 data bytes
; Run Time:         00:01.6
; Elapsed Time:     00:03.2
; Lines/CPU Min:     6348
; Lexemes/CPU-Min: 11341
; Memory Used:  50 pages
; Compilation Complete
```

Table of contents

(1) 183 Libraries, Macros, and Equated Symbols
(1) 212 Work Psect Declarations
(1) 320 Data Psect Declarations
(1) 338 DSV$SetPage Routine - Set terminal page size
(1) 394 DSV$ShowPage Routine - Show terminal page size
(1) 440 DSX$Type_Out Routine - Do actual print
(1) 680 DSX$Print Routine - Load type code byte
(1) 756 DSX$PrintX Routines - Extended print routines
(1) 926 DSX$Printrev Routine
(1) 1292 DSX$CvtReg Routine - Mnemonic register conversion routine

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element PRINT.MAR
0000     2 ; *1     6-FEB-1986 15:21:48 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element PRINT.MAR
0000     4         .TITLE  PRINT   *** PRINT Formatted print routines
0000     5         .IDENT  /07-34/
0000     6         .NoShow Conditionals            ;                       [10]
0000     7
0000     8 ;
0000     9 ; Copyright (c) 1977, 1983, 1985
0000    10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    11 ;
0000    12 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    13 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    14 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    15 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    16 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    17 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    18 ; REMAIN IN DEC.
0000    19 ;
0000    20 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    21 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    22 ; CORPORATION.
0000    23 ;
0000    24 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    26 ;--
```

```
0000     28 ;++
0000     29 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR
0000     30 ;
0000     31 ; ABSTRACT:
0000     32 ;
0000     33 ; ENVIRONMENT:
0000     34 ;
0000     35 ; AUTHOR:       TOM SOUTTER     10-NOV-77       VERSION 01
0000     36 ;
0000     37 ; MODIFICATIONS:
0000     38 ;
0000     39 ;       01      Roger Riggs     12-Jun-78
0000     40 ;               Added DSX$PRINTI similar to DSX$PRINTF but always prints even
0000     41 ;               in APT mode. Changed prefix on print routines to DSX$PRINTx
0000     42 ;
0000     43 ;       02      Roger Riggs     12-Jun-78
0000     44 ;               Added usermode cancel and setmode
0000     45 ;
0000     46 ;       03      Roger Riggs     1-OCT-1979
0000     47 ;               Changed W^ to L^ on DSA$GL_FLAGS references
0000     48 ;
0000     49 ;       04      Roger Riggs, 14-Jan-1979, Version 5.2
0000     50 ;               Added code to DSX$CVTREG to allow the user
0000     51 ;               to specify a code for the mnemonic string instead of
0000     52 ;               the string itself codes are used for transportabile
0000     53 ;               diagnostics and represent a generic type of data,
0000     54 ;               i.e. UNIBUS map register.
0000     55 ;
0000     56 ;       05      Dave Butenhof 29-apr-80
0000     57 ;               Use RMS for output if DS$V_BATCH flag is set (running
0000     58 ;               under VMS batch/command file).
0000     59 ;
0000     60 ;       06      Dave Butenhof, 16-oct-1980, version 6.1
0000     61 ;               Set flag DS$V_OUTPUT on PRINT call.
0000     62 ;               Also have print use stack buffer for FAO output to avoid
0000     63 ;               conflict with SCRIPT prompt formatting.
0000     64 ;
0000     65 ;       07      - Jack Stansbury, 28-Oct-1981, Version 6.-
0000     66 ;               Added .LIBRARY statements for $DS and $DIAG. Fixed
0000     67 ;               truncation errors.
0000     68 ;
0000     69 ;       08      - Dave Butenhof, 02-Nov-1981, version 6.-
0000     70 ;               Fix truncation error.
0000     71 ;
0000     72 ;       09      - Dave Butenhof, 16-Nov-1981, version 6.5
0000     73 ;               Enhance to provide capability of outputting a binary 'type
0000     74 ;               code' byte preceding each and every message output by the
0000     75 ;               Supervisor.  This is controlled by the SET [CLEAR] BINARY
0000     76 ;               command.
0000     77 ;
0000     78 ;       10      - Jack Stansbury, 23-Mar-1982, Version 6.?
0000     79 ;               Took out the RTypeMsg routine because it is now obselete
0000     80 ;               with typecoding and with the Print routine. Also added a
0000     81 ;               "!/" to the message output by the print routine when
0000     82 ;               Debug_The_Sucker has been set to one.
0000     83 ;
0000     84 ;       11      Jack Stansbury, 8-Apr-1982, Version 6.7
```

```
0000     85 ;              Added code to the DSX$TypeOut routine to call the CRD
0000     86 ;              routine. Also added a .Library statement for /$CRD/,
0000     87 ;              the new CRD library. Also changed some statements to
0000     88 ;              the new DS macros (Br_If_ ...).
0000     89 ;
0000     90 ;        12    Marion Baggett, 9-Apr-1982, Version 6.7
0000     91 ;              Ordered .library statements.
0000     92 ;
0000     93 ;        13    Jack Stansbury, 12-Apr-1982, Version 6.7
0000     94 ;              Changed the code in DSX$TypeOut a bit so that the CRD routine
0000     95 ;              is not called if the TypeCode is CRD_AutoTest. This allows CRD
0000     96 ;              Print's to go through without the CRD routine being called.
0000     97 ;
0000     98 ;        14    Jack Stansbury, 6-May-1982, Version 6.8
0000     99 ;              Added code in DSX$TypeOut to supress outputting the binary
0000    100 ;              typecode when CRD-AutoTest is running (which, in itself, implies
0000    101 ;              that the BINARY flag has been set on).
0000    102 ;
0000    103 ;        15    Jack Stansbury, 24-July-1982, Version 6.9
0000    104 ;              Added support for the Menu Test of CRD.
0000    105 ;
0000    106 ;        16    Bob Bergazzi    30-March-1983   Version 6.11
0000    107 ;              Added guts of SET PAGE command: DSV$SHOWPAGE, DSV$SETPAGE, and
0000    108 ;              in DSX$TYPE_OUT.
0000    109 ;
0000    110 ;        17    John Ciukaj,    2-Apr.-1983,    Version 6.11
0000    111 ;              - Changed references to CRD bits in DSA Flags since
0000    112 ;                they were redefined to distinguish between online
0000    113 ;                and offline.
0000    114 ;
0000    115 ;        18    Peter Green     18-JUL-1983     Version 6.12
0000    116 ;              Changed print macros to test R0 for status such that
0000    117 ;              on error an immediate exit occurs instead of attempting
0000    118 ;              to print the message.
0000    119 ;
0000    120 ;        19    Richard Brown   7-June-1984     Version 7.0
0000    121 ;              Added null arguments to SYS$QIO call in DSX$TYPE_OUT.
0000    122 ;              Changed channel number from TTIN to TTOUT in two $QIO
0000    123 ;              calls in DSX$TYPE_OUT, prompting for user response
0000    124 ;              to display next page of terminal output.
0000    125 ;
0000    126 ;        20    Ron Parker      21-Jan-1985
0000    127 ;              Added test for CRD online condition
0000    128 ;
0000    129 ;        21    Amy Iorio       15-Feb-1985     Version 8.1
0000    130 ;              Added routine for $DS_PRINTREV
0000    131 ;
0000    132 ;        22    Amy Iorio       18-Mar-1985     Version 8.2
0000    133 ;              Modified Printrev to work with error
0000    134 ;
0000    135 ;        23    Amy Iorio       27-Mar-1985     Version 8.2
0000    136 ;              Fix to Printrev (bits set and prlink print)
0000    137 ;
0000    138 ;        24    Ted Salem       8-April-1985    Version 8.2
0000    139 ;              Added code to interlock all print services.
0000    140 ;
0000    141 ;        25    Amy Iorio       18-April-1985   Version 8.2
```

```
0000    142 ;               Fix to Printrev (greaterbit and all bit)
0000    143 ;
0000    144 ;       26      Amy Iorio       17-May-1985     Version 8.2
0000    145 ;               Enhancement of Printrev; patch-level information
0000    146 ;               checked.
0000    147 ;
0000    148 ;       27      Amy Iorio       18-June-1985    Version 8.3
0000    149 ;               Fix to Printrev; recognize actual revision level
0000    150 ;               of zero.
0000    151 ;
0000    152 ;       28      Amy Iorio       16-Aug-1985     Version 8.3
0000    153 ;               Addition to Printrev of revision level statement.
0000    154 ;
0000    155 ;       29      Domenic Andella 16-August-1984  Version 8.3
0000    156 ;               Made modification at DSX$PRINTX routines/print_x:
0000    157 ;               to skip jsb to kb_check if an mp system. This is
0000    158 ;               to eliminate repeating printouts from MP$_PRIMARY_
0000    159 ;               CPU_CHECK code that is called at each $DS_BREAK.
0000    160 ;
0000    161 ;       30      Amy Iorio       05-Sept-1985    Version 9.0
0000    162 ;               Made modification to DSX$PRINTREV in order to allow
0000    163 ;               more error messages for hardware revision levels to
0000    164 ;               print out.
0000    165 ;
0000    166 ;       31      Ted Salem       14-Oct-1985     Version 9.0
0000    167 ;               Modified DSX$PRINTREV so that a letter code will
0000    168 ;               be optionally printed for the hardware revision.
0000    169 ;
0000    170 ;       32      Ted Salem       15-Oct-1985     Version 9.0
0000    171 ;               Fixed the method in which the inhibit print
0000    172 ;               masks were being checked in DSX$PRINTREV
0000    173 ;
0000    174 ;       33      Ted Salem       16-Oct-1985     Version 9.0
0000    175 ;               Added code in DSX$PRINTREV to check to see if
0000    176 ;               module number exists on the argument stack.
0000    177 ;
0000    178 ;       34      Ted Salem       29-Oct-1985     Version 9.1
0000    179 ;               Put in bounds check and other error statuses.
0000    180 ;
0000    181 ;--
```

```
                     0000    183          .Subtitle       Libraries, Macros, and Equated Symbols
                     0000    184 ;
                     0000    185 ; INCLUDE FILES:
                     0000    186 ;
                     0000    187          .Library        /Sys$Library:Lib/        ;                                  [12]
                     0000    188          .Library        /$CRD/                   ; The CRD library                  [11]
                     0000    189          .Library        /$DS/                    ;
                     0000    190          .Library        /$Diag/                  ;
                     0000    191 ;
                     0000    192 ; EXTERNAL REFERENCES:
                     0000    193 ;
                     0000    194 ;
                     0000    195 ; EQUATED SYMBOLS:
                     0000    196 ;
                     0000    197
                     0000    198          $SSDEF                                   ;                                  [18]
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000    199          $DS_DSDEF                                ; DIAGNOSTIC SUPERVISOR ERROR CODES
                     0000    200          $DS_HPODEF                               ; Define HP$Q_DEVICE
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
00000008             0000                 .IIF    NB,.BLKQ,       .BLKQ   1
                     0008                 .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
0000000A             0008                 .IIF    NB,.BLKW,       .BLKW   1
                     000A                 .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
0000000B             000A                 .IIF    NB,.BLKB,       .BLKB   1
                     000B                 .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
0000000C             000B                 .IIF    NB,.BLKB,       .BLKB   1
                     000C                 .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
00000018             000C                 .IIF    NB,.BLKB,       .BLKB   12
                     0018                 .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
0000001C             0018                 .IIF    NB,.BLKL,       .BLKL   1
                     001C                 .IIF    NB,HP$A_DVA,    HP$A_DVA:
00000020             001C                 .IIF    NB,.BLKL,       .BLKL   1
                     0020                 .IIF    NB,HP$A_LINK,   HP$A_LINK:
00000024             0020                 .IIF    NB,.BLKL,       .BLKL   1
                     0024                 .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
00000026             0024                 .IIF    NB,.BLKW,       .BLKW   1
                     0026                 .IIF    NB,HP$T_TYPE,   HP$T_TYPE:
00000032             0026                 .IIF    NB,.BLKB,       .BLKB   12
                     0032                 .IIF    NB,HP$A_DEPENDENT,      HP$A_DEPENDENT:
                     0000                 .RESTORE
                     0000    201          $DS_DSADEF                               ; CONTROL FLAG DEFINITIONS
                     0000    202          $DS_CVTDEF                               ; Define CVTREG code values
                     0000    203          DSFDEF
                     0000    204          $RABDEF                                  ; Define RAB offsets
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .PSECT  $ABS$,ABS
00000000             FE70                 .=0
                     0000                 .RESTORE
                     0000    205          $DS_TypeDef                              ; Define the TypeCode literals     [13]
                     0000    206          $CRD_Literals                            ; Define the CRD literals          [11]
                     0000    207          CLIDEF                                   ; Define CLI offsets               [16]
                     0000    208          .show
                     0000    209          $DS_PRINTREV_DEF                         ; Define PRINTREV variables
                     0000                 $OFFDEF PRINTREV,<LOG_UNIT,ACTUAL_REV,EXPECTED_REV,REV_TYPE,-
```

```
                0000            PRINTMASK,PRLINK,MODULENUM>
                0000                    .LIST
                0000
                0000            $DS_PRVDEF
                0000             $GBLINI
                0000            .IF     IDN <LOCAL> <GLOBAL>
                0000            .MACRO  $DEF    SYM,ALLOC,SIZ
                0000            .IIF    NB,SYM, SYM::
                0000            .IIF    NB,ALLOC,       ALLOC   SIZ
                0000            .ENDM   $DEF
                0000            .MACRO  $EQU    SYM,VAL
                0000            SYM==VAL
                0000            .ENDM   $EQU
                0000            .MACRO  $VIELD1 MOD,SEP,SYM,SIZ,MSK
                0000            SIZ...=1
                0000            .IIF    NB,SIZ, SIZ...=SIZ
                0000            .IF     NB,SYM
                0000            MOD'SEP'V_'SYM==BIT...
                0000            .IIF    NB,SIZ, MOD'SEP'S_'SYM==SIZ
                0000            .IIF    NB,MSK, MOD'SEP'M_'SYM==<<<1@SIZ...>-1>@BIT...>
                0000            .ENDC
                0000            BIT...=BIT...+SIZ...
                0000            .ENDM   $VIELD1
                0000            .IFF
                0000            .IIF    DIF <LOCAL> <LOCAL>,.ERROR ;ARG MUST BE "GLOBAL","LOCAL",OR NULL
                0000            .MACRO  $DEF    SYM,ALLOC,SIZ
                0000            .IIF    NB,SYM, SYM:
                0000            .IIF    NB,ALLOC,       ALLOC   SIZ
                0000            .ENDM   $DEF
                0000            .MACRO  $EQU    SYM,VAL
                0000            SYM=VAL
                0000            .ENDM   $EQU
                0000            .MACRO  $VIELD1 MOD,SEP,SYM,SIZ,MSK
                0000            SIZ...=1
                0000            .IIF    NB,SIZ, SIZ...=SIZ
                0000            .IF     NB,SYM
                0000            MOD'SEP'V_'SYM=BIT...
                0000            .IIF    NB,SIZ, MOD'SEP'S_'SYM=SIZ
                0000            .IIF    NB,MSK, MOD'SEP'M_'SYM=<<<1@SIZ...>-1>@BIT...>
                0000            .ENDC
                0000            BIT...=BIT...+SIZ...
                0000            .ENDM   $VIELD1
                0000            .ENDC
                0000
                0000             $vield PRV,0,<-
                0000                        <HWREV,,M>-         ; SET IF REV TYPE IS HARDWARE
                0000                        <FWREV,,M>-         ; SET IF REV TYPE IS FIRMWARE
                0000                        <PREV,,M>-          ; SET IF REV TYPE IS PATCH
                0000                        >
00000000        0000            .IIF    NB,0,   BIT...=0
                0000            .IRP    L,<                     <HWREV,,M>                      <FWREV,,M>
                0000            $VIELD1 PRV,$,L
                0000            .ENDR
                0000            $VIELD1 PRV,$,HWREV,,M
00000001        0000            SIZ...=1
                0000            .IIF    NB,,    SIZ...=
                0000            .IF     NB,HWREV
```

```
00000000  0000              PRV$V_HWREV=BIT...
                            .IIF    NB,,    PRV$S_HWREV=
00000001  0000              .IIF    NB,M,   PRV$M_HWREV=<<<1aSIZ...>-1>aBIT...>
                            .ENDC
00000001  0000              BIT...=BIT...+SIZ...
                            $VIELD1 PRV,$,FWREV,,M
00000001  0000              SIZ...=1
                            .IIF    NB,,    SIZ...=
                            .IF     NB,FWREV
00000001  0000              PRV$V_FWREV=BIT...
                            .IIF    NB,,    PRV$S_FWREV=
00000002  0000              .IIF    NB,M,   PRV$M_FWREV=<<<1aSIZ...>-1>aBIT...>
                            .ENDC
00000002  0000              BIT...=BIT...+SIZ...
                            $VIELD1 PRV,$,PREV,,M
00000001  0000              SIZ...=1
                            .IIF    NB,,    SIZ...=
                            .IF     NB,PREV
00000002  0000              PRV$V_PREV=BIT...
                            .IIF    NB,,    PRV$S_PREV=
00000004  0000              .IIF    NB,M,   PRV$M_PREV=<<<1aSIZ...>-1>aBIT...>
                            .ENDC
00000003  0000              BIT...=BIT...+SIZ...
                            $DS_PRMSKDEF
                             $GBLINI
                            .IF     IDN <LOCAL> <GLOBAL>
                            .MACRO  $DEF    SYM,ALLOC,SIZ
                            .IIF    NB,SYM, SYM::
                            .IIF    NB,ALLOC,       ALLOC   SIZ
                            .ENDM   $DEF
                            .MACRO  $EQU    SYM,VAL
                            SYM==VAL
                            .ENDM   $EQU
                            .MACRO  $VIELD1 MOD,SEP,SYM,SIZ,MSK
                            SIZ...=1
                            .IIF    NB,SIZ, SIZ...=SIZ
                            .IF     NB,SYM
                            MOD'SEP'V_'SYM==BIT...
                            .IIF    NB,SIZ, MOD'SEP'S_'SYM==SIZ
                            .IIF    NB,MSK, MOD'SEP'M_'SYM==<<<1aSIZ...>-1>aBIT...>
                            .ENDC
                            BIT...=BIT...+SIZ...
                            .ENDM   $VIELD1
                            .IFF
                            .IIF    DIF <LOCAL> <LOCAL>,.ERROR ;ARG MUST BE "GLOBAL","LOCAL",OR NULL
                            .MACRO  $DEF    SYM,ALLOC,SIZ
                            .IIF    NB,SYM, SYM:
                            .IIF    NB,ALLOC,       ALLOC   SIZ
                            .ENDM   $DEF
```

```
                     0000               .MACRO  $EQU    SYM,VAL
                     0000               SYM=VAL
                     0000               .ENDM   $EQU
                     0000               .MACRO  $VIELD1 MOD,SEP,SYM,SIZ,MSK
                     0000               SIZ...=1
                     0000               .IIF    NB,SIZ, SIZ...=SIZ
                     0000               .IF     NB,SYM
                     0000               MOD'SEP'V_'SYM=BIT...
                     0000               .IIF    NB,SIZ, MOD'SEP'S_'SYM=SIZ
                     0000               .IIF    NB,MSK, MOD'SEP'M_'SYM=<<<1@SIZ...>-1>@BIT...>
                     0000               .ENDC
                     0000               BIT...=BIT...+SIZ...
                     0000               .ENDM   $VIELD1
                     0000               .ENDC
                     0000
                     0000                $vield PRMSK,0,<-
                     0000                                <LSS,,M>-       ; IF SET INHIBITS LESS THAN MESSAGE
                     0000                                <EQL,,M>-       ; IF SET INHIBITS EQUAL TO MESSAGE
                     0000                                <GTR,,M>-       ; IF SET INHIBITS GREATER THAN MESSAGE
                     0000                                <ALWAYS,,M>-    ; IF SET INHIBITS ONLY NON-RELATIONAL MESSAG
                     0000                                <TRANSL,,M>-    ; IF SET TRANSLATE HARDWARE REV TO LETTER CO
                     0000                                >
            00000000 0000               .IIF    NB,0,   BIT...=0
                     0000               .IRP    L,<                             <LSS,,M>                        <EQL,,M>
                     0000               $VIELD1 PRMSK,$,L
                     0000               .ENDR
                     0000               $VIELD1 PRMSK,$,LSS,,M
            00000001 0000               SIZ...=1
                     0000               .IIF    NB,,    SIZ...=
                     0000               .IF     NB,LSS
            00000000 0000               PRMSK$V_LSS=BIT...
                     0000               .IIF    NB,,    PRMSK$S_LSS=
            00000001 0000               .IIF    NB,M,   PRMSK$M_LSS=<<<1@SIZ...>-1>@BIT...>
                     0000               .ENDC
            00000001 0000               BIT...=BIT...+SIZ...
                     0000
                     0000
                     0000               $VIELD1 PRMSK,$,EQL,,M
            00000001 0000               SIZ...=1
                     0000               .IIF    NB,,    SIZ...=
                     0000               .IF     NB,EQL
            00000001 0000               PRMSK$V_EQL=BIT...
                     0000               .IIF    NB,,    PRMSK$S_EQL=
            00000002 0000               .IIF    NB,M,   PRMSK$M_EQL=<<<1@SIZ...>-1>@BIT...>
                     0000               .ENDC
            00000002 0000               BIT...=BIT...+SIZ...
                     0000
                     0000
                     0000               $VIELD1 PRMSK,$,GTR,,M
            00000001 0000               SIZ...=1
                     0000               .IIF    NB,,    SIZ...=
                     0000               .IF     NB,GTR
            00000002 0000               PRMSK$V_GTR=BIT...
                     0000               .IIF    NB,,    PRMSK$S_GTR=
            00000004 0000               .IIF    NB,M,   PRMSK$M_GTR=<<<1@SIZ...>-1>@BIT...>
                     0000               .ENDC
            00000003 0000               BIT...=BIT...+SIZ...
```

```
                        0000
                        0000
                        0000            $VIELD1 PRMSK,$,ALWAYS,,M
                 00000001  0000            SIZ...=1
                        0000            .IIF    NB,,    SIZ...=
                        0000            .IF     NB,ALWAYS
                 00000003  0000            PRMSK$V_ALWAYS=BIT...
                        0000            .IIF    NB,,    PRMSK$S_ALWAYS=
                 00000008  0000            .IIF    NB,M,   PRMSK$M_ALWAYS=<<<1@SIZ...>-1>@BIT...>
                        0000            .ENDC
                 00000004  0000            BIT...=BIT...+SIZ...
                        0000
                        0000
                        0000            $VIELD1 PRMSK,$,TRANSL,,M
                 00000001  0000            SIZ...=1
                        0000            .IIF    NB,,    SIZ...=
                        0000            .IF     NB,TRANSL
                 00000004  0000            PRMSK$V_TRANSL=BIT...
                        0000            .IIF    NB,,    PRMSK$S_TRANSL=
                 00000010  0000            .IIF    NB,M,   PRMSK$M_TRANSL=<<<1@SIZ...>-1>@BIT...>
                        0000            .ENDC
                 00000005  0000            BIT...=BIT...+SIZ...
                        0000
                        0000
                        0000
                        0000
                        0000
```

```
                        0000   212          .SubTitle       Work Psect Declarations
                    00000000   213          .Psect  Work, Noshr, Noexe, Wrt, Long
                        0000   214
                        0000   215 ;
                        0000   216 ; Beginning of data structure for dsx$printrev
                        0000   217 ;--------------------------------------------------------------------------------
                000001A0 0000   218 devrev$t::      .blkl 104                ; Printrev/holds logical_unit and rev_types
                        01A0   219
                000001A4 01A0   220 firmexist$t::   .blkl 1                  ; Flag set if firm type moved into devrev$t
                000001A8 01A4   221 hardexist$t::   .blkl 1                  ; Flag set if hard type moved into devrev$t
                000001AC 01A8   222 patchexist$t::  .blkl 1                  ; Flag set if patch level moved into devrev$
                000001B0 01AC   223 offset$t:       .blkl 1                  ; Printrev offset into devrev$t data structu
                000001B4 01B0   224 p_table:        .blkl 1                  ; Printrev ptable address                [21]
                000001B8 01B4   225 WARECODE:       .blkl 1                  ; Printrev device type statement         [21]
                000001BC 01B8   226 RELATION:       .blkl 1                  ; Printrev relational operator statement[21]
                000001CC 01BC   227 DEVNAME:        .blkq 2                  ; Printrev holds device name             [21]
                000001D0 01CC   228 LOGNUM::        .blkl 1
                        01D0   229
                        01D0   230 devrev$t_x::
                        01D0   231 ;--------------------------------------------------------------------------------
                        01D0   232 ; Ending of data structure for dsx$printrev
                        01D0   233 ;
              0000 0000'01D0   234 QBUFF:   .WORD   DS$K_BUFSIZ,0            ; QUADWORD BUFFER DESCRIPTOR
               00000000'01D4   235          .LONG   DS$GT_BUFFER
                        01D8   236
                      00 01D8   237 Width_Too_Large:        .byte   0         ; Bit 0 is set if the last line was      [16]
                        01D9   238                                            ; longer than the terminal width         [16]
                00000000 01D9   239 Line_Count::    .long   0                 ; current number of lines output since   [16]
                        01DD   240                                            ; last PAGE                              [16]
                00000000 01DD   241 DS$GL_Page::    .long   0                 ; SET PAGE 0 is default                  [16]
                        01E1   242 T_Page:                                    ;                                        [16]
6D 20 65 7A 69 73 20 65 67 61 50 00' 01E1   243          .ASCIC  "Page size must be greater than 0!/"    ;
74 61 65 72 67 20 65 62 20 74 73 75  01ED
   2F 21 30 20 6E 61 68 74 20 72 65  01F9
                              22     01E1
                        0204   244 T_Show_Page:                               ;                                        [16]
61 70 20 6C 61 6E 69 6D 72 65 54 00' 0204   245          .ASCIC  "Terminal page size is !ZB line!%S!/"   ;
21 20 73 69 20 65 7A 69 73 20 65 67  0210
2F 21 53 25 21 65 6E 69 6C 20 42 5A  021C
                              23     0204
                        0228   246
                        0228   247
                00000000 00000000 0228   248 Q_DIVD:         .QUAD   0         ; DIVIDEND for EDIV                      [31]
                00000000  0230   249 L_QUO:          .LONG   0                 ; QUOTIENT for EDIV                      [31]
                00000000  0234   250 L_REM:          .LONG   0                 ; REMAINDER for EDIV                     [31]
                        0238   251
                        0238   252
                      00 0238   253 ACT_REV:        .BYTE   0                 ; Character Storage                      [31]
03'03'03'03'03'03'03'03'03'03'03'03' 0239   254                  .BYTE   3[20]             ; init. as blanks                [31]
            03'03'03'03'03'03'03'03' 0245
                      00 024D   255 EXP_REV:        .BYTE   0                 ; Character Storage                      [31]
03'03'03'03'03'03'03'03'03'03'03'03' 024E   256                  .BYTE   3[20]             ; init. as blanks                [31]
            03'03'03'03'03'03'03'03' 025A
                        0262   257
                00000008  0262   258 FAO_DESC:        .LONG   8                 ; Descriptor                             [31]
                00000239' 0266   259                  .LONG   ACT_REV+1         ; for                                    [31]
                    0000  026A   260 FAO_LEN:         .WORD   0                 ; with length = 0                        [31]
```

```
                                   026C
                        00000008   026C  FAO_DESC_2:     .LONG   8                  ; Descriptor                          [31]
                        0000024E'  0270                  .LONG   EXP_REV+1          ; for                                 [31]
                            0000   0274  FAO_LEN_2:      .WORD   0                  ; with length = 0                     [31]
                                   0276
        4C 55 21 0000027E'010E0000'  0276  CONTROL:        .ASCID  /!UL/              ; CONTROL STRING FOR FORMATING        [31]
                                   0281
                                   0281
                                   0281  FMT_BEG_ERR:                                ; Printrev error message              [21]
73 69 76 65 72 20 43 41 21 2F 21 00'  0281          .ASCIC  "!/!AC revision level of !AC"
66 6F 20 6C 65 76 65 6C 20 6E 6F 69   028D
                      43 41 21 20     0299
                               1B     0281
                                   029D
                                   029D  FMT_MODULE:
           20 29 43 41 21 28 20 00'  029D          .ASCIC  " (!AC) "
                               07     029D
                                   02A5
                                   02A5  FMT_END_ERR:
61 68 74 20 43 41 21 20 73 69 20 00'  02A5          .ASCIC  " is !AC that expected by!/"-
62 20 64 65 74 63 65 70 78 65 20 74   02B1
                         2F 21 79     02BD
70 20 63 69 74 73 6F 6E 67 61 69 64   02C0                  "diagnostic program.!/"-
        2F 21 2E 6D 61 72 67 6F 72    02CC
43 41 21 20 20 20 20 20 20 20 20 20   02D5                  "         !AC revision level:          !ZL!/"-
65 6C 20 6E 6F 69 73 69 76 65 72 20   02E1
   2F 21 4C 5A 21 09 09 3A 6C 65 76   02ED
76 65 52 20 20 20 20 20 20 20 20 20   02F8                  "         Revision level expected by program:   !ZL!/"
20 6C 65 76 65 6C 20 6E 6F 69 73 69   0304
20 79 62 20 64 65 74 63 65 70 78 65   0310
4C 5A 21 09 3A 6D 61 72 67 6F 72 70   031C
                            2F 21     0328
                               84     02A5
                                   032A
                                   032A  FMT_H_END_ERR:
61 68 74 20 43 41 21 20 73 69 20 00'  032A          .ASCIC  " is !AC that expected by!/"-
62 20 64 65 74 63 65 70 78 65 20 74   0336
                         2F 21 79     0342
70 20 63 69 74 73 6F 6E 67 61 69 64   0345                  "diagnostic program.!/"-
        2F 21 2E 6D 61 72 67 6F 72    0351
43 41 21 20 20 20 20 20 20 20 20 20   035A                  "         !AC revision level:          !AC!/"-
65 6C 20 6E 6F 69 73 69 76 65 72 20   0366
   2F 21 43 41 21 09 09 3A 6C 65 76   0372
76 65 52 20 20 20 20 20 20 20 20 20   037D                  "         Revision level expected by program:   !AC!/"
20 6C 65 76 65 6C 20 6E 6F 69 73 69   0389
20 79 62 20 64 65 74 63 65 70 78 65   0395
43 41 21 09 3A 6D 61 72 67 6F 72 70   03A1
                            2F 21     03AD
                               84     032A
                                   03AF
                                   03AF  FMT_P_F_END:                                ; Printrev revision message           [28]
        2F 21 4C 5A 21 20 20 3A 00'  03AF          .ASCIC  ":  !ZL!/"
                               08     03AF
                                   03B8
                                   03B8  FMT_H_END:                                  ; Printrev revision message     [31]
        2F 21 43 41 21 20 20 3A 00'  03B8          .ASCIC  ":  !AC!/"
                               08     03B8
```

```
                                03C1   292
                                03C1   293 HARDMESSA:                              ; DSX$PRINTREV device type message      [21]
        65 72 61 77 64 72 61 48 00' 03C1   294         .ASCIC  "Hardware"
                                08  03C1
                                03CA   295
                                03CA   296 FIRMMESSA:                              ; printrev device type message          [21]
        65 72 61 77 6D 72 69 46 00' 03CA   297         .ASCIC  "Firmware"
                                08  03CA
                                03D3   298
                                03D3   299 PATCHMESSA:
        20 20 20 68 63 74 61 50 00' 03D3   300         .ASCIC  "Patch   "
                                08  03D3
                                03DC   301
                                03DC   302 LESSTHAN:                               ; DSX$PRINTREV actual rev. to expected rev[2
      6E 61 68 74 20 73 73 65 6C 00' 03DC   303         .ASCIC  "less than"
                                09  03DC
                                03E6   304
                                03E6   305 GREATTHAN:                              ; Printrev relation message             [21]
61 68 74 20 72 65 74 61 65 72 67 00' 03E6   306         .ASCIC  "greater than"
                                6E  03F2
                                0C  03E6
                                03F3   307
                                03F3   308 EQUALMESS:                              ; Used in printrev, relation or revision lev
        6F 74 20 6C 61 75 71 65 00' 03F3   309         .ASCIC  "equal to"
                                08  03F3
                                03FC   310
                        0000000D  03FC   311 CR=13
                        0000000A  03FC   312 LF=10
                             0A 0D  03FC   313 Page_String:    .ascii  <CR><LF> -      ;                                      [16]
4E 52 55 54 45 52 20 73 73 65 72 50 03FE   314                         /Press RETURN for more, Control-C to exit.../   ;      [16]
43 20 2C 65 72 6F 6D 20 72 6F 66 20 040A
20 6F 74 20 43 2D 6C 6F 72 74 6E 6F 0416
              2E 2E 2E 74 69 78 65  0422
                                0429   315
                                0429   316 DS$GB_VDS_Debug::                       ; Make more contemporary.               [15]
                                0429   317 Debug_The_Sucker::                      ; Print typecode number if = 1          [15]
                             00 0429   318         .Byte   0                      ; Initial 0                             [15]
```

```
                     042A   320          .SubTitle       Data Psect Declarations
                 00000000   321          .Psect  DATA, SHR, NOEXE, NOWRT, BYTE
                     0000   322
                     0000   323 ;+
                     0000   324 ; This address list is indexed by (-CODE-1) passed to DS$CVTREG as MNEADR
                     0000   325 ; Each address points to data in ONLY??? lists of the form:
                     0000   326 ;       CVT$T_....:
                     0000   327 ;                .ADDRESS        10$     Address of ASCIC string
                     0000   328 ;                .LONG   V1,...V6        Parameters V1-V6, only if required
                     0000   329 ;       10$:     .ASCIC  "mnemonic string"
                     0000   330 ;-
                     0000   331
                     0000   332 CVT$A_CODELIST:
            00000000' 0000   333          .ADDRESS        CVT$T_UBADPR    ; Args for UNIBUS datapath CVTREG
            00000000' 0004   334          .ADDRESS        CVT$T_UBAMAP    ; Args for UNIBUS MAP CVTREG
            00000000' 0008   335          .ADDRESS        CVT$T_MBASR     ; Args for MASSBUS Status CVTREG
            00000000' 000C   336          .ADDRESS        CVT$T_MBAMAP    ; Args for MASSBUS MAP CVTREG
```

```
                     0010   338          .Subtitle       DSV$SetPage Routine - Set terminal page size
                 00000000   339          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0000   340 ;**                                                                          [16]
                     0000   341 ; Functional Description:                                                  [16]
                     0000   342 ;                                                                          [16]
                     0000   343 ;       This is a CLI command routine.  It loads the global DS$GL_PAGE with [16]
                     0000   344 ;       a value specified to CLI.  If the page size is out of range, an error [16]
                     0000   345 ;       message is typed and it returns a failure status code (though CLI  [16]
                     0000   346 ;       does not currently recognize return status).                       [16]
                     0000   347 ;                                                                          [16]
                     0000   348 ; Calling sequence:                                                       [16]
                     0000   349 ;                                                                          [16]
                     0000   350 ;       JSB     DSV$SETPAGE                                              [16]
                     0000   351 ;                                                                          [16]
                     0000   352 ; Input Parameters:                                                       [16]
                     0000   353 ;                                                                          [16]
                     0000   354 ;       none                                                             [16]
                     0000   355 ;                                                                          [16]
                     0000   356 ; Output Parameters:                                                      [16]
                     0000   357 ;                                                                          [16]
                     0000   358 ;       none                                                             [16]
                     0000   359 ;                                                                          [16]
                     0000   360 ; Implicit Inputs:                                                       [16]
                     0000   361 ;                                                                          [16]
                     0000   362 ;       R2 points to CLI data table                                      [16]
                     0000   363 ;                                                                          [16]
                     0000   364 ; Implicit Outputs:                                                      [16]
                     0000   365 ;                                                                          [16]
                     0000   366 ;       none                                                             [16]
                     0000   367 ;                                                                          [16]
                     0000   368 ; Side effects:                                                          [16]
                     0000   369 ;                                                                          [16]
                     0000   370 ;       Changes the value of DS$GL_PAGE variable                         [16]
                     0000   371 ;                                                                          [16]
                     0000   372 ; Status returns:                                                        [16]
                     0000   373 ;                                                                          [16]
                     0000   374 ;       0       failure (value out of range)                             [16]
                     0000   375 ;       1       success                                                  [16]
                     0000   376 ;--                                                                       [16]
```

```
                                0000   378 DSV$SetPage::                                  ;                                        [16]
          50    1C A2    D0     0000   379          MOVL    CLI$L_DATA(R2), R0           ; Get the page size                      [16]
                   0B    19     0004   380          BLSS    10$                          ; Too small?                             [16]
                                0006   381
   000001DD'EF    50    D0     0006   382          MOVL    R0, DS$GL_PAGE               ; No, store the value                    [16]
                                000D   383
             50    01    D0     000D   384          MOVL    #1, R0                       ; Success                                [16]
                         05     0010   385          RSB                                  ;                                        [16]
                                0011   386
                                0011   387 10$:     $Print  -                            ; Page size out of range                 [16]
                                0011   388                  #DS$K_Type_Command_Err, -          ; typecode                        [16]
                                0011   389                  #DS$K_PrintF, -              ;                                        [16]
                                0011   390                  T_Page                       ; the message                            [16]
        000001E1'EF    9F     0011                        PUSHAB  T_Page
             7E    01    9A     0017                        movzbl  #DS$K_PrintF, -(sp)
             7E    15    98     001A                        cvtbl   #DS$K_Type_Command_Err, -(sp)
   000002D3'GF    03    FB     001D                        CALLS   #$$N+2, G^DSX$PRINT
                   50    D4     0024   391          CLRL    R0                           ; Failure                                [16]
                         05     0026   392          RSB                                  ;                                        [16]
```

```
                     0027   394          .Subtitle       DSV$ShowPage Routine - Show terminal page size
                     0027   395 ;++                                                                          [16]
                     0027   396 ; Functional Description:                                                  [16]
                     0027   397 ;                                                                           [16]
                     0027   398 ;       This is a CLI command routine.  It prints the current DS$GL_PAGE.  [16]
                     0027   399 ;                                                                           [16]
                     0027   400 ; Calling sequence:                                                       [16]
                     0027   401 ;                                                                           [16]
                     0027   402 ;       JSB     DSV$SHOWPAGE                                              [16]
                     0027   403 ;                                                                           [16]
                     0027   404 ; Input Parameters:                                                       [16]
                     0027   405 ;                                                                           [16]
                     0027   406 ;       none                                                              [16]
                     0027   407 ;                                                                           [16]
                     0027   408 ; Output Parameters:                                                      [16]
                     0027   409 ;                                                                           [16]
                     0027   410 ;       none                                                              [16]
                     0027   411 ;                                                                           [16]
                     0027   412 ; Implicit Inputs:                                                       [16]
                     0027   413 ;                                                                           [16]
                     0027   414 ;       none                                                              [16]
                     0027   415 ;                                                                           [16]
                     0027   416 ; Implicit Outputs:                                                      [16]
                     0027   417 ;                                                                           [16]
                     0027   418 ;       none                                                              [16]
                     0027   419 ;                                                                           [16]
                     0027   420 ; Side effects:                                                          [16]
                     0027   421 ;                                                                           [16]
                     0027   422 ;       none                                                              [16]
                     0027   423 ;
                     0027   424 ; Status returns:                                                        [16]
                     0027   425 ;                                                                           [16]
                     0027   426 ;       1       success                                                   [16]
                     0027   427 ;--                                                                        [16]
```

```
                                 0027   429 DSV$ShowPage::                                  ;                                        [16]
      50   000001DD'EF    9A     0027   430         MOVZBL  L^DS$GL_Page, R0                ; Get the current page size              [16]
                                 002E   431         $Print  -                               ; Print with typecode                    [16]
                                 002E   432                 #DS$K_Type_Command_Out, -           ; typecode                       [16]
                                 002E   433                 #DS$K_PrintF, -                 ; the type of PRINT routine              [16]
                                 002E   434                 T_Show_Page, -                  ; the message                            [16]
                                 002E   435                 R0                              ; the page size                          [16]
                      50    DD     002E                     PUSHL   R0
           00000204'EF    9F     0030                     PUSHAB  T_Show_Page
                 7E   01    9A     0036                     movzbl  #DS$K_PrintF, -(sp)
                 7E   16    98     0039                     cvtbl   #DS$K_Type_Command_Out, -(sp)
      000002D3'GF    04    FB     003C                     CALLS   #$$N+2, G^DSX$PRINT
                 50   01    D0     0043   436         MOVL    #1, R0                          ; Success                                [16]
                            05     0046   437         RSB                                     ;                                        [16]
                                 0047   438
```

```
                     0047   440          .Subtitle       DSX$Type_Out Routine - Do actual print
                 00000047   441          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0047   442 ;++
                     0047   443 ; Functional Description:
                     0047   444 ;
                     0047   445 ;       This routine does the actual work of outputting.  If the flag
                     0047   446 ;       DSA$V_BATCH is set, then it will use RMS to output to a file
                     0047   447 ;       (presumably) otherwise it will use QIO.  This routine also does          [16]
                     0047   448 ;       the work for SET PAGE online and standalone by traipsing through         [16]
                     0047   449 ;       the ouput buffer and comparing the number of linefeeds to the value      [16]
                     0047   450 ;       specified by the most recent SET PAGE command.  The routine also must    [16]
                     0047   451 ;       predict where the VDS and VMS will insert a CRLF due to the line being   [16]
                     0047   452 ;       too long for the current terminal width.;
                     0047   453 ;
                     0047   454 ; Input Parameters:
                     0047   455 ;
                     0047   456 ;       4(ap)           Length of string to output
                     0047   457 ;       8(ap)           Address of string
                     0047   458 ;
                     0047   459 ; Output Parameters:
                     0047   460 ;
                     0047   461 ;       none
                     0047   462 ;
                     0047   463 ; Implicit Inputs:
                     0047   464 ;
                     0047   465 ;       DS$GB_WIDTH     terminal width
                     0047   466 ;       DS$GL_PAGE      terminal page size
                     0047   467 ;
                     0047   468 ; Implicit Outputs:
                     0047   469 ;
                     0047   470 ;       none
                     0047   471 ;
                     0047   472 ; Side effects:
                     0047   473 ;
                     0047   474 ;       If either the CRD_AutoTest bit or the CRD_MenuTest bit is set (in the    [15]
                     0047   475 ;       DSA flags), this routine will call a CRD routine to determine if the     [15]
                     0047   476 ;       output should actually be printed. If the CRD routine returns success,  [15]
                     0047   477 ;       do NOT print.                                                            [15]
                     0047   478 ;
                     0047   479 ; Status returns:
                     0047   480 ;
                     0047   481 ;       Any code from QIOW or PUT
                     0047   482 ;--
```

```
                    00FC  0047   484 .Entry DSX$Type_Out, ^M<R2,R3,R4,R5,R6,R7>              ;                          [16]
                          0049   485
00000000'EF   1A    91    0049   486         CmpB    #DS$K_Type_CRD_AutoTest, - ; Compare the current typecode to ...[13]
                          0050   487                   L^DS$GB_TypeCode       ; ... to CRD_AutoTest. The same?          [13]
              35    13    0050   488         Beql    50$                      ; If same, branch to print it.            [13]
                          0052   489                                          ; If not same, continue ...               [13]
                          0052   490
                          0052   491         Br_If_CRD_AutoTest_Off  10$      ; If under Auto Test, then branch         [17]
0000FE00'EF   0B    E0    0052               BBS     #DSA$V_CRD_Autotest_Off, -       ; If bit set, branch              [36]
              12          0059                         L^DSA$GL_Flags, 10$           ;                                 [36]
                          005A   492         Br_If_CRD_MenuTest_Off  10$      ; If under Menu Test, then branch         [17]
0000FE00'EF   10    E0    005A               BBS     #DSA$V_CRD_MenuTest_Off, -       ; If bit set, branch              [36]
              0A          0061                         L^DSA$GL_Flags, 10$           ;                                 [36]
                          0062   493         Br_If_CRD_MenuTest_On   10$      ; If under Menu Test, then branch         [20]
0000FE00'EF   12    E0    0062               BBS     #DSA$V_CRD_MenuTest_On, -        ; If bit set, branch              [36]
              02          0069                         L^DSA$GL_Flags, 10$           ;                                 [36]
              1B    11    006A   494         Brb     50$                      ; Else, not under either one.             [15]
                          006C   495
           08 AC    DD    006C   496 10$:    PushL   8(AP)                    ; Push Address as param-4                 [15]
      7E   04 AC    3C    006F   497         MovZWL  4(AP), -(SP)             ; Push Length as param-3                  [11]
7E   00000000'EF    9A    0073   498         MovZBL  L^DS$GB_TypeCode, -(SP) ; Push TypeCode as param-2                [11]
              01    DD    007A   499         PushL   #CRD$K_TypeCode          ; Push "type of call" as param-1          [11]
00000000'FF   04    FB    007C   500         CallS   #4, @L^DS$GA_CRD_DS_Interface   ; Call indirectly the ...         [11]
                          0083   501                                          ; ... CRD$DS_Interface routine.           [11]
           01 50    E9    0083   502         BlbC    R0, 50$                  ; Print if the routine returns failure    [11]
                    04    0086   503         Ret                              ; Return if routine returns success       [11]
                          0087   504
                          0087   505 50$:    Br_If_Not_Batch 100$             ; Branch if use QIO for output            [11]
00000000'EF   16    E1    0087               BBC     #DS$V_Batch, -                   ; If bit not set, branch          [28]
              3F          008E                         L^DS$GL_Flags, 100$           ;                                 [28]
                          008F   506
                          008F   507         ;+
                          008F   508         ; We are running in Batch mode. That is, we are using RMS to output       [14]
                          008F   509         ; everything. Test to see if either CRD Auto Test or Menu Test is         [15]
                          008F   510         ; executing. If so, assume the BINARY flag has been set and remove the    [15]
                          008F   511         ; binary typecode from the start of the string.                           [15]
                          008F   512         ;-
                          008F   513
50   00000000'EF    9E    008F   514         MovAB   L^DS$Rab_Output, R0      ; Point to RAB                            [09]
      51   04 AC    3C    0096   515         MovZWL  4(AP), R1                ; Length to R1                            [14]
      52   08 AC    D0    009A   516         MovL    8(AP), R2                ; Address to R2                           [14]
                          009E   517
                          009E   518         Br_If_CRD_AutoTest_Off  60$      ; If running under Auto Test, branch      [17]
0000FE00'EF   0B    E0    009E               BBS     #DSA$V_CRD_Autotest_Off, -       ; If bit set, branch              [36]
              12          00A5                         L^DSA$GL_Flags, 60$           ;                                 [36]
                          00A6   519         Br_If_CRD_MenuTest_Off  60$      ; If running under Menu Test, branch      [17]
0000FE00'EF   10    E0    00A6               BBS     #DSA$V_CRD_MenuTest_Off, -       ; If bit set, branch              [36]
              0A          00AD                         L^DSA$GL_Flags, 60$           ;                                 [36]
                          00AE   520         Br_If_CRD_MenuTest_On   60$      ; If running under Menu Test, branch      [20]
0000FE00'EF   12    E0    00AE               BBS     #DSA$V_CRD_MenuTest_On, -        ; If bit set, branch              [36]
              02          00B5                         L^DSA$GL_Flags, 60$           ;                                 [36]
              04    11    00B6   521         Brb     70$                      ; Not either one, so skip over it         [15]
                          00B8   522
              51    D7    00B8   523 60$:    DecL    R1                       ; Length = Length - 1                     [15]
              52    D6    00BA   524         IncL    R2                       ; Address = Address + 1 (byte)            [14]
                          00BC   525
      22 A0   51    B0    00BC   526 70$:    MovW    R1, B^Rab$W_Rsz(R0)      ; Get record length                       [14]
```

```
                28 A0    52   D0  00C0    527          MovL    R2, B^Rab$L_Rbf(R0)     ; Get record address                    [14]
                                  00C4    528          $Put    Rab=(R0)                ; Output string                         [09]
                         60   DF  00C4                 PUSHAL  (R0)
        00000000'GF      01   FB  00C6                 CALLS   #$$.TMP1,G^SYS$PUT
                              04  00CD    529          Ret                             ; Go to caller.                         [09]
                                  00CE    530
                                  00CE    531          ;+
                                  00CE    532          ; We are not running in Batch mode - we are using QIO to output     [14]
                                  00CE    533          ; everything. Perform the same check as the above performs for CRD. [14]
                                  00CE    534          ;-
                                  00CE    535
          03 00000000'EF      E9  00CE    536 100$:    blbc    L^ds$gb_qioact, 200$    ; Branch if QIO not active              [09]
                      0396    30  00D5    537          bsbw    reset_input             ; Reset input channel if active         [09]
                              00D8    538
                         7E   7C  00D8    539 200$:    clrq    -(sp)                   ; p5/p6 zero                            [09]
                         7E   7C  00DA    540          clrq    -(sp)                   ; p3/p4 zero                            [09]
                              00DC    541
                 51   04 AC   3C  00DC    542          MovZWL  4(AP), R1               ; Move length to R1                     [14]
                 52   08 AC   D0  00E0    543          MovL    8(AP), R2               ; Move address to R2                    [14]
                              00E4    544
                              00E4    545          Br_If_CRD_AutoTest_Off  250$    ; If running under Auto Test, branch    [17]
        0000FE00'EF      0B  E0  00E4                 BBS     #DSA$V_CRD_Autotest_Off, -      ; If bit set, branch            [36]
                         12           00EB                    L^DSA$GL_Flags, 250$    ;                               [36]
                              00EC    546          Br_If_CRD_MenuTest_Off  250$    ; If running under Menu Test, branch    [17]
        0000FE00'EF      10  E0  00EC                 BBS     #DSA$V_CRD_MenuTest_Off, -      ; If bit set, branch            [36]
                         0A           00F3                    L^DSA$GL_Flags, 250$    ;                               [36]
                              00F4    547          Br_If_CRD_MenuTest_On   250$    ; If running under Menu Test, branch    [20]
        0000FE00'EF      12  E0  00F4                 BBS     #DSA$V_CRD_MenuTest_On, -       ; If bit set, branch            [36]
                         02           00FB                    L^DSA$GL_Flags, 250$    ;                               [36]
                         04  11  00FC    548          Brb     300$                    ; Not either one, so skip over it       [15]
                              00FE    549
                         51  D7  00FE    550 250$:    DecL    R1                      ; Length = Length - 1                   [15]
                         52  D6  0100    551          IncL    R2                      ; Address = Address + 1 (byte)          [14]
                              0102    552
                    55   51  D0  0102    553 300$:    MovL    R1, R5                  ; R5 gets length                        [16]
                         54  D4  0105    554          ClrL    R4                      ; Number of characters in current line  [16]
                    56   55  D0  0107    555 310$:    Movl    R5, R6                  ; Save original length of string        [16]
                    57   52  D0  010A    556          Movl    R2, R7                  ; Save original address of string       [16]
                000001DD'EF  D5  010D    557          TstL    DS$GL_Page              ; SET PAGE 0 ?                          [16]
                         73  13  0113    558          BEql    370$                    ; Yes, go right to QIOW                 [16]
                              0115    559
    000001DD'EF 000001D9'EF  D1  0115    560          CmpL    Line_Count, DS$GL_Page  ; Are we at page limit?                 [16]
                         03  19  0120    561          BLss    320$                    ; No                                    [16]
                      0087   31  0122    562          BrW     380$                    ; Yes, prompt user                      [16]
                              0125    563
              53 00000000'EF  9A  0125    564 320$:    MovZBL  DS$GB_Width, R3         ; Get terminal width                    [16]
                 000001D8'EF  94  012C    565          ClrB    Width_Too_Large         ; Clear flag indicating line too long   [16]
                              0132    566
                         55  D7  0132    567 330$:    DecL    R5                      ; Any characters left?                  [16]
                         4C  19  0134    568          BLss    355$                    ; No, we have less than a page of chars [16]
                              0136    569
                    62   0A  91  0136    570          CmpB    #^X0A, (R2)             ; LF ?                                  [16]
                         37  13  0139    571          BEql    340$                    ; Yes                                   [16]
                              013B    572
                    82   0D  91  013B    573          CmpB    #^X0D, (R2)+            ; No, CR ?                              [16]
                         04  12  013E    574          BNEq    331$                    ; No                                    [16]
                         54  D4  0140    575          ClrL    R4                      ; Yes, zero width count                 [16]
```

```
                        EE   11  0142   576          BrB     330$                      ; Continue with next character          [16]
                                 0144   577
                FF A2   09   91  0144   578 331$:    CmpB    #^X09, -1(R2)             ; No, TAB ?                             [16]
                        09   12  0148   579          BNEq    335$                      ; No                                    [16]
                        54   D6  014A   580 333$:    IncL    R4                        ; Yes, add 1 to width                   [16]
                   54   07   93  014C   581          BiTB    #7, R4                    ; Are we at TAB boundary?               [16]
                        F9   12  014F   582          BNEq    333$                      ; No, count another                     [16]
                        54   D7  0151   583          DecL    R4                        ; Yes, we added one to many so subtract [16]
                DB 54   53   F2  0153   584 335$:    AOBLss  R3, R4, 330$              ; Width gets larger by 1                [16]
                                 0157   585          Br_If_Not_User  337$              ; S/A doesn't pad next line if TAB      [16]
          0000FE00'EF   1C   E1  0157                BBC     #DSA$V_User, -                    ; If bit not set, branch
                        05       015E                          L^DSA$GL_Flags, 337$          ;
                   54   53   C2  015F   586          SubL2   R3, R4                    ; How far over the edge of width?       [16]
                        12   12  0162   587          BNEq    353$                      ; Too far, don't do extra CRLF, save R4 [16]
                                 0164   588                                            ; We are at the edge of the width,      [16]
                   62   0D   91  0164   589 337$:    CmpB    #^X0D, (R2)               ; If CR is next, line is not too long,  [16]
                        C9   13  0167   590          BEql    330$                      ; so print CRLF also                    [16]
          000001D8'EF   01   88  0169   591          BiSB    #1, Width_Too_Large       ; No CRLF there, we need to do an extra [16]
                                 0170   592                                            ; CRLF if we are also at the page limit [16]
                        02   11  0170   593          BrB     350$                      ; Count another line                    [16]
                                 0172   594
                        52   D6  0172   595 340$:    IncL    R2                        ; Point to next character               [16]
                        54   D4  0174   596 350$:    ClrL    R4                        ; Clear width count                     [16]
  A3 000001D9'EF   000001DD'EF   F2  0176   597 353$:    AOBLss  DS$GL_Page, Line_Count, 320$    ; Enough for a page?            [16]
                                 0182   598
                        54   D4  0182   599 355$:    ClrL    R4                        ; Clear width count                     [16]
                55   52   57   C3  0184   600          SubL3   R7, R2, R5                ; R5 gets length of this string         [16]
                                 0188   601
                                 0188   602 370$:
                        7E   7C  0188   603          CLRQ    -(SP)                     ; Two null arguments            [19]
                        7E   7C  018A   604          CLRQ    -(SP)                     ; Two null arguments            [19]
                        55   DD  018C   605          PushL   R5                        ; Push length of string                 [16]
                        57   DD  018E   606          PushL   R7                        ; Push address of string                [16]
                                 0190   607
                        7E   7C  0190   608          clrq    -(sp)                     ; astadr/astprm zero                    [14]
                        7E   D4  0192   609          clrl    -(sp)                     ; iosb zero                             [09]
                        00'  DD  0194   610          pushl   S^#io$_writevblk          ; function                              [09]
                00000000'EF  DD  0196   611          pushl   L^ds$gw_ttout             ; channel                               [09]
                        7E   D4  019C   612          clrl    -(sp)                     ; event flag                            [09]
          00000000'GF   0C   FB  019E   613          calls   #12, G^sys$qiow           ; do the qio                            [09]
                                 01A5   614
                55   56   55   C3  01A5   615          SubL3   R5, R6, R5                ; R1 gets number of characters left     [16]
                        01   14  01A9   616          BGtr    380$                      ; There's more, prompt user             [16]
                             04  01AB   617          Ret                               ; None left                             [16]
                                 01AC   618
          000001D8'EF   01   93  01AC   619 380$:    BiTB    #1, Width_Too_Large       ; Was the last line too long?           [16]
                        29   13  01B3   620          BEql    385$                      ; No                                    [16]
                                 01B5   621          $QIOW_S , -                       ; Yes, do an extra CRLF before prompt   [16]
                                 01B5   622                  L^DS$GW_TTOUT, -          ; Channel number                        [16]
                                 01B5   623                  S^#IO$_WRITEVBLK, -       ; Function                              [16]
                                 01B5   624                  P1 = Page_String, -       ; Output buffer address                 [16]
                                 01B5   625                  P2 = #2                   ; Output buffer size                    [16]
                        7E   7C  01B5                        CLRQ    -(SP)
                        7E   7C  01B7                        CLRQ    -(SP)
                        02   DD  01B9                        PUSHL   #2
                000003FC'EF  DF  01BB                        PUSHAL  Page_String
                        7E   7C  01C1                        CLRQ    -(SP)
```

```
                 00   DD  01C3                          PUSHL   #0
            7E   00'  3C  01C5                          MOVZWL  S^#IO$_WRITEVBLK,-(SP)
7E   00000000'EF      3C  01C8                          MOVZWL  L^DS$GW_TTOUT,-(SP)
                 00   DD  01CF                          PUSHL   #0
00000000'GF      0C   FB  01D1                          CALLS   #12,G^SYS$QIOW
     000001D8'EF      94  01D8   626          ClrB      Width_Too_Large          ; Reset width flag                    [16]
                          01DE   627
                          01DE   628 385$:    $QIOW_S , -                        ; Prompt for more output              [16]
                          01DE   629                    L^DS$GW_TTOUT, -         ; Channel number                      [16]
                          01DE   630                    #IO$_WRITEVBLK, -        ; Function code                       [16]
                          01DE   631                    P1 = Page_String, -      ; Output buffer address               [16]
                          01DE   632                    P2 = #45                 ; Output buffer size                  [16]
                 7E   7C  01DE                          CLRQ    -(SP)
                 7E   7C  01E0                          CLRQ    -(SP)
                 2D   DD  01E2                          PUSHL   #45
     000003FC'EF      DF  01E4                          PUSHAL  Page_String
                 7E   7C  01EA                          CLRQ    -(SP)
                 00   DD  01EC                          PUSHL   #0
   7E   0000'8F       3C  01EE                          MOVZWL  #IO$_WRITEVBLK,-(SP)
7E   00000000'EF      3C  01F3                          MOVZWL  L^DS$GW_TTOUT,-(SP)
                 00   DD  01FA                          PUSHL   #0
00000000'GF      0C   FB  01FC                          CALLS   #12,G^SYS$QIOW
                 7E   94  0203   633          ClrB      -(SP)                    ; Space for input buffer              [16]
            57   6E   DE  0205   634          MovAL     (SP), R7                 ; Input buffer on stack               [16]
                          0208   635 387$:    Br_If_Not_User  388$               ;                                     [16]
0000FE00'EF      1C   E1  0208                  BBC     #DSA$V_User, -                   ; If bit not set, branch
                 2A       020F                            L^DSA$GL_Flags, 388$           ;
00000000'EF      01   88  0210   636          BisB      #1, DS$GB_QIOACT         ; Set input QIO active                [16]
                          0217   637                                            ; In user mode,                       [16]
                          0217   638                                            ; suppress echo of termination char   [16]
                          0217   639                                            ; because VMS sometimes echoes it and [16]
                          0217   640                                            ; sometimes doesn't.                  [16]
                          0217   641          $QIOW_S , -                        ; Read the user's response            [16]
                          0217   642                    L^DS$GW_TTIN, -          ; Channel number                      [16]
                          0217   643                    #IO$_READVBLK!IO$M_TRMNOECHO, - ; Function                    [16]
                          0217   644                    P1 = (R7), -             ; Input buffer address                [16]
                          0217   645                    P2 = #1                  ; Input buffer size                   [16]
                 7E   7C  0217                          CLRQ    -(SP)
                 7E   7C  0219                          CLRQ    -(SP)
                 01   DD  021B                          PUSHL   #1
                 67   DF  021D                          PUSHAL  (R7)
                 7E   7C  021F                          CLRQ    -(SP)
                 00   DD  0221                          PUSHL   #0
   7E   0000'8F       3C  0223                          MOVZWL  #IO$_READVBLK!IO$M_TRMNOECHO,-(SP)
7E   00000000'EF      3C  0228                          MOVZWL  L^DS$GW_TTIN,-(SP)
                 00   DD  022F                          PUSHL   #0
00000000'GF      0C   FB  0231                          CALLS   #12,G^SYS$QIOW
                 21   11  0238   646          BrB       389$                     ;                                     [16]
                          023A   647 388$:    $QIOW_S , -                        ; Read the user's response            [16]
                          023A   648                    L^DS$GW_TTIN, -          ; Channel number                      [16]
                          023A   649                    #IO$_READVBLK!IO$M_TRMNOECHO, - ; Function                    [16]
                          023A   650                    P1 = (R7), -             ; Input buffer address                [16]
                          023A   651                    P2 = #0                  ; S/A buffer works differently        [16]
                 7E   7C  023A                          CLRQ    -(SP)
                 7E   7C  023C                          CLRQ    -(SP)
                 00   DD  023E                          PUSHL   #0
                 67   DF  0240                          PUSHAL  (R7)
```

```
                 7E   7C  0242                    CLRQ    -(SP)
                 00   DD  0244                    PUSHL   #0
        7E   0000'8F  3C  0246                    MOVZWL  #IO$_READVBLK!IO$M_TRMNOECHO,-(SP)
   7E   00000000'EF   3C  024B                    MOVZWL  L^DS$GW_TTIN,-(SP)
                 00   DD  0252                    PUSHL   #0
   00000000'GF   0C   FB  0254                    CALLS   #12,G^SYS$QIOW
            0D   6E   91  025B  652 389$:  CmpB    (SP),#^X0D                ; CR ?                                  [16]
                 0F   13  025E  653        BEql    390$                      ; Yes                                   [16]
                 8E   95  0260  654        TstB    (SP)+                     ; No, pop the input buffer in case ^C   [16]
        00000000'EF   16  0262  655        JSb     L^KB_Check                ; Did we get ^C ?                       [16]
                 7E   94  0268  656        ClrB    -(SP)                     ; If we come back from KB_Check, then   [16]
            57   6E   DE  026A  657        MovAL   (SP), R7                  ; it was not a ^C, so try again         [16]
                 99   11  026D  658        BrB     387$                      ;                                       [16]
                 8E   95  026F  659 390$:  TstB    (SP)+                     ; Pop the input buffer                  [16]
                          0271  660        Br_If_Not_User  393$              ;                                       [16]
   0000FE00'EF   1C   E1  0271            BBC     #DSA$V_User, -                    ; If bit not set, branch
                 23       0278                      L^DSA$GL_Flags, 393$            ;
                          0279  661        $QIOW_S , -                       ; Do an extra CRLF after prompt         [16]
                          0279  662                L^DS$GW_TTOUT, -          ; Channel number                        [16]
                          0279  663                S^#IO$_WRITEVBLK, -       ; Function                              [16]
                          0279  664                P1 = Page_String, -       ; Output buffer address                 [16]
                          0279  665                P2 = #2                   ; Output buffer size                    [16]
                 7E   7C  0279                    CLRQ    -(SP)
                 7E   7C  027B                    CLRQ    -(SP)
                 02   DD  027D                    PUSHL   #2
        000003FC'EF   DF  027F                    PUSHAL  Page_String
                 7E   7C  0285                    CLRQ    -(SP)
                 00   DD  0287                    PUSHL   #0
            7E   00'  3C  0289                    MOVZWL  S^#IO$_WRITEVBLK,-(SP)
   7E   00000000'EF   3C  028C                    MOVZWL  L^DS$GW_TTOUT,-(SP)
                 00   DD  0293                    PUSHL   #0
   00000000'GF   0C   FB  0295                    CALLS   #12,G^SYS$QIOW
                          029C  666 393$:  $QIOW_S , -                       ; Do an extra CRLF after prompt         [16]
                          029C  667                L^DS$GW_TTOUT, -          ; Channel number                        [16]
                          029C  668                S^#IO$_WRITEVBLK, -       ; Function                              [16]
                          029C  669                P1 = Page_String, -       ; Output buffer address                 [16]
                          029C  670                P2 = #2                   ; Output buffer size                    [16]
                 7E   7C  029C                    CLRQ    -(SP)
                 7E   7C  029E                    CLRQ    -(SP)
                 02   DD  02A0                    PUSHL   #2
        000003FC'EF   DF  02A2                    PUSHAL  Page_String
                 7E   7C  02A8                    CLRQ    -(SP)
                 00   DD  02AA                    PUSHL   #0
            7E   00'  3C  02AC                    MOVZWL  S^#IO$_WRITEVBLK,-(SP)
   7E   00000000'EF   3C  02AF                    MOVZWL  L^DS$GW_TTOUT,-(SP)
                 00   DD  02B6                    PUSHL   #0
   00000000'GF   0C   FB  02B8                    CALLS   #12,G^SYS$QIOW
                          02BF  671
        00000000'EF   94  02BF  672 395$:  ClrB    DS$GB_QIOACT              ; Clear QIO active                      [16]
        000001D9'EF   D4  02C5  673        ClrL    Line_Count                ; Reset line counter                    [16]
                 7E   7C  02CB  674        ClrQ    -(SP)                     ; P5/P6 zero                            [16]
                 7E   7C  02CD  675        ClrQ    -(SP)                     ; P4/P3 zero                            [16]
              FE35   31  02CF  676        BrW     310$                      ; Do another page                       [16]
                     04  02D2  677        ret                               ; Return                                [09]
                          02D3  678
```

```
                     680          .Subtitle       DSX$Print Routine - Load type code byte
                     681 ;++
                     682 ; Functional Description:
                     683 ;
                     684 ;       This routine is the control point for Supervisor type coding.
                     685 ;       It sets the typecode byte to the proper value, and calls the
                     686 ;       desired PRINT routine.  Any print which does not go through
                     687 ;       this routine, and which does not set DS$GB_TYPECODE explicitly
                     688 ;       will have the default (current) type code [normally 0].
                     689 ;
                     690 ; Input Parameters:
                     691 ;
                     692 ;       4(ap)           Type code value (negative of typecode indicates
                     693 ;                       that code is 'sticky', and will not be cleared.
                     694 ;       8(ap)           Which PRINTx is desired
                     695 ;       12(ap)          Address of ASCIC format string
                     696 ;       16(ap)          First FAO argument (as in DSX$PRINTx routines)
                     697 ;         .
                     698 ;         .
                     699 ;         .
                     700 ;
                     701 ; Output Parameters:
                     702 ;
                     703 ;       none
                     704 ;
                     705 ; Implicit Inputs:
                     706 ;
                     707 ;       none
                     708 ;
                     709 ; Implicit Outputs:
                     710 ;
                     711 ;       DS$GL_TYPECODE  set to input value.
                     712 ;
                     713 ; Side effects:
                     714 ;
                     715 ;       none
                     716 ;
                     717 ; Status returns:
                     718 ;
                     719 ;       none
                     720 ;--
```

```
                    0000  02D3   722 .Entry  dsx$print, ^M<>                          ;                                [09]
                 00   DD  02D5   723         pushl   #0                                ; Allocate a zero longword       [09]
          50   04 AC  D0  02D7   724         movl    4(ap), r0                         ; Copy type code                 [09]
                 50   D5  02DB   725         tstl    r0                                ; Check typecode                 [09]
                 05   18  02DD   726         bgeq    10$                               ; Branch if not sticky           [09]
             50  50   CE  02DF   727         mnegl   r0, r0                            ; If sticky, then negate and     [09]
                 6E   D6  02E2   728         incl    (sp)                              ; ... mark as sticky             [09]
                          02E4   729
   00000000'EF   50   90  02E4   730 10$:    movb    r0, L^ds$gb_typecode              ; Store the thing                [09]
                          02EB   731         case    src=8(ap),type=b,limit=#0, -      ; Case on the dispatch code      [09]
                          02EB   732                  displist=<-                      ;                                [09]
                          02EB   733                     20$, -                        ; PRINTI                         [09]
                          02EB   734                     30$, -                        ; PRINTF                         [09]
                          02EB   735                     40$, -                        ; PRINTB                         [09]
                          02EB   736                     50$>                          ; PRINTX                         [09]
   03'  00   08 AC   8F   02EB              CASEb   8(ap),#0,S^#<<30001$-30000$>/2>-1
                          02F0       30000$:
                    0008' 02F0              .SIGNED_WORD    20$-30000$
                    000E' 02F2              .SIGNED_WORD    30$-30000$
                    0014' 02F4              .SIGNED_WORD    40$-30000$
                    001A' 02F6              .SIGNED_WORD    50$-30000$
                          02F8       30001$:
                          02F8   737
          50   68'AF  9E  02F8   738 20$:    movab   B^dsx$printi, r0                  ; Address for PRINTI             [09]
                 10   11  02FC   739         brb     60$                               ;                                [09]
                          02FE   740
          50   53'AF  9E  02FE   741 30$:    movab   B^dsx$printf, r0                  ; Address for PRINTF             [09]
                 0A   11  0302   742         brb     60$                               ;                                [09]
                          0304   743
          50   3B'AF  9E  0304   744 40$:    movab   B^dsx$printb, r0                  ; Address for PRINTB             [09]
                 04   11  0308   745         brb     60$                               ;                                [09]
                          030A   746
          50   2E'AF  9E  030A   747 50$:    movab   B^dsx$printx, r0                  ; Address for PRINTX             [09]
                          030E   748
   08 AC   6C   02   C3  030E   749 60$:    subl3   #2, (ap), 8(ap)                   ; Shorten arg list               [09]
          60   08 AC  FA  0313   750         callg   8(ap), (r0)                       ; Call the routine               [09]
               06 8E  E8  0317   751         blbs    (sp)+, 70$                        ; Return if sticky type          [09]
       00000000'EF   94  031A   752         clrb    L^ds$gb_typecode                  ; Otherwise clear it             [09]
                          0320   753
                     04  0320   754 70$:    ret                                       ; Return                         [09]
```

```
0321    756          .Subtitle       DSX$PrintX Routines - Extended print routines
0321    757 ;++
0321    758 ; Functional Description:
0321    759 ;
0321    760 ;       DSX$PRINTS      ROUTINE TO PRINT SUMMARY MESSAGES
0321    761 ;       DSX$PRINTX      ROUTINE TO PRINT EXTENDED MESSAGES
0321    762 ;       DSX$PRINTB      ROUTINE TO PRINT BASIC MESSAGES
0321    763 ;       DSX$PRINTF      ROUTINE TO PRINT FORCED MESSAGES
0321    764 ;       DSX$PRINTI      ROUTINE TO PRINT EVEN IN APT MODE
0321    765 ;
0321    766 ; Calling Sequence:
0321    767 ;
0321    768 ;       Procedure Call Thru Vectors In Entry
0321    769 ;
0321    770 ; Input Parameters:
0321    771 ;
0321    772 ;       (AP)            = NUMBER OF PARAMETERS + 1
0321    773 ;       4(AP)           = ADDRESS OF COUNTED ASCII FORMAT STRING
0321    774 ;       8(AP)           = FIRST PARAMETER
0321    775 ;       (AP)[(AP)]      = LAST PARAMETER
0321    776 ;
0321    777 ; Implicit Inputs:
0321    778 ;
0321    779 ;       NONE
0321    780 ;
0321    781 ; Output Parameters:
0321    782 ;
0321    783 ;       NONE
0321    784 ;
0321    785 ; Implicit Outputs:
0321    786 ;
0321    787 ;       NONE
0321    788 ;
0321    789 ; Completion Codes:
0321    790 ;
0321    791 ;       NONE
0321    792 ;
0321    793 ; Side Effects:
0321    794 ;
0321    795 ;       NONE
0321    796 ;
0321    797 ;--
```

```
                        0321   799          .ENABL  LSB
                        0321   800
                  0000  0321   801 .ENTRY  DSX$PRINTS,^M<>                          ; ENTRY MASK
                        0323   802
                        0323   803          Br_If_Not_IES   30$                     ; If summary not inhibited, go  [11]
0000FE00'EF   07   E1   0323                BBC     #DSA$V_IES, -                  ; If bit not set, branch        [29]
                2A      032A                  L^DSA$GL_Flags, 30$                  ;                               [29]
                        032B   804 ;        BBC     #DSA$V_IES,L^DSA$GL_FLAGS,30$  ; If summary not inhibited go
              0131   31  032B   805          BRW     PRINT_X                        ; SUMMARY INHIBITED
                        032E   806
                  0000  032E   807 .ENTRY  DSX$PRINTX,^M<>
                        0330   808
                        0330   809          Br_If_Not_IE3   10$                     ; Inhibit Level-3 flag          [11]
0000FE00'EF   06   E1   0330                BBC     #DSA$V_IE3, -                  ; If bit not set, branch        [29]
                05      0337                  L^DSA$GL_Flags, 10$                  ;                               [29]
                        0338   810 ;        BBC     #DSA$V_IE3,L^DSA$GL_FLAGS,10$  ; INHIBIT LEVEL-3 FLAG
              0124   31  0338   811          BRW     PRINT_X                        ; LEVEL-3 MESSAGES INHIBITED
                        033B   812
                  0000  033B   813 .ENTRY  DSX$PRINTB,^M<>
                        033D   814
                        033D   815 10$:     Br_If_Not_IE2   20$                     ; Inhibit Level-2 flag          [11]
0000FE00'EF   05   E1   033D                BBC     #DSA$V_IE2, -                  ; If bit not set, branch        [29]
                03      0344                  L^DSA$GL_Flags, 20$                  ;                               [29]
                        0345   816 ;10$:    BBC     #DSA$V_IE2,L^DSA$GL_FLAGS,20$  ; INHIBIT LEVEL-2 FLAG
              0117   31  0345   817          BRW     PRINT_X                        ; LEVEL-2 INHIBITED
                        0348   818
                        0348   819 20$:     Br_If_Not_IE1   30$                     ; Inhibit Level-1 flag          [11]
0000FE00'EF   04   E1   0348                BBC     #DSA$V_IE1, -                  ; If bit not set, branch        [29]
                05      034F                  L^DSA$GL_Flags, 30$                  ;                               [29]
                        0350   820 ;20$:    BBC     #DSA$V_IE1,L^DSA$GL_FLAGS,30$  ; INHIBIT LEVEL-1 FLAG
              010C   31  0350   821          BRW     PRINT_X                        ; LEVEL-1 INHIBITED...
                        0353   822                                                 ; ... => ALL INHIBITED
                        0353   823
                  0000  0353   824 .ENTRY  DSX$PRINTF,^M<>
                        0355   825
                        0355   826 30$:     Br_If_Not_Apt   40$                     ; If not in APT mode, proceed   [11]
0000FE00'EF   1F   E1   0355                BBC     #DSA$V_Apt, -                  ; If bit not set, branch
                0D      035C                  L^DSA$GL_Flags, 40$                  ;
                        035D   827          Br_If_Not_NoRpt 40$                    ; If not supressing output, go  [11]
0000FE00'EF   1B   E1   035D                BBC     #DSA$V_NoRpt, -                ; If bit not set, branch        [29]
                05      0364                  L^DSA$GL_Flags, 40$                  ;                               [29]
                        0365   828 ;30$:    BBC     #DSA$V_APT, -                  ; IF NOT IN APT MODE PROCEED
                        0365   829 ;                L^DSA$GL_FLAGS,40$
                        0365   830 ;        BBC     #DSA$V_NORPT, -                ; APT MODE, IF NOT SUPPRESSING ...
                        0365   831 ;                L^DSA$GL_FLAGS,40$             ; ... OUTPUT, branch
              00F7   31  0365   832          BRW     PRINT_X                        ; YES, IN APT MODE AND OUTPUT ...
                        0368   833                                                 ; ... SUPPRESSED.
                        0368   834
                  0000  0368   835 .ENTRY  DSX$PRINTI,^M<>
                        036A   836
                        036A   837 40$:
    50   0000FE00'EF   DE   036A   838          MOVAL   L^DSA$GL_FLAGS,R0          ; POINT TO FLAGS LONGWORD
      5E   0001'CE   9E   0371   839          MOVAB   -DS$K_BUFSIZ+1(SP),SP      ; Allocate space                [09]
000001D4'EF   01   AE   9E   0376   840          MOVAB   1(SP), L^QBUFF+4           ;                               [09]
7E  60   F7FFFFFF   8F   CB   037E   841          BICL3   #^CDSA$M_NORPT,(R0),-(SP)  ; SAVE STATE OF DSA$M_NORPT
          60   6E   CA   0386   842          BICL2   (SP),(R0)                  ; CLEAR NORPT FLAG
                        0389   843          Set_Output                         ; Set OUTPUT flag               [11]
```

```
                00000000'EF   17   E2  0389                BBSS    #DS$V_Output, -          ; Branch if OUTPUT bit is set to        [29]
                              00       0390                  L^DS$GL_Flags, 30002$  ; ...                                   [29]
                                       0391          30002$:                              ; ... here. Set bit regardless.         [29]
                                       0391    844
                 7E    04 AC    01   C1  0391    845         ADDL3   #1,4(AP),-(SP)         ; COPY STRING ORIGIN
                       7E    04 BC   9A  0396    846         MOVZBL  @4(AP),-(SP)          ; COPY CONTROL STRING LENGTH
                             08 AC   DF  039A    847         PUSHAL  8(AP)                 ; ADDRESS OF INSERT LIST
                    000001D0'EF      7F  039D    848         PUSHAQ  L^QBUFF               ; ADDRESS OF OUTPUT BUFFER DESCRIPTOR   [07]
                    00000000'EF      3F  03A3    849         PUSHAW  L^DS$GL_BUFLEN        ; ADDRESS TO RETURN STRING LENGTH       [07]
                             0C AE   7F  03A9    850         PUSHAQ  12(SP)                ; ADDRESS OF CONTROL STRING DESCRIPTOR
                00000000'9F      06   FB  03AC    851         CALLS   #6,@#SYS$FAOL         ; DO THE EDITING
                                       03B3    852                                       ;                                       [18]
                                 01   50  D1  03B3    853         CMPL    R0,#SS$_NORMAL        ; ARE PRINT MACRO'S OK, CORRECT PARAM.  [18]
                                      03   13  03B6    854         BEQL    45$                   ;                                       [18]
                                    00A4   31  03B8    855         BRW     PRINT_X               ; EXIT IF NOT CORRECT.                  [18]
                                       03BB    856   45$:                                    ;                                       [18]
                                       03BB    857         Br_If_Not_Binary        60$       ; Skip funny stuff if not binary        [10]
                0000FE00'EF      01   E1  03BB                BBC     #DSA$V_Binary, -          ; If bit not set, branch                [29]
                                      1A       03C2                  L^DSA$GL_Flags, 60$     ;                                       [29]
                    00000000'EF      D6  03C3    858         incl    L^ds$gl_buflen        ; Increment length of final string      [09]
                    000001D4'EF      D7  03C9    859         decl    L^qbuff+4             ; Decrment address                      [09]
                50  00000000'EF      9E  03CF    860         movab   L^ds$gb_typecode, r0  ; Get address of type code              [09]
                000001D4'FF          60  90  03D6    861         movb    (r0), aL^qbuff+4      ; Set first byte of buffer = type code  [09]
                                       03DD    862
                50  0000FE68'EF      7E  03DD    863   60$:  MOVAQ   L^DSA$GQ_MSGPTR,R0    ; POINT TO TEST DESCRIPTOR
                80  00000000'EF      3C  03E4    864         MOVZWL  L^DS$GL_BUFLEN,(R0)+  ; STRING LENGTH TO MAILBOX               [07]
                51  000001D4'EF      DE  03EB    865         moval   L^Qbuff+4, r1         ; Get address of Qbuff                  [09]
                          60   61   D0  03F2    866         movl    (r1), (r0)            ; Buffer adr to mailbox                 [09]
                                       03F5    867
                                       03F5    868   ;
                                       03F5    869   ; DEBUG ***
                                       03F5    870   ;
                   4E 00000429'EF      E9  03F5    871         blbc    L^Debug_the_sucker, 68$ ; Skip debug stuff if flag clear      [99]
                                       03FC    872         Br_If_Not_Binary        68$       ; Only do debug if BINARY is set        [10]
                0000FE00'EF      01   E1  03FC                BBC     #DSA$V_Binary, -          ; If bit not set, branch                [29]
                                      46       0403                  L^DSA$GL_Flags, 68$     ;                                       [29]
                                      1F   BB  0404    873         pushr   #^M<r0,r1,r2,r3,r4>   ; Make debug transparent                [99]
                                54    6E   9E  0406    874         movab   (sp), r4              ; Save current sp                       [99]
                                      12   11  0409    875         brb     62$                   ; Skip format string                    [99]
                                       040B    876
5A 21 3D 70 74 5B 00000413'010E0000'   040B    877   61$:  .ascid  "[tp=!ZB]!/"          ; Format string (with <CR><LF>)         [10]
                      2F 21 5D 42      0419
                                       041D    878
                                5E   0A   C2  041D    879   62$:  subl2   #10, sp                ; Allocate space on stack               [99]
                                   6E   9F  0420    880         pushab  (sp)                  ; Set it's address                      [99]
                                7E   0A   9A  0422    881         movzbl  #10, -(sp)            ; And it's length                       [99]
                                53   6E   9E  0425    882         movab   (sp), r3              ; And remember it                       [99]
                             7E    00   B1  9A  0428    883         movzbl  @(r1), -(sp)          ; Final argument for FAO                [99]
                                   00   B1  94  042C    884         clrb    @(r1)                 ; Clear it out to avoid tty problems    [99]
                                      63   9F  042F    885         pushab  (r3)                  ; Where to put length                   [99]
                                      63   9F  0431    886         pushab  (r3)                  ; Address of result buffer              [99]
                                   D5   AF   9F  0433    887         pushab  61$                   ; Address of format string              [99]
                          00000000'GF      04   FB  0436    888         calls   #4, G^sys$fao         ; FAO it                                [99]
                                7E   63   7D  043D    889         movq    (r3), -(sp)           ; Push address of result string         [99]
                             FC02    CF   02   FB  0440    890         calls   #2, dsx$type_out      ; Print it                              [99]
                                5E   64   9E  0445    891         movab   (r4), sp              ; Restore initial stack                 [99]
                                      1F   BA  0448    892         popr    #^M<r0,r1,r2,r3,r4>   ; Restore registers                     [99]
```

```
                                  044A
                                  044A  68$:                                          ;                                          [99]
                                  044A
                                  044A  ;
                                  044A  ; DEBUG ***
                                  044A  ;
                      61    DD    044A          pushl   (r1)                          ; Push address of buffer                   [09]
      7E   00000000'EF    3C    044C          movzwl  L^ds$gl_buflen, -(sp)         ; And length of same                       [09]
          FBEF CF   02    FB    0453          calls   #2, dsx$type_out              ; Do the actual output                     [09]
                                  0458
      0000FE00'EF   8E    C8    0458  90$:    BISL2   (SP)+,L^DSA$GL_FLAGS          ; RESTORE STATE OF SUPPRESS FLAG
                                  045F
                                  045F  PRINT_X:
           00000000'EF    D5    045F          TSTL    MP$GR_BOOTED_PROCESSORS       ; Are we an mp system?                     [29]
                      06    12    0465          BNEQ    100$                          ; Branch if mp                             [29]
           00000000'EF    16    0467          jsb     L^kb_check                    ; Check for Control-C                      [08]
                                  046D  100$:
                            04    046D          RET
                                  046E
                                  046E          .DSABL  LSB
                                  046E
                                  046E  ;+
                                  046E  ; Cancel current input QIO operation and reset ^C handler
                                  046E  ;-
                                  046E
                                  046E  RESET_INPUT:
                                  046E          $CANCEL_S CHAN=L^DS$GW_TTIN           ; Cancel I/O on input channel              [07]
      7E   00000000'EF    3C    046E                  MOVZWL  L^DS$GW_TTIN,-(SP)
      00000000'GF   01    FB    0475                  CALLS   #1,G^SYS$CANCEL
                                  047C          $QIOW_S CHAN=L^DS$GW_TTIN, -          ; Reset ^C handler                         [07]
                                  047C                  FUNC=#IO$_SETMODE!IO$M_CTRLCAST, -
                                  047C                  P1=RCTRLC
                      7E    7C    047C                  CLRQ    -(SP)
                      7E    7C    047E                  CLRQ    -(SP)
                      00    DD    0480                  PUSHL   #0
           00000000'EF    DF    0482                  PUSHAL  RCTRLC
                      7E    7C    0488                  CLRQ    -(SP)
                      00    DD    048A                  PUSHL   #0
          7E   0000'8F    3C    048C                  MOVZWL  #IO$_SETMODE!IO$M_CTRLCAST,-(SP)
      7E   00000000'EF    3C    0491                  MOVZWL  L^DS$GW_TTIN,-(SP)
                      00    DD    0498                  PUSHL   #0
      00000000'GF   0C    FB    049A                  CALLS   #12,G^SYS$QIOW
                            05    04A1          RSB                                   ; Return
                                  04A2
```

```
                        .SBTTL  DSX$Printrev Routine
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine alerts the user to a difference between the revision
                ;       level of the hardware, firmware, patch level  and the revision level
                ;       the diagnostic was written for.  The operator has the choice as to
                ;       whether all, some, or none of the error messages will be displayed.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       $DS_PRINTREV_X
                ;               CALLS   #7,a#DSX$PRINTREV
                ;
                ; INPUT PARAMETERS:
                ;
                ;       LOG_UNIT(AP)      = UNIT NUMBER OF DEVICE BEING TESTED
                ;       ACTUAL_REV(AP)    = REVISION LEVEL OF UNIT BEING TESTED
                ;       EXPECTED_REV(AP)= CURRENT DIAGNOSTIC'S REVISION LEVEL
                ;       REV_TYPE(AP)      = HARDWARE, FIRMWARE, OR PATCH LEVEL
                ;       PRINTMASK(AP)     = INHIBITS PRINTING OF SOME OR ALL MESSAGES
                ;       PRLINK(AP)        = ADDRESS OF USER'S ERROR PRINT ROUTINE
                ;       MODULENUM(AP)     = MODULE NUMBER BEING TESTED
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       NONE
                ;
                ; DATA STRUCTURES:
                ;
                ;       DEVREV$T :  +-------------------------+
                ;                   | address of device name  |
                ;                   +-------------------------+
                ;             +4    | Module number 1         |
                ;                   +-------------------------+
                ;             +8    | Hardware Rev (actual)   |
                ;                   +-------------------------+
                ;            +12    | Firmware Rev (actual)   |
                ;                   +-------------------------+
                ;            +16    | Patch Revision (actual) |
                ;                   +-------------------------+
                ;            +20    | Module number 2         |
                ;                   +-------------------------+
                ;                             ...                 ==> Up to 4 Module Numbers
                ;                             ...
                ;                   +-------------------------+
                ;            +68    | address of device name  |
                ;                   +-------------------------+
                ;            +72    | Module number 1         |
                ;                   +-------------------------+
                ;                             ...                 ==> Up to 4 Devices (logical units)
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
```

```
                                04A2    983 ;       Modifications to DEVREV$T Data Structure
                                04A2    984 ;
                                04A2    985 ; COMPLETION CODES:
                                04A2    986 ;
                                04A2    987 ;       ds$_overflow == Exceeding maximum number of devices, or modules in DEVREV$T
                                04A2    988 ;       ds$_illunit  == Logical unit does not exist in P-table
                                04A2    989 ;       ds$_badtype  == Revision type is invalid
                                04A2    990 ;       ds$_error    == Hardware revision is out of bounds
                                04A2    991 ;       ds$_normal   == normal return status
                                04A2    992 ;
                                04A2    993 ;
                                04A2    994 ; SIDE EFFECTS:
                                04A2    995 ;
                                04A2    996 ;       NONE
                                04A2    997 ;
                                04A2    998 ;--
                                04A2    999
                         01FC   04A2   1000 .ENTRY  DSX$PRINTREV, ^M<R2,R3,R4,R5,R6,R7,R8>
                                04A4   1001
00000044 8F    04 AC    C5      04A4   1002        MULL3   printrev$_log_unit(AP),#68,offset$t     ; Get offset for specific un
         000001AC'EF             04AC
000001CC'EF    04 AC    D0      04B1   1003        MOVL    printrev$_log_unit(AP),lognum          ; Hold on to logunit number
   52    00000000'EF    DE      04B9   1004        MOVAL   DEVREV$T,R2                            ; TO ADJUST ADDRESS
   52    000001AC'EF    C0      04C0   1005        ADDL2   offset$t,R2                            ; Increment to specific unit
                 62     D5      04C7   1006        TSTL    (R2)
                 12     12      04C9   1007        BNEQ    1$                                     ; Log unit already seen
         000001A4'EF    D4      04CB   1008        CLRL    HARDEXIST$T                            ; clear flags
         000001A0'EF    D4      04D1   1009        CLRL    FIRMEXIST$T
         000001A8'EF    D4      04D7   1010        CLRL    PATCHEXIST$T
                                04DD   1011
                                04DD   1012 1$:     $DS_GPHARD_S    printrev$_log_unit(AP),a#p_table;Get device's p-table
         000001B0'9F    DF      04DD                        PUSHAL  a#p_table
               04 AC    DD      04E3                        PUSHL   printrev$_log_unit(AP)
      00000000'9F       02      04E6                        CALLS   #2, a#DS$GPHARD
      56   000001B0'EF  D0      04ED   1013        MOVL    p_table,R6                             ; Get address of hp-table
                 56     D5      04F4   1014        TSTL    R6                                     ; If no device go to error
                 5F     13      04F6   1015        BEQL    8$
         000001BC'EF    7C      04F8   1016        CLRQ    DEVNAME                                ; clear name
      000001BC'EF       66      04FE   1017        MOVB    HP$Q_DEVICE(R6),DEVNAME                ; Length of device name
            57    04 A6  D0      0505   1018        MOVL    4(R6),R7                               ; Offset to device name
                 04     BB      0509   1019        PUSHR   #^M<R2>                                ; LOST AFTER MOVC
000001BD'EF      67     66      28     050B   1020 MOVC3   HP$Q_DEVICE(R6),(R7),DEVNAME+1        ; Put char. in devname
                 04     BA      0513   1021        POPR    #^M<R2>                                ; RESTORED
      56   000001BC'EF  DE      0515   1022        MOVAL   DEVNAME,R6                             ; Store Adr. in reg
            62    56     D0      051C   1023        MOVL    R6,(R2)                                ; and in data structure
                                051F   1024
            52    04     C0      051F   1025        ADDL2   #4,R2                                  ; Increment Pointer
            51    04     D0      0522   1026        MOVL    #4,R1                                  ; Counter for TNAMEs
                                0525   1027 4$:
                 62     D5      0525   1028        TSTL    (R2)                                   ; See if TNAME there
                 0B     12      0527   1029        BNEQ    5$
                                0529   1030
            6C    07     D1      0529   1031        CMPL    #7,(AP)                                ; Branch if module # is
                 14     12      052C   1032        BNEQ    6$                                     ; not on the stack.
         62    1C AC    D0      052E   1033        MOVL    printrev$_MODULENUM(AP),(R2)
                 0E     11      0532   1034        BRB     6$                                     ; Skip error return
                                0534   1035
```

```
            6C   07   D1  0534  1036  5$:    CMPL    #7,(AP)                                  ; Branch if module # is
                 09   12  0537  1037         BNEQ    6$                                       ; not on the stack.
         50   1C AC   D0  0539  1038         MOVL    printrev$_MODULENUM(AP),R0                ; See if already have a T na
            62   50   D1  053D  1039         CMPL    R0,(R2)                                  ; if equal tname move to har
                 05   12  0540  1040         BNEQ    7$
                          0542  1041
            52   04   C0  0542  1042  6$:    ADDL2   #4,R2
                 1A   11  0545  1043         BRB     10$
                          0547  1044
            52   10   C0  0547  1045  7$:    ADDL2   #16,R2                                   ; Check next TNAME
              D8 51   F5  054A  1046         SOBGTR  R1,4$
      50   00660008 8F   D0  054D  1047         MOVL    #DS$_OVERFLOW,R0                         ; DEVREV$T data struture ful
              03C6   31  0554  1048         BRW     100$                                     ;
                          0557  1049
      50   00660100 8F   D0  0557  1050  8$:    MOVL    #DS$_ILLUNIT,R0                          ; Error return
              03BC   31  055E  1051         BRW     100$                                     ; Return
                          0561  1052
         10 AC   01   D1  0561  1053 10$:    CMPL    #prv$m_hwrev,printrev$_rev_type(AP)
                 22   13  0565  1054         BEQL    14$                                      ;
            52   04   C0  0567  1055         ADDL    #4,R2                                    ; update pointer
         10 AC   02   D1  056A  1056         CMPL    #prv$m_fwrev,printrev$_rev_type(AP)
                 03   12  056E  1057         BNEQ    12$                                      ; Continue
              00EA   31  0570  1058         BRW     25$                                      ; Firmware
            52   04   C0  0573  1059 12$:    ADDL    #4,R2
         10 AC   04   D1  0576  1060         CMPL    #prv$m_prev,printrev$_rev_type(AP)        ;[26]
                 03   12  057A  1061         BNEQ    13$                                      ; Continue
              00F7   31  057C  1062         BRW     30$                                      ; Patch level
      50   006600E8 8F   D0  057F  1063 13$:    MOVL    #DS$_BADTYPE,R0                          ; not hard,firmware, or patc
              0394   31  0586  1064         BRW     100$                                     ; Return
                          0589  1065 ;-----------------------------------------------------------------------------
                          0589  1066 ; Hardware rev type
                          0589  1067 ;
   08 AC   000002BE 8F   D1  0589  1068 14$:    CMPL    #702,PRINTREV$_ACTUAL_REV(AP)            ; Is rev. > 702    [34]
                 27   19  0591  1069         BLSS    15$                                      ; yes, then error  [34]
   0C AC   000002BE 8F   D1  0593  1070         CMPL    #702,PRINTREV$_EXPECTED_REV(AP)          ; Is rev. > 702    [34]
                 1D   19  059B  1071         BLSS    15$                                      ; yes, then error  [34]
                          059D  1072
      00000238'EF   02   90  059D  1073         MOVB    #2,ACT_REV                               ; Set up a counted [31]
      00000239'EF   20   90  05A4  1074         MOVB    #^X20,ACT_REV+1                          ; ascic string for [31]
      0000023A'EF   20   90  05AB  1075         MOVB    #^X20,ACT_REV+2                          ; output           [31]
              08 AC   D5  05B2  1076         TSTL    PRINTREV$_ACTUAL_REV(AP)                 ; Test if rev = 0  [31]
                 0D   12  05B5  1077         BNEQ    16$                                      ; if not 0 branch  [31]
              0087   31  05B7  1078         BRW     20$                                      ; print blanks     [31]
                          05BA  1079
      50   00660002 8F   D0  05BA  1080 15$:    MOVL    #DS$_ERROR,R0                            ; Hardware revision[34]
              0359   31  05C1  1081         BRW     100$                                     ; out of bounds    [34]
                          05C4  1082
         03 14 AC   04   E0  05C4  1083 16$:    BBS     #PRMSK$V_TRANSL, -
                          05C9  1084                 Printrev$_PRINTMASK(AP),17$              ; Trans to code    [31]
              004E   31  05C9  1085         BRW     19$                                      ; Trans # to ascic [31]
   00000228'EF   08 AC   D0  05CC  1086 17$:    MOVL    PRINTREV$_ACTUAL_REV(AP),Q_DIVD          ; Move to quadword [31]
      00000228'EF   1B   7B  05D4  1087         EDIV    #27,Q_DIVD,-                             ; Use Extended Div.[31]
00000234'EF   00000230'EF   05DB  1088                 L_QUO, L_REM                             ; to find quotent  [31]
         00000230'EF   D5  05E5  1089         TSTL    L_QUO                                    ; and remainder.   [31]
                 1D   13  05EB  1090         BEQL    18$                                      ; Branch if 1 - 26 [31]
   00000230'EF   40 8F   81  05ED  1091         ADDB3   #^x40,L_QUO,ACT_REV+1                    ; Compute Ax - Zx  [31]
         00000239'EF       05F5
```

```
00000234'EF   41 8F   81  05FA  1092          ADDB3   #^x41,L_REM,ACT_REV+2                   ; Compute xA - xZ  [31]
          0000023A'EF       0602
                0037   31  0607  1093          BRW     20$                                     ; Continue         [31]
00000234'EF   40 8F   81  060A  1094 18$:      ADDB3   #^x40,L_REM,ACT_REV+1                   ; Compute A - Z    [31]
          00000239'EF       0612
                0027   31  0617  1095          BRW     20$
                          061A  1096 19$:
                          061A  1097          $FAO_S  CTRSTR=CONTROL,-                        ; Convert from     [31]
                          061A  1098                  OUTBUF=FAO_DESC,-                       ; longword to      [31]
                          061A  1099                  OUTLEN=FAO_LEN,-                        ; ASCII            [31]
                          061A  1100                  P1=PRINTREV$_ACTUAL_REV(AP)             ;                  [31]
                   08 AC  DD  061A                    PUSHL   PRINTREV$_ACTUAL_REV(AP)
          00000262'EF   7F  061D                    PUSHAQ  FAO_DESC
          0000026A'EF   3F  0623                    PUSHAW  FAO_LEN
          00000276'EF   7F  0629                    PUSHAQ  CONTROL
    00000000'GF   04    FB  062F                    CALLS   #$$T2,G^SYS$FAO
00000238'EF   0000026A'EF   90  0636  1101          MOVB    FAO_LEN,ACT_REV                 ; And form ascic   [31]
                          0641  1102
000001B4'EF   000003C1'EF   DE  0641  1103 20$:     MOVAL   hardmessa,warecode              ; Rev_type message
        000001A4'EF     01  90  064C  1104          MOVB    #1,hardexist$t                  ; flag set rev type moved in
                          0653  1105
        62    00000238'EF   D0  0653  1106 23$:     MOVL    ACT_REV,(R2)                    ; Put # in data structure
                0002F   31  065A  1107          BRW     40$
                          065D  1108
                          065D  1109 ;-----------------------------------------------------------------------------
                          065D  1110 ; Firmware rev type
                          065D  1111 ;
000001B4'EF   000003CA'EF   DE  065D  1112 25$:     MOVAL   firmmessa,warecode              ; Rev_type message
        000001A0'EF     01  90  0668  1113          MOVB    #1,firmexist$t                  ; flag set rev type moved in
                          066F  1114
          62    08 AC   D0  066F  1115 27$:     MOVL    PRINTREV$_ACTUAL_REV(AP),(R2)   ; Put # in data structure
                0016   31  0673  1116          BRW     40$
                          0676  1117
                          0676  1118 ;-----------------------------------------------------------------------------
                          0676  1119 ; Patch rev type
                          0676  1120 ;
000001B4'EF   000003D3'EF   DE  0676  1121 30$:     MOVAL   PATCHMESSA,warecode             ; Rev_type message
        000001A8'EF     01  90  0681  1122          MOVB    #1,patchexist$t                 ; flag set rev type moved in
                          0688  1123
          62    08 AC   D0  0688  1124 35$:     MOVL    PRINTREV$_ACTUAL_REV(AP),(R2)   ; Put # in data structure
                          068C  1125
        000001B8'EF     D4  068C  1126 40$:     CLRL    relation                        ; Delete previous difference
     03 14 AC     03    E0  0692  1127          BBS     #PRMSK$V_ALWAYS,printrev$_printmask(AP),41$ ;Print plain revision
                00B4   31  0697  1128          BRW     50$                             ; If bit is set
                          069A  1129
          10 AC   01    D1  069A  1130 41$:     CMPL    #prv$m_hwrev,printrev$_rev_type(AP)
                  57    13  069E  1131          BEQL    45$                             ;
          57    08 AC   D0  06A0  1132          MOVL    printrev$_actual_rev(AP),R7
                          06A4  1133          $PRINT  #DS$K_TYPE_COMMAND_OUT,-        ; Typecode
                          06A4  1134                  #DS$K_PrintF,-                  ; ...use printf
                          06A4  1135                  fmt_beg_err,-                   ; ...message format
                          06A4  1136                  warecode,-                      ; ...firm or hardware
                          06A4  1137                  R6                              ; ...device name
                    56    DD  06A4                    PUSHL   R6
          000001B4'EF     DD  06A6                    PUSHL   warecode
          00000281'EF     9F  06AC                    PUSHAB  fmt_beg_err
              7E    01    9A  06B2                    movzbl  #DS$K_PrintF, -(sp)
```

```
                7E   16   98  06B5                        cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
      000002D3'GF    05   FB  06B8                        CALLS   #$$N+2, G^DSX$PRINT
                6C   07   D1  06BF  1138          CMPL    #7,(AP)                         ; Branch if module # is
                     1B   12  06C2  1139          BNEQ    43$                             ; not on the stack.
                  1C AC   D5  06C4  1140          TSTL    printrev$_modulenum(ap)
                     16   13  06C7  1141          BEQL    43$
                          06C9  1142          $PRINT  #DS$K_TYPE_COMMAND_OUT,-
                          06C9  1143                  #DS$K_PrintF,-
                          06C9  1144                  FMT_MODULE,-
                          06C9  1145                  PRINTREV$_MODULENUM(AP)
                  1C AC   DD  06C9                    PUSHL   PRINTREV$_MODULENUM(AP)
            0000029D'EF   9F  06CC                    PUSHAB  FMT_MODULE
                7E   01   9A  06D2                    movzbl  #DS$K_PrintF, -(sp)
                7E   16   98  06D5                    cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
      000002D3'GF    04   FB  06D8                    CALLS   #$$N+2, G^DSX$PRINT
                          06DF  1146 43$:     $PRINT  #DS$K_TYPE_COMMAND_OUT,-
                          06DF  1147                  #DS$K_PrintF,-
                          06DF  1148                  fmt_p_f_end,-
                          06DF  1149                  R7                              ; ...unit revision level
                     57   DD  06DF                    PUSHL   R7
            000003AF'EF   9F  06E1                    PUSHAB  fmt_p_f_end
                7E   01   9A  06E7                    movzbl  #DS$K_PrintF, -(sp)
                7E   16   98  06EA                    cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
      000002D3'GF    04   FB  06ED                    CALLS   #$$N+2, G^DSX$PRINT
                   0057   31  06F4  1150          BRW     50$
                          06F7  1151 45$:
      57    00000238'EF   DE  06F7  1152          MOVAL   ACT_REV,R7                      ; type ascic string
                          06FE  1153          $PRINT  #DS$K_TYPE_COMMAND_OUT,-        ; Typecode
                          06FE  1154                  #DS$K_PrintF,-                  ; ...use printf
                          06FE  1155                  fmt_beg_err,-                   ; ...message format
                          06FE  1156                  warecode,-                      ; ...firm or hardware
                          06FE  1157                  R6                              ; ...device name
                     56   DD  06FE                    PUSHL   R6
            000001B4'EF   DD  0700                    PUSHL   warecode
            00000281'EF   9F  0706                    PUSHAB  fmt_beg_err
                7E   01   9A  070C                    movzbl  #DS$K_PrintF, -(sp)
                7E   16   98  070F                    cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
      000002D3'GF    05   FB  0712                    CALLS   #$$N+2, G^DSX$PRINT
                6C   07   D1  0719  1158          CMPL    #7,(AP)                         ; Branch if module # is
                     1B   12  071C  1159          BNEQ    47$                             ; not on the stack.
                  1C AC   D5  071E  1160          TSTL    printrev$_modulenum(ap)
                     16   13  0721  1161          BEQL    47$
                          0723  1162          $PRINT  #DS$K_TYPE_COMMAND_OUT,-
                          0723  1163                  #DS$K_PrintF,-
                          0723  1164                  fmt_module,-
                          0723  1165                  PRINTREV$_MODULENUM(AP)
                  1C AC   DD  0723                    PUSHL   PRINTREV$_MODULENUM(AP)
            0000029D'EF   9F  0726                    PUSHAB  fmt_module
                7E   01   9A  072C                    movzbl  #DS$K_PrintF, -(sp)
                7E   16   98  072F                    cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
      000002D3'GF    04   FB  0732                    CALLS   #$$N+2, G^DSX$PRINT
                          0739  1166 47$:     $PRINT  #DS$K_TYPE_COMMAND_OUT,-
                          0739  1167                  #DS$K_PrintF,-
                          0739  1168                  fmt_h_end,-
                          0739  1169                  R7                              ; ...unit revision level
                     57   DD  0739                    PUSHL   R7
            000003B8'EF   9F  073B                    PUSHAB  fmt_h_end
```

```
                 7E   01   9A  0741                    movzbl  #DS$K_PrintF, -(sp)
                 7E   16   98  0744                    cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
        000002D3'GF   04   FB  0747                    CALLS   #$$N+2, G^DSX$PRINT
                               074E
          0C AC   08 AC   D1  074E  50$:    CMPL    printrev$_actual_rev(AP),printrev$_expected_rev(AP); Unit and diagno
                     17   13  0753          BEQL    60$                                  ; Equal revision numbers
                     2A   1A  0755          BGTRU   70$                                  ; Actual > expected
                               0757
                               0757  ;-----------------------------------------------------
                               0757  ;       Actual rev. is less than Expected rev.
                               0757
          03 14 AC   00   E1  0757          BBC     #prmsk$v_lss,printrev$_printmask(AP),58$ ; if no inhibit, branch
                   01B7   31  075C          BRW     95$                                  ; else, normal return
                               075F
 000001B8'EF   000003DC'EF   DE  075F  58$:    MOVAL   lessthan,relation                    ; 'less than' message
                     28   11  076A          BRB     80$                                  ; branch to print
                               076C  ;-----------------------------------------------------
                               076C  ;       Actual rev. is equal to Expected rev.
                               076C
          03 14 AC   01   E1  076C  60$:    BBC     #prmsk$v_eql,printrev$_printmask(AP),68$; if no inhibit, branch
                   01A2   31  0771          BRW     95$                                  ; normal return
                               0774
 000001B8'EF   000003F3'EF   DE  0774  68$:    MOVAL   equalmess,relation                   ; 'equal to' message
                     13   11  077F          BRB     80$                                  ; branch to print
                               0781
                               0781  ;-----------------------------------------------------
                               0781  ;       Actual rev. is greater than Expected rev.
                               0781
          03 14 AC   02   E1  0781  70$:    BBC     #prmsk$v_gtr,printrev$_printmask(AP),78$ ; if no inhibit, branch
                   018D   31  0786          BRW     95$                                  ; Normal return
                               0789
 000001B8'EF   000003E6'EF   DE  0789  78$:    MOVAL   greatthan,relation                   ; 'greater than' message
                               0794                                                       ; continue to print..
                               0794
                               0794  80$:
             10 AC   01   D1  0794          CMPL    #prv$m_hwrev,printrev$_rev_type(AP)
                     03   13  0798          BEQL    81$                                  ; Find rev_type  [31]
                   0109   31  079A          BRW     90$                                  ; branch to print[31]
        0000024D'EF   02   90  079D  81$:    MOVB    #2,EXP_REV                           ; Set up a counted [31]
        0000024E'EF   20   90  07A4          MOVB    #^X20,EXP_REV+1                      ; ascic string for [31]
        0000024F'EF   20   90  07AB          MOVB    #^X20,EXP_REV+2                      ; output           [31]
                 0C AC   D5  07B2          TSTL    PRINTREV$_EXPECTED_REV(AP)           ; Test if rev = 0  [31]
                     03   12  07B5          BNEQ    82$                                  ; if not 0 branch  [31]
                   007D   31  07B7          BRW     87$                                  ; print blanks     [31]
          03 14 AC   04   E0  07BA  82$:    BBS     #PRMSK$V_TRANSL, -
                               07BF                  printrev$_PRINTMASK(AP),83$          ; Trans to code    [31]
                   004E   31  07BF          BRW     86$                                  ; Trans # to ascic [31]
      00000228'EF   0C AC   D0  07C2  83$:    MOVL    PRINTREV$_EXPECTED_REV(AP),Q_DIVD    ; Move to quadword [31]
        00000228'EF   1B   7B  07CA          EDIV    #27,Q_DIVD,-                         ; Use Extended Div.[31]
 00000234'EF   00000230'EF      07D1                  L_QUO, L_REM                         ; to find quotent  [31]
              00000230'EF   D5  07DB          TSTL    L_QUO                                ; and remainder.   [31]
                     1D   13  07E1          BEQL    85$                                  ; Branch if 1 - 26 [31]
      00000230'EF   40 8F   81  07E3          ADDB3   #^x40,L_QUO,EXP_REV+1                ; Compute Ax - Zx  [31]
              0000024E'EF      07EB
      00000234'EF   41 8F   81  07F0          ADDB3   #^x41,L_REM,EXP_REV+2                ; Compute xA - xZ  [31]
              0000024F'EF      07F8
                   0037   31  07FD          BRW     87$                                  ; Continue         [31]
```

```
00000234'EF   40 8F   81   0800  1222 85$:    ADDB3   #^x40,L_REM,EXP_REV+1                    ; Compute A - Z    [31]
          0000024E'EF        0808
                0027   31   080D  1223         BRW     87$
                            0810  1224 86$:
                            0810  1225         $FAO_S  CTRSTR=CONTROL,-                        ; Convert from     [31]
                            0810  1226                 OUTBUF=FAO_DESC_2,-                     ; longword to      [31]
                            0810  1227                 OUTLEN=FAO_LEN_2,-                      ; ASCII            [31]
                            0810  1228                 P1=PRINTREV$_EXPECTED_REV(AP)           ;                  [31]
                   0C AC    DD   0810                 PUSHL   PRINTREV$_EXPECTED_REV(AP)
          0000026C'EF       7F   0813                 PUSHAQ  FAO_DESC_2
          00000274'EF       3F   0819                 PUSHAW  FAO_LEN_2
          00000276'EF       7F   081F                 PUSHAQ  CONTROL
     00000000'GF    04      FB   0825                 CALLS   #$$T2,G^SYS$FAO
0000024D'EF   00000274'EF   90   082C  1229         MOVB    FAO_LEN_2,EXP_REV                   ; And form ascic   [31]
                            0837  1230
          57  00000238'EF   DE   0837  1231 87$:    MOVAL   ACT_REV,R7                          ;                  [31]
          58  0000024D'EF   DE   083E  1232         MOVAL   EXP_REV,R8                          ;                  [31]
                            0845  1233         $PRINT  #DS$K_TYPE_COMMAND_OUT,-                ; Typecode
                            0845  1234                 #DS$K_PrintF,-                          ; ...use printf
                            0845  1235                 fmt_beg_err,-                           ; ...message format
                            0845  1236                 warecode,-                              ; ...firm or hardware
                            0845  1237                 R6                                      ; ...device name
                      56    DD   0845                 PUSHL   R6
          000001B4'EF       DD   0847                 PUSHL   warecode
          00000281'EF       9F   084D                 PUSHAB  fmt_beg_err
                7E    01    9A   0853                 movzbl  #DS$K_PrintF, -(sp)
                7E    16    98   0856                 cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
     000002D3'GF    05      FB   0859                 CALLS   #$$N+2, G^DSX$PRINT
                            0860  1238
          6C    07    D1   0860  1239         CMPL    #7,(AP)                                 ; Branch if module # is
                1B    12   0863  1240         BNEQ    88$                                     ; not on the stack.
             1C AC    D5   0865  1241         TSTL    printrev$_modulenum(ap)
                16    13   0868  1242         BEQL    88$
                            086A  1243         $PRINT  #DS$K_TYPE_COMMAND_OUT,-                ; Typecode
                            086A  1244                 #DS$K_PrintF,-                          ; ...use printf
                            086A  1245                 fmt_module,-
                            086A  1246                 PRINTREV$_MODULENUM(AP)                 ;
                   1C AC    DD   086A                 PUSHL   PRINTREV$_MODULENUM(AP)
          0000029D'EF       9F   086D                 PUSHAB  fmt_module
                7E    01    9A   0873                 movzbl  #DS$K_PrintF, -(sp)
                7E    16    98   0876                 cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
     000002D3'GF    04      FB   0879                 CALLS   #$$N+2, G^DSX$PRINT
                            0880  1247
                            0880  1248 88$:    $PRINT  #DS$K_TYPE_COMMAND_OUT,-                ; Typecode
                            0880  1249                 #DS$K_PrintF,-                          ; ...use printf
                            0880  1250                 fmt_h_end_err,-
                            0880  1251                 relation,-                              ; ...relation message
                            0880  1252                 warecode,-                              ; ...second waretype message
                            0880  1253                 R7,-                                    ; ...unit revision level
                            0880  1254                 R8                                      ; ...diagnostic rev. level
                      58    DD   0880                 PUSHL   R8
                      57    DD   0882                 PUSHL   R7
          000001B4'EF       DD   0884                 PUSHL   warecode
          000001B8'EF       DD   088A                 PUSHL   relation
          0000032A'EF       9F   0890                 PUSHAB  fmt_h_end_err
                7E    01    9A   0896                 movzbl  #DS$K_PrintF, -(sp)
                7E    16    98   0899                 cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
```

```
000002D3'GF   07   FB  089C                          CALLS   #$$N+2, G^DSX$PRINT
                       08A3
            0060   31  08A3                  BRW     93$                                  ; continue
                       08A6          90$:
                       08A6                  $PRINT  #DS$K_TYPE_COMMAND_OUT,-             ; Typecode
                       08A6                          #DS$K_PrintF,-                       ; ...use printf
                       08A6                          fmt_beg_err,-                        ; ...message format
                       08A6                          warecode,-                           ; ...firm or hardware
                       08A6                          R6                                   ; ...device name
              56   DD  08A6                          PUSHL   R6
     000001B4'EF   DD  08A8                          PUSHL   warecode
     00000281'EF   9F  08AE                          PUSHAB  fmt_beg_err
          7E   01  9A  08B4                          movzbl  #DS$K_PrintF, -(sp)
          7E   16  98  08B7                          cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
000002D3'GF   05   FB  08BA                          CALLS   #$$N+2, G^DSX$PRINT
                       08C1
          6C   07  D1  08C1                  CMPL    #7,(AP)                              ; Branch if module # is
              1B   12  08C4                  BNEQ    91$                                  ; not on the stack.
          1C AC    D5  08C6                  TSTL    printrev$_modulenum(ap)
              16   13  08C9                  BEQL    91$
                       08CB                  $PRINT  #DS$K_TYPE_COMMAND_OUT,-             ; Typecode
                       08CB                          #DS$K_PrintF,-
                       08CB                          FMT_MODULE,-                                 ; ...use printf
                       08CB                          PRINTREV$_MODULENUM(AP)
          1C AC    DD  08CB                          PUSHL   PRINTREV$_MODULENUM(AP)
     0000029D'EF   9F  08CE                          PUSHAB  FMT_MODULE
          7E   01  9A  08D4                          movzbl  #DS$K_PrintF, -(sp)
          7E   16  98  08D7                          cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
000002D3'GF   04   FB  08DA                          CALLS   #$$N+2, G^DSX$PRINT
                       08E1
                       08E1          91$:    $PRINT  #DS$K_TYPE_COMMAND_OUT,-             ; Typecode
                       08E1                          #DS$K_PrintF,-                       ; ...use printf
                       08E1                          fmt_end_err,-
                       08E1                          relation,-                           ; ...relation message
                       08E1                          warecode,-                           ; ...second waretype message
                       08E1                          printrev$_actual_rev(AP),-           ; ...unit revision level
                       08E1                          PRINTREV$_EXPECTED_REV(AP)           ; ...diagnostic rev. level
          0C AC    DD  08E1                          PUSHL   PRINTREV$_EXPECTED_REV(AP)
          08 AC    DD  08E4                          PUSHL   printrev$_actual_rev(AP)
     000001B4'EF   DD  08E7                          PUSHL   warecode
     000001B8'EF   DD  08ED                          PUSHL   relation
     000002A5'EF   9F  08F3                          PUSHAB  fmt_end_err
          7E   01  9A  08F9                          movzbl  #DS$K_PrintF, -(sp)
          7E   16  98  08FC                          cvtbl   #DS$K_TYPE_COMMAND_OUT, -(sp)
000002D3'GF   07   FB  08FF                          CALLS   #$$N+2, G^DSX$PRINT
                       0906
          18 AC    D5  0906          93$:    TSTL    printrev$_prlink(AP)                 ; Error message to report
              0B   13  0909                  BEQL    95$                                  ; No message
              09   90  090B                  MOVB    #ds$k_type_error_body,-              ; Set type code to error bod
     00000000'EF       090D                           L^ds$gb_typecode
                       0912
      18 BC   6C   FA  0912                  CALLG   (AP),@printrev$_prlink(ap)           ; Call user's print routine
                       0916
50   00660001 8F   D0  0916          95$:    MOVL    #DS$_NORMAL,R0                       ; No error return
                   04  091D          100$:   ret
                       091E
```

```
                        1292          .Subtitle       DSX$CvtReg Routine - Mnemonic register conversion routine
                        1293 ;**
                        1294 ; FUNCTIONAL DESCRIPTION:
                        1295 ;
                        1296 ;       ASSOCIATES REGISTER BITS AND/OR FIELDS WITH ASCII MNEMONICS
                        1297 ;       GIVEN AN ASCII CONTROL STRING AND THE REGISTER CONTENTS
                        1298 ;       Instead of the mnemonic address a code may specified directing
                        1299 ;       CVTREG to use a predefined mnemonic string.  This allows
                        1300 ;       transportable diagnostics to format generic register values,
                        1301 ;       i.e. UNIBUS adapter MAP registers.
                        1302 ;
                        1303 ; CALLING SEQUENCE:
                        1304 ;
                        1305 ;       CALLG   ARGLIST,a#DS$CVTREG
                        1306 ;       CALLS   #11,a#DS$CVTREG
                        1307 ;
                        1308 ; INPUT PARAMETERS:
                        1309 ;
                        1310 ;        4(AP)  = STARTING BIT POSITION (MSB)
                        1311 ;        8(AP)  = REGISTER CONTENTS
                        1312 ;       12(AP)  = COUNTED ASCII BIT MNEMONIC STRING ADDRESS OR CODE
                        1313 ;       16(AP)  = OUTPUT BUFFER ADDRESS
                        1314 ;       20(AP)  = OUTPUT BUFFER SIZE
                        1315 ;       24(AP)  = OPTIONAL ARGUMENTS FOR MNEMONIC STRING DIRECTIVES (UP TO 6)
                        1316 ;
                        1317 ; IMPLICIT INPUTS:
                        1318 ;
                        1319 ;       NONE
                        1320 ;
                        1321 ; OUTPUT PARAMETERS:
                        1322 ;
                        1323 ;       NONE
                        1324 ;
                        1325 ; IMPLICIT OUTPUTS:
                        1326 ;
                        1327 ;       NONE
                        1328 ;
                        1329 ; COMPLETION CODES:
                        1330 ;
                        1331 ;       DS$_NORMAL
                        1332 ;       DS$_PROGERR
                        1333 ;
                        1334 ; SIDE EFFECTS:
                        1335 ;
                        1336 ;       R1 = LENGTH OF OUTPUT STRING + COUNT (NOT TRUNCATED)
                        1337 ;
                        1338 ; REGISTER USAGE:
                        1339 ;
                        1340 ;        R0  CURRENT BIT POSITION
                        1341 ;        R1  CURRENT OUTPUT BUFFER LENGTH
                        1342 ;        R2  CURRENT OUTPUT BUFFER POINTER
                        1343 ;        R3  CURRENT MNEMONIC STRING POINTER
                        1344 ;        R4  MNEMONIC STRING TERMINATOR (LAST BYTE ADDRESS + 1)
                        1345 ;        R5  FIELD SIZE
                        1346 ;        R6  EMUL OVERFLOW, EDIV QUOTIENT AND GENERAL USAGE
                        1347 ;        R7  EDIV DIVIDEND (HIGH ORDER)
                        1348 ;        R8  RADIX FOR EDIV
```

```
                        091E  1349 ;        R9  LOOP CONTROL
                        091E  1350 ;       R10  LOOP CONTROL
                        091E  1351 ;       R11  OPTIONAL ARGUMENT POINTER
                        091E  1352 ;--
```

```
                     0FFC  091E  1354 .ENTRY DSX$CVTREG,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                           0920  1355
                6C   0B   D1  0920  1356         CMPL    #11,(AP)                ; CHECK ARGLIST VALIDITY
                     03   13  0923  1357         BEQL    10$                     ; OK, PROCEED
                   0199   31  0925  1358         BRW     330$                    ; SOMETHING WRONG !
                           0928  1359
              20   04 AC   D1  0928  1360 10$:    CMPL    4(AP),#32               ; CHECK MSB POSITION
                     03   1F  092C  1361         BLSSU   30$                     ; OK, PROCEED
                           092E  1362
                   0190   31  092E  1363 20$:    BRW     330$                    ; SOMETHING WRONG !
                           0931  1364                                             ;
              50   04 AC   D0  0931  1365 30$:    MOVL    4(AP),R0                ; GET STARTING BIT POSITION
                   51   01   D0  0935  1366         MOVL    #1,R1                   ; SET UP OUTPUT LENGTH (COUNT BYTE)
           52   01   10 AC   C1  0938  1367         ADDL3   16(AP),#1,R2            ; GET OUTPUT BUFFER ADDRESS
                5B   18 AC   DE  093D  1368         MOVAL   24(AP),R11              ; Get address of optional args
                53   0C AC   D0  0941  1369         MOVL    12(AP),R3               ; GET MNEMONIC STRING ADDRESS
                     1B   18  0945  1370         BGEQ    40$                     ; Branch if string address
           53   FFFFFFFC 8F   D1  0947  1371         CMPL    #CVT$_MAX,R3            ; Last assigned code
                     DE   14  094E  1372         BGTR    20$                     ; Unassigned code in R3
                53   53   CE  0950  1373         MNEGL   R3,R3                   ; Negate it making it positive
        53   FFFFFFFC'EF43   D0  0953  1374         MOVL    CVT$A_CODELIST-4[R3],R3 ; Address of list, use one origin
             5B   04 A3   DE  095B  1375         MOVAL   4(R3),R11               ; Address of AUX arglist
                53   63   D0  095F  1376         MOVL    (R3),R3                 ; Point to ASCIC mnemonic string
                           0962  1377
                54   83   9A  0962  1378 40$:    MOVZBL  (R3)+,R4                ; GET MNEMONIC STRING LENGTH
                54   53   C0  0965  1379         ADDL2   R3,R4                   ; GET MNEMONIC STRING TERMINATOR
                           0968  1380                                             ;
                     06   BB  0968  1381 50$:    PUSHR   #^M<R1,R2>              ; CURRENT OUTPUT LENGTH & ADR
                01   51   D1  096A  1382         CMPL    R1,#1                   ; LOOK AT OUTPUT LENGTH
                     0F   13  096D  1383         BEQL    70$                     ; SKIP IF EMPTY
             14 AC   51   D1  096F  1384         CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                     04   19  0973  1385         BLSS    60$                     ; NO, PROCEED
           52   10 AC   D0  0975  1386         MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                           0979  1387
                82   2C   90  0979  1388 60$:    MOVB    #^A',',(R2)+            ; STICK IN A SEPARATOR
                     51   D6  097C  1389         INCL    R1                      ; BUMP OUTPUT LENGTH
                           097E  1390                                             ;
                54   53   D1  097E  1391 70$:    CMPL    R3,R4                   ; MNEMONIC STRING EXHAUSTED ?
                     46   1E  0981  1392         BGEQU   120$                    ; YES, GO CHECK LAST BIT
                2C   63   91  0983  1393         CMPB    (R3),#^A','             ; END OF THIS MNEMONIC ?
                     41   13  0986  1394         BEQL    120$                    ; YES, GO CHECK THE BIT
                           0988  1395                                             ;
             14 AC   51   D1  0988  1396         CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                     04   19  098C  1397         BLSS    80$                     ; NO, PROCEED
           52   10 AC   D0  098E  1398         MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                           0992  1399
                82   63   90  0992  1400 80$:    MOVB    (R3),(R2)+              ; COPY CHAR TO OUTPUT BUFFER
                     51   D6  0995  1401         INCL    R1                      ; BUMP OUTPUT LENGTH
                3D   83   91  0997  1402         CMPB    (R3)+,#^A'='            ; FIELD DIRECTIVE ?
                     E2   12  099A  1403         BNEQ    70$                     ; NO, LOOP FOR NEXT CHARACTER
                     55   D4  099C  1404         CLRL    R5                      ; YES, INITIALIZE FIELD SIZE
                           099E  1405                                             ;
                54   53   D1  099E  1406 90$:    CMPL    R3,R4                   ; MNEMONIC STRING EXHAUSTED ?
                     03   1F  09A1  1407         BLSSU   100$                    ; NO, PROCEED
                   011B   31  09A3  1408         BRW     330$                    ; YES, BAD PLACE TO RUN OUT OF GAS
                           09A6  1409                                             ;
             5E 8F   63   91  09A6  1410 100$:   CMPB    (R3),#^A'^'             ; NUMERIC DIRECTIVE ?
```

```
                    2A   13   09AA  1411          BEQL    140$                    ; YES, GO EXTRACT A FIELD
                40 8F   63   91   09AC  1412          CMPB    (R3),#^A'@'             ; INDIRECT DIRECTIVE ?
                    24   13   09B0  1413          BEQL    140$                    ; YES, GO EXTRACT A FIELD
                         09B2  1414                                                  ;
                    56   83   9A   09B2  1415          MOVZBL  (R3)+,R6                ; GET NEXT CHARACTER
                    56   30   C2   09B5  1416          SUBL2   #^A'0',R6               ; CONVERT TO BINARY NUMERIC
                    0C   19   09B8  1417          BLSS    110$                    ; NO GOOD, WENT NEGATIVE
                    0A   56   D1   09BA  1418          CMPL    R6,#10                  ; DECIMAL DIGIT ?
                    07   18   09BD  1419          BGEQ    110$                    ; NO, BAD CHARACTER
            55  56  0A   55   7A   09BF  1420          EMUL    R5,#10,R6,R5            ; EXTEND AND ADD IN DIGIT
                    D8   11   09C4  1421          BRB     90$                     ; GO BACK FOR NEXT DIGIT
                         09C6  1422
                00F8   31   09C6  1423 110$:     BRW     330$                    ; SOMETHING WRONG !
                         09C9  1424
            05 08 AC    50   E0   09C9  1425 120$:     BBS     R0,8(AP),130$           ; ASSOCIATED BIT SET ?
                    06   BA   09CE  1426          POPR    #^M<R1,R2>              ; NO, REMOVE CURRENT MNEMONIC
                00B6   31   09D0  1427          BRW     290$                    ; AND GO CLEAN UP
                         09D3  1428
                00B0   31   09D3  1429 130$:     BRW     280$                    ; ELSE PRESERVE OUTPUT AS IS
                         09D6  1430
                    50   D6   09D6  1431 140$:     INCL    R0                      ; ADJUST FOR POSITIONING
                50  55   C2   09D8  1432          SUBL2   R5,R0                   ; POSITION TO RIGHTMOST BIT
        56  08 AC  55   50   EF   09DB  1433          EXTZV   R0,R5,8(AP),R6          ; GET THE FIELD FROM THE REGISTER
                40 8F   63   91   09E1  1434          CMPB    (R3),#^A'@'             ; WAS IT AN INDIRECT DIRECTIVE ?
                    21   12   09E5  1435          BNEQ    170$                    ; NO, PROCEED WITH DIRECT CONVERSION
                    53   D6   09E7  1436          INCL    R3                      ; SKIP OVER DIRECTIVE DELIMITER
                57  8B   D0   09E9  1437          MOVL    (R11)+,R7               ; ADDRESS OF STRING LIST
        57  04 A746   D0   09EC  1438          MOVL    4(R7)[R6],R7            ; ADDRESS OF SELECTED STRING
            59  87   9A   09F1  1439          MOVZBL  (R7)+,R9                ; LENGTH OF SELECTED STRING
                         09F4  1440                                                  ;
          14 AC  51   D1   09F4  1441 150$:     CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                    04   19   09F8  1442          BLSS    160$                    ; NO, PROCEED
          52  10 AC  D0   09FA  1443          MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                         09FE  1444
                82  87   90   09FE  1445 160$:     MOVB    (R7)+,(R2)+             ; COPY CHAR TO OUTPUT BUFFER
                    51   D6   0A01  1446          INCL    R1                      ; BUMP OUTPUT LENGTH
                  EE 59   F5   0A03  1447          SOBGTR  R9,150$                 ; LOOP TILL DONE
                    74   11   0A06  1448          BRB     270$                    ; THEN GO CLEAN UP
                         0A08  1449                                                  ;
                    53   D6   0A08  1450 170$:     INCL    R3                      ; SKIP OVER DIRECTIVE DELIMITER
            58 8F   63   91   0A0A  1451          CMPB    (R3),#^A'X'             ; HEX CONVERSION ?
                    0C   12   0A0E  1452          BNEQ    180$                    ; NO, TRY DECIMAL
                58  10   D0   0A10  1453          MOVL    #16,R8                  ; SET UP RADIX
            59  55  01   C3   0A13  1454          SUBL3   #1,R5,R9                ; ADJUST FIELD SIZE
                59  04   C6   0A17  1455          DIVL2   #4,R9                   ; CONVERT TO HEX DIGIT COUNT -1
                    27   11   0A1A  1456          BRB     210$                    ; AND SKIP
                         0A1C  1457
            44 8F   63   91   0A1C  1458 180$:     CMPB    (R3),#^A'D'             ; DECIMAL CONVERSION ?
                    0C   12   0A20  1459          BNEQ    190$                    ; NO, TRY OCTAL
                58  0A   D0   0A22  1460          MOVL    #10,R8                  ; SET UP RADIX
            59  03  55   C5   0A25  1461          MULL3   R5,#3,R9                ; ADJUST FIELD SIZE
                59  0A   C6   0A29  1462          DIVL2   #10,R9                  ; CONVERT TO DECIMAL DIGIT COUNT -1
                    15   11   0A2C  1463          BRB     210$                    ; AND SKIP
                         0A2E  1464
            4F 8F   63   91   0A2E  1465 190$:     CMPB    (R3),#^A'O'             ; OCTAL CONVERSION ?
                    0C   12   0A32  1466          BNEQ    200$                    ; NO, GIVE UP
                58  08   D0   0A34  1467          MOVL    #8,R8                   ; SET UP RADIX
```

```
                59   55   01   C3   0A37  1468          SUBL3   #1,R5,R9                ; ADJUST FIELD SIZE
                     59   03   C6   0A3B  1469          DIVL2   #3,R9                   ; CONVERT TO OCTAL DIGIT COUNT -1
                          03   11   0A3E  1470          BRB     210$                    ; AND SKIP
                                    0A40  1471
                        007E   31   0A40  1472 200$:     BRW     330$                    ; SOMETHING WRONG !
                                    0A43  1473                                          ;
                          7E   29   90   0A43  1474 210$:     MOVB    #^A')',-(SP)            ; FOR CLARIFICATION,
                          7E   83   90   0A46  1475          MOVB    (R3)+,-(SP)             ; RADIX DIRECTIVE IS
                          7E   28   90   0A49  1476          MOVB    #^A'(',-(SP)            ; APPENDED TO NUMERIC DISPLAY
                5A   03   59   C1   0A4C  1477          ADDL3   R9,#3,R10               ; COPY ADJUSTED LOOP COUNT
                                    0A50  1478                                          ;
                          57   D4   0A50  1479 220$:     CLRL    R7                      ; DON'T NEED HIGH ORDER
             57   56   56   58   7B   0A52  1480          EDIV    R8,R6,R6,R7             ; NEXT DIGIT TO R7
                          0A   57   D1   0A57  1481          CMPL    R7,#10                  ; ALPHA DIGIT (I.E., HEX) ?
                               05   19   0A5A  1482          BLSS    230$                    ; NO, PROCEED
                          57   37   C0   0A5C  1483          ADDL2   #^A'A'-10,R7            ; YES, CONVERT TO 'HEX' ASCII
                               03   11   0A5F  1484          BRB     240$                    ; AND SKIP
                                    0A61  1485
                          57   30   C0   0A61  1486 230$:     ADDL2   #^A'0',R7               ; CONVERT TO ASCII DIGIT
                                    0A64  1487
                          7E   57   90   0A64  1488 240$:     MOVB    R7,-(SP)                ; STACK ASCII DIGIT
                            E6 59   F4   0A67  1489          SOBGEQ  R9,220$                 ; LOOP TILL DONE
                                    0A6A  1490                                          ;
                       14   AC   51   D1   0A6A  1491 250$:     CMPL    R1,20(AP)               ; OUTPUT BUFFER FULL ?
                                    04   19   0A6E  1492          BLSS    260$                    ; NO, PROCEED
                       52     10   AC   D0   0A70  1493          MOVL    16(AP),R2               ; YES, USE FIRST BYTE AS SCRATCH
                                    0A74  1494
                          82   8E   90   0A74  1495 260$:     MOVB    (SP)+,(R2)+             ; STORE THE NEXT CHARACTER
                               51   D6   0A77  1496          INCL    R1                      ; BUMP OUTPUT LENGTH
                            EE 5A   F4   0A79  1497          SOBGEQ  R10,250$                ; LOOP TILL DONE
                                    0A7C  1498                                          ;
                          54   53   D1   0A7C  1499 270$:     CMPL    R3,R4                   ; MNEMONICS EXHAUSTED ?
                               16   1E   0A7F  1500          BGEQU   300$                    ; YES, JUST SKIP OUT QUIETLY
                          2C   63   91   0A81  1501          CMPB    (R3),#^A','             ; LOOKING AT A COMMA ?
                               3B   12   0A84  1502          BNEQ    330$                    ; NO, THEN SOMETHING'S WRONG
                                    0A86  1503
                          5E   08   C0   0A86  1504 280$:     ADDL2   #8,SP                   ; PRESERVE OUTPUT AS IS
                                    0A89  1505
                               53   D6   0A89  1506 290$:     INCL    R3                      ; SKIP COMMA IN MNEMONICS
                               50   D7   0A8B  1507          DECL    R0                      ; SHIFT TO NEXT BIT POSITION
                               08   19   0A8D  1508          BLSS    300$                    ; DONE, GO FINISH UP
                          54   53   D1   0A8F  1509          CMPL    R3,R4                   ; PERHAPS MNEMONICS EXHAUSTED
                               03   1E   0A92  1510          BGEQU   300$                    ; IT IS, JUST MEANS DONE
                             FED1   31   0A94  1511          BRW     50$                     ; ELSE GO BACK TO DO SOME MORE
                                    0A97  1512
                00000100 8F   51   D1   0A97  1513 300$:     CMPL    R1,#1a8                 ; COUNTED FORMAT OVERFLOW ?
                               08   14   0A9E  1514          BGTR    310$                    ; YES, GO FIX IT
                       14   AC   51   D1   0AA0  1515          CMPL    R1,20(AP)               ; CALLER'S BUFFER OVERFLOW ?
                               02   14   0AA4  1516          BGTR    310$                    ; YES, GO FIX IT
                               25   11   0AA6  1517          BRB     350$                    ; BOTH OK, DO NORMAL THING
                                    0AA8  1518                                          ;
             00000100 8F   14   AC   D1   0AA8  1519 310$:     CMPL    20(AP),#1a8             ; FIND THE SMALLER ONE
                               07   19   0AB0  1520          BLSS    320$                    ; CALLER'S BUFFER IS .LSS. 256
                   10   BC   FF   8F   90   0AB2  1521          MOVB    #<1a8>-1,a16(AP)        ; STORE MAX COUNT, BUFFER IS BIG ENOUGH
                               0B   11   0AB7  1522          BRB     340$                    ; AND SKIP OUT
                                    0AB9  1523
             10   BC   14   AC   01   83   0AB9  1524 320$:     SUBB3   #1,20(AP),a16(AP)       ; COPY CALLER'S MAX LENGTH
```

```
                      03   11   0ABF  1525          BRB     340$                    ; AND SKIP OUT
                                0AC1  1526                                          ;
                   10 BC   94   0AC1  1527 330$:     CLRB    @16(AP)                 ; INDICATE NO OUTPUT
                                0AC4  1528                                          ;
          50    00660020 8F   D0   0AC4  1529 340$:     MOVL    #DS$_PROGERR,R0         ; INDICATE A PROBLEM
                      0C   11   0ACB  1530          BRB     RCVTREGX                ; AND SKIP OUT
                                0ACD  1531                                          ;
           10 BC    51   01   83   0ACD  1532 350$:     SUBB3   #1,R1,@16(AP)           ; STORE STRING LENGTH
          50    00660001 8F   D0   0AD2  1533          MOVL    #DS$_NORMAL,R0          ; INDICATE SUCCESS
                                0AD9  1534
                                0AD9  1535 RCVTREGX:                                ;
                      04   0AD9  1536          RET                                   ; RETURN TO CALLER
                                0ADA  1537
                                0ADA  1538          .END
```

```
$$.TMP1                         = 00000001    D
$$.TMP2                         = 00000060
$$ARGS                          = 00000007    D
$$N                             = 00000005    D
$$T1                            = 00000001    D
$$T2                            = 00000004    D
ACT_REV                           00000238 R  D  02
BIT...                          = 00000005    D
CLI$K_BUFSIZ                    = 00000100    D
CLI$K_SIZE                        0000044C    D
CLI$L_ADDRESS                     00000018    D
CLI$L_COMMAND                     00000004    D
CLI$L_DATA                        0000001C    D
CLI$L_FLAGS                       00000000    D
CLI$L_FLAGS2                      00000444    D
CLI$L_LAST                        00000024    D
CLI$L_NEXT                        00000030    D
CLI$L_PASS                        0000002C    D
CLI$L_SUBT                        00000028    D
CLI$L_TEMP_BASE                   00000448    D
CLI$L_TEST                        00000020    D
CLI$M_START_OR_RUN              = 00000008    D
CLI$Q_BUFQWD                      00000034    D
CLI$Q_FILE                        00000008    D
CLI$Q_SECTION                     00000010    D
CLI$Q_TIME                        0000043C    D
CLI$T_BUFFER                      0000003C    D
CLI$V_ADAPTER                   = 00000018    D
CLI$V_ADR                       = 0000000B    D
CLI$V_ASCII                     = 00000013    D
CLI$V_BASE_QUAL                 = 00000002    D
CLI$V_BREAK                     = 0000000A    D
CLI$V_BRIEF                     = 0000001B    D
CLI$V_BYTE                      = 0000000D    D
CLI$V_CLEAR                     = 00000002    D
CLI$V_DEC                       = 00000010    D
CLI$V_DEFAULT                   = 0000000C    D
CLI$V_DEPOSIT                   = 00000019    D
CLI$V_EVENT                     = 00000008    D
CLI$V_EXAM                      = 00000005    D
CLI$V_FLAGS                     = 00000009    D
CLI$V_HEX                       = 00000012    D
CLI$V_KERNEL                    = 00000017    D
CLI$V_LOAD                      = 00000006    D
CLI$V_LONG                      = 0000000F    D
CLI$V_NEXT                      = 00000001    D
CLI$V_NOALLOC                   = 00000000    D
CLI$V_NOTNUF                    = 00000001    D
CLI$V_OCT                       = 00000011    D
CLI$V_PREG                      = 0000001A    D
CLI$V_QA                        = 00000007    D
CLI$V_QACKLOOPLOOPS             = 0000001C    D
CLI$V_QAERRORPRINTS             = 0000001B    D
CLI$V_QAMULTIPLEPASS            = 0000001F    D
CLI$V_QASUBTESTLOOPS            = 0000001E    D
CLI$V_QATESTLOOPS               = 0000001D    D
CLI$V_REG                       = 00000014    D
CLI$V_REQUIRED                  = 00000000    D
CLI$V_RUN                       = 00000015    D
CLI$V_SET                       = 00000003    D
CLI$V_SHOW                      = 00000004    D
CLI$V_START_OR_RUN              = 00000003    D
CLI$V_VALSEC                    = 00000016    D
CLI$V_WORD                      = 0000000E    D
CONTROL                           00000276 R  D  02
CR                              = 0000000D    D
CRD$K_CRD_INITIALIZATION        = 00000000  G D
CRD$K_DS_CONTROL_C              = 00000001  G D
CRD$K_DS_FLAGS                  = 00000000  G D
CRD$K_FAILURE                   = 00000000    D
CRD$K_HOOK_POINT                = 00000000  G D
CRD$K_LAST_ACCESS_CODE          = 00000002  G D
CRD$K_LAST_DS_VARIABLE          = 00000002  G D
CRD$K_LAST_FUNCTION             = 00000002  G D
CRD$K_LAST_HOOK_POINT           = 00000001  G D
CRD$K_READ                      = 00000000  G D
CRD$K_SUCCESS                   = 00000001    D
CRD$K_TYPECODE                  = 00000001  G D
CRD$K_WRITE                     = 00000001  G D
CVT$A_CODELIST                    00000000 R  D  03
CVT$T_MBAMAP                      ********    X  03
CVT$T_MBASR                       ********    X  03
CVT$T_UBADPR                      ********    X  03
CVT$T_UBAMAP                      ********    X  03
CVT$_MAX                        = FFFFFFFC    D
CVT$_MBAMAP                     = FFFFFFFC    D
CVT$_MBASR                      = FFFFFFFD    D
CVT$_UBADPR                     = FFFFFFFF    D
CVT$_UBAMAP                     = FFFFFFFE    D
DEBUG_THE_SUCKER                  00000429 RG D  02
DEVNAME                           000001BC R  D  02
DEVREV$T                          00000000 RG D  02
DEVREV$T_X                        000001D0 RG D  02
DS$GA_CRD_DS_INTERFACE            ********    X  04
DS$GB_QIOACT                      ********    X  04
DS$GB_TYPECODE                    ********    X  04
DS$GB_VDS_DEBUG                   00000429 RG D  02
DS$GB_WIDTH                       ********    X  04
DS$GL_BUFLEN                      ********    X  04
DS$GL_FLAGS                       ********    X  04
DS$GL_PAGE                        000001DD RG D  02
DS$GPHARD                         ********    X  04
DS$GT_BUFFER                      ********    X  02
DS$GW_TTIN                        ********    X  04
DS$GW_TTOUT                       ********    X  04
DS$K_BUFSIZ                       ********    X  02
DS$K_ERROR                      = 00000002    D
DS$K_NORMAL                     = 00000001    D
DS$K_PRINTB                     = 00000002    D
DS$K_PRINTF                     = 00000001    D
DS$K_PRINTI                     = 00000000    D
DS$K_PRINTX                     = 00000003    D
DS$K_SEVERE                     = 00000004    D
DS$K_SUBSYS                     = 00000066    D
```

```
DS$K_TYPE_ABORT_PROGRAM         = 00000014   D
DS$K_TYPE_ABORT_TEST            = 00000013   D
DS$K_TYPE_COMMAND_ERR           = 00000015   D
DS$K_TYPE_COMMAND_OUT           = 00000016   D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A   D
DS$K_TYPE_DS_PROMPT             = 00000001   D
DS$K_TYPE_DS_START              = 0000001D   D
DS$K_TYPE_ERRDEV                = 00000008   D
DS$K_TYPE_ERRHARD               = 00000006   D
DS$K_TYPE_ERROR_BODY            = 00000009   D
DS$K_TYPE_ERROR_END             = 0000000A   D
DS$K_TYPE_ERRPREP               = 0000001B   D
DS$K_TYPE_ERRSOFT               = 00000007   D
DS$K_TYPE_ERRSUP                = 00000004   D
DS$K_TYPE_ERRSYS                = 00000005   D
DS$K_TYPE_ERR_HALT              = 0000000D   D
DS$K_TYPE_EXCEPTION             = 0000000C   D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B   D
DS$K_TYPE_FIRST_PASS            = 00000011   D
DS$K_TYPE_GENERAL               = 00000000   D
DS$K_TYPE_GENERAL_ERROR         = 00000003   D
DS$K_TYPE_NO_TESTS              = 00000012   D
DS$K_TYPE_PARAM_ERROR           = 0000001C   D
DS$K_TYPE_PROGRAM_END           = 00000010   D
DS$K_TYPE_PROGRAM_INFO          = 00000017   D
DS$K_TYPE_PROGRAM_START         = 0000000F   D
DS$K_TYPE_QIO_INVADP            = 00000024   D
DS$K_TYPE_QIO_NODRIVER          = 00000022   D
DS$K_TYPE_QIO_WRONGVER          = 00000023   D
DS$K_TYPE_SCRIPT_ECHO           = 00000021   D
DS$K_TYPE_SCRIPT_PNF            = 0000001E   D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020   D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F   D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019   D
DS$K_TYPE_START_ERR             = 00000018   D
DS$K_TYPE_START_LIST            = 00000025   D
DS$K_TYPE_SUMMARY               = 0000000E   D
DS$K_TYPE_USER_PROMPT           = 00000002   D
DS$K_WARNING                    = 00000000   D
DS$M_ABRTFLG                    = 00000040   D
DS$M_BADTIME                    = 00100000   D
DS$M_BATCH                      = 00400000   D
DS$M_BRKCLR                     = 00001000   D
DS$M_BRKPT                      = 00000800   D
DS$M_CHARFLG                    = 00000100   D
DS$M_CMDFLG                     = 00000080   D
DS$M_CTRLC                      = 00000001   D
DS$M_CTRLO                      = 00010000   D
DS$M_DEVFLG                     = 00000200   D
DS$M_DISABLCC                   = 01000000   D
DS$M_DONFLG                     = 00002000   D
DS$M_ERRFLG                     = 00000010   D
DS$M_EXCEPT                     = 00080000   D
DS$M_EXETST                     = 00040000   D
DS$M_HLTFLG                     = 00000008   D
DS$M_LODFLG                     = 00000002   D
DS$M_MEMMGT                     = 00008000   D
DS$M_NI                         = 04000000   D
DS$M_OUTPUT                     = 00800000   D
DS$M_RUBFLG                     = 00000020   D
DS$M_SCRIPT                     = 00200000   D
DS$M_SETIMR                     = 02000000   D
DS$M_STRFLG                     = 00000004   D
DS$M_SUBT                       = 00004000   D
DS$M_SYSFLG                     = 00000400   D
DS$M_TIMED                      = 02000000   D
DS$M_TIMED_OUT                  = 08000000   D
DS$M_TIMRON                     = 00020000   D
DS$RAB_OUTPUT                     ********   X   04
DS$V_ABRTFLG                    = 00000006   D
DS$V_BADTIME                    = 00000014   D
DS$V_BATCH                      = 00000016   D
DS$V_BRKCLR                     = 0000000C   D
DS$V_BRKPT                      = 0000000B   D
DS$V_CHARFLG                    = 00000008   D
DS$V_CMDFLG                     = 00000007   D
DS$V_CTRLC                      = 00000000   D
DS$V_CTRLO                      = 00000010   D
DS$V_DEVFLG                     = 00000009   D
DS$V_DISABLCC                   = 00000018   D
DS$V_DONFLG                     = 0000000D   D
DS$V_ERRFLG                     = 00000004   D
DS$V_EXCEPT                     = 00000013   D
DS$V_EXETST                     = 00000012   D
DS$V_HLTFLG                     = 00000003   D
DS$V_LODFLG                     = 00000001   D
DS$V_MEMMGT                     = 0000000F   D
DS$V_NI                         = 0000001A   D
DS$V_OUTPUT                     = 00000017   D
DS$V_RUBFLG                     = 00000005   D
DS$V_SCRIPT                     = 00000015   D
DS$V_SETIMR                     = 00000019   D
DS$V_STRFLG                     = 00000002   D
DS$V_SUBT                       = 0000000E   D
DS$V_SYSFLG                     = 0000000A   D
DS$V_TIMED                      = 00000019   D
DS$V_TIMED_OUT                  = 0000001B   D
DS$V_TIMRON                     = 00000011   D
DS$_AP_NORMAL_BREAK             = 00660150   D
DS$_ARITH                       = 006600D0   D
DS$_ASBE                        = 00660118   D
DS$_BADLINK                     = 006600F0   D
DS$_BADTYPE                     = 006600E8   D
DS$_BIIC                        = 00660120   D
DS$_CHME                        = 006600A8   D
DS$_CHMK                        = 006600E0   D
DS$_DEVNAME                     = 00660108   D
DS$_ERROR                       = 00660002   D
DS$_FHWE                        = 00660068   D
DS$_FRAGBUF                     = 00660080   D
DS$_ICBUSY                      = 006600C8   D
DS$_ICERR                       = 006600C0   D
DS$_IHWE                        = 00660060   D
DS$_ILLCHAR                     = 00660018   D
```

```
DS$_ILLPAGCNT                   = 00660078     D
DS$_ILLUNIT                     = 00660100     D
DS$_INIT_FAIL                   = 00660140     D
DS$_INSFMEM                     = 00660050     D
DS$_INVCPU                      = 00660130     D
DS$_IPL2HI                      = 006600B8     D
DS$_IVADDR                      = 00660040     D
DS$_IVVECT                      = 00660038     D
DS$_KRNLSTK                     = 00660090     D
DS$_LOGIC                       = 00660070     D
DS$_MCHK                        = 00660088     D
DS$_MEM_ALLOC_ERR               = 00660138     D
DS$_MMOFF                       = 00660058     D
DS$_NEEDUNIT                    = 006600F8     D
DS$_NODE                        = 00660128     D
DS$_NOPCS                       = 00660110     D
DS$_NORMAL                      = 00660001     D
DS$_NOSUPPORT                   = 006600B1     D
DS$_NOTALLOWMP                  = 00660148     D
DS$_NOTDON                      = 00660030     D
DS$_NOTIMP                      = 006600B0     D
DS$_NULLSTR                     = 00660010     D
DS$_OVERFLOW                    = 00660008     D
DS$_POWER                       = 00660098     D
DS$_PROGERR                     = 00660020     D
DS$_SEVERE                      = 00660004     D
DS$_TRANSL                      = 006600A0     D
DS$_TRUNCATE                    = 00660028     D
DS$_UNEXPINT                    = 006600D8     D
DS$_UNSUPPDEV                   = 00660158     D
DS$_VASFULL                     = 00660048     D
DS$_WARNING                     = 00660000     D
DSA$AL_APTMAIL                    0000FE00     D
DSA$AT_APTTXT                     0000FA00     D
DSA$GL_APTCOM                     0000FE04     D
DSA$GL_DEVLEN                     0000FE58     D
DSA$GL_ERRNO                      0000FE44     D
DSA$GL_EVENT                      0000FE48     D
DSA$GL_FLAGS                      0000FE00     D
DSA$GL_MSGTYP                     0000FE40     D
DSA$GL_PASSES                     0000FE08     D
DSA$GL_PASSNO                     0000FE54     D
DSA$GL_SECTNO                     0000FE10     D
DSA$GL_SID                        0000FE14     D
DSA$GL_SUBTNO                     0000FE4C     D
DSA$GL_TESTNO                     0000FE50     D
DSA$GL_UNITS                      0000FE0C     D
DSA$GQ_MSGPTR                     0000FE68     D
DSA$GT_DEVNAM                     0000FE5C     D
DSA$M_NORPT                     = 08000000     D
DSA$V_APT                       = 0000001F     D
DSA$V_BINARY                    = 00000001     D
DSA$V_CRD_AUTOTEST_OFF          = 0000000B     D
DSA$V_CRD_MENUTEST_OFF          = 00000010     D
DSA$V_CRD_MENUTEST_ON           = 00000012     D
DSA$V_IE1                       = 00000004     D
DSA$V_IE2                       = 00000005     D
DSA$V_IE3                       = 00000006     D
DSA$V_IES                       = 00000007     D
DSA$V_NORPT                     = 0000001B     D
DSA$V_USER                      = 0000001C     D
DSV$SETPAGE                       00000000 RG  D  04
DSV$SHOWPAGE                      00000027 RG  D  04
DSX$CVTREG                        0000091E RG  D  04
DSX$PRINT                         000002D3 RG  D  04
DSX$PRINTB                        0000033B RG  D  04
DSX$PRINTF                        00000353 RG  D  04
DSX$PRINTI                        00000368 RG  D  04
DSX$PRINTREV                      000004A2 RG  D  04
DSX$PRINTS                        00000321 RG  D  04
DSX$PRINTX                        0000032E RG  D  04
DSX$TYPE_OUT                      00000047 RG  D  04
EQUALMESS                         000003F3 R   D  02
EXP_REV                           0000024D R   D  02
FALSE                           = 00000000     D
FAO_DESC                          00000262 R   D  02
FAO_DESC_2                        0000026C R   D  02
FAO_LEN                           0000026A R   D  02
FAO_LEN_2                         00000274 R   D  02
FIRMEXIST$T                       000001A0 RG  D  02
FIRMMESSA                         000003CA R   D  02
FMT_BEG_ERR                       00000281 R   D  02
FMT_END_ERR                       000002A5 R   D  02
FMT_H_END                         000003B8 R   D  02
FMT_H_END_ERR                     0000032A R   D  02
FMT_MODULE                        0000029D R   D  02
FMT_P_F_END                       000003AF R   D  02
GREATTHAN                         000003E6 R   D  02
HARDEXIST$T                       000001A4 RG  D  02
HARDMESSA                         000003C1 R   D  02
HP$A_DEPENDENT                    00000032     D
HP$A_DEVICE                       00000018     D
HP$A_DVA                          0000001C     D
HP$A_LINK                         00000020     D
HP$B_DRIVE                        0000000B     D
HP$B_FLAGS                        0000000A     D
HP$Q_DEVICE                       00000000     D
HP$T_DEVICE                       0000000C     D
HP$T_TYPE                         00000026     D
HP$W_SIZE                         00000008     D
HP$W_VECTOR                       00000024     D
IO$M_CTRLCAST                     ********     X   04
IO$M_TRMNOECHO                    ********     X   04
IO$_READVBLK                      ********     X   04
IO$_SETMODE                       ********     X   04
IO$_WRITEVBLK                     ********     X   04
KB_CHECK                          ********     X   04
LESSTHAN                          000003DC R   D   02
LF                              = 0000000A     D
LINE_COUNT                        000001D9 RG  D   02
LOGNUM                            000001CC RG  D   02
L_QUO                             00000230 R   D   02
L_REM                             00000234 R   D   02
MP$GR_BOOTED_PROCESSORS           ********     X   04
```

```
OFF                          = 00000000    D
OFFSET$T                       000001AC R  D  02
ON                           = 00000001    D
PAGE_STRING                    000003FC R  D  02
PATCHEXIST$T                   000001A8 RG D  02
PATCHMESSA                     000003D3 R  D  02
PRINTREV$_ACTUAL_REV         = 00000008    D
PRINTREV$_EXPECTED_REV       = 0000000C    D
PRINTREV$_LOG_UNIT           = 00000004    D
PRINTREV$_MODULENUM          = 0000001C    D
PRINTREV$_NARGS              = 00000007    D
PRINTREV$_PRINTMASK          = 00000014    D
PRINTREV$_PRLINK             = 00000018    D
PRINTREV$_REV_TYPE           = 00000010    D
PRINT_X                        0000045F R  D  04
PRMSK$M_ALWAYS               = 00000008    D
PRMSK$M_EQL                  = 00000002    D
PRMSK$M_GTR                  = 00000004    D
PRMSK$M_LSS                  = 00000001    D
PRMSK$M_TRANSL               = 00000010    D
PRMSK$V_ALWAYS               = 00000003    D
PRMSK$V_EQL                  = 00000001    D
PRMSK$V_GTR                  = 00000002    D
PRMSK$V_LSS                  = 00000000    D
PRMSK$V_TRANSL               = 00000004    D
PRV$M_FWREV                  = 00000002    D
PRV$M_HWREV                  = 00000001    D
PRV$M_PREV                   = 00000004    D
PRV$V_FWREV                  = 00000001    D
PRV$V_HWREV                  = 00000000    D
PRV$V_PREV                   = 00000002    D
P_TABLE                        000001B0 R  D  02
QBUFF                          000001D0 R  D  02
Q_DIVD                         00000228 R  D  02
RAB$L_RBF                    = 00000028    D
RAB$W_RSZ                    = 00000022    D
RCTRLC                         ********   X   04
RCVTREGX                       00000AD9 R  D  04
RELATION                       000001B8 R  D  02
RESET_INPUT                    0000046E R  D  04
SIZ...                       = 00000001    D
SS$_NORMAL                   = 00000001    D
SYS$CANCEL                     ********  GX   04
SYS$FAO                        ********   X   04
SYS$FAOL                       ********   X   04
SYS$PUT                        ********  GX   04
SYS$QIOW                       ********  GX   04
TRUE                         = 00000001    D
T_PAGE                         000001E1 R  D  02
T_SHOW_PAGE                    00000204 R  D  02
WARECODE                       000001B4 R  D  02
WIDTH_TOO_LARGE                000001D8 R  D  02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	0000042A (1066.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000010 (16.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
CODE	00000ADA (2778.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$.TMP1	=00000001	528 (1)	528 (1)
$$.TMP2	=00000060	528 (1)	528 (1)
$$ARGS	=00000007	209 (1)	209 (1)
$$N	=00000005	1279 (1)	1137 (1) 1145 (1) 1149 (1) 1157 (1) 1165 (1) 1169 (1) 1237 (1) 1246 (1) 1254 (1) 1262 (1) 1271 (1) 1279 (1)
$$T1	=00000001	922 (1)	#-390 (1) 435 (1) 209 (1) 625 (1) 632 (1) 645 (1) 651 (1) 665 (1) 670 (1) 922 (1)
$$T2	=00000004	1228 (1)	1100 (1) 1228 (1)
ACT_REV	00000238-R	253 (1)	#-1073 (1) #-1074 (1) #-1075 (1) #-1091 (1) #-1092 (1) #-1094 (1) #-1101 (1) #-1106 (1) 1152 (1) 1231 (1) 259 (1)
BIT...	=00000005	209 (1)	199 (1) 202 (1) 203 (1) 205 (1) 206 (1) 209 (1)
CLI$K_BUFSIZ	=00000100	207 (1)	207 (1)
CLI$K_SIZE	0000044C	207 (1)	
CLI$L_ADDRESS	00000018	207 (1)	
CLI$L_COMMAND	00000004	207 (1)	
CLI$L_DATA	0000001C	207 (1)	#-379 (1)
CLI$L_FLAGS	00000000	207 (1)	
CLI$L_FLAGS2	00000444	207 (1)	
CLI$L_LAST	00000024	207 (1)	
CLI$L_NEXT	00000030	207 (1)	
CLI$L_PASS	0000002C	207 (1)	
CLI$L_SUBT	00000028	207 (1)	
CLI$L_TEMP_BASE	00000448	207 (1)	
CLI$L_TEST	00000020	207 (1)	
CLI$M_START_OR_RUN	=00000008	207 (1)	
CLI$Q_BUFQWD	00000034	207 (1)	
CLI$Q_FILE	00000008	207 (1)	
CLI$Q_SECTION	00000010	207 (1)	
CLI$Q_TIME	0000043C	207 (1)	
CLI$T_BUFFER	0000003C	207 (1)	
CLI$V_ADAPTER	=00000018	207 (1)	
CLI$V_ADR	=0000000B	207 (1)	
CLI$V_ASCII	=00000013	207 (1)	
CLI$V_BASE_QUAL	=00000002	207 (1)	
CLI$V_BREAK	=0000000A	207 (1)	
CLI$V_BRIEF	=0000001B	207 (1)	
CLI$V_BYTE	=0000000D	207 (1)	
CLI$V_CLEAR	=00000002	207 (1)	
CLI$V_DEC	=00000010	207 (1)	
CLI$V_DEFAULT	=0000000C	207 (1)	
CLI$V_DEPOSIT	=00000019	207 (1)	
CLI$V_EVENT	=00000008	207 (1)	
CLI$V_EXAM	=00000005	207 (1)	
CLI$V_FLAGS	=00000009	207 (1)	
CLI$V_HEX	=00000012	207 (1)	
CLI$V_KERNEL	=00000017	207 (1)	

```
CLI$V_LOAD                      =00000006     207     (1)
CLI$V_LONG                      =0000000F     207     (1)
CLI$V_NEXT                      =00000001     207     (1)
CLI$V_NOALLOC                   =00000000     207     (1)
CLI$V_NOTNUF                    =00000001     207     (1)
CLI$V_OCT                       =00000011     207     (1)
CLI$V_PREG                      =0000001A     207     (1)
CLI$V_QA                        =00000007     207     (1)
CLI$V_QACKLOOPLOOPS             =0000001C     207     (1)
CLI$V_QAERRORPRINTS             =0000001B     207     (1)
CLI$V_QAMULTIPLEPASS            =0000001F     207     (1)
CLI$V_QASUBTESTLOOPS            =0000001E     207     (1)
CLI$V_QATESTLOOPS               =0000001D     207     (1)
CLI$V_REG                       =00000014     207     (1)
CLI$V_REQUIRED                  =00000000     207     (1)
CLI$V_RUN                       =00000015     207     (1)
CLI$V_SET                       =00000003     207     (1)
CLI$V_SHOW                      =00000004     207     (1)
CLI$V_START_OR_RUN              =00000003     207     (1)       207     (1)
CLI$V_VALSEC                    =00000016     207     (1)
CLI$V_WORD                      =0000000E     207     (1)
CONTROL                          00000276-R   266     (1)       1100    (1)       1228    (1)
CR                              =0000000D     311     (1)       313     (1)
CRD$K_CRD_INITIALIZATION        =00000000     206     (1)
CRD$K_DS_CONTROL_C              =00000001     206     (1)
CRD$K_DS_FLAGS                  =00000000     206     (1)
CRD$K_FAILURE                   =00000000     206     (1)
CRD$K_HOOK_POINT                =00000000     206     (1)
CRD$K_LAST_ACCESS_CODE          =00000002     206     (1)
CRD$K_LAST_DS_VARIABLE          =00000002     206     (1)
CRD$K_LAST_FUNCTION             =00000002     206     (1)
CRD$K_LAST_HOOK_POINT           =00000001     206     (1)
CRD$K_READ                      =00000000     206     (1)
CRD$K_SUCCESS                   =00000001     206     (1)
CRD$K_TYPECODE                  =00000001     206     (1)     #-499     (1)
CRD$K_WRITE                     =00000001     206     (1)
CVT$A_CODELIST                   00000000-R   332     (1)     #-1374    (1)
CVT$T_MBAMAP                     00000000-XR                    336     (1)
CVT$T_MBASR                      00000000-XR                    335     (1)
CVT$T_UBADPR                     00000000-XR                    333     (1)
CVT$T_UBAMAP                     00000000-XR                    334     (1)
CVT$_MAX                        =FFFFFFFC     202     (1)     #-1371    (1)
CVT$_MBAMAP                     =FFFFFFFC     202     (1)       202     (1)
CVT$_MBASR                      =FFFFFFFD     202     (1)
CVT$_UBADPR                     =FFFFFFFF     202     (1)
CVT$_UBAMAP                     =FFFFFFFE     202     (1)
DEBUG_THE_SUCKER                 00000429-R   317     (1)     #-871     (1)
DEVNAME                          000001BC-R   227     (1)     #-1016    (1)     #-1017    (1)       1020    (1)       1022    (1)
DEVREV$T                         00000000-R   218     (1)       1004    (1)
DEVREV$T_X                       000001D0-R   230     (1)
DS$GA_CRD_DS_INTERFACE           00000000-XR                    500     (1)
DS$GB_QIOACT                     00000000-XR                    #-536     (1)     #-636     (1)     #-672     (1)
DS$GB_TYPECODE                   00000000-XR                    #-1284    (1)     #-487     (1)     #-498     (1)     #-730     (1)
                                                                #-752     (1)       860     (1)
DS$GB_VDS_DEBUG                  00000429-R   316     (1)
DS$GB_WIDTH                      00000000-XR                    #-564     (1)
DS$GL_BUFLEN                     00000000-XR                    849     (1)     #-858     (1)     #-864     (1)     #-900     (1)
```

```
DS$GL_FLAGS                     00000000-XR                    505    (1)     843    (1)
DS$GL_PAGE                      000001DD-R    241     (1)    #-382   (1)    #-430   (1)    #-557   (1)    #-560   (1)
                                                             #-597   (1)
DS$GPHARD                       00000000-XR                   1012    (1)
DS$GT_BUFFER                    00000000-XR                    235    (1)
DS$GW_TTIN                      00000000-XR                   #-645   (1)    #-651   (1)    #-919   (1)    #-922   (1)
DS$GW_TTOUT                     00000000-XR                   #-611   (1)    #-625   (1)    #-632   (1)    #-665   (1)
                                                             #-670   (1)
DS$K_BUFSIZ                     00000000-XR                    234    (1)     839    (1)
DS$K_ERROR                      =00000002     199     (1)
DS$K_NORMAL                     =00000001     199     (1)
DS$K_PRINTB                     =00000002     205     (1)
DS$K_PRINTF                     =00000001     205     (1)    #-1137   (1)    #-1145   (1)    #-1149   (1)    #-1157   (1)
                                                             #-1165   (1)    #-1169   (1)    #-1237   (1)    #-1246   (1)
                                                             #-1254   (1)    #-1262   (1)    #-1271   (1)    #-1279   (1)
                                                             #-390    (1)    #-435    (1)
DS$K_PRINTI                     =00000000     205     (1)
DS$K_PRINTX                     =00000003     205     (1)
DS$K_SEVERE                     =00000004     199     (1)
DS$K_SUBSYS                     =00000066     199     (1)     199    (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     205     (1)
DS$K_TYPE_ABORT_TEST            =00000013     205     (1)
DS$K_TYPE_COMMAND_ERR           =00000015     205     (1)    #-390    (1)
DS$K_TYPE_COMMAND_OUT           =00000016     205     (1)    #-1137   (1)    #-1145   (1)    #-1149   (1)    #-1157   (1)
                                                             #-1165   (1)    #-1169   (1)    #-1237   (1)    #-1246   (1)
                                                             #-1254   (1)    #-1262   (1)    #-1271   (1)    #-1279   (1)
                                                             #-435    (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     205     (1)    #-486    (1)
DS$K_TYPE_DS_PROMPT             =00000001     205     (1)
DS$K_TYPE_DS_START              =0000001D     205     (1)
DS$K_TYPE_ERRDEV                =00000008     205     (1)
DS$K_TYPE_ERRHARD               =00000006     205     (1)
DS$K_TYPE_ERROR_BODY            =00000009     205     (1)    #-1283   (1)
DS$K_TYPE_ERROR_END             =0000000A     205     (1)
DS$K_TYPE_ERRPREP               =0000001B     205     (1)
DS$K_TYPE_ERRSOFT               =00000007     205     (1)
DS$K_TYPE_ERRSUP                =00000004     205     (1)
DS$K_TYPE_ERRSYS                =00000005     205     (1)
DS$K_TYPE_ERR_HALT              =0000000D     205     (1)
DS$K_TYPE_EXCEPTION             =0000000C     205     (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     205     (1)
DS$K_TYPE_FIRST_PASS            =00000011     205     (1)
DS$K_TYPE_GENERAL               =00000000     205     (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     205     (1)
DS$K_TYPE_NO_TESTS              =00000012     205     (1)
DS$K_TYPE_PARAM_ERROR           =0000001C     205     (1)
DS$K_TYPE_PROGRAM_END           =00000010     205     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     205     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     205     (1)
DS$K_TYPE_QIO_INVADP            =00000024     205     (1)
DS$K_TYPE_QIO_NODRIVER          =00000022     205     (1)
DS$K_TYPE_QIO_WRONGVER          =00000023     205     (1)
DS$K_TYPE_SCRIPT_ECHO           =00000021     205     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     205     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     205     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     205     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     205     (1)
```

```
DS$K_TYPE_START_ERR             =00000018     205     (1)
DS$K_TYPE_START_LIST            =00000025     205     (1)
DS$K_TYPE_SUMMARY               =0000000E     205     (1)
DS$K_TYPE_USER_PROMPT           =00000002     205     (1)
DS$K_WARNING                    =00000000     199     (1)
DS$M_ABRTFLG                    =00000040     203     (1)
DS$M_BADTIME                    =00100000     203     (1)
DS$M_BATCH                      =00400000     203     (1)
DS$M_BRKCLR                     =00001000     203     (1)
DS$M_BRKPT                      =00000800     203     (1)
DS$M_CHARFLG                    =00000100     203     (1)
DS$M_CMDFLG                     =00000080     203     (1)
DS$M_CTRLC                      =00000001     203     (1)
DS$M_CTRLO                      =00010000     203     (1)
DS$M_DEVFLG                     =00000200     203     (1)
DS$M_DISABLCC                   =01000000     203     (1)
DS$M_DONFLG                     =00002000     203     (1)
DS$M_ERRFLG                     =00000010     203     (1)
DS$M_EXCEPT                     =00080000     203     (1)
DS$M_EXETST                     =00040000     203     (1)
DS$M_HLTFLG                     =00000008     203     (1)
DS$M_LODFLG                     =00000002     203     (1)
DS$M_MEMMGT                     =00008000     203     (1)
DS$M_NI                         =04000000     203     (1)
DS$M_OUTPUT                     =00800000     203     (1)
DS$M_RUBFLG                     =00000020     203     (1)
DS$M_SCRIPT                     =00200000     203     (1)
DS$M_SETIMR                     =02000000     203     (1)
DS$M_STRFLG                     =00000004     203     (1)
DS$M_SUBT                       =00004000     203     (1)
DS$M_SYSFLG                     =00000400     203     (1)
DS$M_TIMED                      =02000000     203     (1)
DS$M_TIMED_OUT                  =08000000     203     (1)
DS$M_TIMRON                     =00020000     203     (1)
DS$RAB_OUTPUT                    00000000-XR                  514     (1)
DS$V_ABRTFLG                    =00000006     203     (1)
DS$V_BADTIME                    =00000014     203     (1)
DS$V_BATCH                      =00000016     203     (1)     #-505   (1)
DS$V_BRKCLR                     =0000000C     203     (1)
DS$V_BRKPT                      =0000000B     203     (1)
DS$V_CHARFLG                    =00000008     203     (1)
DS$V_CMDFLG                     =00000007     203     (1)
DS$V_CTRLC                      =00000000     203     (1)
DS$V_CTRLO                      =00000010     203     (1)
DS$V_DEVFLG                     =00000009     203     (1)
DS$V_DISABLCC                   =00000018     203     (1)
DS$V_DONFLG                     =0000000D     203     (1)
DS$V_ERRFLG                     =00000004     203     (1)
DS$V_EXCEPT                     =00000013     203     (1)
DS$V_EXETST                     =00000012     203     (1)
DS$V_HLTFLG                     =00000003     203     (1)
DS$V_LODFLG                     =00000001     203     (1)
DS$V_MEMMGT                     =0000000F     203     (1)
DS$V_NI                         =0000001A     203     (1)
DS$V_OUTPUT                     =00000017     203     (1)     #-843   (1)
DS$V_RUBFLG                     =00000005     203     (1)
DS$V_SCRIPT                     =00000015     203     (1)
```

```
DS$V_SETIMR                     =00000019     203    (1)
DS$V_STRFLG                     =00000002     203    (1)
DS$V_SUBT                       =0000000E     203    (1)
DS$V_SYSFLG                     =0000000A     203    (1)
DS$V_TIMED                      =00000019     203    (1)
DS$V_TIMED_OUT                  =0000001B     203    (1)
DS$V_TIMRON                     =00000011     203    (1)
DS$_AP_NORMAL_BREAK             =00660150     199    (1)
DS$_ARITH                       =006600D0     199    (1)
DS$_ASBE                        =00660118     199    (1)
DS$_BADLINK                     =006600F0     199    (1)
DS$_BADTYPE                     =006600E8     199    (1)    #-1063   (1)
DS$_BIIC                        =00660120     199    (1)
DS$_CHME                        =006600A8     199    (1)
DS$_CHMK                        =006600E0     199    (1)
DS$_DEVNAME                     =00660108     199    (1)
DS$_ERROR                       =00660002     199    (1)    #-1080   (1)
DS$_FHWE                        =00660068     199    (1)
DS$_FRAGBUF                     =00660080     199    (1)
DS$_ICBUSY                      =006600C8     199    (1)
DS$_ICERR                       =006600C0     199    (1)
DS$_IHWE                        =00660060     199    (1)
DS$_ILLCHAR                     =00660018     199    (1)
DS$_ILLPAGCNT                   =00660078     199    (1)
DS$_ILLUNIT                     =00660100     199    (1)    #-1050   (1)
DS$_INIT_FAIL                   =00660140     199    (1)
DS$_INSFMEM                     =00660050     199    (1)
DS$_INVCPU                      =00660130     199    (1)
DS$_IPL2HI                      =006600B8     199    (1)
DS$_IVADDR                      =00660040     199    (1)
DS$_IVVECT                      =00660038     199    (1)
DS$_KRNLSTK                     =00660090     199    (1)
DS$_LOGIC                       =00660070     199    (1)
DS$_MCHK                        =00660088     199    (1)
DS$_MEM_ALLOC_ERR               =00660138     199    (1)
DS$_MMOFF                       =00660058     199    (1)
DS$_NEEDUNIT                    =006600F8     199    (1)
DS$_NODE                        =00660128     199    (1)
DS$_NOPCS                       =00660110     199    (1)
DS$_NORMAL                      =00660001     199    (1)    #-1288   (1)    #-1533   (1)
DS$_NOSUPPORT                   =006600B1     199    (1)
DS$_NOTALLOWMP                  =00660148     199    (1)
DS$_NOTDON                      =00660030     199    (1)
DS$_NOTIMP                      =006600B0     199    (1)      199    (1)
DS$_NULLSTR                     =00660010     199    (1)
DS$_OVERFLOW                    =00660008     199    (1)    #-1047   (1)
DS$_POWER                       =00660098     199    (1)
DS$_PROGERR                     =00660020     199    (1)    #-1529   (1)
DS$_SEVERE                      =00660004     199    (1)
DS$_TRANSL                      =006600A0     199    (1)
DS$_TRUNCATE                    =00660028     199    (1)
DS$_UNEXPINT                    =006600D8     199    (1)
DS$_UNSUPPDEV                   =00660158     199    (1)
DS$_VASFULL                     =00660048     199    (1)
DS$_WARNING                     =00660000     199    (1)      199    (1)
DSA$GL_FLAGS                     0000FE00                     491    (1)      492    (1)      493    (1)      518    (1)
                                                              519    (1)      520    (1)      545    (1)      546    (1)
```

```
                                                    547    (1)      585    (1)      635    (1)      660    (1)
                                                    803    (1)      809    (1)      815    (1)      819    (1)
                                                    826    (1)      827    (1)      838    (1)      857    (1)
                                                    872    (1)    #-903    (1)
DSA$GQ_MSGPTR                    0000FE68           863    (1)
DSA$M_NORPT                     =08000000        #-841    (1)
DSA$V_APT                       =0000001F        #-826    (1)
DSA$V_BINARY                    =00000001        #-857    (1)    #-872    (1)
DSA$V_CRD_AUTOTEST_OFF          =0000000B        #-491    (1)    #-518    (1)    #-545    (1)
DSA$V_CRD_MENUTEST_OFF          =00000010        #-492    (1)    #-519    (1)    #-546    (1)
DSA$V_CRD_MENUTEST_ON           =00000012        #-493    (1)    #-520    (1)    #-547    (1)
DSA$V_IE1                       =00000004        #-819    (1)
DSA$V_IE2                       =00000005        #-815    (1)
DSA$V_IE3                       =00000006        #-809    (1)
DSA$V_IES                       =00000007        #-803    (1)
DSA$V_NORPT                     =0000001B        #-827    (1)
DSA$V_USER                      =0000001C        #-585    (1)    #-635    (1)    #-660    (1)
DSV$SETPAGE                      00000000-R  378    (1)
DSV$SHOWPAGE                     00000027-R  429    (1)
DSX$CVTREG                       0000091E-R  1354   (1)
DSX$PRINT                        000002D3-R  722    (1)    1137    (1)      1145    (1)      1149    (1)      1157    (1)
                                                    1165    (1)      1169    (1)      1237    (1)      1246    (1)
                                                    1254    (1)      1262    (1)      1271    (1)      1279    (1)
                                                    390    (1)      435    (1)
DSX$PRINTB                       0000033B-R  813    (1)    744    (1)
DSX$PRINTF                       00000353-R  824    (1)    741    (1)
DSX$PRINTI                       00000368-R  835    (1)    738    (1)
DSX$PRINTREV                     000004A2-R  1000   (1)
DSX$PRINTS                       00000321-R  801    (1)
DSX$PRINTX                       0000032E-R  807    (1)    747    (1)
DSX$TYPE_OUT                     00000047-R  484    (1)    890    (1)      901    (1)
EQUALMESS                        000003F3-R  308    (1)    1189    (1)
EXP_REV                          0000024D-R  255    (1)  #-1205    (1)    #-1206    (1)    #-1207    (1)    #-1219    (1)
                                                    #-1220    (1)    #-1222    (1)    #-1229    (1)      1232    (1)
                                                    263    (1)
FALSE                           =00000000        206    (1)
FAO_DESC                         00000262-R  258    (1)    1100    (1)
FAO_DESC_2                       0000026C-R  262    (1)    1228    (1)
FAO_LEN                          0000026A-R  260    (1)    1100    (1)    #-1101    (1)
FAO_LEN_2                        00000274-R  264    (1)    1228    (1)    #-1229    (1)
FIRMEXIST$T                      000001A0-R  220    (1)  #-1009    (1)    #-1113    (1)
FIRMMESSA                        000003CA-R  296    (1)    1112    (1)
FMT_BEG_ERR                      00000281-R  269    (1)    1137    (1)      1157    (1)      1237    (1)      1262    (1)
FMT_END_ERR                      000002A5-R  275    (1)    1279    (1)
FMT_H_END                        000003B8-R  290    (1)    1169    (1)
FMT_H_END_ERR                    0000032A-R  281    (1)    1254    (1)
FMT_MODULE                       0000029D-R  272    (1)    1145    (1)      1165    (1)      1246    (1)      1271    (1)
FMT_P_F_END                      000003AF-R  287    (1)    1149    (1)
GREATTHAN                        000003E6-R  305    (1)    1198    (1)
HARDEXIST$T                      000001A4-R  221    (1)  #-1008    (1)    #-1104    (1)
HARDMESSA                        000003C1-R  293    (1)    1103    (1)
HP$Q_DEVICE                      00000000         #-1017    (1)    #-1020    (1)
IO$M_CTRLCAST                    00000000-XR      #-922    (1)
IO$M_TRMNOECHO                   00000000-XR      #-645    (1)    #-651    (1)
IO$_READVBLK                     00000000-XR      #-645    (1)    #-651    (1)
IO$_SETMODE                      00000000-XR      #-922    (1)
IO$_WRITEVBLK                    00000000-XR      #-610    (1)    #-625    (1)    #-632    (1)    #-665    (1)
```

```
                                                        #-670    (1)
KB_CHECK                  00000000-XR                     655    (1)      908    (1)
LESSTHAN                  000003DC-R   302    (1)       1181    (1)
LF                       =0000000A     312    (1)        313    (1)
LINE_COUNT                000001D9-R   239    (1)     #-560    (1)    #-597    (1)    #-673    (1)
LOGNUM                    000001CC-R   228    (1)     #-1003   (1)
L_QUO                     00000230-R   249    (1)     #-1088   (1)    #-1089   (1)    #-1091   (1)    #-1216   (1)
                                                      #-1217   (1)    #-1219   (1)
L_REM                     00000234-R   250    (1)     #-1088   (1)    #-1092   (1)    #-1094   (1)    #-1216   (1)
                                                      #-1220   (1)    #-1222   (1)
MP$GR_BOOTED_PROCESSORS   00000000-XR                 #-906    (1)
OFF                      =00000000     206    (1)
OFFSET$T                  000001AC-R   223    (1)     #-1002   (1)    #-1005   (1)
ON                       =00000001     206    (1)
PAGE_STRING               000003FC-R   313    (1)        625    (1)      632    (1)      665    (1)      670    (1)
PATCHEXIST$T              000001A8-R   222    (1)     #-1010   (1)    #-1122   (1)
PATCHMESSA                000003D3-R   299    (1)       1121    (1)
PRINTREV$_ACTUAL_REV     =00000008     209    (1)     #-1068   (1)    #-1076   (1)    #-1086   (1)    #-1100   (1)
                                                      #-1115   (1)    #-1124   (1)    #-1132   (1)    #-1171   (1)
                                                      #-1279   (1)
PRINTREV$_EXPECTED_REV   =0000000C     209    (1)     #-1070   (1)    #-1171   (1)    #-1208   (1)    #-1214   (1)
                                                      #-1228   (1)    #-1279   (1)
PRINTREV$_LOG_UNIT       =00000004     209    (1)     #-1002   (1)    #-1003   (1)    #-1012   (1)
PRINTREV$_MODULENUM      =0000001C     209    (1)     #-1033   (1)    #-1038   (1)    #-1140   (1)    #-1145   (1)
                                                      #-1160   (1)    #-1165   (1)    #-1241   (1)    #-1246   (1)
                                                      #-1266   (1)    #-1271   (1)
PRINTREV$_NARGS          =00000007     209    (1)
PRINTREV$_PRINTMASK      =00000014     209    (1)       1084    (1)      1127    (1)      1178    (1)      1186    (1)
                                                        1195    (1)      1212    (1)
PRINTREV$_PRLINK         =00000018     209    (1)     #-1281   (1)      1286    (1)
PRINTREV$_REV_TYPE       =00000010     209    (1)     #-1053   (1)    #-1056   (1)    #-1060   (1)    #-1130   (1)
                                                      #-1202   (1)
PRINT_X                   0000045F-R   905    (1)     #-805    (1)    #-811    (1)    #-817    (1)    #-821    (1)
                                                      #-832    (1)    #-855    (1)
PRMSK$M_ALWAYS           =00000008     209    (1)
PRMSK$M_EQL              =00000002     209    (1)
PRMSK$M_GTR              =00000004     209    (1)
PRMSK$M_LSS              =00000001     209    (1)
PRMSK$M_TRANSL           =00000010     209    (1)
PRMSK$V_ALWAYS           =00000003     209    (1)     #-1127   (1)
PRMSK$V_EQL              =00000001     209    (1)     #-1186   (1)
PRMSK$V_GTR              =00000002     209    (1)     #-1195   (1)
PRMSK$V_LSS              =00000000     209    (1)     #-1178   (1)
PRMSK$V_TRANSL           =00000004     209    (1)     #-1083   (1)    #-1211   (1)
PRV$M_FWREV              =00000002     209    (1)     #-1056   (1)
PRV$M_HWREV              =00000001     209    (1)     #-1053   (1)    #-1130   (1)    #-1202   (1)
PRV$M_PREV               =00000004     209    (1)     #-1060   (1)
PRV$V_FWREV              =00000001     209    (1)
PRV$V_HWREV              =00000000     209    (1)
PRV$V_PREV               =00000002     209    (1)
P_TABLE                   000001B0-R   224    (1)       1012    (1)    #-1013   (1)
QBUFF                     000001D0-R   234    (1)     #-840    (1)      848    (1)    #-859    (1)    #-861    (1)
                                                        865    (1)
Q_DIVD                    00000228-R   248    (1)     #-1086   (1)    #-1087   (1)    #-1214   (1)    #-1215   (1)
RAB$L_RBF                =00000028                    #-527    (1)
RAB$W_RSZ                =00000022                    #-526    (1)
RCTRLC                    00000000-XR                   922    (1)
```

```
RCVTREGX                    00000AD9-R   1535   (1)   #-1530   (1)
RELATION                    000001B8-R   226    (1)   #-1126   (1)   #-1181   (1)   #-1189   (1)   #-1198   (1)
                                                      #-1254   (1)   #-1279   (1)
RESET_INPUT                 0000046E-R   918    (1)   #-537    (1)
SIZ...                     =00000001     209    (1)     203    (1)     209    (1)
SS$_NORMAL                 =00000001                  #-853    (1)
SYS$CANCEL                  00000000-XR                919    (1)
SYS$FAO                     00000000-XR                1100   (1)     1228   (1)     888    (1)
SYS$FAOL                    00000000-XR                851    (1)
SYS$PUT                     00000000-XR                528    (1)
SYS$QIOW                    00000000-XR                613    (1)     625    (1)     632    (1)     645    (1)
                                                       651    (1)     665    (1)     670    (1)     922    (1)
TRUE                       =00000001     206    (1)
T_PAGE                      000001E1-R   242    (1)     390    (1)
T_SHOW_PAGE                 00000204-R   244    (1)     435    (1)
WARECODE                    000001B4-R   225    (1)   #-1103   (1)   #-1112   (1)   #-1121   (1)   #-1137   (1)
                                                      #-1157   (1)   #-1237   (1)   #-1254   (1)   #-1262   (1)
                                                      #-1279   (1)
WIDTH_TOO_LARGE             000001D8-R   237    (1)   #-565    (1)   #-591    (1)   #-619    (1)   #-626    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CANCEL_S	1	919 (1)	919 (1)
$CRD_LITERALS	5	206 (1)	206 (1)
$D1_PRINT_S	1	390 (1)	1137 (1) 1145 (1) 1149 (1) 1157 (1) 1165 (1) 1169 (1) 1237 (1) 1246 (1) 1254 (1) 1262 (1) 1271 (1) 1279 (1) 390 (1) 435 (1)
$DEF	1	209 (1)	
$DEFINI	1	198 (1)	198 (1) 200 (1) 201 (1) 204 (1)
$DS_CVTDEF	1	202 (1)	202 (1)
$DS_DSADEF	5	201 (1)	201 (1)
$DS_DSDEF	3	199 (1)	199 (1)
$DS_GPHARD_S	1	1012 (1)	1012 (1)
$DS_HPODEF	2	200 (1)	200 (1)
$DS_PRINTREV_DEF	1	209 (1)	209 (1)
$DS_PRMSKDEF	1	209 (1)	209 (1)
$DS_PRVDEF	1	209 (1)	209 (1)
$DS_TYPEDEF	4	205 (1)	205 (1)
$EQU	1	209 (1)	199 (1) 202 (1) 205 (1) 206 (1)
$EQULS1	1	206 (1)	199 (1) 202 (1) 205 (1) 206 (1)
$EQULST	1	199 (1)	199 (1) 202 (1) 205 (1) 206 (1)
$FAO_S	2	1097 (1)	1097 (1) 1225 (1)
$GBLINI	2		199 (1) 202 (1) 203 (1) 205 (1) 206 (1) 209 (1)
$OFFDEF	1	209 (1)	209 (1)
$PRINT	2	388 (1)	1133 (1) 1142 (1) 1146 (1) 1153 (1) 1162 (1) 1166 (1) 1233 (1) 1243 (1) 1248 (1) 1258 (1) 1268 (1) 1273 (1) 388 (1) 432 (1)
$PUSHADR	1	625 (1)	1100 (1) 1228 (1) 625 (1) 632 (1) 645 (1) 651 (1) 665 (1) 670 (1) 922 (1)
$PUSHTWO	1	625 (1)	625 (1) 632 (1) 645 (1) 651 (1) 665 (1) 670 (1) 922 (1)
$PUT	1	528 (1)	528 (1)
$QIOPUSH	1	625 (1)	625 (1) 632 (1) 645 (1) 651 (1) 665 (1) 670 (1) 922 (1)
$QIOW_S	1	621 (1)	621 (1) 628 (1) 641 (1) 647 (1) 661 (1) 666 (1) 920 (1)
$RABDEF	6	204 (1)	204 (1)
$RMSCALL	2	528 (1)	528 (1)
$SSDEF	21	198 (1)	198 (1)
$VIELD	1		203 (1) 209 (1)
$VIELD1	1	209 (1)	203 (1) 209 (1)
BR_IF_CRD_AUTOTEST_OFF	1	491 (1)	491 (1) 518 (1) 545 (1)
BR_IF_CRD_MENUTEST_OFF	1	492 (1)	492 (1) 519 (1) 546 (1)
BR_IF_CRD_MENUTEST_ON	1	493 (1)	493 (1) 520 (1) 547 (1)
BR_IF_NOT_APT	1	826 (1)	826 (1)
BR_IF_NOT_BATCH	1	505 (1)	505 (1)
BR_IF_NOT_BINARY	1	857 (1)	857 (1) 872 (1)
BR_IF_NOT_IE1	1	819 (1)	819 (1)

BR_IF_NOT_IE2	1	815	(1)	815	(1)				
BR_IF_NOT_IE3	1	809	(1)	809	(1)				
BR_IF_NOT_IES	1	803	(1)	803	(1)				
BR_IF_NOT_NORPT	1	827	(1)	827	(1)				
BR_IF_NOT_USER	1	585	(1)	585	(1)	635	(1)	660	(1)
CASE	1	731	(1)	731	(1)				
CLIDEF	4	207	(1)	207	(1)				
DSFDEF	3	203	(1)	203	(1)				
SET_OUTPUT	1	843	(1)	843	(1)				

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDB3	0081	1091 (1) 1092 (1) 1094 (1) 1219 (1) 1220 (1) 1222 (1)
ADDL	00C0	1055 (1) 1059 (1)
ADDL2	00C0	1005 (1) 1025 (1) 1042 (1) 1045 (1) 1379 (1) 1483 (1) 1486 (1)
		1504 (1)
ADDL3	00C1	1367 (1) 1477 (1) 845 (1)
AOBLSS	00F2	584 (1) 597 (1)
BBC	00E1	1178 (1) 1186 (1) 1195 (1) 505 (1) 585 (1) 635 (1) 660 (1)
		803 (1) 809 (1) 815 (1) 819 (1) 826 (1) 827 (1) 857 (1)
		872 (1)
BBS	00E0	1083 (1) 1127 (1) 1211 (1) 1425 (1) 491 (1) 492 (1) 493 (1)
		518 (1) 519 (1) 520 (1) 545 (1) 546 (1) 547 (1)
BBSS	00E2	843 (1)
BEQL	0013	1015 (1) 1054 (1) 1090 (1) 1131 (1) 1141 (1) 1161 (1) 1172 (1)
		1203 (1) 1218 (1) 1242 (1) 1267 (1) 1282 (1) 1357 (1) 1383 (1)
		1394 (1) 1411 (1) 1413 (1) 488 (1) 558 (1) 571 (1) 590 (1)
		620 (1) 653 (1) 854 (1)
BGEQ	0018	1370 (1) 1419 (1) 726 (1)
BGEQU	001E	1392 (1) 1500 (1) 1510 (1)
BGTR	0014	1372 (1) 1514 (1) 1516 (1) 616 (1)
BGTRU	001A	1173 (1)
BICL2	00CA	842 (1)
BICL3	00CB	841 (1)
BISB	0088	591 (1) 636 (1)
BISL2	00C8	903 (1)
BITB	0093	581 (1) 619 (1)
BLBC	00E9	502 (1) 536 (1) 871 (1)
BLBS	00E8	751 (1)
BLSS	0019	1069 (1) 1071 (1) 1385 (1) 1397 (1) 1417 (1) 1442 (1) 1482 (1)
		1492 (1) 1508 (1) 1520 (1) 380 (1) 561 (1) 568 (1)
BLSSU	001F	1361 (1) 1407 (1)
BNEQ	0012	1007 (1) 1029 (1) 1032 (1) 1037 (1) 1040 (1) 1057 (1) 1061 (1)
		1077 (1) 1139 (1) 1159 (1) 1209 (1) 1240 (1) 1265 (1) 1403 (1)
		1435 (1) 1452 (1) 1459 (1) 1466 (1) 1502 (1) 574 (1) 579 (1)
		582 (1) 587 (1) 907 (1)
BRB	0011	1034 (1) 1043 (1) 1182 (1) 1190 (1) 1421 (1) 1448 (1) 1456 (1)
		1463 (1) 1470 (1) 1484 (1) 1517 (1) 1522 (1) 1525 (1) 1530 (1)
		494 (1) 521 (1) 548 (1) 576 (1) 593 (1) 646 (1) 658 (1)
		739 (1) 742 (1) 745 (1) 875 (1)
BRW	0031	1048 (1) 1051 (1) 1058 (1) 1062 (1) 1064 (1) 1078 (1) 1081 (1)
		1085 (1) 1093 (1) 1095 (1) 1107 (1) 1116 (1) 1128 (1) 1150 (1)
		1179 (1) 1187 (1) 1196 (1) 1204 (1) 1210 (1) 1213 (1) 1221 (1)
		1223 (1) 1256 (1) 1358 (1) 1363 (1) 1408 (1) 1423 (1) 1427 (1)
		1429 (1) 1472 (1) 1511 (1) 562 (1) 676 (1) 805 (1) 811 (1)
		817 (1) 821 (1) 832 (1) 855 (1)
BSBW	0030	537 (1)
CALLG	00FA	1286 (1) 750 (1)
CALLS	00FB	1012 (1) 1100 (1) 1137 (1) 1145 (1) 1149 (1) 1157 (1) 1165 (1)
		1169 (1) 1228 (1) 1237 (1) 1246 (1) 1254 (1) 1262 (1) 1271 (1)
		1279 (1) 390 (1) 435 (1) 500 (1) 528 (1) 613 (1) 625 (1)
		632 (1) 645 (1) 651 (1) 665 (1) 670 (1) 851 (1) 888 (1)

		890	(1)	901	(1)	919	(1)	922	(1)						
CASEB	008F	736	(1)												
CLRB	0094	1527	(1)	565	(1)	626	(1)	633	(1)	656	(1)	672	(1)	752	(1)
		884	(1)												
CLRL	00D4	1008	(1)	1009	(1)	1010	(1)	1126	(1)	1404	(1)	1479	(1)	391	(1)
		554	(1)	575	(1)	596	(1)	599	(1)	609	(1)	612	(1)	673	(1)
CLRQ	007C	1016	(1)	539	(1)	540	(1)	603	(1)	604	(1)	608	(1)	625	(1)
		632	(1)	645	(1)	651	(1)	665	(1)	670	(1)	674	(1)	675	(1)
		922	(1)												
CMPB	0091	1393	(1)	1402	(1)	1410	(1)	1412	(1)	1434	(1)	1451	(1)	1458	(1)
		1465	(1)	1501	(1)	486	(1)	570	(1)	573	(1)	578	(1)	589	(1)
		652	(1)												
CMPL	00D1	1031	(1)	1036	(1)	1039	(1)	1053	(1)	1056	(1)	1060	(1)	1068	(1)
		1070	(1)	1130	(1)	1138	(1)	1158	(1)	1171	(1)	1202	(1)	1239	(1)
		1264	(1)	1356	(1)	1360	(1)	1371	(1)	1382	(1)	1384	(1)	1391	(1)
		1396	(1)	1406	(1)	1418	(1)	1441	(1)	1481	(1)	1491	(1)	1499	(1)
		1509	(1)	1513	(1)	1515	(1)	1519	(1)	560	(1)	853	(1)		
CVTBL	0098	1137	(1)	1145	(1)	1149	(1)	1157	(1)	1165	(1)	1169	(1)	1237	(1)
		1246	(1)	1254	(1)	1262	(1)	1271	(1)	1279	(1)	390	(1)	435	(1)
DECL	00D7	1507	(1)	523	(1)	550	(1)	567	(1)	583	(1)	859	(1)		
DIVL2	00C6	1455	(1)	1462	(1)	1469	(1)								
EDIV	007B	1087	(1)	1215	(1)	1480	(1)								
EMUL	007A	1420	(1)												
EXTZV	00EF	1433	(1)												
INCL	00D6	1389	(1)	1401	(1)	1431	(1)	1436	(1)	1446	(1)	1450	(1)	1496	(1)
		1506	(1)	524	(1)	551	(1)	580	(1)	595	(1)	728	(1)	858	(1)
JSB	0016	655	(1)	908	(1)										
MNEGL	00CE	1373	(1)	727	(1)										
MOVAB	009E	514	(1)	738	(1)	741	(1)	744	(1)	747	(1)	839	(1)	840	(1)
		860	(1)	874	(1)	882	(1)	891	(1)						
MOVAL	00DE	1004	(1)	1022	(1)	1103	(1)	1112	(1)	1121	(1)	1152	(1)	1181	(1)
		1189	(1)	1198	(1)	1231	(1)	1232	(1)	1368	(1)	1375	(1)	634	(1)
		657	(1)	838	(1)	865	(1)								
MOVAQ	007E	863	(1)												
MOVB	0090	1017	(1)	1073	(1)	1074	(1)	1075	(1)	1101	(1)	1104	(1)	1113	(1)
		1122	(1)	1205	(1)	1206	(1)	1207	(1)	1229	(1)	1283	(1)	1388	(1)
		1400	(1)	1445	(1)	1474	(1)	1475	(1)	1476	(1)	1488	(1)	1495	(1)
		1521	(1)	730	(1)	861	(1)								
MOVC3	0028	1020	(1)												
MOVL	00D0	1003	(1)	1013	(1)	1018	(1)	1023	(1)	1026	(1)	1033	(1)	1038	(1)
		1047	(1)	1050	(1)	1063	(1)	1080	(1)	1086	(1)	1106	(1)	1115	(1)
		1124	(1)	1132	(1)	1214	(1)	1288	(1)	1365	(1)	1366	(1)	1369	(1)
		1374	(1)	1376	(1)	1386	(1)	1398	(1)	1437	(1)	1438	(1)	1443	(1)
		1453	(1)	1460	(1)	1467	(1)	1493	(1)	1529	(1)	1533	(1)	379	(1)
		382	(1)	384	(1)	436	(1)	516	(1)	527	(1)	543	(1)	553	(1)
		555	(1)	556	(1)	724	(1)	866	(1)						
MOVQ	007D	889	(1)												
MOVW	00B0	526	(1)												
MOVZBL	009A	1137	(1)	1145	(1)	1149	(1)	1157	(1)	1165	(1)	1169	(1)	1237	(1)
		1246	(1)	1254	(1)	1262	(1)	1271	(1)	1279	(1)	1378	(1)	1415	(1)
		1439	(1)	390	(1)	430	(1)	435	(1)	498	(1)	564	(1)	846	(1)
		881	(1)	883	(1)										
MOVZWL	003C	497	(1)	515	(1)	542	(1)	625	(1)	632	(1)	645	(1)	651	(1)
		665	(1)	670	(1)	864	(1)	900	(1)	919	(1)	922	(1)		
MULL3	00C5	1002	(1)	1461	(1)										
POPR	00BA	1021	(1)	1426	(1)	892	(1)								
PUSHAB	009F	1137	(1)	1145	(1)	1149	(1)	1157	(1)	1165	(1)	1169	(1)	1237	(1)

		1246	(1)	1254	(1)	1262	(1)	1271	(1)	1279	(1)	390	(1)	435	(1)
		880	(1)	885	(1)	886	(1)	887	(1)						
PUSHAL	00DF	1012	(1)	528	(1)	625	(1)	632	(1)	645	(1)	651	(1)	665	(1)
		670	(1)	847	(1)	922	(1)								
PUSHAQ	007F	1100	(1)	1228	(1)	848	(1)	850	(1)						
PUSHAW	003F	1100	(1)	1228	(1)	849	(1)								
PUSHL	00DD	1012	(1)	1100	(1)	1137	(1)	1145	(1)	1149	(1)	1157	(1)	1165	(1)
		1169	(1)	1228	(1)	1237	(1)	1246	(1)	1254	(1)	1262	(1)	1271	(1)
		1279	(1)	435	(1)	496	(1)	499	(1)	605	(1)	606	(1)	610	(1)
		611	(1)	625	(1)	632	(1)	645	(1)	651	(1)	665	(1)	670	(1)
		723	(1)	899	(1)	922	(1)								
PUSHR	00BB	1019	(1)	1381	(1)	873	(1)								
RET	0004	1289	(1)	1536	(1)	503	(1)	529	(1)	617	(1)	677	(1)	754	(1)
		910	(1)												
RSB	0005	385	(1)	392	(1)	437	(1)	923	(1)						
SOBGEQ	00F4	1489	(1)	1497	(1)										
SOBGTR	00F5	1046	(1)	1447	(1)										
SUBB3	0083	1524	(1)	1532	(1)										
SUBL2	00C2	1416	(1)	1432	(1)	586	(1)	879	(1)						
SUBL3	00C3	1454	(1)	1468	(1)	600	(1)	615	(1)	749	(1)				
TSTB	0095	654	(1)	659	(1)										
TSTL	00D5	1006	(1)	1014	(1)	1028	(1)	1076	(1)	1089	(1)	1140	(1)	1160	(1)
		1208	(1)	1217	(1)	1241	(1)	1266	(1)	1281	(1)	557	(1)	725	(1)
		906	(1)												

```
                                  +-----------------------------+
                                  ! Directives Cross Reference !
                                  +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.ADDRESS        333    (1)    334    (1)    335    (1)    336    (1)
.ASCIC          243    (1)    245    (1)    270    (1)    273    (1)    276    (1)    282    (1)    288    (1)    291    (1)
                294    (1)    297    (1)    300    (1)    303    (1)    306    (1)    309    (1)
.ASCID          266    (1)    877    (1)
.ASCII          313    (1)
.BLKL           207    (1)    218    (1)    220    (1)    221    (1)    222    (1)    223    (1)    224    (1)    225    (1)
                226    (1)    228    (1)
.BLKQ           227    (1)
.BYTE           237    (1)    253    (1)    254    (1)    255    (1)    256    (1)    318    (1)
.DSABL          912    (1)
.ENABL          799    (1)
.END            1538   (1)
.ENDC           1100   (1)    1137   (1)    1145   (1)    1149   (1)    1157   (1)    1165   (1)    1169   (1)    1228   (1)
                1237   (1)    1246   (1)    1254   (1)    1262   (1)    1271   (1)    1279   (1)    199    (1)    202    (1)
                203    (1)    205    (1)    206    (1)    209    (1)    390    (1)    435    (1)    528    (1)    625    (1)
                632    (1)    645    (1)    651    (1)    665    (1)    670    (1)    922    (1)
.ENDM           1012   (1)    1097   (1)    1100   (1)    1228   (1)    198    (1)    199    (1)    200    (1)    201    (1)
                202    (1)    203    (1)    204    (1)    205    (1)    206    (1)    207    (1)    209    (1)    388    (1)
                390    (1)    491    (1)    492    (1)    493    (1)    505    (1)    528    (1)    585    (1)    621    (1)
                625    (1)    731    (1)    803    (1)    809    (1)    815    (1)    819    (1)    826    (1)    827    (1)
                843    (1)    857    (1)    919    (1)
.ENDR           1137   (1)    1145   (1)    1149   (1)    1157   (1)    1165   (1)    1169   (1)    1237   (1)    1246   (1)
                1254   (1)    1262   (1)    1271   (1)    1279   (1)    199    (1)    202    (1)    203    (1)    205    (1)
                206    (1)    209    (1)    390    (1)    435    (1)    736    (1)
.ENTRY          1000   (1)    1354   (1)    484    (1)    722    (1)    801    (1)    807    (1)    813    (1)    824    (1)
                835    (1)
.GLOBL          528    (1)    625    (1)    632    (1)    645    (1)    651    (1)    665    (1)    670    (1)    919    (1)
                922    (1)
.IDENT          5      (1)
.IF             1100   (1)    1137   (1)    1145   (1)    1149   (1)    1157   (1)    1165   (1)    1169   (1)    1228   (1)
                1237   (1)    1246   (1)    1254   (1)    1262   (1)    1271   (1)    1279   (1)    199    (1)    202    (1)
                203    (1)    205    (1)    206    (1)    209    (1)    390    (1)    435    (1)    528    (1)    625    (1)
                632    (1)    645    (1)    651    (1)    665    (1)    670    (1)    922    (1)
.IFF            1100   (1)    1228   (1)    199    (1)    202    (1)    203    (1)    205    (1)    206    (1)    209    (1)
                528    (1)    625    (1)    632    (1)    645    (1)    651    (1)    665    (1)    670    (1)    922    (1)
.IIF            199    (1)    202    (1)    203    (1)    205    (1)    206    (1)    209    (1)
.IRP            1100   (1)    1137   (1)    1145   (1)    1149   (1)    1157   (1)    1165   (1)    1169   (1)    1228   (1)
                1237   (1)    1246   (1)    1254   (1)    1262   (1)    1271   (1)    1279   (1)    199    (1)    202    (1)
                203    (1)    205    (1)    206    (1)    209    (1)    390    (1)    435    (1)    736    (1)
.LIBRARY        187    (1)    188    (1)    189    (1)    190    (1)
.LIST           201    (1)    207    (1)    209    (1)
.LONG           235    (1)    239    (1)    241    (1)    249    (1)    250    (1)    258    (1)    259    (1)    262    (1)
                263    (1)
.MACRO          199    (1)    202    (1)    203    (1)    205    (1)    206    (1)    209    (1)
.MDELETE        199    (1)    202    (1)
.NLIST          201    (1)    207    (1)    209    (1)
.NOCROSS        198    (1)    200    (1)    201    (1)    204    (1)
.NOSHOW         210    (1)    6      (1)
.NTYPE          528    (1)
.PAGE           1291   (1)    1353   (1)    182    (1)    211    (1)    27     (1)    319    (1)    337    (1)    377    (1)
```

```
                393     (1)     428     (1)     439     (1)     483     (1)     679     (1)     721     (1)     755     (1)     798     (1)
                925     (1)
.PSECT          207     (1)     213     (1)     321     (1)     339     (1)     441     (1)
.QUAD           248     (1)
.RESTORE        198     (1)     200     (1)     201     (1)     204     (1)     207     (1)
.SAVE           198     (1)     200     (1)     201     (1)     204     (1)     207     (1)
.SBTTL          926     (1)
.SHOW           208     (1)
.SIGNED_WORD    736     (1)
.SUBTITLE       1292    (1)     183     (1)     212     (1)     320     (1)     338     (1)     394     (1)     440     (1)     680     (1)
                756     (1)
.TITLE          4       (1)
.WORD           234     (1)     260     (1)     264     (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	85	#-1002	(1)	#-1003	(1)	#-1012	(1)	#-1031	(1)	#-1033	(1)	#-1036	(1)
		#-1038	(1)	#-1053	(1)	#-1056	(1)	#-1060	(1)	#-1068	(1)	#-1070	(1)
		#-1076	(1)	1084	(1)	#-1086	(1)	#-1100	(1)	#-1115	(1)	#-1124	(1)
		1127	(1)	#-1130	(1)	#-1132	(1)	#-1138	(1)	#-1140	(1)	#-1145	(1)
		#-1158	(1)	#-1160	(1)	#-1165	(1)	#-1171	(1)	1178	(1)	1186	(1)
		1195	(1)	#-1202	(1)	#-1208	(1)	1212	(1)	#-1214	(1)	#-1228	(1)
		#-1239	(1)	#-1241	(1)	#-1246	(1)	#-1264	(1)	#-1266	(1)	#-1271	(1)
		#-1279	(1)	#-1281	(1)	1286	(1)	#-1356	(1)	#-1360	(1)	#-1365	(1)
		#-1367	(1)	1368	(1)	#-1369	(1)	#-1384	(1)	#-1386	(1)	#-1396	(1)
		#-1398	(1)	1425	(1)	1433	(1)	#-1441	(1)	#-1443	(1)	#-1491	(1)
		#-1493	(1)	#-1515	(1)	#-1519	(1)	#-1521	(1)	#-1524	(1)	#-1527	(1)
		#-1532	(1)	#-496	(1)	#-497	(1)	#-515	(1)	#-516	(1)	#-542	(1)
		#-543	(1)	#-724	(1)	#-736	(1)	#-749	(1)	750	(1)	#-845	(1)
		#-846	(1)	847	(1)								
R0	49	#-1038	(1)	#-1039	(1)	#-1047	(1)	#-1050	(1)	#-1063	(1)	#-1080	(1)
		#-1288	(1)	#-1365	(1)	#-1425	(1)	#-1431	(1)	#-1432	(1)	#-1433	(1)
		#-1507	(1)	#-1529	(1)	#-1533	(1)	#-379	(1)	#-382	(1)	#-384	(1)
		#-391	(1)	#-430	(1)	#-435	(1)	#-436	(1)	#-502	(1)	#-514	(1)
		#-526	(1)	#-527	(1)	528	(1)	#-724	(1)	#-725	(1)	#-727	(1)
		#-730	(1)	#-738	(1)	#-741	(1)	#-744	(1)	#-747	(1)	750	(1)
		#-838	(1)	#-841	(1)	#-842	(1)	#-853	(1)	#-860	(1)	#-861	(1)
		#-863	(1)	#-864	(1)	#-866	(1)	#-873	(1)	#-892	(1)		
R1	30	#-1026	(1)	#-1046	(1)	#-1366	(1)	#-1381	(1)	#-1382	(1)	#-1384	(1)
		#-1389	(1)	#-1396	(1)	#-1401	(1)	#-1426	(1)	#-1441	(1)	#-1446	(1)
		#-1491	(1)	#-1496	(1)	#-1513	(1)	#-1515	(1)	#-1532	(1)	#-515	(1)
		#-523	(1)	#-526	(1)	#-54[illegible]	(1)	#-550	(1)	#-553	(1)	#-865	(1)
		#-866	(1)	#-873	(1)	#-883	(1)	#-884	(1)	#-892	(1)	#-899	(1)
R10	3	1354	(1)	#-1477	(1)	#-1497	(1)						
R11	4	1354	(1)	#-1368	(1)	#-1375	(1)	#-1437	(1)				
R2	46	1000	(1)	#-1004	(1)	#-1005	(1)	#-1006	(1)	#-1019	(1)	#-1021	(1)
		#-1023	(1)	#-1025	(1)	#-1028	(1)	#-1033	(1)	#-1039	(1)	#-1042	(1)
		#-1045	(1)	#-1055	(1)	#-1059	(1)	#-1106	(1)	#-1115	(1)	#-1124	(1)
		1354	(1)	#-1367	(1)	#-1381	(1)	#-1386	(1)	#-1388	(1)	#-1398	(1)
		#-1400	(1)	#-1426	(1)	#-1443	(1)	#-1445	(1)	#-1493	(1)	#-1495	(1)
		#-379	(1)	484	(1)	#-516	(1)	#-524	(1)	#-527	(1)	#-543	(1)
		#-551	(1)	#-556	(1)	#-570	(1)	#-573	(1)	#-578	(1)	#-589	(1)
		#-595	(1)	#-600	(1)	#-873	(1)	#-892	(1)				
R3	42	1000	(1)	1354	(1)	#-1369	(1)	#-1371	(1)	#-1373	(1)	#-1374	(1)
		1375	(1)	#-1376	(1)	#-1378	(1)	#-1379	(1)	#-1391	(1)	#-1393	(1)
		#-1400	(1)	#-1402	(1)	#-1406	(1)	#-1410	(1)	#-1412	(1)	#-1415	(1)
		#-1434	(1)	#-1436	(1)	#-1450	(1)	#-1451	(1)	#-1458	(1)	#-1465	(1)
		#-1475	(1)	#-1499	(1)	#-1501	(1)	#-1506	(1)	#-1509	(1)	484	(1)
		#-564	(1)	#-584	(1)	#-586	(1)	#-873	(1)	#-882	(1)	885	(1)
		886	(1)	#-889	(1)	#-892	(1)						
R4	22	1000	(1)	1354	(1)	#-1378	(1)	#-1379	(1)	#-1391	(1)	#-1406	(1)
		#-1499	(1)	#-1509	(1)	484	(1)	#-554	(1)	#-575	(1)	#-580	(1)
		#-581	(1)	#-583	(1)	#-584	(1)	#-586	(1)	#-596	(1)	#-599	(1)
		#-873	(1)	#-874	(1)	891	(1)	#-892	(1)				
R5	18	1000	(1)	1354	(1)	#-1404	(1)	#-1420	(1)	#-1432	(1)	#-1433	(1)
		#-1454	(1)	#-1461	(1)	#-1468	(1)	484	(1)	#-553	(1)	#-555	(1)

```
                        #-567    (1)   #-600    (1)   #-605    (1)   #-615    (1)
R6          24            1000   (1)   #-1013   (1)   #-1014   (1)   #-1017   (1)   #-1018   (1)   #-1020   (1)
                        #-1022   (1)   #-1023   (1)   #-1137   (1)   #-1157   (1)   #-1237   (1)   #-1262   (1)
                          1354   (1)   #-1415   (1)   #-1416   (1)   #-1418   (1)   #-1420   (1)   #-1433   (1)
                        #-1438   (1)   #-1480   (1)     484    (1)   #-555    (1)   #-615    (1)
R7          29            1000   (1)   #-1018   (1)    1020    (1)   #-1132   (1)   #-1149   (1)   #-1152   (1)
                        #-1169   (1)   #-1231   (1)   #-1254   (1)     1354   (1)   #-1437   (1)   #-1438   (1)
                        #-1439   (1)   #-1445   (1)   #-1479   (1)   #-1480   (1)   #-1481   (1)   #-1483   (1)
                        #-1486   (1)   #-1488   (1)     484    (1)   #-556    (1)   #-600    (1)   #-606    (1)
                        #-634    (1)     645   (1)     651    (1)   #-657    (1)
R8           8            1000   (1)   #-1232   (1)   #-1254   (1)     1354   (1)   #-1453   (1)   #-1460   (1)
                        #-1467   (1)   #-1480   (1)
R9          11            1354   (1)   #-1439   (1)   #-1447   (1)   #-1454   (1)   #-1455   (1)   #-1461   (1)
                        #-1462   (1)   #-1468   (1)   #-1469   (1)   #-1477   (1)   #-1489   (1)
SP         108          #-1137   (1)   #-1145   (1)   #-1149   (1)   #-1157   (1)   #-1165   (1)   #-1169   (1)
                        #-1237   (1)   #-1246   (1)   #-1254   (1)   #-1262   (1)   #-1271   (1)   #-1279   (1)
                        #-1474   (1)   #-1475   (1)   #-1476   (1)   #-1488   (1)   #-1495   (1)   #-1504   (1)
                        #-390    (1)   #-435    (1)   #-497    (1)   #-498    (1)   #-539    (1)   #-540    (1)
                        #-603    (1)   #-604    (1)   #-608    (1)   #-609    (1)   #-612    (1)   #-625    (1)
                        #-632    (1)   #-633    (1)     634    (1)   #-645    (1)   #-651    (1)   #-652    (1)
                        #-654    (1)   #-656    (1)     657    (1)   #-659    (1)   #-665    (1)   #-670    (1)
                        #-674    (1)   #-675    (1)   #-728    (1)   #-751    (1)     839    (1)     840    (1)
                        #-841    (1)   #-842    (1)   #-845    (1)   #-846    (1)     850    (1)     874    (1)
                        #-879    (1)     880   (1)   #-881    (1)     882    (1)   #-883    (1)   #-889    (1)
                        #-891    (1)   #-900    (1)   #-903    (1)   #-919    (1)   #-922    (1)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    22    00:00:00.02    00:00:00.26
Command processing               896    00:00:00.21    00:00:00.69
Pass 1                          1200    00:00:04.54    00:00:07.77
Symbol table sort                  0    00:00:00.29    00:00:00.30
Pass 2                           105    00:00:01.29    00:00:02.67
Symbol table output                2    00:00:00.06    00:00:00.14
Psect synopsis output              2    00:00:00.01    00:00:00.01
Cross-reference output             9    00:00:00.57    00:00:01.00
Assembler run totals            2238    00:00:06.99    00:00:12.84

The working set limit was 3400 pages.
248434 bytes (486 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 978 non-local and 130 local symbols.
1538 source lines were read in Pass 1, producing 102 object records in Pass 2.
136 pages of virtual memory were used to define 48 macros.
```

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              11
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                16
DISK$DS:[DS.LOCAL.LIBRARY]CRD.MLB;4                     1
SYS$COMMON:[SYSLIB]LIB.MLB;2                            0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5               0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                 0
SYS$COMMON:[SYSLIB]LIB.MLB;2                            0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       16
TOTALS (all libraries)                                 44

1230 GETS were required to define 44 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:PRINT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element PROBE.B32
;    0002  0      ! *1     6-FEB-1986 15:21:52 DS ""
;    0003  0      ! DEC/CMS REPLACEMENT HISTORY, Element PROBE.B32
;    0004  0      %title '*** PROBE Check address accessibility'
;    0005  0      module probe (                  ! Routine to test accessibility of addresses
;    0006  0                    ident = '01-05'
;    0007  0                    ) =
;    0008  0      !
;    0009  0      ! Copyright (c) 1979, 1981, 1983
;    0010  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0011  0      !
;    0012  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0013  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0014  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0015  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0016  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0017  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0018  0      ! REMAIN IN DEC.
;    0019  0      !
;    0020  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0021  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0022  0      ! CORPORATION.
;    0023  0      !
;    0024  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0025  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0026  0
;    0027  0      !++
;    0028  0      ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0029  0      !
;    0030  0      ! ABSTRACT:
;    0031  0      !
;    0032  0      !       This routine tests the accessibility of an address.
;    0033  0      !
;    0034  0      ! AUTHOR:       Roger Riggs, CREATION DATE: 10-November-1979
;    0035  0      ! MODIFIED BY:
;    0036  0      !
;    0037  0      !               - Dave Butenhof, 02-Jun-1981, version 6.4
;    0038  0      !       01      Alter library spec. for flexibility (logical name)
;    0039  0      !
;    0040  0      !       02      - Dave Butenhof, 23-Nov-1981, verison 6.5
;    0041  0      !               Fix truncation error.
;    0042  0      !
;    0043  0      !       03      Bob Bergazzi    May 17, 1983    Version 6.11
;    0044  0      !               Changed the order of searching libraries.
;    0045  0      !
;    0046  0      !       04      Bob Bergazzi    May 24, 1983    Version 6.12
;    0047  0      !               Removed the call to DSR$FAULT_CLEAR from the SELECT statement
;    0048  0      !               in the ACC$HANDLER routine. This routine was being called
;    0049  0      !               twice, once at EXE_MCHK in EXCEPT and then here if the condition
;    0050  0      !               was machine check.
;    0051  0      !
;    0052  0      !       05      Domenic Andella         7-Sep-83        Version 6.13
;    0053  0      !               Deleted External reference to DSR$FAULT_CLEAR routine
;    0054  0      !--
;    0055  0
;    0056  1      begin
;    0057  1      !
```

```
1       ! TABLE OF CONTENTS:
1       !
1
1       forward routine
1           dsx$probe,                                  ! Routine to test address
1           acc$handler;                                ! Exception handler routine for PROBE
1
1       !
1       ! INCLUDE FILES:
1       !
1
1       library '$DIAG';                !
1
1       library '$DS';                  !
1
1       library 'SYS$LIBRARY:STARLET.L32';
1
1       linkage
1           quick = jsb;
1
1       !
1       ! EXTERNAL REFERENCES:
1       !
1
1       external
1           io$gq_physical : vector [2, long]
1               addressing_mode (long_relative);        ! Quad descriptor of I/O space
1
1       !
1       ! PSECT DEFINITIONS:
1       !
1
1       psect
1           plit = data ( share, addressing_mode (long_relative));
1
1       psect
1           code = sep ( read, write, share);
1
```

```
;    0097  1      %sbttl 'DSX$PROBE'
;    0098  1
;    0099  1      global routine dsx$probe (address, length, unit) =
;    0100  1
;    0101  1      !++
;    0102  1      ! FUNCTIONAL DESCRIPTION:
;    0103  1      !
;    0104  1      !       This routine checks that the address specified can be
;    0105  1      !       accessed using instructions that access data with the specified
;    0106  1      !       length.  The Unit number reflects the device expected
;    0107  1      !       to respond to this address, (i.e. unibus adapter)
;    0108  1      !
;    0109  1      ! FORMAL PRAMETERS:
;    0110  1      !
;    0111  1      !       ADDRESS Address to be accessed
;    0112  1      !       LENGTH  Length of reference, (1,2,4)
;    0113  1      !       UNIT    Unit number of device responding to this address
;    0114  1      !
;    0115  1      ! IMPLICIT INPUTS:      NONE
;    0116  1      !
;    0117  1      ! IMPLICIT OUTPUTS:     NONE
;    0118  1      !
;    0119  1      ! COMPLETION CODES:
;    0120  1      !
;    0121  1      ! SIDE EFFECTS:
;    0122  1      !
;    0123  1      !--
;    0124  1
;    0125  2          begin
;    0126  2
;    0127  2          stacklocal
;    0128  2              ch_status : $ds_chs_decl;               ! Storage for channel status
;    0129  2
;    0130  2          local
;    0131  2              rc,
;    0132  2              scratch;
;    0133  2
;    0134  2          builtin
;    0135  2              fp;
;    0136  2
;    0137  2          .fp = acc$handler;                          ! Set exception handler
;    0138  2
;    0139  2          if .address<0, 30> gequ .io$gq_physical [0]
;    0140  2          then
;    0141  3              begin
;    0142  3
;    0143  3              if not (rc = $ds_channel (unit = .unit, func = chc$_clear)) then return .rc;
;    0144  3
;    0145  2              end;
;    0146  2
;    0147  2          case .length from 1 to 4 of
;    0148  2              set
;    0149  2
;    0150  2              [1] :
;    0151  2                  scratch = .(.address)<0, 8>;
;    0152  2
;    0153  2              [2] :
```

```
;   0154  2                 scratch = .(.address)<0, 16>;
;   0155  2
;   0156  2             [4] :
;   0157  2                 scratch = .(.address)<0, 32>;
;   0158  2
;   0159  2             [inrange, outrange] :
;   0160  2                 return ds$_error;
;   0161  2             tes;
;   0162  2
;   0163  2         if .address<0, 30> gequ .io$gq_physical [0]
;   0164  2         then
;   0165  3             begin
;   0166  3
;   0167  3             local
;   0168  3                 rc;
;   0169  3
;   0170  3             if not (rc = $ds_channel (unit = .unit, func = chc$_status, stsadr = ch_status)) then return .rc;
;   0171  3
;   0172  3             if .ch_status [chs$v_errany] neq 0 then return ds$_error;
;   0173  3
;   0174  2             end;
;   0175  2
;   0176  2         ss$_normal
;   0177  1         end;


                                                                  .TITLE  PROBE *** PROBE Check address accessibility
                                                                  .IDENT  \01-05\

                                                                  .EXTRN  IO$GQ_PHYSICAL, DS$CHANNEL

                                                                  .PSECT  SEP,  SHR,2

                                          000C 00000              .ENTRY  DSX$PROBE, Save R2,R3                          ; 0099
                        53 00000000G  9F   9E 00002              MOVAB   @#DS$CHANNEL, R3                               ;
                        52 00000000G  EF   9E 00009              MOVAB   IO$GQ_PHYSICAL, R2                             ;
                        5E            08   C2 00010              SUBL2   #8, SP                                         ;
                        6D     0000V  CF   9E 00013              MOVAB   ACC$HANDLER, (FP)                              ; 0137
   62       04   AC     1E            00   ED 00018              CMPZV   #0, #30, ADDRESS, IO$GQ_PHYSICAL               ; 0139
                                      0F   1F 0001E              BLSSU   1$                                             ;
                                      7E   7C 00020              CLRQ    -(SP)                                          ; 0143
                                      7E   7C 00022              CLRQ    -(SP)                                          ;
                                      06   DD 00024              PUSHL   #6                                             ;
                                 0C   AC   DD 00026              PUSHL   UNIT                                           ;
                        63            06   FB 00029              CALLS   #6, DS$CHANNEL                                 ;
                        48            50   E9 0002C              BLBC    RC, 9$                                         ;
                 03     01       08   AC   CF 0002F 1$:          CASEL   LENGTH, #1, #3                                 ; 0147
 0016          0038   0010          000A      00034 2$:          .WORD   3$-2$,-                                        ;
                                                                        4$-2$,-                                        ;
                                                                        7$-2$,-                                        ;
                                                                        5$-2$                                          ;
                                      2E   11 0003C              BRB     7$                                             ; 0160
                        50       04   BC   9A 0003E 3$:          MOVZBL  @ADDRESS, SCRATCH                              ; 0151
                                      0A   11 00042              BRB     6$                                             ;
                        50       04   BC   3C 00044 4$:          MOVZWL  @ADDRESS, SCRATCH                              ; 0154
                                      04   11 00048              BRB     6$                                             ;
                        50       04   BC   D0 0004A 5$:          MOVL    @ADDRESS, SCRATCH                              ; 0157
```

```
        62      04   AC          1E            00  ED 0004E 6$:    CMPZV   #0, #30, ADDRESS, IO$GQ_PHYSICAL          ; 0163
                                                1E  1F 00054        BLSSU   8$                                        ;
                                                7E  7C 00056        CLRQ    -(SP)                                     ; 0170
                                         08     AE  9F 00058        PUSHAB  CH_STATUS                                 ;
                                     7E         07  7D 0005B        MOVQ    #7, -(SP)                                 ;
                                         0C     AC  DD 0005E        PUSHL   UNIT                                      ;
                                     63         06  FB 00061        CALLS   #6, DS$CHANNEL                            ;
                                     10         50  E9 00064        BLBC    RC, 9$                                    ;
                                     0F         6E  93 00067        BITB    CH_STATUS, #15                            ; 0172
                                                08  13 0006A        BEQL    8$                                        ;
                                     50 00660002 8F  D0 0006C 7$:    MOVL    #6684674, R0                              ;
                                                    04 00073        RET                                               ;
                                     50         01  D0 00074 8$:    MOVL    #1, R0                                    ; 0177
                                                    04 00077 9$:    RET                                               ;

; Routine Size:  120 bytes,    Routine Base:  SEP + 0000


;   0178  1
```

```
;    0180  1      %sbttl 'ACC$HANDLER'
;    0181  1
;    0182  1      global routine acc$handler (signal, mechanism) =
;    0183  1
;    0184  1      !++
;    0185  1      ! FUNCTIONAL DESCRIPTION:
;    0186  1      !
;    0187  1      !       This condition handler routine can be used to field any exceptions
;    0188  1      !       that indicate memory is in accessable.
;    0189  1      !       It does an SYS$UNWIND to the level of the caller of the routine
;    0190  1      !       that generated the condition.  The conditions trapped are
;    0191  1      !       DS$_MCHK, SS$_ACCVIO, and SS$_TRANSL.  The value of the aborted
;    0192  1      !       routine will be one of these values.
;    0193  1      !
;    0194  1      ! FORMAL PARAMETERS:
;    0195  1      !
;    0196  1      !       SIGNAL          Address of standard signal array
;    0197  1      !       MECHANISM       Address of mechanism array
;    0198  1      !
;    0199  1      ! IMPLICIT INPUTS:      NONE
;    0200  1      !
;    0201  1      ! IMPLICIT OUTPUTS:     NONE
;    0202  1      !
;    0203  1      ! CONDITION CODES:
;    0204  1      !
;    0205  1      !       SS$_RESIGNAL    If condition not DS$_MCHK, DS$_TRANSL, SS$_ACCVIO
;    0206  1      !       Otherwise       The value is irrelevent.
;    0207  1      !
;    0208  1      !--
;    0209  1
;    0210  2          begin
;    0211  2
;    0212  2          map
;    0213  2              signal : ref block [, byte],
;    0214  2              mechanism : ref block [, byte];
;    0215  2
;    0216  2          select .signal [chf$l_sig_name] of
;    0217  2              set
;    0218  2
;    0219  2              [ds$_mchk, ss$_accvio, ds$_transl] :
;    0220  3                  begin
;    0221  3                  mechanism [chf$l_mch_savr0] = .signal [chf$l_sig_name];
;    0222  4                  $unwind ()
;    0223  2                  end;
;    0224  2
;    0225  2              [otherwise] :
;    0226  2                  ss$_resignal;
;    0227  2              tes
;    0228  2
;    0229  1          end;


                                                        .EXTRN  SYS$UNWIND

                                        000C 00000      .ENTRY  ACC$HANDLER, Save R2,R3                 ; 0182
                        50      04  AC  D0 00002        MOVL    SIGNAL, R0                              ; 0216
```

```
                                50          04  C0 00006        ADDL2   #4, R0                          ;
                                52          60  D0 00009        MOVL    (R0), R2                        ;
                                53          01  D0 0000C        MOVL    #1, R3                          ;
                                0C          52  D1 0000F        CMPL    R2, #12                         ; 0219
                                            12  13 00012        BEQL    1$                              ;
                      00660088  8F          52  D1 00014        CMPL    R2, #6684808                    ;
                                            09  13 0001B        BEQL    1$                              ;
                      006600A0  8F          52  D1 0001D        CMPL    R2, #6684832                    ;
                                            16  12 00024        BNEQ    2$                              ;
                                            53  D4 00026 1$:    CLRL    R3                              ;
                                50      08  AC  D0 00028        MOVL    MECHANISM, R0                   ; 0221
                            0C  A0          52  D0 0002C        MOVL    R2, 12(R0)                      ;
                                            7E  7C 00030        CLRQ    -(SP)                           ; 0222
                      00000000G 00          02  FB 00032        CALLS   #2, SYS$UNWIND                  ;
                                51          50  D0 00039        MOVL    R0, R1                          ;
                                05          53  E9 0003C 2$:    BLBC    R3, 3$                          ; 0225
                                50    0918  8F  3C 0003F        MOVZWL  #2328, R0                       ;
                                                04 00044 3$:    RET                                     ; 0229

; Routine Size:  69 bytes,    Routine Base:  SEP + 0078


;   0230  1
;   0231  1     end
;   0232  1
;   0233  0     eludom



;                                   PSECT SUMMARY
;
;       Name                    Bytes                       Attributes
;
;  SEP                              189  NOVEC,  WRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...
$DS_CHANNEL	KeyWMacr	Lib01	143 170
$DS_CHS_DECL	Macro	Lib01	128
$DS_CHS_F	Fieldset	Lib01	128
$UNWIND	KeyWMacr	Lib03	222
ACC$HANDLER	GlobRout	182	63f 137a
ADDRESS	RoutForm	99	139. 151a. 154a. 157a. 163.
CHC$_CLEAR	Literal	Lib01	143
CHC$_STATUS	Literal	Lib01	170
CHF$L_MCH_SAVR0	Macro	Lib03	221
CHF$L_SIG_NAME	Macro	Lib03	216 221
CHS$V_ERRANY	Field	Lib01	172.
CH_STATUS	Stackloc	128	170a 172.
DS$CHANNEL	ExtRout	143	143c 170e 170c
DS$_ERROR	Literal	Lib01	160 172
DS$_MCHK	Literal	Lib01	219
DS$_TRANSL	Literal	Lib01	219
DSX$PROBE	GlobRout	99	62f
FP	Builtin	135	137a=
IO$GQ_PHYSICAL	External	83	139. 163.
LENGTH	RoutForm	99	147.
MECHANISM	RoutForm	182	214m 221a=
QUICK	Linkage	76	
RC	Local	131	143= 143.
	Local	168	170= 170.
SCRATCH	Local	132	151= 154= 157=
SDL$STARLET_CONCAT	Macro	Lib03	222
SIGNAL	RoutForm	182	213m 216a. 221a.
SS$_ACCVIO	Literal	Lib03	219
SS$_NORMAL	Literal	Lib03	176
SS$_RESIGNAL	Literal	Lib03	226
SYS$UNWIND	ExtRout	222	222c
UNIT	RoutForm	99	143. 170.

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]PROBE.B32;1
5	Module PROBE	
69	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
71	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
73	Library #3	SYS$COMMON:[SYSLIB]STARLET.L32;2
233	Eludom PROBE	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
a	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

; Library Statistics
;

; File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
; DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	40	4	96	00:00.0
; DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	0	0	47	00:00.0
; SYS$COMMON:[SYSLIB]STARLET.L32;2	10593	7	0	624	00:00.5

;
; COMMAND QUALIFIERS
;
; BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DSRW:/LIST=DSRW: DSRW:PROBE.B32

; Size: 189 code + 0 data bytes
; Run Time: 00:01.4
; Elapsed Time: 00:02.8
; Lines/CPU Min: 10204
; Lexemes/CPU-Min: 36262
; Memory Used: 68 pages
; Compilation Complete

Table of contents

(2) 57 DECLARATIONS
(3) 200 UDA50 Bootstrap device initialization
(4) 344 UDA50 Bootstrap driver QIO

```
0000    ; DEC/CMS REPLACEMENT HISTORY, Element PUBTDRIVR.MAR
0000    ; *2    11-NOV-1987 08:50:37 GEARIN "ADDED SUPPORT FOR LARGE DEVICE #"
0000    ; *1     6-FEB-1986 15:21:54 DS ""
0000    ; DEC/CMS REPLACEMENT HISTORY, Element PUBTDRIVR.MAR
0000            .TITLE  PUBTDRIVR - UDA50 BOOT DRIVER
0000            .IDENT  '10.10-2'                                                    ;G:2
0000
0000    ;
0000    ;*******************************************************************************
0000    ;*                                                                             *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982 BY                                          *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                     *
0000    ;*  ALL RIGHTS RESERVED.                                                       *
0000    ;*                                                                             *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED      *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE      *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER      *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY      *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY      *
0000    ;*  TRANSFERRED.                                                               *
0000    ;*                                                                             *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE      *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT      *
0000    ;*  CORPORATION.                                                               *
0000    ;*                                                                             *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS      *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
0000    ;*                                                                             *
0000    ;*                                                                             *
0000    ;*******************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:     BOOTS
0000    ;
0000    ; ABSTRACT:
0000    ;       This module contains the bootstrap device driver for the
0000    ;       UDA 50 disks.
0000    ;
0000    ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    ;
0000    ; AUTHOR:  Kerbey T. Altmann,   CREATION DATE:  20-Nov-1981
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ; [01] Dave Butenhof, 15-Apr-1982
0000    ;       Modify to run under VDS. Partic, change $BOOT_DRIVER macro
0000    ;       to exclude fields not supported by VDS, and have I/O entry
0000    ;       call init routine @ each call (since diagnostics may be
0000    ;       run, etc. between calls).
0000    ;                                                                             ;G:2
0000    ;       David Gearin    06-Nov-1987                                           ;G:2
0000    ;       Modify move from RPB$W_UNIT from byte to word.                        ;G:2
0000    ;       This allows for access to large device #'s.                           ;G:2
0000    ;--
```

```
                                        .SBTTL  DECLARATIONS
                                ;
                                ;       marcros
                                ;
                                
                                        .MACRO  $PRDEF,$GBL
                                
                                        $DEFINI PR,$GBL
                                
                                
                                $EQU    PR$_KSP 0
                                $EQU    PR$_ESP 1
                                $EQU    PR$_SSP 2
                                $EQU    PR$_USP 3
                                $EQU    PR$_ISP 4
                                $EQU    PR$_P0BR        8
                                $EQU    PR$_P0LR        9
                                $EQU    PR$_P1BR        10
                                $EQU    PR$_P1LR        11
                                $EQU    PR$_SBR 12
                                $EQU    PR$_SLR 13
                                $EQU    PR$_PCBB        16
                                $EQU    PR$_SCBB        17
                                $EQU    PR$_IPL 18
                                $EQU    PR$_ASTLVL      19
                                $EQU    PR$_SIRR        20
                                $EQU    PR$_SISR        21
                                $EQU    PR$_ICCS        24
                                $EQU    PR$_NICR        25
                                $EQU    PR$_ICR 26
                                $EQU    PR$_TODR        27
                                $EQU    PR$_RXCS        32
                                $EQU    PR$_RXDB        33
                                $EQU    PR$_TXCS        34
                                $EQU    PR$_TXDB        35
                                $EQU    PR$_ACCS        40
                                $EQU    PR$_ACCR        41
                                $EQU    PR$_MAPEN       56
                                $EQU    PR$_TBIA        57
                                $EQU    PR$_TBIS        58
                                $EQU    PR$_PME 61
                                $EQU    PR$_SID 62
                                $EQU    PR$_TBCHK       63
                                $EQU    PR$V_SID_SN     0
                                $EQU    PR$S_SID_SN     12
                                $EQU    PR$V_SID_PL     12
                                $EQU    PR$S_SID_PL     3
                                $EQU    PR$V_SID_ECO    15
                                $EQU    PR$S_SID_ECO    9
                                $EQU    PR$V_SID_TYPE   24
                                $EQU    PR$S_SID_TYPE   8
                                $EQU    PR$_SID_TYP780  1
                                $EQU    PR$_SID_TYP750  2
                                $EQU    PR$_SID_TYP730  3
                                $EQU    PR$_SID_TYP7VV  4
                                $EQU    PR$_SID_TYPMAX  4
                                $EQU    PR$_WCSA        44
```

```
0000    114  $EQU   PR$_WCSD        45
0000    115  $EQU   PR$_SBIFS       48
0000    116  $EQU   PR$_SBIS        49
0000    117  $EQU   PR$_SBISC       50
0000    118  $EQU   PR$_SBIMT       51
0000    119  $EQU   PR$_SBIER       52
0000    120  $EQU   PR$_SBITA       53
0000    121  $EQU   PR$_SBIQC       54
0000    122  $EQU   PR$_CMIERR      23
0000    123  $EQU   PR$_CSRS        28
0000    124  $EQU   PR$_CSRD        29
0000    125  $EQU   PR$_CSTS        30
0000    126  $EQU   PR$_CSTD        31
0000    127  $EQU   PR$_TBDR        36
0000    128  $EQU   PR$_CADR        37
0000    129  $EQU   PR$_MCESR       38
0000    130  $EQU   PR$_CAER        39
0000    131  $EQU   PR$_UBRESET     55
0000    132  $EQU   PR$_PAMACC      64
0000    133  $EQU   PR$_PAMLOC      65
0000    134  $EQU   PR$_CSWP        66
0000    135  $EQU   PR$_CRBT        67
0000    136  $EQU   PR$_MCTL1       68
0000    137  $EQU   PR$_MCTL2       69
0000    138  $EQU   PR$_MGEN        70
0000    139  $EQU   PR$_MTBER       71
0000    140  $EQU   PR$_MEAR        72
0000    141  $EQU   PR$_MDCR1       73
0000    142  $EQU   PR$_MEDR        74
0000    143  $EQU   PR$_MECCR       75
0000    144
0000    145          $DEFEND PR,$GBL,DEF
0000    146
0000    147          .ENDM   $PRDEF
0000    148 ;
0000    149 ; INCLUDE FILES:
0000    150 ;
0000    151
0000    152          $BTDDEF                         ; Boot device types
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    153          $IODEF                          ; I/O function codes
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    154          $MSCPDEF                        ; MSCP definitions
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    155          $PRDEF                          ; Processor registers
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    156          $PTEDEF                         ; Page table entries
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    157          $RPBDEF                         ; RPB offsets
0000                 .SAVE   LOCAL_BLOCK
0000                 .RESTORE
0000    158          $SSDEF                          ; Status codes
```

```
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .RESTORE
                     0000    159       $UBADEF                         ; UBA definitions
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .RESTORE
                     0000    160       $UBIDEF                         ; 11/750 UBA definitions
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .RESTORE
                     0000    161       $VADEF                          ; Virtual addresses
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .RESTORE
                     0000    162
                     0000    163
                     0000    164 ;
                     0000    165 ; EQUATED SYMBOLS:
                     0000    166 ;
                     0000    167
00000000             0000    168       UDAIP   = 0
00000002             0000    169       UDASA   = 2
00000001             0000    170       GO      = 1
00008000             0000    171       OWN     = 1a15
0000000B             0000    172       S1      = 11
0000000E             0000    173       S4      = 14
000001EE             0000    174       UMR     = 494                 ; Last-1 UNIBUS Mapping Register
                     0000    175
                     0000    176 ;
                     0000    177 ; OWN STORAGE:
                     0000    178 ;
                     0000    179
                     0000    180 ;
                     0000    181 ; Boot driver table entry
                     0000    182 ;
                     0000    183
                00000000     184 .Psect  Bootdrivr_2,    page    ; Define psect allocation
                     0000    185
                     0000    186       $BOOT_DRIVER    DEVTYPE = BTD$K_UDA,-   ; Device type (UDA50)
                     0000    187                       SIZE = UD_DRVSIZ,-      ; Driver size
                     0000    188                       ADDR = START,-          ; Driver starting address
                     0000    189                       ENTRY = UD_DRIVER       ; Driver entry point
                00000000             .PSECT  BOOTDRIVR_4
                    FFFF     0000    .WORD   -1
                    0011     0002    .WORD   BTD$K_UDA
            00000000         0004    .LONG   0
            000002E8'        0008    .LONG   UD_DRVSIZ
            00000000'        000C    .LONG   START-$TABLE
            00000280'        0010    .LONG   UD_DRIVER-START
            00000000'        0014    .LONG   -START
                00000000             .PSECT  BOOTDRIVR_2
                     0000    190 ;                   UNIT_INIT = UD_INIT,-   ; Driver unit init entry
                     0000    191 ;                   DRIVRNAME = DSKDRVNAME,-; Driver disk name
                     0000    192 ;                   AUXDRNAME = PRTDRVNAME  ; Driver port name
                     0000    193
                     0000    194 START:
                     0000    195 ;DSKDRVNAME:
                     0000    196 ;       .ASCIC  /DUDRIVER.EXE/          ; Disk class driver filename
                     0000    197 ;PRTDRVNAME:
                     0000    198 ;       .ASCIC  /PUDRIVER.EXE/          ; Port driver filename
```

```
                                0000   200          .SBTTL  UDA50 Bootstrap device initialization
                                0000   201
                                0000   202 ;++
                                0000   203 ;
                                0000   204 ; Inputs:
                                0000   205 ;
                                0000   206 ;       R9 -->  RPB
                                0000   207 ;
                                0000   208 ; Outputs:
                                0000   209 ;
                                0000   210 ;       R0 - status code
                                0000   211 ;
                                0000   212 ;--
                                0000   213
                          01FC  0000   214 .Entry UD_INIT, ^M<R2,R3,R4,R5,R6,R7,R8>
                                0002   215
                                0002   216          .ENABLE LSB
                                0002   217
                     50   38   DB  0002   218          MFPR    #PR$_MAPEN, R0          ; Get the mapping status
                                0005   219 ;        BLBS    R0,10$                  ; If virtual, skip some set up
                                0005   220 ;
                                0005   221 ; Set up the SYSTEMID for the local UDA.
                                0005   222 ;
                                0005   223 ;        MOVL    RPB$L_IOVEC(R9),R1      ; Point to iovec
                                0005   224 ;        CLRB    B^<BOO$GB_UMR_DP-BOO$AL_VECTOR>(R1) ; Set for Direct Data Path
                                0005   225 ;        INCL    VMB$L_FLAGS(AP)         ; Set a flag to load SCS code
                                0005   226 ;        MOVL    RPB$L_BOOTR2(R9),-
                                0005   227 ;                VMB$B_SYSTEMID(AP)      ; Low 32 bits is CSR of UDA
                                0005   228 ;        MOVW    RPB$L_BOOTR1(R9),-
                                0005   229 ;                VMB$B_SYSTEMID+4(AP)    ; Hi 16 bits is TR of UBA
                                0005   230 ;        BBSS    #47,VMB$B_SYSTEMID(AP),10$ ; Set bit 47
                                0005   231 ;
                                0005   232 ; Set up an interrupt vector.
                                0005   233 ;
                                0005   234 ;10$:   MOVW    #<127*4>,RPB$W_R0UBVEC(R9) ; Use the highest possible
                                0005   235 ;
                                0005   236 ; Set up a UNIBUS mapping register(s) to cover the ring and buffers.
                                0005   237 ; To make things easy, we will grab the last two register (494 & 495).
                                0005   238 ; These registers are necessary since the ring area will be accessed by
                                0005   239 ; the controller which is a UNIBUS device that does not do any mapping.
                                0005   240 ;
           56   0003DC00 8F    D0  0005   241          MOVL    #<UMRa9>, R6            ; Set up a constant
                52   0200'CF    9E  000C   242          MOVAB   W^INTTBL, R2            ; Get the address of the ring
     51    52   0000009C'EF    CB  0011   243          BICL3   BYTE_OFF, R2, R1        ; Get the byte offset in page
            02 A2    51   56    C9  0019   244          BISL3   R6, R1, 2(R2)           ; Set in the UNIBUS addr
                 02 A2   10'    C0  001E   245          ADDL    S^#<RING-INTTBL>,2(R2)  ; Make it the ring address
        52   52   15   09    EF  0022   246          EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R2 ; Get the page frame
                                0027   247          ASSUME  RPB$L_ADPVIR EQ RPB$L_ADPPHY+4
                53   5C A940    D0  0027   248          MOVL    RPB$L_ADPPHY(R9)[R0],R3 ; Get correct pointer to UBA reg
                                002C   249          ASSUME  RPB$L_CSRVIR EQ RPB$L_CSRPHY+4
                57   54 A940    D0  002C   250          MOVL    RPB$L_CSRPHY(R9)[R0],R7 ; Get correct address of device CSR
                       0C 50    E9  0031   251          BLBC    R0,20$                  ; If clr, then physical
                52   50 B942    D0  0034   252          MOVL    @RPB$L_SVASPT(R9)[R2],R2; Virtual, get physical
           52    FFE00000 8F    CA  0039   253          BICL    #^C<PTE$M_PFN>,R2       ; Now a physical PFN
                54   0FB8 C3    DE  0040   254 20$:     MOVAL   UBA$L_MAP+<UMR*4>(R3),R4; Get the last two UMR's
     84    52   000000E0'EF    C9  0045   255          BISL3   VALID,R2,(R4)+           ; Set as valid w/PFN
                         52    D6  004D   256          INCL    R2                      ; Set next page just in case
```

```
                64   52   000000E0'EF   C9  004F   257          BISL3   VALID,R2,(R4)           ; Set as valid w/PFN
                                        0057   258 ;
                                        0057   259 ; Now go thru the ridiculously complicated startup sequence.  This is a
                                        0057   260 ; fugue in four parts.
                                        0057   261 ;
                        58   02   D0     0057   262          MOVL    #2,R8                   ; Make two tries at this
                    53  0200'CF   9E     005A   263 RETRY:   MOVAB   W^INTTBL, R3
                        52   0B   D0     005F   264          MOVL    #S1, R2                 ; Step flag
                             67   B4     0062   265          CLRW    UDAIP(R7)               ; Poke the controller's CSR
                                        0064   266 ;
                                        0064   267 ; Wait 10 seconds.
                                        0064   268 ;
                                        0064   269 TIME:
                                        0064   270          TIMEDWAIT TIME=#1000*1000,-             ; wait for 10 seconds
                                        0064   271                    INS1    = <MOVW UDASA(R7),R4>,- ; check the setup register
                                        0064   272                    INS2    = <BLSS 25$>,-          ; bit 15 set ?
                                        0064   273                    INS3    = <BBS  R2,R4,25$>,-    ; done with this step ?
                                        0064   274                    DONELBL = 25$                   ; label for exiting wait loop
                        50   01   3C     0064   275          MOVZWL  #SS$_NORMAL,R0
51  00000000'GF    000F4240 8F   C5     0067   276          MULL3   #1000*1000,G^EXE$GL_TENUSEC,R1
                             7E   D4     0073   277          CLRL    -(SP)
                                        0075   278      30000$:
                         54   02 A7   B0     0075   279          MOVW UDASA(R7),R4
                              13   19     0079   280          BLSS    25$
                         0F 54   52   E0     007B   281          BBS     R2,R4,25$
                6E  00000000'GF   D0     007F   282          MOVL    G^EXE$GL_UBDELAY,(SP)
                           FD 6E   F5     0086   283      30001$: SOBGTR  (SP),30001$
                           E9 51   F5     0089   284          SOBGTR  R1,30000$
                              50   D4     008C   285          CLRL    R0
                                        008E   286      25$:
                              8E   D5     008E   287          TSTL    (SP)+
                           0D 50   E8     0090   288          BLBS    R0,26$                  ; br if not timed out
                                        0093   289
                           C4 58   F5     0093   290 ERROR:   SOBGTR  R8,RETRY                ; Try once again
                50    0054 8F   3C     0096   291          MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                              04   009B   292          RET
                                        009C   293 BYTE_OFF:
                      FFFFFE00          009C   294          .LONG   ^C<^X1FF>               ; Mask for byte offset in page
                                        00A0   295
                                        00A0   296 26$:
                              54   B5     00A0   297          TSTW    R4                      ; check status register for error
                              EF   19     00A2   298          BLSS    ERROR                   ; br if error
                                        00A4   299
                        02 A7   83   B0     00A4   300 30$:     MOVW    (R3)+,UDASA(R7)         ; Send the controller the next step
                        B8 52   0E   F3     00A8   301          AOBLEQ  #S4,R2,TIME             ; Set for next step
                                        00AC   302 ;
                                        00AC   303 ; Initialization complete.  Write the packet address in the ring.
                                        00AC   304 ;
                    51   0250'CF   9E     00AC   305          MOVAB   W^RSPPKT, R1            ; Get the address of response packet
                      51    E8 AF   CA     00B1   306          BICL    BYTE_OFF, R1
            0210'CF   51    56   C9     00B5   307          BISL3   R6, R1, W^RD            ; Set byte offset in ring desc
                    51   021C'CF   9E     00BB   308          MOVAB   W^CMDPKT, R1            ; Get the address of command packet
                         55   51   D0     00C0   309          MOVL    R1,R5                   ; Save pointer
                      51    D6 AF   CA     00C3   310          BICL    BYTE_OFF, R1
            0214'CF   51    56   C9     00C7   311          BISL3   R6, R1,W^CD             ; Set byte offset in ring desc
                                        00CD   312 ;
                                        00CD   313 ; Now bring the device on-line
```

```
                        00CD   ;
      85   01   D0      00CD           MOVL    #1,(R5)+                  ; Set command ref number
   85   64 A9   3C      00D0           MOVZWL  RPB$W_UNIT(R9),(R5)+      ; Put unit number in cmd packet field ;G:2
      85   09   9A      00D4           MOVZBL  #MSCP$K_OP_ONLIN,(R5)+    ; Set opcode to bring drive online
           85   7C      00D7           CLRQ    (R5)+                     ; Clear byte count, buff desc
           65   7C      00D9           CLRQ    (R5)                      ;  buff desc and LBN
         01CE   30      00DB           BSBW    IO                        ; Send it out
                04      00DE           RET
                        00DF
                        00DF
                        00DF           .DISABLE LSB
         000000E0       00DF           .=<.+1>&-2
         80000000       00E0   VALID:  .LONG   ^X80000000                ; SIgn bit set
                        00E4   Ds$Gb_Uda50_Init::                        ; Global label
                        00E4   Init:                                     ; convenient local label
                00      00E4           .Byte   0                         ; .. init flag
                        00E5   ;
                        00E5   ; RINGS
                        00E5   ;
                        00E5   ;
                        00E5
                        00E5   .Align  Page
                        0200
             8000       0200   INTTBL: .WORD   OWN                 ; Step 1 pattern
         00000000       0202           .LONG   0                   ; Step 2 & 3 pattern
             0001       0206           .WORD   GO
                        0208
         0000 0000      0208           .WORD   0,0                 ; Reserved
             0000       020C   CMDINT: .WORD   0                   ; Command status word
             0000       020E   RSPINT: .WORD   0                   ; Response status word
                        0210   RING:
         00000000       0210   RD:     .LONG   0                   ; UNIBUS address of response ring
         00000000       0214   CD:     .LONG   0                   ; UNIBUS address of command ring
                        0218   ;
             0030       0218           .WORD   48                  ; Length of message
             0001       021A           .WORD   1                   ; ID
             0001       021C   CMDPKT: .WORD   1
         0000024C       021E           .BLKW   23                  ; Full envelope
                        024C   ;
         00000250       024C           .BLKW   2
                        0250   UDA_Response::                      ; Global label for CONFIG
         00000280       0250   RSPPKT: .BLKW   24
```

```
                                0280   344          .SBTTL  UDA50 Bootstrap driver QIO
                                0280   345
                                0280   346 ;++
                                0280   347 ;
                                0280   348 ; Inputs:
                                0280   349 ;
                                0280   350 ;       R3      - base address of adapter's register space
                                0280   351 ;       R5      - lbn for current piece of transfer
                                0280   352 ;       R6      - contains 0
                                0280   353 ;       R7      - address of the device's CSR
                                0280   354 ;       R8      - size of transfer in bytes
                                0280   355 ;       R9      - address of the RPB
                                0280   356 ;       R10     - starting address of transfer (byte offset in first
                                0280   357 ;                 page ORed with starting map register number)
                                0280   358 ;       R11     - LBN at start of transfer
                                0280   359 ;
                                0280   360 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                                0280   361 ;       SIZE(AP)- Size of transfer in bytes
                                0280   362 ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
                                0280   363 ;
                                0280   364 ; Implicit inputs:
                                0280   365 ;
                                0280   366 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                                0280   367 ;
                                0280   368 ; Outputs:
                                0280   369 ;
                                0280   370 ;       R0 - status code
                                0280   371 ;
                                0280   372 ;               SS$_NORMAL      - successful transfer
                                0280   373 ;               SS$_NOSUCHDEV   - unsupported device
                                0280   374 ;               SS$_CTRLERR     - fatal controller error
                                0280   375 ;
                                0280   376 ;       R3 - must be preserved
                                0280   377 ;
                                0280   378 ;
                                0280   379 ;--
                                0280   380
                00000010        0280   381 FUNC = 16
                00000014        0280   382 MODE = 20
                                0280   383
                                0280   384 UD_DRIVER:                                ; UDA50 device driver.
                                0280   385
                                0280   386 ;
                                0280   387 ; Start out by re-initing the UDA if it hasn't been inited yet,
                                0280   388 ; or if there was an error previously.
                                0280   389 ;
                                0280   390
  0D FE60 CF    E8              0280   391          BlbS    W^Init, 10$             ; Branch if already set
FD76 CF    00   FB              0285   392          CallS   #0, Ud_Init             ; Call init code
     51 50    E9                028A   393          BlbC    R0, Error1              ; Exit if error
FE52 CF    01   90              028D   394          MovB    #1, W^Init              ; Set init byte
                                0292   395
                                0292   396 ;
                                0292   397 ; Translate the I/O function code into a device-dependent function
                                0292   398 ; code for this disk.
                                0292   399 ;
                                0292   400
```

```
                8E AF    21    90   0292   401 10$:    MOVB    #MSCP$K_OP_READ, -      ; Assume read
                                    0296   402                 CMDPKT+MSCP$B_OPCODE
                20   10 AC    D1   0296   403         CMPL    FUNC(AP),#IO$_WRITELBLK ; Check for write function
                           04    12   029A   404         BNEQ    20$                     ; No, do read
                84 AF    22    90   029C   405         MOVB    #MSCP$K_OP_WRITE, -     ; Set write function code
                                    02A0   406                 CMDPKT+MSCP$B_OPCODE
                94 AF    55    D0   02A0   407 20$:    MOVL    R5,CMDPKT+MSCP$L_LBN    ; Set the logical block number
                80 AF    58    D0   02A4   408         MOVL    R8,CMDPKT+MSCP$L_BYTE_CNT ; Set the byte count
                80 AF    5A    D0   02A8   409         MOVL    R10,CMDPKT+MSCP$B_BUFFER; Set the UNIBUS map register
                54   02 A7    B0   02AC   410 IO:     MOVW    UDASA(R7),R4            ; Controller offline?
                           2C    12   02B0   411         BNEQ    ERROR1                  ; Yep, error out
          FF5D CF    8000 8F    A8   02B2   412         BISW    #OWN, CD+2              ; Set controller ownership
          FF52 CF    8000 8F    A8   02B9   413         BISW    #OWN, RD+2              ; Ditto
                54    67    B0   02C0   414         MOVW    UDAIP(R7),R4            ; Tell controller to read
                54   02 A7    B0   02C3   415         MOVW    UDASA(R7),R4            ; Any problems?
                           15    12   02C7   416         BNEQ    ERROR1                  ; Yes
                54   02 A7    B0   02C9   417 30$:    MovW    UdaSa(R7), R4           ; Any problems?
                           0F    12   02CD   418         BNeq    Error1                  ; Yes
                   FF3F CF    B5   02CF   419         TSTW    RD+2                    ; Any response back?
                           F4    19   02D3   420         BLSS    30$                     ; No, spin until there is
                  82 AF    B5   02D5   421         TSTW    RSPPKT+MSCP$W_STATUS    ; Any drive errors?
                           04    12   02D8   422         BNEQ    ERROR1                  ; Yes
                                    02DA   423
                                    02DA   424 ;
                                    02DA   425 ; Transfer is complete. Return with success status code.
                                    02DA   426 ;
                50   01    3C   02DA   427         MOVZWL  #SS$_NORMAL,R0          ; SET COMPLETION CODE
                           05    02DD   428         RSB                             ; AND RETURN
                                    02DE   429 ;
                                    02DE   430 ; Error occured during transfer.  Return and retry.
                                    02DE   431 ;
                                    02DE   432
                                    02DE   433 ERROR1:
          FE02 CF    94   02DE   434         ClrB    W^Init                  ; Must re-init next time
50   0054 8F    3C   02E2   435         MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                           05    02E7   436         RSB                             ; Return to BOOTDRIVR
                                    02E8   437
                000002E8    02E8   438 UD_DRVSIZ=.-START
                                    02E8   439
                                    02E8   440         .END
```

```
$TABLE                          = 00000000 R  D  03
BTD$K_UDA                       = 00000011    D
BYTE_OFF                          0000009C R  D  02
CD                                00000214 R  D  02
CMDINT                            0000020C R  D  02
CMDPKT                            0000021C R  D  02
DS$GB_UDA50_INIT                  000000E4 RG D  02
ERROR                             00000093 R  D  02
ERROR1                            000002DE R  D  02
EXE$GL_TENUSEC                    ******** X    02
EXE$GL_UBDELAY                    ******** X    02
FUNC                            = 00000010    D
GO                              = 00000001    D
INIT                              000000E4 R  D  02
INTTBL                            00000200 R  D  02
IO                                000002AC R  D  02
IO$_WRITELBLK                   = 00000020    D
MODE                            = 00000014    D
MSCP$B_BUFFER                   = 00000010    D
MSCP$B_OPCODE                   = 00000008    D
MSCP$K_OP_ONLIN                 = 00000009    D
MSCP$K_OP_READ                  = 00000021    D
MSCP$K_OP_WRITE                 = 00000022    D
MSCP$L_BYTE_CNT                 = 0000000C    D
MSCP$L_LBN                      = 0000001C    D
MSCP$W_STATUS                   = 0000000A    D
OWN                             = 00008000    D
PR$_MAPEN                       = 00000038    D
PTE$M_PFN                       = 001FFFFF    D
RD                                00000210 R  D  02
RETRY                             0000005A R  D  02
RING                              00000210 R  D  02
RPB$L_ADPPHY                    = 0000005C    D
RPB$L_ADPVIR                    = 00000060    D
RPB$L_CSRPHY                    = 00000054    D
RPB$L_CSRVIR                    = 00000058    D
RPB$L_SVASPT                    = 00000050    D
RPB$W_UNIT                      = 00000064    D
RSPINT                            0000020E R  D  02
RSPPKT                            00000250 R  D  02
S1                              = 0000000B    D
S4                              = 0000000E    D
SS$_CTRLERR                     = 00000054    D
SS$_NORMAL                      = 00000001    D
START                             00000000 R  D  02
TIME                              00000064 R  D  02
UBA$L_MAP                       = 00000800    D
UDAIP                           = 00000000    D
UDASA                           = 00000002    D
UDA_RESPONSE                      00000250 RG D  02
UD_DRIVER                         00000280 R  D  02
UD_DRVSIZ                       = 000002E8    D
UD_INIT                           00000000 RG D  02
UMR                             = 000001EE    D
VA$S_VPN                        = 00000015    D
VA$V_VPN                        = 00000009    D
VALID                             000000E0 R  D  02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	000002E8 (744.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
BOOTDRIVR_4	00000018 (24.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	189 (2)	189 (2)
BTD$K_UDA	=00000011		189 (2)
BYTE_OFF	0000009C-R	280 (3)	#-243 (3) #-293 (3) #-297 (3)
CD	00000214-R	333 (3)	#-298 (3) #-412 (4)
CMDINT	0000020C-R	329 (3)	
CMDPKT	0000021C-R	337 (3)	295 (3) #-402 (4) #-406 (4) #-407 (4)
			#-408 (4) #-409 (4)
DS$GB_UDA50_INIT	000000E4-R	314 (3)	
ERROR	00000093-R	277 (3)	#-285 (3)
ERROR1	000002DE-R	433 (4)	#-393 (4) #-411 (4) #-416 (4) #-418 (4)
			#-422 (4)
EXE$GL_TENUSEC	00000000-XR		#-274 (3)
EXE$GL_UBDELAY	00000000-XR		#-274 (3)
FUNC	=00000010	381 (4)	#-403 (4)
GO	=00000001	170 (2)	326 (3)
INIT	000000E4-R	315 (3)	#-391 (4) #-394 (4) #-434 (4)
INTTBL	00000200-R	324 (3)	242 (3) #-245 (3) 263 (3)
IO	000002AC-R	410 (4)	#-307 (3)
IO$_WRITELBLK	=00000020		#-403 (4)
MODE	=00000014	382 (4)	
MSCP$B_BUFFER	=00000010		#-409 (4)
MSCP$B_OPCODE	=00000008		#-402 (4) #-406 (4)
MSCP$K_OP_ONLIN	=00000009		#-304 (3)
MSCP$K_OP_READ	=00000021		#-401 (4)
MSCP$K_OP_WRITE	=00000022		#-405 (4)
MSCP$L_BYTE_CNT	=0000000C		#-408 (4)
MSCP$L_LBN	=0000001C		#-407 (4)
MSCP$W_STATUS	=0000000A		#-421 (4)
OWN	=00008000	171 (2)	324 (3) #-412 (4) #-413 (4)
PR$_MAPEN	=00000038		#-218 (3)
PTE$M_PFN	=001FFFFF		#-253 (3)
RD	00000210-R	332 (3)	#-294 (3) #-413 (4) #-419 (4)
RETRY	0000005A-R	263 (3)	#-277 (3)
RING	00000210-R	331 (3)	#-245 (3)
RPB$L_ADPPHY	=0000005C		247 (3) #-248 (3)
RPB$L_ADPVIR	=00000060		247 (3)
RPB$L_CSRPHY	=00000054		249 (3) #-250 (3)
RPB$L_CSRVIR	=00000058		249 (3)
RPB$L_SVASPT	=00000050		#-252 (3)
RPB$W_UNIT	=00000064		#-303 (3)
RSPINT	0000020E-R	330 (3)	
RSPPKT	00000250-R	342 (3)	292 (3) #-421 (4)
S1	=0000000B	172 (2)	#-264 (3)
S4	=0000000E	173 (2)	#-288 (3)
SS$_CTRLERR	=00000054		#-278 (3) #-435 (4)
SS$_NORMAL	=00000001		#-274 (3) #-427 (4)
START	00000000-R	194 (2)	189 (2) 438 (4)
TIME	00000064-R	269 (3)	#-288 (3)
UBA$L_MAP	=00000800		254 (3)
UDAIP	=00000000	168 (2)	#-265 (3) #-414 (4)

```
UDASA                           =00000002     169    (2)    #-274    (3)    #-287    (3)    #-410    (4)    #-415    (4)
                                                            #-417    (4)
UDA_RESPONSE                     00000250-R   341    (3)
UD_DRIVER                        00000280-R   384    (4)      189    (2)
UD_DRVSIZ                       =000002E8     438    (4)      189    (2)
UD_INIT                          00000000-R   214    (3)      392    (4)
UMR                             =000001EE     174    (2)    #-241    (3)      254    (3)
VA$S_VPN                        =00000015                   #-246    (3)
VA$V_VPN                        =00000009                   #-246    (3)
VALID                            000000E0-R   313    (3)    #-255    (3)    #-257    (3)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO           SIZE        DEFINITION      REFERENCES...
-----           ----        ----------      -------------
$BOOT_DRIVER    1           186     (2)     186     (2)
$BTDDEF         2           152     (2)     152     (2)
$DEFINI         1           152     (2)     152     (2)     153     (2)     154     (2)     155     (2)     156     (2)
                                            157     (2)     158     (2)     159     (2)     160     (2)     161     (2)
$IODEF          15          153     (2)     153     (2)
$MSCPDEF        25          154     (2)     154     (2)
$PRDEF          4           62      (2)     155     (2)
$PTEDEF         3           156     (2)     156     (2)
$RPBDEF         5           157     (2)     157     (2)
$SSDEF          21          158     (2)     158     (2)
$UBADEF         6           159     (2)     159     (2)
$UBIDEF         2           160     (2)     160     (2)
$VADEF          1           161     (2)     161     (2)
ASSUME          1           247     (3)     247     (3)     249     (3)
TIMEDWAIT       1           270     (3)     270     (3)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	245 (3)
AOBLEQ	00F3	288 (3)
BBS	00E0	274 (3)
BICL	00CA	253 (3) 293 (3) 297 (3)
BICL3	00CB	243 (3)
BISL3	00C9	244 (3) 255 (3) 257 (3) 294 (3) 298 (3)
BISW	00A8	412 (4) 413 (4)
BLBC	00E9	251 (3) 393 (4)
BLBS	00E8	275 (3) 391 (4)
BLSS	0019	274 (3) 285 (3) 420 (4)
BNEQ	0012	404 (4) 411 (4) 416 (4) 418 (4) 422 (4)
BSBW	0030	307 (3)
CALLS	00FB	392 (4)
CLRB	0094	434 (4)
CLRL	00D4	274 (3)
CLRQ	007C	305 (3) 306 (3)
CLRW	00B4	265 (3)
CMPL	00D1	403 (4)
EXTZV	00EF	246 (3)
INCL	00D6	256 (3)
MFPR	00DB	218 (3)
MOVAB	009E	242 (3) 263 (3) 292 (3) 295 (3)
MOVAL	00DE	254 (3)
MOVB	0090	394 (4) 401 (4) 405 (4)
MOVL	00D0	241 (3) 248 (3) 250 (3) 252 (3) 262 (3) 264 (3) 274 (3)
		296 (3) 302 (3) 407 (4) 408 (4) 409 (4)
MOVW	00B0	274 (3) 287 (3) 410 (4) 414 (4) 415 (4) 417 (4)
MOVZBL	009A	304 (3)
MOVZWL	003C	274 (3) 278 (3) 303 (3) 427 (4) 435 (4)
MULL3	00C5	274 (3)
RET	0004	279 (3) 308 (3)
RSB	0005	428 (4) 436 (4)
SOBGTR	00F5	274 (3) 277 (3)
TSTL	00D5	274 (3)
TSTW	00B5	284 (3) 419 (4) 421 (4)

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ALIGN	322	(3)														
.BLKW	338	(3)	340	(3)	342	(3)										
.BYTE	316	(3)														
.DISABLE	311	(3)														
.ENABLE	216	(3)														
.END	440	(4)														
.ENDC	189	(2)	247	(3)	249	(3)	274	(3)								
.ENDM	147	(2)	152	(2)	153	(2)	154	(2)	156	(2)	157	(2)	158	(2)	159	(2)
	160	(2)	161	(2)	186	(2)	247	(3)	270	(3)						
.ENTRY	214	(3)														
.IDENT	6	(1)														
.IF	189	(2)	247	(3)	249	(3)	274	(3)								
.IFF	189	(2)	247	(3)	249	(3)										
.LONG	189	(2)	281	(3)	313	(3)	325	(3)	332	(3)	333	(3)				
.MACRO	62	(2)														
.NLIST	274	(3)														
.NOCROSS	152	(2)	153	(2)	154	(2)	155	(2)	156	(2)	157	(2)	158	(2)	159	(2)
	160	(2)	161	(2)												
.PSECT	184	(2)	189	(2)												
.RESTORE	152	(2)	153	(2)	154	(2)	155	(2)	156	(2)	157	(2)	158	(2)	159	(2)
	160	(2)	161	(2)												
.SAVE	152	(2)	153	(2)	154	(2)	155	(2)	156	(2)	157	(2)	158	(2)	159	(2)
	160	(2)	161	(2)												
.SBTTL	200	(3)	344	(4)	57	(2)										
.TITLE	5	(1)														
.WORD	189	(2)	324	(3)	326	(3)	328	(3)	329	(3)	330	(3)	335	(3)	336	(3)
	337	(3)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...										
AP	1	#-403	(4)									
R0	11	#-218	(3)	#-248	(3)	#-250	(3)	#-251	(3)	#-274	(3)	#-275	(3)
		#-278	(3)	#-393	(4)	#-427	(4)	#-435	(4)				
R1	11	#-243	(3)	#-244	(3)	#-274	(3)	#-292	(3)	#-293	(3)	#-294	(3)
		#-295	(3)	#-296	(3)	#-297	(3)	#-298	(3)				
R10	1	#-409	(4)										
R2	16	214	(3)	#-242	(3)	#-243	(3)	#-244	(3)	#-245	(3)	246	(3)
		#-252	(3)	#-253	(3)	#-255	(3)	#-256	(3)	#-257	(3)	#-264	(3)
		#-274	(3)	#-288	(3)								
R3	5	214	(3)	#-248	(3)	254	(3)	#-263	(3)	#-287	(3)		
R4	11	214	(3)	#-254	(3)	#-255	(3)	#-257	(3)	#-274	(3)	#-284	(3)
		#-410	(4)	#-414	(4)	#-415	(4)	#-417	(4)				
R5	8	214	(3)	#-296	(3)	#-302	(3)	#-303	(3)	#-304	(3)	#-305	(3)
		#-306	(3)	#-407	(4)								
R6	5	214	(3)	#-241	(3)	#-244	(3)	#-294	(3)	#-298	(3)		
R7	9	214	(3)	#-250	(3)	#-265	(3)	#-274	(3)	#-287	(3)	#-410	(4)
		#-414	(4)	#-415	(4)	#-417	(4)						
R8	4	214	(3)	#-262	(3)	#-277	(3)	#-408	(4)				
R9	4	#-248	(3)	#-250	(3)	#-252	(3)	#-303	(3)				
SP	4	#-274	(3)										

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.21
Command processing	875	00:00:00.20	00:00:00.66
Pass 1	682	00:00:03.52	00:00:04.78
Symbol table sort	0	00:00:00.47	00:00:00.46
Pass 2	8	00:00:00.84	00:00:01.50
Symbol table output	0	00:00:00.01	00:00:00.06
Psect synopsis output	0	00:00:00.02	00:00:00.17
Cross-reference output	2	00:00:00.11	00:00:00.14
Assembler run totals	1589	00:00:05.20	00:00:07.98

The working set limit was 3400 pages.
169684 bytes (332 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1676 non-local and 9 local symbols.
440 source lines were read in Pass 1, producing 109 object records in Pass 2.
57 pages of virtual memory were used to define 19 macros.

```
                                    +---------------------------+
                                    ! Macro library statistics !
                                    +---------------------------+

Macro library name                   Macros defined
------------------                   --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5          0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6            1
SYS$COMMON:[SYSLIB]LIB.MLB;2                       8
SYS$COMMON:[SYSLIB]STARLET.MLB;2                   6
TOTALS (all libraries)                            15

1715 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:PUBTDRIVR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element QACHECKS.B32
;    0002  0      ! *2    19-OCT-1987 17:32:57 LI "Modified loop-on-subtest"
;    0003  0      ! *1     6-FEB-1986 15:21:56 DS ""
;    0004  0      ! DEC/CMS REPLACEMENT HISTORY, Element QACHECKS.B32
;    0005  0      %TITLE '*** QA Check Routines'
;    0006  0      MODULE QACHECKS (
;    0007  0                ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE),
;    0008  0                IDENT = '10.10-02'                                              !G:2
;    0009  0                ) =
;    0010  0      !
;    0011  0      ! COPYRIGHT (c) 1979, 1981, 1983
;    0012  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0013  0      !
;    0014  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0015  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0016  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0017  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0018  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0019  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0020  0      ! REMAIN IN DEC.
;    0021  0      !
;    0022  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0023  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0024  0      ! CORPORATION.
;    0025  0      !
;    0026  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0027  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0028  0
;    0029  0      !++
;    0030  0      ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0031  0      !
;    0032  0      ! ABSTRACT:
;    0033  0      !
;    0034  0      !       This module contains the check routines required to perform QA on diagnostic programs.
;    0035  0      !
;    0036  0      ! AUTHOR:
;    0037  0      !
;    0038  0      !       Jack Stansbury
;    0039  0      !
;    0040  0      ! CREATION DATE:
;    0041  0      !
;    0042  0      !               15-October-1981, Version 6.5-00
;    0043  0      !       00      Put in code for all the support routines plus the following check routines: QA$Normal_Start,
;    0044  0      !               QA$Multiple_Pass, QA$Loop_On_Test, QA$Run_Backwards.
;    0045  0      !
;    0046  0      ! MODIFIED:
;    0047  0      !
;    0048  0      !       01      14-December-1981, Jack Stansbury, Version 6.5-01
;    0049  0      !               Added remaining QA check routines. Separated the check routines (into QACHECKS.B32) and the
;    0050  0      !               support routines (QAMAIN.B32) because the QA.B32 module was getting too large!
;    0051  0      !
;    0052  0      !       02      19-Jan-1983, Jack Stansbury, Version 6.10 (?)
;    0053  0      !               Took out the Error Phase One check because it doesn't work properly with
;    0054  0      !               issuing error messages from subroutines in the diagnostic program.
;    0055  0      !
;    0056  0      !       03      Bob Bergazzi    May 17, 1983    Version 6.11
;    0057  0      !               Changed the order of searching libraries.
```

```
;   0058  0     !                                                                                    !G:2
;   0059  0     !             Daniel Li       Oct 19, 1987    Version 10.10                          !G:2
;   0060  0     !             Modified QA$Loop_On_Subtest to reset ds$gl_subtest to 0 when last test was done.  !G:2
;   0061  0     !                                                                                    !G:2
;   0062  0     !--
```

```
;    0064  0     %SBTTL  'Table of Contents'
;    0065  1     BEGIN
;    0066  1     !
;    0067  1     ! TABLE OF CONTENTS:
;    0068  1     !
;    0069  1
;    0070  1     FORWARD ROUTINE
;    0071  1             QA$Normal_Start         :               ADDRESSING_MODE (LONG_RELATIVE),
;    0072  1             QA$Multiple_Pass        :               ADDRESSING_MODE (LONG_RELATIVE),
;    0073  1             QA$Loop_On_Test         :               ADDRESSING_MODE (LONG_RELATIVE),
;    0074  1             QA$Loop_On_Subtest      :               ADDRESSING_MODE (LONG_RELATIVE),        !                       [01]
;    0075  1             QA$Run_Backwards        :               ADDRESSING_MODE (LONG_RELATIVE);        !                       [02]
```

```
;    0077  1     %SBTTL  'Libraries'
;    0078  1     !
;    0079  1     ! LIBRARIES:
;    0080  1     !
;    0081  1
;    0082  1     LIBRARY
;    0083  1             '$DIAG';                !                                   [03]
;    0084  1
;    0085  1     LIBRARY
;    0086  1             '$DS';                  !                                   [03]
;    0087  1
;    0088  1     LIBRARY
;    0089  1             'SYS$LIBRARY:LIB';      !                                   [03]
```

```
;    0091  1     %SBTTL  'Included Macros and Literals'
;    0092  1     !
;    0093  1     ! INCLUDED MACROS AND LITERALS
;    0094  1     !
;    0095  1
;    0096  1             $DS_QADEF;                      ! Define the branch codes.
;    0097  1             $DS_DSADEF;                     ! Define the DSA flags.
;    0098  1             DSQA;                           ! Define the DSQA$K_ constants for the routine names.
;    0099  1             DS_QADEFS;                      ! Define the common QA definitions like QA$K_.          [1]
```

```
;    0101  1      %SBTTL  'External Routines'
;    0102  1      !
;    0103  1      ! EXTERNAL ROUTINE REFERENCES:
;    0104  1      !
;    0105  1
;    0106  1      EXTERNAL ROUTINE
;    0107  1              QA$Find_User_Call_Frame :          ADDRESSING_MODE (LONG_RELATIVE),      ! Find the user's last frame.   [01]
;    0108  1              QA$Print_Forced_Errors  : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),      ! Print the Forced-Error table. [01]
;    0109  1              QA$Add_Forced_Error     : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),      ! Add a forced error.           [01]
;    0110  1              QA$Add_QA_Error         : NOVALUE ADDRESSING_MODE (LONG_RELATIVE),      ! Add a QA error.               [01]
;    0111  1
;    0112  1              DSX$Abort;
```

```
;    0114  1     %SBTTL  'External Own Storage'
;    0115  1     !
;    0116  1     ! EXTERNAL OWN STORAGE
;    0117  1     !
;    0118  1
;    0119  1     EXTERNAL
;    0120  1             DS$GL_FSTTEST,                  ! First test to execute
;    0121  1             DS$GL_LSTTEST,                  ! Last test to execute
;    0122  1             DS$GL_SUBTEST,                  ! The subtest to loop on                    [01]
;    0123  1
;    0124  1             DS$GL_CLIBASE,                  ! Base of CLI data table
;    0125  1
;    0126  1             QA$W_TestLoops    : WORD,       ! These are defined in the QAFLAGS module.  [02]
;    0127  1             QA$W_SubtestLoops : WORD,       ! They contain the current settings of the
;    0128  1             QA$W_MultiplePass : WORD,       ! QA flags.
;    0129  1
;    0130  1             !+                                                                          [1]
;    0131  1             ! The following are defined in the QAMAIN.B32 module.                       [1]
;    0132  1             !-                                                                          [1]
;    0133  1
;    0134  1             QA$AOB_Routine_State : BITVECTOR,   !                                       [1]
;    0135  1             QA$AOB_Check_State   : BITVECTOR,   !                                       [1]
;    0136  1             QA$AOB_Flags         : BITVECTOR,   !                                       [1]
;    0137  1                                                                                 !G:2
;    0138  1             QA$W_Branch       : WORD,       !                                           [1]
;    0139  1             QA$W_Error_Number : WORD,       !                                           [1]
;    0140  1             QA$W_Save_Test    : SIGNED WORD,! The saved starting test number.           [1]
;    0141  1             QA$W_Save_Last    : SIGNED WORD;! The saved ending test number.             [1]
```

```
;    0143  1      %SBTTL  'Psect Definitions'
;    0144  1      !
;    0145  1      ! PSECT DEFINITIONS:
;    0146  1      !
;    0147  1
;    0148  1      PSECT
;    0149  1              Plit   = Data (NoExecute, Share, NoWrite,
;    0150  1                             Addressing_Mode (Long_Relative));
;    0151  1
;    0152  1      PSECT
;    0153  1              Code   = Code (Execute, Share, NoWrite,
;    0154  1                             Addressing_Mode (Long_Relative));
;    0155  1
;    0156  1      PSECT
;    0157  1              Own    = Work (NoExecute, NoShare, Write,
;    0158  1                             Addressing_Mode (Long_Relative));
;    0159  1
;    0160  1      PSECT
;    0161  1              Global = Work (NoExecute, NoShare, Write,
;    0162  1                             Addressing_Mode (Long_Relative));
```

```
;    0164  1     %SBTTL  'Local Macro Definitions'
;    0165  1     !
;    0166  1     ! LOCAL MACRO DEFINITIONS:
;    0167  1     !
;    0168  1
;    0169  1     MACRO
;    0170  1             !+
;    0171  1             ! Usage:
;    0172  1             !       ASSERT ((.A GTR .B), 'A <= B');
;    0173  1             !
;    0174  1             ! If the specified condition is not true, a call to the Logic_Error macro is made with the specified
;    0175  1             ! message string. ASSERT is turned on whenever the compilation is made with the /VARIANT:n qualifier,
;    0176  1             ! where n <> 0.
;    0177  1             !-
;    0178  1
; M  0179  1             ASSERT (Condition, Assert_String) =
; M  0180  1                     %IF %VARIANT
; M  0181  1                     %THEN
; M  0182  1                             (
; M  0183  1                             IF (NOT (Condition)) THEN
; M  0184  1                                     Logic_Error (Assert_String)
; M  0185  1                             )
; M  0186  1                     %ELSE
; M  0187  1                             ! No code put in.
; M  0188  1                     %FI
;    0189  1             %;
;    0190  1
;    0191  1     MACRO
;    0192  1             !+
;    0193  1             ! Usage:
;    0194  1             !       Logic_Error ('String to be printed out');
;    0195  1             !
;    0196  1             ! This macro is used whenever a logic error is detected. It is called from ASSERT whenever the assertion
;    0197  1             ! is false. This can be called anywhere else where an error is known to exist and and expression does not
;    0198  1             ! have to be evaluated to determine whether or not an error exists.
;    0199  1             !-
;    0200  1
; M  0201  1             Logic_Error (Error_String) =
; M  0202  1                     (
; M  0203  1                     BIND
; M  0204  1                             ES = $ASCIC ('!/**QA Internal Error: ', Error_String, '!/');    !                    [1]
; M  0205  1
; M  0206  1                     $DS_PRINTF (ES);
; M  0207  1                     )
;    0208  1             %;
;    0209  1
```

```
;    0211  1      MACRO
;    0212  1              !+
;    0213  1              ! This macro is used only in the debugging stages. It's only parameter is a string giving, e.g., the
;    0214  1              ! name of a routine. DEBUG is turned on whenever the compilation is made with the /VARIANT:n qualifier,
;    0215  1              ! where n <> 0.
;    0216  1              !-
;    0217  1
; M  0218  1              DEBUG (Debug_String) =
; M  0219  1                      %IF %VARIANT
; M  0220  1                      %THEN
; M  0221  1                              (
; M  0222  1                              IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN
; M  0223  1                                      $DS_PRINTF ($ASCIC ('!/', Debug_String, '!/'))
; M  0224  1                              )
; M  0225  1                      %ELSE
; M  0226  1                              ! No code put in.
; M  0227  1                      %FI
;    0228  1              %;
;    0229  1
;    0230  1      MACRO
;    0231  1              !+
;    0232  1              ! Define a macro that takes an expression as an argument. The value of this macro is either
;    0233  1              ! one or zero. All bits but the least significant one are cleared.
;    0234  1              !-
;    0235  1
; M  0236  1              Boolean (Expression) =
;    0237  1                      ((Expression) AND 1) %;
```

```
;    0239  1      %SBTTL  'Module-Global Static Storage'
;    0240  1
;    0241  1      !
;    0242  1      ! MODULE-GLOBAL DECLARATIONS:
;    0243  1      !
;    0244  1
;    0245  1              MODNAM ('QACHECKS');                    ! Define module name                    [1]
```

```
%SBTTL  'ASCIC Bindings'

!+
! These are the error messages output by the check routines when they find a QA error.
!-

GLOBAL BIND                                                     !                                                       [01]
        QA$T_Operator_Request_Message =                         !                                                       [01]
                $ASCIC ('!/**QA ERROR** Program requested operator intervention.!/');

BIND                                                            !                                                       [01]
        Cleanup_Error_Message =
                $ASCIC ('!/**QA ERROR** Error reported in cleanup code!/'),

        Error_Reported_Message =
                $ASCIC ('!/**QA ERROR** Error reported!/'),

        !+
        ! One of these messages in printed AFTER one of the above three is printed.                                   [02]
        !-

        Normal_Start_Message =
                $ASCIC ('**QA ERROR** Normal start check failed!/'),                    !                               [02]

        Multiple_Pass_Message =
                $ASCIC ('**QA ERROR** Multiple pass check failed!/'),                   !                               [02]

        Infinite_Test_Message =
                $ASCIC ('**QA ERROR** Infinite loop-on-test check failed!/'),           !                               [02]

        Infinite_Subtest_Message =                                                      !                               [01]
                $ASCIC ('**QA ERROR** Infinite loop-on-subtest check failed!/'),        !                               [02]

        Run_Backwards_Message =
                $ASCIC ('**QA ERROR** Run individual tests in reverse order check failed!/');   !                       [02]
```

```
;    0283  1      !+
;    0284  1      ! These are the header lines that each check routine will print in its own initialization code section.
;    0285  1      ! The first three are defined globally because QA$Dump uses them.
;    0286  1      !-
;    0287  1
;    0288  1      GLOBAL BIND
;    0289  1              QA$T_Header_1 =
;    0290  1                      $ASCIC ('!^!/!/!25** QUALITY ASSURANCE !26**!/'),
;    0291  1
;    0292  1              QA$T_Header_2 =
;    0293  1                      $ASCIC ('!10* !38<PROGRAM: !AC!> REV: !UL.!UL!/'),
;    0294  1
;    0295  1              QA$T_Header_3 =
;    0296  1                      $ASCIC ('!23* !%D!/!/');
```

```
;   0298  1     BIND
;   0299  1
;   0300  1             Header_NS =
;   0301  1                     $ASCIC ('!28* Normal Start!/!/'),
;   0302  1
;   0303  1             Header_MP =
;   0304  1                     $ASCIC ('!28* Multiple Pass!/!/'),
;   0305  1
;   0306  1             Header_LT =
;   0307  1                     $ASCIC ('!24* Infinite Loop-On-Test!/!/'),
;   0308  1
;   0309  1             Header_LS =                                     !                                   [01]
;   0310  1                     $ASCIC ('!22* Infinite Loop-On-Subtest!/!/'),   !                           [01]
;   0311  1
;   0312  1             Header_RB =
;   0313  1                     $ASCIC ('!25* Run Tests Backwards!/!/');        !                           [02]
```

```
;    0315  1      %SBTTL 'QA$Normal_Start'
;    0316  1
;    0317  1      GLOBAL ROUTINE QA$Normal_Start (Hook_Point) =
;    0318  1
;    0319  1      !++
;    0320  1      ! FUNCTIONAL DESCRIPTION:
;    0321  1      !
;    0322  1      !       This routine performs the Normal Start QA check.
;    0323  1      !
;    0324  1      ! CALLER:
;    0325  1      !
;    0326  1      !       QA$Main
;    0327  1      !
;    0328  1      ! FORMAL PARAMETERS:
;    0329  1      !
;    0330  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0331  1      !                    The constants are defined in the DSQA macro.
;    0332  1      !
;    0333  1      ! IMPLICIT INPUTS:
;    0334  1      !
;    0335  1      !       DSA$GL_FLAGS longword.
;    0336  1      !       QA$AOB_Check_State bitvector.
;    0337  1      !
;    0338  1      ! IMPLICIT OUTPUTS:
;    0339  1      !
;    0340  1      !       NONE
;    0341  1      !
;    0342  1      ! COMPLETION CODES:
;    0343  1      !
;    0344  1      !       0 - QA error detected; unsuccessful return
;    0345  1      !       1 - No QA errors; successful return
;    0346  1      !
;    0347  1      ! SIDE EFFECTS:
;    0348  1      !
;    0349  1      !       NONE
;    0350  1      !
;    0351  1      !--
```

```
;    0353  2     BEGIN
;    0354  2             SELECTONE .Hook_Point OF
;    0355  2                     SET
;    0356  2                     [DSQA$K_Start_VRReStart]:
;    0357  3                             BEGIN
;    0358  3                             $DS_PRINTF (QA$T_Header_1);
;    0359  3                             $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0360  3                             $DS_PRINTF (QA$T_Header_3, 0);  ! 0 is for the current time
;    0361  3                             $DS_PRINTF (Header_NS);
;    0362  3
;    0363  3                             QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0364  4                             RETURN (QA$K_Success)
;    0365  2                             END;
;    0366  2
;    0367  2                     [DSQA$K_Error_Error_End]:
;    0368  3                             BEGIN
;    0369  3                             IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN
;    0370  4                                     $DS_PRINTF (Cleanup_Error_Message)
;    0371  3                             ELSE
;    0372  3                                     $DS_PRINTF (Error_Reported_Message);
;    0373  3
;    0374  3                             $DS_PRINTF (Normal_Start_Message);              !                                       [02]
;    0375  3
;    0376  3                             QA$Add_QA_Error (QA$K_Normal_Start);
;    0377  3                             QA$AOB_Routine_State [QA$V_Error_Found] = True;
;    0378  3                             !+
;    0379  3                             ! Return unsucessfully.
;    0380  3                             !-
;    0381  2                             END;
;    0382  2
;    0383  2                     [DSQA$K_Dispat_DSX$EndPass_Mid]:
;    0384  3                             BEGIN
;    0385  3                             DSA$V_TRACE                                     = Off;  ! Turn TRACE bit off.           [01]
;    0386  3                             QA$AOB_Check_State [QA$K_Normal_Start]  = Off;  ! Stop Normal Start check.             [01]
;    0387  3                             QA$AOB_Check_State [QA$K_Multiple_Pass] = On;   ! Start Multiple Pass check.           [01]
;    0388  3                             QA$AOB_Routine_State [QA$V_Check_Done]  = True; ! Indicate end of this check.          [01]
;    0389  4                             RETURN (QA$K_Success)
;    0390  4
;    0391  2                             END;
```

```
;    0393  2                    !+
;    0394  2                    ! The following are not used in this check. So, just return Success for them.
;    0395  2                    !-
;    0396  2
;    0397  2                    [DSQA$K_Dispat_Before_Init,
;    0398  2                     DSQA$K_Dispat_After_Init,
;    0399  2                     DSQA$K_Dispat_CallTest,
;    0400  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,        !                                    [01]
;    0401  2                     DSQA$K_Dispat_DSX$EndPass_End,        !                                    [01]
;    0402  2                     DSQA$K_Dispat_Done_Tests,             !                                    [01]
;    0403  2
;    0404  2                     DSQA$K_Error_Error_Bgn,               !                                    [01]
;    0405  2
;    0406  2                     DSQA$K_Loop_DSX$BgnSub,
;    0407  2                     DSQA$K_Loop_DSX$EndSub,
;    0408  2                     DSQA$K_Loop_DSX$CkLoop,
;    0409  2
;    0410  2                     DSQA$K_QA_DSX$Branch]:
;    0411  3                             BEGIN
;    0412  4                             RETURN (QA$K_Success)
;    0413  2                             END;
;    0414  2
;    0415  2                    [OTHERWISE]:
;    0416  3                             BEGIN
;    0417  4                             Logic_Error ('Normal Start, Illegal Hook_Point')
;    0418  2                             END;
;    0419  2                    TES;
;    0420  2
;    0421  2                !+
;    0422  2                ! Come to here only if a QA error was found. Return QA$K_Failure to QA$Main, which then takes
;    0423  2                ! appropriate action.
;    0424  2                !-
;    0425  2
;    0426  3                RETURN (QA$K_Failure)
;    0427  1        END;


                                                                .TITLE  QACHECKS *** QA Check Routines
                                                                .IDENT  \10.10-02\

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                                53  4B  43  45  48  43  41  51  08  00000 P.AAA:  .ASCII  <8>\QACHECKS\                                ;
2A  2A  52  4F  52  52  45  20  41  51  2A  2A  2F  21  39  00009 P.AAB:  .ASCII  \9!/**QA ERROR** Program requested operat\        ;
73  65  75  71  65  72  20  6D  61  72  67  6F  72  50  20  00018                                                                   ;
                        74  61  72  65  70  6F  20  64  65  74  00027                                                               ;
6E  6F  69  74  6E  65  76  72  65  74  6E  69  20  72  6F  00031          .ASCII  \or intervention.!/\                               ;
                                            2F  21  2E  00040                                                                       ;
2A  2A  52  4F  52  52  45  20  41  51  2A  2A  2F  21  2F  00043 P.AAC:  .ASCII  \/!/**QA ERROR** Error reported in cleanu\        ;
64  65  74  72  6F  70  65  72  20  72  6F  72  72  45  20  00052                                                                   ;
                        75  6E  61  65  6C  63  20  6E  69  20  00061                                                               ;
                                2F  21  65  64  6F  63  20  70  0006B          .ASCII  \p code!/\                                     ;
2A  2A  52  4F  52  52  45  20  41  51  2A  2A  2F  21  1F  00073 P.AAD:  .ASCII  <31>\!/**QA ERROR** Error reported!/\             ;
64  65  74  72  6F  70  65  72  20  72  6F  72  72  45  20  00082                                                                   ;
                                                    2F  21  00091                                                                   ;
4E  20  2A  2A  52  4F  52  52  45  20  41  51  2A  2A  28  00093 P.AAE:  .ASCII  \(**QA ERROR** Normal start check failed!\        ;
65  68  63  20  74  72  61  74  73  20  6C  61  6D  72  6F  000A2                                                                   ;
```

```
                                21 64 65 6C 69 61 66 20 6B 63 000B1                                                    ;
                                                           2F 000BB             .ASCII  \/\                             ;
4D 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 29 000BC P.AAF:   .ASCII  \)**QA ERROR** Multiple pass check failed\       ;
68 63 20 73 73 61 70 20 65 6C 70 69 74 6C 75 000CB                                                                     ;
                                64 65 6C 69 61 66 20 6B 63 65 000DA                                                    ;
                                                           2F 21 000E4             .ASCII  \!/\                            ;
49 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 31 000E6 P.AAG:   .ASCII  \1**QA ERROR** Infinite loop-on-test chec\      ;
6E 6F 2D 70 6F 6F 6C 20 65 74 69 6E 69 66 6E 000F5                                                                     ;
                                63 65 68 63 20 74 73 65 74 2D 00104                                                    ;
                                2F 21 64 65 6C 69 61 66 20 6B 0010E             .ASCII  \k failed!/\                    ;
49 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 34 00118 P.AAH:   .ASCII  \4**QA ERROR** Infinite loop-on-subtest c\      ;
6E 6F 2D 70 6F 6F 6C 20 65 74 69 6E 69 66 6E 00127                                                                     ;
                                63 20 74 73 65 74 62 75 73 2D 00136                                                    ;
                   2F 21 64 65 6C 69 61 66 20 6B 63 65 68 00140             .ASCII  \heck failed!/\                 ;
52 20 2A 2A 52 4F 52 52 45 20 41 51 2A 2A 41 0014D P.AAI:   .ASCII  \A**QA ERROR** Run individual tests in re\      ;
74 20 6C 61 75 64 69 76 69 64 6E 69 20 6E 75 0015C                                                                     ;
                                65 72 20 6E 69 20 73 74 73 65 0016B                                                    ;
65 68 63 20 72 65 64 72 6F 20 65 73 72 65 76 00175             .ASCII  \verse order check failed!/\                    ;
                                2F 21 64 65 6C 69 61 66 20 6B 63 00184                                                 ;
55 51 20 2A 2A 35 32 21 2F 21 2F 21 5E 21 25 0018F P.AAJ:   .ASCII  \%!^!/!/!25** QUALITY ASSURANCE !26**!/\       ;
45 43 4E 41 52 55 53 53 41 20 59 54 49 4C 41 0019E                                                                     ;
                                2F 21 2A 2A 36 32 21 20 001AD                                                          ;
52 47 4F 52 50 3C 38 33 21 20 2A 30 31 21 26 001B5 P.AAK:   .ASCII  \&!10* !38<PROGRAM: !AC!> REV: !UL.!UL!/\       ;
20 3A 56 45 52 20 3E 21 43 41 21 20 3A 4D 41 001C4                                                                     ;
                   2F 21 4C 55 21 2E 4C 55 21 001D3                                                                    ;
      2F 21 2F 21 44 25 21 20 2A 33 32 21 0C 001DC P.AAL:   .ASCII  <12>\!23* !%D!/!/\                              ;
74 53 20 6C 61 6D 72 6F 4E 20 2A 38 32 21 15 001E9 P.AAM:   .ASCII  <21>\!28* Normal Start!/!/\                       ;
                                2F 21 2F 21 74 72 61 001F8                                                             ;
20 65 6C 70 69 74 6C 75 4D 20 2A 38 32 21 16 001FF P.AAN:   .ASCII  <22>\!28* Multiple Pass!/!/\                      ;
                                2F 21 2F 21 73 73 61 50 0020E                                                          ;
20 65 74 69 6E 69 66 6E 49 20 2A 34 32 21 1E 00216 P.AAO:   .ASCII  <30>\!24* Infinite Loop-On-Test!/!/\           ;
21 2F 21 74 73 65 54 2D 6E 4F 2D 70 6F 6F 4C 00225                                                                     ;
                                                           2F 00234                                                    ;
20 65 74 69 6E 69 66 6E 49 20 2A 32 32 21 21 00235 P.AAP:   .ASCII  \!!22* Infinite Loop-On-Subtest!/!/\           ;
74 73 65 74 62 75 53 2D 6E 4F 2D 70 6F 6F 4C 00244                                                                     ;
                                2F 21 2F 21 00253                                                                      ;
73 74 73 65 54 20 6E 75 52 20 2A 35 32 21 1C 00257 P.AAQ:   .ASCII  <28>\!25* Run Tests Backwards!/!/\               ;
   2F 21 2F 21 73 64 72 61 77 6B 63 61 42 20 00266                                                                     ;
61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 39 00274 P.AAR:   .ASCII  \9!/**QA Internal Error: Normal Start, II\      ;
6C 61 6D 72 6F 4E 20 3A 72 6F 72 72 45 20 6C 00283                                                                     ;
                   6C 49 20 2C 74 72 61 74 53 20 00292                                                                 ;
6E 69 6F 50 5F 6B 6F 6F 48 20 6C 61 67 65 6C 0029C             .ASCII  \legal Hook_Point!/\                            ;
                                                  2F 21 74 002AB                                                       ;

DSA$AL_APTMAIL=      65024
DSA$GL_FLAGS=        65024
DSA$GL_APTCOM=       65028
DSA$GL_PASSES=       65032
DSA$GL_UNITS=        65036
DSA$GL_SECTNO=       65040
DSA$GL_SID=          65044
DSA$GL_MSGTYP=       65088
DSA$GL_ERRNO=        65092
DSA$GL_EVENT=        65096
DSA$GL_SUBTNO=       65100
DSA$GL_TESTNO=       65104
DSA$GL_PASSNO=       65108
```

```
                                                DSA$GL_DEVLEN=      65112
                                                DSA$GL_DEVNAM=      65116
                                                DSA$GL_MSGPTR=      65128
                                                $MODULE=            P.AAA
                                                QA$T_OPERATOR_REQUEST_MESSAGE==
                                                                    P.AAB
                                                CLEANUP_ERROR_MESSAGE=
                                                                    P.AAC
                                                ERROR_REPORTED_MESSAGE=
                                                                    P.AAD
                                                NORMAL_START_MESSAGE=
                                                                    P.AAE
                                                MULTIPLE_PASS_MESSAGE=
                                                                    P.AAF
                                                INFINITE_TEST_MESSAGE=
                                                                    P.AAG
                                                INFINITE_SUBTEST_MESSAGE=
                                                                    P.AAH
                                                RUN_BACKWARDS_MESSAGE=
                                                                    P.AAI
                                                QA$T_HEADER_1==     P.AAJ
                                                QA$T_HEADER_2==     P.AAK
                                                QA$T_HEADER_3==     P.AAL
                                                HEADER_NS=          P.AAM
                                                HEADER_MP=          P.AAN
                                                HEADER_LT=          P.AAO
                                                HEADER_LS=          P.AAP
                                                HEADER_RB=          P.AAQ
                                                ES=                 P.AAR
                                                        .EXTRN  QA$FIND_USER_CALL_FRAME
                                                        .EXTRN  QA$PRINT_FORCED_ERRORS
                                                        .EXTRN  QA$ADD_FORCED_ERROR
                                                        .EXTRN  QA$ADD_QA_ERROR
                                                        .EXTRN  DSX$ABORT, DS$GL_FSTTEST
                                                        .EXTRN  DS$GL_LSTTEST, DS$GL_SUBTEST
                                                        .EXTRN  DS$GL_CLIBASE, QA$W_TESTLOOPS
                                                        .EXTRN  QA$W_SUBTESTLOOPS
                                                        .EXTRN  QA$W_MULTIPLEPASS
                                                        .EXTRN  QA$AOB_ROUTINE_STATE
                                                        .EXTRN  QA$AOB_CHECK_STATE
                                                        .EXTRN  QA$AOB_FLAGS, QA$W_BRANCH
                                                        .EXTRN  QA$W_ERROR_NUMBER
                                                        .EXTRN  QA$W_SAVE_TEST, QA$W_SAVE_LAST
                                                        .EXTRN  DS$PRINTF

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                  007C 00000            .ENTRY  QA$NORMAL_START, Save R2,R3,R4,R5,R6          ; 0317
56 00000000G   EF   9E 00002                            MOVAB   QA$AOB_CHECK_STATE, R6                        ;
55 00000000G   EF   9E 00009                            MOVAB   QA$AOB_ROUTINE_STATE, R5                      ;
54 00000000G   9F   9E 00010                            MOVAB   a#DS$PRINTF, R4                               ;
53 00000000'   EF   9E 00017                            MOVAB   QA$T_HEADER_1, R3                             ;
52        04   AC   D0 0001E                            MOVL    HOOK_POINT, R2                                ; 0354
12             52   D1 00022                            CMPL    R2, #18                                       ; 0356
               2B   12 00025                            BNEQ    1$                                            ;
               53   DD 00027                            PUSHL   R3                                            ; 0358
64             01   FB 00029                            CALLS   #1, DS$PRINTF                                 ;
```

```
                        7E 0000020C   9F  7D 0002C        MOVQ    a#^X0000020C, -(SP)                    ; 0359
                           00000208   9F  DD 00033        PUSHL   a#^X00000208                           ;
                                 26   A3  9F 00039        PUSHAB  QA$T_HEADER_2                          ;
                        64            04  FB 0003C        CALLS   #4, DS$PRINTF                          ;
                                      7E  D4 0003F        CLRL    -(SP)                                  ; 0360
                                 4D   A3  9F 00041        PUSHAB  QA$T_HEADER_3                          ;
                        64            02  FB 00044        CALLS   #2, DS$PRINTF                          ;
                                 5A   A3  9F 00047        PUSHAB  HEADER_NS                              ; 0361
                        64            01  FB 0004A        CALLS   #1, DS$PRINTF                          ;
                        65            01  88 0004D        BISB2   #1, QA$AOB_ROUTINE_STATE               ; 0363
                                      6B  11 00050        BRB     9$                                     ; 0364
                        05            52  D1 00052 1$:    CMPL    R2, #5                                 ; 0367
                                      26  12 00055        BNEQ    4$                                     ;
                 06     65            02  E1 00057        BBC     #2, QA$AOB_ROUTINE_STATE, 2$           ; 0369
                              FEB4   C3  9F 0005B        PUSHAB  CLEANUP_ERROR_MESSAGE                  ; 0370
                                      04  11 0005F        BRB     3$                                     ;
                              FEE4   C3  9F 00061 2$:    PUSHAB  ERROR_REPORTED_MESSAGE                 ; 0372
                        64            01  FB 00065 3$:    CALLS   #1, DS$PRINTF                          ;
                              FF04   C3  9F 00068        PUSHAB  NORMAL_START_MESSAGE                   ; 0374
                        64            01  FB 0006C        CALLS   #1, DS$PRINTF                          ;
                                      7E  D4 0006F        CLRL    -(SP)                                  ; 0376
              00000000G EF            01  FB 00071        CALLS   #1, QA$ADD_QA_ERROR                    ;
                        65            02  88 00078        BISB2   #2, QA$AOB_ROUTINE_STATE               ; 0377
                                      4B  11 0007B        BRB     11$                                    ;
                                      52  D5 0007D 4$:    TSTL    R2                                     ; 0383
                                      12  12 0007F        BNEQ    5$                                     ;
              0000FE01  9F            04  8A 00081        BICB2   #4, a#^X0000FE01                       ; 0385
                        66            01  8A 00088        BICB2   #1, QA$AOB_CHECK_STATE                 ; 0386
                        66            02  88 0008B        BISB2   #2, QA$AOB_CHECK_STATE                 ; 0387
                        65            08  88 0008E        BISB2   #8, QA$AOB_ROUTINE_STATE               ; 0388
                                      2A  11 00091        BRB     9$                                     ; 0389
                                      05  15 00093 5$:    BLEQ    6$                                     ; 0397
                        02            52  D1 00095        CMPL    R2, #2                                 ;
                                      23  15 00098        BLEQ    9$                                     ;
                        06            52  D1 0009A 6$:    CMPL    R2, #6                                 ;
                                      1E  13 0009D        BEQL    9$                                     ;
                        09            52  D1 0009F        CMPL    R2, #9                                 ;
                                      05  19 000A2        BLSS    7$                                     ;
                        0B            52  D1 000A4        CMPL    R2, #11                                ;
                                      14  15 000A7        BLEQ    9$                                     ;
                        11            52  D1 000A9 7$:    CMPL    R2, #17                                ;
                                      0F  13 000AC        BEQL    9$                                     ;
                        13            52  D1 000AE        CMPL    R2, #19                                ;
                                      05  19 000B1        BLSS    8$                                     ;
                        15            52  D1 000B3        CMPL    R2, #21                                ;
                                      05  15 000B6        BLEQ    9$                                     ;
                        18            52  D1 000B8 8$:    CMPL    R2, #24                                ;
                                      04  12 000BB        BNEQ    10$                                    ;
                        50            01  D0 000BD 9$:    MOVL    #1, R0                                 ; 0412
                                          04 000C0        RET                                            ;
                              00E5   C3  9F 000C1 10$:   PUSHAB  ES                                     ; 0417
                        64            01  FB 000C5        CALLS   #1, DS$PRINTF                          ;
                                      50  D4 000C8 11$:   CLRL    R0                                     ; 0426
                                          04 000CA        RET                                            ; 0427

; Routine Size:  203 bytes,    Routine Base:  CODE + 0000
```

```
;   0428  1
```

```
;    0430  1      %SBTTL 'QA$Multiple_Pass'
;    0431  1
;    0432  1      GLOBAL ROUTINE QA$Multiple_Pass (Hook_Point) =
;    0433  1
;    0434  1      !++
;    0435  1      ! FUNCTIONAL DESCRIPTION:
;    0436  1      !
;    0437  1      !       This routine performs the QA Multiple Pass check.
;    0438  1      !
;    0439  1      ! CALLER(S):
;    0440  1      !
;    0441  1      !       QA$Main
;    0442  1      !
;    0443  1      ! FORMAL PARAMETERS:
;    0444  1      !
;    0445  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0446  1      !                    The constants are defined in the DSQA macro.
;    0447  1      !
;    0448  1      ! IMPLICIT INPUTS:
;    0449  1      !
;    0450  1      !       NONE
;    0451  1      !
;    0452  1      ! IMPLICIT OUTPUTS:
;    0453  1      !
;    0454  1      !       NONE
;    0455  1      !
;    0456  1      ! COMPLETION CODES:
;    0457  1      !
;    0458  1      !       0 - QA error detected; unsuccessful return
;    0459  1      !       1 - No QA errors; successful return
;    0460  1      !
;    0461  1      ! SIDE EFFECTS:
;    0462  1      !
;    0463  1      !       NONE
;    0464  1      !
;    0465  1      !--
```

```
;    0467  2     BEGIN
;    0468  2             SELECTONE .Hook_Point OF
;    0469  2                     SET
;    0470  2                     [DSQA$K_Start_VRReStart]:                !                                          [01]
;    0471  3                             BEGIN
;    0472  3                             MAP                              !                                          [01]
;    0473  3                                     DS$GL_CLIBASE : BLOCK [, BYTE]; ! Access the CLI date vector.      [01]
;    0474  3
;    0475  3                             $DS_PRINTF (QA$T_Header_1);
;    0476  3                             $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0477  3                             $DS_PRINTF (QA$T_Header_3, 0);
;    0478  3                             $DS_PRINTF (Header_MP);
;    0479  3
;    0480  3                             DS$GL_CLIBASE [CLI$L_PASS] = .QA$W_MultiplePass;! Set number of passes to requested amt.[01]
;    0481  3                             QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0482  4                             RETURN (QA$K_Success)
;    0483  2                             END;
;    0484  2
;    0485  2                     [DSQA$K_Error_Error_End]:
;    0486  3                             BEGIN
;    0487  3                             IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !                   [01]
;    0488  4                                     $DS_PRINTF (Cleanup_Error_Message)      !                          [01]
;    0489  3                             ELSE                                            !                          [01]
;    0490  3                                     $DS_PRINTF (Error_Reported_Message);    !                          [01]
;    0491  3
;    0492  3                             $DS_PRINTF (Multiple_Pass_Message);             !                          [02]
;    0493  3
;    0494  3                             QA$Add_QA_Error (QA$K_Multiple_Pass);
;    0495  3                             QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0496  3                             !+
;    0497  3                             ! Return unsucessfully.
;    0498  3                             !-
;    0499  2                             END;
;    0500  2
;    0501  2                     [DSQA$K_Dispat_DSX$EndPass_Mid]:         !                                          [01]
;    0502  3                             BEGIN
;    0503  3                             !+                                                                         [01]
;    0504  3                             ! All done the Multiple Pass check. Clean up.                              [01]
;    0505  3                             !-                                                                         [01]
;    0506  3
;    0507  3                             QA$AOB_Check_State [QA$K_Multiple_Pass] = Off;  ! Stop Multiple Pass check.  [01]
;    0508  3                             QA$AOB_Check_State [QA$K_Loop_On_Test]  = On;   ! Start Loop on Test check. [01]
;    0509  3                             QA$AOB_Routine_State [QA$V_Check_Done]  = True; ! Indicate check is done.   [01]
;    0510  4                             RETURN (QA$K_Success)                           !                          [01]
;    0511  2                             END;
```

```
;   0513  2                    !+
;   0514  2                    ! The following are not used in this check.
;   0515  2                    !-
;   0516  2
;   0517  2                    [DSQA$K_Dispat_Before_Init,
;   0518  2                     DSQA$K_Dispat_After_Init,              !                                                [01]
;   0519  2                     DSQA$K_Dispat_CallTest,
;   0520  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,         !                                                [01]
;   0521  2                     DSQA$K_Dispat_DSX$EndPass_End,         !                                                [01]
;   0522  2                     DSQA$K_Dispat_Done_Tests,              !                                                [01]
;   0523  2
;   0524  2                     DSQA$K_Error_Error_Bgn,                !                                                [01]
;   0525  2
;   0526  2                     DSQA$K_Loop_DSX$BgnSub,
;   0527  2                     DSQA$K_Loop_DSX$EndSub,
;   0528  2                     DSQA$K_Loop_DSX$CkLoop,
;   0529  2
;   0530  2                     DSQA$K_QA_DSX$Branch]:
;   0531  3                            BEGIN
;   0532  4                            RETURN (QA$K_Success)
;   0533  2                            END;
;   0534  2
;   0535  2                    [OTHERWISE]:
;   0536  3                            BEGIN
;   0537  4                            Logic_Error ('Multiple Pass, Illegal Hook_Point')
;   0538  2                            END;
;   0539  2                    TES;
;   0540  3              RETURN (QA$K_Failure)
;   0541  1      END;


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61  6E  72  65  74  6E  49  20  41  51  2A  2A  2F  21  3A  002AE P.AAS:  .ASCII  \:!/**QA Internal Error: Multiple Pass, I\  ;
70  69  74  6C  75  4D  20  3A  72  6F  72  72  45  20  6C  002BD                                                            ;
                    49  20  2C  73  73  61  50  20  65  6C  002CC                                                            ;
69  6F  50  5F  6B  6F  6F  48  20  6C  61  67  65  6C  6C  002D6         .ASCII  \llegal Hook_Point!/\                       ;
                                          2F  21  74  6E  002E5                                                            ;

                                                                ES=                 P.AAS


                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                  007C 00000            .ENTRY  QA$MULTIPLE_PASS, Save R2,R3,R4,R5,R6       ; 0432
                                  56 00000000G  EF  9E 00002            MOVAB   QA$AOB_CHECK_STATE, R6                      ;
                                  55 00000000G  EF  9E 00009            MOVAB   QA$AOB_ROUTINE_STATE, R5                    ;
                                  54 00000000G  9F  9E 00010            MOVAB   a#DS$PRINTF, R4                             ;
                                  53 00000000'  EF  9E 00017            MOVAB   QA$T_HEADER_1, R3                           ;
                                  52       04   AC  D0 0001E            MOVL    HOOK_POINT, R2                              ; 0468
                                  12            52  D1 00022            CMPL    R2, #18                                     ; 0470
                                                36  12 00025            BNEQ    1$                                          ;
                                                53  DD 00027            PUSHL   R3                                          ; 0475
                                  64            01  FB 00029            CALLS   #1, DS$PRINTF                               ;
                                  7E 0000020C   9F  7D 0002C            MOVQ    a#^X0000020C, -(SP)                         ; 0476
                                     00000208   9F  DD 00033            PUSHL   a#^X00000208                                ;
```

```
                                  26   A3  9F 00039          PUSHAB  QA$T_HEADER_2                                  ;
                          64           04  FB 0003C          CALLS   #4, DS$PRINTF                                  ;
                                       7E  D4 0003F          CLRL    -(SP)                                          ; 0477
                                  4D   A3  9F 00041          PUSHAB  QA$T_HEADER_3                                  ;
                          64           02  FB 00044          CALLS   #2, DS$PRINTF                                  ;
                                  70   A3  9F 00047          PUSHAB  HEADER_MP                                      ; 0478
                          64           01  FB 0004A          CALLS   #1, DS$PRINTF                                  ;
                 00000000G  EF 00000000G  EF  3C 0004D          MOVZWL  QA$W_MULTIPLEPASS, DS$GL_CLIBASE+44            ; 0480
                          65           01  88 00058          BISB2   #1, QA$AOB_ROUTINE_STATE                       ; 0481
                                       64  11 0005B          BRB     9$                                             ; 0482
                          05           52  D1 0005D 1$:      CMPL    R2, #5                                         ; 0485
                                       26  12 00060          BNEQ    4$                                             ;
                06        65           02  E1 00062          BBC     #2, QA$AOB_ROUTINE_STATE, 2$                   ; 0487
                                  FEB4 C3  9F 00066          PUSHAB  CLEANUP_ERROR_MESSAGE                          ; 0488
                                       04  11 0006A          BRB     3$                                             ;
                                  FEE4 C3  9F 0006C 2$:      PUSHAB  ERROR_REPORTED_MESSAGE                         ; 0490
                          64           01  FB 00070 3$:      CALLS   #1, DS$PRINTF                                  ;
                                  FF2D C3  9F 00073          PUSHAB  MULTIPLE_PASS_MESSAGE                          ; 0492
                          64           01  FB 00077          CALLS   #1, DS$PRINTF                                  ;
                                       01  DD 0007A          PUSHL   #1                                             ; 0494
                 00000000G  EF           01  FB 0007C          CALLS   #1, QA$ADD_QA_ERROR                            ;
                          65           02  88 00083          BISB2   #2, QA$AOB_ROUTINE_STATE                       ; 0495
                                       44  11 00086          BRB     11$                                            ;
                                       52  D5 00088 4$:      TSTL    R2                                             ; 0501
                                       0B  12 0008A          BNEQ    5$                                             ;
                          66           02  8A 0008C          BICB2   #2, QA$AOB_CHECK_STATE                         ; 0507
                          66           04  88 0008F          BISB2   #4, QA$AOB_CHECK_STATE                         ; 0508
                          65           08  88 00092          BISB2   #8, QA$AOB_ROUTINE_STATE                       ; 0509
                                       2A  11 00095          BRB     9$                                             ; 0510
                                       05  15 00097 5$:      BLEQ    6$                                             ; 0517
                          02           52  D1 00099          CMPL    R2, #2                                         ;
                                       23  15 0009C          BLEQ    9$                                             ;
                          06           52  D1 0009E 6$:      CMPL    R2, #6                                         ;
                                       1E  13 000A1          BEQL    9$                                             ;
                          09           52  D1 000A3          CMPL    R2, #9                                         ;
                                       05  19 000A6          BLSS    7$                                             ;
                          0B           52  D1 000A8          CMPL    R2, #11                                        ;
                                       14  15 000AB          BLEQ    9$                                             ;
                          11           52  D1 000AD 7$:      CMPL    R2, #17                                        ;
                                       0F  13 000B0          BEQL    9$                                             ;
                          13           52  D1 000B2          CMPL    R2, #19                                        ;
                                       05  19 000B5          BLSS    8$                                             ;
                          15           52  D1 000B7          CMPL    R2, #21                                        ;
                                       05  15 000BA          BLEQ    9$                                             ;
                          18           52  D1 000BC 8$:      CMPL    R2, #24                                        ;
                                       04  12 000BF          BNEQ    10$                                            ;
                          50           01  D0 000C1 9$:      MOVL    #1, R0                                         ; 0532
                                           04 000C4          RET                                                    ;
                                  011F C3  9F 000C5 10$:     PUSHAB  ES                                             ; 0537
                          64           01  FB 000C9          CALLS   #1, DS$PRINTF                                  ;
                                       50  D4 000CC 11$:     CLRL    R0                                             ; 0540
                                           04 000CE          RET                                                    ; 0541

; Routine Size:  207 bytes,    Routine Base:  CODE + 00CB
```

```
;    0543  1      %SBTTL 'QA$Loop_On_Test'
;    0544  1
;    0545  1      GLOBAL ROUTINE QA$Loop_On_Test (Hook_Point) =
;    0546  1
;    0547  1      !++
;    0548  1      ! FUNCTIONAL DESCRIPTION:
;    0549  1      !
;    0550  1      !       This routine performs the QA Loop on Test check.
;    0551  1      !
;    0552  1      ! CALLER(S):
;    0553  1      !
;    0554  1      !       QA$Main
;    0555  1      !
;    0556  1      ! FORMAL PARAMETERS:
;    0557  1      !
;    0558  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0559  1      !                    The constants are defined in the DSQA macro.
;    0560  1      !
;    0561  1      ! IMPLICIT INPUTS:
;    0562  1      !
;    0563  1      !       NONE
;    0564  1      !
;    0565  1      ! IMPLICIT OUTPUTS:
;    0566  1      !
;    0567  1      !       NONE
;    0568  1      !
;    0569  1      ! COMPLETION CODES:
;    0570  1      !
;    0571  1      !       0 - QA error detected; unsuccessful return
;    0572  1      !       1 - No QA errors; successful return
;    0573  1      !
;    0574  1      ! SIDE EFFECTS:
;    0575  1      !
;    0576  1      !       NONE
;    0577  1      !
;    0578  1      !--
```

```
;    0580  2      BEGIN
;    0581  2              OWN
;    0582  2                      Loop_Counter : WORD;
;    0583  2
;    0584  2              SELECTONE .Hook_Point OF
;    0585  2                      SET
;    0586  2                      [DSQA$K_Start_VRReStart]:
;    0587  3                              BEGIN
;    0588  3                              MAP                                     !                                       [01]
;    0589  3                                      DS$GL_CLIBASE : BLOCK [, BYTE]; ! Access the CLI data vector.           [01]
;    0590  3
;    0591  3                              !+
;    0592  3                              ! The initializations required for this check and that are specific to this
;    0593  3                              ! check must be done here.
;    0594  3                              !-
;    0595  3
;    0596  3                              Loop_Counter = 1;                       ! Loop counter for each test.           [01]
;    0597  3                              QA$AOB_Flags [QA$K_Loop_Test_Done] = True;
;    0598  3
;    0599  3                              $DS_PRINTF (QA$T_Header_1);
;    0600  3                              $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0601  3                              $DS_PRINTF (QA$T_Header_3, 0);
;    0602  3                              $DS_PRINTF (Header_LT);
;    0603  3
;    0604  3                              DS$GL_CLIBASE [CLI$L_PASS] = 1; ! Reset to one pass after the Multiple Pass check.  [01]
;    0605  3                              QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0606  4                              RETURN (QA$K_Success)
;    0607  2                              END;
;    0608  2
;    0609  2                      [DSQA$K_Error_Error_End]:
;    0610  3                              BEGIN
;    0611  3                              IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !                       [01]
;    0612  4                                      $DS_PRINTF (Cleanup_Error_Message)      !                               [01]
;    0613  3                              ELSE                                            !                               [01]
;    0614  3                                      $DS_PRINTF (Error_Reported_Message);    !                               [01]
;    0615  3
;    0616  3                              $DS_PRINTF (Infinite_Test_Message);             !                               [02]
;    0617  3
;    0618  3                              QA$Add_QA_Error (QA$K_Loop_On_Test);
;    0619  3                              QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0620  2                              END;
```

```
;    0622  2                    [DSQA$K_Dispat_DSX$EndPass_Mid]:        !                                                [01]
;    0623  3                            BEGIN
;    0624  3                            !+
;    0625  3                            ! All done the Loop on Test check. Clean up.                                 [01]
;    0626  3                            !-
;    0627  3
;    0628  3                            QA$AOB_Check_State [QA$K_Loop_On_Test]    = Off;
;    0629  3                            QA$AOB_Check_State [QA$K_Loop_On_Subtest] = On;
;    0630  3                            QA$AOB_Routine_State [QA$V_Check_Done]    = True;
;    0631  3                            QA$AOB_Flags [QA$K_First_Time]            = True;       ! For the next check (LS).      [01]
;    0632  4                            RETURN (QA$K_Success)
;    0633  2                            END;
;    0634  2
;    0635  2                    [DSQA$K_Dispat_CallTest]:
;    0636  3                            BEGIN
;    0637  3                            IF (.Loop_Counter EQL .QA$W_TestLoops) THEN
;    0638  4                                    BEGIN
;    0639  4                                    Loop_Counter = 1;
;    0640  4                                    QA$AOB_Flags [QA$K_Loop_Test_Done] = True
;    0641  4                                    END
;    0642  3                            ELSE
;    0643  4                                    BEGIN
;    0644  4                                    Loop_Counter = .Loop_Counter + 1;
;    0645  4                                    QA$AOB_Flags [QA$K_Loop_Test_Done] = False
;    0646  3                                    END;
;    0647  4                            RETURN (QA$K_Success)
;    0648  2                            END;
```

```
;   0650  2                    !+
;   0651  2                    ! The following are not used in this check.
;   0652  2                    !-
;   0653  2
;   0654  2                    [DSQA$K_Dispat_Before_Init,
;   0655  2                     DSQA$K_Dispat_After_Init,              !                                                [01]
;   0656  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,         !                                                [01]
;   0657  2                     DSQA$K_Dispat_DSX$EndPass_End,         !                                                [01]
;   0658  2                     DSQA$K_Dispat_Done_Tests,              !                                                [01]
;   0659  2
;   0660  2                     DSQA$K_Error_Error_Bgn,                !                                                [01]
;   0661  2
;   0662  2                     DSQA$K_Loop_DSX$BgnSub,
;   0663  2                     DSQA$K_Loop_DSX$EndSub,
;   0664  2                     DSQA$K_Loop_DSX$CkLoop,
;   0665  2
;   0666  2                     DSQA$K_QA_DSX$Branch]:
;   0667  3                            BEGIN
;   0668  4                            RETURN (QA$K_Success)
;   0669  2                            END;
;   0670  2
;   0671  2                    [OTHERWISE]:
;   0672  3                            BEGIN
;   0673  4                            Logic_Error ('Loop on Test, Illegal Hook_Point')
;   0674  2                            END;
;   0675  2                    TES;
;   0676  3              RETURN (QA$K_Failure)
;   0677  1      END;


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61  6E  72  65  74  6E  49  20  41  51  2A  2A  2F  21  39  002E9 P.AAT:  .ASCII  \9!/**QA Internal Error: Loop on Test, Il\  ;
6F  20  70  6F  6F  4C  20  3A  72  6F  72  72  45  20  6C  002F8                                                            ;
                    6C  49  20  2C  74  73  65  54  20  6E  00307                                                            ;
6E  69  6F  50  5F  6B  6F  6F  48  20  6C  61  67  65  6C  00311         .ASCII  \legal Hook_Point!/\                        ;
                                            2F  21  74  00320                                                            ;

                                                                        .PSECT  WORK,NOEXE,2

                                                        00000 LOOP_COUNTER:
                                                                        .BLKB   2

                                                              ES=                P.AAT


                                                                        .PSECT  CODE,NOWRT,  SHR,2

                                                  01FC 00000            .ENTRY  QA$LOOP_ON_TEST, Save R2,R3,R4,R5,R6,R7,R8  ; 0545
                                  58 00000000G  EF  9E 00002            MOVAB   QA$AOB_CHECK_STATE, R8                      ;
                                  57 00000000'  EF  9E 00009            MOVAB   LOOP_COUNTER, R7                            ;
                                  56 00000000G  EF  9E 00010            MOVAB   QA$AOB_ROUTINE_STATE, R6                    ;
                                  55 00000000G  EF  9E 00017            MOVAB   QA$AOB_FLAGS, R5                            ;
                                  54 00000000G  9F  9E 0001E            MOVAB   a#DS$PRINTF, R4                             ;
                                  53 00000000'  EF  9E 00025            MOVAB   QA$T_HEADER_1, R3                           ;
                                  52       04   AC  D0 0002C            MOVL    HOOK_POINT, R2                              ; 0584
```

```
                    12          52  D1 00030        CMPL    R2, #18                                  ; 0586
                                39  12 00033        BNEQ    1$                                       ;
                    67          01  B0 00035        MOVW    #1, LOOP_COUNTER                         ; 0596
                    65          10  88 00038        BISB2   #16, QA$AOB_FLAGS                        ; 0597
                                53  DD 0003B        PUSHL   R3                                       ; 0599
                    64          01  FB 0003D        CALLS   #1, DS$PRINTF                            ;
                    7E 0000020C 9F  7D 00040        MOVQ    @#^X0000020C, -(SP)                      ; 0600
                       00000208 9F  DD 00047        PUSHL   @#^X00000208                             ;
                            26  A3  9F 0004D        PUSHAB  QA$T_HEADER_2                            ;
                    64          04  FB 00050        CALLS   #4, DS$PRINTF                            ;
                                7E  D4 00053        CLRL    -(SP)                                    ; 0601
                            4D  A3  9F 00055        PUSHAB  QA$T_HEADER_3                            ;
                    64          02  FB 00058        CALLS   #2, DS$PRINTF                            ;
                          0087  C3  9F 0005B        PUSHAB  HEADER_LT                                ; 0602
                    64          01  FB 0005F        CALLS   #1, DS$PRINTF                            ;
          00000000G EF          01  D0 00062        MOVL    #1, DS$GL_CLIBASE+44                     ; 0604
                    66          01  88 00069        BISB2   #1, QA$AOB_ROUTINE_STATE                 ; 0605
                                58  11 0006C        BRB     7$                                       ; 0606
                    05          52  D1 0006E 1$:    CMPL    R2, #5                                   ; 0609
                                26  12 00071        BNEQ    4$                                       ;
        06          66          02  E1 00073        BBC     #2, QA$AOB_ROUTINE_STATE, 2$             ; 0611
                          FEB4  C3  9F 00077        PUSHAB  CLEANUP_ERROR_MESSAGE                    ; 0612
                                04  11 0007B        BRB     3$                                       ;
                          FEE4  C3  9F 0007D 2$:    PUSHAB  ERROR_REPORTED_MESSAGE                   ; 0614
                    64          01  FB 00081 3$:    CALLS   #1, DS$PRINTF                            ;
                          FF57  C3  9F 00084        PUSHAB  INFINITE_TEST_MESSAGE                    ; 0616
                    64          01  FB 00088        CALLS   #1, DS$PRINTF                            ;
                                02  DD 0008B        PUSHL   #2                                       ; 0618
          00000000G EF          01  FB 0008D        CALLS   #1, QA$ADD_QA_ERROR                      ;
                    66          02  88 00094        BISB2   #2, QA$AOB_ROUTINE_STATE                 ; 0619
                                66  11 00097        BRB     14$                                      ;
                                52  D5 00099 4$:    TSTL    R2                                       ; 0622
                                0E  12 0009B        BNEQ    5$                                       ;
                    68          04  8A 0009D        BICB2   #4, QA$AOB_CHECK_STATE                   ; 0628
                    68          08  88 000A0        BISB2   #8, QA$AOB_CHECK_STATE                   ; 0629
                    66          08  88 000A3        BISB2   #8, QA$AOB_ROUTINE_STATE                 ; 0630
                    65          20  88 000A6        BISB2   #32, QA$AOB_FLAGS                        ; 0631
                                49  11 000A9        BRB     12$                                      ; 0632
                    15          52  D1 000AB 5$:    CMPL    R2, #21                                  ; 0635
                                18  12 000AE        BNEQ    8$                                       ;
          00000000G EF          67  B1 000B0        CMPW    LOOP_COUNTER, QA$W_TESTLOOPS             ; 0637
                                08  12 000B7        BNEQ    6$                                       ;
                    67          01  B0 000B9        MOVW    #1, LOOP_COUNTER                         ; 0639
                    65          10  88 000BC        BISB2   #16, QA$AOB_FLAGS                        ; 0640
                                33  11 000BF        BRB     12$                                      ;
                                67  B6 000C1 6$:    INCW    LOOP_COUNTER                             ; 0644
                    65          10  8A 000C3        BICB2   #16, QA$AOB_FLAGS                        ; 0645
                                2C  11 000C6 7$:    BRB     12$                                      ; 0647
                                52  D5 000C8 8$:    TSTL    R2                                       ; 0654
                                05  15 000CA        BLEQ    9$                                       ;
                    02          52  D1 000CC        CMPL    R2, #2                                   ;
                                23  15 000CF        BLEQ    12$                                      ;
                    06          52  D1 000D1 9$:    CMPL    R2, #6                                   ;
                                1E  13 000D4        BEQL    12$                                      ;
                    09          52  D1 000D6        CMPL    R2, #9                                   ;
                                05  19 000D9        BLSS    10$                                      ;
                    0B          52  D1 000DB        CMPL    R2, #11                                  ;
```

```
                                14  15 000DE          BLEQ    12$                                  ;
                    11          52  D1 000E0 10$:     CMPL    R2, #17                              ;
                                0F  13 000E3          BEQL    12$                                  ;
                    13          52  D1 000E5          CMPL    R2, #19                              ;
                                05  19 000E8          BLSS    11$                                  ;
                    14          52  D1 000EA          CMPL    R2, #20                              ;
                                05  15 000ED          BLEQ    12$                                  ;
                    18          52  D1 000EF 11$:     CMPL    R2, #24                              ;
                                04  12 000F2          BNEQ    13$                                  ;
                    50          01  D0 000F4 12$:     MOVL    #1, R0                               ; 0668
                                    04 000F7          RET                                          ;
                          015A  C3  9F 000F8 13$:     PUSHAB  ES                                   ; 0673
                    64          01  FB 000FC          CALLS   #1, DS$PRINTF                        ;
                                50  D4 000FF 14$:     CLRL    R0                                   ; 0676
                                    04 00101          RET                                          ; 0677

; Routine Size:  258 bytes,    Routine Base:  CODE + 019A
```

```
;    0679  1     %SBTTL 'QA$Loop_On_Subtest'
;    0680  1
;    0681  1     GLOBAL ROUTINE QA$Loop_On_Subtest (Hook_Point) =        !                                   Start of [1]
;    0682  1
;    0683  1     !++
;    0684  1     ! FUNCTIONAL DESCRIPTION:
;    0685  1     !
;    0686  1     !       This routine performs the QA Loop on Subtest check.
;    0687  1     !
;    0688  1     ! CALLER(S):
;    0689  1     !
;    0690  1     !       QA$Main
;    0691  1     !
;    0692  1     ! FORMAL PARAMETERS:
;    0693  1     !
;    0694  1     !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant value.
;    0695  1     !                    The constants are defined in the DSQA macro.
;    0696  1     !
;    0697  1     ! IMPLICIT INPUTS:
;    0698  1     !
;    0699  1     !       NONE
;    0700  1     !
;    0701  1     ! IMPLICIT OUTPUTS:
;    0702  1     !
;    0703  1     !       NONE
;    0704  1     !
;    0705  1     ! COMPLETION CODES:
;    0706  1     !
;    0707  1     !       0 - QA error detected; unsuccessful return
;    0708  1     !       1 - No QA errors; successful return
;    0709  1     !
;    0710  1     ! SIDE EFFECTS:
;    0711  1     !
;    0712  1     !       NONE
;    0713  1     !
;    0714  1     !--
```

```
;   0716  2     BEGIN
;   0717  2             MAP                                         !                                                   [01]
;   0718  2                     DS$GL_CLIBASE : BLOCK [, BYTE];     ! Access the Cli data vector                        [01]
;   0719  2
;   0720  2             SELECTONE .Hook_Point OF
;   0721  2                     SET
;   0722  2                     [DSQA$K_Start_VRReStart]:           !                                                   [01]
;   0723  3                             BEGIN
;   0724  3                             !+
;   0725  3                             ! The initializations required for this check and that are specific to this check
;   0726  3                             ! must be done here.
;   0727  3                             !+
;   0728  3
;   0729  3                             IF (.QA$AOB_Flags [QA$K_First_Time]) THEN       !                                   [01]
;   0730  4                                     BEGIN
;   0731  4                                     $DS_PRINTF (QA$T_Header_1);
;   0732  4                                     $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;   0733  4                                     $DS_PRINTF (QA$T_Header_3, 0);
;   0734  4                                     $DS_PRINTF (Header_LS);
;   0735  4
;   0736  4                                     QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done initializing
;   0737  4                                     QA$AOB_Flags [QA$K_First_Time]        = False;  ! No more first time        [01]
;   0738  4                                     QA$AOB_Flags [QA$K_No_Header]         = False;  ! Print the header          [01]
;   0739  4
;   0740  4                                     !+
;   0741  4                                     ! Set up the subtest and pass numbers for looping on a subtest.             [01]
;   0742  4                                     !-
;   0743  4
;   0744  4                                     DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;   ! Set the test to loop on   [01]
;   0745  4                                     DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Test;   ! Set the last test to loop on [01]
;   0746  4                                     DS$GL_CLIBASE [CLI$L_SUBT] = 1;                 ! Set the subtest to loop on [01]
;   0747  4                                     DS$GL_CLIBASE [CLI$L_PASS] = .QA$W_SubtestLoops;! Set the no. of loops      [01]
;   0748  4                                     END                     !                                                   [01]
```

```
;   0750  3                 ELSE                                !                                                   [01]
;   0751  4                         BEGIN                       !                                                   [01]
;   0752  4                         !+
;   0753  4                         ! This is the nth (n > 1) time we have gotten to the VRReStart label. Thus, we  [01]
;   0754  4                         ! got there by a branch from the EndSub routine, executed when the required     [01]
;   0755  4                         ! number of loops on a subtest had been done. Thus, go onto the next subtest in [01]
;   0756  4                         ! the same test. This implies that if the subtest just completed was the last   [01]
;   0757  4                         ! subtest in that test, then the increment below will cause the subtest to not  [01]
;   0758  4                         ! be found in the next execution of that test. So, the test will return to the  [01]
;   0759  4                         ! Dispatch routine, where the test number will be incremented and the subtest   [01]
;   0760  4                         ! number will be reset to one. Thus, if x is the last subtest number in         [01]
;   0761  4                         ! test a, and test b follows test a, the sequence will be as follows:           [01]
;   0762  4                         ! Test a.x-1, VRReStart, Test a.x, VrReStart, Test a, Dispatch, Test b.1,       [01]
;   0763  4                         ! VrReStart, Test b.2, etc. Note that the above gives only those subtest        [01]
;   0764  4                         ! numbers (x-1, x, 1, etc.) that will be looped on. Note that all the subtests  [01]
;   0765  4                         ! prior to that subtest number would be executed before looping on the nth      [01]
;   0766  4                         ! subtest. I must also reset the Test, Last, and Pass areas in the Cli          [01]
;   0767  4                         ! data vector because they get destroyed somewhere (Init_Context?).             [01]
;   0768  4                         !-
;   0769  4
;   0770  4                         QA$AOB_Flags [QA$K_No_Header] = True;           ! Do not print header anymore   [01]
;   0771  4
;   0772  4                         DS$GL_CLIBASE [CLI$L_SUBT] = .DS$GL_SUBTEST + 1;! Do next subtest               [01]
;   0773  4                         DS$GL_CLIBASE [CLI$L_TEST] = .DS$GL_FSTTEST;    ! Reset first test number       [01]
;   0774  4                         DS$GL_CLIBASE [CLI$L_LAST] = .DS$GL_FSTTEST;    ! Reset last test number        [01]
;   0775  4                         DS$GL_CLIBASE [CLI$L_PASS] = .QA$W_SubtestLoops;! Reset pass number             [01]
;   0776  3                         END;                        !                                                   [01]
;   0777  4                 RETURN (QA$K_Success)
;   0778  2                 END;
```

```
;    0780  2                    [DSQA$K_Error_Error_End]:
;    0781  3                            BEGIN
;    0782  3                            IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !                    [01]
;    0783  4                                    $DS_PRINTF (Cleanup_Error_Message)             !                    [01]
;    0784  3                            ELSE                                                   !                    [01]
;    0785  3                                    $DS_PRINTF (Error_Reported_Message);           !                    [01]
;    0786  3
;    0787  3                            $DS_PRINTF (Infinite_Subtest_Message);                 !                    [02]
;    0788  3
;    0789  3                            QA$Add_QA_Error (QA$K_Loop_On_Subtest);
;    0790  3                            QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0791  2                            END;
;    0792  2
;    0793  2                    [DSQA$K_Dispat_DSX$EndPass_Mid]:
;    0794  3                            BEGIN
;    0795  3                            !+
;    0796  3                            ! All done the Loop on Subtest check. Clean up. Restore the first and last test    [01]
;    0797  3                            ! numbers just in case....                                                        [01]
;    0798  3                            !-
;    0799  3
;    0800  3                            DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;   ! Restore the old first test #   [01]
;    0801  3                            DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last;   ! Restore the old last test #    [01]
;    0802  3
;    0803  3                            QA$AOB_Check_State [QA$K_Loop_On_Subtest] = Off;
;    0804  3                            QA$AOB_Check_State [QA$K_Run_Backwards]   = On;
;    0805  3                            QA$AOB_Routine_State [QA$V_Check_Done]    = True;
;    0806  3                            QA$AOB_Flags [QA$K_No_Header]             = False;      ! Print header again   [01]
;    0807  4                            RETURN (QA$K_Success)
;    0808  2                            END;
;    0809  2                                                                                                  !G:2
;    0810  2                    [DSQA$K_Dispat_Done_Tests]:                                   !                    !G:
;    0811  3                            BEGIN                                                 ! Reset ds$gl_subtest to 0 !G:2
;    0812  3                            DS$GL_SUBTEST = 0;                                    ! when the last test was done !G:2
;    0813  3                            RETURN (QA$K_Success);                                !                    !G:2
;    0814  2                            END;                                                                       !G:2
```

```
;    0816  2                    !+
;    0817  2                    ! The following are not used in this check.
;    0818  2                    !-
;    0819  2
;    0820  2                    [DSQA$K_Dispat_Before_Init,
;    0821  2                     DSQA$K_Dispat_After_Init,              !                                [01]
;    0822  2                     DSQA$K_Dispat_CallTest,                !                                [01]
;    0823  2                     DSQA$K_Dispat_DSX$EndPass_Bgn,         !                                [01]
;    0824  2                     DSQA$K_Dispat_DSX$EndPass_End,         !                                [01]
;    0825  2
;    0826  2                     DSQA$K_Error_Error_Bgn,                !                                [01]
;    0827  2
;    0828  2                     DSQA$K_Loop_DSX$BgnSub,
;    0829  2                     DSQA$K_Loop_DSX$EndSub,
;    0830  2                     DSQA$K_Loop_DSX$CkLoop,
;    0831  2
;    0832  2                     DSQA$K_QA_DSX$Branch]:
;    0833  3                            BEGIN
;    0834  4                            RETURN (QA$K_Success)
;    0835  2                            END;
;    0836  2
;    0837  2                    [OTHERWISE]:
;    0838  3                            BEGIN
;    0839  4                            Logic_Error ('Loop on Subtest, Illegal Hook_Point')
;    0840  2                            END;
;    0841  2                    TES;
;    0842  3             RETURN (QA$K_Failure)
;    0843  1     END;


                                                    .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 3C  00323 P.AAU:  .ASCII  \<!/**QA Internal Error: Loop on Subtest,\  ;
6F 20 70 6F 6F 4C 20 3A 72 6F 72 72 45 20 6C  00332                                                          ;
                  2C 74 73 65 74 62 75 53 20 6E  00341                                                          ;
50 5F 6B 6F 6F 48 20 6C 61 67 65 6C 6C 49 20  0034B          .ASCII  \ Illegal Hook_Point!/\                  ;
                     2F 21 74 6E 69 6F          0035A                                                          ;

                                                ES=                 P.AAU

                                                    .PSECT  CODE,NOWRT,  SHR,2

                                          0FFC 00000        .ENTRY  QA$LOOP_ON_SUBTEST, Save R2,R3,R4,R5,R6,R7,-; 0681
                                                            R8,R9,R10,R11                                       ;
                                5B 00000000G   EF  9E 00002         MOVAB   QA$AOB_CHECK_STATE, R11             ;
                                5A 00000000G   EF  9E 00009         MOVAB   DS$GL_FSTTEST, R10                  ;
                                59 00000000G   EF  9E 00010         MOVAB   DS$GL_SUBTEST, R9                   ;
                                58 00000000G   EF  9E 00017         MOVAB   QA$W_SAVE_TEST, R8                  ;
                                57 00000000G   EF  9E 0001E         MOVAB   QA$AOB_FLAGS, R7                    ;
                                56 00000000G   EF  9E 00025         MOVAB   QA$AOB_ROUTINE_STATE, R6            ;
                                55 00000000G   9F  9E 0002C         MOVAB   @#DS$PRINTF, R5                     ;
                                54 00000000'   EF  9E 00033         MOVAB   QA$T_HEADER_1, R4                   ;
                                53 00000000G   EF  9E 0003A         MOVAB   DS$GL_CLIBASE+32, R3                ;
                                52       04    AC  D0 00041         MOVL    HOOK_POINT, R2                      ; 0720
                                12             52  D1 00045         CMPL    R2, #18                             ; 0722
```

```
                                    59  12 00048          BNEQ    3$                                          ;
          3B            67          05  E1 0004A          BBC     #5, QA$AOB_FLAGS, 1$                        ; 0729
                                    54  DD 0004E          PUSHL   R4                                          ; 0731
                        65          01  FB 00050          CALLS   #1, DS$PRINTF                               ;
                        7E 0000020C 9F  7D 00053          MOVQ    @#^X0000020C, -(SP)                         ; 0732
                           00000208 9F  DD 0005A          PUSHL   @#^X00000208                                ;
                              26    A4  9F 00060          PUSHAB  QA$T_HEADER_2                               ;
                        65          04  FB 00063          CALLS   #4, DS$PRINTF                               ;
                                    7E  D4 00066          CLRL    -(SP)                                       ; 0733
                              4D    A4  9F 00068          PUSHAB  QA$T_HEADER_3                               ;
                        65          02  FB 0006B          CALLS   #2, DS$PRINTF                               ;
                            00A6    C4  9F 0006E          PUSHAB  HEADER_LS                                   ; 0734
                        65          01  FB 00072          CALLS   #1, DS$PRINTF                               ;
                        66          01  88 00075          BISB2   #1, QA$AOB_ROUTINE_STATE                    ; 0736
                        67    60    8F  8A 00078          BICB2   #96, QA$AOB_FLAGS                           ; 0737
                        63          68  32 0007C          CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+32            ; 0744
                  04    A3          68  32 0007F          CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+36            ; 0745
                  08    A3          01  D0 00083          MOVL    #1, DS$GL_CLIBASE+40                        ; 0746
                                    10  11 00087          BRB     2$                                          ; 0747
                        67    40    8F  88 00089 1$:      BISB2   #64, QA$AOB_FLAGS                           ; 0770
     08   A3            69          01  C1 0008D          ADDL3   #1, DS$GL_SUBTEST, DS$GL_CLIBASE+40         ; 0772
                        63          6A  D0 00092          MOVL    DS$GL_FSTTEST, DS$GL_CLIBASE+32             ; 0773
                  04    A3          6A  D0 00095          MOVL    DS$GL_FSTTEST, DS$GL_CLIBASE+36             ; 0774
                  0C    A3 00000000G EF 3C 00099 2$:      MOVZWL  QA$W_SUBTESTLOOPS, DS$GL_CLIBASE+44         ; 0747
                                    78  11 000A1          BRB     11$                                         ; 0777
                        05          52  D1 000A3 3$:      CMPL    R2, #5                                      ; 0780
                                    25  12 000A6          BNEQ    6$                                          ;
          06            66          02  E1 000A8          BBC     #2, QA$AOB_ROUTINE_STATE, 4$                ; 0782
                            FEB4    C4  9F 000AC          PUSHAB  CLEANUP_ERROR_MESSAGE                       ; 0783
                                    04  11 000B0          BRB     5$                                          ;
                            FEE4    C4  9F 000B2 4$:      PUSHAB  ERROR_REPORTED_MESSAGE                      ; 0785
                        65          01  FB 000B6 5$:      CALLS   #1, DS$PRINTF                               ;
                              89    A4  9F 000B9          PUSHAB  INFINITE_SUBTEST_MESSAGE                    ; 0787
                        65          01  FB 000BC          CALLS   #1, DS$PRINTF                               ;
                                    03  DD 000BF          PUSHL   #3                                          ; 0789
           00000000G    EF          01  FB 000C1          CALLS   #1, QA$ADD_QA_ERROR                         ;
                        66          02  88 000C8          BISB2   #2, QA$AOB_ROUTINE_STATE                    ; 0790
                                    59  11 000CB          BRB     13$                                         ;
                                    52  D5 000CD 6$:      TSTL    R2                                          ; 0793
                                    1A  12 000CF          BNEQ    7$                                          ;
                        63          68  32 000D1          CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+32            ; 0800
                  04    A3 00000000G EF 32 000D4          CVTWL   QA$W_SAVE_LAST, DS$GL_CLIBASE+36            ; 0801
                        6B          08  8A 000DC          BICB2   #8, QA$AOB_CHECK_STATE                      ; 0803
                        6B          10  88 000DF          BISB2   #16, QA$AOB_CHECK_STATE                     ; 0804
                        66          08  88 000E2          BISB2   #8, QA$AOB_ROUTINE_STATE                    ; 0805
                        67    40    8F  8A 000E5          BICB2   #64, QA$AOB_FLAGS                           ; 0806
                                    30  11 000E9          BRB     11$                                         ; 0807
                        18          52  D1 000EB 7$:      CMPL    R2, #24                                     ; 0810
                                    04  12 000EE          BNEQ    8$                                          ;
                                    69  D4 000F0          CLRL    DS$GL_SUBTEST                               ; 0812
                                    27  11 000F2          BRB     11$                                         ; 0813
                                    52  D5 000F4 8$:      TSTL    R2                                          ; 0820
                                    05  15 000F6          BLEQ    9$                                          ;
                        02          52  D1 000F8          CMPL    R2, #2                                      ;
                                    1E  15 000FB          BLEQ    11$                                         ;
                        06          52  D1 000FD 9$:      CMPL    R2, #6                                      ;
                                    19  13 00100          BEQL    11$                                         ;
```

```
                    09          52  D1 00102          CMPL    R2, #9                          ;
                                05  19 00105          BLSS    10$                             ;
                    0B          52  D1 00107          CMPL    R2, #11                         ;
                                0F  15 0010A          BLEQ    11$                             ;
                    11          52  D1 0010C 10$:     CMPL    R2, #17                         ;
                                0A  13 0010F          BEQL    11$                             ;
                    13          52  D1 00111          CMPL    R2, #19                         ;
                                09  19 00114          BLSS    12$                             ;
                    15          52  D1 00116          CMPL    R2, #21                         ;
                                04  14 00119          BGTR    12$                             ;
                    50          01  D0 0011B 11$:     MOVL    #1, R0                          ; 0834
                                    04 0011E          RET                                     ;
                          0194  C4  9F 0011F 12$:     PUSHAB  ES                              ; 0839
                    65          01  FB 00123          CALLS   #1, DS$PRINTF                   ;
                                50  D4 00126 13$:     CLRL    R0                              ; 0842
                                    04 00128          RET                                     ; 0843

; Routine Size:  297 bytes,    Routine Base:  CODE + 029C
```

```
;    0845  1      %SBTTL 'QA$Run_Backwards'
;    0846  1
;    0847  1      GLOBAL ROUTINE QA$Run_Backwards (Hook_Point) =
;    0848  1
;    0849  1      !++
;    0850  1      ! FUNCTIONAL DESCRIPTION:
;    0851  1      !
;    0852  1      !       This routine performs the QA Run Backwards check.
;    0853  1      !
;    0854  1      ! CALLER(S):
;    0855  1      !
;    0856  1      !       QA$Main
;    0857  1      !
;    0858  1      ! FORMAL PARAMETERS:
;    0859  1      !
;    0860  1      !       Hook_Point : The point in the Supervisor from which this routine
;    0861  1      !                    was called. This is a constant value. The constants are
;    0862  1      !                    defined in the DSQA macro.
;    0863  1      !
;    0864  1      ! IMPLICIT INPUTS:
;    0865  1      !
;    0866  1      !       NONE
;    0867  1      !
;    0868  1      ! IMPLICIT OUTPUTS:
;    0869  1      !
;    0870  1      !       NONE
;    0871  1      !
;    0872  1      ! COMPLETION CODES:
;    0873  1      !
;    0874  1      !       0 - QA error detected; unsuccessful return
;    0875  1      !       1 - No QA errors; successful return
;    0876  1      !
;    0877  1      ! SIDE EFFECTS:
;    0878  1      !
;    0879  1      !       NONE
;    0880  1      !
;    0881  1      !--
```

```
;    0883  2     BEGIN
;    0884  2             SELECTONE .Hook_Point OF
;    0885  2                     SET
;    0886  2                     [DSQA$K_Start_VRReStart]:               !                                       [01]
;    0887  3                             BEGIN
;    0888  3                             !+
;    0889  3                             ! The initializations required for this check and that are specific to this check
;    0890  3                             ! must be done here.
;    0891  3                             !-
;    0892  3
;    0893  3                             $DS_PRINTF (QA$T_Header_1);
;    0894  3                             $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    0895  3                             $DS_PRINTF (QA$T_Header_3, 0);
;    0896  3                             $DS_PRINTF (Header_RB);
;    0897  3
;    0898  3                             QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done init'ing
;    0899  4                             RETURN (QA$K_Success)
;    0900  2                             END;
;    0901  2
;    0902  2                     [DSQA$K_Error_Error_End]:
;    0903  3                             BEGIN
;    0904  3                             IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN    !               [01]
;    0905  4                                     $DS_PRINTF (Cleanup_Error_Message)      !                       [01]
;    0906  3                             ELSE                                            !                       [01]
;    0907  3                                     $DS_PRINTF (Error_Reported_Message);    !                       [01]
;    0908  3
;    0909  3                             $DS_PRINTF (Run_Backwards_Message);             !                       [02]
;    0910  3
;    0911  3                             QA$Add_QA_Error (QA$K_Run_Backwards);
;    0912  3                             QA$AOB_Routine_State [QA$V_Error_Found] = True
;    0913  2                             END;
;    0914  2
;    0915  2                     [DSQA$K_Dispat_DSX$EndPass_Mid]:        !                                       [01]
;    0916  3                             BEGIN
;    0917  3                             QA$AOB_Check_State [QA$K_Run_Backwards]   = Off;
;    0918  3                             QA$AOB_Routine_State [QA$V_Check_Done]    = True;
;    0919  3                             QA$AOB_Flags [QA$K_First_Time]            = True;       ! For the next check    [01]
;    0920  4                             RETURN (QA$K_Success)
;    0921  2                             END;
;    0922  2
;    0923  2                     [DSQA$K_Dispat_After_Init]:             !                                       [01]
;    0924  3                             BEGIN                           !                                       [01]
;    0925  3                             DSA$GL_TESTNO = .DS$GL_LSTTEST; ! Start at the last requested test number (i.e.,  [01]
;    0926  3                                                             ! at 'y' in /TEST:x:y), or else at L$A_TSTCNT.  [01]
;    0927  4                             RETURN (QA$K_Success)           !                                       [01]
;    0928  2                             END;
```

```
;   0930  2                     !+
;   0931  2                     ! The following are not used in this check.
;   0932  2                     !-
;   0933  2
;   0934  2                     [DSQA$K_Dispat_Before_Init,
;   0935  2                      DSQA$K_Dispat_CallTest,                 !                                                  [01]
;   0936  2                      DSQA$K_Dispat_DSX$EndPass_Bgn,          !                                                  [01]
;   0937  2                      DSQA$K_Dispat_DSX$EndPass_End,          !                                                  [01]
;   0938  2                      DSQA$K_Dispat_Done_Tests,               !                                                  [01]
;   0939  2
;   0940  2                      DSQA$K_Error_Error_Bgn,                 !                                                  [01]
;   0941  2
;   0942  2                      DSQA$K_Loop_DSX$BgnSub,
;   0943  2                      DSQA$K_Loop_DSX$EndSub,
;   0944  2                      DSQA$K_Loop_DSX$CkLoop,
;   0945  2
;   0946  2                      DSQA$K_QA_DSX$Branch]:
;   0947  3                             BEGIN
;   0948  4                             RETURN (QA$K_Success)
;   0949  2                             END;
;   0950  2
;   0951  2                     [OTHERWISE]:
;   0952  3                             BEGIN
;   0953  4                             Logic_Error ('Run Backwards, Illegal Hook_Point')
;   0954  2                             END;
;   0955  2                     TES;
;   0956  3             RETURN (QA$K_Failure)
;   0957  1     END;


                                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61  6E  72  65  74  6E  49  20  41  51  2A  2A  2F  21  3A  00360 P.AAV:  .ASCII  \:!/**QA Internal Error: Run Backwards, I\  ;
61  42  20  6E  75  52  20  3A  72  6F  72  72  45  20  6C  0036F                                                            ;
                    49  20  2C  73  64  72  61  77  6B  63  0037E                                                            ;
69  6F  50  5F  6B  6F  6F  48  20  6C  61  67  65  6C  6C  00388          .ASCII  \llegal Hook_Point!/\                       ;
                                            2F  21  74  6E  00397                                                            ;

                                                                    ES=                 P.AAV


                                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                        003C 00000          .ENTRY  QA$RUN_BACKWARDS, Save R2,R3,R4,R5          ; 0847
                                    55 00000000G  EF  9E 00002          MOVAB   QA$AOB_ROUTINE_STATE, R5                    ;
                                    54 00000000G  9F  9E 00009          MOVAB   @#DS$PRINTF, R4                             ;
                                    53 00000000'  EF  9E 00010          MOVAB   QA$T_HEADER_1, R3                           ;
                                    52       04   AC  D0 00017          MOVL    HOOK_POINT, R2                              ; 0884
                                    12            52  D1 0001B          CMPL    R2, #18                                     ; 0886
                                                  2C  12 0001E          BNEQ    1$                                          ;
                                                  53  DD 00020          PUSHL   R3                                          ; 0893
                                    64            01  FB 00022          CALLS   #1, DS$PRINTF                               ;
                                    7E 0000020C   9F  7D 00025          MOVQ    @#^X0000020C, -(SP)                         ; 0894
                                       00000208   9F  DD 0002C          PUSHL   @#^X00000208                                ;
                                             26   A3  9F 00032          PUSHAB  QA$T_HEADER_2                               ;
                                    64            04  FB 00035          CALLS   #4, DS$PRINTF                               ;
```

```
                                    7E  D4 00038          CLRL    -(SP)                                  ; 0895
                              4D    A3  9F 0003A          PUSHAB  QA$T_HEADER_3                          ;
                      64            02  FB 0003D          CALLS   #2, DS$PRINTF                          ;
                            00C8    C3  9F 00040          PUSHAB  HEADER_RB                              ; 0896
                      64            01  FB 00044          CALLS   #1, DS$PRINTF                          ;
                      65            01  88 00047          BISB2   #1, QA$AOB_ROUTINE_STATE               ; 0898
                                    7F  11 0004A          BRB     9$                                     ; 0899
                      05            52  D1 0004C 1$:      CMPL    R2, #5                                 ; 0902
                                    25  12 0004F          BNEQ    4$                                     ;
          06          65            02  E1 00051          BBC     #2, QA$AOB_ROUTINE_STATE, 2$           ; 0904
                            FEB4    C3  9F 00055          PUSHAB  CLEANUP_ERROR_MESSAGE                  ; 0905
                                    04  11 00059          BRB     3$                                     ;
                            FEE4    C3  9F 0005B 2$:      PUSHAB  ERROR_REPORTED_MESSAGE                 ; 0907
                      64            01  FB 0005F 3$:      CALLS   #1, DS$PRINTF                          ;
                              BE    A3  9F 00062          PUSHAB  RUN_BACKWARDS_MESSAGE                  ; 0909
                      64            01  FB 00065          CALLS   #1, DS$PRINTF                          ;
                                    04  DD 00068          PUSHL   #4                                     ; 0911
             00000000G  EF          01  FB 0006A          CALLS   #1, QA$ADD_QA_ERROR                    ;
                      65            02  88 00071          BISB2   #2, QA$AOB_ROUTINE_STATE               ; 0912
                                    60  11 00074          BRB     11$                                    ;
                                    52  D5 00076 4$:      TSTL    R2                                     ; 0915
                                    13  12 00078          BNEQ    5$                                     ;
             00000000G  EF          10  8A 0007A          BICB2   #16, QA$AOB_CHECK_STATE                ; 0917
                      65            08  88 00081          BISB2   #8, QA$AOB_ROUTINE_STATE               ; 0918
             00000000G  EF          20  88 00084          BISB2   #32, QA$AOB_FLAGS                      ; 0919
                                    3E  11 0008B          BRB     9$                                     ; 0920
                      14            52  D1 0008D 5$:      CMPL    R2, #20                                ; 0923
                                    0D  12 00090          BNEQ    6$                                     ;
             0000FE50   9F 00000000G EF  D0 00092          MOVL    DS$GL_LSTTEST, @#^X0000FE50            ; 0925
                                    2C  11 0009D          BRB     9$                                     ; 0927
                                    52  D5 0009F 6$:      TSTL    R2                                     ; 0934
                                    05  15 000A1          BLEQ    7$                                     ;
                      02            52  D1 000A3          CMPL    R2, #2                                 ;
                                    23  15 000A6          BLEQ    9$                                     ;
                      06            52  D1 000A8 7$:      CMPL    R2, #6                                 ;
                                    1E  13 000AB          BEQL    9$                                     ;
                      09            52  D1 000AD          CMPL    R2, #9                                 ;
                                    05  19 000B0          BLSS    8$                                     ;
                      0B            52  D1 000B2          CMPL    R2, #11                                ;
                                    14  15 000B5          BLEQ    9$                                     ;
                      11            52  D1 000B7 8$:      CMPL    R2, #17                                ;
                                    0F  13 000BA          BEQL    9$                                     ;
                      13            52  D1 000BC          CMPL    R2, #19                                ;
                                    0A  13 000BF          BEQL    9$                                     ;
                      15            52  D1 000C1          CMPL    R2, #21                                ;
                                    05  13 000C4          BEQL    9$                                     ;
                      18            52  D1 000C6          CMPL    R2, #24                                ;
                                    04  12 000C9          BNEQ    10$                                    ;
                      50            01  D0 000CB 9$:      MOVL    #1, R0                                 ; 0948
                                        04 000CE          RET                                            ;
                            01D1    C3  9F 000CF 10$:     PUSHAB  ES                                     ; 0953
                      64            01  FB 000D3          CALLS   #1, DS$PRINTF                          ;
                                    50  D4 000D6 11$:     CLRL    R0                                     ; 0956
                                        04 000D8          RET                                            ; 0957

; Routine Size:  217 bytes,    Routine Base:  CODE + 03C5
```

```
;    0959  1      !%SBTTL 'QA$Error_Phase_One'
;    0960  1      !
;    0961  1      !GLOBAL ROUTINE QA$Error_Phase_One (Hook_Point) =       !                                    Start of [1]
;    0962  1      !
;    0963  1      !!++
;    0964  1      !! FUNCTIONAL DESCRIPTION:
;    0965  1      !!
;    0966  1      !!      This routine performs Phase One of the QA Error check.
;    0967  1      !!
;    0968  1      !! CALLER(S):
;    0969  1      !!
;    0970  1      !!      QA$Main
;    0971  1      !!
;    0972  1      !! FORMAL PARAMETERS:
;    0973  1      !!
;    0974  1      !!      Hook_Point : The point in the Supervisor from which this routine
;    0975  1      !!                   was called. This is a constant value. The constants are
;    0976  1      !!                   defined in the DSQA macro.
;    0977  1      !!
;    0978  1      !! IMPLICIT INPUTS:
;    0979  1      !!
;    0980  1      !!      NONE
;    0981  1      !!
;    0982  1      !! IMPLICIT OUTPUTS:
;    0983  1      !!
;    0984  1      !!      NONE
;    0985  1      !!
;    0986  1      !! COMPLETION CODES:
;    0987  1      !!
;    0988  1      !!      0 - QA error detected; unsuccessful return
;    0989  1      !!      1 - No QA errors; successful return
;    0990  1      !!
;    0991  1      !! SIDE EFFECTS:
;    0992  1      !!
;    0993  1      !!--
```

```
;    0994  1      !
;    0995  1      !BEGIN
;    0996  1      !       OWN
;    0997  1      !               Abort_Now : BYTE,                               ! When 1, abort the diagnostic
;    0998  1      !               Last_Branch_PC;                                 ! The PC of the last branch macro executed
;    0999  1      !
;    1000  1      !       MAP
;    1001  1      !               DS$GL_CLIBASE : BLOCK [, BYTE];
;    1002  1      !
;    1003  1      !       LITERAL                                                 ! Used to set the Last_Branch_PC so as to insure
;    1004  1      !               QA$K_Bad_PC = xX'FFFFFFFF';                     ! that the PC is not found.
;    1005  1      !
;    1006  1      !       SELECTONE .Hook_Point OF
;    1007  1      !               SET
;    1008  1      !               [DSQA$K_Start_VRReStart]:
;    1009  1      !                       BEGIN
;    1010  1      !                       IF (.QA$AOB_Flags [QA$K_First_Time]) THEN
;    1011  1      !                               BEGIN
;    1012  1      !                               $DS_PRINTF (QA$T_Header_1);
;    1013  1      !                               $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);
;    1014  1      !                               $DS_PRINTF (QA$T_Header_3, 0);                  ! 0 is for the current time
;    1015  1      !                               $DS_PRINTF (Header_E1);
;    1016  1      !
;    1017  1      !                               Last_Branch_PC = 0;                             ! Zero means do next error call
;    1018  1      !                               Abort_Now = False;                              ! Do not abort on next one
;    1019  1      !                               QA$AOB_Flags [QA$K_First_Time] = False;         ! Not first time anymore
;    1020  1      !                               QA$AOB_Flags [QA$K_No_Header]  = False;         ! Print the header only once
;    1021  1      !
;    1022  1      !                               DS$GL_CLIBASE [CLI$L_PASS] = 1;                 ! One pass
;    1023  1      !                               DS$GL_CLIBASE [CLI$L_SUBT] = 0;                 ! No looping on subtests
;    1024  1      !                               DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;   ! First test number
;    1025  1      !                               DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last;   ! Last test number
;    1026  1      !
;    1027  1      !                               QA$AOB_Routine_State [QA$V_Init_Done] = True;   ! Done initializing
;    1028  1      !                               END
;    1029  1      !                       ELSE
;    1030  1      !                               BEGIN
;    1031  1      !                               QA$AOB_Flags [QA$K_No_Header] = True;           ! Print no more header till done
;    1032  1      !
;    1033  1      !                               DS$GL_CLIBASE [CLI$L_PASS] = 1;                 ! One pass
;    1034  1      !                               DS$GL_CLIBASE [CLI$L_SUBT] = 0;                 ! No looping on subtests
;    1035  1      !                               DS$GL_CLIBASE [CLI$L_TEST] = .DSA$GL_TestNo;    ! Set to same test number
;    1036  1      !                               DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last;   ! Last test number
;    1037  1      !                               END;
;    1038  1      !                       RETURN (QA$K_Success)
;    1039  1      !                       END;
```

```
;    1040  1      !
;    1041  1      !                  [DSQA$K_Error_Error_End]:
;    1042  1      !                          BEGIN
;    1043  1      !                          IF ((.Abort_Now) AND (NOT .QA$AOB_Routine_State [QA$V_Doing_Cleanup])) THEN
;    1044  1      !                                  BEGIN
;    1045  1      !                                  MAP
;    1046  1      !                                          DSA$GL_ERRNO : WORD;                    ! In the DS, it is a word.
;    1047  1      !                                  !+
;    1048  1      !                                  ! This error call was forced. Add the error number to the forced error summary table
;    1049  1      !                                  ! and abort the diagnostic. This will cause the DSX$Abort routine to branch back to
;    1050  1      !                                  ! the VRReStart routine, and start over again. Note: use the error number that the
;    1051  1      !                                  ! RRepError routine stores in DSA$GL_ErrNo, rather than that passes to the Branch
;    1052  1      !                                  ! routine. The former number is more accurate.
;    1053  1      !                                  !-
;    1054  1      !                                  QA$Add_Forced_Error (.DSA$GL_ERRNO);            ! Add another forced error
;    1055  1      !                                  Abort_Now = False;                              ! Reset for next time
;    1056  1      !                                  DSX$Abort ();                                   ! Abort the diagnostic
;    1057  1      !                                  !+
;    1058  1      !                                  ! The above Abort will not come back to here.
;    1059  1      !                                  !-
;    1060  1      !                                  END
;    1061  1      !                          ELSE
;    1062  1      !                                  BEGIN
;    1063  1      !                                  !+
;    1064  1      !                                  ! This error call was not forced. Therefore, it is a QA error. Print the
;    1065  1      !                                  ! appropriate error message and return Failure.
;    1066  1      !                                  !-
;    1067  1      !                                  IF (.QA$AOB_Routine_State [QA$V_Doing_Cleanup]) THEN
;    1068  1      !                                          $DS_PRINTF (Cleanup_Error_Message)
;    1069  1      !                                  ELSE
;    1070  1      !                                          $DS_PRINTF (Error_Reported_Message);
;    1071  1      !                                  QA$Add_QA_Error (QA$K_Error_Phase_One);
;    1072  1      !                                  QA$AOB_Routine_State [QA$V_Error_Found] = True;
;    1073  1      !                                  END;
;    1074  1      !                          !+
;    1075  1      !                          ! Return Failure.
;    1076  1      !                          !-
;    1077  1      !                          END;
;    1078  1      !
;    1079  1      !                  [DSQA$K_Dispat_Done_Tests]:
;    1080  1      !                          BEGIN
;    1081  1      !                          !+
;    1082  1      !                          ! We have done all the tests. Set Abort_Now to False so that the error macros in
;    1083  1      !                          ! the initialization code are not done again.
;    1084  1      !                          !-
;    1085  1      !                          Abort_Now      = False;
;    1086  1      !                          Last_Branch_PC = QA$K_Bad_PC;
;    1087  1      !                          RETURN (QA$K_Success)
;    1088  1      !                          END;
```

```
;    1089  1      !
;    1090  1      !                   [DSQA$K_Dispat_DSX$EndPass_Mid]:
;    1091  1      !                           BEGIN
;    1092  1      !                           QA$AOB_Check_State [QA$K_Error_Phase_One]  = Off;       ! Stop Phase One check.
;    1093  1      !                           QA$AOB_Check_State [QA$K_Error_Phase_Two]  = On;        ! Start Phase Two check.
;    1094  1      !                           QA$AOB_Flags [QA$K_No_Header]              = False;     ! Print headers again.
;    1095  1      !
;    1096  1      !                           QA$Print_Forced_Errors ();                              ! Print the table.
;    1097  1      !
;    1098  1      !                           QA$AOB_Routine_State [QA$V_Check_Done]     = True;      ! Indicate end of this check.
;    1099  1      !                           RETURN (QA$K_Success)
;    1100  1      !                           END;
;    1101  1      !
;    1102  1      !                   [DSQA$K_QA_DSX$Branch]:
;    1103  1      !                           BEGIN
;    1104  1      !                           LOCAL
;    1105  1      !                                   Frame_Start,
;    1106  1      !                                   Saved_PC;
;    1107  1      !
;    1108  1      !                           IF (NOT QA$Find_User_Call_Frame (Frame_Start)) THEN
;    1109  1      !                                   RETURN (QA$K_Failure);
;    1110  1      !
;    1111  1      !                           BEGIN
;    1112  1      !                                   BIND
;    1113  1      !                                           Call_Frame = .Frame_Start : BLOCK [, BYTE];
;    1114  1      !
;    1115  1      !                                   Saved_PC = .Call_Frame [SF$L_SAVE_PC]
;    1116  1      !                           END;
;    1117  1      !
;    1118  1      !                           IF (.Last_Branch_PC EQL 0) THEN
;    1119  1      !                                   BEGIN
;    1120  1      !                                   Last_Branch_PC = .Saved_PC;
;    1121  1      !                                   Abort_Now      = True;
;    1122  1      !                                   QA$W_Branch    = Boolean (NOT .QA$W_Branch);
;    1123  1      !                                   END
;    1124  1      !                           ELSE IF (.Last_Branch_PC EQL .Saved_PC) THEN
;    1125  1      !                                   BEGIN
;    1126  1      !                                   Last_Branch_PC = 0;
;    1127  1      !                                   END;
;    1128  1      !                           !+
;    1129  1      !                           ! ELSE:
;    1130  1      !                           !       I have not yet found the correct branch macro. Keep looking.
;    1131  1      !                           !-
;    1132  1      !                           RETURN (QA$K_Success)
;    1133  1      !                           END;
```

```
;    1134  1      !
;    1135  1      !                   !+
;    1136  1      !                   ! The following are not used in this check. So, just return Success for them.
;    1137  1      !                   !-
;    1138  1      !
;    1139  1      !                   [DSQA$K_Dispat_Before_Init,
;    1140  1      !                    DSQA$K_Dispat_After_Init,
;    1141  1      !                    DSQA$K_Dispat_CallTest,
;    1142  1      !                    DSQA$K_Dispat_DSX$EndPass_Bgn,          !                                    [01]
;    1143  1      !                    DSQA$K_Dispat_DSX$EndPass_End,          !                                    [01]
;    1144  1      !
;    1145  1      !                    DSQA$K_Error_Error_Bgn,                 !                                    [01]
;    1146  1      !
;    1147  1      !                    DSQA$K_Loop_DSX$BgnSub,
;    1148  1      !                    DSQA$K_Loop_DSX$EndSub,
;    1149  1      !                    DSQA$K_Loop_DSX$CkLoop]:
;    1150  1      !                           BEGIN
;    1151  1      !                           RETURN (QA$K_Success)
;    1152  1      !                           END;
;    1153  1      !
;    1154  1      !                   [OTHERWISE]:
;    1155  1      !                           BEGIN
;    1156  1      !                           Logic_Error ('Error Phase One, Illegal Hook_Point')
;    1157  1      !                           END;
;    1158  1      !                   TES;
;    1159  1      !
;    1160  1      !           !+
;    1161  1      !           ! Come to here only if a QA error was found. Return QA$K_Failure to QA$Main, which then takes
;    1162  1      !           ! the appropriate action.
;    1163  1      !           !-
;    1164  1      !
;    1165  1      !           RETURN (QA$K_Failure)
;    1166  1      !END;
```

```
;    1167  1      !
;    1168  1      !xSBTTL 'QA$Error_Phase_Two'
;    1169  1      !
;    1170  1      !GLOBAL ROUTINE QA$Error_Phase_Two (Hook_Point) =
;    1171  1      !
;    1172  1      !!++
;    1173  1      !! FUNCTIONAL DESCRIPTION:
;    1174  1      !!
;    1175  1      !!      This routine performs Phase Two of the QA Error check.
;    1176  1      !!
;    1177  1      !! CALLER(S):
;    1178  1      !!
;    1179  1      !!      QA$Main
;    1180  1      !!
;    1181  1      !! FORMAL PARAMETERS:
;    1182  1      !!
;    1183  1      !!      Hook_Point : The point in the Supervisor from which this routine
;    1184  1      !!                   was called. This is a constant value. The constants are
;    1185  1      !!                   defined in the DSQA macro.
;    1186  1      !!
;    1187  1      !! IMPLICIT INPUTS:
;    1188  1      !!
;    1189  1      !!      NONE
;    1190  1      !!
;    1191  1      !! IMPLICIT OUTPUTS:
;    1192  1      !!
;    1193  1      !!      NONE
;    1194  1      !!
;    1195  1      !! COMPLETION CODES:
;    1196  1      !!
;    1197  1      !!      0 - QA error detected; unsuccessful return
;    1198  1      !!      1 - No QA errors; successful return
;    1199  1      !!
;    1200  1      !! SIDE EFFECTS:
;    1201  1      !!
;    1202  1      !!--
;    1203  1      !
;    1204  1      !BEGIN
;    1205  1      !       RETURN (QA$K_Failure)
;    1206  1      !END;
```

```
;    1207  1      !
;    1208  1      !%SBTTL 'QA$Error_Phase_Three'
;    1209  1      !
;    1210  1      !GLOBAL ROUTINE QA$Error_Phase_Three (Hook_Point) =
;    1211  1      !
;    1212  1      !!++
;    1213  1      !! FUNCTIONAL DESCRIPTION:
;    1214  1      !!
;    1215  1      !!      This routine performs Phase Three of the QA Error check.
;    1216  1      !!
;    1217  1      !! CALLER(S):
;    1218  1      !!
;    1219  1      !!      QA$Main
;    1220  1      !!
;    1221  1      !! FORMAL PARAMETERS:
;    1222  1      !!
;    1223  1      !!      Hook_Point : The point in the Supervisor from which this routine
;    1224  1      !!                   was called. This is a constant value. The constants are
;    1225  1      !!                   defined in the DSQA macro.
;    1226  1      !!
;    1227  1      !! IMPLICIT INPUTS:
;    1228  1      !!
;    1229  1      !!      NONE
;    1230  1      !!
;    1231  1      !! IMPLICIT OUTPUTS:
;    1232  1      !!
;    1233  1      !!      NONE
;    1234  1      !!
;    1235  1      !! COMPLETION CODES:
;    1236  1      !!
;    1237  1      !!      0 - QA error detected; unsuccessful return
;    1238  1      !!      1 - No QA errors; successful return
;    1239  1      !!
;    1240  1      !! SIDE EFFECTS:
;    1241  1      !!
;    1242  1      !!--
;    1243  1      !
;    1244  1      !BEGIN
;    1245  1      !       RETURN (QA$K_Failure)
;    1246  1      !END;
```

```
;   1248  1     %SBTTL 'End of Module QACHECKS'
;   1249  1
;   1250  0     END ELUDOM
```

```
;                                  PSECT SUMMARY
;
;       Name                    Bytes                          Attributes
;
;  DATA                           923  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          1182  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                             2  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...										
$ASCIC	Unbound		204	223									
	Macro	Lib01	255	259	262	269	272	275	278	281	290	293	296
			301	304	307	310	313	417	537	673	839	953	
$DS_DSADEF	Macro	Lib01	97										
$DS_PRINTF	Unbound		206	223									
	Macro	Lib01	358	359	360	361	370	372	374	417	475	476	477
			478	488	490	492	537	599	600	601	602	612	614
			616	673	731	732	733	734	783	785	787	839	893
			894	895	896	905	907	909	953				
$DS_QADEF	Macro	Lib01	96										
$EQULST	Macro	Lib03	96	98	99								
$ER	Comptime	245											
$MODULE	Bind	245											
ASSERT	Macro	179											
ASSERT_STRING	MacrForm	179	184										
BOOLEAN	Macro	236											
CLEANUP_ERROR_MESSAGE	Bind	258	370a	488a	612a	783a	905a						
CLI$L_LAST	Macro	Lib02	745	774	801								
CLI$L_PASS	Macro	Lib02	480	604	747	775							
CLI$L_SUBT	Macro	Lib02	746	772									
CLI$L_TEST	Macro	Lib02	744	773	800								
CONDITION	MacrForm	179	183										
DEBUG	Macro	218											
DEBUG_STRING	MacrForm	218	223										
DS$GL_CLIBASE	External	124	473m	480=	589m	604=	718m	744=	745=	746=	747=	772=	773=
			774=	775=	800=	801=							
DS$GL_FSTTEST	External	120	773.	774.									
DS$GL_LSTTEST	External	121	925.										
DS$GL_SUBTEST	External	122	772.	812=									
DS$K_BBC	Literal	96											
DS$K_BBCC	Literal	96											
DS$K_BBCCI	Literal	96											
DS$K_BBCS	Literal	96											
DS$K_BBS	Literal	96											

Symbol	Type	Defined	Referenced ...
DS$K_BBSC	Literal	96	
DS$K_BBSS	Literal	96	
DS$K_BBSSI	Literal	96	
DS$K_BCC_M	Literal	96	
DS$K_BCS_M	Literal	96	
DS$K_BEQLU_B	Literal	96	
DS$K_BEQLU_M	Literal	96	
DS$K_BEQL_B	Literal	96	
DS$K_BEQL_M	Literal	96	
DS$K_BERROR	Literal	96	
DS$K_BGEQU_B	Literal	96	
DS$K_BGEQU_M	Literal	96	
DS$K_BGEQ_B	Literal	96	
DS$K_BGEQ_M	Literal	96	
DS$K_BGTRU_B	Literal	96	
DS$K_BGTRU_M	Literal	96	
DS$K_BGTR_B	Literal	96	
DS$K_BGTR_M	Literal	96	
DS$K_BLBC	Literal	96	
DS$K_BLBS	Literal	96	
DS$K_BLEQU_B	Literal	96	
DS$K_BLEQU_M	Literal	96	
DS$K_BLEQ_B	Literal	96	
DS$K_BLEQ_M	Literal	96	
DS$K_BLSSU_B	Literal	96	
DS$K_BLSSU_M	Literal	96	
DS$K_BLSS_B	Literal	96	99
DS$K_BLSS_M	Literal	96	
DS$K_BNEQU_B	Literal	96	
DS$K_BNEQU_M	Literal	96	
DS$K_BNEQ_B	Literal	96	
DS$K_BNEQ_M	Literal	96	
DS$K_BNERROR	Literal	96	99
DS$K_BVC_M	Literal	96	
DS$K_BVS_M	Literal	96	
DS$K_DS_STARTING_LOCATION	Literal	99	
DS$PRINTF	ExtRout	358	358c 359e 359c 360e 360c 361e 361c 370e 370c 372e 372c 374e 374c 417e 417c 475e 475c 476e 476c 477e 477c 478e 478c 488e 488c 490e 490c 492e 492c 537e 537c 599e 599c 600e 600c 601e 601c 602e 602c 612e 612c 614e 614c 616e 616c 673e 673c 731e 731c 732e 732c 733e 733c 734e 734c 783e 783c 785e 785c 787e 787c 839e 839c 893e 893c 894e 894c 895e 895c 896e 896c 905e 905c 907e 907c 909e 909c 953e 953c
DSA$AL_APTMAIL	Bind	97	97
DSA$GL_APTCOM	Bind	97	
DSA$GL_DEVLEN	Bind	97	
DSA$GL_DEVNAM	Bind	97	
DSA$GL_ERRNO	Bind	97	
DSA$GL_EVENT	Bind	97	
DSA$GL_FLAGS	Bind	97	385
DSA$GL_MSGPTR	Bind	97	
DSA$GL_MSGTYP	Bind	97	
DSA$GL_PASSES	Bind	97	
DSA$GL_PASSNO	Bind	97	

Symbol	Type	Defined	Referenced ...				
DSA$GL_SECTNO	Bind	97					
DSA$GL_SID	Bind	97					
DSA$GL_SUBTNO	Bind	97					
DSA$GL_TESTNO	Bind	97	925				
DSA$GL_UNITS	Bind	97					
DSA$V_TRACE	Macro	Lib01	385				
DSQA	Macro	Lib02	98				
DSQA$K_DISPAT_AFTER_INIT	Literal	98	398	518	655	821	923
DSQA$K_DISPAT_BEFORE_INIT	Literal	98	397	517	654	820	934
DSQA$K_DISPAT_CALLTEST	Literal	98	399	519	635	822	935
DSQA$K_DISPAT_DONE_TESTS	Literal	98	402	522	658	810	938
DSQAK_DISPAT_DSXENDPASS_BGN	Literal	98	400	520	656	823	936
DSQAK_DISPAT_DSXENDPASS_END	Literal	98	401	521	657	824	937
DSQAK_DISPAT_DSXENDPASS_MID	Literal	98	383	501	622	793	915
DSQA$K_DISPAT_DS_CLEANUP_BGN	Literal	98					
DSQA$K_DISPAT_DS_CLEANUP_END	Literal	98					
DSQA$K_DUMMY_1	Literal	98					
DSQA$K_ERROR_ERROR_BGN	Literal	98	404	524	660	826	940
DSQA$K_ERROR_ERROR_END	Literal	98	367	485	609	780	902
DSQA$K_KERNEL_INIT_CONTEXT	Literal	98					
DSQAK_LOOP_DSXBGNSUB	Literal	98	406	526	662	828	942
DSQAK_LOOP_DSXCKLOOP	Literal	98	408	528	664	830	944
DSQAK_LOOP_DSXENDSUB	Literal	98	407	527	663	829	943
DSQAK_PARAM_DSXGETADDRESS	Literal	98					
DSQAK_PARAM_DSXGETDATA	Literal	98					
DSQAK_PARAM_DSXGETLOGICAL	Literal	98					
DSQAK_PARAM_DSXGETSTRING	Literal	98					
DSQAK_PARAM_DSXGETVIELD	Literal	98					
DSQAK_QA_DSXBRANCH	Literal	98	410	530	666	832	946
DSQA$K_START_BEFORE_HEADER	Literal	98					
DSQA$K_START_VRRESTART	Literal	98	356	470	586	722	886
DSQA$K_START_VRSTART	Literal	98					
DSX$ABORT	ExtRout	112					
DS_QADEFS	Macro	Lib02	99				
ERROR_REPORTED_MESSAGE	Bind	261	372a	490a	614a	785a	907a
ERROR_STRING	MacrForm	201	204				
ES	Unbound		204	206			
	Bind	417	417a				
	Bind	537	537a				
	Bind	673	673a				
	Bind	839	839a				
	Bind	953	953a				
EXPRESSION	MacrForm	236	237				
FALSE	Literal	99	645	737	738	806	
GET1ST_	Macro	Lib03	96	98	99		
GET2ND_	Unbound		96	98	99		
HEADER_LS	Bind	309	734a				
HEADER_LT	Bind	306	602a				
HEADER_MP	Bind	303	478a				
HEADER_NS	Bind	300	361a				
HEADER_RB	Bind	312	896a				
HOOK_POINT	RoutForm	317	354.				
	RoutForm	432	468.				
	RoutForm	545	584.				
	RoutForm	681	720.				

Symbol	Type	Defined	Referenced ...
	RoutForm	847	884.
INFINITE_SUBTEST_MESSAGE	Bind	277	787a
INFINITE_TEST_MESSAGE	Bind	274	616a
L$A_NAME	Literal	Lib01	359 476 600 732 894
L$L_REV	Literal	Lib01	359 476 600 732 894
L$L_UPDATE	Literal	Lib01	359 476 600 732 894
LOGIC_ERROR	Unbound		184
	Macro	201	417 537 673 839 953
LOOP_COUNTER	Own	582	596= 637. 639= 644= 644.
MODNAM	Macro	Lib02	245
MULTIPLE_PASS_MESSAGE	Bind	271	492a
NORMAL_START_MESSAGE	Bind	268	374a
NUL2ND_	Macro	Lib03	96 98 99
OFF	Literal	99	385 386 507 628 803 917
ON	Literal	99	387 508 629 804
QA$ADD_FORCED_ERROR	ExtRout	109	
QA$ADD_QA_ERROR	ExtRout	110	376c 494c 618c 789c 911c
QA$AOB_CHECK_STATE	External	135	386= 387= 507= 508= 628= 629= 803= 804= 917=
QA$AOB_FLAGS	Unbound		222
	External	136	597= 631= 640= 645= 729. 737= 738= 770= 806= 919=
QA$AOB_ROUTINE_STATE	External	134	363= 369. 377= 388= 481= 487. 495= 509= 605= 611. 619=
			630= 736= 782. 790= 805= 898= 904. 912= 918=
QA$FIND_USER_CALL_FRAME	ExtRout	107	
QA$K_ABORT_NOW	Literal	99	
QA$K_APT_PHASE_ONE	Literal	99	
QA$K_APT_PHASE_TWO	Literal	99	
QA$K_BAD_ADDRESS	Literal	99	
QA$K_CONTROL_C_CONT	Literal	99	
QA$K_DEALLOCATION	Literal	99	
QA$K_ERROR_PHASE_ONE	Literal	99	
QA$K_ERROR_PHASE_THREE	Literal	99	
QA$K_ERROR_PHASE_TWO	Literal	99	
QA$K_FAILURE	Literal	99	426 540 676 842 956
QA$K_FIRST_BRANCH_CODE	Literal	99	
QA$K_FIRST_TIME	Literal	99	631 729 737 919
QA$K_INIT_CONTEXT_DONE	Literal	99	
QA$K_LAST_BRANCH_CODE	Literal	99	
QA$K_LOOP_ON_SUBTEST	Literal	99	629 789 803
QA$K_LOOP_ON_TEST	Literal	99	508 618 628
QA$K_LOOP_TEST_DONE	Literal	99	597 640 645
QA$K_MEMORY_MANAGE	Literal	99	
QA$K_MULTIPLE_PASS	Literal	99	387 494 507
QA$K_NODEF	Literal	99	
QA$K_NORMAL_START	Literal	99	376 386
QA$K_NO_HEADER	Literal	99	738 770 806
QA$K_NUMBER_OF_CHECKS	Literal	99	
QA$K_NUMBER_QA_FLAGS	Literal	99	
QA$K_NUMB_ROUTINE_STATES	Literal	99	
QA$K_QA_DEBUG	Unbound		222
	Literal	99	
QA$K_QA_DONE	Literal	99	
QA$K_RUN_BACKWARDS	Literal	99	804 911 917
QA$K_SECTION	Literal	99	
QA$K_SUCCESS	Literal	99	364 389 412 482 510 532 606 632 647 668 777
			807 813 834 899 920 927 948

Symbol	Type	Defined	Referenced ...									
QA$LOOP_ON_SUBTEST	GlobRout	681	74f									
QA$LOOP_ON_TEST	GlobRout	545	73f									
QA$MULTIPLE_PASS	GlobRout	432	72f									
QA$NORMAL_START	GlobRout	317	71f									
QA$PRINT_FORCED_ERRORS	ExtRout	108										
QA$RUN_BACKWARDS	GlobRout	847	75f									
QA$T_HEADER_1	GlobBind	289	358a	475a	599a	731a	893a					
QA$T_HEADER_2	GlobBind	292	359a	476a	600a	732a	894a					
QA$T_HEADER_3	GlobBind	295	360a	477a	601a	733a	895a					
QA$T_OPERATOR_REQUEST_MESSAGE	GlobBind	254										
QA$V_CHECK_DONE	Literal	99	388	509	630	805	918					
QA$V_DOING_CLEANUP	Literal	99	369	487	611	782	904					
QA$V_ERROR_FOUND	Literal	99	377	495	619	790	912					
QA$V_INIT_DONE	Literal	99	363	481	605	736	898					
QA$W_BRANCH	External	138										
QA$W_ERROR_NUMBER	External	139										
QA$W_MULTIPLEPASS	External	128	480.									
QA$W_SAVE_LAST	External	141	801.									
QA$W_SAVE_TEST	External	140	744.	745.	800.							
QA$W_SUBTESTLOOPS	External	127	747.	775.								
QA$W_TESTLOOPS	External	126	637.									
RUN_BACKWARDS_MESSAGE	Bind	280	909a									
TRUE	Literal	99	363	377	388	481	495	509	597	605	619	630	631
			640	736	770	790	805	898	912	918	919		

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]QACHECKS.B32;1
6	Module QACHECKS	
83	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
86	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
89	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
1250	Eludom QACHECKS	

KEY TO REFERENCE TYPE FLAGS

- . Fetch
- = Store
- c Routine call
- a Address use
- a Indirect use
- e External, external routine, or external literal declaration
- f Forward or forward routine declaration
- h Condition handler enabling
- m Map declaration
- u Undeclare declaration

Library Statistics

;
;

```
;                                        -------- Symbols --------      Pages      Processing
;       File                             Total    Loaded   Percent      Mapped     Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                          940         8         0          96        00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                          675         7         1          47        00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2          20450         3         0        1101        00:00.8
```

```
;                                  COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:QACHECKS.B32

; Size:         1182 code + 925 data bytes
; Run Time:        00:05.8
; Elapsed Time:    00:08.4
; Lines/CPU Min:   13043
; Lexemes/CPU-Min:115147
; Memory Used:  174 pages
; Compilation Complete
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element QAFLAGS.B32
;    0002  0      ! *1     6-FEB-1986 15:22:00 DS ""
;    0003  0      ! DEC/CMS REPLACEMENT HISTORY, Element QAFLAGS.B32
;    0004  0      %TITLE '*** Setting and Showing QA Flags'
;    0005  0      MODULE QAFLAGS (
;    0006  0                IDENT = '6.6-04'
;    0007  0                ) =
;    0008  0      !
;    0009  0      ! COPYRIGHT (c) 1979, 1981, 1983
;    0010  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0011  0      !
;    0012  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0013  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0014  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0015  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0016  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0017  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0018  0      ! REMAIN IN DEC.
;    0019  0      !
;    0020  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0021  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0022  0      ! CORPORATION.
;    0023  0      !
;    0024  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0025  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0026  0
;    0027  0      !++
;    0028  0      ! FACILITY: DIAGNOSTIC SUPERVISOR
;    0029  0      !
;    0030  0      ! ABSTRACT:
;    0031  0      !
;    0032  0      !       This module contains the routines required to SET and SHOW the
;    0033  0      !       various QA flags.
;    0034  0      !
;    0035  0      ! AUTHOR:
;    0036  0      !
;    0037  0      !       Jack Stansbury
;    0038  0      !
;    0039  0      ! CREATION DATE:
;    0040  0      !
;    0041  0      !       25-October-1981
;    0042  0      !
;    0043  0      ! MODIFIED:
;    0044  0      !
;    0045  0      !       01      Jack Stansbury, 21-Nov-1981, Version 6.5
;    0046  0      !               Added INITIAL attributes to the flag word storage.
;    0047  0      !
;    0048  0      !       02      Jack Stansbury, 14-Jan-1982, Version 6.6
;    0049  0      !               Changed the lower limit on the QASUBTESTLOOPS DS flag from
;    0050  0      !               1 to 2. This makes the Loop_On_Subtest routine much easier.
;    0051  0      !               Also changed the Out of Range message to say "greater than or
;    0052  0      !               equal to" rather than just "greater than".
;    0053  0      !
;    0054  0      !       03      Jack Stansbury, 20-Jan-1983, Version 6.11
;    0055  0      !               Took out all references to CkLoopLoops and ErrorPrints.
;    0056  0      !
;    0057  0      !       04      Bob Bergazzi    May 17, 1983    Version 6.11
```

```
;   0058  0     !                   Changed the order of searching libraries.
;   0059  0     !--
```

```
;    0061  1     BEGIN
;    0062  1     %SBTTL  'Libraries'
;    0063  1     !
;    0064  1     ! LIBRARIES:
;    0065  1     !
;    0066  1
;    0067  1     LIBRARY '$DIAG';                    !                                   [04]
;    0068  1
;    0069  1     LIBRARY '$DS';                      !                                   [04]
;    0070  1
;    0071  1     LIBRARY 'SYS$LIBRARY:LIB';
```

```
;    0073  1      %SBTTL  'Table of Contents'
;    0074  1      !
;    0075  1      ! TABLE OF CONTENTS:
;    0076  1      !
;    0077  1
;    0078  1      FORWARD ROUTINE
;    0079  1                  VRSetQA         : JSB_CLI,
;    0080  1                  VRShowQA        : JSB_CLI;
```

```
;    0082  1      xSBTTL  'Psect Definitions'
;    0083  1      !
;    0084  1      ! PSECT DEFINITIONS:
;    0085  1      !
;    0086  1
;    0087  1      PSECT
;    0088  1              Plit    = Data (NoExecute, Share, NoWrite,
;    0089  1                               Addressing_Mode (Long_Relative));
;    0090  1
;    0091  1      PSECT
;    0092  1              Code    = Code (Execute, Share, NoWrite);
;    0093  1
;    0094  1      PSECT
;    0095  1              Own     = Work (NoExecute, NoShare, Write,
;    0096  1                               Addressing_Mode (Long_Relative));
;    0097  1
;    0098  1      PSECT
;    0099  1              Global  = Work (NoExecute, NoShare, Write,
;    0100  1                               Addressing_Mode (Long_Relative));
```

```
;    0102  1      %SBTTL  'Literals'
;    0103  1      !
;    0104  1      ! MODULE-GLOBAL LITERALS:
;    0105  1      !
;    0106  1      LITERAL
;    0107  1              QA$K_TestLoops_Def      = 100,          ! Defaults for the 5 QA flags
;    0108  1              QA$K_SubtestLoops_Def   = 100,
;    0109  1              QA$K_MultiplePass_Def   = 10,
;    0110  1
;    0111  1              QA$K_Success            = 1,            ! Routine Success or
;    0112  1              QA$K_Failure            = 0;            !   Failure
;    0113  1
;    0114  1      LITERAL
;    0115  1              !+
;    0116  1              ! Shorthand forms for the QA flags
;    0117  1              !
;    0118  1              !       TL => TestLoops
;    0119  1              !       SL => SubtestLoops
;    0120  1              !       MP => MultiplePass
;    0121  1              !-
;    0122  1
;    0123  1              QATL_Low        = 1,    ! Lower limit of QATESTLOOPS
;    0124  1              QATL_High       = 32767,! Upper limit of QATESTLOOPS
;    0125  1
;    0126  1              QASL_Low        = 2,    ! Lower limit of QASUBTESTLOOPS                 [02]
;    0127  1              QASL_High       = 32767,! Upper limit of QASUBTESTLOOPS
;    0128  1
;    0129  1              QAMP_Low        = 1,    ! Lower limit of QAMULTIPLEPASS
;    0130  1              QAMP_High       = 32767;! Upper limit of QAMULTIPLEPASS
;    0131  1
```

```
;    0133  1      %SBTTL  'Static Storage'
;    0134  1      !
;    0135  1      ! MODULE-GLOBAL DECLARATIONS:
;    0136  1      !
;    0137  1
;    0138  1      GLOBAL
;    0139  1              !+
;    0140  1              ! For storing the current values of the QA flags.
;    0141  1              !-
;    0142  1
;    0143  1              QA$W_TestLoops          : WORD INITIAL (QA$K_TestLoops_Def),
;    0144  1              QA$W_SubtestLoops       : WORD INITIAL (QA$K_SubtestLoops_Def),
;    0145  1              QA$W_MultiplePass       : WORD INITIAL (QA$K_MultiplePass_Def);
```

```
;    0147  1      %SBTTL  'Global Bindings'
;    0148  1      !
;    0149  1      ! Module-Global Bindings
;    0150  1      !
;    0151  1      BIND
;    0152  1              QAMP_Out = $ASCIC ('QAMULTIPLEPASS'),
;    0153  1              QATL_Out = $ASCIC ('QATESTLOOPS'),
;    0154  1              QASL_Out = $ASCIC ('QASUBTESTLOOPS');
```

```
;    0156  1      %SBTTL 'VRSetQA'
;    0157  1
;    0158  1      GLOBAL ROUTINE VRSetQA (Cli_Base) : JSB_CLI =
;    0159  1
;    0160  1      !++
;    0161  1      ! FUNCTIONAL DESCRIPTION:
;    0162  1      !
;    0163  1      !       This routine is called from the CLI module. It sets the specified
;    0164  1      !       QA flag to the requested value (within limits).
;    0165  1      !
;    0166  1      ! CALLER(S):
;    0167  1      !
;    0168  1      !       CLI
;    0169  1      !
;    0170  1      ! FORMAL PARAMETERS:
;    0171  1      !
;    0172  1      !       R2 - Points to the CLI data table base.
;    0173  1      !
;    0174  1      ! IMPLICIT INPUTS:
;    0175  1      !
;    0176  1      !       Cli_Base [CLI$L_FLAGS] - Indicates which flag to change.
;    0177  1      !       Cli_Base [CLI$L_DATA]  - Indicates the new desired value of the flag.
;    0178  1      !
;    0179  1      ! IMPLICIT OUTPUTS:
;    0180  1      !
;    0181  1      !       New value(s) for the QAx variables declared above.
;    0182  1      !
;    0183  1      ! COMPLETION CODES:
;    0184  1      !
;    0185  1      !       0 - Failure, value out of range.
;    0186  1      !       1 - Success.
;    0187  1      !
;    0188  1      ! SIDE EFFECTS:
;    0189  1      !
;    0190  1      !       Changes the value of the specifed flag, iff the value requested is
;    0191  1      !       within range.
;    0192  1      !--
```

```
;    0194  1
;    0195  2     BEGIN
;    0196  2             MAP
;    0197  2                     Cli_Base : REF BLOCK [, BYTE];
;    0198  2
;    0199  2             LOCAL
;    0200  2                     New_Value;
;    0201  2
;    0202  2             BIND
;    0203  2                     Out_Of_Range_Message =
;    0204  2                             $ASCIC ('!AC must be greater than or equal to !UW and less than or equal to !UW.!/');   !
;    0205  2
;    0206  2             MACRO
; M  0207  2                     Check_Case (Short_Name, Data_Name, Cli_Name) =
; M  0208  2                             [.Cli_Base [%NAME ('CLI$V_', Cli_Name)]]:
; M  0209  2                                     IF (
; M  0210  2                                        (.New_Value LSS %NAME (Short_Name, '_Low'))
; M  0211  2                                       OR
; M  0212  2                                        (.New_Value GTR %NAME (Short_Name, '_High'))
; M  0213  2                                        )
; M  0214  2                                     THEN
; M  0215  2                                             BEGIN
; M  0216  2                                             $DS_PRINTF (
; M  0217  2                                                     Out_Of_Range_Message,
; M  0218  2                                                     %NAME (Short_Name, '_Out'),
; M  0219  2                                                     %NAME (Short_name, '_Low'),
; M  0220  2                                                     %NAME (Short_Name, '_High')
; M  0221  2                                                             );
; M  0222  2                                             RETURN (QA$K_Failure)
; M  0223  2                                             END
; M  0224  2                                     ELSE
; M  0225  2                                             Data_Name = .New_Value
;    0226  2                             %;
```

```
;    0228  2
;    0229  2              New_Value = .Cli_Base [CLI$L_DATA];
;    0230  2
;    0231  2              SELECTONE 1 OF
;    0232  2                      SET
;    0233  2                      Check_Case (QATL, QA$W_TestLoops,    QATestLoops);
;    0234  2                      Check_Case (QASL, QA$W_SubtestLoops, QASubtestLoops);
;    0235  2                      Check_Case (QAMP, QA$W_MultiplePass, QAMultiplePass);
;    0236  2                      [.Cli_Base [CLI$V_DEFAULT]]:
;    0237  2                              !
;    0238  2                              ! Set all of them to their defaults:
;    0239  2                              !
;    0240  3                              BEGIN
;    0241  3                              QA$W_TestLoops    = QA$K_TestLoops_Def;
;    0242  3                              QA$W_SubtestLoops = QA$K_SubtestLoops_Def;
;    0243  3                              QA$W_MultiplePass = QA$K_MultiplePass_Def
;    0244  3                              END
;    0245  2                      TES;
;    0246  2
;    0247  3              RETURN (QA$K_Success)
;    0248  1      END;


                                                                .TITLE  QAFLAGS *** Setting and Showing QA Flags
                                                                .IDENT  \6.6-04\

                                                                .PSECT  WORK,NOEXE,2

                                                        0064  00000 QA$W_TESTLOOPS::
                                                                .WORD   100                                     ;
                                                        0064  00002 QA$W_SUBTESTLOOPS::
                                                                .WORD   100                                     ;
                                                        000A  00004 QA$W_MULTIPLEPASS::
                                                                .WORD   10                                      ;

                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

53  53  41  50  45  4C  50  49  54  4C  55  4D  41  51  0E  00000 P.AAA:   .ASCII  <14>\QAMULTIPLEPASS\                      ;
            53  50  4F  4F  4C  54  53  45  54  41  51  0B  0000F P.AAB:   .ASCII  <11>\QATESTLOOPS\                       ;
53  50  4F  4F  4C  54  53  45  54  42  55  53  41  51  0E  0001B P.AAC:   .ASCII  <14>\QASUBTESTLOOPS\                    ;
72  67  20  65  62  20  74  73  75  6D  20  43  41  21  49  0002A P.AAD:   .ASCII  \I!AC must be greater than or equal to !U\ ;
65  20  72  6F  20  6E  61  68  74  20  72  65  74  61  65  00039                                                       ;
                    55  21  20  6F  74  20  6C  61  75  71  00048                                                       ;
6E  61  68  74  20  73  73  65  6C  20  64  6E  61  20  57  00052           .ASCII  \W and less than or equal to !UW.!/\          ;
55  21  20  6F  74  20  6C  61  75  71  65  20  72  6F  20  00061                                                       ;
                                        2F  21  2E  57  00070                                                       ;

                                                                QAMP_OUT=               P.AAA
                                                                QATL_OUT=               P.AAB
                                                                QASL_OUT=               P.AAC
                                                                OUT_OF_RANGE_MESSAGE=
                                                                                        P.AAD
                                                                .EXTRN  DS$PRINTF

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                                        53  DD  00000 VRSETQA::
```

```
                                                                 PUSHL   R3                                              ; 0158
                                53       1C    A2  D0 00002      MOVL    28(CLI_BASE), NEW_VALUE                         ; 0229
                          23    62             1D  E1 00006      BBC     #29, (CLI_BASE), 3$                             ; 0233
                                               09  15 0000A      BLEQ    1$                                              ;
                             00007FFF  8F       53  D1 0000C      CMPL    NEW_VALUE, #32767                               ;
                                               0F  15 00013      BLEQ    2$                                              ;
                                7E     7FFF    8F  3C 00015 1$:  MOVZWL  #32767, -(SP)                                   ;
                                               01  DD 0001A      PUSHL   #1                                              ;
                                   00000000'   EF  9F 0001C      PUSHAB  QATL_OUT                                        ;
                                               51  11 00022      BRB     8$                                              ;
                             00000000'  EF     53  B0 00024 2$:  MOVW    NEW_VALUE, QA$W_TESTLOOPS                       ;
                                               76  11 0002B      BRB     11$                                             ;
                          26    62             1E  E1 0002D 3$:  BBC     #30, (CLI_BASE), 6$                             ; 0234
                                02             53  D1 00031      CMPL    NEW_VALUE, #2                                   ;
                                               09  19 00034      BLSS    4$                                              ;
                             00007FFF  8F       53  D1 00036      CMPL    NEW_VALUE, #32767                               ;
                                               0F  15 0003D      BLEQ    5$                                              ;
                                7E     7FFF    8F  3C 0003F 4$:  MOVZWL  #32767, -(SP)                                   ;
                                               02  DD 00044      PUSHL   #2                                              ;
                                   00000000'   EF  9F 00046      PUSHAB  QASL_OUT                                        ;
                                               27  11 0004C      BRB     8$                                              ;
                             00000000'  EF     53  B0 0004E 5$:  MOVW    NEW_VALUE, QA$W_SUBTESTLOOPS                    ;
                                               4C  11 00055      BRB     11$                                             ;
                                               62  D5 00057 6$:  TSTL    (CLI_BASE)                                      ; 0235
                                               32  18 00059      BGEQ    10$                                             ;
                                               53  D5 0005B      TSTL    NEW_VALUE                                       ;
                                               09  15 0005D      BLEQ    7$                                              ;
                             00007FFF  8F       53  D1 0005F      CMPL    NEW_VALUE, #32767                               ;
                                               1C  15 00066      BLEQ    9$                                              ;
                                7E     7FFF    8F  3C 00068 7$:  MOVZWL  #32767, -(SP)                                   ;
                                               01  DD 0006D      PUSHL   #1                                              ;
                                   00000000'   EF  9F 0006F      PUSHAB  QAMP_OUT                                        ;
                                   00000000'   EF  9F 00075 8$:  PUSHAB  OUT_OF_RANGE_MESSAGE                            ;
                          00000000G  9F             04  FB 0007B      CALLS   #4, a#DS$PRINTF                           ;
                                               24  11 00082      BRB     12$                                             ;
                             00000000'  EF     53  B0 00084 9$:  MOVW    NEW_VALUE, QA$W_MULTIPLEPASS                    ;
                                               16  11 0008B      BRB     11$                                             ;
                          12    62             0C  E1 0008D 10$: BBC     #12, (CLI_BASE), 11$                            ; 0236
                             00000000'  EF 00640064  8F  D0 00091      MOVL    #6553700, QA$W_TESTLOOPS                  ; 0241
                             00000000'  EF     0A  B0 0009C      MOVW    #10, QA$W_MULTIPLEPASS                          ; 0243
                                50             01  D0 000A3 11$: MOVL    #1, R0                                          ; 0247
                                               02  11 000A6      BRB     13$                                             ;
                                               50  D4 000A8 12$: CLRL    R0                                              ; 0248
                                               08  BA 000AA 13$: POPR    #^M<R3>                                         ;
                                                   05 000AC      RSB                                                     ;

; Routine Size:  173 bytes,    Routine Base:  CODE + 0000

;   0249  1
```

```
;    0251  1      %SBTTL 'VRShowQA'
;    0252  1
;    0253  1      GLOBAL ROUTINE VRShowQA (Cli_Base) : JSB_CLI =
;    0254  1
;    0255  1      !++
;    0256  1      ! FUNCTIONAL DESCRIPTION:
;    0257  1      !
;    0258  1      !       This global routine is called from the CLI module. It shows the
;    0259  1      !       current value of the specified QA flag.
;    0260  1      !
;    0261  1      ! CALLER(S):
;    0262  1      !
;    0263  1      !       CLI
;    0264  1      !
;    0265  1      ! FORMAL PARAMETERS:
;    0266  1      !
;    0267  1      !       R2 - Points to the CLI data-table base.
;    0268  1      !
;    0269  1      ! IMPLICIT INPUTS:
;    0270  1      !
;    0271  1      !       Cli_Base [CLI$L_FLAGS] - Indicates which flag to "SHOW".
;    0272  1      !
;    0273  1      ! IMPLICIT OUTPUTS:
;    0274  1      !
;    0275  1      !       NONE
;    0276  1      !
;    0277  1      ! COMPLETION CODES:
;    0278  1      !
;    0279  1      !       1 - Success.
;    0280  1      !
;    0281  1      ! SIDE EFFECTS:
;    0282  1      !
;    0283  1      !       NONE
;    0284  1      !--
```

```
;    0286  1
;    0287  2      BEGIN
;    0288  2              MAP
;    0289  2                      Cli_Base : REF BLOCK [, BYTE];
;    0290  2
;    0291  2              BIND
;    0292  2                      Output_String   = $ASCIC ('!AC is set to !UW.!/'),
;    0293  2                      Default_String  = $ASCIC ('The default for !AC is !UW.!/');
;    0294  2
;    0295  2
;    0296  2              MACRO
; M  0297  2                      Check_Case (Short_Name, Data_Name, Cli_Name) =
; M  0298  2                              [.Cli_Base [%NAME ('CLI$V_', Cli_Name)]]:
; M  0299  2                                      $DS_PRINTF (Output_String,
; M  0300  2                                                  %NAME (Short_Name, '_Out'),
; M  0301  2                                                  .Data_Name)
;    0302  2                      %;
```

```
;    0304  2
;    0305  2              SELECTONE 1 OF
;    0306  2                      SET
;    0307  2                      Check_Case (QATL, QA$W_TestLoops,    QATestLoops);
;    0308  2                      Check_Case (QASL, QA$W_SubtestLoops, QASubtestLoops);
;    0309  2                      Check_Case (QAMP, QA$W_MultiplePass, QAMultiplePass);
;    0310  2                      [.Cli_Base [CLI$V_DEFAULT]]:
;    0311  3                              BEGIN
; P  0312  3                              $DS_PRINTF (Default_String, QAMP_Out,
;    0313  3                                      QA$K_MultiplePass_Def);
;    0314  3
; P  0315  3                              $DS_PRINTF (Default_String, QATL_Out,
;    0316  3                                      QA$K_TestLoops_Def);
;    0317  3
; P  0318  3                              $DS_PRINTF (Default_String, QASL_Out,
;    0319  4                                      QA$K_SubtestLoops_Def)
;    0320  2                              END;
;    0321  2                      TES;
;    0322  2
;    0323  3              RETURN (QA$K_Success)
;    0324  1      END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20  6F  74  20  74  65  73  20  73  69  20  43  41  21  14  00074 P.AAE:  .ASCII  <20>\!AC is set to !UW.!/\                    ;
                                    2F  21  2E  57  55  21  00083                                                            ;
6F  66  20  74  6C  75  61  66  65  64  20  65  68  54  1D  00089 P.AAF:  .ASCII  <29>\The default for !AC is !UW.!/\           ;
2F  21  2E  57  55  21  20  73  69  20  43  41  21  20  72  00098                                                            ;

                                                                OUTPUT_STRING=      P.AAE
                                                                DEFAULT_STRING=     P.AAF


                                                                .PSECT  CODE,NOWRT,  SHR,2

                        0F              62              1D  E1 00000 VRSHOWQA::
                                                                      BBC     #29, (CLI_BASE), 1$                               ; 0307
                                        7E 00000000'    EF  3C 00004          MOVZWL  QA$W_TESTLOOPS, -(SP)                     ;
                                           00000000'    EF  9F 0000B          PUSHAB  QATL_OUT                                  ;
                                                        24  11 00011          BRB     3$                                        ;
                        0F              62              1E  E1 00013 1$:      BBC     #30, (CLI_BASE), 2$                       ; 0308
                                        7E 00000000'    EF  3C 00017          MOVZWL  QA$W_SUBTESTLOOPS, -(SP)                  ;
                                           00000000'    EF  9F 0001E          PUSHAB  QASL_OUT                                  ;
                                                        11  11 00024          BRB     3$                                        ;
                                                        62  D5 00026 2$:      TSTL    (CLI_BASE)                                ; 0309
                                                        15  18 00028          BGEQ    4$                                        ;
                                        7E 00000000'    EF  3C 0002A          MOVZWL  QA$W_MULTIPLEPASS, -(SP)                  ;
                                           00000000'    EF  9F 00031          PUSHAB  QAMP_OUT                                  ;
                                           00000000'    EF  9F 00037 3$:      PUSHAB  OUTPUT_STRING                             ;
                                                        40  11 0003D          BRB     5$                                        ;
                        43              62              0C  E1 0003F 4$:      BBC     #12, (CLI_BASE), 6$                       ; 0310
                                                        0A  DD 00043          PUSHL   #10                                       ; 0313
                                           00000000'    EF  9F 00045          PUSHAB  QAMP_OUT                                  ;
                                           00000000'    EF  9F 0004B          PUSHAB  DEFAULT_STRING                            ;
                            00000000G   9F              03  FB 00051          CALLS   #3, @#DS$PRINTF                           ;
```

```
                        7E        64   8F  9A 00058        MOVZBL  #100, -(SP)                          ; 0316
                           00000000'   EF  9F 0005C        PUSHAB  QATL_OUT                             ;
                           00000000'   EF  9F 00062        PUSHAB  DEFAULT_STRING                       ;
                00000000G  9F          03  FB 00068        CALLS   #3, a#DS$PRINTF                      ;
                        7E        64   8F  9A 0006F        MOVZBL  #100, -(SP)                          ; 0319
                           00000000'   EF  9F 00073        PUSHAB  QASL_OUT                             ;
                           00000000'   EF  9F 00079        PUSHAB  DEFAULT_STRING                       ;
                00000000G  9F          03  FB 0007F 5$:    CALLS   #3, a#DS$PRINTF                      ;
                        50             01  D0 00086 6$:    MOVL    #1, R0                               ; 0323
                                           05 00089        RSB                                          ; 0324

; Routine Size:  138 bytes,    Routine Base:  CODE + 00AD
```

```
;   0326  1     %SBTTL 'End of Module QAFLAGS'
;   0327  1
;   0328  0     END ELUDOM
```

```
;                                  PSECT SUMMARY
;
;       Name                    Bytes                        Attributes
;
;  WORK                             6  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  DATA                           167  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                           311  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...											
$ASCIC	Macro	Lib01	152	153	154	204	292	293						
$DS_PRINTF	Unbound		216	299										
	Macro	Lib01	233	234	235	307	308	309	312	315	318			
CHECK_CASE	Macro	207	233	234	235									
	Macro	297	307	308	309									
CLI$L_DATA	Macro	Lib02	229											
CLI$V_DEFAULT	Macro	Lib02	236	310										
CLI$V_QAMULTIPLEPASS	Macro	Lib02	235	309										
CLI$V_QASUBTESTLOOPS	Macro	Lib02	234	308										
CLI$V_QATESTLOOPS	Macro	Lib02	233	307										
CLI_BASE	Unbound		208	298										
	RoutForm	158	197m	229a.	233a.	234a.	235a.	236a.						
	RoutForm	253	289m	307a.	308a.	309a.	310a.							
CLI_NAME	MacrForm	207	208											
	MacrForm	297	298											
DATA_NAME	MacrForm	207	225											
	MacrForm	297	301											
DEFAULT_STRING	Bind	293	313a	316a	319a									
DS$PRINTF	ExtRout	233	233c	234e	234c	235e	235c	307e	307c	308e	308c	309e	309c	313e
			313c	316e	316c	319e	319c							
JSB_CLI	Linkage	Lib02	79	80	158	253								
NEW_VALUE	Unbound		210	212	225									
	Local	200	229=	233.	234.	235.								
OUTPUT_STRING	Unbound		299											
	Bind	292	307a	308a	309a									
OUT_OF_RANGE_MESSAGE	Unbound		217											
	Bind	203	233a	234a	235a									
QA$K_FAILURE	Unbound		222											
	Literal	112	233	234	235									
QA$K_MULTIPLEPASS_DEF	Literal	109	145	243	313									
QA$K_SUBTESTLOOPS_DEF	Literal	108	144	242	319									
QA$K_SUCCESS	Literal	111	247	323										
QA$K_TESTLOOPS_DEF	Literal	107	143	241	316									
QA$W_MULTIPLEPASS	Global	145	145=	235=	243=	309.								

Symbol	Type	Defined	Referenced ...				
QA$W_SUBTESTLOOPS	Global	144	144=	234=	242=	308.	
QA$W_TESTLOOPS	Global	143	143=	233=	241=	307.	
QAMP_HIGH	Literal	130	235				
QAMP_LOW	Literal	129	235				
QAMP_OUT	Bind	152	235a	309a	313a		
QASL_HIGH	Literal	127	234				
QASL_LOW	Literal	126	234				
QASL_OUT	Bind	154	234a	308a	319a		
QATL_HIGH	Literal	124	233				
QATL_LOW	Literal	123	233				
QATL_OUT	Bind	153	233a	307a	316a		
SHORT_NAME	MacrForm	207	210	212	218	219	220
	MacrForm	297	300				
VRSETQA	GlobRout	158	79f				
VRSHOWQA	GlobRout	253	80f				

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]QAFLAGS.B32;1
5	Module QAFLAGS	
67	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
69	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
71	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
328	Eludom QAFLAGS	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
a	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

; Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	2	0	96	00:00.0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	6	0	47	00:00.0
SYS$COMMON:[SYSLIB]LIB.L32;2	20450	0	0	1101	00:00.8

```
;                                   COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:QAFLAGS.B32

; Size:         311 code + 173 data bytes
; Run Time:         00:02.2
; Elapsed Time:     00:04.2
; Lines/CPU Min:     8785
; Lexemes/CPU-Min: 50705
; Memory Used:  96 pages
; Compilation Complete
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element QAMAIN.B32
;    0002  0      ! *1     6-FEB-1986 15:22:02 DS ""
;    0003  0      ! DEC/CMS REPLACEMENT HISTORY, Element QAMAIN.B32
;    0004  0      %TITLE '*** QA Routines'
;    0005  0      MODULE QAMAIN (
;    0006  0                ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE),
;    0007  0                IDENT = '1-08'
;    0008  0                ) =
;    0009  0      !
;    0010  0      ! COPYRIGHT (c) 1979, 1981, 1983
;    0011  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0012  0      !
;    0013  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0014  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0015  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0016  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0017  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0018  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0019  0      ! REMAIN IN DEC.
;    0020  0      !
;    0021  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0022  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0023  0      ! CORPORATION.
;    0024  0      !
;    0025  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0026  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0027  0
;    0028  0      !++
;    0029  0      ! FACILITY:
;    0030  0      !
;    0031  0      !       DIAGNOSTIC SUPERVISOR
;    0032  0      !
;    0033  0      ! ABSTRACT:
;    0034  0      !
;    0035  0      !       This module contains the support routines required to perform QA on diagnostic programs.
;    0036  0      !
;    0037  0      ! AUTHOR:
;    0038  0      !
;    0039  0      !       Jack Stansbury
;    0040  0      !
;    0041  0      ! CREATION DATE:
;    0042  0      !
;    0043  0      !               15-October-1981, Version 6.5-00
;    0044  0      !       00      Put in code for all the support routines plus the following check routines: QA$Normal_Start,
;    0045  0      !               QA$Multiple_Pass, QA$Loop_On_Test, QA$Run_Backwards.
```

```
;    0047  0      ! MODIFIED:
;    0048  0      !
;    0049  0      !       01      Jack Stansbury, 14-December-1981, Version 6.6
;    0050  0      !               Added remaining QA check routines. Split QA.B32 into two different modules: QAMAIN.B32
;    0051  0      !               (this one) and QACHECKS.B32. While this one contains the support routines, the latter contains
;    0052  0      !               the actual check routines.
;    0053  0      !
;    0054  0      !       02      Jack Stansbury, 5-Jan-1982, Version 6.6
;    0055  0      !               Changed the code in QA$Dump to not print R0 and R1 in the register dump. There is no need to
;    0056  0      !               print these since they are not saved by the CALLx instructions.
;    0057  0      !
;    0058  0      !       03      Jack Stansbury, 1-Feb-1982, Version 6.6
;    0059  0      !               Released the three Supervisors to BSDE to find bugs. This includes all the check routine up to
;    0060  0      !               and including the Error Phase One check.
;    0061  0      !
;    0062  0      !       04      Jack Stansbury, 1-Feb-1982, Version 6.6
;    0063  0      !               Changed the headings on the two table printouts to make the table name more distinctive.
;    0064  0      !
;    0065  0      !       05      Jack Stansbury, 11-Feb-1982, Version 6.6X
;    0066  0      !               Changed the DSX$Branch routine so as to make it compatible with Sam Duncan! He wanted R0 saved
;    0067  0      !               across the branch macro code expansion, so I made significant changes to both the branch macros
;    0068  0      !               (in DIAG.*) and to this branch routine. All of that just for Sam!
;    0069  0      !
;    0070  0      !       06      Jack Stansbury, 27-July-1982, Version 6.9
;    0071  0      !               Changed the spelling of the VDS ^C handler variable.
;    0072  0      !
;    0073  0      !       07      Jack Stansbury, 19-Jan-1983, Version 6.10 (?)
;    0074  0      !               Took out all the references to any of the checks above the Run Backwards check. This
;    0075  0      !               includes taking out the previously entered Error Phase One check. It doesn't work
;    0076  0      !               properly with subroutines in diagnostic programs.
;    0077  0      !
;    0078  0      !       08      Bob Bergazzi    May 17, 1983    Version 6.11
;    0079  0      !               Changed the order of searching libraries.
;    0080  0      !--
```

```
;    0082  0     %SBTTL  'Linkage Definitions'
;    0083  1     BEGIN
;    0084  1     !
;    0085  1     ! LINKAGE DEFINITIONS
;    0086  1     !
;    0087  1
;    0088  1     LINKAGE
;    0089  1             !+
;    0090  1             ! For the DSX$Branch routine, the last three parameters are passed in the STANDARD manner. That is      [05]
;    0091  1             ! by an offset to the AP on the stack. The first parameter passed is general register R0. Also,         [05]
;    0092  1             ! register R1 should be preserved in the routine. DSX$Branch is defined to be a VALUE routine also, so  [05]
;    0093  1             ! that values can be returned in R0. This is not compatible with the VAX calling standard however.     [05]
;    0094  1             !-
;    0095  1             Call_DSX$Branch = CALL (REGISTER=0, STANDARD, STANDARD, STANDARD) : PRESERVE (1);                !       [05]
```

```
;    0097  1     %SBTTL  'Table of Contents'
;    0098  1     !
;    0099  1     ! TABLE OF CONTENTS:
;    0100  1     !
;    0101  1
;    0102  1     FORWARD ROUTINE
;    0103  1             DSX$Branch                  :             ADDRESSING_MODE (LONG_RELATIVE) Call_DSX$Branch,        !       [05]
;    0104  1             QA$Abort_QA                 :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE),        !                     [02]
;    0105  1             QA$Add_Forced_Error         :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE),        !                     [02]
;    0106  1             QA$Add_QA_Error             :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE),
;    0107  1             QA$Control_C_Handler        :             ADDRESSING_MODE (LONG_RELATIVE),        !                     [06]
;    0108  1             QA$Dump                     :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE),
;    0109  1             QA$Find_User_Call_Frame     :             ADDRESSING_MODE (LONG_RELATIVE),
;    0110  1             QA$Init                     :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE),
;    0111  1             QA$Main                     :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE),
;    0112  1             QA$Print_Forced_Errors      :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE),        !                     [02]
;    0113  1             QA$Print_Overall_Errors     :   NOVALUE   ADDRESSING_MODE (LONG_RELATIVE);        !                     [02]
```

```
;    0115  1     %SBTTL  'Libraries'
;    0116  1     !
;    0117  1     ! LIBRARIES:
;    0118  1     !
;    0119  1
;    0120  1     LIBRARY
;    0121  1             '$DIAG';                    !                                   [08]
;    0122  1
;    0123  1     LIBRARY
;    0124  1             '$DS';                      !                                   [08]
;    0125  1
;    0126  1     LIBRARY
;    0127  1             'SYS$LIBRARY:LIB';
```

```
;    0129  1      %SBTTL  'Included Macros and Literals'
;    0130  1      !
;    0131  1      ! INCLUDED MACROS AND LITERALS
;    0132  1      !
;    0133  1
;    0134  1              $DS_QADEF;                      ! Define the branch-codes.
;    0135  1              $DS_DSADEF;                     ! Define DSA flags
;    0136  1              DSQA;                           ! The DSQA$K_ constants for the routine/label names.
;    0137  1              DS_QADEFS;                      ! Common QA definitions like QA$K_.
```

```
;    0139  1      %SBTTL  'External Routines'
;    0140  1      !
;    0141  1      ! EXTERNAL ROUTINE REFERENCES:
;    0142  1      !
;    0143  1
;    0144  1      EXTERNAL ROUTINE
;    0145  1
;    0146  1              QA$Normal_Start         :             ADDRESSING_MODE (LONG_RELATIVE),
;    0147  1              QA$Multiple_Pass        :             ADDRESSING_MODE (LONG_RELATIVE),
;    0148  1              QA$Loop_On_Test         :             ADDRESSING_MODE (LONG_RELATIVE),
;    0149  1              QA$Loop_On_Subtest      :             ADDRESSING_MODE (LONG_RELATIVE),      !                    [01]
;    0150  1              QA$Run_Backwards        :             ADDRESSING_MODE (LONG_RELATIVE),
;    0151  1
;    0152  1              EXE$AloNonPaged         : JSB_ALLOC   ADDRESSING_MODE (LONG_RELATIVE),      !                    [01]
;    0153  1              EXE$DeaNonPaged         : JSB_DEALL   ADDRESSING_MODE (LONG_RELATIVE),      !                    [01]
;    0154  1
;    0155  1              DSV$ShowDevice          : JSB_CLI,            ! Pass parameter in R2
;    0156  1              DSV$ShowSelect          : JSB_CLI,            ! Pass parameter in R2
;    0157  1              DSX$Abort,                                    ! Pass no parameters,
;    0158  1                                                            ! (take no prisoners).
;    0159  1              VRShowFlg               : JSB_NONE,           ! No parameters required
;    0160  1              VRShowEf                : JSB_NONE;           ! No parameters required
```

```
;    0162  1      %SBTTL  'External Own Storage'
;    0163  1      !
;    0164  1      ! EXTERNAL OWN STORAGE
;    0165  1      !
;    0166  1
;    0167  1      EXTERNAL
;    0168  1              DS$GL_FSTTEST,                          ! First test to execute
;    0169  1              DS$GL_LSTTEST,                          ! Last test to execute
;    0170  1
;    0171  1              DS$GA_DS_Ctrl_C_First,                  ! DS address of 1st ^C routine handler          [06]
;    0172  1
;    0173  1              DS$GL_CLIBASE,                          ! Base of CLI data table
;    0174  1
;    0175  1              DS$GB_WIDTH,                            ! Width of the terminal                         [02]
;    0176  1
;    0177  1              !+
;    0178  1              ! These are defined in the QAFLAGS module. They contain
;    0179  1              ! the current settings of the QA flags.
;    0180  1              !-
;    0181  1
;    0182  1              QA$W_TestLoops    : WORD,
;    0183  1              QA$W_SubtestLoops : WORD,
;    0184  1              QA$W_MultiplePass : WORD,
;    0185  1
;    0186  1              QA$T_Header_1,                          ! The three lines printed at the top            [01]
;    0187  1              QA$T_Header_2,                          ! of a page.                                    [01]
;    0188  1              QA$T_Header_3,                          !                                               [01]
;    0189  1
;    0190  1              QA$T_Operator_Request_Message;          ! The QA error message for requesting operator input  [02]
;    0191  1                                                     ! with no default value given. It is declared in the  [02]
;    0192  1                                                     ! QACHECKS module because all the error messages are  [02]
;    0193  1                                                     ! declared there!                                     [02]
```

```
;    0195  1      %SBTTL  'Psect Definitions'
;    0196  1      !
;    0197  1      ! PSECT DEFINITIONS:
;    0198  1      !
;    0199  1
;    0200  1      PSECT
;    0201  1              Plit   = Data (NoExecute, Share, NoWrite,
;    0202  1                             Addressing_Mode (Long_Relative));
;    0203  1
;    0204  1      PSECT
;    0205  1              Code   = Code (Execute, Share, NoWrite,
;    0206  1                             Addressing_Mode (Long_Relative));
;    0207  1
;    0208  1      PSECT
;    0209  1              Own    = Work (NoExecute, NoShare, Write,
;    0210  1                             Addressing_Mode (Long_Relative));
;    0211  1
;    0212  1      PSECT
;    0213  1              Global = Work (NoExecute, NoShare, Write,
;    0214  1                             Addressing_Mode (Long_Relative));
```

```
;    0216  1      %SBTTL  'Local Macro Definitions'
;    0217  1      !
;    0218  1      ! LOCAL MACRO DEFINITIONS:
;    0219  1      !
;    0220  1
;    0221  1      MACRO
;    0222  1              !+
;    0223  1              ! Usage:
;    0224  1              !       ASSERT ((.A GTR .B), 'A <= B');
;    0225  1              !
;    0226  1              ! If the specified condition is not true, a call to the Logic_Error macro is made with the specified
;    0227  1              ! message string. ASSERT is turned on whenever the compilation is made with the /VARIANT:n qualifier,
;    0228  1              ! where n <> 0.
;    0229  1              !-
;    0230  1
; M  0231  1              ASSERT (Condition, Assert_String) =
; M  0232  1                      %IF %VARIANT
; M  0233  1                      %THEN
; M  0234  1                              (
; M  0235  1                              IF (NOT (Condition)) THEN
; M  0236  1                                      Logic_Error (Assert_String)
; M  0237  1                              )
; M  0238  1                      %ELSE
; M  0239  1                              ! No code put in.
; M  0240  1                      %FI
;    0241  1              %;
;    0242  1
;    0243  1      MACRO
;    0244  1              !+
;    0245  1              ! Usage:
;    0246  1              !       Logic_Error ('String to be printed out');
;    0247  1              !
;    0248  1              ! This macro is used whenever a logic error is detected. It is called from ASSERT whenever the
;    0249  1              ! assertion is false. It can also be called whenever an expression need not be evaluated before
;    0250  1              ! determining that something is wrong.
;    0251  1              !-
;    0252  1
; M  0253  1              Logic_Error (Error_String) =
; M  0254  1                      (
; M  0255  1                      BIND
; M  0256  1                              ES = $ASCIC ('!/**QA Internal Error: ', Error_String, '!/');    !                    [01]
; M  0257  1
; M  0258  1                      $DS_PRINTF (ES);
; M  0259  1                      )
;    0260  1              %;
;    0261  1
;    0262  1      MACRO                                                           !                    [02]
;    0263  1              !+                                                                           [02]
;    0264  1              ! Define a macro that will expand into the declaration for the Forced-Error records.  [02]
;    0265  1              !-                                                                           [02]
;    0266  1
; M  0267  1              Error_Record =                                          !                    [02]
; M  0268  1                      REF BLOCK [, LONG] FIELD (Forced_Error_Fields)  !                    [02]
;    0269  1              %;                                                      !                    [02]
```

```
;   0271  1     MACRO
;   0272  1             !+
;   0273  1             ! Usage:
;   0274  1             !
;   0275  1             ! This macro is used only in the debugging stages. It's only parameter is a string giving, e.g.,
;   0276  1             ! the name of a routine. DEBUG is turned on whenever the compilation is made with the
;   0277  1             ! /VARIANT:n qualifier, where n <> 0.
;   0278  1             !-
;   0279  1
; M 0280  1             DEBUG (Debug_String) =
; M 0281  1                     %IF %VARIANT
; M 0282  1                     %THEN
; M 0283  1                             (
; M 0284  1                             IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN
; M 0285  1                                     $DS_PRINTF ($ASCIC ('!/', Debug_String, '!/'))
; M 0286  1                             )
; M 0287  1                     %ELSE
; M 0288  1                             ! No code put in.
; M 0289  1                     %FI
;   0290  1             %;
;   0291  1
;   0292  1     MACRO                                                   !                                                   [02]
;   0293  1             !+                                                                                                  [02]
;   0294  1             ! Define a macro that takes an expression as an argument. The value of this macro is either one or [02]
;   0295  1             ! zero. All bits but the least significant one are cleared.                                         [02]
;   0296  1             !-                                                                                                  [02]
;   0297  1
; M 0298  1             Boolean (Expression) =                          !                                                   [02]
; M 0299  1                     ((Expression) AND 1)                    !                                                   [02]
;   0300  1             %;                                              !                                                   [02]
```

```
;    0302  1      %SBTTL  'Module-Global Literals'
;    0303  1
;    0304  1      !+                                                                                  [02]
;    0305  1      ! MODULE-GLOBAL LITERALS                                                            [02]
;    0306  1      !-                                                                                  [02]
;    0307  1
;    0308  1      LITERAL                                        !                                    [02]
;    0309  1              QA$K_Nil = 0,                          ! Pascal NIL. Use address zero, since it is inacessible.[02]
;    0310  1
;    0311  1              QA$K_Forced_Error_Record_Size = 16,    ! The size of the record for the forced-error records.  [02]
;    0312  1
;    0313  1              QA$K_Last_Check = QA$K_Run_Backwards;  ! The last check currently implemented. This must       [07]
;    0314  1                                                     ! change as more check routines are added.              [07]
```

```
;    0316  1     %SBTTL  'Field Definitions'
;    0317  1
;    0318  1     !                                                                                   [02]
;    0319  1     ! MODULE-GLOBAL FIELD DEFINITIONS                                                   [02]
;    0320  1     !                                                                                   [02]
;    0321  1
;    0322  1     FIELD                                                   !                           [02]
;    0323  1             Forced_Error_Fields =                           !                           [02]
;    0324  1                     SET                                     !                           [02]
;    0325  1                     QA$VW_Error_Number    = [0, 0,  16, 0],  !                           [02]
;    0326  1                     QA$VW_Test_Number     = [0, 16, 16, 0],  !                           [02]
;    0327  1                     QA$VL_Error_Count     = [1, 0,  32, 0],  !                           [02]
;    0328  1                     QA$VL_Subtest_Number  = [2, 0,  32, 0],  !                           [02]
;    0329  1                     Qa$VA_Next_Error      = [3, 0,  32, 0]   !                           [02]
;    0330  1                     TES;                                    !                           [02]
```

```
;    0332  1      %SBTTL  'Module-Global Static Storage'
;    0333  1
;    0334  1      !
;    0335  1      ! MODULE-GLOBAL DECLARATIONS:
;    0336  1      !
;    0337  1
;    0338  1              MODNAM ('QAMAIN');                                  ! Define module name.
;    0339  1      OWN
;    0340  1              QA$AW_Overall_Error_Table : VECTOR [QA$K_Number_Of_Checks, WORD];
;    0341  1                                                                  ! A vector containing the number of
;    0342  1                                                                  ! errors encountered in each of
;    0343  1                                                                  ! the checks.
;    0344  1
;    0345  1      !       QA$A_Error_List_Head,                               ! Pointer to head of a linked list of Forced-Error      [07]
;    0346  1      !                                                           ! records.                                              [07]
;    0347  1      !
;    0348  1      !       QA$A_Prev_Test_Tail;                                ! Pointer to the previous test's last record in the     [07]
;    0349  1      !                                                           ! Forced-Error linked list.                             [07]
```

```
%SBTTL  'Supervisor-Global Static Storage'

!+
! SUPERVISOR-GLOBAL DECLARATIONS:
!-

GLOBAL
        QA$W_Branch       : WORD,                          ! Storage for passing the branch-code                [01]
                                                           ! from DSX$BRANCH to QA$MAIN.                        [01]

        QA$W_Error_Number : WORD,                          ! Storage for passing the error number               [01]
                                                           ! from DSX$BRANCH to QA$MAIN.                        [01]

        QA$W_Save_Test    : SIGNED WORD,                   ! Save the x in /TEST:x:y from the command line.     [02]

        QA$W_Save_Last    : SIGNED WORD,                   ! Save the y in /TEST:x:y from the command line.     [02]

        QA$L_Saved_DSA_Flags,                              ! Save the DSA flags before starting QA              [01]
                                                           ! and restore and end of QA execution.               [01]

        QA$AOB_Routine_State : BITVECTOR [QA$K_Numb_Routine_States],    !                                   [02]
                                                           ! The Routine State vector. This vector keeps track  [02]
                                                           ! of the current state that a check routine is in.   [02]

        QA$AOB_Check_State : BITVECTOR [QA$K_Number_Of_Checks],
                                                           ! The check state bitvector. It contains one and only
                                                           ! one bit = 1 implying that the check bound to that bit
                                                           ! position is currently being performed.

        QA$AOB_Flags       : BITVECTOR [QA$K_Number_QA_Flags];
                                                           ! QA$AOB_Flags [QA$K_NoDef] =
                                                           !   1 => No default was given for ASK...
                                                           !   0 => A default was given for ASK...
                                                           !
                                                           ! QA$AOB_Flags [QA$K_QA_Debug] =
                                                           !   1 => Debugging is turned on
                                                           !   0 => No debugging messages
                                                           !
                                                           ! QA$AOB_Flags [QA$K_QA_Done] =
                                                           !   1 => QA is finished
                                                           !   0 => QA is not finished
                                                           !
                                                           ! QA$AOB_Flags [QA$K_Init_Context_Done] =
                                                           !   1 => Init_Context was done
                                                           !   0 => Init_Context has not been done
                                                           !
                                                           ! QA$AOB_Flags [QA$K_Loop_Test_Done] =
                                                           !   1 => Finished looping on the current test
                                                           !   0 => Loop at least once more on the current test
```

```
;    0401  1      %SBTTL 'DSX$BRANCH'
;    0402  1
;    0403  1      GLOBAL ROUTINE DSX$Branch (Register_0, Branch_Code, Error_Number, Branch_Value) : Call_DSX$Branch =
;    0404  1
;    0405  1      !++
;    0406  1      ! FUNCTIONAL DESCRIPTION:
;    0407  1      !
;    0408  1      !       This global routine is called by the $DS_Branch macros. It decides if the branch should be taken.
;    0409  1      !       If the QA flag is set, it will call QA$Main, which calls the appropriate check routine. NOTE:
;    0410  1      !       This routine preserves R1 and attempts to preserve R0 when not forcing an error. At most, the low
;    0411  1      !       bit of R0 will be changed.
;    0412  1      !
;    0413  1      ! FORMAL PARAMETERS:
;    0414  1      !
;    0415  1      !       Register_0    : The general register R0.
;    0416  1      !
;    0417  1      !       Branch_Code   : A literal defining what type of branch is being executed by the diagnostic.
;    0418  1      !
;    0419  1      !       Error_Number  : A diagnostic error number.
;    0420  1      !
;    0421  1      !       Branch_Value  : The value that will determine whether or not the branch should actually be taken.
;    0422  1      !
;    0423  1      ! CALLER(S):
;    0424  1      !
;    0425  1      !       The diagnostic program.
;    0426  1      !
;    0427  1      ! IMPLICIT INPUTS:
;    0428  1      !
;    0429  1      !       NONE
;    0430  1      !
;    0431  1      ! IMPLICIT OUTPUTS:
;    0432  1      !
;    0433  1      !       NONE
;    0434  1      !
;    0435  1      ! COMPLETION CODES:
;    0436  1      !
;    0437  1      !       See below.                                                                              [05]
;    0438  1      !
;    0439  1      ! SIDE EFFECTS:
;    0440  1      !
;    0441  1      !       If running QA and this branch should be reversed, the low bit of R0 will be cleared. If not running QA, [05]
;    0442  1      !       the low bit of R0 will be set or cleared as mandated by the type of branch and by the Branch_Value.     [05]
;    0443  1      !       Under all other conditions, R0 will be left as is. Note: Register_0, the first parameter, is actually   [05]
;    0444  1      !       general register R0.
;    0445  1      !
;    0446  1      !--
```

```
;    0448  2      BEGIN
;    0449  2              LOCAL
;    0450  2                      Save_Register_0,                                  !                                          [05]
;    0451  2                      Branch : BYTE;                                    !                                          [05]
;    0452  2
;    0453  2              Save_Register_0 = .Register_0;                            !                                          [05]
;    0454  2
;    0455  2              CASE .Branch_Code
;    0456  2                  FROM QA$K_First_Branch_Code TO QA$K_Last_Branch_Code OF
;    0457  2                      SET
;    0458  2                      !+
;    0459  2                      ! These are the BLISS specific branch codes.
;    0460  2                      !-
;    0461  2                      [DS$K_BLSS_B] :
;    0462  2                              Branch = (.Branch_Value LSS 0);
;    0463  2
;    0464  2                      [DS$K_BLSSU_B] :
;    0465  2                              Branch = (.Branch_Value LSSU 0);
;    0466  2
;    0467  2                      [DS$K_BLEQ_B] :
;    0468  2                              Branch = (.Branch_Value LEQ 0);
;    0469  2
;    0470  2                      [DS$K_BLEQU_B] :
;    0471  2                              Branch = (.Branch_Value LEQU 0);
;    0472  2
;    0473  2                      [DS$K_BEQL_B] :
;    0474  2                              Branch = (.Branch_Value EQL 0);
;    0475  2
;    0476  2                      [DS$K_BEQLU_B] :
;    0477  2                              Branch = (.Branch_Value EQLU 0);
;    0478  2
;    0479  2                      [DS$K_BGEQ_B] :
;    0480  2                              Branch = (.Branch_Value GEQ 0);
;    0481  2
;    0482  2                      [DS$K_BGEQU_B] :
;    0483  2                              Branch = (.Branch_Value GEQU 0);
;    0484  2
;    0485  2                      [DS$K_BGTR_B] :
;    0486  2                              Branch = (.Branch_Value GTR 0);
;    0487  2
;    0488  2                      [DS$K_BGTRU_B] :
;    0489  2                              Branch = (.Branch_Value GTRU 0);
;    0490  2
;    0491  2                      [DS$K_BNEQ_B] :
;    0492  2                              Branch = (.Branch_Value NEQ 0);
;    0493  2
;    0494  2                      [DS$K_BNEQU_B] :
;    0495  2                              Branch = (.Branch_Value NEQU 0);
```

```
;    0497  2                    !+
;    0498  2                    ! For these MACRO specific branch codes, the Branch_Value is the PSL at the time
;    0499  2                    ! of the call. "To branch or not to branch" is based on this PSL.
;    0500  2                    !-
;    0501  2
;    0502  2                    [DS$K_BLSS_M] :
;    0503  3                            BEGIN
;    0504  3                            MAP
;    0505  3                                    Branch_Value : BLOCK [, BYTE];
;    0506  3
;    0507  3                            Branch = .Branch_Value [PSL$V_N]
;    0508  2                            END;
;    0509  2
;    0510  2                    [DS$K_BLSSU_M] :
;    0511  3                            BEGIN
;    0512  3                            MAP
;    0513  3                                    Branch_Value : BLOCK [, BYTE];
;    0514  3
;    0515  3                            Branch = .Branch_Value [PSL$V_C]
;    0516  2                            END;
;    0517  2
;    0518  2                    [DS$K_BLEQ_M] :
;    0519  3                            BEGIN
;    0520  3                            MAP
;    0521  3                                    Branch_Value : BLOCK [, BYTE];
;    0522  3
;    0523  3                            Branch = .Branch_Value [PSL$V_N] OR
;    0524  3                                     .Branch_Value [PSL$V_Z]
;    0525  2                            END;
;    0526  2
;    0527  2                    [DS$K_BLEQU_M] :
;    0528  3                            BEGIN
;    0529  3                            MAP
;    0530  3                                    Branch_Value : BLOCK [, BYTE];
;    0531  3
;    0532  3                            Branch = .Branch_Value [PSL$V_C] OR
;    0533  3                                     .Branch_Value [PSL$V_Z]
;    0534  2                            END;
;    0535  2
;    0536  2                    [DS$K_BEQL_M] :
;    0537  3                            BEGIN
;    0538  3                            MAP
;    0539  3                                    Branch_Value : BLOCK [, BYTE];
;    0540  3
;    0541  3                            Branch = .Branch_Value [PSL$V_Z]
;    0542  2                            END;
```

```
;    0544  2                    [DS$K_BEQLU_M] :
;    0545  3                            BEGIN
;    0546  3                            MAP
;    0547  3                                    Branch_Value : BLOCK [, BYTE];
;    0548  3
;    0549  3                            Branch = .Branch_Value [PSL$V_Z]
;    0550  2                            END;
;    0551  2
;    0552  2                    [DS$K_BGEQ_M] :
;    0553  3                            BEGIN
;    0554  3                            MAP
;    0555  3                                    Branch_Value : BLOCK [, BYTE];
;    0556  3
;    0557  3                            Branch = NOT .Branch_Value [PSL$V_N]
;    0558  2                            END;
;    0559  2
;    0560  2                    [DS$K_BGEQU_M] :
;    0561  3                            BEGIN
;    0562  3                            MAP
;    0563  3                                    Branch_Value : BLOCK [, BYTE];
;    0564  3
;    0565  3                            Branch = NOT .Branch_Value [PSL$V_C]
;    0566  2                            END;
;    0567  2
;    0568  2                    [DS$K_BGTR_M] :
;    0569  3                            BEGIN
;    0570  3                            MAP
;    0571  3                                    Branch_Value : BLOCK [, BYTE];
;    0572  3
;    0573  3                            Branch = (NOT .Branch_Value [PSL$V_N]) OR
;    0574  4                                     (NOT .Branch_Value [PSL$V_Z])
;    0575  2                            END;
;    0576  2
;    0577  2                    [DS$K_BGTRU_M] :
;    0578  3                            BEGIN
;    0579  3                            MAP
;    0580  3                                    Branch_Value : BLOCK [, BYTE];
;    0581  3
;    0582  3                            Branch = (NOT .Branch_Value [PSL$V_C]) OR
;    0583  4                                     (NOT .Branch_Value [PSL$V_Z])
;    0584  2                            END;
;    0585  2
;    0586  2                    [DS$K_BNEQ_M] :
;    0587  3                            BEGIN
;    0588  3                            MAP
;    0589  3                                    Branch_Value : BLOCK [, BYTE];
;    0590  3
;    0591  3                            Branch = NOT .Branch_Value [PSL$V_Z]
;    0592  2                            END;
```

```
;    0594  2                    [DS$K_BNEQU_M] :
;    0595  3                            BEGIN
;    0596  3                            MAP
;    0597  3                                    Branch_Value : BLOCK [, BYTE];
;    0598  3
;    0599  3                            Branch = NOT .Branch_Value [PSL$V_Z]
;    0600  2                            END;
;    0601  2
;    0602  2                    [DS$K_BVS_M] :
;    0603  3                            BEGIN
;    0604  3                            MAP
;    0605  3                                    Branch_Value : BLOCK [, BYTE];
;    0606  3
;    0607  3                            Branch = .Branch_Value [PSL$V_V]
;    0608  2                            END;
;    0609  2
;    0610  2                    [DS$K_BVC_M] :
;    0611  3                            BEGIN
;    0612  3                            MAP
;    0613  3                                    Branch_Value : BLOCK [, BYTE];
;    0614  3
;    0615  3                            Branch = NOT .Branch_Value [PSL$V_V]
;    0616  2                            END;
;    0617  2
;    0618  2                    [DS$K_BCS_M] :
;    0619  3                            BEGIN
;    0620  3                            MAP
;    0621  3                                    Branch_Value : BLOCK [, BYTE];
;    0622  3
;    0623  3                            Branch = .Branch_Value [PSL$V_C]
;    0624  2                            END;
;    0625  2
;    0626  2                    [DS$K_BCC_M] :
;    0627  3                            BEGIN
;    0628  3                            MAP
;    0629  3                                    Branch_Value : BLOCK [, BYTE];
;    0630  3
;    0631  3                            Branch = NOT .Branch_Value [PSL$V_C]
;    0632  2                            END;
;    0633  2
;    0634  2                    [DS$K_BBS] :
;    0635  2                            Branch = .Branch_Value <0, 1, 0>;
;    0636  2
;    0637  2                    [DS$K_BBC] :
;    0638  2                            Branch = NOT .Branch_Value <0, 1, 0>;
```

```
;    0640  2                    !+
;    0641  2                    ! These are passed the value returned by the BUILTIN function for that branch (i.e.,
;    0642  2                    ! TESTBITSS for BBSS, etc.). Thus, the branch is based upon the value of the low bit.
;    0643  2                    !-
;    0644  2
;    0645  2                    [DS$K_BBSS] :
;    0646  2                            Branch = .Branch_Value <0, 1, 0>;
;    0647  2
;    0648  2                    [DS$K_BBCS] :
;    0649  2                            Branch = .Branch_Value <0, 1, 0>;
;    0650  2
;    0651  2                    [DS$K_BBSC] :
;    0652  2                            Branch = .Branch_Value <0, 1, 0>;
;    0653  2
;    0654  2                    [DS$K_BBCC] :
;    0655  2                            Branch = .Branch_Value <0, 1, 0>;
;    0656  2
;    0657  2                    [DS$K_BBSSI] :
;    0658  2                            Branch = .Branch_Value <0, 1, 0>;
;    0659  2
;    0660  2                    [DS$K_BBCCI] :
;    0661  2                            Branch = .Branch_Value <0, 1, 0>;
;    0662  2
;    0663  2                    !+
;    0664  2                    ! These are passed the value of the data whose low bit is being checked.
;    0665  2                    !-
;    0666  2
;    0667  2                    [DS$K_BLBS] :
;    0668  2                            Branch = .Branch_Value <0, 1, 0>;
;    0669  2
;    0670  2                    [DS$K_BLBC] :
;    0671  2                            Branch = NOT .Branch_Value <0, 1, 0>;
;    0672  2
;    0673  2                    !+
;    0674  2                    ! Treat the BERROR branch macro the same as the BNERROR branch macro. This means that the value [05]
;    0675  2                    ! of R0 will remain the same for the two cases of QA not running and of QA running but not      [05]
;    0676  2                    ! forcing errors. When QA is running and this branch macro should be reversed-sensed (yuch),   [05]
;    0677  2                    ! then R0 will return from a BERROR with the low bit complemented (as will happen with a       [05]
;    0678  2                    ! BNERROR). However, upon return from this routine, the branch macro for BERROR will do a BLBC [05]
;    0679  2                    ! instruction, as oposed to doing a BLBS (which is what the BNERROR macro will do). So, these  [05]
;    0680  2                    ! two branch macros will save the contents of R0 for the two cases above. They will also return [05]
;    0681  2                    ! the complemented low bit of R0 when forcing errors.                                          [05]
;    0682  2                    !-
;    0683  2
;    0684  2                    [DS$K_BERROR] :                         !                                                      [05]
;    0685  2                            Branch = .Branch_Value <0, 1, 0>;       ! These two should be the same.               [05]
;    0686  2
;    0687  2                    [DS$K_BNERROR] :                        !                                                      [05]
;    0688  2                            Branch = .Branch_Value <0, 1, 0>;       !                                             [05]
;    0689  2                    TES;
```

```
;   0691  2             !+
;   0692  2             ! If not running QA, simply return an indication of whether or not to branch.
;   0693  2             !-
;   0694  2
;   0695  2             IF (NOT .DSA$V_QA) THEN
;   0696  3                     BEGIN                                   !                                   [05]
;   0697  3                     Save_Register_0 <0, 1, 0> = .Branch <0, 1, 0>;  !                                   [05]
;   0698  4                     RETURN (.Save_Register_0)               !                                   [05]
;   0699  2                     END;                                    !                                   [05]
;   0700  2
;   0701  2             !+
;   0702  2             ! Make the Branch value a one or a zero by clearing out all the other bits.                [02]
;   0703  2             !-
;   0704  2
;   0705  2             Branch = Boolean (.Branch);                     !                                   [02]
;   0706  2
;   0707  2             !+
;   0708  2             ! Save the error number and branch value. Call QA$Main with the appropriate branch code.
;   0709  2             !-
;   0710  2
;   0711  2             QA$W_Branch       = .Branch;
;   0712  2             QA$W_Error_Number = .Error_Number;
;   0713  2
;   0714  2             QA$MAIN (DSQA$K_QA_DSX$Branch);
;   0715  2
;   0716  2             !+
;   0717  2             ! Set the low bit of R0 to what was set up in the Error Phase One routine in the QA$W_Branch variable.  [05]
;   0718  2             ! If I am forcing errors, the low bit of R0 can change.                                  [05]
;   0719  2             !-
;   0720  2
;   0721  2             Save_Register_0 <0, 1, 0> = .QA$W_Branch <0, 1, 0>;          !                                   [05]
;   0722  3             RETURN (.Save_Register_0)                                    !                                   [05]
;   0723  1     END;


                                                        .TITLE  QAMAIN *** QA Routines
                                                        .IDENT  \1-08\

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

                    4E  49  41  4D  41  51  06  00000 P.AAA:  .ASCII  <6>\QAMAIN\                                  ;

                                                        .PSECT  WORK,NOEXE,2

                                                00000 QA$AW_OVERALL_ERROR_TABLE:
                                                        .BLKB   30
                                                0001E QA$W_BRANCH::
                                                        .BLKB   2
                                                00020 QA$W_ERROR_NUMBER::
                                                        .BLKB   2
                                                00022 QA$W_SAVE_TEST::
                                                        .BLKB   2
                                                00024 QA$W_SAVE_LAST::
                                                        .BLKB   2
                                                00026   .BLKB   2
                                                00028 QA$L_SAVED_DSA_FLAGS::
                                                        .BLKB   4
```

```
                                                  0002C QA$AOB_ROUTINE_STATE::
                                                                .BLKB   1
                                                  0002D         .BLKB   3
                                                  00030 QA$AOB_CHECK_STATE::
                                                                .BLKB   2
                                                  00032         .BLKB   2
                                                  00034 QA$AOB_FLAGS::
                                                                .BLKB   1

                                                        DSA$AL_APTMAIL=     65024
                                                        DSA$GL_FLAGS=       65024
                                                        DSA$GL_APTCOM=      65028
                                                        DSA$GL_PASSES=      65032
                                                        DSA$GL_UNITS=       65036
                                                        DSA$GL_SECTNO=      65040
                                                        DSA$GL_SID=         65044
                                                        DSA$GL_MSGTYP=      65088
                                                        DSA$GL_ERRNO=       65092
                                                        DSA$GL_EVENT=       65096
                                                        DSA$GL_SUBTNO=      65100
                                                        DSA$GL_TESTNO=      65104
                                                        DSA$GL_PASSNO=      65108
                                                        DSA$GL_DEVLEN=      65112
                                                        DSA$GL_DEVNAM=      65116
                                                        DSA$GL_MSGPTR=      65128
                                                        $MODULE=            P.AAA
                                                                .EXTRN  QA$NORMAL_START
                                                                .EXTRN  QA$MULTIPLE_PASS
                                                                .EXTRN  QA$LOOP_ON_TEST
                                                                .EXTRN  QA$LOOP_ON_SUBTEST
                                                                .EXTRN  QA$RUN_BACKWARDS
                                                                .EXTRN  EXE$ALONONPAGED
                                                                .EXTRN  EXE$DEANONPAGED
                                                                .EXTRN  DSV$SHOWDEVICE, DSV$SHOWSELECT
                                                                .EXTRN  DSX$ABORT, VRSHOWFLG
                                                                .EXTRN  VRSHOWEF, DS$GL_FSTTEST
                                                                .EXTRN  DS$GL_LSTTEST, DS$GA_DS_CTRL_C_FIRST
                                                                .EXTRN  DS$GL_CLIBASE, DS$GB_WIDTH
                                                                .EXTRN  QA$W_TESTLOOPS, QA$W_SUBTESTLOOPS
                                                                .EXTRN  QA$W_MULTIPLEPASS
                                                                .EXTRN  QA$T_HEADER_1, QA$T_HEADER_2
                                                                .EXTRN  QA$T_HEADER_3, QA$T_OPERATOR_REQUEST_MESSAGE

                                                                .PSECT  CODE,NOWRT,  SHR,2

                                          001E 00000            .ENTRY  DSX$BRANCH, Save R1,R2,R3,R4                    ; 0403
                          54 00000000'  EF  9E 00002            MOVAB   QA$W_BRANCH, R4                                 ;
                          52            50  D0 00009            MOVL    REGISTER_0, SAVE_REGISTER_0                     ; 0453
                 27       00        04  AC  CF 0000C            CASEL   BRANCH_CODE, #0, #39                            ; 0455
0066          005D      0059          0050     00011 1$:        .WORD   2$-1$,-                                         ;
0078          006F      0066          0066     00019                    3$-1$,-                                         ;
0085          0085      0085          007C     00021                    4$-1$,-                                         ;
00A0          0098      0101          0090     00029                    5$-1$,-                                         ;
00F6          00BA      00B2          00B2     00031                    5$-1$,-                                         ;
00DE          00DE      00CA          00C2     00039                    5$-1$,-                                         ;
00F6          0101      00EE          00E6     00041                    6$-1$,-                                         ;
0101          0101      00F6          0101     00049                    7$-1$,-                                         ;
```

```
     0101        0101        0101        0101     00051                8$-1$,-                                  ;
     0101        0101        00F6        0101     00059                9$-1$,-                                  ;
                                                                       9$-1$,-                                  ;
                                                                       9$-1$,-                                  ;
                                                                       11$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       12$-1$,-                                 ;
                                                                       13$-1$,-                                 ;
                                                                       15$-1$,-                                 ;
                                                                       15$-1$,-                                 ;
                                                                       17$-1$,-                                 ;
                                                                       25$-1$,-                                 ;
                                                                       18$-1$,-                                 ;
                                                                       19$-1$,-                                 ;
                                                                       21$-1$,-                                 ;
                                                                       21$-1$,-                                 ;
                                                                       22$-1$,-                                 ;
                                                                       24$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       25$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       25$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       25$-1$,-                                 ;
                                                                       27$-1$,-                                 ;
                                                                       27$-1$                                   ;
                                  51  D4 00061 2$:     CLRL     R1                                       ; 0462
                              0C  AC  D5 00063         TSTL     BRANCH_VALUE                             ;
                                  61  18 00066         BGEQ     16$                                      ;
                                  33  11 00068         BRB      10$                                      ;
                                  51  D4 0006A 3$:     CLRL     R1                                       ; 0465
                                  5B  11 0006C         BRB      16$                                      ;
                                  51  D4 0006E 4$:     CLRL     R1                                       ; 0468
                              0C  AC  D5 00070         TSTL     BRANCH_VALUE                             ;
                                  54  14 00073         BGTR     16$                                      ;
                                  26  11 00075         BRB      10$                                      ;
                                  51  D4 00077 5$:     CLRL     R1                                       ; 0471
                              0C  AC  D5 00079         TSTL     BRANCH_VALUE                             ;
                                  7F  12 0007C         BNEQ     23$                                      ;
                                  1D  11 0007E         BRB      10$                                      ;
                                  51  D4 00080 6$:     CLRL     R1                                       ; 0480
                              0C  AC  D5 00082         TSTL     BRANCH_VALUE                             ;
                                  76  19 00085         BLSS     23$                                      ;
                                  14  11 00087         BRB      10$                                      ;
                                  51  D4 00089 7$:     CLRL     R1                                       ; 0483
                                  10  11 0008B         BRB      10$                                      ;
                                  51  D4 0008D 8$:     CLRL     R1                                       ; 0486
                              0C  AC  D5 0008F         TSTL     BRANCH_VALUE                             ;
                                  69  15 00092         BLEQ     23$                                      ;
                                  07  11 00094         BRB      10$                                      ;
                                  51  D4 00096 9$:     CLRL     R1                                       ; 0495
```

```
                                    0C   AC  D5 00098          TSTL    BRANCH_VALUE                                  ;
                                         7B  13 0009B          BEQL    28$                                           ;
                                         51  D6 0009D 10$:     INCL    R1                                            ;
                                         77  11 0009F          BRB     28$                                           ;
     51      0C   AC          01         03  EF 000A1 11$:     EXTZV   #3, #1, BRANCH_VALUE, R1                      ; 0507
                                         6F  11 000A7          BRB     28$                                           ;
     51      0C   AC          01         03  EF 000A9 12$:     EXTZV   #3, #1, BRANCH_VALUE, R1                      ; 0524
                                         06  11 000AF          BRB     14$                                           ;
     51      0C   AC          01         00  EF 000B1 13$:     EXTZV   #0, #1, BRANCH_VALUE, R1                      ; 0533
     53      0C   AC          01         02  EF 000B7 14$:     EXTZV   #2, #1, BRANCH_VALUE, R3                      ;
                  50          51         53  89 000BD          BISB3   R3, R1, BRANCH                                ;
                                         58  11 000C1          BRB     29$                                           ; 0528
     51      0C   AC          01         02  EF 000C3 15$:     EXTZV   #2, #1, BRANCH_VALUE, R1                      ; 0549
                                         4D  11 000C9 16$:     BRB     28$                                           ;
     51      0C   AC          01         03  EF 000CB 17$:     EXTZV   #3, #1, BRANCH_VALUE, R1                      ; 0557
                                         3A  11 000D1          BRB     26$                                           ;
     51      0C   AC          01         03  EF 000D3 18$:     EXTZV   #3, #1, BRANCH_VALUE, R1                      ; 0574
                                         06  11 000D9          BRB     20$                                           ;
     51      0C   AC          01         00  EF 000DB 19$:     EXTZV   #0, #1, BRANCH_VALUE, R1                      ; 0583
     53      0C   AC          01         02  EF 000E1 20$:     EXTZV   #2, #1, BRANCH_VALUE, R3                      ;
                              53         53  D2 000E7          MCOML   R3, R3                                        ;
                              51         53  CA 000EA          BICL2   R3, R1                                        ;
                                         1E  11 000ED          BRB     26$                                           ; 0582
     51      0C   AC          01         02  EF 000EF 21$:     EXTZV   #2, #1, BRANCH_VALUE, R1                      ; 0599
                                         16  11 000F5          BRB     26$                                           ;
     51      0C   AC          01         01  EF 000F7 22$:     EXTZV   #1, #1, BRANCH_VALUE, R1                      ; 0607
                                         19  11 000FD 23$:     BRB     28$                                           ;
     51      0C   AC          01         01  EF 000FF 24$:     EXTZV   #1, #1, BRANCH_VALUE, R1                      ; 0615
                                         06  11 00105          BRB     26$                                           ;
     51      0C   AC          01         00  EF 00107 25$:     EXTZV   #0, #1, BRANCH_VALUE, R1                      ; 0671
                              50         51  92 0010D 26$:     MCOMB   R1, BRANCH                                    ;
                                         09  11 00110          BRB     29$                                           ;
     51      0C   AC          01         00  EF 00112 27$:     EXTZV   #0, #1, BRANCH_VALUE, R1                      ; 0688
                              50         51  90 00118 28$:     MOVB    R1, BRANCH                                    ;
                                 0000FE01 9F  95 0011B 29$:     TSTB    @#^X0000FE01                                  ; 0695
                                         07  19 00121          BLSS    30$                                           ;
     52          01           00         50  F0 00123          INSV    BRANCH, #0, #1, SAVE_REGISTER_0               ; 0697
                                         1E  11 00128          BRB     31$                                           ; 0698
     51          50           01         00  EF 0012A 30$:     EXTZV   #0, #1, BRANCH, R1                            ; 0705
                              50         51  90 0012F          MOVB    R1, BRANCH                                    ;
                              64         50  9B 00132          MOVZBW  BRANCH, QA$W_BRANCH                           ; 0711
                        02    A4     08  AC  B0 00135          MOVW    ERROR_NUMBER, QA$W_ERROR_NUMBER               ; 0712
                                         11  DD 0013A          PUSHL   #17                                           ; 0714
                  00000000V   EF         01  FB 0013C          CALLS   #1, QA$MAIN                                   ;
     52          01           00         64  F0 00143          INSV    QA$W_BRANCH, #0, #1, SAVE_REGISTER_0          ; 0721
                              50         52  D0 00148 31$:     MOVL    SAVE_REGISTER_0, R0                           ; 0722
                                             04 0014B          RET                                                   ; 0723

; Routine Size:  332 bytes,    Routine Base:  CODE + 0000
```

```
%SBTTL 'QA$Abort_QA'

ROUTINE QA$Abort_QA : NOVALUE =                          !                                  This whole routine is [02]

!++
! FUNCTIONAL DESCRIPTION:
!
!       Dump out debugging information, print the overall error summary table, restore the DSA flags,
!       and abort the diagnostic (which will cause DS_CLEANUP to be executed, then branches to BEGIN).
!
! CALLERS:
!
!       QA$Main
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       DSA$V_QA            - The QA bit in the DSA flags. It is cleared to indicate QA is no longer running.
!       QA$L_Saved_QA_Flags - The saved DSA flags. The flags are restored here to what they were before
!                             the START/QA (or RUN/QA) command started executing.
!
! IMPLICIT OUTPUTS:
!
!       Debugging information, Overall Error Summary Table, and an abortion message.
!
! COMPLETION CODES:
!
!       NONE - This routine does not return to its caller(s). It calls the DSX$Abort routine, which
!              branches to BEGIN, which goes back to DS$Cli.
!
! SIDE EFFECTS:
!
!       The diagnostic is aborted.
!
!--
```

```
;   0764  2     BEGIN                                                   !                                                   [02]
;   0765  2             QA$Dump ();                                     ! Dump out information for the user                 [02]
;   0766  2             QA$Print_Overall_Errors ();                     ! Print the summary table                           [02]
;   0767  2             DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;           ! Restore DSA flags                                 [02]
;   0768  2             QA$Init ();                                     ! Initialize for next run                           [02]
;   0769  2             DSA$V_QA = Off;                                 ! Turn off QA                                       [02]
;   0770  2             DSX$Abort ()                                    ! Abort the diagnostic because of the error         [02]
;   0771  2                                                             ! that was found. DSX$Abort will not return to here!  [02]
;   0772  1     END;


                                                  0000 00000 QA$ABORT_QA:
                                                                  .WORD   Save nothing                                  ; 0727
                        00000000V  EF            00  FB 00002     CALLS   #0, QA$DUMP                                   ; 0765
                        00000000V  EF            00  FB 00009     CALLS   #0, QA$PRINT_OVERALL_ERRORS                   ; 0766
                        0000FE00   9F 00000000'  EF  D0 00010     MOVL    QA$L_SAVED_DSA_FLAGS, a#^X0000FE00            ; 0767
                        00000000V  EF            00  FB 0001B     CALLS   #0, QA$INIT                                   ; 0768
                        0000FE01   9F       80   8F  8A 00022     BICB2   #128, a#^X0000FE01                            ; 0769
                        00000000G  EF            00  FB 0002A     CALLS   #0, DSX$ABORT                                 ; 0770
                                                     04 00031     RET                                                   ; 0772

; Routine Size:  50 bytes,    Routine Base:  CODE + 014C
```

```
%SBTTL 'QA$Add_Forced_Error'

GLOBAL ROUTINE QA$Add_Forced_Error (Error_Number : WORD) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine adds an error to the Forced Error Summary Table.
!
! CALLER (S):
!
!       The Error check routines.
!
! FORMAL PARAMETERS:
!
!       Error_Number : The DS error number.
!
! IMPLICIT INPUTS:
!
!       QA$A_Error_List_Head - Pointer to the head of a linked list of Forced-Error data entries.
!       QA$A_Prev_Test_Tail  - Pointer to the previous test's last entry in the linked list.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Adds a record to the linked list of forced-error records.
!
! FIGURES:
!
!       The Forced-Error Record:
!
!       ------------------------------------------------------------------------------------------------
!       |             |             |                       |                       |                       |
!       |Error_Number | Test_Number |      Error_Count      |      Subtest_Number   |      Next_Error       |
!       |     |       |     |       |     |     |     |     |     |     |     |     |     |     |     |
!       ------------------------------------------------------------------------------------------------
!       Error_Number   - The error number passed to the $DS_ERRxxxx routine.
!       Test_Number    - The test that the error occurred in.
!       Error_Count    - The number of times that error has been forced.
!       Subtest_Number - The subtest that the error occurred in.
!       Next_Error     - The pointer to the next Forced-Error record in the linked list, or nil.
!--
```

```
;    0824  1     %SBTTL 'New'
;    0825  2     BEGIN
;    0826  2     !       !++
;    0827  2     !       ! FUNCTIONAL DESCRIPTION:
;    0828  2     !       !
;    0829  2     !       !       This routine returns a 16-byte vector for use by the QA$Add_Forced_Error routine.
;    0830  2     !       !
;    0831  2     !       ! CALLER (S):
;    0832  2     !       !
;    0833  2     !       !       QA$K_Add_Forced_Error
;    0834  2     !       !
;    0835  2     !       ! FORMAL PARAMETERS:
;    0836  2     !       !
;    0837  2     !       !       Record_Ptr - Contains the address of the 16-byte vector.
;    0838  2     !       !
;    0839  2     !       ! IMPLICIT INPUTS:
;    0840  2     !       !
;    0841  2     !       !       NONE
;    0842  2     !       !
;    0843  2     !       ! IMPLICIT OUTPUTS:
;    0844  2     !       !
;    0845  2     !       !       NONE
;    0846  2     !       !
;    0847  2     !       ! COMPLETION CODES:
;    0848  2     !       !
;    0849  2     !       !       NONE
;    0850  2     !       !
;    0851  2     !       ! SIDE EFFECTS:
;    0852  2     !       !
;    0853  2     !       !       Will abort the QA sequence if the returned size of the vector is not 16, or if the vector
;    0854  2     !       !       cannot be allocated.
;    0855  2     !       !
;    0856  2     !       !--
```

```
;    0857  2      !
;    0858  2      !       ROUTINE New (Record_Ptr) : NOVALUE =
;    0859  2      !       BEGIN
;    0860  2      !               LOCAL
;    0861  2      !                       Returned_Size;
;    0862  2      !
;    0863  2      !               IF (NOT EXE$AloNonPaged (QA$K_Forced_Error_Record_Size; Returned_Size, .Record_Ptr)) THEN
;    0864  2      !                       BEGIN
;    0865  2      !                       Logic_Error ('Cannot allocate forced-error record');
;    0866  2      !                       QA$Abort_QA ();
;    0867  2      !                       END;
;    0868  2      !
;    0869  2      !               IF (.Returned_Size NEQ QA$K_Forced_Error_Record_Size) THEN
;    0870  2      !                       BEGIN
;    0871  2      !                       Logic_Error ('Wrong size forced-error record allocated');
;    0872  2      !                       QA$Abort_QA ();
;    0873  2      !                       END;
;    0874  2      !       END;
```

```
;    0875  2      !
;    0876  2      !%SBTTL 'Add_Error_Record'
;    0877  2      !
;    0878  2      !       ROUTINE Add_Error_Record (Record_Ptr, Error_Number : WORD) : NOVALUE =
;    0879  2      !
;    0880  2      !       !++
;    0881  2      !       ! FUNCTIONAL DESCRIPTION:
;    0882  2      !       !
;    0883  2      !       !       This routine adds one record to the linked list of forced-error records. It also fills in the
;    0884  2      !       !       fields of the new record. The Error_Count is assumed to be one. The Test and Subtest numbers
;    0885  2      !       !       are taken from the DSA area. This assumes that the forced-error is being added as they occur
;    0886  2      !       !       (i.e., there is no change in the test/subtest numbers from the time the error is forced to the
;    0887  2      !       !       time it is added).
;    0888  2      !       !
;    0889  2      !       ! CALLER (S):
;    0890  2      !       !
;    0891  2      !       !       QA$Add_Forced_Error
;    0892  2      !       !
;    0893  2      !       ! FORMAL PARAMETERS:
;    0894  2      !       !
;    0895  2      !       !       Record_Ptr      - Contains the address of the record which will preceed this new record.
;    0896  2      !       !       Error_Number    - The error number that will be stored in the new record.
;    0897  2      !       !
;    0898  2      !       ! IMPLICIT INPUTS:
;    0899  2      !       !
;    0900  2      !       !       DSA$GL_TESTNO - The DSA test number.
;    0901  2      !       !       DSA$GL_SUBTNO - The DSA subtest number.
;    0902  2      !       !
;    0903  2      !       ! IMPLICIT OUTPUTS:
;    0904  2      !       !
;    0905  2      !       !       NONE
;    0906  2      !       !
;    0907  2      !       ! COMPLETION CODES:
;    0908  2      !       !
;    0909  2      !       !       NONE
;    0910  2      !       !
;    0911  2      !       ! SIDE EFFECTS:
;    0912  2      !       !
;    0913  2      !       !       NONE
;    0914  2      !       !
;    0915  2      !       !--
```

```
;    0916  2      !
;    0917  2      !       BEGIN
;    0918  2      !               LOCAL
;    0919  2      !                       New_Record_Ptr;
;    0920  2      !
;    0921  2      !               MAP
;    0922  2      !                       DSA$GL_TESTNO : WORD,
;    0923  2      !                       Record_Ptr    : Error_Record;
;    0924  2      !
;    0925  2      !               New (New_Record_Ptr);
;    0926  2      !
;    0927  2      !               BEGIN
;    0928  2      !                       MAP
;    0929  2      !                               New_Record_Ptr : Error_Record;
;    0930  2      !
;    0931  2      !                       New_Record_Ptr [QA$VW_Error_Number]   = .Error_Number;
;    0932  2      !                       New_Record_Ptr [QA$VW_Test_Number]    = .DSA$GL_TESTNO;
;    0933  2      !                       New_Record_Ptr [QA$VL_Subtest_Number] = .DSA$GL_SUBTNO;
;    0934  2      !                       New_Record_Ptr [QA$VL_Error_Count]    = 1;
;    0935  2      !                       New_Record_Ptr [QA$VA_Next_Error]     = .Record_Ptr [QA$VA_Next_Error]
;    0936  2      !               END;
;    0937  2      !
;    0938  2      !               Record_Ptr [QA$VA_Next_Error] = .New_Record_Ptr;
;    0939  2      !       END;
```

```
;    0940  2      !
;    0941  2      !%SBTTL 'QA$Add_Forced_Error'
;    0942  2      !
;    0943  2      !       LOCAL
;    0944  2      !               Test : WORD,                            ! The DSA$GL_TESTNO.
;    0945  2      !
;    0946  2      !               Subtest,                                ! The DSA$GL_SUBTNO.
;    0947  2      !
;    0948  2      !               Prev_Record_Ptr,                        ! The pointer to the record before the Record_Ptr record.
;    0949  2      !
;    0950  2      !               Record_Ptr;                             ! The pointer used to sneak through the linked list.
;    0951  2      !
;    0952  2      !       !+
;    0953  2      !       ! Check to see if the Error_List_Head is nil. If so, this is the first record to be added. Add a new
;    0954  2      !       ! record that contains all zero fields. Then, add another record after this one that will contain the
;    0955  2      !       ! correct information. The null record is used to make adding the first record easier. It also helps
;    0956  2      !       ! facilitate adding records before the first record. Set both of the global linked list pointers to
;    0957  2      !       ! point to this null record.
;    0958  2      !       !-
;    0959  2      !
;    0960  2      !       IF (.QA$A_Error_List_Head EQL QA$K_Nil) THEN
;    0961  2      !               BEGIN
;    0962  2      !               LOCAL
;    0963  2      !                       New_Record_Ptr;
;    0964  2      !
;    0965  2      !               New (New_Record_Ptr);
;    0966  2      !
;    0967  2      !               BEGIN
;    0968  2      !                       MAP
;    0969  2      !                               New_Record_Ptr : Error_Record;
;    0970  2      !
;    0971  2      !               New_Record_Ptr [QA$VW_Error_Number]    = 0;
;    0972  2      !               New_Record_Ptr [QA$VW_Test_Number]     = 0;
;    0973  2      !               New_Record_Ptr [QA$VL_Subtest_Number] = 0;
;    0974  2      !               New_Record_Ptr [QA$VL_Error_Count]     = 0;
;    0975  2      !               New_Record_Ptr [QA$VA_Next_Error]      = QA$K_Nil;
;    0976  2      !               END;
;    0977  2      !
;    0978  2      !               Add_Error_Record (.New_Record_Ptr, .Error_Number);
;    0979  2      !
;    0980  2      !               QA$A_Prev_Test_Tail  = .New_Record_Ptr;
;    0981  2      !               QA$A_Error_List_Head = .New_Record_Ptr;
;    0982  2      !               RETURN
;    0983  2      !               END;
```

```
;    0984  2      !
;    0985  2      !       !+
;    0986  2      !       ! Start looking at the current head of the linked list that has all the test and subtest numbers
;    0987  2      !       ! the same. This list appears at the end of the forced-error linked list.
;    0988  2      !       !-
;    0989  2      !
;    0990  2      !       BEGIN
;    0991  2      !               MAP
;    0992  2      !                       QA$A_Prev_Test_Tail : Error_Record;
;    0993  2      !
;    0994  2      !               Prev_Record_Ptr = .QA$A_Prev_Test_Tail;
;    0995  2      !               Record_Ptr      = .QA$A_Prev_Test_Tail [QA$VA_Next_Error]
;    0996  2      !       END;
;    0997  2      !
;    0998  2      !       BEGIN
;    0999  2      !               MAP
;    1000  2      !                       Record_Ptr : Error_Record;
;    1001  2      !
;    1002  2      !               Test    = .Record_Ptr [QA$VW_Test_Number];
;    1003  2      !               Subtest = .Record_Ptr [QA$VL_Subtest_Number]
;    1004  2      !       END;
;    1005  2      !
;    1006  2      !       !+
;    1007  2      !       ! Check the start of the last link of like test/subtest numbers. See if this forced error happened
;    1008  2      !       ! in a new test and/or subtest. If so, add one record to the end of this list.
;    1009  2      !       !-
;    1010  2      !
;    1011  2      !       IF ((.Test NEQ .DSA$GL_TESTNO) OR (.Subtest NEQ .DSA$GL_SUBTNO)) THEN
;    1012  2      !               BEGIN
;    1013  2      !               !+
;    1014  2      !               ! Then we have forced an error call in a new test and/or subtest. Find the end of the linked list
;    1015  2      !               ! of forced-errors and add a new record onto the end. Save a pointer to the previous end record.
;    1016  2      !               !-
;    1017  2      !               MAP
;    1018  2      !                       Record_Ptr : Error_Record;
;    1019  2      !
;    1020  2      !               While (.Record_Ptr [QA$VA_Next_Error] NEQ QA$K_Nil) DO
;    1021  2      !                       Record_Ptr = .Record_Ptr [QA$VA_Next_Error];
;    1022  2      !
;    1023  2      !               QA$A_Prev_Test_Tail = .Record_Ptr;
;    1024  2      !
;    1025  2      !               Add_Error_Record (.Record_Ptr, .Error_Number);
;    1026  2      !               RETURN
;    1027  2      !               END;
```

```
;    1028  2      !
;    1029  2      !       !+
;    1030  2      !       ! Thus, at this point we do NOT have:
;    1031  2      !       !       1. A new list (i.e., no records).
;    1032  2      !       !       2. A new test and/or subtest number (i.e., the error number being added happened in the same
;    1033  2      !       !          test and subtest as the previously-added error number happened in.
;    1034  2      !       ! So, look along the error numbers in the linked list of records whose test/subtest numbers match for the
;    1035  2      !       ! correct place to insert this error number.
;    1036  2      !       !-
;    1037  2      !
;    1038  2      !       BEGIN
;    1039  2      !               MAP
;    1040  2      !                       Record_Ptr : Error_Record;
;    1041  2      !
;    1042  2      !               LOCAL
;    1043  2      !                       Curr_Error_Number;
;    1044  2      !
;    1045  2      !               WHILE (1) DO
;    1046  2      !                       BEGIN
;    1047  2      !                       Curr_Error_Number = .Record_Ptr [QA$VW_Error_Number];
;    1048  2      !
;    1049  2      !                       IF (.Curr_Error_Number LSS .Error_Number) THEN
;    1050  2      !                               BEGIN
;    1051  2      !                               !+
;    1052  2      !                               ! If the next one is nil, add the new one at the end. If it is not nil,
;    1053  2      !                               ! advance the pointer and continue looping.
;    1054  2      !                               !-
;    1055  2      !                               IF (.Record_Ptr [QA$VA_Next_Error] EQL QA$K_Nil) THEN
;    1056  2      !                                       BEGIN
;    1057  2      !                                       Add_Error_Record (.Record_Ptr, .Error_Number);
;    1058  2      !                                       RETURN
;    1059  2      !                                       END
;    1060  2      !                               ELSE
;    1061  2      !                                       BEGIN
;    1062  2      !                                       Prev_Record_Ptr = .Record_Ptr;
;    1063  2      !                                       Record_Ptr      = .Record_Ptr [QA$VA_Next_Error]
;    1064  2      !                                       END
;    1065  2      !                               END
;    1066  2      !                       ELSE IF (.Curr_Error_Number EQL .Error_Number) THEN
;    1067  2      !                               BEGIN
;    1068  2      !                               !+
;    1069  2      !                               ! This error number is already in the linked list. Add one to the total number
;    1070  2      !                               ! of times it has occurred.
;    1071  2      !                               !-
;    1072  2      !                               Record_Ptr [QA$VL_Error_Count] = .Record_Ptr [QA$VL_Error_Count] + 1;
;    1073  2      !                               RETURN
;    1074  2      !                               END
;    1075  2      !                       ELSE                    ! Curr_Error_Number GTR Error_Number
;    1076  2      !                               BEGIN
;    1077  2      !                               !+
;    1078  2      !                               ! This error number belongs just before the current record being looked at.
;    1079  2      !                               ! Therefore, add a new record there.
;    1080  2      !                               !-
;    1081  2      !                               Add_Error_Record (.Prev_Record_Ptr, .Error_Number);
;    1082  2      !                               RETURN
;    1083  2      !                               END;
;    1084  2      !                       END;                    ! WHILE ...
```

```
;   1085  2     !       END;                                    ! BEGIN ...
;   1086  2
;   1087  2             1;                                      !
;   1088  2
;   1089  1     END;


                                        0000 00000        .ENTRY  QA$ADD_FORCED_ERROR, Save nothing          ; 0776
                                          04 00002        RET                                                ; 1089

; Routine Size:  3 bytes,    Routine Base:  CODE + 017E
```

```
;   1091  1     %SBTTL 'QA$Add_QA_Error'
;   1092  1
;   1093  1     GLOBAL ROUTINE QA$Add_QA_Error (Routine_Constant) : NOVALUE =
;   1094  1
;   1095  1     !++
;   1096  1     ! FUNCTIONAL DESCRIPTION:
;   1097  1     !
;   1098  1     !       This routine adds an error to the QA$AW_Overall_Error_Table word vector.
;   1099  1     !
;   1100  1     ! CALLER(S):
;   1101  1     !
;   1102  1     !       The check routines.
;   1103  1     !
;   1104  1     ! FORMAL PARAMETERS:
;   1105  1     !
;   1106  1     !       Routine_Constant - One of the QA$K_check-name constants.
;   1107  1     !
;   1108  1     ! IMPLICIT INPUTS:
;   1109  1     !
;   1110  1     !       QA$AW_Overall_Error_Table
;   1111  1     !
;   1112  1     ! IMPLICIT OUTPUTS:
;   1113  1     !
;   1114  1     !       NONE
;   1115  1     !
;   1116  1     ! COMPLETION CODES:
;   1117  1     !
;   1118  1     !       NONE
;   1119  1     !
;   1120  1     ! SIDE EFFECTS:
;   1121  1     !
;   1122  1     !       Adds one to the Routine_Constant entry in the QA$AW_Overall_Error_Table.
;   1123  1     !
;   1124  1     !--
;   1125  1
;   1126  2     BEGIN
;   1127  2             QA$AW_Overall_Error_Table [.Routine_Constant] =
;   1128  2                     .QA$AW_Overall_Error_Table [.Routine_Constant] + 1
;   1129  1     END;



                                        0000 00000        .ENTRY  QA$ADD_QA_ERROR, Save nothing                  ; 1093
                          50       04   AC  D0 00002        MOVL    ROUTINE_CONSTANT, R0                           ; 1127
                             00000000'EF40  B6 00006        INCW    QA$AW_OVERALL_ERROR_TABLE[R0]                  ; 1128
                                            04 0000D        RET                                                    ; 1129

; Routine Size:  14 bytes,    Routine Base:  CODE + 0181
```

```
%SBTTL  'QA$Control_C_Handler'

ROUTINE QA$Control_C_Handler =                                  !                                                    [06]

!++                                                                                                                  [06]
! FUNCTIONAL DESCRIPTION:                                                                                            [06]
!                                                                                                                    [06]
!       This routine is called by the Supervisor's Control-C (i.e., ^C) handler, KB_Check in the KERNEL.MAR          [06]
!       module. It will abort QA. This routine is set up as a handler by QA$Main.                                   [06]
!                                                                                                                    [06]
! FORMAL PARAMETERS:                                                                                                 [06]
!                                                                                                                    [06]
!       NONE.                                                                                                        [06]
!                                                                                                                    [06]
! CALLER(S):                                                                                                         [06]
!                                                                                                                    [06]
!       KB_CHECK in the KERNEL.MAR module.                                                                           [06]
!                                                                                                                    [06]
! IMPLICIT INPUTS:                                                                                                   [06]
!                                                                                                                    [06]
!       NONE                                                                                                         [06]
!                                                                                                                    [06]
! IMPLICIT OUTPUTS:                                                                                                  [06]
!                                                                                                                    [06]
!       NONE                                                                                                         [06]
!                                                                                                                    [06]
! COMPLETION CODES:                                                                                                  [06]
!                                                                                                                    [06]
!       None, really. It is defined as a VALUE routine howver, because KB_CHECK looks at the return status          [06]
!       by this routine (however, this routine does not return)!                                                     [06]
!                                                                                                                    [06]
! SIDE EFFECTS:                                                                                                      [06]
!                                                                                                                    [06]
!       Aborts the diagnostic and the QA sequence of checks. There are no completion codes because control does [06]
!       not return to here after the DSX$Abort routine call. Control will eventually get to DS$Cli in the           [06]
!       CLI.MAR module.                                                                                              [06]
!                                                                                                                    [06]
!--                                                                                                                  [06]
```

```
;   1170  2     BEGIN                                                   !                                                   [06]
;   1171  2             $DS_CNTRLC (DISABL=1);                          ! Disable Control-C's                               [06]
;   1172  2             DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;           ! Restore DSA flags                                 [06]
;   1173  2             QA$Init ();                                     ! Initialize for next run                           [06]
;   1174  2             DSA$V_QA = Off;                                 ! Turn off QA                                       [06]
;   1175  2             DSX$Abort ();                                   ! Abort the diagnostic, return to CLI.              [06]
;   1176  3             RETURN (1)                                      ! This will never be executed, because of the       [06]
;   1177  3                                                             ! above Abort.                                      [06]
;   1178  1     END;                                                    !                                                   [06]


                                                                .EXTRN  DS$CNTRLC

                                                  0000 00000 QA$CONTROL_C_HANDLER:
                                                                .WORD   Save nothing                                    ; 1133
                                                 01  DD 00002   PUSHL   #1                                              ; 1171
                                                 7E  D4 00004   CLRL    -(SP)                                           ;
                    00000000G  9F                02  FB 00006   CALLS   #2, @#DS$CNTRLC                                 ;
                    0000FE00   9F 00000000'      EF  D0 0000D   MOVL    QA$L_SAVED_DSA_FLAGS, @#^X0000FE00              ; 1172
                    00000000V  EF                00  FB 00018   CALLS   #0, QA$INIT                                     ; 1173
                    0000FE01   9F        80      8F  8A 0001F   BICB2   #128, @#^X0000FE01                              ; 1174
                    00000000G  EF                00  FB 00027   CALLS   #0, DSX$ABORT                                   ; 1175
                               50                01  D0 0002E   MOVL    #1, R0                                          ; 1176
                                                     04 00031   RET                                                     ; 1178

; Routine Size:  50 bytes,    Routine Base:  CODE + 018F
```

```
;    1180  1      %SBTTL 'QA$Dump'
;    1181  1
;    1182  1      ROUTINE QA$Dump : NOVALUE =
;    1183  1
;    1184  1      !++
;    1185  1      ! FUNCTIONAL DESCRIPTION:
;    1186  1      !
;    1187  1      !       This routine dumps pertinent information whenever a QA error is
;    1188  1      !       detected. If the users diagnostic has been aborted (i.e., an
;    1189  1      !       INIT_CONTEXT has been done), this routine will fail because the
;    1190  1      !       user's call frames are gone.
;    1191  1      !
;    1192  1      ! CALLER:
;    1193  1      !
;    1194  1      !       QA$Main - This routine should be called when a QA error is
;    1195  1      !                 detected. QA$Main should be its only caller.
;    1196  1      !
;    1197  1      ! FORMAL PARAMETERS:
;    1198  1      !
;    1199  1      !       NONE
;    1200  1      !
;    1201  1      ! IMPLICIT INPUTS:
;    1202  1      !
;    1203  1      !       Various DSA and diagnostic header (i.e., L$x_) values and flags.
;    1204  1      !
;    1205  1      ! IMPLICIT OUTPUTS:
;    1206  1      !
;    1207  1      !       If possible, helpful debugging information is printed.
;    1208  1      !
;    1209  1      ! COMPLETION CODES:
;    1210  1      !
;    1211  1      !       NONE
;    1212  1      !
;    1213  1      ! SIDE EFFECTS:
;    1214  1      !
;    1215  1      !       NONE
;    1216  1      !
;    1217  1      !--
```

```
;   1219  2     BEGIN
;   1220  2             BUILTIN
;   1221  2                     FP, SP, AP,
;   1222  2                     MOVPSL;
;   1223  2
;   1224  2             LITERAL
;   1225  2                     Reg_Not_Saved       = %X'00000000',     ! Special value to indicate that the register wasn't saved
;   1226  2                     Max_Regs_Saved      = 12;               ! Maximum that can be saved in a call frame.
;   1227  2
;   1228  2             LOCAL
;   1229  2                     Frame_Start,                            ! Start of the user's call frame
;   1230  2
;   1231  2                     !+
;   1232  2                     ! Stuff contained in a call frame:
;   1233  2                     !-
;   1234  2
;   1235  2                     Diag_PSW : WORD,                        ! Only get PSW - not PSL
;   1236  2                     Register_Mask,                          ! Call frame register mask
;   1237  2                     Diag_AP,
;   1238  2                     Diag_FP,
;   1239  2                     Diag_PC,                                ! Saved registers
;   1240  2                     Diag_SP,
;   1241  2
;   1242  2                     Number_Of_Regs;                         ! Number of regs in the register save mask
;   1243  2
;   1244  2             LOCAL
;   1245  2                     Fake_CLI_Table : VECTOR [4, LONG],      ! Fake DS$GL_CLIBASE
;   1246  2                     Reg_Vector     : VECTOR [Max_Regs_Saved, LONG]; ! Store 12 registers
;   1247  2
;   1248  2             BIND
;   1249  2                     Dump_Header_4 = $ASCIC ('!25* QA ERROR DETECTED!/!/'),
;   1250  2                     Dump_Line_1   = $ASCIC ('Test: !3UL!9* Subtest: !3UL!6* Pass: !3UL!9* Section: !AC!/!/'),   !    [02]
;   1251  2                     Dump_Line_2   = $ASCIC ('R2:  !8XL(X)!2* R3:  !8XL(X)!2* R4:  !8XL(X)!2* R5:  !8XL(X)!/'),  !    [02]
;   1252  2                     Dump_Line_3   = $ASCIC ('R6:  !8XL(X)!2* R7:  !8XL(X)!2* R8:  !8XL(X)!2* R9:  !8XL(X)!/'),  !    [02]
;   1253  2                     Dump_Line_4   = $ASCIC ('R10: !8XL(X)!2* R11: !8XL(X)!2* AP:  !8XL(X)!2* FP:  !8XL(X)!/'),  !    [02]
;   1254  2                     Dump_Line_5   = $ASCIC ('SP:  !8XL(X)!2* PC:  !8XL(X)!2* PSW: !8XW(X)!/'),                   !    [07]
;   1255  2                     Dump_Line_7   = $ASCIC ('!/SHOW FLAGS:!/'),                                                  !    [07]
;   1256  2                     Dump_Line_8   = $ASCIC ('!/SHOW EVENT FLAGS:!/'),                                            !    [07]
;   1257  2                     Dump_Line_9   = $ASCIC ('!/SHOW DEVICE:!/'),                                                 !    [07]
;   1258  2                     Dump_Line_10  = $ASCIC ('!/SHOW SELECTED:!/');                                               !    [07]
```

```
;    1260  2      !+
;    1261  2      ! Picture of a call frame:
;    1262  2      !
;    1263  2      !       ------------------------------------------------------------------------------
;    1264  2      !       |                        PUSHx arguments for the CALLx                       | High Memory
;    1265  2      !       ------------------------------------------------------------------------------
;    1266  2      !       |                                                | Number of arguments       | :(AP)
;    1267  2      !       ------------------------------------------------------------------------------
;    1268  2      !       |                              Saved R11                                     | +n+4
;    1269  2      !       ------------------------------------------------------------------------------
;    1270  2      !       |                              Saved R10                                     | +n
;    1271  2      !       ------------------------------------------------------------------------------
;    1272  2      !                                       .
;    1273  2      !                                       .
;    1274  2      !                                       .
;    1275  2      !       ------------------------------------------------------------------------------
;    1276  2      !       |                               Saved R0                                     | +20
;    1277  2      !       ------------------------------------------------------------------------------
;    1278  2      !       |                               Saved PC                                     | +16
;    1279  2      !       ------------------------------------------------------------------------------
;    1280  2      !       |                               Saved FP                                     | +12
;    1281  2      !       ------------------------------------------------------------------------------
;    1282  2      !       |                               Saved AP                                     | +8
;    1283  2      !       ------------------------------------------------------------------------------
;    1284  2      !       | SPA | S | 0 |Register save mask <11:0>|          PSW <15:5>          |  0  | +4
;    1285  2      !       ------------------------------------------------------------------------------
;    1286  2      !       |                        Condition Handler (Initially Zero)                  | :(FP) - Low Memory
;    1287  2      !       ------------------------------------------------------------------------------
;    1288  2      !       |                     Start of called routine's stack area                   |
;    1289  2      !       ------------------------------------------------------------------------------
;    1290  2      !-
```

```
;   1292  2          !+
;   1293  2          ! Print the header lines for this routine.
;   1294  2          !-
;   1295  2
;   1296  2          $DS_PRINTF (QA$T_Header_1);                    !                                                [01]
;   1297  2          $DS_PRINTF (QA$T_Header_2, .L$A_NAME, .L$L_REV, .L$L_UPDATE);   !                                [01]
;   1298  2          $DS_PRINTF (QA$T_Header_3, 0);                 ! Specify 0 to get system time.                  [01]
;   1299  2          $DS_PRINTF (Dump_Header_4);
;   1300  2
;   1301  2          !+
;   1302  2          ! Print the test number, subtest number, pass number, and section name.
;   1303  2          !-
;   1304  2
;   1305  3          BEGIN
;   1306  3                  BIND
;   1307  3                          Sec_Name_Table = ((.L$A_SECNAM) + 4) : VECTOR [, LONG];
;   1308  3
; P 1309  3                  $DS_PRINTF (Dump_Line_1, .DSA$GL_TESTNO,
; P 1310  3                              .DSA$GL_SUBTNO, .DSA$GL_PASSNO,
;   1311  4                              .Sec_Name_Table [.DSA$GL_SECTNO])
;   1312  2          END;
;   1313  2
;   1314  2          !+
;   1315  2          ! If an Init_Context has been done, then cannot follow stack frames back to the user's last
;   1316  2          ! call frame. Therefore, just print above lines and return.
;   1317  2          !-
;   1318  2
;   1319  2          IF (.QA$AOB_Flags [QA$K_Init_Context_Done]) THEN
;   1320  3                  BEGIN
;   1321  3                  QA$AOB_Flags [QA$K_Init_Context_Done] = False;  ! Turn it back off
;   1322  3                  RETURN
;   1323  2                  END;
```

```
;    1325  2              !+
;    1326  2              ! If cannot follow stack frames back to the user's program, the Find_User_Call_Frame routine will
;    1327  2              ! print an error message.
;    1328  2              !-
;    1329  2
;    1330  2              IF (NOT QA$Find_User_Call_Frame (Frame_Start)) THEN
;    1331  3                      BEGIN
;    1332  3                      RETURN                                  ! Failure
;    1333  2                      END;
;    1334  2
;    1335  2              !+
;    1336  2              ! Frame_Start now equals the start of the user's last call frame. Set Diag_FP equal to the start of
;    1337  2              ! the frame that was just found. This frame is the one from which I am extracting everything, so it
;    1338  2              ! makes sense to use the FP that points to this frame. This is done instead of using the FP that is
;    1339  2              ! contained in the frame we are looking at (i.e., that points to the frame of the caller of the
;    1340  2              ! user's routine that bombed).
;    1341  2              !-
;    1342  2
;    1343  3              BEGIN
;    1344  3                      BIND
;    1345  3                              Call_Frame = .Frame_Start : BLOCK [, BYTE];
;    1346  3
;    1347  3                      Diag_PSW        = .Call_Frame [SF$W_SAVE_PSW];
;    1348  3                      Register_Mask   = .Call_Frame [SF$V_SAVE_MASK];
;    1349  3                      Diag_AP         = .Call_Frame [SF$L_SAVE_AP];
;    1350  3                      Diag_FP         = .Frame_Start;
;    1351  3                      Diag_PC         = .Call_Frame [SF$L_SAVE_PC];
;    1352  2              END;
```

```
;    1354  2              !+
;    1355  2              ! Get those registers that were saved. Assume Reg_Not_Saved for non-saved ones. This will only get
;    1356  2              ! those registers that were saved in the user's last call frame. That is, those registers saved by
;    1357  2              ! the last called Supervisor routine, which through a series of CALL's, BSBW's, etc. got to here.
;    1358  2              ! However, those Supervisor routines which, by virtue of their being called by the diagnostic,        [02]
;    1359  2              ! cause a QA error must have all the registers (i.e., R2 through R11) saved in the entry mask for     [02]
;    1360  2              ! their routine. This allows the following code to access all these registers.                       [02]
;    1361  2              !-
;    1362  2
;    1363  2              INCR Reg_Number FROM 0 TO (Max_Regs_Saved - 1) DO
;    1364  2                      Reg_Vector [.Reg_Number] = Reg_Not_Saved;
;    1365  2
;    1366  2              !+
;    1367  2              ! Number_Of_Regs will actually end up to be one less than the actual number of registers that were
;    1368  2              ! saved. This is so that I can use it to index the Start_Registers vector, which starts at zero, not one!
;    1369  2              !-
;    1370  2
;    1371  2              Number_Of_Regs = -1;
;    1372  2
;    1373  2              INCR Bit_Number FROM 0 TO (Max_Regs_Saved - 1) DO
;    1374  3                      BEGIN
;    1375  3                      BIND
;    1376  3                              !+
;    1377  3                              ! Start_registers is the start of a vector of saved registers in the call frame. This
;    1378  3                              ! vector starts at offset 20 from the start of the frame.
;    1379  3                              !-
;    1380  3
;    1381  3                              Start_Registers = ((.Frame_Start) + 20) : VECTOR [, LONG];
;    1382  3
;    1383  3                      MAP
;    1384  3                              Register_Mask   : BITVECTOR [32];       ! Reference as a Bit Vector
;    1385  3
;    1386  3                      IF (.Register_Mask [.Bit_Number]) THEN
;    1387  4                              BEGIN
;    1388  4
;    1389  4                              Number_Of_Regs = .Number_Of_Regs + 1;
;    1390  4                              Reg_Vector [.Bit_Number] = .Start_Registers [.Number_Of_Regs];
;    1391  3                              END;
;    1392  2                      END;
```

```
;    1394  2              Number_Of_Regs = .Number_Of_Regs + 1;
;    1395  2
;    1396  2              !+
;    1397  2              ! Number_Of_Regs now equals the actual number of registers that were saved in the call frame.
;    1398  2              ! Compute the value of the Stack Pointer before the user's CALLx was executed. This is found
;    1399  2              ! by computing the size of the last call frame put on by the user. This size is the added to
;    1400  2              ! the starting address of the call frame and the stack pointer alignment is added in,
;    1401  2              ! producing the value of the SP.
;    1402  2              !-
;    1403  2
;    1404  3              BEGIN
;    1405  3                      BIND
;    1406  3                              Call_Frame = .Frame_Start : BLOCK [, BYTE];
;    1407  3
;    1408  3                      Diag_SP = .Frame_Start +                        ! Start of call frame
;    1409  3                                (5 * 4) +                             ! 5 longwords
;    1410  3                                (.Number_Of_Regs * 4) +               ! Longword for each reg
;    1411  3                                .Call_Frame [SF$V_STACKOFFS];         ! SPA
;    1412  2              END;
;    1413  2
;    1414  2              !+
;    1415  2              ! Print the registers and the PSW. Do not print R0 and R1 since they may not be correct.          [02]
;    1416  2              !-
;    1417  2
;    1418  2              $DS_PRINTF (Dump_line_2, .Reg_Vector [2],  .Reg_Vector [3],  .Reg_Vector [4],  .Reg_Vector [5]);   !    [02]
;    1419  2              $DS_PRINTF (Dump_line_3, .Reg_Vector [6],  .Reg_Vector [7],  .Reg_Vector [8],  .Reg_Vector [9]);   !    [02]
;    1420  2              $DS_PRINTF (Dump_line_4, .Reg_Vector [10], .Reg_Vector [11], .Diag_AP,         .Diag_FP);          !    [02]
;    1421  2              $DS_PRINTF (Dump_line_5, .Diag_SP,         .Diag_PC,         .Diag_PSW);                        !    [02]
```

```
;    1423  2              !+
;    1424  2              ! Print the control flags, event flags, and selected devices.
;    1425  2              !-
;    1426  2
;    1427  2              $DS_PRINTF (Dump_line_7);
;    1428  2              VRShowFlg ();                                   ! Show the control flags
;    1429  2
;    1430  2              $DS_PRINTF (Dump_line_8);
;    1431  2              VRShowEf  ();                                   ! Show the event flags
;    1432  2
;    1433  2              !+
;    1434  2              ! Set up a fake CLI database table. This is needed because the following two routines access the
;    1435  2              ! CLI$Q_FILE quadword in this table in order to determine what device should be SHOW'n. All that
;    1436  2              ! is needed in this table is a null quadword at the CLI$Q_FILE offset. Zero out the entire table
;    1437  2              ! to play it safe though. Only 4 longwords are allocated since the FILE quadword starts at the
;    1438  2              ! third longword in the real CLI table.
;    1439  2              !-
;    1440  2
;    1441  2              INCR I FROM 0 TO 3 DO
;    1442  2                      Fake_CLI_Table [.I] = 0;
;    1443  2
;    1444  2              $DS_PRINTF (Dump_line_9);
;    1445  2              DSV$ShowDevice (Fake_CLI_Table);                ! Show the devices
;    1446  2
;    1447  2              $DS_PRINTF (Dump_line_10);
;    1448  2              DSV$ShowSelect (Fake_CLI_Table)                 ! Show the selected devices
;    1449  1      END;


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

20 52 4F 52 52 45 20 41 51 20 2A 35 32 21 1A  00007 P.AAB:  .ASCII  <26>\!25* QA ERROR DETECTED!/!/\                ;
   2F 21 2F 21 44 45 54 43 45 54 45 44         00016                                                            ;
20 2A 39 21 4C 55 33 21 20 3A 74 73 65 54 3D  00022 P.AAC:  .ASCII  \=Test: !3UL!9* Subtest: !3UL!6* Pass: !3\   ;
36 21 4C 55 33 21 20 3A 74 73 65 74 62 75 53  00031                                                            ;
            33 21 20 3A 73 73 61 50 20 2A     00040                                                            ;
20 3A 6E 6F 69 74 63 65 53 20 2A 39 21 4C 55  0004A         .ASCII  \UL!9* Section: !AC!/!/\                    ;
               2F 21 2F 21 43 41 21           00059                                                            ;
32 21 29 58 28 4C 58 38 21 20 20 3A 32 52 3E  00060 P.AAD:  .ASCII  \>R2:  !8XL(X)!2* R3:  !8XL(X)!2* R4:  !8\   ;
21 29 58 28 4C 58 38 21 20 20 3A 33 52 20 2A  0006F                                                            ;
            38 21 20 20 3A 34 52 20 2A 32     0007E                                                            ;
21 20 20 3A 35 52 20 2A 32 21 29 58 28 4C 58  00088         .ASCII  \XL(X)!2* R5:  !8XL(X)!/\                   ;
               2F 21 29 58 28 4C 58 38        00097                                                            ;
32 21 29 58 28 4C 58 38 21 20 20 3A 36 52 3E  0009F P.AAE:  .ASCII  \>R6:  !8XL(X)!2* R7:  !8XL(X)!2* R8:  !8\   ;
21 29 58 28 4C 58 38 21 20 20 3A 37 52 20 2A  000AE                                                            ;
            38 21 20 20 3A 38 52 20 2A 32     000BD                                                            ;
21 20 20 3A 39 52 20 2A 32 21 29 58 28 4C 58  000C7         .ASCII  \XL(X)!2* R9:  !8XL(X)!/\                   ;
               2F 21 29 58 28 4C 58 38        000D6                                                            ;
32 21 29 58 28 4C 58 38 21 20 3A 30 31 52 3E  000DE P.AAF:  .ASCII  \>R10: !8XL(X)!2* R11: !8XL(X)!2* AP:  !8\   ;
21 29 58 28 4C 58 38 21 20 3A 31 31 52 20 2A  000ED                                                            ;
            38 21 20 20 3A 50 41 20 2A 32     000FC                                                            ;
21 20 20 3A 50 46 20 2A 32 21 29 58 28 4C 58  00106         .ASCII  \XL(X)!2* FP:  !8XL(X)!/\                   ;
               2F 21 29 58 28 4C 58 38        00115                                                            ;
32 21 29 58 28 4C 58 38 21 20 20 3A 50 53 2E  0011D P.AAG:  .ASCII  \.SP:  !8XL(X)!2* PC:  !8XL(X)!2* PSW: !8\   ;
21 29 58 28 4C 58 38 21 20 20 3A 43 50 20 2A  0012C                                                            ;
            38 21 20 3A 57 53 50 20 2A 32     0013B                                                            ;
```

```
                                                
```

```
                54        06    A1              0C      00  EF 00090          EXTZV   #0, #12, 6(R1), REGISTER_MASK                   ; 1348
                                                57  08  A1  D0 00096          MOVL    8(R1), DIAG_AP                                  ; 1349
                                                56      51  D0 0009A          MOVL    R1, DIAG_FP                                     ; 1350
                                                55  10  A1  D0 0009D          MOVL    16(R1), DIAG_PC                                 ; 1351
                                                    50      D4 000A1          CLRL    REG_NUMBER                                      ; 1363
                                                    04 AE40 D4 000A3 3$:      CLRL    REG_VECTOR[REG_NUMBER]                          ; 1364
                        F8                      50      0B  F3 000A7          AOBLEQ  #11, REG_NUMBER, 3$                             ;
                                                50      01  CE 000AB          MNEGL   #1, NUMBER_OF_REGS                              ; 1371
                                                53  14  A1  9E 000AE          MOVAB   20(R1), R3                                      ; 1381
                                                    52      D4 000B2          CLRL    BIT_NUMBER                                      ;
                        08                      54      52  E1 000B4 4$:      BBC     BIT_NUMBER, REGISTER_MASK, 5$                   ; 1386
                                                    50      D6 000B8          INCL    NUMBER_OF_REGS                                  ; 1389
                                            04 AE42         6340  D0 000BA    MOVL    (R3)[NUMBER_OF_REGS], REG_VECTOR-               ; 1390
                                                                                      [BIT_NUMBER]                                    ;
                        F0                      52      0B  F3 000C0 5$:      AOBLEQ  #11, BIT_NUMBER, 4$                             ;
                                                    50      D6 000C4          INCL    NUMBER_OF_REGS                                  ; 1394
                                                50          6140  DE 000C6    MOVAL   (R1)[NUMBER_OF_REGS], R0                        ; 1409
                51        07    A1              02      06  EF 000CA          EXTZV   #6, #2, 7(R1), R1                               ; 1411
                                                52  14  A140  9E 000D0        MOVAB   20(R1)[R0], DIAG_SP                             ; 1410
                                                    18  AE  DD 000D5          PUSHL   REG_VECTOR+20                                   ; 1418
                                                    18  AE  DD 000D8          PUSHL   REG_VECTOR+16                                   ;
                                                    18  AE  DD 000DB          PUSHL   REG_VECTOR+12                                   ;
                                                    18  AE  DD 000DE          PUSHL   REG_VECTOR+8                                    ;
                                                    59  AA  9F 000E1          PUSHAB  DUMP_LINE_2                                     ;
                                                69          05  FB 000E4      CALLS   #5, DS$PRINTF                                   ;
                                                    28  AE  DD 000E7          PUSHL   REG_VECTOR+36                                   ; 1419
                                                    28  AE  DD 000EA          PUSHL   REG_VECTOR+32                                   ;
                                                    28  AE  DD 000ED          PUSHL   REG_VECTOR+28                                   ;
                                                    28  AE  DD 000F0          PUSHL   REG_VECTOR+24                                   ;
                                                  0098  CA  9F 000F3          PUSHAB  DUMP_LINE_3                                     ;
                                                69          05  FB 000F7      CALLS   #5, DS$PRINTF                                   ;
                                                    56      DD 000FA          PUSHL   DIAG_FP                                         ; 1420
                                                    57      DD 000FC          PUSHL   DIAG_AP                                         ;
                                                    38  AE  DD 000FE          PUSHL   REG_VECTOR+44                                   ;
                                                    38  AE  DD 00101          PUSHL   REG_VECTOR+40                                   ;
                                                  00D7  CA  9F 00104          PUSHAB  DUMP_LINE_4                                     ;
                                                69          05  FB 00108      CALLS   #5, DS$PRINTF                                   ;
                                                7E          58  3C 0010B      MOVZWL  DIAG_PSW, -(SP)                                 ; 1421
                                                    24      BB 0010E          PUSHR   #^M<R2,R5>                                      ;
                                                  0116  CA  9F 00110          PUSHAB  DUMP_LINE_5                                     ;
                                                69          04  FB 00114      CALLS   #4, DS$PRINTF                                   ;
                                                  0145  CA  9F 00117          PUSHAB  DUMP_LINE_7                                     ; 1427
                                                69          01  FB 0011B      CALLS   #1, DS$PRINTF                                   ;
                                             00000000G      EF  16 0011E      JSB     VRSHOWFLG                                       ; 1428
                                                  0155  CA  9F 00124          PUSHAB  DUMP_LINE_8                                     ; 1430
                                                69          01  FB 00128      CALLS   #1, DS$PRINTF                                   ;
                                             00000000G      EF  16 0012B      JSB     VRSHOWEF                                        ; 1431
                                                    50      D4 00131          CLRL    I                                               ; 1441
                                                    34 AE40 D4 00133 6$:      CLRL    FAKE_CLI_TABLE[I]                               ; 1442
                F8                              50      03  F3 00137          AOBLEQ  #3, I, 6$                                       ;
                                                  016B  CA  9F 0013B          PUSHAB  DUMP_LINE_9                                     ; 1444
                                                69          01  FB 0013F      CALLS   #1, DS$PRINTF                                   ;
                                                52      34  AE  9E 00142      MOVAB   FAKE_CLI_TABLE, R2                              ; 1445
                                             00000000G      EF  16 00146      JSB     DSV$SHOWDEVICE                                  ;
                                                  017C  CA  9F 0014C          PUSHAB  DUMP_LINE_10                                    ; 1447
                                                69          01  FB 00150      CALLS   #1, DS$PRINTF                                   ;
                                                52      34  AE  9E 00153      MOVAB   FAKE_CLI_TABLE, R2                              ; 1448
```

```
                                00000000G  EF  16 00157        JSB     DSV$SHOWSELECT                          ;
                                           04 0015D         RET                                             ; 1449

; Routine Size:  350 bytes,    Routine Base:  CODE + 01C1
```

```
;    1451  1      %SBTTL  'QA$Find_User_Call_Frame'
;    1452  1
;    1453  1      GLOBAL ROUTINE QA$Find_User_Call_Frame (Frame_Start) =
;    1454  1
;    1455  1      !++
;    1456  1      ! FUNCTIONAL DESCRIPTION:
;    1457  1      !
;    1458  1      !       Find the user's last call frame by following the linked list
;    1459  1      !       of call frames. Find the first call frame whose saved PC is
;    1460  1      !       less than the starting location of the Supervisor. This should
;    1461  1      !       insure that that call frame was put on the stack when the user
;    1462  1      !       called a Supervisor service routine. This called frame will
;    1463  1      !       contain the saved PC, AP, FP, PSW, perhaps some of the general
;    1464  1      !       registers, and (indirectly) the SP.
;    1465  1      !
;    1466  1      ! CALLER:
;    1467  1      !
;    1468  1      !       QA$Dump - to find the call frame so it can print the debugging
;    1469  1      !                 information.
;    1470  1      !
;    1471  1      ! FORMAL PARAMETERS:
;    1472  1      !
;    1473  1      !       Frame_Start - will contain the starting location of the
;    1474  1      !                     found call frame.
;    1475  1      !
;    1476  1      ! IMPLICIT INPUTS:
;    1477  1      !
;    1478  1      !       NONE
;    1479  1      !
;    1480  1      ! IMPLICIT OUTPUTS:
;    1481  1      !
;    1482  1      !       Messages indicating that the call frame list cannot be followed.
;    1483  1      !
;    1484  1      ! COMPLETION CODES:
;    1485  1      !
;    1486  1      !       R0 EQL 1 => Success, Frame_Start contains the starting location
;    1487  1      !                                of the call frame.
;    1488  1      !       R0 EQL 1 => Failure, Frame_Start contains crap.
;    1489  1      !
;    1490  1      ! SIDE EFFECTS:
;    1491  1      !
;    1492  1      !       Hopefully none.
;    1493  1      !
;    1494  1      !--
```

```
;   1496  2     BEGIN
;   1497  2             BIND
;   1498  2                     FP_Ptr = .Frame_Start;
;   1499  2
;   1500  2             BUILTIN
;   1501  2                     FP, PROBEW;
;   1502  2
;   1503  2             LITERAL
;   1504  2                     Max_Frames_Checked = 25;                ! Maximum number of frames to check
;   1505  2
;   1506  2             LOCAL
;   1507  2                     Last_PC,
;   1508  2                     Number_Frames_Checked;
```

```
;    1510  2              !+
;    1511  2              ! Find the last call frame put on the stack by the user. Verify that the FP in the call frame
;    1512  2              ! can be accessed. This partially insures that I am not following invalid call frames.
;    1513  2              !-
;    1514  2
;    1515  2              FP_Ptr = .FP;                                   ! Start at the current FP
;    1516  2              Number_Frames_Checked = 0;
;    1517  2
;    1518  2              WHILE 1 DO
;    1519  3                      BEGIN
;    1520  3
;    1521  3                      IF (NOT PROBEW (%REF (0), %REF (3), .FP_Ptr)) THEN
;    1522  4                              BEGIN
;    1523  4                              BIND
; P  1524  4                                      Invalid_Call_Frame = $ASCIC
;    1525  4                                              ('!/**QA Internal Error: Invalid call frame. FP = !XL(X).!/!/'); !      [01]
;    1526  4
;    1527  4                              $DS_PRINTF (Invalid_Call_Frame, .FP_Ptr);
;    1528  5                              RETURN (QA$K_Failure)
;    1529  3                              END;
;    1530  3
;    1531  3                      IF ((Number_Frames_Checked = .Number_Frames_Checked + 1) GTR Max_Frames_Checked) THEN
;    1532  4                              BEGIN
;    1533  4                              BIND
; P  1534  4                                      Too_Many_Frames = $ASCIC
;    1535  4                              ('!/**QA Internal Error: Cannot follow stack frames back to the diagnostic!/!/'); !      [01]
;    1536  4
;    1537  4                              $DS_PRINTF (Too_Many_Frames);
;    1538  5                              RETURN (QA$K_Failure)
;    1539  3                              END;
;    1540  3
;    1541  3                      !+
;    1542  3                      ! The contents of FP_Ptr is the starting address of the current frame being looked at. Pick out the
;    1543  3                      ! PC from the call frame. If this PC is less than the starting location of the Supervisor, then this
;    1544  3                      ! call frame must be the last call frame of the user. If not, backtrack to the next call frame.
;    1545  3                      !-
;    1546  3
;    1547  3                      Last_PC = .(.FP_Ptr + 16);                      ! PC is offset 16 from the start of the call frame.
;    1548  3
;    1549  3                      IF (.Last_PC LSS DS$K_DS_Starting_Location) THEN
;    1550  4                              BEGIN
;    1551  5                              RETURN (QA$K_Success)
;    1552  3                              END;
;    1553  3
;    1554  3                      FP_Ptr = .(.FP_Ptr + 12);                       ! FP is offset 12 from the start of the call frame.
;    1555  3                      END
;    1556  1      END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61  6E  72  65  74  6E  49  20  41  51  2A  2A  2F  21  3B  00196 P.AAL:  .ASCII  \;!/**QA Internal Error: Invalid call fra\  ;
69  6C  61  76  6E  49  20  3A  72  6F  72  72  45  20  6C  001A5                                                              ;
                    61  72  66  20  6C  6C  61  63  20  64  001B4                                                              ;
29  58  28  4C  58  21  20  3D  20  50  46  20  2E  65  6D  001BE          .ASCII  \me. FP = !XL(X).!/!/\                      ;
                                    2F  21  2F  21  2E  001CD                                                              ;
```

```
61  6E  72  65  74  6E  49  20  41  51  2A  2A  2F  21  4C  001D2 P.AAM:  .ASCII  \L!/**QA Internal Error: Cannot follow st\  ;
74  6F  6E  6E  61  43  20  3A  72  6F  72  72  45  20  6C  001E1                                                            ;
                    74  73  20  77  6F  6C  6C  6F  66  20  001F0                                                            ;
6B  63  61  62  20  73  65  6D  61  72  66  20  6B  63  61  001FA         .ASCII  \ack frames back to the diagnostic!/!/\     ;
73  6F  6E  67  61  69  64  20  65  68  74  20  6F  74  20  00209                                                            ;
                                2F  21  2F  21  63  69  74  00218                                                            ;

                                                                    INVALID_CALL_FRAME= P.AAL
                                                                    TOO_MANY_FRAMES=    P.AAM


                                                                         .PSECT  CODE,NOWRT,  SHR,2

                                                  003C 00000             .ENTRY  QA$FIND_USER_CALL_FRAME, Save R2,R3,R4,R5   ; 1453
                                55 00000000G  9F  9E 00002             MOVAB   @#DS$PRINTF, R5                             ;
                          04    BC            5D  D0 00009             MOVL    FP, @FRAME_START                            ; 1515
                                              53  D4 0000D             CLRL    NUMBER_FRAMES_CHECKED                       ; 1516
                                52       04   BC  D0 0000F 1$:         MOVL    @FRAME_START, R2                            ; 1521
                    62          03            00  0D 00013             PROBEW  #0, #3, (R2)                                ;
                                              0D  12 00017             BNEQ    2$                                          ;
                                              52  DD 00019             PUSHL   R2                                          ; 1527
                                   00000000'  EF  9F 0001B             PUSHAB  INVALID_CALL_FRAME                          ;
                                65            02  FB 00021             CALLS   #2, DS$PRINTF                               ;
                                              10  11 00024             BRB     3$                                          ; 1528
                                              53  D6 00026 2$:         INCL    NUMBER_FRAMES_CHECKED                       ; 1531
                                19            53  D1 00028             CMPL    NUMBER_FRAMES_CHECKED, #25                  ;
                                              0C  15 0002B             BLEQ    4$                                          ;
                                   00000000'  EF  9F 0002D             PUSHAB  TOO_MANY_FRAMES                             ; 1537
                                65            01  FB 00033             CALLS   #1, DS$PRINTF                               ;
                                              50  D4 00036 3$:         CLRL    R0                                          ; 1538
                                                  04 00038             RET                                                 ;
                                54       10   A2  D0 00039 4$:         MOVL    16(R2), LAST_PC                             ; 1547
                       00010000 8F            54  D1 0003D             CMPL    LAST_PC, #65536                             ; 1549
                                              04  18 00044             BGEQ    5$                                          ;
                                50            01  D0 00046             MOVL    #1, R0                                      ; 1551
                                                  04 00049             RET                                                 ;
                          04    BC       0C   A2  D0 0004A 5$:         MOVL    12(R2), @FRAME_START                        ; 1554
                                              BE  11 0004F             BRB     1$                                          ;

; Routine Size:  81 bytes,    Routine Base:  CODE + 031F
```

```
;    1558  1      %SBTTL 'QA$Init'
;    1559  1
;    1560  1      ROUTINE QA$Init : NOVALUE =
;    1561  1
;    1562  1      !++
;    1563  1      ! FUNCTIONAL DESCRIPTION:
;    1564  1      !
;    1565  1      !       This routine performs all the global initializations required
;    1566  1      !       for the QA routines. This routine must be called both when a
;    1567  1      !       START or RUN command is being processed and when a QA execution
;    1568  1      !       completes (to set up a subsequent execution).
;    1569  1      !
;    1570  1      ! CALLER(S):
;    1571  1      !
;    1572  1      !       QA$Main
;    1573  1      !
;    1574  1      ! FORMAL PARAMETERS:
;    1575  1      !
;    1576  1      !       NONE
;    1577  1      !
;    1578  1      ! IMPLICIT INPUTS:
;    1579  1      !
;    1580  1      !       The QA$AOB_Check_State bit vector.
;    1581  1      !       The QA$AW_Overall_Error_Table word vector.
;    1582  1      !       The QA$AOB_Flags bit vector.
;    1583  1      !
;    1584  1      ! IMPLICIT OUTPUTS:
;    1585  1      !
;    1586  1      !       The QA$AOB_Check_State bit vector.
;    1587  1      !       The QA$AW_Overall_Error_Table word vector.
;    1588  1      !       The QA$AOB_Flags bit vector.
;    1589  1      !
;    1590  1      ! COMPLETION CODES:
;    1591  1      !
;    1592  1      !       NONE
;    1593  1      !
;    1594  1      ! SIDE EFFECTS:
;    1595  1      !
;    1596  1      !       The QA$AOB_Check_State bit vector is zeroed.
;    1597  1      !       The QA$AW_Overall_Error_Table word vector is zeroed.
;    1598  1      !       The QA$AOB_Flags bit vector is zeroed.
;    1599  1      !--
;    1600  1
;    1601  2      BEGIN
;    1602  2              INCR Flag FROM 0 TO (QA$K_Number_QA_Flags - 1) DO
;    1603  2                      QA$AOB_Flags [.Flag] = Off;
;    1604  2
;    1605  2              INCR Bit_Pos FROM 0 TO (QA$K_Number_Of_Checks - 1) DO
;    1606  2                      QA$AOB_Check_State [.Bit_Pos] = Off;
;    1607  2
;    1608  2              INCR Error_Entry FROM 0 TO (QA$K_Number_Of_Checks - 1) DO
;    1609  2                      QA$AW_Overall_Error_Table [.Error_Entry] = 0;
;    1610  2
;    1611  2      !       QA$A_Prev_Test_Tail  = QA$K_Nil;
;    1612  2      !       QA$A_Error_List_Head = QA$K_Nil;
;    1613  2              DS$GA_DS_Ctrl_C_First = 0;       ! Disable Supervisor catching of Control-C's.          [06]
;    1614  1      END;
```

```
                                  0004 00000 QA$INIT:.WORD   Save R2                                  ; 1560
                  52 00000000'  EF  9E 00002         MOVAB   QA$AOB_FLAGS, R2                        ;
                                50  D4 00009         CLRL    FLAG                                    ; 1602
        00        62            50  E5 0000B 1$:     BBCC    FLAG, QA$AOB_FLAGS, 2$                  ; 1603
        F8        50            07  F3 0000F 2$:     AOBLEQ  #7, FLAG, 1$                            ;
                                50  D4 00013         CLRL    BIT_POS                                 ; 1605
        00     FC A2            50  E5 00015 3$:     BBCC    BIT_POS, QA$AOB_CHECK_STATE, 4$         ; 1606
        F7        50            0E  F3 0001A 4$:     AOBLEQ  #14, BIT_POS, 3$                        ;
                                50  D4 0001E         CLRL    ERROR_ENTRY                             ; 1608
                          CC A240  B4 00020 5$:     CLRW    QA$AW_OVERALL_ERROR_TABLE[ERROR_ENTRY]  ; 1609
        F8        50            0E  F3 00024         AOBLEQ  #14, ERROR_ENTRY, 5$                    ;
                     00000000G  EF  D4 00028         CLRL    DS$GA_DS_CTRL_C_FIRST                   ; 1613
                                    04 0002E         RET                                             ; 1614

; Routine Size:  47 bytes,    Routine Base:  CODE + 0370
```

```
;    1616  1      %SBTTL 'QA$Main'
;    1617  1
;    1618  1      GLOBAL ROUTINE QA$Main (Hook_Point) : NOVALUE =
;    1619  1
;    1620  1      !++
;    1621  1      ! FUNCTIONAL DESCRIPTION:
;    1622  1      !
;    1623  1      !       This global routine is called from different points in the Supervisor and from the DSX$Branch
;    1624  1      !       routine. It decides, based on the QA$AOB_Check_State bit vector, which check is currently being
;    1625  1      !       executed. If a check routine returns with an unsuccessful return, this will call the QA$Dump
;    1626  1      !       routine and then halt. If the diagnostic was not run with the /QA qualifier, do nothing.
;    1627  1      !
;    1628  1      ! CALLERS:
;    1629  1      !
;    1630  1      !       DSX$Branch
;    1631  1      !       CLI
;    1632  1      !       DISPAT
;    1633  1      !       ERROR
;    1634  1      !       KERNEL
;    1635  1      !       LOOP
;    1636  1      !       PARAM
;    1637  1      !       START
;    1638  1      !
;    1639  1      ! FORMAL PARAMETERS:
;    1640  1      !
;    1641  1      !       Hook_Point : The point in the Supervisor from which this routine was called. This is a constant
;    1642  1      !       value. The constants are defined in the DSQA macro.
;    1643  1      !
;    1644  1      ! IMPLICIT INPUTS:
;    1645  1      !
;    1646  1      !       NONE
;    1647  1      !
;    1648  1      ! IMPLICIT OUTPUTS:
;    1649  1      !
;    1650  1      !       NONE
;    1651  1      !
;    1652  1      ! COMPLETION CODES:
;    1653  1      !
;    1654  1      !       NONE
;    1655  1      !
;    1656  1      ! SIDE EFFECTS:
;    1657  1      !
;    1658  1      !       NONE
;    1659  1      !
;    1660  1      !--
```

```
;    1662  2      BEGIN
;    1663  2              MACRO
;    1664  2                      !+
;    1665  2                      ! Define a macro
;    1666  2                      !
;    1667  2                      !       Check_Case (Normal_Start)
;    1668  2                      !
;    1669  2                      ! that will expand to:
;    1670  2                      !
;    1671  2                      !       [.QA$AOB_Check_State [QA$K_Normal_Start]:
;    1672  2                      !               IF (QA$Normal_Start (.Hook_Point)) THEN
;    1673  2                      !                       RETURN
;    1674  2                      !
;    1675  2                      ! for the case of the Normal Start check. That is, if the appropriate bit is set in the bit
;    1676  2                      ! vector, call the appropriate routine (corresponding to that bit) with the Hook_Point parameter
;    1677  2                      ! that was passed to QA$Main. If the routine returns successfully, return to the caller.
;    1678  2                      !-
;    1679  2
; M  1680  2                      Check_Case (Routine_Name) =
; M  1681  2                              [ .QA$AOB_Check_State
; M  1682  2                                      [ %NAME ('QA$K_', Routine_Name) ]
; M  1683  2                              ]:
; M  1684  2                                      IF (%NAME ('QA$', Routine_Name) (.Hook_Point))
; M  1685  2                                      THEN
; M  1686  2                                              RETURN
;    1687  2                      %;
;    1688  2
;    1689  2              BUILTIN
;    1690  2                      SP;
;    1691  2
;    1692  2              MAP                                             !                       [02]
;    1693  2                      DS$GL_CLIBASE : BLOCK [, BYTE];         !                       [02]
```

```
;   1695  2              IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN
; P 1696  2                      $DS_PRINTF ($ASCIC ('!/QA$Main Entry, Hook_Point = !UL.!/'),
;   1697  2                              .Hook_Point);
;   1698  2
;   1699  2              !+
;   1700  2              ! If not running QA, return.
;   1701  2              !-
;   1702  2
;   1703  2              IF (NOT .DSA$V_QA) THEN
;   1704  2                      RETURN;
;   1705  2
;   1706  2              !+
;   1707  2              ! Check several special cases of hook points:
;   1708  2              !
;   1709  2              !       a) If called from the START module, VRStart label, perform all the global initializations       [02]
;   1710  2              !          for this module. Do not call any check routines.                                            [02]
;   1711  2              !
;   1712  2              !       b) If called from the START module, VRReStart label, re-initialize the CLI vector and call     [02]
;   1713  2              !          the appropriate check routine. This hook-point is the starting point for each check.        [02]
;   1714  2              !
;   1715  2              !       c) If called from the START module, Before_Header area, and this is the first time that this    [02]
;   1716  2              !          hook point has been seen, then save the first and last test numbers after they have been    [02]
;   1717  2              !          set up by the START routine. Do not call any check routines.                                [02]
;   1718  2              !
;   1719  2              !       d) If called from one of the GET routines in the PARAM module, print a QA error message,     [02]
;   1720  2              !          find out what check routine is currently running, add an error to that routines total,     [02]
;   1721  2              !          and abort this diagnostic's QA. This is done here because each check routine has to make    [02]
;   1722  2              !          this check.                                                                                 [02]
;   1723  2              !
;   1724  2              !       e) If called from the midpoint of the EndPass routine in the DISPAT module, and the last      [02]
;   1725  2              !          check was executing, do things necessary to end the QA sequence of checks. That is,         [02]
;   1726  2              !          print the summary table, re-init for next time, turn the QA DSA bit off, and return to      [02]
;   1727  2              !          the EndPass routine. The act of turning the QA bit off will cause the EndPass routine       [02]
;   1728  2              !          to then branch to BEGIN, which then calls DS$Cli (effectively ending this QA sequence).     [02]
;   1729  2              !
;   1730  2              !       f) If called from the KERNEL module, Init_Context label, and if the QA debug flag is set,     [02]
;   1731  2              !          print a debugging message for me.                                                           [02]
;   1732  2              !
;   1733  2              !       g) If called either the beginning or the end of the DS_Cleanup routine in the DISPAT module,  [02]
;   1734  2              !          set (or clear) the Doing_Cleanup bit in the Routine_State bitvector. This tells the check   [02]
;   1735  2              !          routines if a reported error happened during Cleanup or not.                               [02]
;   1736  2              !-
```

```
;  2          IF (.Hook_Point EQL DSQA$K_Start_VRStart) THEN          !                                          [02]
;  3                  BEGIN
;  3                  !+
;  3                  ! Initialize QA. It should get to this hook point only when the START routine is called from   [02]
;  3                  ! the DS$Cli routine. Set the Normal Start check bit so that the Normal Start routine is       [02]
;  3                  ! executed first. Return to the Start module (i.e., do not execute any check routines).        [02]
;  3                  ! NOTE: this hook point is seen only ONCE per QA sequence of checks (that is, once for each    [02]
;  3                  ! invocation of the diagnostic with the /QA qualifier).                                       [02]
;  3                  !-
;  3
;  3                  DS$GL_CLIBASE [CLI$L_PASS] = 1;          ! Assume one for the number of passes.                [02]
;  3                  DS$GL_CLIBASE [CLI$L_SUBT] = 0;          ! Start at first subtest.                             [02]
;  3                  QA$W_Save_Test = -1;                     ! Set to -1 to indicate that it has not been saved.   [02]
;  3                  QA$W_Save_Last = -1;                     ! Set to -1 to indicate that it has not been saved.   [02]
;  3
;  3                  QA$L_Saved_DSA_Flags = .DSA$GL_FLAGS;    ! Save the current DSA flag settings                  [02]
;  3
;  3                  DSA$V_BELL   = Off;                      ! Set BELL on error off                               [02]
;  3                  DSA$V_HALT   = Off;                      ! Set HALT off                                        [02]
;  3                  DSA$V_IE1    = Off;                      ! Set IE1 off                                         [02]
;  3                  DSA$V_IE2    = Off;                      ! Set IE2 off                                         [02]
;  3                  DSA$V_IE3    = Off;                      ! Set IE3 off                                         [02]
;  3                  DSA$V_IES    = Off;                      ! Set IES off                                         [02]
;  3                  DSA$V_LOOP   = Off;                      ! Set LOOP ON ERROR off                               [02]
;  3                  DSA$V_OPER   = Off;                      ! Set OPER off                                        [02]
;  3                  DSA$V_PROMPT = Off;                      ! Set PROMPT off                                      [02]
;  3                  DSA$V_QUICK  = Off;                      ! Set QUICK off                                       [02]
;  3                  DSA$V_SEARCH = Off;                      ! Set SEARCH off                                      [02]
;  3                  DSA$V_TRACE  = On;                       ! Set TRACE flag on.                                  [02]
;  3
;  3                  INCR State FROM 0 TO (QA$K_Numb_Routine_States - 1) DO  !                                      [02]
;  3                          QA$AOB_Routine_State [.State] = Off;            !                                      [02]
;  3
;  3                  QA$Init ();
;  3                  DS$GA_DS_Ctrl_C_First = QA$Control_C_Handler;           ! Enable DS catching of Control-C's.   [06]
;  3                  QA$AOB_Check_State [QA$K_Normal_Start] = On;
;  3                  RETURN                                                  !                                      [02]
;  3                  END
```

```
;  1777  2          ELSE IF (.Hook_Point EQL DSQA$K_Start_VRReStart) THEN          !                                    [02]
;  1778  3                  BEGIN                                                  !                                    [02]
;  1779  3                  !+
;  1780  3                  ! Restore the CLI vector. This is done at the start of each check routine to protect against    [02]
;  1781  3                  ! the CLI values getting lost while a check routine is executing the diagnostic. Only restore   [02]
;  1782  3                  ! the First and Last test numbers if they have been saved (i.e., if the saved First test number [02]
;  1783  3                  ! is not -1).                                                                                [02]
;  1784  3                  !-
;  1785  3
;  1786  3                  DS$GL_CLIBASE [CLI$L_PASS] = 1;          ! Assume one for the number of passes.                [02]
;  1787  3                  DS$GL_CLIBASE [CLI$L_SUBT] = 0;          ! Start at first subtest.                             [02]
;  1788  3
;  1789  3                  IF (.QA$W_Save_Test NEQ -1) THEN         !                                                    [02]
;  1790  4                          BEGIN                            ! The first and last tests have been saved.           [02]
;  1791  4                          DS$GL_CLIBASE [CLI$L_TEST] = .QA$W_Save_Test;   ! Restore saved first test.           [02]
;  1792  4                          DS$GL_CLIBASE [CLI$L_LAST] = .QA$W_Save_Last    ! Restore saved last test.            [02]
;  1793  4                          END                              !                                                    [02]
;  1794  3                  END                                      !                                                    [02]
;  1795  3
;  1796  2          ELSE IF (.Hook_Point EQL DSQA$K_Start_Before_Header) THEN       !                                    [02]
;  1797  3                  BEGIN
;  1798  3                  IF (.QA$W_Save_Test EQL -1) THEN         !                                                    [02]
;  1799  4                          BEGIN                            !                                                    [02]
;  1800  4                          !+
;  1801  4                          ! The first and last tests have not been saved. Save them on the first time only.       [02]
;  1802  4                          !-
;  1803  4
;  1804  4                          QA$W_Save_Test = .DS$GL_FSTTEST;! Save first test number.                             [02]
;  1805  4                          QA$W_Save_Last = .DS$GL_LSTTEST ! Save last test number.                              [02]
;  1806  3                          END;                             !                                                    [02]
;  1807  3                  RETURN                                   !                                                    [02]
;  1808  3                  END                                      !                                                    [02]
```

```
;    1810  2              ELSE IF (.Hook_Point EQL DSQA$K_Param_DSX$GetData) OR          !  [02]
;    1811  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetVield) OR        !  [02]
;    1812  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetAddress) OR      !  [02]
;    1813  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetLogical) OR      !  [02]
;    1814  2                      (.Hook_Point EQL DSQA$K_Param_DSX$GetString) THEN     !  [02]
;    1815  3                     BEGIN                                                  !  [02]
;    1816  3                     !+                                                        [02]
;    1817  3                     ! User is trying to request information from the operator. It is a QA error if no     [02]
;    1818  3                     ! default value is given.                                 [02]
;    1819  3                     !-                                                        [02]
;    1820  3
;    1821  3                     IF (.QA$AOB_Flags [QA$K_NoDef]) THEN                   !  [02]
;    1822  4                         BEGIN                                              !  [02]
;    1823  4                         $DS_PRINTF (QA$T_Operator_Request_Message);        !  [02]
;    1824  4
;    1825  5                         BEGIN                                              !  [02]
;    1826  5                         !+                                                    [02]
;    1827  5                         ! Find which check routine is currently running. If none are running, it is a logic  [02]
;    1828  5                         ! error. Use the number of the first-bit-set as the QA$K_ check routine constant, and [02]
;    1829  5                         ! add an error to this check routines summary table total.                           [02]
;    1830  5                         !-                                                    [02]
;    1831  5                         LOCAL                                              !  [02]
;    1832  5                                 First_Bit_Set;          ! Return bit position  [02]
;    1833  5
;    1834  5                         BUILTIN                                            !  [02]
;    1835  5                                 FFS;                    ! Find First Set bit   [02]
;    1836  5
;    1837  5                         FFS (%REF (0), %REF (QA$K_Number_Of_Checks), QA$AOB_Check_State, First_Bit_Set);  !  [02]
;    1838  5
;    1839  5                         IF (.First_Bit_Set EQL QA$K_Number_Of_Checks) THEN                    !           [02]
;    1840  6                                 Logic_Error ('No check state bit set - GET routine called')   !           [02]
;    1841  5                         ELSE
;    1842  6                                 BEGIN                                      !  [02]
;    1843  6                                 QA$Add_QA_Error (.First_Bit_Set);          !  [02]
;    1844  6                                 QA$AOB_Routine_State [QA$V_Error_Found] = True  !  [02]
;    1845  5                                 END;                                       !  [02]
;    1846  4                         END;                                               !  [02]
;    1847  4
;    1848  4                         QA$Abort_QA ()                  ! Abort the QA sequence  [02]
;    1849  4                         END
;    1850  3                 ELSE
;    1851  3                         !+                                                    [02]
;    1852  3                         ! There was a default value given for the GET supervisor call. Simply return.  [02]
;    1853  3                         !-                                                    [02]
;    1854  3                         RETURN
;    1855  3                 END
```

```
;   1857  2          ELSE IF (.Hook_Point EQL DSQA$K_Dispat_DSX$EndPass_Mid) THEN    ! If midpoint of EndPass routine,       [02]
;   1858  3              BEGIN                                                       !                                       [02]
;   1859  3              IF (.QA$AOB_Check_State [QA$K_Last_Check]) THEN             ! If executing last check,              [02]
;   1860  4                  BEGIN                                                   ! Then                                  [02]
;   1861  4                  QA$Print_Forced_Errors ();                              ! Print the forced-errors,              [02]
;   1862  4                  DS$GL_FSTTEST = .QA$W_Save_Test;                        ! Restore the saved test number,        [02]
;   1863  4                  DS$GL_LSTTEST = .QA$W_Save_Last;                        ! Restore the saved last test number,   [02]
;   1864  4                  QA$Print_Overall_Errors ();                             ! Print the summary table,              [02]
;   1865  4                  DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;                   ! Restore the DSA flags,                [02]
;   1866  4                  QA$Init ();                                             ! Initialize for next run,              [02]
;   1867  4                  DSA$V_QA = Off;                                         ! Turn off QA bit,                      [02]
;   1868  4                  RETURN                                                  ! And return to the EndPass routine.    [02]
;   1869  4                  END                                                     !                                       [02]
;   1870  3              END                                                         !                                       [02]
;   1871  3
;   1872  2          ELSE IF (.Hook_Point EQL DSQA$K_Kernel_Init_Context) THEN       !                                       [02]
;   1873  3              BEGIN                                                       !                                       [02]
;   1874  3              IF (.QA$AOB_Flags [QA$K_QA_Debug]) THEN                     !                                       [02]
;   1875  3                  $DS_PRINTF ($ASCIC ('!/QA_Main: Init_Context done!/')); !                                       [02]
;   1876  3              RETURN
;   1877  3              END
;   1878  3
;   1879  2          ELSE IF (.Hook_Point EQL DSQA$K_Dispat_DS_Cleanup_Bgn) THEN     !                                       [02]
;   1880  3              BEGIN                                                       !                                       [02]
;   1881  3              QA$AOB_Routine_State [QA$V_Doing_Cleanup] = True;           !                                       [02]
;   1882  3              RETURN                                                      !                                       [02]
;   1883  3              END                                                         !                                       [02]
;   1884  3
;   1885  2          ELSE IF (.Hook_Point EQL DSQA$K_Dispat_DS_Cleanup_End) THEN     !                                       [02]
;   1886  3              BEGIN                                                       !                                       [02]
;   1887  3              QA$AOB_Routine_State [QA$V_Doing_Cleanup] = False;          !                                       [02]
;   1888  3              RETURN                                                      !                                       [02]
;   1889  2              END;                                                        !                                       [02]
```

```
;    1891  2              !+
;    1892  2              ! Dispatch to the proper check routine based on what bit is set in the QA$AOB_Check_State bit vector.
;    1893  2              !-
;    1894  2
;    1895  2              SELECTONE 1 OF
;    1896  2                      SET
;    1897  2                      Check_Case (Normal_Start);
;    1898  2                      Check_Case (Multiple_Pass);
;    1899  2                      Check_Case (Loop_On_Test);
;    1900  2                      Check_Case (Loop_On_Subtest);           !                                                [01]
;    1901  2                      Check_Case (Run_Backwards);             !                                                [07]
;    1902  2                      [OTHERWISE]:
;    1903  3                              BEGIN
;    1904  3                              !+
;    1905  3                              ! We have found a QA internal bug. Print message for me and abort me.            [02]
;    1906  3                              !-
;    1907  3
;    1908  3                              Logic_Error ('QA$Main: No check state bit set.');       !                        [02]
;    1909  3                              DSA$GL_FLAGS = .QA$L_Saved_DSA_Flags;                   ! Restore DSA flags
;    1910  3                              QA$Init ();                                             ! Initialize for next run
;    1911  3                              DSA$V_QA = Off;                                         ! Turn off QA
;    1912  3                              DSX$Abort ();                                           ! Good way to stop QA.
;    1913  3                              RETURN
;    1914  2                              END;
;    1915  2                      TES;
;    1916  2
;    1917  2              !+
;    1918  2              ! If it gets to here, then a check routine failed. This implies that we detected a QA error.     [02]
;    1919  2              ! Abort me and the diagnostic.                                                                   [02]
;    1920  2              !-
;    1921  2
;    1922  2              QA$Abort_QA ()                                  !                                                [02]
;    1923  1      END;


                                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

72 74 6E 45 20 6E 69 61 4D 24 41 51 2F 21 24  0021F P.AAN:   .ASCII  \$!/QA$Main Entry, Hook_Point = !UL.!/\      ;
3D 20 74 6E 69 6F 50 5F 6B 6F 6F 48 20 2C 79  0022E                                                             ;
                        2F 21 2E 4C 55 21 20  0023D                                                             ;
61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 44  00244 P.AAO:   .ASCII  \D!/**QA Internal Error: No check state b\  ;
65 68 63 20 6F 4E 20 3A 72 6F 72 72 45 20 6C  00253                                                             ;
                  62 20 65 74 61 74 73 20 6B  00262                                                             ;
6F 72 20 54 45 47 20 2D 20 74 65 73 20 74 69  0026C           .ASCII  \it set - GET routine called!/\           ;
   2F 21 64 65 6C 6C 61 63 20 65 6E 69 74 75  0027B                                                             ;
69 6E 49 20 3A 6E 69 61 4D 5F 41 51 2F 21 1E  00289 P.AAP:   .ASCII  <30>\!/QA_Main: Init_Context done!/\       ;
21 65 6E 6F 64 20 74 78 65 74 6E 6F 43 5F 74  00298                                                             ;
                                       2F     002A7                                                             ;
61 6E 72 65 74 6E 49 20 41 51 2A 2A 2F 21 39  002A8 P.AAQ:   .ASCII  \9!/**QA Internal Error: QA$Main: No chec\  ;
69 61 4D 24 41 51 20 3A 72 6F 72 72 45 20 6C  002B7                                                             ;
                  63 65 68 63 20 6F 4E 20 3A  002C6                                                             ;
74 65 73 20 74 69 62 20 65 74 61 74 73 20 6B  002D0           .ASCII  \k state bit set.!/\                      ;
                                    2F 21 2E  002DF                                                             ;

                                                        ES=             P.AAO
                                                        ES=             P.AAQ
```

```
                                                                  .PSECT  CODE,NOWRT,  SHR,2

                                          03FC 00000              .ENTRY  QA$MAIN, Save R2,R3,R4,R5,R6,R7,R8,R9          ; 1618
                      59 00000000G  EF   9E 00002              MOVAB   DS$GL_LSTTEST, R9                              ;
                      58 00000000G  EF   9E 00009              MOVAB   DS$GL_FSTTEST, R8                              ;
                      57 00000000'  EF   9E 00010              MOVAB   P.AAN, R7                                      ;
                      56 00000000G  9F   9E 00017              MOVAB   @#DS$PRINTF, R6                                ;
                      55 00000000G  EF   9E 0001E              MOVAB   DS$GL_CLIBASE+44, R5                           ;
                      54        A9  AF   9E 00025              MOVAB   QA$INIT, R4                                    ;
                      53 00000000'  EF   9E 00029              MOVAB   QA$AOB_CHECK_STATE, R3                         ;
          08      04  A3            01   E1 00030              BBC     #1, QA$AOB_FLAGS, 1$                           ; 1695
                              04    AC   DD 00035              PUSHL   HOOK_POINT                                     ; 1697
                                    57   DD 00038              PUSHL   R7                                             ;
                      66            02   FB 0003A              CALLS   #2, DS$PRINTF                                  ;
                         0000FE01   9F   95 0003D 1$:          TSTB    @#^X0000FE01                                   ; 1703
                                    01   19 00043              BLSS    2$                                             ;
                                         04 00045              RET                                                    ;
                      52        04  AC   D0 00046 2$:          MOVL    HOOK_POINT, R2                                 ; 1738
                      16            52   D1 0004A              CMPL    R2, #22                                        ;
                                    45   12 0004D              BNEQ    5$                                             ;
                      65            01   D0 0004F              MOVL    #1, DS$GL_CLIBASE+44                           ; 1748
                              FC    A5   D4 00052              CLRL    DS$GL_CLIBASE+40                               ; 1749
                  F2  A3            01   CE 00055              MNEGL   #1, QA$W_SAVE_TEST                             ; 1750
                  F8  A3 0000FE00   9F   D0 00059              MOVL    @#^X0000FE00, QA$L_SAVED_DSA_FLAGS             ; 1753
            0000FE00  9F      10FD  8F   AA 00061              BICW2   #4349, @#^X0000FE00                            ; 1762
            0000FE01  9F        61  8F   8A 0006A              BICB2   #97, @#^X0000FE01                              ; 1765
            0000FE01  9F            04   88 00072              BISB2   #4, @#^X0000FE01                               ; 1766
                                    50   D4 00079              CLRL    STATE                                          ; 1768
          00      FC  A3            50   E5 0007B 3$:          BBCC    STATE, QA$AOB_ROUTINE_STATE, 4$                ; 1769
          F7          50            03   F3 00080 4$:          AOBLEQ  #3, STATE, 3$                                  ;
                      64            00   FB 00084              CALLS   #0, QA$INIT                                    ; 1771
            00000000G EF      FE1F  C4   9E 00087              MOVAB   QA$CONTROL_C_HANDLER, DS$GA_DS_CTRL_C_FIRST    ; 1772
                      63            01   88 00090              BISB2   #1, QA$AOB_CHECK_STATE                         ; 1773
                                         04 00093              RET                                                    ;
                      12            52   D1 00094 5$:          CMPL    R2, #18                                        ; 1777
                                    1A   12 00097              BNEQ    6$                                             ;
                      65            01   D0 00099              MOVL    #1, DS$GL_CLIBASE+44                           ; 1786
                              FC    A5   D4 0009C              CLRL    DS$GL_CLIBASE+40                               ; 1787
                FFFF  8F        F2  A3   B1 0009F              CMPW    QA$W_SAVE_TEST, #-1                            ; 1789
                                    6C   13 000A5              BEQL    13$                                            ;
                  F4  A5        F2  A3   32 000A7              CVTWL   QA$W_SAVE_TEST, DS$GL_CLIBASE+32               ; 1791
                  F8  A5        F4  A3   32 000AC              CVTWL   QA$W_SAVE_LAST, DS$GL_CLIBASE+36               ; 1792
                                    60   11 000B1              BRB     13$                                            ;
                      17            52   D1 000B3 6$:          CMPL    R2, #23                                        ; 1796
                                    12   12 000B6              BNEQ    8$                                             ;
                FFFF  8F        F2  A3   B1 000B8              CMPW    QA$W_SAVE_TEST, #-1                            ; 1798
                                    01   13 000BE              BEQL    7$                                             ;
                                         04 000C0              RET                                                    ;
                  F2  A3            68   B0 000C1 7$:          MOVW    DS$GL_FSTTEST, QA$W_SAVE_TEST                  ; 1804
                  F4  A3            69   B0 000C5              MOVW    DS$GL_LSTTEST, QA$W_SAVE_LAST                  ; 1805
                                         04 000C9              RET                                                    ;
                      0C            52   D1 000CA 8$:          CMPL    R2, #12                                        ; 1810
                                    14   13 000CD              BEQL    9$                                             ;
                      0D            52   D1 000CF              CMPL    R2, #13                                        ; 1811
                                    0F   13 000D2              BEQL    9$                                             ;
```

```
                                0E          52  D1 000D4          CMPL    R2, #14                                  ; 1812
                                            0A  13 000D7          BEQL    9$                                       ;
                                0F          52  D1 000D9          CMPL    R2, #15                                  ; 1813
                                            05  13 000DC          BEQL    9$                                       ;
                                10          52  D1 000DE          CMPL    R2, #16                                  ; 1814
                                            32  12 000E1          BNEQ    14$                                      ;
                                01      04  A3  E8 000E3 9$:     BLBS    QA$AOB_FLAGS, 10$                        ; 1821
                                                04 000E7          RET                                              ;
                            00000000G       EF  9F 000E8 10$:     PUSHAB  QA$T_OPERATOR_REQUEST_MESSAGE            ; 1823
                                66          01  FB 000EE          CALLS   #1, DS$PRINTF                            ;
    50          63              0F          00  EA 000F1          FFS     #0, #15, QA$AOB_CHECK_STATE, FIRST_BIT_SET ; 1837
                                0F          50  D1 000F6          CMPL    FIRST_BIT_SET, #15                       ; 1839
                                            08  12 000F9          BNEQ    11$                                      ;
                                        25  A7  9F 000FB          PUSHAB  ES                                       ; 1840
                                66          01  FB 000FE          CALLS   #1, DS$PRINTF                            ;
                                            0B  11 00101          BRB     12$                                      ;
                                            50  DD 00103 11$:     PUSHL   FIRST_BIT_SET                            ; 1843
                        FE11    C4          01  FB 00105          CALLS   #1, QA$ADD_QA_ERROR                      ;
                          FC    A3          02  88 0010A          BISB2   #2, QA$AOB_ROUTINE_STATE                 ; 1844
                        FDDC    C4          00  FB 0010E 12$:     CALLS   #0, QA$ABORT_QA                          ; 1848
                                            58  11 00113 13$:     BRB     19$                                      ;
                                            52  D5 00115 14$:     TSTL    R2                                       ; 1857
                                            2E  12 00117          BNEQ    15$                                      ;
                    50          63          04  E1 00119          BBC     #4, QA$AOB_CHECK_STATE, 19$              ; 1859
                        00000000V   EF      00  FB 0011D          CALLS   #0, QA$PRINT_FORCED_ERRORS               ; 1861
                                68      F2  A3  32 00124          CVTWL   QA$W_SAVE_TEST, DS$GL_FSTTEST            ; 1862
                                69      F4  A3  32 00128          CVTWL   QA$W_SAVE_LAST, DS$GL_LSTTEST            ; 1863
                        00000000V   EF      00  FB 0012C          CALLS   #0, QA$PRINT_OVERALL_ERRORS              ; 1864
                        0000FE00    9F  F8  A3  D0 00133          MOVL    QA$L_SAVED_DSA_FLAGS, @#^X0000FE00       ; 1865
                                64          00  FB 0013B          CALLS   #0, QA$INIT                              ; 1866
                        0000FE01    9F  80  8F  8A 0013E          BICB2   #128, @#^X0000FE01                       ; 1867
                                                04 00146          RET                                              ;
                                08          52  D1 00147 15$:     CMPL    R2, #8                                   ; 1872
                                            0D  12 0014A          BNEQ    17$                                      ;
                    01      04  A3          01  E0 0014C          BBS     #1, QA$AOB_FLAGS, 16$                    ; 1874
                                                04 00151          RET                                              ;
                                        6A  A7  9F 00152 16$:     PUSHAB  P.AAP                                    ; 1875
                                66          01  FB 00155          CALLS   #1, DS$PRINTF                            ;
                                                04 00158          RET                                              ;
                                03          52  D1 00159 17$:     CMPL    R2, #3                                   ; 1879
                                            05  12 0015C          BNEQ    18$                                      ;
                          FC    A3          04  88 0015E          BISB2   #4, QA$AOB_ROUTINE_STATE                 ; 1881
                                                04 00162          RET                                              ;
                                04          52  D1 00163 18$:     CMPL    R2, #4                                   ; 1885
                                            05  12 00166          BNEQ    19$                                      ;
                          FC    A3          04  8A 00168          BICB2   #4, QA$AOB_ROUTINE_STATE                 ; 1887
                                                04 0016C          RET                                              ;
                                0B          63  E9 0016D 19$:     BLBC    QA$AOB_CHECK_STATE, 20$                  ; 1897
                                            52  DD 00170          PUSHL   R2                                       ;
                        00000000G   EF      01  FB 00172          CALLS   #1, QA$NORMAL_START                      ;
                                            3A  11 00179          BRB     24$                                      ;
                    0B          63          01  E1 0017B 20$:     BBC     #1, QA$AOB_CHECK_STATE, 21$              ; 1898
                                            52  DD 0017F          PUSHL   R2                                       ;
                        00000000G   EF      01  FB 00181          CALLS   #1, QA$MULTIPLE_PASS                     ;
                                            2B  11 00188          BRB     24$                                      ;
                    0B          63          02  E1 0018A 21$:     BBC     #2, QA$AOB_CHECK_STATE, 22$              ; 1899
                                            52  DD 0018E          PUSHL   R2                                       ;
```

```
          00000000G  EF          01  FB 00190        CALLS   #1, QA$LOOP_ON_TEST                          ;
                                 1C  11 00197        BRB     24$                                         ;
       0B            63          03  E1 00199 22$:   BBC     #3, QA$AOB_CHECK_STATE, 23$                 ; 1900
                                 52  DD 0019D        PUSHL   R2                                          ;
          00000000G  EF          01  FB 0019F        CALLS   #1, QA$LOOP_ON_SUBTEST                      ;
                                 0D  11 001A6        BRB     24$                                         ;
       0D            63          04  E1 001A8 23$:   BBC     #4, QA$AOB_CHECK_STATE, 25$                 ; 1901
                                 52  DD 001AC        PUSHL   R2                                          ;
          00000000G  EF          01  FB 001AE        CALLS   #1, QA$RUN_BACKWARDS                        ;
                     23          50  E9 001B5 24$:   BLBC    R0, 26$                                     ;
                                     04 001B8        RET                                                 ;
                          0089   C7  9F 001B9 25$:   PUSHAB  ES                                          ; 1908
                     66          01  FB 001BD        CALLS   #1, DS$PRINTF                               ;
          0000FE00   9F      F8   A3  D0 001C0        MOVL    QA$L_SAVED_DSA_FLAGS, a#^X0000FE00          ; 1909
                     64          00  FB 001C8        CALLS   #0, QA$INIT                                 ; 1910
          0000FE01   9F      80   8F  8A 001CB        BICB2   #128, a#^X0000FE01                          ; 1911
          00000000G  EF          00  FB 001D3        CALLS   #0, DSX$ABORT                               ; 1912
                                     04 001DA        RET                                                 ;
              FDDC   C4          00  FB 001DB 26$:   CALLS   #0, QA$ABORT_QA                             ; 1922
                                     04 001E0        RET                                                 ; 1923

; Routine Size:  481 bytes,    Routine Base:  CODE + 039F
```

```
;    1925  1      %SBTTL 'QA$Print_Forced_Errors'
;    1926  1
;    1927  1      GLOBAL ROUTINE QA$Print_Forced_Errors : NOVALUE =       !                                    All [02]
;    1928  1
;    1929  1      !++
;    1930  1      ! FUNCTIONAL DESCRIPTION:
;    1931  1      !
;    1932  1      !       This routine prints the Forced-Errors Summary Table, which gives the number of forced errors for the
;    1933  1      !       three Error checks.
;    1934  1      !
;    1935  1      ! CALLER(S):
;    1936  1      !
;    1937  1      !       The Error check routines.
;    1938  1      !
;    1939  1      ! FORMAL PARAMETERS:
;    1940  1      !
;    1941  1      !       NONE
;    1942  1      !
;    1943  1      ! IMPLICIT INPUTS:
;    1944  1      !
;    1945  1      !
;    1946  1      !
;    1947  1      ! IMPLICIT OUTPUTS:
;    1948  1      !
;    1949  1      !       Forced-Errors Summary Table printout.
;    1950  1      !
;    1951  1      ! COMPLETION CODES:
;    1952  1      !
;    1953  1      !       NONE
;    1954  1      !
;    1955  1      ! SIDE EFFECTS:
;    1956  1      !
;    1957  1      !       NONE
;    1958  1      !
;    1959  1      !--
```

```
;    1961  1     %SBTTL 'Dispose'
;    1962  2     BEGIN                                                   ! Print_Forced_Errors Routine
;    1963  2     !
;    1964  2     !       !++
;    1965  2     !       ! FUNCTIONAL DESCRIPTION:
;    1966  2     !       !
;    1967  2     !       !       This routine disposes of a 16-byte vector.
;    1968  2     !       !
;    1969  2     !       ! CALLER (S):
;    1970  2     !       !
;    1971  2     !       !       QA$Print_Forced_Errors
;    1972  2     !       !
;    1973  2     !       ! FORMAL PARAMETERS:
;    1974  2     !       !
;    1975  2     !       !       Record_Ptr - Contains the address of the 16-byte vector.
;    1976  2     !       !
;    1977  2     !       ! IMPLICIT INPUTS:
;    1978  2     !       !
;    1979  2     !       !       NONE
;    1980  2     !       !
;    1981  2     !       ! IMPLICIT OUTPUTS:
;    1982  2     !       !
;    1983  2     !       !       NONE
;    1984  2     !       !
;    1985  2     !       ! COMPLETION CODES:
;    1986  2     !       !
;    1987  2     !       !       NONE
;    1988  2     !       !
;    1989  2     !       ! SIDE EFFECTS:
;    1990  2     !       !
;    1991  2     !       !       NONE
;    1992  2     !       !
;    1993  2     !       !--
```

```
;   1994  2     !
;   1995  2     !
;   1996  2     ! ROUTINE Dispose (Record_Ptr) : NOVALUE =
;   1997  2     !       BEGIN
;   1998  2     !               MAP
;   1999  2     !                       Record_Ptr : REF VECTOR [, WORD];               ! Treat the Record_Ptr as a vector of words
;   2000  2     !
;   2001  2     !               Record_Ptr [4] = QA$K_Forced_Error_Record_Size;         ! Put in the size of the data structure
;   2002  2     !               EXE$DeaNonPaged (.Record_Ptr);                          ! Deallocate it
;   2003  2     !       END;
```

```
;    2004  2      !
;    2005  2      !%SBTTL 'QA$Print_Forced_Errors'
;    2006  2      !       !+
;    2007  2      !       ! The are the strings that get printed in the table.
;    2008  2      !       !-
;    2009  2      !
;    2010  2      !       BIND
;    2011  2      !               Header_Line_1 =                                         !                       [04]
;    2012  2      !                       $ASCIC ('!^!/!/!#* Summary of Forced-Errors!/'),!                       [04]
;    2013  2      !
;    2014  2      !               Header_Line_2 =                                         !                       [04]
;    2015  2      !                       $ASCIC ('!#* ------------------------!/!/'),    !                       [04]
;    2016  2      !
;    2017  2      !               Test_Subtest_Line_1 =
;    2018  2      !                       $ASCIC ('Test !3UW   Subtest !3UL!/'),
;    2019  2      !
;    2020  2      !               Test_Subtest_Line_2 =
;    2021  2      !                       $ASCIC ('!8*-   !11*-!/'),
;    2022  2      !
;    2023  2      !               Error_Count_Line =
;    2024  2      !                       $ASCIC ('!5UW-!4<!UL!> '),
;    2025  2      !
;    2026  2      !               Eol_Line =
;    2027  2      !                       $ASCIC ('!/');
;    2028  2      !
;    2029  2      !       LITERAL
;    2030  2      !               QA$K_Error_Count_Line_Len = 11,                 ! The length (in chars) of the                  [04]
;    2031  2      !                                                               ! Error_Count_Line line.                        [04]
;    2032  2      !
;    2033  2      !               QA$K_Minimum_Width = 24;                        ! The minimum width that can be used to print
;    2034  2      !                                                               ! the forced-error table. Currently this is the
;    2035  2      !                                                               ! length of the Header_Line.
;    2036  2      !
;    2037  2      !       LOCAL
;    2038  2      !               Number_Of_Error_Counts : WORD,                  ! The number of error number/error counts to
;    2039  2      !                                                               ! print per line.
;    2040  2      !
;    2041  2      !               Number_Printed : WORD,                          ! The running total of the number of error
;    2042  2      !                                                               ! number/error counts printed per line.
;    2043  2      !
;    2044  2      !               Record_Ptr,                                     ! The pointer that is used to acess all the
;    2045  2      !                                                               ! records in the linked list.
;    2046  2      !
;    2047  2      !               Temp_Ptr,                                       ! The pointer to dispose.
;    2048  2      !
;    2049  2      !               Width : WORD,                                   ! The maximum width of the output line.         [04]
;    2050  2      !
;    2051  2      !               Test : WORD,                                    ! The Test_Number from the forced-error record.
;    2052  2      !
;    2053  2      !               Subtest;                                        ! The Subtest_Number from the forced-error record.
```

```
;    2054  2      !
;    2055  2      !       Width                   = MAXU (QA$K_Minimum_Width, .DS$GB_WIDTH);      !                [04]
;    2056  2      !       Number_Of_Error_Counts = .Width  / QA$K_Error_Count_Line_Len;            !                [04]
;    2057  2      !
;    2058  2      !       $DS_PRINTF (Header_Line_1, ((.Width - QA$K_Minimum_Width + 1) / 2));    !                [04]
;    2059  2      !       $DS_PRINTF (Header_Line_2, ((.Width - QA$K_Minimum_Width + 1) / 2));    !                [04]
;    2060  2      !
;    2061  2      !       !+
;    2062  2      !       ! If it exists, skip the first null record and dispose of it.
;    2063  2      !       !-
;    2064  2      !
;    2065  2      !       Record_Ptr = .QA$A_Error_List_Head;
;    2066  2      !
;    2067  2      !       IF (.Record_Ptr NEQ QA$K_Nil) THEN
;    2068  2      !               BEGIN
;    2069  2      !                       MAP
;    2070  2      !                               QA$A_Error_List_Head : Error_Record;
;    2071  2      !
;    2072  2      !                       QA$A_Error_List_Head = .QA$A_Error_List_Head [QA$VA_Next_Error];
;    2073  2      !                       Dispose (.Record_Ptr);
;    2074  2      !                       Record_Ptr = .QA$A_Error_List_Head
;    2075  2      !               END;
```

```
;    2076  2      !
;    2077  2      !       WHILE (.QA$A_Error_List_Head NEQ QA$K_Nil) DO
;    2078  2      !               BEGIN
;    2079  2      !               MAP
;    2080  2      !                       Record_Ptr : Error_Record;
;    2081  2      !
;    2082  2      !               Test            = .Record_Ptr [QA$VW_Test_Number];
;    2083  2      !               Subtest         = .Record_Ptr [QA$VL_Subtest_Number];
;    2084  2      !               Number_Printed = 0;
;    2085  2      !
;    2086  2      !               $DS_PRINTF (Test_Subtest_Line_1, .Test, .Subtest);
;    2087  2      !               $DS_PRINTF (Test_Subtest_Line_2);
;    2088  2      !
;    2089  2      !               DO
;    2090  2      !                       BEGIN
;    2091  2      !                       $DS_PRINTF (Error_Count_Line,
;    2092  2      !                               .Record_Ptr [QA$VW_Error_Number],
;    2093  2      !                               .Record_Ptr [QA$VL_Error_Count]);
;    2094  2      !
;    2095  2      !                       IF ((Number_Printed = .Number_Printed + 1) EQL .Number_Of_Error_Counts) THEN
;    2096  2      !                               BEGIN
;    2097  2      !                               Number_Printed = 0;
;    2098  2      !                               $DS_PRINTF (Eol_Line);
;    2099  2      !                               END;
;    2100  2      !
;    2101  2      !                       Temp_Ptr              = .Record_Ptr;
;    2102  2      !                       Record_Ptr            = .Record_Ptr [QA$VA_Next_Error];
;    2103  2      !                       QA$A_Error_List_Head = .Record_Ptr;
;    2104  2      !
;    2105  2      !                       Dispose (.Temp_Ptr);
;    2106  2      !
;    2107  2      !                       IF (.Record_Ptr EQL QA$K_Nil) THEN
;    2108  2      !                               EXITLOOP;
;    2109  2      !                       END
;    2110  2      !               UNTIL ((.Test NEQ .Record_Ptr [QA$VW_Test_Number]) OR
;    2111  2      !                       (.Subtest NEQ .Record_Ptr [QA$VL_Subtest_Number]));
;    2112  2      !
;    2113  2      !               $DS_PRINTF (Eol_Line);
;    2114  2      !               IF (.Number_Printed NEQ 0) THEN
;    2115  2      !                       $DS_PRINTF (Eol_Line)
;    2116  2      !               END;
;    2117  2      !
;    2118  2      !       QA$A_Prev_Test_Tail  = QA$K_Nil;
;    2119  2      !       QA$A_Error_List_Head = QA$K_Nil;
;    2120  2
;    2121  2              1;                                              !                                       [07]
;    2122  2
;    2123  1      END;


                                        0000 00000          .ENTRY  QA$PRINT_FORCED_ERRORS, Save nothing        ; 1927
                                          04 00002          RET                                                 ; 2123

; Routine Size:  3 bytes,    Routine Base:  CODE + 0580
```

```
;    2125  1      %SBTTL 'QA$Print_Overall_Errors'
;    2126  1
;    2127  1      ROUTINE QA$Print_Overall_Errors : NOVALUE =
;    2128  1
;    2129  1      !++
;    2130  1      ! FUNCTIONAL DESCRIPTION:
;    2131  1      !
;    2132  1      !       This routine prints the Overall Error Summary Table, which summarizes the results for one QA execution
;    2133  1      !       of the diagnostic.
;    2134  1      !
;    2135  1      ! CALLER(S):
;    2136  1      !
;    2137  1      !       QA$Main
;    2138  1      !
;    2139  1      ! FORMAL PARAMETERS:
;    2140  1      !
;    2141  1      !       NONE
;    2142  1      !
;    2143  1      ! IMPLICIT INPUTS:
;    2144  1      !
;    2145  1      !       QA$W_MultiplePass          - The number of passes to make for the Multiple Pass Check.
;    2146  1      !       QA$W_TestLoops             - The number of loops for the Infinite Loop-On-Test Check.
;    2147  1      !       QA$W_SubtestLoops          - The number of loops for the Infinite Loop-On-Subtest Check.
;    2148  1      !       QA$AW_Overall_Error_Table - The number of errors that occurred in each check.
;    2149  1      !
;    2150  1      ! IMPLICIT OUTPUTS:
;    2151  1      !
;    2152  1      !       Overall Error Summary Table printout.
;    2153  1      !
;    2154  1      ! COMPLETION CODES:
;    2155  1      !
;    2156  1      !       NONE
;    2157  1      !
;    2158  1      ! SIDE EFFECTS:
;    2159  1      !
;    2160  1      !       NONE
;    2161  1      !
;    2162  1      !--
```

```
;    2164  2      BEGIN
;    2165  2
;    2166  2              !+
;    2167  2              ! The are the strings that get printed in each line of the Overall Error Summary Table.
;    2168  2              !-
;    2169  2
;    2170  2              BIND
;    2171  2                      Header_Line_1 =                                                 !                       [04]
;    2172  2                              $ASCIC ('!^!/!/!11* Overall Error Summary Table!/'),    !                       [04]
;    2173  2
;    2174  2                      Header_Line_2 =                                                 !                       [04]
;    2175  2                              $ASCIC ('!11* --------------------------!/'),           !                       [04]
;    2176  2
;    2177  2                      Header_Line_3 =                                                 !                       [04]
;    2178  2                              $ASCIC ('QA Checklist Item!16* Number of Errors!/'),
;    2179  2
;    2180  2                      Header_Line_4 =                                                 !                       [04]
;    2181  2                              $ASCIC ('!17*-!16* !16*-!/'),
;    2182  2
;    2183  2                      NS_Line =
;    2184  2                              $ASCIC ('!41<Normal Start!50*.!>!UW!/'),
;    2185  2
;    2186  2                      MP_Line =
;    2187  2                              $ASCIC ('!41<Multiple Pass (!UW passes)!50*.!>!UW!/'),
;    2188  2
;    2189  2                      LT_Line =
;    2190  2                              $ASCIC ('!41<Infinite Loop-On-Test (!UW passes)!50*.!>!UW!/'),
;    2191  2
;    2192  2                      LS_Line =                                                                       !       [01]
;    2193  2                              $ASCIC ('!41<Infinite Loop-On-Subtest (!UW passes)!50*.!>!UW!/'),       !       [01]
;    2194  2
;    2195  2                      RB_Line =
;    2196  2                              $ASCIC ('!41<Run Tests Backwards!50*.!>!UW!/'),
;    2197  2
;    2198  2                      OK_Line =                                                                       !       [01]
;    2199  2                              $ASCIC ('!/** QA ** !16AC passed Auto-QA portion of Quality Assurance Checklist!/');
```

```
;    2201  2              !+
;    2202  2              ! If the /TEST qualifier was given, do not print this table out. This prevents someone from running QA  [02]
;    2203  2              ! on only a small portion of the diagnostic, and having that portion work. The following must therefore [02]
;    2204  2              ! be true:                                                                                              [02]
;    2205  2              !       .DS$GL_FSTTEST = 1                                                                             [02]
;    2206  2              !       .DS$GL_LSTTEST = ..L$A_TSTCNT                                                                  [02]
;    2207  2              ! If either one of these is false, simply return. Note: accept the following if a diagnsotic contains   [02]
;    2208  2              ! x tests: /TEST:1:x.
;    2209  2              !-
;    2210  2
;    2211  2              IF ((.DS$GL_FSTTEST NEQ 1) OR (.DS$GL_LSTTEST NEQ ..L$A_TSTCNT)) THEN   !                              [02]
;    2212  2                      RETURN;                                                          !                              [02]
;    2213  2
;    2214  2              !+
;    2215  2              ! Print header lines.
;    2216  2              !-
;    2217  2
;    2218  2              $DS_PRINTF (Header_Line_1);                                             !                              [04]
;    2219  2              $DS_PRINTF (Header_Line_2);                                             !                              [04]
;    2220  2              $DS_PRINTF (Header_Line_3);                                             !                              [04]
;    2221  2              $DS_PRINTF (Header_Line_4);                                             !                              [04]
;    2222  2
;    2223  2              !+
;    2224  2              ! Print each line for the checks, until a non-zero error count is
;    2225  2              ! reached, or until the last check. If all the checks were non-zero,
;    2226  2              ! print the "passed QA" message.
;    2227  2              !-
```

```
;    2229  2              !+
;    2230  2              ! Print and check Normal Start errors
;    2231  2              !-
;    2232  2
; P  2233  2              $DS_PRINTF (NS_Line,
;    2234  2                      .QA$AW_Overall_Error_Table [QA$K_Normal_Start]);
;    2235  2
;    2236  2              IF (.QA$AW_Overall_Error_Table [QA$K_Normal_Start] NEQ 0) THEN
;    2237  2                  RETURN;
;    2238  2
;    2239  2              !+
;    2240  2              ! Print and check Multiple Pass errors
;    2241  2              !-
;    2242  2
; P  2243  2              $DS_PRINTF (MP_Line, .QA$W_MultiplePass,
;    2244  2                      .QA$AW_Overall_Error_Table [QA$K_Multiple_Pass]);
;    2245  2
;    2246  2              IF (.QA$AW_Overall_Error_Table [QA$K_Multiple_Pass] NEQ 0) THEN
;    2247  2                  RETURN;
;    2248  2
;    2249  2              !+
;    2250  2              ! Print and check Loop-On-Test erros
;    2251  2              !-
;    2252  2
; P  2253  2              $DS_PRINTF (LT_Line, .QA$W_TestLoops,
;    2254  2                      .QA$AW_Overall_Error_Table [QA$K_Loop_On_Test]);
;    2255  2
;    2256  2              IF (.QA$AW_Overall_Error_Table [QA$K_Loop_On_Test] NEQ 0) THEN
;    2257  2                  RETURN;
;    2258  2
;    2259  2              !+                                                                  [01]
;    2260  2              ! Print and check Loop-On-Subtest errors                            [01]
;    2261  2              !-                                                                  [01]
;    2262  2
; P  2263  2              $DS_PRINTF (LS_Line, .QA$W_SubtestLoops,                    !       [01]
;    2264  2                      .QA$AW_Overall_Error_Table [QA$K_Loop_On_Subtest]);  !       [01]
;    2265  2
;    2266  2              IF (.QA$AW_Overall_Error_Table [QA$K_Loop_On_Subtest] NEQ 0) THEN   !   [01]
;    2267  2                  RETURN;                                                 !       [01]
```

```
;    2269  2              !+
;    2270  2              ! Print and check Run Tests Backwards errors
;    2271  2              !-
;    2272  2
; P  2273  2              $DS_PRINTF (RB_Line,
;    2274  2                      .QA$AW_Overall_Error_Table [QA$K_Run_Backwards]);
;    2275  2
;    2276  2              IF (.QA$AW_Overall_Error_Table [QA$K_Run_Backwards] NEQ 0) THEN
;    2277  2                  RETURN;
;    2278  2
;    2279  2              $DS_PRINTF (OK_Line, .L$A_NAME);
;    2280  2
;    2281  2              RETURN
;    2282  1      END;


                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

65  76  4F  20  2A  31  31  21  2F  21  2F  21  5E  21  28   002E2 P.AAR:  .ASCII  \(!^!/!/!11* Overall Error Summary Table!\  ;
6D  6D  75  53  20  72  6F  72  72  45  20  6C  6C  61  72   002F1                                                            ;
                    21  65  6C  62  61  54  20  79  72  61   00300                                                            ;
                                                        2F   0030A          .ASCII  \/\                                      ;
2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  20  2A  31  31  21   0030B P.AAS:  .ASCII  \"!11* ---------------------------!/\      ;
2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D   0031A                                                            ;
                                    2F  21  2D  2D  2D  2D   00329                                                            ;
49  20  74  73  69  6C  6B  63  65  68  43  20  41  51  28   0032E P.AAT:  .ASCII  \(QA Checklist Item!16* Number of Errors!\  ;
20  72  65  62  6D  75  4E  20  2A  36  31  21  6D  65  74   0033D                                                            ;
                    21  73  72  6F  72  72  45  20  66  6F   0034C                                                            ;
                                                        2F   00356          .ASCII  \/\                                      ;
2A  36  31  21  20  2A  36  31  21  2D  2A  37  31  21  11   00357 P.AAU:  .ASCII  <17>\!17*-!16* !16*-!/\                   ;
                                                    2F  21   00366                                                            ;
61  74  53  20  6C  61  6D  72  6F  4E  3C  31  34  21  1C   00369 P.AAV:  .ASCII  <28>\!41<Normal Start!50*.!>!UW!/\         ;
    2F  21  57  55  21  3E  21  2E  2A  30  35  21  74  72   00378                                                            ;
50  20  65  6C  70  69  74  6C  75  4D  3C  31  34  21  2A   00386 P.AAW:  .ASCII  \*!41<Multiple Pass (!UW passes)!50*.!>!U\  ;
73  65  73  73  61  70  20  57  55  21  28  20  73  73  61   00395                                                            ;
                    55  21  3E  21  2E  2A  30  35  21  29   003A4                                                            ;
                                                    2F  21   003AE          .ASCII  \W!/\                                    ;
4C  20  65  74  69  6E  69  66  6E  49  3C  31  34  21  32   003B1 P.AAX:  .ASCII  \2!41<Infinite Loop-On-Test (!UW passes)!\ ;
55  21  28  20  74  73  65  54  2D  6E  4F  2D  70  6F  6F   003C0                                                            ;
                    21  29  73  65  73  73  61  70  20  57   003CF                                                            ;
                2F  21  57  55  21  3E  21  2E  2A  30  35   003D9          .ASCII  \50*.!>!UW!/\                            ;
4C  20  65  74  69  6E  69  66  6E  49  3C  31  34  21  35   003E4 P.AAY:  .ASCII  \5!41<Infinite Loop-On-Subtest (!UW passe\ ;
20  74  73  65  74  62  75  53  2D  6E  4F  2D  70  6F  6F   003F3                                                            ;
                    65  73  73  61  70  20  57  55  21  28   00402                                                            ;
    2F  21  57  55  21  3E  21  2E  2A  30  35  21  29  73   0040C          .ASCII  \s)!50*.!>!UW!/\                         ;
20  73  74  73  65  54  20  6E  75  52  3C  31  34  21  23   0041A P.AAZ:  .ASCII  \#!41<Run Tests Backwards!50*.!>!UW!/\      ;
21  2E  2A  30  35  21  73  64  72  61  77  6B  63  61  42   00429                                                            ;
                                    2F  21  57  55  21  3E   00438                                                            ;
36  31  21  20  2A  2A  20  41  51  20  2A  2A  2F  21  48   0043E P.ABA:  .ASCII  \H!/** QA ** !16AC passed Auto-QA portion\  ;
2D  6F  74  75  41  20  64  65  73  73  61  70  20  43  41   0044D                                                            ;
                    6E  6F  69  74  72  6F  70  20  41  51   0045C                                                            ;
73  73  41  20  79  74  69  6C  61  75  51  20  66  6F  20   00466          .ASCII  \ of Quality Assurance Checklist!/\      ;
73  69  6C  6B  63  65  68  43  20  65  63  6E  61  72  75   00475                                                            ;
                                                    2F  21  74   00484                                                            ;

                                                                HEADER_LINE_1=      P.AAR
```

```
                                HEADER_LINE_2=      P.AAS
                                HEADER_LINE_3=      P.AAT
                                HEADER_LINE_4=      P.AAU
                                NS_LINE=            P.AAV
                                MP_LINE=            P.AAW
                                LT_LINE=            P.AAX
                                LS_LINE=            P.AAY
                                RB_LINE=            P.AAZ
                                OK_LINE=            P.ABA


                                        .PSECT  CODE,NOWRT,  SHR,2

                                001C 00000 QA$PRINT_OVERALL_ERRORS:
                                                .WORD   Save R2,R3,R4                                   ; 2127
                    54 00000000'    EF  9E 00002        MOVAB   QA$AW_OVERALL_ERROR_TABLE, R4           ;
                    53 00000000G    9F  9E 00009        MOVAB   a#DS$PRINTF, R3                         ;
                    52 00000000'    EF  9E 00010        MOVAB   HEADER_LINE_1, R2                       ;
                    01 00000000G    EF  D1 00017        CMPL    DS$GL_FSTTEST, #1                       ; 2211
                                    78  12 0001E        BNEQ    1$                                      ;
                    50 00000254     9F  D0 00020        MOVL    a#^X00000254, R0                        ;
                    60 00000000G    EF  D1 00027        CMPL    DS$GL_LSTTEST, (R0)                     ;
                                    78  12 0002E        BNEQ    2$                                      ;
                                    52  DD 00030        PUSHL   R2                                      ; 2218
                    63              01  FB 00032        CALLS   #1, DS$PRINTF                           ;
                            29      A2  9F 00035        PUSHAB  HEADER_LINE_2                           ; 2219
                    63              01  FB 00038        CALLS   #1, DS$PRINTF                           ;
                            4C      A2  9F 0003B        PUSHAB  HEADER_LINE_3                           ; 2220
                    63              01  FB 0003E        CALLS   #1, DS$PRINTF                           ;
                            75      A2  9F 00041        PUSHAB  HEADER_LINE_4                           ; 2221
                    63              01  FB 00044        CALLS   #1, DS$PRINTF                           ;
                    7E              64  3C 00047        MOVZWL  QA$AW_OVERALL_ERROR_TABLE, -(SP)        ; 2234
                            0087    C2  9F 0004A        PUSHAB  NS_LINE                                 ;
                    63              02  FB 0004E        CALLS   #2, DS$PRINTF                           ;
                                    64  B5 00051        TSTW    QA$AW_OVERALL_ERROR_TABLE               ; 2236
                                    62  12 00053        BNEQ    3$                                      ;
                    7E          02  A4  3C 00055        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+2, -(SP)      ; 2244
                    7E 00000000G    EF  3C 00059        MOVZWL  QA$W_MULTIPLEPASS, -(SP)                ;
                            00A4    C2  9F 00060        PUSHAB  MP_LINE                                 ;
                    63              03  FB 00064        CALLS   #3, DS$PRINTF                           ;
                            02      A4  B5 00067        TSTW    QA$AW_OVERALL_ERROR_TABLE+2             ; 2246
                                    4B  12 0006A        BNEQ    3$                                      ;
                    7E          04  A4  3C 0006C        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+4, -(SP)      ; 2254
                    7E 00000000G    EF  3C 00070        MOVZWL  QA$W_TESTLOOPS, -(SP)                   ;
                            00CF    C2  9F 00077        PUSHAB  LT_LINE                                 ;
                    63              03  FB 0007B        CALLS   #3, DS$PRINTF                           ;
                            04      A4  B5 0007E        TSTW    QA$AW_OVERALL_ERROR_TABLE+4             ; 2256
                                    34  12 00081        BNEQ    3$                                      ;
                    7E          06  A4  3C 00083        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+6, -(SP)      ; 2264
                    7E 00000000G    EF  3C 00087        MOVZWL  QA$W_SUBTESTLOOPS, -(SP)                ;
                            0102    C2  9F 0008E        PUSHAB  LS_LINE                                 ;
                    63              03  FB 00092        CALLS   #3, DS$PRINTF                           ;
                            06      A4  B5 00095        TSTW    QA$AW_OVERALL_ERROR_TABLE+6             ; 2266
                                    1D  12 00098 1$:    BNEQ    3$                                      ;
                    7E          08  A4  3C 0009A        MOVZWL  QA$AW_OVERALL_ERROR_TABLE+8, -(SP)      ; 2274
                            0138    C2  9F 0009E        PUSHAB  RB_LINE                                 ;
                    63              02  FB 000A2        CALLS   #2, DS$PRINTF                           ;
```

```
                    08   A4  B5 000A5          TSTW    QA$AW_OVERALL_ERROR_TABLE+8                ; 2276
                         0D  12 000A8 2$:      BNEQ    3$                                         ;
              00000208   9F  DD 000AA          PUSHL   @#^X00000208                               ; 2279
                  015C   C2  9F 000B0          PUSHAB  OK_LINE                                    ;
          63             02  FB 000B4          CALLS   #2, DS$PRINTF                              ;
                             04 000B7 3$:      RET                                                ; 2282

; Routine Size:  184 bytes,    Routine Base:  CODE + 0583
```

```
;   2284  1     %SBTTL 'End of Module QAMAIN'
;   2285  1
;   2286  0     END ELUDOM
```

```
;                                  PSECT SUMMARY
;
;       Name                    Bytes                        Attributes
;
;  DATA                          1159  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                            53  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          1595  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

Symbol	Type	Defined	Referenced ...										
$ASCIC	Unbound		256	285									
	Macro	Lib01	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1524
			1534	1696	1840	1875	1908	2172	2175	2178	2181	2184	2187
			2190	2193	2196	2199							
$DS_CNTRLC	KeyWMacr	Lib01	1171										
$DS_DSADEF	Macro	Lib01	135										
$DS_PRINTF	Unbound		258	285									
	Macro	Lib01	1296	1297	1298	1299	1309	1418	1419	1420	1421	1427	1430
			1444	1447	1527	1537	1696	1823	1840	1875	1908	2218	2219
			2220	2221	2233	2243	2253	2263	2273	2279			
$DS_QADEF	Macro	Lib01	134										
$EQULST	Macro	Lib03	134	136	137								
$ER	Comptime	338											
$MODULE	Bind	338											
AP	Builtin	1221											
ASSERT	Macro	231											
ASSERT_STRING	MacrForm	231	236										
BIT_NUMBER	Local	1373	1386.	1390.									
BIT_POS	Local	1605	1606.										
BOOLEAN	Macro	298	705										
BRANCH	Local	451	462=	465=	468=	471=	474=	477=	480=	483=	486=	489=	492=
			495=	507=	515=	523=	532=	541=	549=	557=	565=	573=	582=
			591=	599=	607=	615=	623=	631=	635=	638=	646=	649=	652=
			655=	658=	661=	668=	671=	685=	688=	697.	705=	705.	711.
BRANCH_CODE	RoutForm	403	455.										
BRANCH_VALUE	RoutForm	403	462.	465.	468.	471.	474.	477.	480.	483.	486.	489.	492.
			495.	505m	507.	513m	515.	521m	523.	524.	530m	532.	533.
			539m	541.	547m	549.	555m	557.	563m	565.	571m	573.	574.
			580m	582.	583.	589m	591.	597m	599.	605m	607.	613m	615.
			621m	623.	629m	631.	635.	638.	646.	649.	652.	655.	658.
			661.	668.	671.	685.	688.						
CALL_DSX$BRANCH	Linkage	95	103	403									
CALL_FRAME	Bind	1345	1347.	1348.	1349.	1351.							
	Bind	1406	1411.										

Symbol	Type	Defined	Referenced ...
CHECK_CASE	Macro	1680	1897 1898 1899 1900 1901
CLI$L_LAST	Macro	Lib02 1792	
CLI$L_PASS	Macro	Lib02 1748	1786
CLI$L_SUBT	Macro	Lib02 1749	1787
CLI$L_TEST	Macro	Lib02 1791	
CONDITION	MacrForm	231	235
DEBUG	Macro	280	
DEBUG_STRING	MacrForm	280	285
DIAG_AP	Local	1237	1349= 1420.
DIAG_FP	Local	1238	1350= 1420.
DIAG_PC	Local	1239	1351= 1421.
DIAG_PSW	Local	1235	1347= 1421.
DIAG_SP	Local	1240	1408= 1421.
DS$CNTRLC	ExtRout	1171	1171c
DS$GA_DS_CTRL_C_FIRST	External	171	1613= 1772=
DS$GB_WIDTH	External	175	
DS$GL_CLIBASE	External	173	1693m 1748= 1749= 1786= 1787= 1791= 1792=
DS$GL_FSTTEST	External	168	1804. 1862= 2211.
DS$GL_LSTTEST	External	169	1805. 1863= 2211.
DS$K_BBC	Literal	134	637
DS$K_BBCC	Literal	134	654
DS$K_BBCCI	Literal	134	660
DS$K_BBCS	Literal	134	648
DS$K_BBS	Literal	134	634
DS$K_BBSC	Literal	134	651
DS$K_BBSS	Literal	134	645
DS$K_BBSSI	Literal	134	657
DS$K_BCC_M	Literal	134	626
DS$K_BCS_M	Literal	134	618
DS$K_BEQLU_B	Literal	134	476
DS$K_BEQLU_M	Literal	134	544
DS$K_BEQL_B	Literal	134	473
DS$K_BEQL_M	Literal	134	536
DS$K_BERROR	Literal	134	684
DS$K_BGEQU_B	Literal	134	482
DS$K_BGEQU_M	Literal	134	560
DS$K_BGEQ_B	Literal	134	479
DS$K_BGEQ_M	Literal	134	552
DS$K_BGTRU_B	Literal	134	488
DS$K_BGTRU_M	Literal	134	577
DS$K_BGTR_B	Literal	134	485
DS$K_BGTR_M	Literal	134	568
DS$K_BLBC	Literal	134	670
DS$K_BLBS	Literal	134	667
DS$K_BLEQU_B	Literal	134	470
DS$K_BLEQU_M	Literal	134	527
DS$K_BLEQ_B	Literal	134	467
DS$K_BLEQ_M	Literal	134	518
DS$K_BLSSU_B	Literal	134	464
DS$K_BLSSU_M	Literal	134	510
DS$K_BLSS_B	Literal	134	137 461
DS$K_BLSS_M	Literal	134	502
DS$K_BNEQU_B	Literal	134	494
DS$K_BNEQU_M	Literal	134	594
DS$K_BNEQ_B	Literal	134	491

Symbol	Type	Defined	Referenced ...									
DS$K_BNEQ_M	Literal	134	586									
DS$K_BNERROR	Literal	134	137	687								
DS$K_BVC_M	Literal	134	610									
DS$K_BVS_M	Literal	134	602									
DS$K_DS_STARTING_LOCATION	Literal	137	1549									
DS$PRINTF	ExtRout	1296	1296c	1297e	1297c	1298e	1298c	1299e	1299c	1311e	1311c	1418e	1418c
			1419e	1419c	1420e	1420c	1421e	1421c	1427e	1427c	1430e	1430c	1444e
			1444c	1447e	1447c	1527e	1527c	1537e	1537c	1697e	1697c	1823e	1823c
			1840e	1840c	1875e	1875c	1908e	1908c	2218e	2218c	2219e	2219c	2220e
			2220c	2221e	2221c	2234e	2234c	2244e	2244c	2254e	2254c	2264e	2264c
			2274e	2274c	2279e	2279c							
DSA$AL_APTMAIL	Bind	135	135									
DSA$GL_APTCOM	Bind	135										
DSA$GL_DEVLEN	Bind	135										
DSA$GL_DEVNAM	Bind	135										
DSA$GL_ERRNO	Bind	135										
DSA$GL_EVENT	Bind	135										
DSA$GL_FLAGS	Bind	135	695	767	769	1172	1174	1703	1753	1755	1756	1757	1758
			1759	1760	1761	1762	1763	1764	1765	1766	1865	1867	1909
			1911										
DSA$GL_MSGPTR	Bind	135										
DSA$GL_MSGTYP	Bind	135										
DSA$GL_PASSES	Bind	135										
DSA$GL_PASSNO	Bind	135	1311									
DSA$GL_SECTNO	Bind	135	1311									
DSA$GL_SID	Bind	135										
DSA$GL_SUBTNO	Bind	135	1311									
DSA$GL_TESTNO	Bind	135	1311									
DSA$GL_UNITS	Bind	135										
DSA$V_BELL	Macro	Lib01	1755									
DSA$V_HALT	Macro	Lib01	1756									
DSA$V_IE1	Macro	Lib01	1757									
DSA$V_IE2	Macro	Lib01	1758									
DSA$V_IE3	Macro	Lib01	1759									
DSA$V_IES	Macro	Lib01	1760									
DSA$V_LOOP	Macro	Lib01	1761									
DSA$V_OPER	Macro	Lib01	1762									
DSA$V_PROMPT	Macro	Lib01	1763									
DSA$V_QA	Macro	Lib01	695	769	1174	1703	1867	1911					
DSA$V_QUICK	Macro	Lib01	1764										
DSA$V_SEARCH	Macro	Lib01	1765										
DSA$V_TRACE	Macro	Lib01	1766										
DSQA	Macro	Lib02	136										
DSQA$K_DISPAT_AFTER_INIT	Literal	136											
DSQA$K_DISPAT_BEFORE_INIT	Literal	136											
DSQA$K_DISPAT_CALLTEST	Literal	136											
DSQA$K_DISPAT_DONE_TESTS	Literal	136											
DSQAK_DISPAT_DSXENDPASS_BGN	Literal	136											
DSQAK_DISPAT_DSXENDPASS_END	Literal	136											
DSQAK_DISPAT_DSXENDPASS_MID	Literal	136	1857										
DSQA$K_DISPAT_DS_CLEANUP_BGN	Literal	136	1879										
DSQA$K_DISPAT_DS_CLEANUP_END	Literal	136	1885										
DSQA$K_DUMMY_1	Literal	136											
DSQA$K_ERROR_ERROR_BGN	Literal	136											
DSQA$K_ERROR_ERROR_END	Literal	136											

Symbol	Type	Defined	Referenced ...					
DSQA$K_KERNEL_INIT_CONTEXT	Literal	136	1872					
DSQAK_LOOP_DSXBGNSUB	Literal	136						
DSQAK_LOOP_DSXCKLOOP	Literal	136						
DSQAK_LOOP_DSXENDSUB	Literal	136						
DSQAK_PARAM_DSXGETADDRESS	Literal	136	1812					
DSQAK_PARAM_DSXGETDATA	Literal	136	1810					
DSQAK_PARAM_DSXGETLOGICAL	Literal	136	1813					
DSQAK_PARAM_DSXGETSTRING	Literal	136	1814					
DSQAK_PARAM_DSXGETVIELD	Literal	136	1811					
DSQAK_QA_DSXBRANCH	Literal	136	714					
DSQA$K_START_BEFORE_HEADER	Literal	136	1796					
DSQA$K_START_VRRESTART	Literal	136	1777					
DSQA$K_START_VRSTART	Literal	136	1738					
DSV$SHOWDEVICE	ExtRout	155	1445c					
DSV$SHOWSELECT	ExtRout	156	1448c					
DSX$ABORT	ExtRout	157	770c	1175c	1912c			
DSX$BRANCH	GlobRout	403	103f					
DS_QADEFS	Macro	Lib02	137					
DUMP_HEADER_4	Bind	1249	1299a					
DUMP_LINE_1	Bind	1250	1311a					
DUMP_LINE_10	Bind	1258	1447a					
DUMP_LINE_2	Bind	1251	1418a					
DUMP_LINE_3	Bind	1252	1419a					
DUMP_LINE_4	Bind	1253	1420a					
DUMP_LINE_5	Bind	1254	1421a					
DUMP_LINE_7	Bind	1255	1427a					
DUMP_LINE_8	Bind	1256	1430a					
DUMP_LINE_9	Bind	1257	1444a					
ERROR_ENTRY	Local	1608	1609.					
ERROR_NUMBER	RoutForm	403	712.					
	RoutForm	776						
ERROR_RECORD	Macro	267						
ERROR_STRING	MacrForm	253	256					
ES	Unbound		256	258				
	Bind	1840	1840a					
	Bind	1908	1908a					
EXE$ALONONPAGED	ExtRout	152						
EXE$DEANONPAGED	ExtRout	153						
EXPRESSION	MacrForm	298	299					
FAKE_CLI_TABLE	Local	1245	1442=	1445a	1448a			
FALSE	Literal	137	1321	1887				
FFS	Builtin	1835	1837					
FIRST_BIT_SET	Local	1832	1837=	1839.	1843.			
FLAG	Local	1602	1603.					
FORCED_ERROR_FIELDS	Unbound		268					
	Fieldset	324						
FP	Builtin	1221						
	Builtin	1501	1515.					
FP_PTR	Bind	1498	1515=	1521.	1527.	1547.	1554=	1554.
FRAME_START	Local	1229	1330a	1345.	1350.	1381.	1406.	1408.
	RoutForm	1453	1498.					
GET1ST_	Macro	Lib03	134	136	137			
GET2ND_	Unbound		134	136	137			
HEADER_LINE_1	Bind	2171	2218a					
HEADER_LINE_2	Bind	2174	2219a					

Symbol	Type	Defined	Referenced ...
HEADER_LINE_3	Bind	2177	2220a
HEADER_LINE_4	Bind	2180	2221a
HOOK_POINT	Unbound		1684
	RoutForm	1618	1697. 1738. 1777. 1796. 1810. 1811. 1812. 1813. 1814. 1857. 1872. 1879. 1885. 1897. 1898. 1899. 1900. 1901.
I	Local	1441	1442.
INVALID_CALL_FRAME	Bind	1524	1527a
JSB_ALLOC	Linkage	Lib02	152
JSB_CLI	Linkage	Lib02	155 156
JSB_DEALL	Linkage	Lib02	153
JSB_NONE	Linkage	Lib02	159 160
L$A_NAME	Literal	Lib01	1297 2279
L$A_SECNAM	Literal	Lib01	1307
L$A_TSTCNT	Literal	Lib01	2211
L$L_REV	Literal	Lib01	1297
L$L_UPDATE	Literal	Lib01	1297
LAST_PC	Local	1507	1547= 1549.
LOGIC_ERROR	Unbound		236
	Macro	253	1840 1908
LS_LINE	Bind	2192	2264a
LT_LINE	Bind	2189	2254a
MAX_FRAMES_CHECKED	Literal	1504	1531
MAX_REGS_SAVED	Literal	1226	1246 1363 1373
MODNAM	Macro	Lib02	338
MOVPSL	Builtin	1222	
MP_LINE	Bind	2186	2244a
MS_LINE	Bind	2183	2234a
NUL2ND_	Macro	Lib03	134 136 137
NUMBER_FRAMES_CHECKED	Local	1508	1516= 1531= 1531.
NUMBER_OF_REGS	Local	1242	1371= 1389= 1389. 1390. 1394= 1394. 1410.
OFF	Literal	137	769 1174 1603 1606 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1769 1867 1911
OK_LINE	Bind	2198	2279a
ON	Literal	137	1766 1773
PROBEW	Builtin	1501	1521
PSL$V_C	Macro	Lib03	515 532 565 582 623 631
PSL$V_N	Macro	Lib03	507 523 557 573
PSL$V_V	Macro	Lib03	607 615
PSL$V_Z	Macro	Lib03	524 533 541 549 574 583 591 599
QA$ABORT_QA	Routine	727	104f 1848c 1922c
QA$ADD_FORCED_ERROR	GlobRout	776	105f
QA$ADD_QA_ERROR	GlobRout	1093	106f 1843c
QA$AOB_CHECK_STATE	Unbound		1681
	Global	375	1606= 1773= 1837. 1859. 1897. 1898. 1899. 1900. 1901.
QA$AOB_FLAGS	Unbound		284
	Global	380	1319. 1321= 1603= 1695. 1821. 1874.
QA$AOB_ROUTINE_STATE	Global	371	1769= 1844= 1881= 1887=
QA$AW_OVERALL_ERROR_TABLE	Own	340	1127= 1128. 1609= 2234. 2236. 2244. 2246. 2254. 2256. 2264. 2266. 2274. 2276.
QA$CONTROL_C_HANDLER	Routine	1133	107f 1772a
QA$DUMP	Routine	1182	108f 765c
QA$FIND_USER_CALL_FRAME	GlobRout	1453	109f 1330c
QA$INIT	Routine	1560	110f 768c 1173c 1771c 1866c 1910c
QA$K_ABORT_NOW	Literal	137	
QA$K_APT_PHASE_ONE	Literal	137	

Symbol	Type	Defined	Referenced ...					
QA$K_APT_PHASE_TWO	Literal	137						
QA$K_BAD_ADDRESS	Literal	137						
QA$K_CONTROL_C_CONT	Literal	137						
QA$K_DEALLOCATION	Literal	137						
QA$K_ERROR_PHASE_ONE	Literal	137						
QA$K_ERROR_PHASE_THREE	Literal	137						
QA$K_ERROR_PHASE_TWO	Literal	137						
QA$K_FAILURE	Literal	137	1528	1538				
QA$K_FIRST_BRANCH_CODE	Literal	137	456					
QA$K_FIRST_TIME	Literal	137						
QA$K_FORCED_ERROR_RECORD_SIZE	Literal	311						
QA$K_INIT_CONTEXT_DONE	Literal	137	1319	1321				
QA$K_LAST_BRANCH_CODE	Literal	137	456					
QA$K_LAST_CHECK	Literal	313	1859					
QA$K_LOOP_ON_SUBTEST	Literal	137	1900	2264	2266			
QA$K_LOOP_ON_TEST	Literal	137	1899	2254	2256			
QA$K_LOOP_TEST_DONE	Literal	137						
QA$K_MEMORY_MANAGE	Literal	137						
QA$K_MULTIPLE_PASS	Literal	137	1898	2244	2246			
QA$K_NIL	Literal	309						
QA$K_NODEF	Literal	137	1821					
QA$K_NORMAL_START	Literal	137	1773	1897	2234	2236		
QA$K_NO_HEADER	Literal	137						
QA$K_NUMBER_OF_CHECKS	Literal	137	340	375	1605	1608	1837	1839
QA$K_NUMBER_QA_FLAGS	Literal	137	380	1602				
QA$K_NUMB_ROUTINE_STATES	Literal	137	371	1768				
QA$K_QA_DEBUG	Unbound		284					
	Literal	137	1695	1874				
QA$K_QA_DONE	Literal	137						
QA$K_RUN_BACKWARDS	Literal	137	313	1901	2274	2276		
QA$K_SECTION	Literal	137						
QA$K_SUCCESS	Literal	137	1551					
QA$LOOP_ON_SUBTEST	ExtRout	149	1900c					
QA$LOOP_ON_TEST	ExtRout	148	1899c					
QA$L_SAVED_DSA_FLAGS	Global	368	767.	1172.	1753=	1865.	1909.	
QA$MAIN	GlobRout	1618	111f	714c				
QA$MULTIPLE_PASS	ExtRout	147	1898c					
QA$NORMAL_START	ExtRout	146	1897c					
QA$PRINT_FORCED_ERRORS	GlobRout	1927	112f	1861c				
QA$PRINT_OVERALL_ERRORS	Routine	2127	113f	766c	1864c			
QA$RUN_BACKWARDS	ExtRout	150	1901c					
QA$T_HEADER_1	External	186	1296a					
QA$T_HEADER_2	External	187	1297a					
QA$T_HEADER_3	External	188	1298a					
QA$T_OPERATOR_REQUEST_MESSAGE	External	190	1823a					
QA$VA_NEXT_ERROR	Field	329						
QA$VL_ERROR_COUNT	Field	327						
QA$VL_SUBTEST_NUMBER	Field	328						
QA$VW_ERROR_NUMBER	Field	325						
QA$VW_TEST_NUMBER	Field	326						
QA$V_CHECK_DONE	Literal	137						
QA$V_DOING_CLEANUP	Literal	137	1881	1887				
QA$V_ERROR_FOUND	Literal	137	1844					
QA$V_INIT_DONE	Literal	137						
QA$W_BRANCH	Global	358	711=	721.				

Symbol	Type	Defined	Referenced ...
QA$W_ERROR_NUMBER	Global	361	712=
QA$W_MULTIPLEPASS	External	184	2244.
QA$W_SAVE_LAST	Global	366	1751= 1792. 1805= 1863.
QA$W_SAVE_TEST	Global	364	1750= 1789. 1791. 1798. 1804= 1862.
QA$W_SUBTESTLOOPS	External	183	2264.
QA$W_TESTLOOPS	External	182	2254.
RB_LINE	Bind	2195	2274a
REGISTER_0	RoutForm	403	453.
REGISTER_MASK	Local	1236	1348= 1384m 1386.
REG_NOT_SAVED	Literal	1225	1364
REG_NUMBER	Local	1363	1364.
REG_VECTOR	Local	1246	1364= 1390= 1418. 1419. 1420.
ROUTINE_CONSTANT	RoutForm	1093	1127. 1128.
ROUTINE_NAME	MacrForm	1680	1682 1684
SAVE_REGISTER_0	Local	450	453= 697= 698. 721= 722.
SEC_NAME_TABLE	Bind	1307	1311.
SF$L_SAVE_AP	Macro	Lib03	1349
SF$L_SAVE_PC	Macro	Lib03	1351
SF$V_SAVE_MASK	Macro	Lib03	1348
SF$V_STACKOFFS	Macro	Lib03	1411
SF$W_SAVE_PSW	Macro	Lib03	1347
SP	Builtin	1221	
	Builtin	1690	
START_REGISTERS	Bind	1381	1390.
STATE	Local	1768	1769.
TOO_MANY_FRAMES	Bind	1534	1537a
TRUE	Literal	137	1844 1881
VRSHOWEF	ExtRout	160	1431c
VRSHOWFLG	ExtRout	159	1428c

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]QAMAIN.B32;1
5	Module QAMAIN	
121	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
124	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
127	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
2286	Eludom QAMAIN	

```
        KEY TO REFERENCE TYPE FLAGS

            .   Fetch
            =   Store
            c   Routine call
            a   Address use
            @   Indirect use
            e   External, external routine, or external literal declaration
            f   Forward or forward routine declaration
            h   Condition handler enabling
            m   Map declaration
            u   Undeclare declaration


;                           Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                          940        23         2        96          00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                          675        11         1        47          00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2           20450        12         0      1101          00:00.8




;                                   COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:QAMAIN.B32

; Size:          1595 code + 1212 data bytes
; Run Time:         00:08.9
; Elapsed Time:     00:12.7
; Lines/CPU Min:    15498
; Lexemes/CPU-Min: 99410
; Memory Used:  334 pages
; Compilation Complete
```

Table of contents

(1)	146	Libraries and Equated Symbols
(1)	294	External Symbols
(1)	409	EXE$QIO QIO DISPATCHING ROUTINE
(2)	514	QIO$INITIALIZE Initialize QIO database
(4)	571	QIO$CLEANUP Remove old QIO database
(5)	670	QIO$ADDUNIT Add a unit to the QIO database
(7)	929	INI_MBADP BUILD ADP AND INITIALIZE MBA
(9)	1026	INI_UBADP BUILD ADP AND INITIALIZE UBA
(10)	1083	INI_DRADP BUILD ADP AND INITIALIZE DRA
(12)	1180	MAXIMIZE ACCESS MODE
(13)	1206	BUG$CHECK Bugcheck notification routine
(14)	1241	QIO$LOAD_DB Insert unit into data base
(16)	1523	DB_ERROR ERROR LOADING DATABASE
(16)	1595	I/O DATABASE INTERLOCKS
(17)	1603	IOGEN$CNTRL_INI Initialize a controller
(18)	1657	MEMORY ALLOCATION SUBROUTINES
(19)	1699	ADPLINK Link adapter to end of ADP list
(19)	1733	Exe$PwrTimChk - Check reasonable interval since power recovery ;[14]

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element QIO.MAR
0000     2 ; *8    29-APR-1987 16:00:57 MARTIN "MODIFIED FOR PSTAR"
0000     3 ; *7    26-MAR-1987 14:19:23 MARTIN "REF TO PSTAR"
0000     4 ; *6    17-MAR-1987 10:46:56 MARTIN "FIX"
0000     5 ; *5    29-MAY-1986 08:31:14 MUSE "<no reason given>"
0000     6 ; *4    28-MAY-1986 13:34:19 MUSE "<no reason given>"
0000     7 ; *3    28-MAY-1986 12:38:03 MUSE "<no reason given>"
0000     8 ; *2    13-MAY-1986 14:02:43 SALEM "DONE"
0000     9 ; *1     6-FEB-1986 15:22:06 DS ""
0000    10 ; DEC/CMS REPLACEMENT HISTORY, Element QIO.MAR
0000    11         .TITLE  QIO
0000    12         .IDENT  /09-28/                                                    ;G:6
0000    13         .NoShow Conditionals
0000    14         .DSABL  GBL
0000    15
0000    16 ;
0000    17 ; Copyright (c) 1977, 1982, 1984,1985,1987                               ;G:6
0000    18 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    19 ;
0000    20 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    21 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    22 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    23 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    24 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    25 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    26 ; REMAIN IN DEC.
0000    27 ;
0000    28 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    29 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    30 ; CORPORATION.
0000    31 ;
0000    32 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    33 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    34
0000    35 ;++
0000    36 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    37 ;
0000    38 ; ABSTRACT:
0000    39 ;
0000    40 ; ENVIRONMENT:
0000    41 ;
0000    42 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
0000    43 ; MODIFIED BY:
0000    44 ;                N. HOWGATE      20-FEB-78       VERSION 02 (ESSAA-4.00).
0000    45 ;       01       QIO SUPPORT
0000    46 ;                TOM SOUTTER     22-FEB-78
0000    47 ;       02       ADDED CALL TO VMS/QIO HANDLER
0000    48 ;                R. RIGGS        1-OCT-78        VERSION 3 (ESSAA-5.00)
0000    49 ;       03       INCLUDED QIO DATABASE INITIALIZATION ROUTINES
0000    50 ;                N.Howgate       26-Jun-79       Version 4 (ESSAA-5.01)
0000    51 ;       04       Add support for 11/750
0000    52 ;                Roger Riggs     03-NOV-1979
0000    53 ;       05       Added code to load drivers if not found in
0000    54 ;                IOC$GL_DPTLIST
0000    55 ;       06       Added fudge to generate 2 vectors for LPA11K
0000    56 ;                Roger Riggs, 5-Feb-1980, Version 5.2
0000    57 ;       07       Revamped setup of ACF block, Added code to maintain reference
```

```
0000  ;                 count in DPT headers, added code to handle headers on drivers
0000  ;                 header contains the number of vectors for unibus  devices
0000  ;                 Removes need for fudge in edit 06.
0000  ;--
0000  ;
0000  ;         08      John Ciukaj     5-May-80        Version 5.4
0000  ;
0000  ;                   Added Dummy return for QIO related calls
0000  ;
0000  ;                 Dave Butenhof   18-jun-1980,    version 5.5
0000  ;         09      Remove all dummy returns from this module, and move to
0000  ;                 new module VMSDUMMY.
0000  ;                 Roger Riggs, 10-Aug-1980
0000  ;         10      Use QIO driver prefix from controller instead of device
0000  ;                 this allows for terminals to be connected to more than
0000  ;                 one type of Async controller.  The first prefix
0000  ;                 encountered while linking back through the P-tables is
0000  ;                 used.
0000  ;
0000  ;         11      - Jack Stansbury, 22-Oct-1981, Version 6.-
0000  ;                 Fixed truncation errors. Also added .LIBRARY statements
0000  ;                 for $DS and $DIAG. Also changed .PSECT SEP to WORK, CODE,
0000  ;                 and DATA where appropriate.
0000  ;
0000  ;         12      - Jack Stansbury, 12-Nov-1981, Version 6.-
0000  ;                 Changed the HALTD to HALT.
0000  ;
0000  ;         13      Marion Baggett, 9-Apr-1982, Version 6.7
0000  ;                 Added .NoShow Conditionals after .IDENT statement.
0000  ;
0000  ;         14      Jack Stansbury, July 1, 1982, Version 6.8
0000  ;                 Added the 'real' Exe$PwrTimChk routine.
0000  ;
0000  ;         15      Marion Baggett, 24-Sept-1982, Version 6.9
0000  ;                 Changed the CASE instruction since WRITEVBLK and READVBLK
0000  ;                 are to far away and a JMP was need.
0000  ;
0000  ;         16      Richard E. Muse, 9 January, 1984, Version 6.14
0000  ;                 Added $PRDEF macro definition - VMS Version 4 change.
0000  ;
0000  ;         17      Domenic Andella, 9-Apr-1984, Version 6.14
0000  ;                 Added code in FILL_ACF to handle case where
0000  ;                 an SBIA is attached, and it is necessary to
0000  ;                 'backtrack' another level.
0000  ;
0000  ;         18      Bob Bergazzi    24-Apr-1984     Version 7.0
0000  ;                         - Changed FILL_ACF to set up R3 with the CSR
0000  ;                           address for the subsequent call to FIND_ADP.
0000  ;                         - Changed FIND_ADP routine to compare ADP$L_CSR
0000  ;                           instead of ADP$W_TR. This accomodates the VENUS
0000  ;                           cpu with multiple SBIs.
0000  ;                         - Changed INI_DRADP to add 100 or 300 to SCB_BASE
0000  ;                           depending on which SBI the adapter is connected to.
0000  ;                         - Removed UBINT... code from INI_UBADP since it
0000  ;                           only needs the size of that code (UBINTSZ), which
0000  ;                           is defined in ONLY8600.
0000  ;
```

```
0000    115 ;       19      Ron Parker      1-NOV-1984
0000    116 ;                       - Fixed truncation errors
0000    117 ;
0000    118 ;       20      Bob Bergazzi    Nov. 26, 1984   Version 8.0
0000    119 ;               Added DWBUA to list of known adapter types.
0000    120 ;
0000    121 ;       21      Ted Salem       Feb. 6, 1985    Version 8.1
0000    122 ;               Added code to in order to read and write
0000    123 ;               accross the NI.
0000    124 ;
0000    125 ;       22      Ted Salem       March 19, 1985  Version 8.2
0000    126 ;               Changed EXE$QIO to do a proper return to MP$INTERLOCK.
0000    127 ;
0000    128 ;       24      R.E.Muse        17 June 1985
0000    129 ;               added code to read the nautilus TODR register
0000    130 ;
0000    131 ;       25      J. Baranski     28-Jan-1986
0000    132 ;               Added .EXTERN DS$GK_SID to fix undefined DS$GK_SID.
0000    133 ;                                                                   ;G:2
0000    134 ;       26      Ted Salem       15-May-1986                         ;G:2
0000    135 ;               Made change #18 to be defined as weak               ;G:2
0000    136 ;                                                                   ;G:4
0000    137 ;       27      Richard E. Muse 28-May-1986                         ;G:4
0000    138 ;               fixed call to ds$load. use dummy buf addr to fool   ;G:4
0000    139 ;               error checking in ds$load so we can get file size.  ;G:4
0000    140 ;                                                                   ;G:6
0000    141 ;       28      Ingrid Martin   13-Mar-1987                         ;G:6
0000    142 ;               Added code so that RRR can read the TODR the same   ;G:7
0000    143 ;               way as Nautilus.                                    ;G:6
0000    144 ;-
```

```
0000    146          .Sbttl  Libraries and Equated Symbols
0000    147
0000    148 ;
0000    149 ; INCLUDE FILES:
0000    150 ;
0000    151          .Library        /SYS$LIBRARY:LIB/
0000    152          .LIBRARY        /$DS/                   ;                       [11]
0000    153          .LIBRARY        /$DIAG/                 ;                       [11]
0000    154
0000    155 ;
0000    156 ; MACROS:
0000    157 ;
0000    158          .MACRO  $PRDEF,$GBL                             ; [16]
0000    159
0000    160
0000    161          $DEFINI PR,$GBL
0000    162
0000    163
0000    164  $EQU   PR$_KSP 0
0000    165  $EQU   PR$_ESP 1
0000    166  $EQU   PR$_SSP 2
0000    167  $EQU   PR$_USP 3
0000    168  $EQU   PR$_ISP 4
0000    169  $EQU   PR$_P0BR        8
0000    170  $EQU   PR$_P0LR        9
0000    171  $EQU   PR$_P1BR        10
0000    172  $EQU   PR$_P1LR        11
0000    173  $EQU   PR$_SBR 12
0000    174  $EQU   PR$_SLR 13
0000    175  $EQU   PR$_PCBB        16
0000    176  $EQU   PR$_SCBB        17
0000    177  $EQU   PR$_IPL 18
0000    178  $EQU   PR$_ASTLVL      19
0000    179  $EQU   PR$_SIRR        20
0000    180  $EQU   PR$_SISR        21
0000    181  $EQU   PR$_ICCS        24
0000    182  $EQU   PR$_NICR        25
0000    183  $EQU   PR$_ICR 26
0000    184  $EQU   PR$_TODR        27
0000    185  $EQU   PR$_RXCS        32
0000    186  $EQU   PR$_RXDB        33
0000    187  $EQU   PR$_TXCS        34
0000    188  $EQU   PR$_TXDB        35
0000    189  $EQU   PR$_ACCS        40
0000    190  $EQU   PR$_ACCR        41
0000    191  $EQU   PR$_MAPEN       56
0000    192  $EQU   PR$_TBIA        57
0000    193  $EQU   PR$_TBIS        58
0000    194  $EQU   PR$_PME 61
0000    195  $EQU   PR$_SID 62
0000    196  $EQU   PR$_TBCHK       63
0000    197  $EQU   PR$V_SID_SN     0
0000    198  $EQU   PR$S_SID_SN     12
0000    199  $EQU   PR$V_SID_PL     12
0000    200  $EQU   PR$S_SID_PL     3
0000    201  $EQU   PR$V_SID_ECO    15
0000    202  $EQU   PR$S_SID_ECO    9
```

```
                     0000   203  $EQU   PR$V_SID_TYPE   24
                     0000   204  $EQU   PR$S_SID_TYPE   8
                     0000   205  $EQU   PR$_SID_TYP780  1
                     0000   206  $EQU   PR$_SID_TYP750  2
                     0000   207  $EQU   PR$_SID_TYP730  3
                     0000   208  $EQU   PR$_SID_TYP7VV  4
                     0000   209  $EQU   PR$_SID_TYPMAX  4
                     0000   210  $EQU   PR$_WCSA        44
                     0000   211  $EQU   PR$_WCSD        45
                     0000   212  $EQU   PR$_SBIFS       48
                     0000   213  $EQU   PR$_SBIS        49
                     0000   214  $EQU   PR$_SBISC       50
                     0000   215  $EQU   PR$_SBIMT       51
                     0000   216  $EQU   PR$_SBIER       52
                     0000   217  $EQU   PR$_SBITA       53
                     0000   218  $EQU   PR$_SBIQC       54
                     0000   219  $EQU   PR$_CMIERR      23
                     0000   220  $EQU   PR$_CSRS        28
                     0000   221  $EQU   PR$_CSRD        29
                     0000   222  $EQU   PR$_CSTS        30
                     0000   223  $EQU   PR$_CSTD        31
                     0000   224  $EQU   PR$_TBDR        36
                     0000   225  $EQU   PR$_CADR        37
                     0000   226  $EQU   PR$_MCESR       38
                     0000   227  $EQU   PR$_CAER        39
                     0000   228  $EQU   PR$_UBRESET     55
                     0000   229  $EQU   PR$_PAMACC      64
                     0000   230  $EQU   PR$_PAMLOC      65
                     0000   231  $EQU   PR$_CSWP        66
                     0000   232  $EQU   PR$_CRBT        67
                     0000   233  $EQU   PR$_MCTL1       68
                     0000   234  $EQU   PR$_MCTL2       69
                     0000   235  $EQU   PR$_MGEN        70
                     0000   236  $EQU   PR$_MTBER       71
                     0000   237  $EQU   PR$_MEAR        72
                     0000   238  $EQU   PR$_MDCR1       73
                     0000   239  $EQU   PR$_MEDR        74
                     0000   240  $EQU   PR$_MECCR       75
                     0000   241  $EQU   PR$_SID_TYP8NN  6                                        ;G:8
                     0000   242  $EQU   PR$_SID_TYP8RR  17          ; RRR sid type ^X11 [28]   ;G:8
                     0000   243         $DEFEND PR,$GBL,DEF
                     0000   244
                     0000   245         .ENDM   $PRDEF
                     0000   246         $PRDEF                              ; CALL PRDEF MACRO
                     0000               .SAVE   LOCAL_BLOCK
                     0000               .RESTORE
                     0000   247
                     0000   248 ;
                     0000   249 ; EQUATED SYMBOLS:
                     0000   250 ;
0000000A             0000   251         LF = 10                             ; [21]
0000000D             0000   252         CR = 13                             ; [21]
00000002             0000   253         CRLF_LENGTH = 2                     ; [21]
                     0000   254
                     0000   255
                     0000   256         .NLIST  MEB
                     0000   257         $ACFDEF
```

```
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .IIF    NB,ACF$L_ADAPTER,       ACF$L_ADAPTER:
00000004             0000                        .BLKL   1
                     0004                .IIF    NB,ACF$L_CONFIGREG,     ACF$L_CONFIGREG:
00000008             0004                        .BLKL   1
                     0008                .IIF    NB,ACF$W_AVECTOR,       ACF$W_AVECTOR:
0000000A             0008                        .BLKW   1
                     000A                .IIF    NB,ACF$B_AUNIT, ACF$B_AUNIT:
0000000B             000A                        .BLKB   1
                     000B                .IIF    NB,ACF$B_AFLAG, ACF$B_AFLAG:
0000000C             000B                        .BLKB   1
                     000C                .IIF    NB,ACF$L_CONTRLREG,     ACF$L_CONTRLREG:
00000010             000C                        .BLKL   1
                     0010                .IIF    NB,ACF$W_CVECTOR,       ACF$W_CVECTOR:
00000012             0010                        .BLKW   1
                     0012                .IIF    NB,ACF$B_CUNIT, ACF$B_CUNIT:
00000013             0012                        .BLKB   1
                     0013                .IIF    NB,ACF$B_CNUMVEC,       ACF$B_CNUMVEC:
00000014             0013                        .BLKB   1
                     0014                .IIF    NB,ACF$L_DEVNAME,       ACF$L_DEVNAME:
00000018             0014                        .BLKL   1
                     0018                .IIF    NB,ACF$L_DRVNAME,       ACF$L_DRVNAME:
0000001C             0018                        .BLKL   1
                     001C                .IIF    NB,ACF$W_MAXUNITS,      ACF$W_MAXUNITS:
0000001E             001C                        .BLKW   1
00000020             001E                        .BLKW   1
                     0020                .IIF    NB,ACF$C_LENGTH,        ACF$C_LENGTH:
                     0020                .IIF    NB,ACF$K_LENGTH,        ACF$K_LENGTH:
                     0000                .RESTORE
                     0000    258         $ADPDEF
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             0020                .=0
                     0000                .IIF    NB,ADP$L_CSR,   ADP$L_CSR:
00000004             0000                        .BLKL   1
                     0004                .IIF    NB,ADP$L_LINK,  ADP$L_LINK:
00000008             0004                        .BLKL   1
                     0008                .IIF    NB,ADP$W_SIZE,  ADP$W_SIZE:
0000000A             0008                        .BLKW   1
                     000A                .IIF    NB,ADP$B_TYPE,  ADP$B_TYPE:
0000000B             000A                        .BLKB   1
                     000B                .IIF    NB,ADP$B_NUMBER,        ADP$B_NUMBER:
0000000C             000B                        .BLKB   1
                     000C                .IIF    NB,ADP$W_TR,    ADP$W_TR:
0000000E             000C                        .BLKW   1
                     000E                .IIF    NB,ADP$W_ADPTYPE,       ADP$W_ADPTYPE:
00000010             000E                        .BLKW   1
                     0010                .IIF    NB,ADP$L_VECTOR,        ADP$L_VECTOR:
                     0010                .IIF    NB,ADP$L_CRB,   ADP$L_CRB:
00000014             0010                        .BLKL   1
                     0014                .IIF    NB,ADP$C_MBAADPLEN,     ADP$C_MBAADPLEN:
                     0014                .IIF    NB,ADP$K_MBAADPLEN,     ADP$K_MBAADPLEN:
                     0014                .IIF    NB,ADP$C_DRADPLEN,      ADP$C_DRADPLEN:
                     0014                .IIF    NB,ADP$K_DRADPLEN,      ADP$K_DRADPLEN:
                     0014                .IIF    NB,ADP$L_DPQFL, ADP$L_DPQFL:
                     0014                .IIF    NB,ADP$L_PRQQFL,        ADP$L_PRQQFL:
00000018             0014                        .BLKL   1
```

```
                     0018              .IIF    NB,ADP$L_DPQBL, ADP$L_DPQBL:
                     0018              .IIF    NB,ADP$L_PRQQBL,        ADP$L_PRQQBL:
0000001C             0018                      .BLKL   1
                     001C              .IIF    NB,ADP$L_MRQFL, ADP$L_MRQFL:
                     001C              .IIF    NB,ADP$L_SHB,   ADP$L_SHB:
00000020             001C                      .BLKL   1
                     0020              .IIF    NB,ADP$L_MRQBL, ADP$L_MRQBL:
                     0020              .IIF    NB,ADP$B_PORT,  ADP$B_PORT:
00000021             0020                      .BLKB   1
00000024             0021                      .BLKB   3
                     0024              .IIF    NB,ADP$W_DPBITMAP,      ADP$W_DPBITMAP:
00000026             0024                      .BLKW   1
                     0026              .IIF    NB,ADP$W_MRBITMAP,      ADP$W_MRBITMAP:
00000064             0026                      .BLKW   31
                     0064              .IIF    NB,ADP$L_INTD,  ADP$L_INTD:
00000070             0064                      .BLKL   3
                     0070              .IIF    NB,ADP$C_MPMADPLEN,     ADP$C_MPMADPLEN:
                     0070              .IIF    NB,ADP$K_MPMADPLEN,     ADP$K_MPMADPLEN:
                     0070              .IIF    NB,ADP$C_UBAADPLEN,     ADP$C_UBAADPLEN:
                     0070              .IIF    NB,ADP$K_UBAADPLEN,     ADP$K_UBAADPLEN:
                     0000              .RESTORE
                     0000    259       $ALLOCDEF
                     0000    260       $CCBDEF
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             0070              .=0
                     0000              .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
00000004             0000                      .BLKL   1
                     0004              .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
00000008             0004                      .BLKL   1
                     0008              .IIF    NB,CCB$B_STS,   CCB$B_STS:
00000009             0008                      .BLKB   1
                     0009              .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
0000000A             0009                      .BLKB   1
                     000A              .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
0000000C             000A                      .BLKW   1
                     000C              .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
00000010             000C                      .BLKL   1
                     0010              .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                     0010              .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                     0000              .RESTORE
                     0000    261       $CRBDEF
                     0000              .SAVE   LOCAL_BLOCK
                     0000              .PSECT  $ABS$,ABS
00000000             0070              .=0
                     0000              .IIF    NB,CRB$L_WQFL,  CRB$L_WQFL:
00000004             0000                      .BLKL   1
                     0004              .IIF    NB,CRB$L_WQBL,  CRB$L_WQBL:
00000008             0004                      .BLKL   1
                     0008              .IIF    NB,CRB$W_SIZE,  CRB$W_SIZE:
0000000A             0008                      .BLKW   1
                     000A              .IIF    NB,CRB$B_TYPE,  CRB$B_TYPE:
0000000B             000A                      .BLKB   1
0000000C             000B                      .BLKB   1
                     000C              .IIF    NB,CRB$W_REFC,  CRB$W_REFC:
0000000E             000C                      .BLKW   1
                     000E              .IIF    NB,CRB$B_MASK,  CRB$B_MASK:
```

```
                    0000000F  000E                            .BLKB   1
                    00000010  000F                            .BLKB   1
                              0010              .IIF    NB,CRB$L_LINK,  CRB$L_LINK:
                    00000014  0010                            .BLKL   1
                              0014              .IIF    NB,CRB$L_INTD,  CRB$L_INTD:
                    00000038  0014                            .BLKL   9
                              0038              .IIF    NB,CRB$C_LENGTH,        CRB$C_LENGTH:
                              0038              .IIF    NB,CRB$K_LENGTH,        CRB$K_LENGTH:
                              0038              .IIF    NB,CRB$L_INTD2, CRB$L_INTD2:
                    0000005C  0038                            .BLKL   9
                              0000              .RESTORE
                              0000    262       $DCDEF
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  0070              .=0
                              0000              .RESTORE
                              0000    263       $DDBDEF
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  0070              .=0
                              0000              .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
                    00000004  0000                            .BLKL   1
                              0004              .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
                    00000008  0004                            .BLKL   1
                              0008              .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
                    0000000A  0008                            .BLKW   1
                              000A              .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
                    0000000B  000A                            .BLKB   1
                    0000000C  000B                            .BLKB   1
                              000C              .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
                    00000010  000C                            .BLKL   1
                              0010              .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
                    00000013  0010                            .BLKB   3
                              0013              .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
                    00000014  0013                            .BLKB   1
                              0014              .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
                    00000015  0014                            .BLKB   1
                    00000024  0015                            .BLKB   15
                              0024              .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
                    00000025  0024                            .BLKB   1
                    00000034  0025                            .BLKB   15
                              0034              .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                              0034              .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                              0000              .RESTORE
                              0000    264       $DDTDEF
                              0000              .SAVE   LOCAL_BLOCK
                              0000              .PSECT  $ABS$,ABS
                    00000000  0070              .=0
                              0000              .IIF    NB,DDT$L_START, DDT$L_START:
                    00000004  0000                            .BLKL   1
                              0004              .IIF    NB,DDT$L_UNSOLINT,      DDT$L_UNSOLINT:
                    00000008  0004                            .BLKL   1
                              0008              .IIF    NB,DDT$L_FDT,   DDT$L_FDT:
                    0000000C  0008                            .BLKL   1
                              000C              .IIF    NB,DDT$L_CANCEL,        DDT$L_CANCEL:
                    00000010  000C                            .BLKL   1
                              0010              .IIF    NB,DDT$L_REGDUMP,       DDT$L_REGDUMP:
```

```
00000014  0010                        .BLKL   1
          0014              .IIF    NB,DDT$W_DIAGBUF,       DDT$W_DIAGBUF:
00000016  0014                        .BLKW   1
          0016              .IIF    NB,DDT$W_ERRORBUF,      DDT$W_ERRORBUF:
00000018  0016                        .BLKW   1
          0018              .IIF    NB,DDT$L_UNITINIT,      DDT$L_UNITINIT:
0000001C  0018                        .BLKL   1
          001C              .IIF    NB,DDT$L_ALTSTART,      DDT$L_ALTSTART:
00000020  001C                        .BLKL   1
          0000              .RESTORE
          0000    265       $DPTDEF
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .IIF    NB,DPT$L_FLINK, DPT$L_FLINK:
00000004  0000                        .BLKL   1
          0004              .IIF    NB,DPT$L_BLINK, DPT$L_BLINK:
00000008  0004                        .BLKL   1
          0008              .IIF    NB,DPT$W_SIZE,  DPT$W_SIZE:
0000000A  0008                        .BLKW   1
          000A              .IIF    NB,DPT$B_TYPE,  DPT$B_TYPE:
0000000B  000A                        .BLKB   1
          000B              .IIF    NB,DPT$B_REFC,  DPT$B_REFC:
0000000C  000B                        .BLKB   1
          000C              .IIF    NB,DPT$B_ADPTYPE,       DPT$B_ADPTYPE:
0000000D  000C                        .BLKB   1
          000D              .IIF    NB,DPT$B_FLAGS, DPT$B_FLAGS:
0000000E  000D                        .BLKB   1
          000E              .IIF    NB,DPT$W_UCBSIZE,       DPT$W_UCBSIZE:
00000010  000E                        .BLKW   1
          0010              .IIF    NB,DPT$W_INITTAB,       DPT$W_INITTAB:
00000012  0010                        .BLKW   1
          0012              .IIF    NB,DPT$W_REINITTAB,     DPT$W_REINITTAB:
00000014  0012                        .BLKW   1
          0014              .IIF    NB,DPT$W_UNLOAD,        DPT$W_UNLOAD:
00000016  0014                        .BLKW   1
          0016              .IIF    NB,DPT$W_MAXUNITS,      DPT$W_MAXUNITS:
00000018  0016                        .BLKW   1
          0018              .IIF    NB,DPT$W_VERSION,       DPT$W_VERSION:
0000001A  0018                        .BLKW   1
0000001E  001A                        .BLKL   1
          001E              .IIF    NB,DPT$T_NAME,  DPT$T_NAME:
0000002A  001E                        .BLKB   12
          002A              .IIF    NB,DPT$C_LENGTH,        DPT$C_LENGTH:
          002A              .IIF    NB,DPT$K_LENGTH,        DPT$K_LENGTH:
          0000              .RESTORE
          0000    266       $DYNDEF GLOBAL
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .RESTORE
          0000    267       $IDBDEF
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  0070              .=0
          0000              .IIF    NB,IDB$L_CSR,   IDB$L_CSR:
00000004  0000                        .BLKL   1
```

```
                        0004                    .IIF    NB,IDB$L_OWNER, IDB$L_OWNER:
                00000008 0004                           .BLKL   1
                        0008                    .IIF    NB,IDB$W_SIZE,  IDB$W_SIZE:
                0000000A 0008                           .BLKW   1
                        000A                    .IIF    NB,IDB$B_TYPE,  IDB$B_TYPE:
                0000000B 000A                           .BLKB   1
                0000000C 000B                           .BLKB   1
                        000C                    .IIF    NB,IDB$W_UNITS, IDB$W_UNITS:
                0000000E 000C                           .BLKW   1
                00000010 000E                           .BLKW   1
                        0010                    .IIF    NB,IDB$L_ADP,   IDB$L_ADP:
                00000014 0010                           .BLKL   1
                        0014                    .IIF    NB,IDB$L_UCBLST,        IDB$L_UCBLST:
                00000034 0014                           .BLKL   8
                        0034                    .IIF    NB,IDB$C_LENGTH,        IDB$C_LENGTH:
                        0034                    .IIF    NB,IDB$K_LENGTH,        IDB$K_LENGTH:
                        0000                    .RESTORE
                        0000    268             $IODEF
                        0000                    .SAVE   LOCAL_BLOCK
                        0000                    .PSECT  $ABS$,ABS
                00000000 0070                   .=0
                        0000                    .RESTORE
                        0000    269             $PRDEF
                        0000    270             $PSLDEF
                        0000                    .SAVE   LOCAL_BLOCK
                        0000                    .PSECT  $ABS$,ABS
                00000000 0070                   .=0
                        0000                    .RESTORE
                        0000    271             $QIODEF
                        0000    272             $SYSGMSGDEF
                        0000                    .SAVE   LOCAL_BLOCK
                        0000                    .PSECT  $ABS$,ABS
                00000000 0070                   .=0
                        0000                    .RESTORE
                        0000    273             $UBADEF
                        0000                    .SAVE   LOCAL_BLOCK
                        0000                    .PSECT  $ABS$,ABS
                00000000 0070                   .=0
                        0000                    .RESTORE
                        0000    274             $UCBDEF
                        0000                    .SAVE   LOCAL_BLOCK
                        0000                    .PSECT  $ABS$,ABS
                00000000 0070                   .=0
                        0000                    .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                        0000                    .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                00000004 0000                           .BLKL   1
                        0004                    .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                        0004                    .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                00000008 0004                           .BLKL   1
                        0008                    .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                0000000A 0008                           .BLKW   1
                        000A                    .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                0000000B 000A                           .BLKB   1
                        000B                    .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                0000000C 000B                           .BLKB   1
                        000C                    .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                        000C                    .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
```

```
                    000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
          0000000D  000C                        .BLKB   1
          00000010  000D                        .BLKB   3
                    0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
          00000014  0010                        .BLKL   1
                    0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
          00000016  0014                        .BLKW   1
                    0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
          00000018  0016                        .BLKW   1
                    0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
          0000001A  0018                        .BLKW   1
                    001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
          0000001C  001A                        .BLKW   1
                    001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
          00000020  001C                        .BLKL   1
                    0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
          00000024  0020                        .BLKL   1
                    0024                .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
          00000028  0024                        .BLKL   1
                    0028                .IIF    NB,UCB$L_PID,   UCB$L_PID:
          0000002C  0028                        .BLKL   1
                    002C                .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
          00000030  002C                        .BLKL   1
                    0030                .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
          00000034  0030                        .BLKL   1
                    0034                .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
          00000038  0034                        .BLKL   1
                    0038                .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
          00000039  0038                        .BLKB   1
                    0039                .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
          0000003A  0039                        .BLKB   1
                    003A                .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
          0000003C  003A                        .BLKW   1
                    003C                .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C                .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C                .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
          0000003D  003C                        .BLKB   1
                    003D                .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                    003D                .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
          0000003E  003D                        .BLKB   1
                    003E                .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                    003E                .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
          0000003F  003E                        .BLKB   1
                    003F                .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
          00000040  003F                        .BLKB   1
                    0040                .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
          00000044  0040                        .BLKL   1
                    0044                .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
          00000048  0044                        .BLKL   1
                    0048                .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
          0000004A  0048                        .BLKW   1
                    004A                .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
```

```
                    004A            .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                              .BLKB   1
                    004B            .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                              .BLKB   1
                    004C            .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                              .BLKL   1
                    0050            .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                              .BLKW   1
                    0052            .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                    0052            .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                              .BLKB   1
                    0053            .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                              .BLKB   1
                    0054            .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                              .BLKL   1
                    0058            .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                              .BLKW   1
                    005A            .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                              .BLKW   1
                    005C            .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                    005C            .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                              .BLKL   1
                    0060            .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                              .BLKL   1
                    0064            .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                    0064            .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                              .BLKL   1
                    0068            .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                              .BLKL   1
                    006C            .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                              .BLKW   1
                    006E            .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                              .BLKW   1
                    0070            .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                              .BLKB   1
                    0071            .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                              .BLKB   1
                    0072            .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                              .BLKW   1
                    0074            .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                    0074            .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074                      . = 0
                    0000            .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                              .BLKW   1
00000074  0002                      . = 116
                    0074            .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                              .BLKL   1
                    0078            .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                              .BLKL   1
                    007C            .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                              .BLKL   1
                    0080            .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                              .BLKL   1
                    0084            .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                              .BLKL   1
                    0088            .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                              .BLKL   1
```

```
                        008C                    .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
                  00000090  008C                            .BLKL   1
                        0090                    .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                        0090                    .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
                  00000074  0090                . = 116
                        0074                    .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
                  00000075  0074                            .BLKB   1
                        0075                    .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
                  00000076  0075                            .BLKB   1
                        0076                    .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
                  00000077  0076                            .BLKB   1
                        0077                    .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
                  00000078  0077                            .BLKB   1
                        0078                    .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
                  0000007C  0078                            .BLKL   1
                  0000007E  007C                            .BLKW   1
                        007E                    .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
                  00000080  007E                            .BLKW   1
                        0080                    .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
                  00000084  0080                            .BLKL   1
                  00000080  0084                . = 128
                  00000084  0080                            .BLKL   1
                        0084                    .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
                  00000088  0084                            .BLKL   1
                        0088                    .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
                  0000008A  0088                            .BLKW   1
                        008A                    .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
                  0000008C  008A                            .BLKW   1
                        008C                    .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                        008C                    .IIF    NB,UCB$W_DA,    UCB$W_DA:
                  0000008E  008C                            .BLKW   1
                        008E                    .IIF    NB,UCB$W_DC,    UCB$W_DC:
                  00000090  008E                            .BLKW   1
                        0090                    .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
                  00000092  0090                            .BLKW   1
                        0092                    .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
                  00000094  0092                            .BLKW   1
                        0094                    .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
                  00000095  0094                            .BLKB   1
                        0095                    .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
                  00000096  0095                            .BLKB   1
                        0096                    .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
                  00000098  0096                            .BLKW   1
                  00000096  0098                . = 150
                  00000098  0096                            .BLKW   1
                        0098                    .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
                  0000009C  0098                            .BLKL   1
                        009C                    .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
                  000000A0  009C                            .BLKL   1
                        00A0                    .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
                  000000A4  00A0                            .BLKL   1
                        00A4                    .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
                  000000A6  00A4                            .BLKW   1
                        00A6                    .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
                  000000A7  00A6                            .BLKB   1
                  000000A8  00A7                            .BLKB   1
                  00000074  00A8                . = 116
```

```
00000078  0074                          .BLKL   1
0000007C  0078                          .BLKL   1
00000080  007C                          .BLKL   1
00000084  0080                          .BLKL   1
00000088  0084                          .BLKL   1
0000008C  0088                          .BLKL   1
          008C              .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                          .BLKL   1
00000094  0090                          .BLKL   1
00000098  0094                          .BLKL   1
0000009C  0098                          .BLKL   1
          009C              .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                          .BLKB   1
          009D              .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                          .BLKB   1
          009E              .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                          .BLKB   1
000000A0  009F                          .BLKB   1
          00A0              .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                          .BLKB   1
          00A1              .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                          .BLKB   1
          00A2              .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                          .BLKB   1
000000A4  00A3                          .BLKB   1
          00A4              .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                          .BLKB   1
          00A5              .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                          .BLKW   1
000000A8  00A7                          .BLKB   1
          00A8              .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                          .BLKL   1
000000B0  00AC                          .BLKL   1
000000B4  00B0                          .BLKL   1
000000B8  00B4                          .BLKL   1
          00B8              .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                          .BLKL   1
          00BC              .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
          00BC              .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC              . = 116
          0074              .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                          .BLKL   1
          0078              .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                          .BLKL   1
          007C              .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                          .BLKW   1
00000080  007E                          .BLKW   1
          0000              .RESTORE
          0000    275       $VECDEF
          0000              .SAVE   LOCAL_BLOCK
          0000              .PSECT  $ABS$,ABS
00000000  00BC              .=0
          0000              .IIF    NB,VEC$Q_DISPATCH,      VEC$Q_DISPATCH:
00000008  0000                          .BLKQ   1
          0008              .IIF    NB,VEC$L_IDB,   VEC$L_IDB:
0000000C  0008                          .BLKL   1
          000C              .IIF    NB,VEC$L_INITIAL,       VEC$L_INITIAL:
```

```
00000010  000C                                  .BLKL   1
          0010                  .IIF    NB,VEC$W_MAPREG,        VEC$W_MAPREG:
00000012  0010                                  .BLKW   1
          0012                  .IIF    NB,VEC$B_NUMREG,        VEC$B_NUMREG:
00000013  0012                                  .BLKB   1
          0013                  .IIF    NB,VEC$B_DATAPATH,      VEC$B_DATAPATH:
00000014  0013                                  .BLKB   1
          0014                  .IIF    NB,VEC$L_ADP,   VEC$L_ADP:
00000018  0014                                  .BLKL   1
          0018                  .IIF    NB,VEC$L_UNITINIT,      VEC$L_UNITINIT:
0000001C  0018                                  .BLKL   1
          001C                  .IIF    NB,VEC$L_START, VEC$L_START:
00000020  001C                                  .BLKL   1
          0020                  .IIF    NB,VEC$L_UNITDISC,      VEC$L_UNITDISC:
00000024  0020                                  .BLKL   1
          0024                  .IIF    NB,VEC$C_LENGTH,        VEC$C_LENGTH:
          0024                  .IIF    NB,VEC$K_LENGTH,        VEC$K_LENGTH:
          0000                  .RESTORE
          0000    276           $DS_TYPEDEF
          0000    277           CMKDEF
          0000    278
          0000    279           $DS_QIODVR_DEF                  ; Offsets in driver header
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
00000008  0000                  .IIF    NB,.BLKQ,       .BLKQ   1
          0008                  .IIF    NB,DVR$W_SIZE,  DVR$W_SIZE:
0000000A  0008                  .IIF    NB,.BLKW,       .BLKW   1
          000A                  .IIF    NB,DVR$B_TYPE,  DVR$B_TYPE:
0000000B  000A                  .IIF    NB,.BLKB,       .BLKB   1
          000B                  .IIF    NB,DVR$B_CNUMVEC,       DVR$B_CNUMVEC:
0000000C  000B                  .IIF    NB,.BLKB,       .BLKB   1
          000C                  .IIF    NB,DVR$W_GENERIC,       DVR$W_GENERIC:
0000000E  000C                  .IIF    NB,.BLKW,       .BLKW   1
          000E                  .IIF    NB,DVR$W_DPT,   DVR$W_DPT:
00000010  000E                  .IIF    NB,.BLKW,       .BLKW   1
          0010                  .IIF    NB,DVR$L_IDENT, DVR$L_IDENT:
00000014  0010                  .IIF    NB,.BLKL,       .BLKL   1
          0014                  .IIF    NB,DVR$C_LEN,   DVR$C_LEN:
          0000                  .RESTORE
          0000    280           $DS_HPODEF
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  00BC                  .=0
          0000                  .IIF    NB,HP$Q_DEVICE, HP$Q_DEVICE:
00000008  0000                  .IIF    NB,.BLKQ,       .BLKQ   1
          0008                  .IIF    NB,HP$W_SIZE,   HP$W_SIZE:
0000000A  0008                  .IIF    NB,.BLKW,       .BLKW   1
          000A                  .IIF    NB,HP$B_FLAGS,  HP$B_FLAGS:
0000000B  000A                  .IIF    NB,.BLKB,       .BLKB   1
          000B                  .IIF    NB,HP$B_DRIVE,  HP$B_DRIVE:
0000000C  000B                  .IIF    NB,.BLKB,       .BLKB   1
          000C                  .IIF    NB,HP$T_DEVICE, HP$T_DEVICE:
00000018  000C                  .IIF    NB,.BLKB,       .BLKB   12
          0018                  .IIF    NB,HP$A_DEVICE, HP$A_DEVICE:
0000001C  0018                  .IIF    NB,.BLKL,       .BLKL   1
          001C                  .IIF    NB,HP$A_DVA,    HP$A_DVA:
```

```
00000020  001C                .IIF    NB,.BLKL,        .BLKL   1
          0020                .IIF    NB,HP$A_LINK,    HP$A_LINK:
00000024  0020                .IIF    NB,.BLKL,        .BLKL   1
          0024                .IIF    NB,HP$W_VECTOR, HP$W_VECTOR:
00000026  0024                .IIF    NB,.BLKW,        .BLKW   1
          0026                .IIF    NB,HP$T_TYPE,    HP$T_TYPE:
00000032  0026                .IIF    NB,.BLKB,        .BLKB   12
          0032                .IIF    NB,HP$A_DEPENDENT,       HP$A_DEPENDENT:
          0000                .RESTORE
          0000    281         $DS_RH780_DEF
          0000                .SAVE   LOCAL_BLOCK
          0000                .PSECT  $ABS$,ABS
00000032  00BC                .=HP$A_DEPENDENT
          0032                .IIF    NB,HP$B_RH780_TR,        HP$B_RH780_TR:
          0032                .IIF    NB,RH780$B_TR,   RH780$B_TR:
00000033  0032                .IIF    NB,.BLKB,        .BLKB   1
          0033                .IIF    NB,HP$B_RH780_BR,        HP$B_RH780_BR:
          0033                .IIF    NB,RH780$B_BR,   RH780$B_BR:
00000034  0033                .IIF    NB,.BLKB,        .BLKB   1
          0034                .IIF    NB,HP$K_RH780_LEN,       HP$K_RH780_LEN:
          0034                .IIF    NB,RH780$K_LEN, RH780$K_LEN:
          0000                .RESTORE
          0000    282         $DS_DW780_DEF
          0000                .SAVE   LOCAL_BLOCK
          0000                .PSECT  $ABS$,ABS
00000032  00BC                .=HP$A_DEPENDENT
          0032                .IIF    NB,DW780$B_TR,   DW780$B_TR:
00000033  0032                .IIF    NB,.BLKB,        .BLKB   1
          0033                .IIF    NB,DW780$B_BR,   DW780$B_BR:
00000034  0033                .IIF    NB,.BLKB,        .BLKB   1
          0034                .IIF    NB,DW780$K_LEN, DW780$K_LEN:
          0000                .RESTORE
          0000    283         $DS_DR780_DEF
          0000                .SAVE   LOCAL_BLOCK
          0000                .PSECT  $ABS$,ABS
00000032  00BC                .=HP$A_DEPENDENT
          0032                .IIF    NB,DR780$B_TR,   DR780$B_TR:
00000033  0032                .IIF    NB,.BLKB,        .BLKB   1
          0033                .IIF    NB,DR780$B_BR,   DR780$B_BR:
00000034  0033                .IIF    NB,.BLKB,        .BLKB   1
          0034                .IIF    NB,DR780$B_SELF,         DR780$B_SELF:
00000035  0034                .IIF    NB,.BLKB,        .BLKB   1
          0035                .IIF    NB,DR780$B_CONFIG,       DR780$B_CONFIG:
00000036  0035                .IIF    NB,.BLKB,        .BLKB   1
          0036                .IIF    NB,DR780$B_UUT, DR780$B_UUT:
00000037  0036                .IIF    NB,.BLKB,        .BLKB   1
          0037                .IIF    NB,DR780$K_LEN, DR780$K_LEN:
          0000                .RESTORE
          0000    284         $DS_SCBDEF
          0000    285         $DS_LOAD_DEF
          0000    286         $DS_DW750_DEF
          0000                .SAVE   LOCAL_BLOCK
          0000                .PSECT  $ABS$,ABS
00000032  0200                .=HP$A_DEPENDENT
          0032                .IIF    NB,DW750$K_LEN, DW750$K_LEN:
          0000                .RESTORE
          0000    287         $DS_DSDEF
```

```
                    0000   288          $DS_DSADEF
                    0000   289
                    0000   290          DSFDEF
                    0000   291
                    0000   292          .LIST MEB
```

```
                                .Subtitle       External Symbols

                                .EXTRN  DS$BREAK
                                .EXTRN  IOC$GL_ADPLIST, IOC$GL_DEVLIST, IOC$GL_DPTLIST
                                .EXTRN  EXE$ALONONPAGED,EXE$DEANONPAGED
                                .EXTRN  IO$TESTCSR_L,   IO$TESTCSR_W
                                .EXTRN  DSX$PRINT,      DS_ERRSUP,      DS$GPHARD
                                .EXTRN  DS$LOAD,        DSR$COMPLETION
                                .EXTRN  MBA$INITIAL,    MBA$INT,        UBA$UNEXINT
                                .EXTRN  DR$INITIAL,     DR$INT
                                .EXTRN  SCB_BASE,       SCB_IMAGE,      UBADP_CPU
                                .EXTRN  DS$K_CCB,       VMS$QIO,        IOGEN$CONN_VEC
                                .EXTRN  WRITEVBLK,      READVBLK,       PREADVBLK
                                .EXTRN  SS$_INSFMEM,    SS$_NORMAL,     SS$_IVCHAN
                                .EXTRN  IOC$INITDRV,    SCAN$SEARCHLIST,SCAN$ALPHA
                                .EXTRN  IOC$IOPOST,     EXE$FORKDSPTH,  IOC$REINITDRV
                                .EXTRN  LOC$PTDESC,     QIOCLEAN_CPU
                                .EXTRN  IOC$RTN
                                .EXTRN  DS$GL_FLAGS,    DS$GK_SID                                       ;[21][25]
                                .EXTRN  REMOTE_NODE_CONTROL_LEVEL, SEND_CONSOLE_RESPONSE                ;[21]

                        ;
                        ;       Weak symbols

                                .WEAK   TODR_BYTE_CNT,  CONSOLEAAA_FLAGS, TODR_STORE, DS$M_TODR_ACTIVE
                                .WEAK   REMOTE_NODE_CONTROL_LEVEL, SEND_CONSOLE_RESPONSE                ;[21]
                                .WEAK   UBA$INITIAL,    UBINTSZ                                 ;[18][26]       ;G:2
                                .WEAK   TXCS$V_RDY
```

```
                                00000000   323          .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                                [11]
                                    0000   324
                                    0000   325          MODNAM  QIO
                         4F 49 51 00'       0000      $MODULE:          .ASCIC  \QIO\
                                      03    0000
                                    0004   326
                                    0004   327 EXE$GL_PWRDONE::                          ; End time for power up interval        [14]
                             00000000       0004   328          .LONG   0                ; Done now                              [14]
                                    0008   329
                                    0008   330 AUNKFUNC:
20 6E 77 6F 6E 6B 6E 55 20 3F 3F 00'       0008   331          .ASCIC  \?? Unknown QIO function or channel number zero.\
6E 6F 69 74 63 6E 75 66 20 4F 49 51        0014
20 6C 65 6E 6E 61 68 63 20 72 6F 20        0020
2E 6F 72 65 7A 20 72 65 62 6D 75 6E        002C
                                  2F       0008
                                    0038   332
                                    0038   333          .ALIGN  LONG
                                    0038   334 CTL$GW_NMIOCH::          ; NUMBER OF CHANNELS
                                FFFF'       0038   335          .WORD   DS$K_CCB - 1
                                    003A   336
                                    003A   337 CTL$GW_CHINDX::          ; MAXIMUM CHANNEL INDEX + 1
                                0000'       003A   338          .WORD   DS$K_CCB @ 4
                                    003C   339
                                    003C   340 MBINT_DISP:                               ;INTERRUPT DISPATCHER MODEL
                              3C    BB      003C   341          PUSHR   #^M<R2,R3,R4,R5>
                    00000000  9F    16      003E   342          JSB     @#0
                                    0044   343
                                    0044   344 UBINT_DISP::
                              3F    BB      0044   345          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
                                    0046   346
                                    0046   347 T_NODRIVER:
21 20 52 45 56 49 52 44 20 3F 3F 00'       0046   348          .ASCIC  "?? DRIVER !AS NOT FOUND FOR !AS!/"
44 4E 55 4F 46 20 54 4F 4E 20 53 41        0052
      2F 21 53 41 21 20 52 4F 46 20        005E
                                  21       0046
                                    0068   349
                                    0068   350 T_WRONGVER:
49 4D 20 54 4E 45 44 49 20 3F 3F 00'       0068   351          .ASCIC  "?? IDENT MISMATCH, !AS=!UW, SUPERVISOR=!UW!/"
3D 53 41 21 20 2C 48 43 54 41 4D 53        0074
49 56 52 45 50 55 53 20 2C 57 55 21        0080
         2F 21 57 55 21 3D 52 4F 53        008C
                                  2C       0068
                                    0095   352
                                    0095   353 T_INVADP:
49 20 53 49 20 43 41 21 20 3F 3F 00'       0095   354          .ASCIC  "?? !AC IS INCORRECT ADAPTER TYPE FOR !AS!/"
41 44 41 20 54 43 45 52 52 4F 43 4E        00A1
4F 46 20 45 50 59 54 20 52 45 54 50        00AD
               2F 21 53 41 21 20 52        00B9
                                  2A       0095
                                    00C0   355
                                    00C0   356
                                    00C0   357 T_CRLF:
                              0A 0D         00C0   358          .ASCII  <CR><LF>         ;       [21]
                                    00C2   359 ;
                                    00C2   360 ; Table to convert adapter name to adapter type, Parallel to T_KNOWN_DEV
                                    00C2   361 ;
                                    00C2   362 T_ADAPTER_TYPE:
```

```
                        01  00C2   363          .BYTE   AT$_UBA                 ; DW780
                        00  00C3   364          .BYTE   AT$_MBA                 ; RH780
                        01  00C4   365          .BYTE   AT$_UBA                 ; DW750
                        00  00C5   366          .BYTE   AT$_MBA                 ; RH750
                        01  00C6   367          .BYTE   AT$_UBA                 ; DW730
                        02  00C7   368          .BYTE   AT$_DR                  ; RH780
                        01  00C8   369          .BYTE   AT$_UBA                 ; DWBUA                         [20]
                            00C9   370
                            00C9   371 T_KNOWN_DEV:
                     0007'  00C9   372          .WORD   $_KNOWN
                     000E'  00CB   373          .WORD   10$-.
                     0012'  00CD   374          .WORD   20$-.
                     0016'  00CF   375          .WORD   30$-.
                     001A'  00D1   376          .WORD   40$-.
                     001E'  00D3   377          .WORD   50$-.
                     0022'  00D5   378          .WORD   60$-.
                     0026'  00D7   379          .WORD   70$-.
                 00000007   00D9   380 $_KNOWN=<.-T_KNOWN_DEV-1>/2
30 38 37 57 44 00'  00D9   381 10$:    .ASCIC  "DW780"
                        05  00D9
30 38 37 48 52 00'  00DF   382 20$:    .ASCIC  "RH780"
                        05  00DF
30 35 37 57 44 00'  00E5   383 30$:    .ASCIC  "DW750"
                        05  00E5
30 35 37 48 52 00'  00EB   384 40$:    .ASCIC  "RH750"
                        05  00EB
30 33 37 57 44 00'  00F1   385 50$:    .ASCIC  "DW730"
                        05  00F1
30 38 37 52 44 00'  00F7   386 60$:    .ASCIC  "DR780"
                        05  00F7
41 55 42 57 44 00'  00FD   387 70$:    .ASCIC  "DWBUA"                         ;       [20]
                        05  00FD
```

```
        00000000          .Psect  Work, NoExe, NoShr, Wrt, Long   ;                                [11]
            0000  QIO$L_ALLOCATE:
00000000    0000          .LONG                                   ; BYTES ALLOCATED TO QIO DATA STRUCTURES
            0004
        00  0004  LASTCHAR :      .BYTE   0                       ; Last character written over the NI    [21]
            0005
            0005  LOAD_FLAGS:                                     ;CONTROL BLOCKS CREATED FLAGS
00000006    0005          .BLKB   1
            0006          _VIELD  LOAD,0,<-                       ; FLAG DEFINTIONS
            0006                  <DDB,,M>,-                      ;  DDB CREATED
            0006                  <CRB,,M>,-                      ;  CRB CREATED
            0006                  <IDB,,M>,-                      ;  IDB CREATED
            0006                  <UCB,,M>,-                      ;  UCB CREATED
            0006                  >
            0006  DDB_BLINK:                                      ;DDB BACKWARD LINK
0000000A    0006          .BLKL   1
            000A
            000A  UCB_BLINK:                                      ;UCB BACKWARD LINK
0000000E    000A          .BLKL   1
```

```
                                  000E          .SBTTL  EXE$QIO         QIO DISPATCHING ROUTINE
                              00000000          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                [11]
                                  0000  ;++
                                  0000  ; FUNCTIONAL DESCRIPTION:
                                  0000  ;
                                  0000  ;
                                  0000  ; CALLING SEQUENCE:
                                  0000  ;
                                  0000  ;       NONE
                                  0000  ;
                                  0000  ; INPUT PARAMETERS:     NONE
                                  0000  ;
                                  0000  ; IMPLICIT INPUTS:      NONE
                                  0000  ;
                                  0000  ; OUTPUT PARAMETERS:    NONE
                                  0000  ;
                                  0000  ; IMPLICIT OUTPUTS:     NONE
                                  0000  ;
                                  0000  ; COMPLETION CODES:     NONE
                                  0000  ;
                                  0000  ; SIDE EFFECTS:         NONE
                                  0000  ;
                                  0000  ;--
                              0FDC  0000  .ENTRY  EXE$QIO,^M<R2,R3,R4,R6,R7,R8,R9,R10,R11>
                                  0002
                                  0002          Br_If_NI  5$                    ; if NI do not call Vms QIO in any case [21]
        00000000'EF    1A    E0   0002          BBS     #DS$V_NI, -                     ; If bit set, branch            [49]
                       42          0009                  L^DS$GL_Flags, 5$               ;                               [49]
                                  000A
                    08 AC    B5   000A          TSTW    QIO$_CHAN(AP)           ; LOOK AT CHANNEL NUMBER
                       3D    13   000D          BEQL    5$                      ; IF 0, DO "DS" CONSOLE I/O
        00000000'EF    6C    FA   000F          CALLG   (AP),L^VMS$QIO                  ; Else call VMS/QIO handler     [11]
                                  0016          Br_if_MP  3$
05000000 8F    00000000'8F    D1   0016          CMPL    #DS$GK_SID,#^X05000000          ; Branch if Scorpio [61]
                       28    13   0021          BEQL    3$                              ; Yes, branch
06000000 8F    00000000'8F    D1   0023          CMPL    #DS$GK_SID,#^X06000000          ; Branch if Nautilus [61]
                       1B    13   002E          BEQL    3$                              ; Yes, branch
11000000 8F    00000000'8F    D1   0030          CMPL    #DS$GK_SID,#^X11000000          ; Branch if RRR [69]          ;G:20
                       0E    13   003B          BEQL    3$                                                            ;G:18
0A000000 8F    00000000'8F    D1   003D          CMPL    #DS$GK_SID,#^X0A000000          ; Branch if LLL [69]          ;G:20
                       01    13   0048          BEQL    3$                                                            ;G:22
                             05   004A          RSB                             ;  and return to ENTRY                  [22]
                             04   004B  3$:     RET                             ;  and return to ENTRYMP                [22]
                                  004C                                          ;
                                  004C
                                  004C  5$:
                                  004C
    50    0C AC    06    00    EF   004C          EXTZV   #IO$V_FCODE,#IO$S_FCODE, -
                                  0052                  QIO$_FUNC(AP),R0        ; Get function code
                                  0052          CASE    R0,TYPE=L,LIMIT=#IO$_WRITEVBLK,DISPLIST=<-
                                  0052                  30$, -                  ; Function ^X30
                                  0052                  40$, -                  ; Function ^X31
                                  0052                  20$, -                  ; Function ^X32
                                  0052                  20$, -                  ; Function ^X33
                                  0052                  20$, -                  ; Function ^X34
                                  0052                  20$, -                  ; Function ^X35
                                  0052                  20$, -                  ; Function ^X36
```

```
                                  0052   456                 40$>                    ; Function ^X37
          07'  30   50    CF      0052              CASEL   R0,#IO$_WRITEVBLK,S^#<<30001$-30000$>/2>-1
                                  0056        30000$:
                          005D'   0056              .SIGNED_WORD    30$-30000$
                          006B'   0058              .SIGNED_WORD    40$-30000$
                          0010'   005A              .SIGNED_WORD    20$-30000$
                          0010'   005C              .SIGNED_WORD    20$-30000$
                          0010'   005E              .SIGNED_WORD    20$-30000$
                          0010'   0060              .SIGNED_WORD    20$-30000$
                          0010'   0062              .SIGNED_WORD    20$-30000$
                          006B'   0064              .SIGNED_WORD    40$-30000$
                                  0066        30001$:
                                  0066   457
                                  0066   458 20$:   ERRSUP_S        MSGADR=AUNKFUNC
          00000000'EF     DF      0066                      PUSHAL  $MODULE
          00000008'EF     DF      006C                      PUSHAL  AUNKFUNC
                    01    DD      0072                      PUSHL   #$ER
     00000000'EF    03    FB      0074                      CALLS   #3, DS_ERRSUP
                    50    D4      007B   459        CLRL    R0                      ; Return error                          [22]
                                  007D   460        Br_if_MP  23$
05000000 8F  00000000'8F  D1      007D              CMPL    #DS$GK_SID,#^X05000000          ; Branch if Scorpio [61]
                    28    13      0088              BEQL    23$                             ; Yes, branch
06000000 8F  00000000'8F  D1      008A              CMPL    #DS$GK_SID,#^X06000000          ; Branch if Nautilus [61]
                    1B    13      0095              BEQL    23$                             ; Yes, branch
11000000 8F  00000000'8F  D1      0097              CMPL    #DS$GK_SID,#^X11000000          ; Branch if RRR [69]          ;G:20
                    0E    13      00A2              BEQL    23$                                                           ;G:18
0A000000 8F  00000000'8F  D1      00A4              CMPL    #DS$GK_SID,#^X0A000000          ; Branch if LLL [69]          ;G:20
                    01    13      00AF              BEQL    23$                                                           ;G:22
                          05      00B1   461        RSB                             ;  and return to ENTRY                  [22]
                          04      00B2   462 23$:   RET                             ;  and return to ENTRYMP                [22]
                                  00B3   463
                                  00B3   464 30$:   Br_If_NI   50$                  ; If NI then write accross the Net.     [21]
     00000000'EF    1A    E0      00B3              BBS     #DS$V_NI, -                     ; If bit set, branch            [49]
                    14            00BA                        L^DS$GL_Flags, 50$            ;                               [49]
                                  00BB   465
                                  00BB   466 35$:
          00000000'EF     17      00BB   467        JMP     WRITEVBLK               ; Changed BRW to JMP                    [15]
                                  00C1   468                                        ;                                       [15]
                                  00C1   469 40$:   Br_If_NI   60$                  ; If NI then read buffer from the Net.  [21]
     00000000'EF    1A    E0      00C1              BBS     #DS$V_NI, -                     ; If bit set, branch            [49]
                    26            00C8                        L^DS$GL_Flags, 60$            ;                               [49]
                                  00C9   470
                                  00C9   471 45$:
          00000000'EF     17      00C9   472        JMP     READVBLK                ; Changed BRW to JMP                    [15]
                                  00CF   473                                        ;                                       [15]
                    7E    D4      00CF   474 50$:   CLRL    -(SP)                   ; ZERO to indicate no-data-lost         [21]
                 20 AC    DD      00D1   475        PUSHL   QIO$_P2(AP)             ; Push length of ascii string           [21]
                 1C AC    DD      00D4   476        PUSHL   QIO$_P1(AP)             ; Push address of ascii string          [21]
     00000000'EF    03    FB      00D7   477        CALLS   #3,SEND_CONSOLE_RESPONSE ; WRITE over the network               [21]
                                  00DE   478
      52   20 AC    01    C3      00DE   479        SUBL3   #1,QIO$_P2(AP),R2       ; Calculate offset of last character writen
           52   1C AC    C0      00E3   480        ADDL2   QIO$_P1(AP),R2          ; and add it to the address of the string.
     00000004'EF    62    90      00E7   481        MOVB    (R2),LASTCHAR           ; Store the last character writen       [21]
                                  00EE   482
                          04      00EE   483        RET                             ; Return                                [21]
                                  00EF   484
                    37    91      00EF   485 60$:   CMPB    #IO$_READPROMPT,-       ; Check if we are doing a READPROMPT    [21]
```

```
                 0C AC      00F1                  QIO$_FUNC(AP)
                    27   12  00F3          BNEQ    80$                     ; If not, then just read packet          [21]
      00000004'EF   0A   91  00F5          CMPB    #LF,LASTCHAR            ; See if last char. written was a line feed
                    11   13  00FC          BEQL    70$                                                              ;[21]
                    7E   D4  00FE          CLRL    -(SP)                   ;  if not, then                          [21]
                    02   DD  0100          PUSHL   #CRLF_LENGTH            ;  push length of Cr Lf                  [21]
          000000C0'EF   9F  0102          PUSHAB  T_CRLF                  ;  push address of Cr Lf                 [21]
      00000000'EF   03   FB  0108          CALLS   #3,SEND_CONSOLE_RESPONSE ; WRITE Cr Lf over the network          [21]
                            010F
                    7E   D4  010F  70$:    CLRL    -(SP)                   ; Now write the prompt:                  [21]
            7E   2C AC   7D  0111          MOVQ    QIO$_P5(AP),-(SP)       ; Push the address and length of Prompt [21]
      00000000'EF   03   FB  0115          CALLS   #3,SEND_CONSOLE_RESPONSE ; WRITE Prompt over the network         [21]
                            011C
                 10 AC   DD  011C  80$:    PUSHL   QIO$_IOSB(AP)           ; Push address of IO status block        [21]
                            011F                                            ; which will contain thelength of input [21]
                            011F                                            ; upon return.                           [21]
                 1C AC   DD  011F          PUSHL   QIO$_P1(AP)             ; Push address of input buffer           [21]
      00000000'EF   02   FB  0122          CALLS   #2,REMOTE_NODE_CONTROL_LEVEL  ; READ from the Network            [21]
            51   10 BC   DE  0129          MOVAL   @QIO$_IOSB(AP),R1       ; Get the address of the IO status block[21]
         61   61   10   78  012D          ASHL    #16,(R1),(R1)           ; Shift character count to 2nd word      [21]
              61   50   B0  0131          MOVW    R0,(R1)                 ; Store operation status in 1st word     [21]
              51   04   C0  0134          ADDL    #4,R1                   ; Point to third word                    [21]
              81   00   B0  0137          MOVW    #0,(R1)+                ; Store terminator character in 3rd word[21]
              81   00   B0  013A          MOVW    #0,(R1)+                ; Store the terminator size in 4th word [21]
                        04  013D          RET                             ; Return                                 [21]
                            013E
                            013E
```

```
                                013E   514          .SBTTL  QIO$INITIALIZE  Initialize QIO database
                                013E   515 ;++
                                013E   516 ; FUNCTIONAL DESCRIPTION:
                                013E   517 ;
                                013E   518 ;       This routine will initialize all of the the database that needs it.
                                013E   519 ;       It resets the DRIVER prologue queue.
                                013E   520 ;
                                013E   521 ; CALLING SEQUENCE:
                                013E   522 ;
                                013E   523 ;       BSBW    QIO$INITIALIZE
                                013E   524 ;
                                013E   525 ; INPUT PARAMETERS:     NONE
                                013E   526 ;
                                013E   527 ; IMPLICIT INPUTS:      NONE
                                013E   528 ;
                                013E   529 ; OUTPUT PARAMETERS:    NONE
                                013E   530 ;
                                013E   531 ; IMPLICIT OUTPUTS:     NONE
                                013E   532 ;
                                013E   533 ; COMPLETION CODES:     NONE
                                013E   534 ;
                                013E   535 ; SIDE EFFECTS:         NONE
                                013E   536 ;--
                                013E   537
                          0000  013E   538 .ENTRY  QIO$INITIALIZE,^M<>
                                0140   539
          51  00000000'EF   DE  0140   540         MOVAL   L^IOC$GL_DPTLIST,R1     ; GET ADDRESS OF DRIVER QUEUE LISTHEAD  [11]
              61    51      D0  0147   541         MOVL    R1,(R1)                 ; INIT FLINK
           04 A1    51      D0  014A   542         MOVL    R1,4(R1)                ; INIT BLINK
                                014E   543
00000000'EF       D4          014E   544         CLRL    L^IOC$GL_DEVLIST        ; CLEAR DDB HEADER LIST                 [11]
00000000'EF       D4          0154   545         CLRL    L^IOC$GL_ADPLIST        ; CLEAR ADAPTER HEADER LIST             [11]
00000000'EF       D4          015A   546         CLRL    L^QIO$L_ALLOCATE        ; CLEAR QIO MEMORY ALLOCATED            [11]
                              0160   547
                          04  0160   548         RET                             ; RETURN
```

```
                                0161    550 ;+
                                0161    551 ; Find DPT given driver name descriptor in R2,R3
                                0161    552 ;-
                                0161    553 LOC$DPT:
                 007C 8F    BB  0161    554          PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save descriptor of driver name
          5B  00000000'EF    9E  0165    555          MOVAB   L^IOC$GL_DPTLIST,R11    ; Address of DPT list head              [11]
                    56  5B   D0  016C    556          MOVL    R11,R6                  ; Save list header
                                016F    557
                    5B  6B   D0  016F    558 10$:     MOVL    DPT$L_FLINK(R11),R11    ; Link to next DPT
                50  5B  56   C3  0172    559          SUBL3   R6,R11,R0               ; End of list?, R0 = ZERO if so
                        15   13  0176    560          BEQL    20$                     ; Branch if end
                                0178    561
                    52  6E   7D  0178    562          MOVQ    (SP),R2                 ; Get descriptor of driver name
                  51  1E AB   9E  017B    563          MOVAB   DPT$T_NAME(R11),R1      ; Get address of driver name
                    50  81   9A  017F    564          MOVZBL  (R1)+,R0                ; Get length
        63  52  00  61  50   2D  0182    565          CMPC5   R0,(R1),#0,R2,(R3)      ; Same?
                        E5   12  0188    566          BNEQ    10$                     ; Branch if not
                    50  00'  D0  018A    567          MOVL    S^#SS$_NORMAL,R0        ; Success
                 007C 8F    BA  018D    568 20$:     POPR    #^M<R2,R3,R4,R5,R6>     ; Restore
                            05  0191    569          RSB                             ; Return R11 has DPT if R0=SS$_NORMAL
```

```
                        0192   571          .SBTTL  QIO$CLEANUP     Remove old QIO database
                        0192   572 ;++
                        0192   573 ; FUNCTIONAL DESCRIPTION:
                        0192   574 ;
                        0192   575 ;       This routine is used to remove and release all of the
                        0192   576 ;       QIO data base.
                        0192   577 ;       Every ADP is deallocated while restoring the SCB for each.
                        0192   578 ;       Scan thru the DDB list, making a list on the stack of the CRB's
                        0192   579 ;       present and deallocating each UCB and then the DDB.
                        0192   580 ;       Finally deallocate the space for each CRB and corresponding IDB.
                        0192   581 ;
                        0192   582 ; CALLING SEQUENCE:
                        0192   583 ;
                        0192   584 ;       CALLG   (SP),QIO$CLEANUP
                        0192   585 ;
                        0192   586 ; INPUT PARAMETERS:     NONE
                        0192   587 ;
                        0192   588 ; IMPLICIT INPUTS:      NONE
                        0192   589 ;
                        0192   590 ; OUTPUT PARAMETERS:    NONE
                        0192   591 ;
                        0192   592 ; IMPLICIT OUTPUTS:     NONE
                        0192   593 ;
                        0192   594 ; COMPLETION CODES:     NONE
                        0192   595 ;
                        0192   596 ; SIDE EFFECTS:
                        0192   597 ;
                        0192   598 ;       The current QIO database is destroyed.
                        0192   599 ;--
                        0192   600
                 007C   0192   601 .ENTRY  QIO$CLEANUP,^M<R2,R3,R4,R5,R6>
                        0194   602
 55  00000000'EF   DE   0194   603         MOVAL   SCB_BASE,R5                     ; BASE SCB
 54  00000000'EF   DE   019B   604         MOVAL   SCB_IMAGE,R4                    ; BASE SCB IMAGE
                        01A2   605         DSBINT                                  ; Don't allow interruptions
       7E   12   DB   01A2                        MFPR    S^#PR$_IPL,-(SP)
       12   1F   DA   01A5                        MTPR    #31,S^#PR$_IPL
                        01A8   606
 53  00000000'EF   DE   01A8   607         MOVAL   L^IOC$GL_ADPLIST,R3             ; POINT TO ADAPTER HEADER                [11]
       50   63   D0   01AF   608 10$:     MOVL    (R3),R0                         ; POINT TO FIRST ADAPTER BLOCK
  00000000'EF   16   01B2   609         JSB     QIOCLEAN_CPU                            ; Do CPU specific QIO cleanup    [11]
00A0 C5   00A0 C4   D0   01B8   610         MOVL    SCB$L_SFTLVL8(R4), -
                        01BF   611                 SCB$L_SFTLVL8(R5)            ; RETURN SOFT VECTOR OF FORK DISPATCH
0090 C5   0090 C4   D0   01BF   612         MOVL    SCB$L_SFTLVL4(R4), -
                        01C6   613                 SCB$L_SFTLVL4(R5)            ; RETURN SOFT VECTOR OF IOPOST
```

```
                7E   D4  01C6          CLRL    -(SP)                      ; TERMINATOR FOR CRB LIST
                         01C8
56   00000000'EF   D0  01C8  30$:      MOVL    L^IOC$GL_DEVLIST,R6        ; GET FIRST DDB                    [11]
                3F   13  01CF          BEQL    100$                       ; BRANCH IF NONE
                         01D1
     50   04 A6    D0  01D1          MOVL    DDB$L_UCB(R6),R0           ; POINT TO FIRST UCB
     55   20 A0    D0  01D5          MOVL    UCB$L_CRB(R0),R5           ; GET CRB ADDRESS
                         01D9  40$:
        51   6E    9E  01D9          MOVAB   (SP),R1                    ; POINT TO FIRST ELIGIBLE
                20   BB  01DC          PUSHR   #^MR5                      ; ASSUME NOT A DUPLICATE
                61   D5  01DE  50$:    TSTL    (R1)                       ; VALID
                07   13  01E0          BEQL    60$                        ; BRANCH IF AT END OF LIST
        55   81    D1  01E2          CMPL    (R1)+,R5                   ; SAME?
                F7   12  01E5          BNEQ    50$                        ; BRANCH IF NOT THE SAME
                8E   D5  01E7          TSTL    (SP)+                      ; REMOVE DUPLICATE CRB
                         01E9  60$:
     55   10 A5    D0  01E9          MOVL    CRB$L_LINK(R5),R5          ; LINK TO NEXT CRB
                EA   12  01ED          BNEQ    40$                        ; CHECK IT
                         01EF  70$:
     55   D8 A6    DE  01EF          MOVAL   DDB$L_UCB-UCB$L_LINK(R6),R5 ; FIRST UCB
     55   2C A5    D0  01F3  80$:    MOVL    UCB$L_LINK(R5),R5          ; LINK TO NEXT UCB
                08   13  01F7          BEQL    90$                        ; BRANCH IF NONE
        50   55    D0  01F9          MOVL    R5,R0                      ; COPY ADDRESS
           08E5    30  01FC          BSBW    QIO$DEALLOCATE             ; RELEASE THE SPACE
                F2   11  01FF          BRB     80$                        ; NEXT UCB
                         0201  90$:
        50   56    D0  0201          MOVL    R6,R0                      ; COPY DDB ADDRESS
00000000'EF   66   D0  0204          MOVL    DDB$L_LINK(R6),L^IOC$GL_DEVLIST ;                             [11]
           08D6    30  020B          BSBW    QIO$DEALLOCATE             ; RELEASE THE DDB
                B8   11  020E          BRB     30$                        ; NEXT DDB
                         0210  100$:
        56   8E    D0  0210          MOVL    (SP)+,R6                   ; GET CRB FROM STACK
                0F   13  0213          BEQL    110$                       ; BRANCH IF LAST ONE
     50   1C A6    D0  0215          MOVL    CRB$L_INTD+VEC$L_IDB(R6),R0 ; POINT TO IDB
           08C8    30  0219          BSBW    QIO$DEALLOCATE             ; RELEASE THE IDB
        50   56    D0  021C          MOVL    R6,R0                      ; COPY CRB ADDRESS
           08C2    30  021F          BSBW    QIO$DEALLOCATE             ; RELEASE THE CRB
                EC   11  0222          BRB     100$                       ; CHECK FOR MORE CRB'S
                         0224  110$:
                         0224
51   00000000'EF   DE  0224          MOVAL   L^IOC$GL_DPTLIST,R1        ; Address of header                [11]
        50   51    D0  022B          MOVL    R1,R0                      ; Duplicate
        50   60    D0  022E  130$:   MOVL    DPT$L_FLINK(R0),R0         ; Link to next
        51   50    D1  0231          CMPL    R0,R1                      ; Point back to header?
                05   13  0234          BEQL    140$                       ; Branch if back to header
           0B A0    94  0236          CLRB    DPT$B_REFC(R0)             ; Clear reference count
                F3   11  0239          BRB     130$                       ; next DPT
                         023B  140$:
                         023B          ENBINT                             ; Enable interrupts again
        12   8E    DA  023B                  MTPR    (SP)+,S^#PR$_IPL
52   00000000'EF   D0  023E          MOVL    QIO$L_ALLOCATE,R2          ; SHOULD ALL BE DEALLOCATED
                11   13  0245          BEQL    120$                       ; BRANCH IF IT WAS
                         0247          ERRSUP_S
     00000000'EF   DF  0247                  PUSHAL  $MODULE
                00   DD  024D                  PUSHL   #0
                02   DD  024F                  PUSHL   #$ER
00000000'EF   03   FB  0251                  CALLS   #3, DS_ERRSUP
```

```
                    0258   667 120$:
                04  0258   668          RET
```

```
0259    670          .SBTTL  QIO$ADDUNIT     Add a unit to the QIO database
0259    671 ;++
0259    672 ; FUNCTIONAL DESCRIPTION:
0259    673 ;
0259    674 ;       This routine takes all the necessary steps to
0259    675 ;       insert a unit into the QIO database, including adding
0259    676 ;       ADP's, CRB's, IDB's, DDB's and UCB's.
0259    677 ;       The generic name of the device is used to locate the
0259    678 ;       device driver for the device.  Starting with the specified
0259    679 ;       device fields are copied from the P-table to the ACF block
0259    680 ;       which contains all the information necessary to build the
0259    681 ;       QIO database.  For unknown devices the unit and vector information
0259    682 ;       is copied. For a TM03 the unit is copied to only one of two fields.
0259    683 ;       For channels, (DR780, RH780 & DW780) the configuration register is set
0259    684 ;       and the ACF is complete. For all non-channel devices this process
0259    685 ;       is repeated for the link device, and then for it's link device...etc.
0259    686 ;       Note that it is impossible to catch the case where the TM03 is
0259    687 ;       omitted from the link chain for a magtape drive... i.e. no
0259    688 ;       error is reported.  Errors are reported if the channel is omitted.
0259    689 ;
0259    690 ; CALLING SEQUENCE:
0259    691 ;
0259    692 ;       CALLG/S arglist/count,QIO$ADDUNIT
0259    693 ;
0259    694 ; INPUT PARAMETERS:
0259    695 ;
0259    696 ;       0(AP)   #1
0259    697 ;       4(AP)   Address of P-table
0259    698 ;
0259    699 ; IMPLICIT INPUTS:
0259    700 ;
0259    701 ;       Existing QIO database
0259    702 ;
0259    703 ; OUTPUT PARAMETERS:    NONE
0259    704 ;
0259    705 ; IMPLICIT OUTPUTS:
0259    706 ;
0259    707 ;       The QIO database may be updated.
0259    708 ;
0259    709 ; COMPLETION CODE:
0259    710 ;
0259    711 ;       SS$_NORMAL      if successful
0259    712 ;
0259    713 ; REGISTER USAGE:
0259    714 ;
0259    715 ;       R3      TR number of adapter
0259    716 ;       R4      Adapter CSR address
0259    717 ;       R5      Address of Driver Prologue Table
0259    718 ;       R6      Address of ADP
0259    719 ;       AP      Address of current P-table
0259    720 ;       R7      Address of ACF block
0259    721 ;       R10     Address of Adapter P-table
0259    722 ;       R11     Address of Controller P-table
0259    723 ;       AP      Address of Unit P-table
0259    724 ;
0259    725 ; SIDE EFFECTS:         NONE
0259    726 ;--
```

```
                     0259   727          $OFFSET 0,NEGATIVE,< -
                     0259   728          <FILE,12>, -                    ; File name of driver
                     0259   729          <FILEDESC,8>, -                 ; Descriptor of driver file name
                     0259   730          <FILEDEF,8>, -                  ; Default descriptor
                     0259   731          <LENGTH,4>, -                   ; Longword to recieve file length
                     0259   732          <ACF,ACF$C_LENGTH>, -           ; Place to build an ACF
                     0259   733          <LOCAL,0>>
                     0259                .SAVE   LOCAL_BLOCK
                     0259                .PSECT $ABS$ ABS
            00000000 FE70                .=0
            FFFFFFF4 0000                .BLKB   -12
                     FFF4        FILE:
            FFFFFFEC FFF4                .BLKB   -8
                     FFEC        FILEDESC:
            FFFFFFE4 FFEC                .BLKB   -8
                     FFE4        FILEDEF:
            FFFFFFE0 FFE4                .BLKB   -4
                     FFE0        LENGTH:
            FFFFFFC0 FFE0                .BLKB   -ACF$C_LENGTH
                     FFC0        ACF:
                     FFC0        LOCAL:
                     0259                .RESTORE
```

```
                     0FFC  0259    735 .ENTRY  QIO$ADDUNIT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                           025B    736
      5E   C0 AD    9E    025B    737         MOVAB   LOCAL(FP),SP            ; Allocate stack based storage
      57   C0 AD    9E    025F    738         MOVAB   ACF(FP),R7              ; Base ACF
50  00000000'EF    DE    0263    739         MOVAL   SCB_BASE,R0             ; Base SCB
                           026A    740
      5C   04 AC    D0    026A    741         MOVL    4(AP),AP                ; Address P-table
                           026E    742
   55   04 AC   01    C1    026E    743         ADDL3   #1,HP$Q_DEVICE+4(AP),R5 ; Point past "_"
      54   6C   01    C3    0273    744         SUBL3   #1,HP$Q_DEVICE(AP),R4   ; Length of device name
    00000000'EF    16    0277    745         JSB     SCAN$ALPHA              ; Determine length of alphabetic
         52   50    D0    027D    746         MOVL    R0,R2                   ; Save length of name
         51   50    C0    0280    747         ADDL    R0,R1                   ; Point past string
         7E   71    90    0283    748 10$:    MOVB    -(R1),-(SP)             ; Copy device name onto stack
           FA 50    F5    0286    749         SOBGTR  R0,10$                  ; Count them all
         7E   52    90    0289    750         MOVB    R2,-(SP)                ; Make it ASCIC
                           028C    751
      14 A7   6E    9E    028C    752         MOVAB   (SP),ACF$L_DEVNAME(R7)  ; Set address of device name
                           0290    753
         56   5C    D0    0290    754         MOVL    AP,R6                   ; Copy P-table address
      51   26 A6    9E    0293    755 20$:    MOVAB   HP$T_TYPE(R6),R1        ; Address of device type
         50   81    9A    0297    756         MOVZBL  (R1)+,R0                ; Get length of device type
    00000000'EF    16    029A    757         JSB     LOC$PTDESC              ; Locate this PT-descriptor
                13   13    02A0    758         BEQL    40$                     ; Branch to NODRIVER
                           02A2    759
         51   80    9A    02A2    760         MOVZBL  (R0)+,R1                ; Get length of type name
      50   02 A041    9E    02A5    761         MOVAB   2(R0)[R1],R0            ; Get address of driver prefix
         52   60    3C    02AA    762         MOVZWL  (R0),R2                 ; Get prefix
                0E   12    02AD    763         BNEQ    50$                     ; Branch if valid prefix
      56   20 A6    D0    02AF    764         MOVL    HP$A_LINK(R6),R6        ; Link to previous device
                DE   12    02B3    765         BNEQ    20$                     ; Check for prefix in controller
50  006600B1 8F    D0    02B5    766 40$:    MOVL    #DS$_NOSUPPORT,R0       ; No QIO support for device
                04   04    02BC    767         RET                             ; Exit
                           02BD    768
                           02BD    769 ;+
                           02BD    770 ; Locate driver, given generic device prefix in R2
                           02BD    771 ;-
50  00000000'EF    7E    02BD    772 50$:    MOVAQ   L^IOC$GL_DPTLIST,R0     ; Point to driver listhead                [11]
         55   50    D0    02C4    773         MOVL    R0,R5                   ; Copy listhead
         55   65    D0    02C7    774 60$:    MOVL    DPT$L_FLINK(R5),R5      ; Link to next driver
         55   50    D1    02CA    775         CMPL    R0,R5                   ; At end?
         09   13    02CD    776 70$:    BEQL    LOAD_DRIVER             ; Branch if more drivers to check
      1F A5   52    B1    02CF    777         CMPW    R2,DPT$T_NAME+1(R5)     ; Match driver prefix
         F2   12    02D3    778         BNEQ    60$                     ; Try again no match
            00CE    31    02D5    779         BRW     FILL_ACF                ; Branch since driver found
```

```
                        02D8   781 ;+
                        02D8   782 ; Load device driver
                        02D8   783 ; Inputs:       R2 - 2 character generic device name
                        02D8   784 ; Outputs:      R4 - Address of DPT
                        02D8   785 ;-
                        02D8   786 LOAD_DRIVER:
F4 AD   3F515645 8F     D0  02D8   787          MOVL    #^A"EVQ?",FILE(FP)          ; Set first part of name
         F7 AD   52     B0  02E0   788          MOVW    R2,FILE+3(FP)               ; Insert generic device name
F9 AD   4558452E 8F     D0  02E4   789          MOVL    #^A".EXE",FILE+5(FP)        ; Read .EXE file
         EC AD   09     D0  02EC   790          MOVL    #9,FILEDESC(FP)             ; Set length of file spec
      F0 AD   F4 AD     9E  02F0   791          MOVAB   FILE(FP),FILEDESC+4(FP)     ; Set address of file spec
              E4 AD     7C  02F5   792          CLRQ    FILEDEF(FP)                 ; Clear default descriptor
                        02F8   793
              E0 AD     DF  02F8   794          PUSHAL  LENGTH(FP)                  ; Address to return length of file
                 01     DD  02FB   795          PUSHL   #1                          ; fake buf addr to fool ds$load [27]  ;G:4
                 7E     D4  02FD   796          CLRL    -(SP)                       ; Length of buffer, and address [27]  ;G:5
              E4 AD     7F  02FF   797          PUSHAQ  FILEDEF(FP)                 ; Address of default
              EC AD     7F  0302   798          PUSHAQ  FILEDESC(FP)                ; Address of file spec
                 05     DD  0305   799          PUSHL   #5                          ; Length of arg list
   00000000'EF    6E     FA  0307   800          CALLG   (SP),DS$LOAD                ; Locate existence and length of file
            1D 50       E8  030E   801          BLBS    R0,50$                      ; Branch if file found
                 01     BB  0311   802          PUSHR   #^M<R0>                     ; Save status code
                 6C     9F  0313   803          PUSHAB  HP$Q_DEVICE(AP)             ; Name of device
              EC AD     9F  0315   804          PUSHAB  FILEDESC(FP)                ; descriptor of file name
      00000046'EF       9F  0318   805          PUSHAB  L^T_NODRIVER                ; Address of edit string                [11]
            7E    01    9A  031E   806          movzbl  #ds$k_printf,-(sp)          ; use PRINTF                            [13]
            7E    22    98  0321   807          cvtbl   #ds$k_type_qio_nodriver,-(sp)   ; code                              [13]
   00000000'GF    05    FB  0324   808          CALLS   #5, G^DSX$PRINT             ; Print the error                       [13]
                 01     BA  032B   809          POPR    #^M<R0>                     ; Restore failing DS$LOAD status
                 04     RET 032D   810          RET                                 ; Exit
                        032E   811
      51   E0 AD     D0  032E   812 50$:     MOVL    LENGTH(FP),R1               ; Length of memory required
      00000000'EF    16  0332   813          JSB     EXE$ALONONPAGED             ; Allocate the memory for the driver
            07 50     E8  0338   814          BLBS    R0,55$                      ; Continue if OK
      00000000'EF    16  033B   815          JSB     DSR$COMPLETION              ; Print the error
                 04     RET 0341   816          RET                                 ; Return
                        0342   817
         53   51     3C  0342   818 55$:     MOVZWL  R1,R3                       ; Save length of file
      0C AE   51     3C  0345   819          MOVZWL  R1,LOAD$_LENGTH(SP)         ; Store length to read
      10 AE   52     D0  0349   820          MOVL    R2,LOAD$_ADDRESS(SP)        ; Store address of buffer
   00000000'EF    8E     FB  034D   821          CALLS   (SP)+,DS$LOAD               ; Now read the file
            08 50     E8  0354   822          BLBS    R0,60$                      ; Continue if OK
      00000000'EF    16  0357   823          JSB     DSR$COMPLETION              ; Print the error
                 2D     BRB 035D   824          BRB     70$                         ; Error return, release driver space
                        035F   825
      08 A2   53     B0  035F   826 60$:     MOVW    R3,DVR$W_SIZE(R2)           ; Stash size of driver
10 A2 FFFF0001 8F     D1  0363   827          CMPL    #DVR$_IDENT, -
                        036B   828                  DVR$L_IDENT(R2)             ; Is ident version same as supervisor?
                 2B     BEQL 036B   829          BEQL    80$                         ; Branch if so
      FFFF0001 8F     DD  036D   830          PUSHL   #DVR$_IDENT                 ; Correct ident
            10 A2     DD  0373   831          PUSHL   DVR$L_IDENT(R2)             ; Current ident
            EC AD     7F  0376   832          PUSHAQ  FILEDESC(FP)                ; Address of driver
      00000068'EF       9F  0379   833          PUSHAB  L^T_WRONGVER                ; Edit string                           [11]
            7E    01    9A  037F   834          movzbl  #ds$k_printf,-(sp)          ; Use PRINTF                            [13]
            7E    23    98  0382   835          cvtbl   #ds$K_type_qio_wrongver,-(sp)   ; code                              [13]
   00000000'GF    06    FB  0385   836          CALLS   #6, G^DSX$PRINT             ; Print the error                       [13]
                        038C   837 ;
```

```
                        038C   838 ; Release the driver and return error
                        038C   839 ;
          50    52      D0   038C   840 70$:    MOVL    R2,R0                   ; Point to buffer
    00000000'EF         16   038F   841         JSB     EXE$DEANONPAGED         ; Release the space
                50      D4   0395   842         CLRL    R0                      ; Error return
                        04   0397   843         RET                             ; Exit
                             0398   844
          55    14 A2   9E   0398   845 80$:    MOVAB   DVR$C_LEN(R2),R5        ; Base driver prologue table
                08 A5   B4   039C   846         CLRW    DPT$W_SIZE(R5)          ; Clear size of DPT, catch bugs later
    00000000'EF   65      0E   039F   847         INSQUE  (R5),IOC$GL_DPTLIST     ; Insert in queue
```

```
                                03A6    849 ;+
                                03A6    850 ; Fill ACF block
                                03A6    851 ;-
                                03A6    852 FILL_ACF:
          18 A7    1E A5    9E  03A6    853             MOVAB   DPT$T_NAME(R5), -
                                03AB    854                     ACF$L_DRVNAME(R7)       ; Address of driver name
                                03AB    855
                                03AB    856
             5B    20 AC    D0  03AB    857             MOVL    HP$A_LINK(AP),R11       ; Address of Controller P-TABLE
                      03    12  03AF    858             BNEQ    20$                     ; Branch if real controller
                   5B    5C   D0  03B1    859             MOVL    AP,R11                  ; Use unit for controller
             5A    20 AB    D0  03B4    860 20$:        MOVL    HP$A_LINK(R11),R10      ; Address of Adapter P-TABLE
                      06    12  03B8    861             BNEQ    40$                     ; Branch if real adapter
                   5A    5B   D0  03BA    862             MOVL    R11,R10                 ; Use controller for adapter
                   5B    5C   D0  03BD    863             MOVL    AP,R11                  ; Use unit for controller
                                03C0    864
       41494253 8F    27 AA    D1  03C0    865 40$:        CMPL    HP$T_TYPE+1(R10),#^A"SBIA" ; SBIA attached?                     ;[17
                      06    12  03C8    866             BNEQ    45$                     ; Branch if not                         ;[17
                   5A    5B   D0  03CA    867             MOVL    R11,R10                 ; backtrack through ptables             ;[17
                   5B    5C   D0  03CD    868             MOVL    AP,R11                  ; ...                                   ;[17
                                03D0    869
             51    26 AA    9E  03D0    870 45$:        MOVAB   HP$T_TYPE(R10),R1       ; Address of adapter type               ;[17
                50    81    9A  03D4    871             MOVZBL  (R1)+,R0                ; Length of adapter type
          53    000000C9'EF    9E  03D7    872             MOVAB   L^T_KNOWN_DEV,R3        ; Point to list of known types          [11]
                00000000'EF    16  03DE    873             JSB     SCAN$SEARCHLIST         ; Locate which type it is
                      29    12  03E4    874             BNEQ    60$                     ; Branch if unknown adapter..error
       52    000000C2'EF40    9A  03E6    875             MOVZBL  T_ADAPTER_TYPE[R0],R2   ; Get adapter type
                0C A5    52    91  03EE    876             CMPB    R2,DPT$B_ADPTYPE(R5)    ; Correct type of adapter?
                      1B    13  03F2    877             BEQL    60$                     ; Branch if correct
                      6C    9F  03F4    878 50$:        PUSHAB  HP$Q_DEVICE(AP)         ; Address of device name descriptor
                   26 AA    9F  03F6    879             PUSHAB  HP$T_TYPE(R10)          ; Address of adapter type
                00000095'EF    9F  03F9    880             PUSHAB  T_INVADP                ; Address of text
                   7E    01    9A  03FF    881             movzbl  #ds$k_printf,-(sp)      ; Use PRINTF                            [13]
                   7E    24    98  0402    882             cvtbl   #ds$k_type_qio_invadp,-(sp)     ; code                  [13]
          00000000'GF    05    FB  0405    883             CALLS   #5, G^DSX$PRINT         ; Print the error                       [13]
                      50    D4  040C    884             CLRL    R0                      ; Error return
                      04    04  040E    885             RET
                                040F    886
                54    18 AA    D0  040F    887 60$:        MOVL    HP$A_DEVICE(R10),R4     ; Get adapter CSR address
          53    54    04    0D  EF  0413    888             EXTZV   #13,#4,R4,R3            ; Get TR number or SLOT-16
                      06F6    30  0418    889             BSBW    FIND_ADP                ; Locate adapter control block
                   1E 50    E8  041B    890             BLBS    R0,70$                  ; Branch if found
                   3C'AF    9F  041E    891             PUSHAB  B^70$                   ; Dummy BSBW return
                                0421    892             CASE    R2,LIMIT=#AT$_MBA,TYPE=L,DISPLIST=-
                                0421    893                     <INI_MBADP,INI_UBADP,INI_DRADP>
             02'  00    52    CF  0421                CASEL   R2,#AT$_MBA,S^#<<30003$-30002$>/2>-1
                                0425            30002$:
                         007B' 0425                    .SIGNED_WORD    INI_MBADP-30002$
                         016F' 0427                    .SIGNED_WORD    INI_UBADP-30002$
                         01F1' 0429                    .SIGNED_WORD    INI_DRADP-30002$
                                042B            30003$:
                                042B    894             ERRSUP_S
                00000000'EF    DF  042B                        PUSHAL  $MODULE
                      00    DD  0431                        PUSHL   #0
                      03    DD  0433                        PUSHL   #$ER
          00000000'EF    03    FB  0435                        CALLS   #3, DS_ERRSUP
                                043C    895
```

```
                67   56   D0  043C   896 70$:    MOVL    R6,ACF$L_ADAPTER(R7)     ; Set ADP control block address
             04 A7   54   D0  043F   897         MOVL    R4,ACF$L_CONFIGREG(R7)   ; Set adapter csr address
          08 A7   24 AA   B0  0443   898         MOVW    HP$W_VECTOR(R10), -
                              0448   899                 ACF$W_AVECTOR(R7)        ; Set adapter vector
          0A A7   0B AB   90  0448   900         MOVB    HP$B_DRIVE(R11), -
                              044D   901                 ACF$B_AUNIT(R7)          ; Set adapter unit
          0C A7   18 AB   D0  044D   902         MOVL    HP$A_DEVICE(R11), -
                              0452   903                 ACF$L_CONTRLREG(R7)      ; Set device csr register
          10 A7   24 AB   B0  0452   904         MOVW    HP$W_VECTOR(R11), -
                              0457   905                 ACF$W_CVECTOR(R7)        ; Set Controller vector
                     05   12  0457   906         BNEQ    80$                      ; Branch if real vector
          10 A7   24 AA   B0  0459   907         MOVW    HP$W_VECTOR(R10), -
                              045E   908                 ACF$W_CVECTOR(R7)        ; Set Controller vector
          12 A7   0B AC   90  045E   909 80$:    MOVB    HP$B_DRIVE(AP), -
                              0463   910                 ACF$B_CUNIT(R7)          ; Set controller unit
          1C A7   16 A5   B0  0463   911         MOVW    DPT$W_MAXUNITS(R5), -
                              0468   912                 ACF$W_MAXUNITS(R7)       ; Set max number of units
             50   EC A5   9E  0468   913         MOVAB   -DVR$C_LEN(R5),R0        ; Base of DVR header
          13 A7   0B A0   90  046C   914         MOVB    DVR$B_CNUMVEC(R0), -
                              0471   915                 ACF$B_CNUMVEC(R7)        ; Set number of controller vectors
                     03   12  0471   916         BNEQ    90$                      ; Branch if really one there
                  13 A7   96  0473   917         INCB    ACF$B_CNUMVEC(R7)        ; Default is one
                  0B A7   94  0476   918 90$:    CLRB    ACF$B_AFLAG(R7)          ; Clear adapter flag
                              0479   919
   00000733'EF       67   FA  0479   920         CALLG   (R7),L^QIO$LOAD_DB       ; Load the database                [11]
                  0B A5   96  0480   921         INCB    DPT$B_REFC(R5)           ; Flag as successful
                  19 50   E8  0483   922         BLBS    R0,999$                  ; Branch if it worked
                  0B A5   97  0486   923         DECB    DPT$B_REFC(R5)           ; It failed, decrement the count
                     50   DD  0489   924         PUSHL   R0                       ; Save return code
                              048B   925         ERRSUP_S                         ; Spaz
        00000000'EF       DF  048B                        PUSHAL  $MODULE
                     00   DD  0491                        PUSHL   #0
                     04   DD  0493                        PUSHL   #$ER
   00000000'EF       03   FB  0495                        CALLS   #3, DS_ERRSUP
                     50 8ED0  049C   926         POPL    R0                       ; Restore reason
                        04  049F   927 999$:    RET
```

```
                                04A0   929          .SBTTL  INI_MBADP       BUILD ADP AND INITIALIZE MBA
                                04A0   930 ;+
                                04A0   931 ; INI_MBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                                04A0   932 ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                                04A0   933 ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                                04A0   934 ; BY CALLING MBA$INITIAL.
                                04A0   935 ;
                                04A0   936 ; INPUT:
                                04A0   937 ;       R3 - TR NUMBER OF MASSBUS ADAPTER
                                04A0   938 ;       R4 - CONFIGURATION REGISTER OF ADAPTER
                                04A0   939 ; OUTPUT:
                                04A0   940 ;       R6 - ADP ADDRESS
                                04A0   941 ;
                                04A0   942 ;-
                                04A0   943
                                04A0   944 INI_MBADP:
          07BC 8F    BB         04A0   945          PUSHR   #^M<R2,R3,R4,R5,R7,R8,R9,R10>
          57   53    D0         04A4   946          MOVL    R3,R7                   ; COPY TR NUMBER
                                04A7   947
          51   38    9A         04A7   948          MOVZBL  #CRB$C_LENGTH,R1        ; SET SIZE OF CRB
               0614  30         04AA   949          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR CRB
             36 50   E9         04AD   950          BLBC    R0,70$                  ; BRANCH IF ALLOCATION FAILED
          5A   52    D0         04B0   951          MOVL    R2,R10                  ; SAVE CRB ADDRESS
    08 AA   51    B0         04B3   952          MOVW    R1,CRB$W_SIZE(R10)      ; SET DATA STRUCTURE SIZE
                                04B7   953
    51   54 8F    9A         04B7   954          MOVZBL  #IDB$C_LENGTH+<8*4>,R1  ; SET SIZE TO ALLOCATE FOR IDB
               0603  30         04BB   955          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR IDB
             1F 50   E9         04BE   956          BLBC    R0,60$                  ; BRANCH IF ALLOCATION FAILED
          59   52    D0         04C1   957          MOVL    R2,R9                   ; COPY ADDRESS OF IDB
    08 A9   51    B0         04C4   958          MOVW    R1,IDB$W_SIZE(R9)       ; SET DATA STRUCTURE SIZE
                                04C8   959
          51   14    9A         04C8   960          MOVZBL  #ADP$C_MBAADPLEN,R1     ; SET SIZE TO ALLOCATE FOR ADP
               05F3  30         04CB   961          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR ADP
             09 50   E9         04CE   962          BLBC    R0,50$                  ; BRANCH IF ALLOCATION FAILED
          56   52    D0         04D1   963          MOVL    R2,R6                   ; ADDRESS OF ADP
    08 A6   51    B0         04D4   964          MOVW    R1,ADP$W_SIZE(R6)       ; SET DATA STRUCTURE SIZE
               14    11         04D8   965          BRB     100$                    ; ALL ALLOCATED FINISH INITIALIZATION
                                04DA   966
                                04DA   967
          50   59    D0         04DA   968 50$:     MOVL    R9,R0                   ; COPY ADDRESS
               0604  30         04DD   969          BSBW    QIO$DEALLOCATE          ; RELEASE
          50   5A    D0         04E0   970 60$:     MOVL    R10,R0                  ; COPY ADDRESS OF CRB
               05FE  30         04E3   971          BSBW    QIO$DEALLOCATE          ; RELEASE
                                04E6   972 70$:
50        0000'8F    3C         04E6   973          MOVZWL  #SS$_INSFMEM,R0         ; RETURN CODE
               0091  31         04EB   974          BRW     999$                    ; EXIT
```

```
                                04EE   976 100$:
                                04EE   977         DSBINT                          ; Disable interrupts
                     7E   12    DB  04EE                 MFPR    S^#PR$_IPL,-(SP)
                     12   1F    DA  04F1                 MTPR    #31,S^#PR$_IPL
                     3C   BB    04F4   978         PUSHR   #^M<R2,R3,R4,R5>        ; SAVE MOVC REGISTERS
     14 AA   00000584'EF  10'   28  04F6   979         MOVC3   S^#MBACRBLEN,L^MBACRB,CRB$L_INTD(R10) ; FILL WITH TEMPLATE      [11]
                     3C   BA    04FF   980         POPR    #^M<R2,R3,R4,R5>        ; RESTORE MOVC REGISTERS
                                0501   981
                                0501   982 ; Add offset of X100 or X300 based on SBIA number gleaned from R4.                ;[17
                                0501   983                                                                                  ;[17
                   0A 54   19   E0  0501   984         BBS     #25,R4,110$             ; CHECK WHICH SBIA 0 OR 1            ;[17
           50   00000100'EF47   DE  0505   985         MOVAL   SCB_BASE+^X100[R7],R0   ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA0);[1
                       08   11    050D   986         BRB     120$                    ; CONTINUE                           ;[17
           50   00000300'EF47   DE  050F   987 110$:   MOVAL   SCB_BASE+^X300[R7],R0   ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA1);[1
                     60   15 AA  DE  0517   988 120$:   MOVAL   CRB$L_INTD+1(R10),(R0)  ; CONNECT VECTOR TO CRB CODE          ;[17
                  40 A0   15 AA  DE  051B   989         MOVAL   CRB$L_INTD+1(R10),64(R0); SAME FOR
               0080 C0   15 AA  DE  0520   990         MOVAL   CRB$L_INTD+1(R10),128(R0); ALL FOUR
               00C0 C0   15 AA  DE  0526   991         MOVAL   CRB$L_INTD+1(R10),192(R0); VECTORS
                  0A AA   05    90  052C   992         MOVB    #DYN$C_CRB,CRB$B_TYPE(R10) ; SET CORRECT TYPE
                     6A   6A    DE  0530   993         MOVAL   CRB$L_WQFL(R10),CRB$L_WQFL(R10) ; INITIALIZE WAIT QUEUE HEADER
                  04 AA   6A    DE  0533   994         MOVAL   CRB$L_WQFL(R10),CRB$L_WQBL(R10) ; FLINK AND BLINK
                  0C AA   08    B0  0537   995         MOVW    #8,CRB$W_REFC(R10)      ; SET REFERENCE COUNT
                                053B   996
                  0A A9   09    90  053B   997         MOVB    #DYN$C_IDB,IDB$B_TYPE(R9); AND TYPE CODE
                  0C A9   08    9B  053F   998         MOVZBW  #8,IDB$W_UNITS(R9)      ; Set count of units
                     69   54    D0  0543   999         MOVL    R4,IDB$L_CSR(R9)        ; SET CSR ADDRESS TO CONFIG REG 0
                  1C AA   59    D0  0546  1000         MOVL    R9,CRB$L_INTD+VEC$L_IDB(R10)    ; SET ADDRESS OF IDB INTO CRB
        20 AA   00000000'9F   9E  054A  1001         MOVAB   @#MBA$INITIAL,CRB$L_INTD+VEC$L_INITIAL(R10) ; SET ADDRESS OF CONTROL
                                0552  1002
                  0A A6   01    90  0552  1003         MOVB    #DYN$C_ADP,ADP$B_TYPE(R6); TYPE CODE
                     66   54    D0  0556  1004         MOVL    R4,ADP$L_CSR(R6)        ; SET ADDRESS OF CONFIGURATION REGISTER
                  0C A6   57    B0  0559  1005         MOVW    R7,ADP$W_TR(R6)         ; SET TR NUMBER OF ADAPTER
                  0E A6   00    B0  055D  1006         MOVW    #AT$_MBA,ADP$W_ADPTYPE(R6); SET ADAPTER TYPE TO MBA
                  10 A6   5A    D0  0561  1007         MOVL    R10,ADP$L_CRB(R6)       ; POINT ADP TO CRB
                  28 AA   56    D0  0565  1008         MOVL    R6,CRB$L_INTD+VEC$L_ADP(R10)    ; SET CRB POINTER TO ADP
                  10 A9   56    D0  0569  1009         MOVL    R6,IDB$L_ADP(R9)        ; AND INTO IDB
                     55   59    D0  056D  1010         MOVL    R9,R5                   ; ADDRESS OF IDB
                        0587    30  0570  1011         BSBW    ADPLINK                 ; LINK ADP TO END OF CHAIN
                00000000'EF   16  0573  1012         JSB     MBA$INITIAL             ; INITIALIZE MASSBUS ADAPTER
                                0579  1013         ENBINT                          ; Re-enable interrupts
                     12   8E    DA  0579                 MTPR    (SP)+,S^#PR$_IPL
                     50   00'   D0  057C  1014         MOVL    S^#SS$_NORMAL,R0        ; SUCCESSFUL
                                057F  1015
                  07BC 8F       BA  057F  1016 999$:   POPR    #^M<R2,R3,R4,R5,R7,R8,R9,R10>; RESTORE ALL REGISTERS
                                05  0583  1017         RSB                             ; RETURN
                                0584  1018
                                0584  1019 MBACRB:                                   ; SKELETON MBA CRB
                     3C   BB    0584  1020         PUSHR   #^M<R2,R3,R4,R5>        ; SAVE REGISTERS
                00000000'9F   16  0586  1021         JSB     @#MBA$INT               ; CALL INTERRUPT DISPATCHER
                        00000000  058C  1022         .LONG   0                       ; CRB$L_INTD+VEC$L_INTD
                        00000000' 0590  1023         .LONG   MBA$INITIAL             ; CRB$L_INTD+VEC$L_INITIAL
                        00000010  0594  1024 MBACRBLEN=.-MBACRB                      ;
```

```
                                0594  1026          .SBTTL  INI_UBADP       BUILD ADP AND INITIALIZE UBA
                                0594  1027 ;+
                                0594  1028 ; INI_UBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
                                0594  1029 ; DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS.  A CALL IS
                                0594  1030 ; THEN MADE TO UBA$INITIAL TO INITIALIZE THE ADAPTER HARDWARE.
                                0594  1031 ;
                                0594  1032 ; INPUT:
                                0594  1033 ;       R3 - TR NUMBER
                                0594  1034 ;       R4 - CONFIGURATION REGISTER OF ADAPTER
                                0594  1035 ; OUTPUT:
                                0594  1036 ;       R6 - ADP ADDRESS
                                0594  1037 ; REGISTERS:
                                0594  1038 ;       R4 - ADAPTER CONFIGURATION REGISTER ADDR
                                0594  1039 ;       R6 - ADP ADDRESS
                                0594  1040 ;       R7 - TR NUMBER
                                0594  1041 ;-
                    00000080    0594  1042          NUMUBAVEC = 128                 ; ALLOW FOR 128 UNIBUS VECTORS
                                0594  1043
                                0594  1044 INI_UBADP:
                00BC 8F   BB    0594  1045          PUSHR   #^M<R2,R3,R4,R5,R7>     ; SAVE R2-R7
                57   53   D0    0598  1046          MOVL    R3,R7                   ; COPY TR NUMBER
            51  0270'8F   3C    059B  1047          MOVZWL  #<ADP$C_UBAADPLEN+UBINTSZ+<NUMUBAVEC*4>>,R1
                                05A0  1048                                          ; SET SIZE OF BLOCK TO ALLOCATE
                  051E    30    05A0  1049          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR ADP
                03 50     E8    05A3  1050          BLBS    R0,10$                  ; EXIT IF ERROR
                  0068    31    05A6  1051          BRW     60$                     ; EXIT ON ERROR
                                05A9  1052 10$:
                                05A9  1053          DSBINT                          ; Disable interrupts
                   7E   12   DB 05A9                        MFPR    S^#PR$_IPL,-(SP)
                   12   1F   DA 05AC                        MTPR    #31,S^#PR$_IPL
                   56   52   D0 05AF  1054          MOVL    R2,R6                   ; COPY ADP ADDRESS
                08 A6   51   B0 05B2  1055          MOVW    R1,ADP$W_SIZE(R6)       ; SET SIZE INTO ADP BLOCK
                0A A6   01   90 05B6  1056          MOVB    #DYN$C_ADP,ADP$B_TYPE(R6); AND SET TYPE OF BLOCK
                0E A6   01   B0 05BA  1057          MOVW    #AT$_UBA,ADP$W_ADPTYPE(R6) ; SET TYPE OF ADAPTER
                   66   54   D0 05BE  1058          MOVL    R4,ADP$L_CSR(R6)        ; SET VA OF CONFIGURATION REG
                0C A6   57   B0 05C1  1059          MOVW    R7,ADP$W_TR(R6)         ; SET TR NUMBER FOR ADAPTER
                50    14 A6  DE 05C5  1060          MOVAL   ADP$L_DPQFL(R6),R0      ; ADDRESS OF DATA PATH WAIT QUEUE
                   60   50   D0 05C9  1061          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
                04 A0   50   D0 05CC  1062          MOVL    R0,4(R0)                ;
                50    1C A6  DE 05D0  1063          MOVAL   ADP$L_MRQFL(R6),R0      ; ADDRESS OF MAP WAIT QUEUE
                   60   50   D0 05D4  1064          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
                04 A0   50   D0 05D7  1065          MOVL    R0,4(R0)                ;
                   50   1E   D0 05DB  1066          MOVL    #30,R0                  ; NUMBER OF WORDS-1 IN MAP BITMAP
             26 A640    01   AE 05DE  1067 20$:     MNEGW   #1,ADP$W_MRBITMAP(R6)[R0] ; SET 16 BITS OF BITMAP
                   F8   50   F4 05E3  1068          SOBGEQ  R0,20$                  ; FILL 31 * 16  => 496 BITS
                   04 A6   D4 05E6  1069          CLRL    ADP$L_LINK(R6)          ; ZAP ADAPTER CHAIN LINK
             00000000'EF   16 05E9  1070          JSB     UBA$INITIAL             ; AND INITIALIZE ADAPTER        ; 19
                    0508   30 05EF  1071          BSBW    ADPLINK                 ; LINK ADP TO END OF LIST
             00000000'EF   16 05F2  1072          JSB     UBADP_CPU               ; CALL CPU DEPENDANT CODE       ; 19
              50    10 A6  D0 05F8  1073          MOVL    ADP$L_VECTOR(R6),R0     ; GET ADDRESS OF VECTOR
            51    0080 8F   3C 05FC  1074          MOVZWL  #NUMUBAVEC,R1           ; SET NUMBER OF VECTORS
         80  00000000'EF   9E 0601  1075 50$:     MOVAB   UBA$UNEXINT,(R0)+       ; FILL VECTOR WITH UNEXPECTED INTERRUPT
                   F6   51   F5 0608  1076          SOBGTR  R1,50$                  ; DO ALL VECTOR POSITIONS
                                060B  1077          ENBINT                          ; Enable interrupts
                   12   8E   DA 060B                        MTPR    (SP)+,S^#PR$_IPL
                   50   01   D0 060E  1078          MOVL    #1,R0                   ; SUCCESS
                                0611  1079 60$:
```

```
                00BC 8F    BA  0611  1080          POPR    #^M<R2,R3,R4,R5,R7>     ; RESTORE R2-R7
                           05  0615  1081          RSB                             ; AND RETURN
```

```
                                0616  1083          .SBTTL  INI_DRADP       BUILD ADP AND INITIALIZE DRA
                                0616  1084 ;+
                                0616  1085 ; INI_DRADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                                0616  1086 ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                                0616  1087 ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                                0616  1088 ; BY CALLING DRA$INITIAL.
                                0616  1089 ;
                                0616  1090 ; INPUT:
                                0616  1091 ;       R3 - TR NUMBER OF MASSBUS ADAPTER
                                0616  1092 ;       R4 - CONFIGURATION REGISTER OF ADAPTER
                                0616  1093 ; OUTPUT:
                                0616  1094 ;       R6 - ADP ADDRESS
                                0616  1095 ;
                                0616  1096 ;-
                                0616  1097
                                0616  1098 INI_DRADP:
            07BC 8F     BB      0616  1099          PUSHR   #^M<R2,R3,R4,R5,R7,R8,R9,R10>
            57   53     D0      061A  1100          MOVL    R3,R7                   ; COPY TR NUMBER
                                061D  1101
            51   38     9A      061D  1102          MOVZBL  #CRB$C_LENGTH,R1        ; SET SIZE OF CRB
                 049E   30      0620  1103          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR CRB
               35 50    E9      0623  1104          BLBC    R0,70$                  ; BRANCH IF ALLOCATION FAILED
            5A   52     D0      0626  1105          MOVL    R2,R10                  ; SAVE CRB ADDRESS
         08 AA   51     B0      0629  1106          MOVW    R1,CRB$W_SIZE(R10)      ; SET DATA STRUCTURE SIZE
                                062D  1107
            51   34     9A      062D  1108          MOVZBL  #IDB$C_LENGTH,R1        ; SET SIZE TO ALLOCATE FOR IDB
                 048E   30      0630  1109          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR IDB
               1F 50    E9      0633  1110          BLBC    R0,60$                  ; BRANCH IF ALLOCATION FAILED
            59   52     D0      0636  1111          MOVL    R2,R9                   ; COPY ADDRESS OF IDB
         08 A9   51     B0      0639  1112          MOVW    R1,IDB$W_SIZE(R9)       ; SET DATA STRUCTURE SIZE
                                063D  1113
            51   14     9A      063D  1114          MOVZBL  #ADP$C_DRADPLEN,R1      ; SET SIZE TO ALLOCATE FOR ADP
                 047E   30      0640  1115          BSBW    QIO$ALLOCATE            ; ALLOCATE SPACE FOR ADP
               09 50    E9      0643  1116          BLBC    R0,50$                  ; BRANCH IF ALLOCATION FAILED
            56   52     D0      0646  1117          MOVL    R2,R6                   ; ADDRESS OF ADP
         08 A6   51     B0      0649  1118          MOVW    R1,ADP$W_SIZE(R6)       ; SET DATA STRUCTURE SIZE
                 14     11      064D  1119          BRB     100$                    ; ALL ALLOCATED FINISH INITIALIZATION
                                064F  1120
                                064F  1121
            50   59     D0      064F  1122 50$:     MOVL    R9,R0                   ; COPY ADDRESS
                 048F   30      0652  1123          BSBW    QIO$DEALLOCATE          ; RELEASE
            50   5A     D0      0655  1124 60$:     MOVL    R10,R0                  ; COPY ADDRESS OF CRB
                 0489   30      0658  1125          BSBW    QIO$DEALLOCATE          ; RELEASE
                                065B  1126 70$:
      50    0000'8F     3C      065B  1127          MOVZWL  #SS$_INSFMEM,R0         ; RETURN CODE
                 008F   31      0660  1128          BRW     999$                    ; EXIT
```

```
                                0663  1130 100$:
                                0663  1131          DSBINT                          ; Disable interrupts
                7E   12   DB    0663                        MFPR    S^#PR$_IPL,-(SP)
                12   1F   DA    0666                        MTPR    #31,S^#PR$_IPL
                     3C   BB    0669  1132          PUSHR   #^M<R2,R3,R4,R5>         ; SAVE MOVC REGISTERS
          14 AA   06F7'CF   10'  28    066B  1133          MOVC3   S^#DRACRBLEN,W^DRACRB,CRB$L_INTD(R10) ; FILL WITH TEMPLATE
                     3C   BA    0672  1134          POPR    #^M<R2,R3,R4,R5>         ; RESTORE MOVC REGISTERS
                                0674  1135
                                0674  1136 ; Add offset of X100 or X300 based on SBIA number gleaned from R4.              [18]
                                0674  1137
              0A 54    19   E0    0674  1138          BBS     #25,R4,110$              ; CHECK WHICH SBIA 0 OR 1                [18]
        50   00000100'EF47   DE    0678  1139          MOVAL   SCB_BASE+^X100[R7],R0    ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA0)[18
                     08   11    0680  1140          BRB     120$                     ; CONTINUE                               [18]
        50   00000300'EF47   DE    0682  1141 110$:    MOVAL   SCB_BASE+^X300[R7],R0    ; COMPUTE ADDRESS OF FIRST VECTOR (SBIA1)[18
              60    15 AA    DE    068A  1142 120$:    MOVAL   CRB$L_INTD+1(R10),(R0)   ; CONNECT VECTOR TO CRB CODE
           40 A0    15 AA    DE    068E  1143          MOVAL   CRB$L_INTD+1(R10),64(R0); SAME FOR
        0080 C0     15 AA    DE    0693  1144          MOVAL   CRB$L_INTD+1(R10),128(R0); ALL FOUR
        00C0 C0     15 AA    DE    0699  1145          MOVAL   CRB$L_INTD+1(R10),192(R0); VECTORS
              0A AA    05    90    069F  1146          MOVB    #DYN$C_CRB,CRB$B_TYPE(R10) ; SET CORRECT TYPE
                 6A    6A    DE    06A3  1147          MOVAL   CRB$L_WQFL(R10),CRB$L_WQFL(R10) ; INITIALIZE WAIT QUEUE HEADER
              04 AA    6A    DE    06A6  1148          MOVAL   CRB$L_WQFL(R10),CRB$L_WQBL(R10) ; FLINK AND BLINK
              0C AA    08    B0    06AA  1149          MOVW    #8,CRB$W_REFC(R10)       ; SET REFERENCE COUNT
                                06AE  1150
              0A A9    09    90    06AE  1151          MOVB    #DYN$C_IDB,IDB$B_TYPE(R9); AND TYPE CODE
              0C A9    01    9B    06B2  1152          MOVZBW  #1,IDB$W_UNITS(R9)       ; Set count of units
                 69    54    D0    06B6  1153          MOVL    R4,IDB$L_CSR(R9)         ; SET CSR ADDRESS TO CONFIG REG 0
              1C AA    59    D0    06B9  1154          MOVL    R9,CRB$L_INTD+VEC$L_IDB(R10)    ; SET ADDRESS OF IDB INTO CRB
        20 AA   00000000'9F   9E    06BD  1155          MOVAB   @#DR$INITIAL,CRB$L_INTD+VEC$L_INITIAL(R10) ; SET ADDRESS OF CONTROLL
                                06C5  1156
              0A A6    01    90    06C5  1157          MOVB    #DYN$C_ADP,ADP$B_TYPE(R6); TYPE CODE
                 66    54    D0    06C9  1158          MOVL    R4,ADP$L_CSR(R6)         ; SET ADDRESS OF CONFIGURATION REGISTER
              0C A6    57    B0    06CC  1159          MOVW    R7,ADP$W_TR(R6)          ; SET TR NUMBER OF ADAPTER
              0E A6    02    B0    06D0  1160          MOVW    #AT$_DR,ADP$W_ADPTYPE(R6); SET ADAPTER TYPE TO DRA
              10 A6    5A    D0    06D4  1161          MOVL    R10,ADP$L_CRB(R6)        ; POINT ADP TO CRB
              28 AA    56    D0    06D8  1162          MOVL    R6,CRB$L_INTD+VEC$L_ADP(R10)    ; SET CRB POINTER TO ADP
              10 A9    56    D0    06DC  1163          MOVL    R6,IDB$L_ADP(R9)         ; AND INTO IDB
                 55    59    D0    06E0  1164          MOVL    R9,R5                    ; ADDRESS OF IDB
                    0414   30    06E3  1165          BSBW    ADPLINK                  ; LINK ADP TO END OF CHAIN
           00000000'EF   16    06E6  1166          JSB     DR$INITIAL               ; INITIALIZE MASSBUS ADAPTER
                                06EC  1167          ENBINT                           ; Re-enable interrupts
                12   8E   DA    06EC                        MTPR    (SP)+,S^#PR$_IPL
                50   00'  D0    06EF  1168          MOVL    S^#SS$_NORMAL,R0         ; SUCCESSFUL
                                06F2  1169
                07BC 8F   BA    06F2  1170 999$:    POPR    #^M<R2,R3,R4,R5,R7,R8,R9,R10>; RESTORE ALL REGISTERS
                     05   06F6  1171          RSB                              ; RETURN
                                06F7  1172
                                06F7  1173 DRACRB:                                  ; SKELETON DRA CRB
                     3C   BB    06F7  1174          PUSHR   #^M<R2,R3,R4,R5>         ; SAVE REGISTERS
           00000000'9F   16    06F9  1175          JSB     @#DR$INT                 ; CALL INTERRUPT DISPATCHER
                00000000    06FF  1176          .LONG   0                        ; CRB$L_INTD+VEC$L_INTD
                00000000'   0703  1177          .LONG   DR$INITIAL               ; CRB$L_INTD+VEC$L_INITIAL
                00000010    0707  1178 DRACRBLEN=.-DRACRB                       ;
```

```
                                  0707          .SBTTL  MAXIMIZE ACCESS MODE
                                  0707  ;+
                                  0707  ; EXE$MAXACMODE - MAXIMIZE ACCESS MODE
                                  0707  ;
                                  0707  ; THIS ROUTINE IS CALLED TO MAXIMIZE A SPECIFIED ACCESS MODE WITH THE PREVIOUS
                                  0707  ; MODE FIELD OF THE CURRENT PSL.
                                  0707  ;
                                  0707  ; INPUTS:
                                  0707  ;
                                  0707  ;       R0 = ACCESS MODE TO MAXIMIZE WITH PREVIOUS MODE FIELD OF PSL.
                                  0707  ;
                                  0707  ; OUTPUTS:
                                  0707  ;
                                  0707  ;       THE SPECIFIED ACCESS MODE IS MAXIMIZED WITH THE PREVIOUS MODE FIELD
                                  0707  ;       OF THE CURRENT PSL AND RETURNED IN REGISTER R0.
                                  0707  ;
                                  0707  ;       REGISTERS R2 AND R3 ARE PRESERVED ACROSS CALL.
                                  0707  ;-
                                  0707
                                  0707  EXE$MAXACMODE::                                  ;MAXIMIZE ACCESS MODE
                        51   DC   0707          MOVPSL  R1                                ;READ CURRENT PSL
     50   51   02     16   ED   0709          CMPZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R0 ;COMPARE WITH PREVIOUS MODE
                        05   15   070E          BLEQ    10$                               ;IF LEQ SPECIFIED ACCESS MODE LESS PRIVILEGE
     50   51   02     16   EF   0710          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R0 ;EXTRACT PREVIOUS MODE FIELD
                             05   0715  10$:    RSB                                       ;
```

```
                        0716  1206          .SBTTL  BUG$CHECK       Bugcheck notification routine
                        0716  1207 ;++
                        0716  1208 ;
                        0716  1209 ; Functional Description:
                        0716  1210 ;
                        0716  1211 ;       This routine types the text of the VMS bugcheck and is equivalent
                        0716  1212 ;       to a supervisor ERRSUP
                        0716  1213 ;
                        0716  1214 ; Calling Sequence:
                        0716  1215 ;       BSBW    BUG$CHECK
                        0716  1216 ;       .ASCIC  "error text"
                        0716  1217 ;
                        0716  1218 ; Input Parameters:
                        0716  1219 ;       a(SP)   -> ASCIC explaination
                        0716  1220 ;
                        0716  1221 ; Implicit Inputs:
                        0716  1222 ;       NONE
                        0716  1223 ;
                        0716  1224 ; Output Parameters:
                        0716  1225 ;       NONE
                        0716  1226 ;
                        0716  1227 ; Implicit Outputs:
                        0716  1228 ;       NONE
                        0716  1229 ;
                        0716  1230 ; Side Effects:
                        0716  1231 ;       NONE
                        0716  1232 ;
                        0716  1233 ;--
                        0716  1234
                        0716  1235 BUG$CHECK::
        50   6E   D0  0716  1236          MOVL    (SP),R0                         ; GET ADDRESS OF TEXT
0000FE00 9F   01   C8  0719  1237          BISL    #DSA$M_HALT,a#DSA$GL_FLAGS ; Force Halt on error                [12]
                        0720  1238          ERRSUP_S ,(R0)
     00000000'EF   DF  0720                       PUSHAL  $MODULE
                60   DF  0726                       PUSHAL  (R0)
                05   DD  0728                       PUSHL   #$ER
00000000'EF     03   FB  072A                       CALLS   #3, DS_ERRSUP
                E3   11  0731  1239          BRB     BUG$CHECK                       ; CAN'T CONTINUE
```

```
0733  1241          .SBTTL  QIO$LOAD_DB     Insert unit into data base
0733  1242 ;++
0733  1243 ; FUNCTIONAL DESCRIPTION:
0733  1244 ;
0733  1245 ;       This routine will load a single unit into an already existing
0733  1246 ;       database.
0733  1247 ;
0733  1248 ; CALLING SEQUENCE:
0733  1249 ;
0733  1250 ;       CALL QIO$LOAD_DB(ACF_LIST)
0733  1251 ;
0733  1252 ; INPUT PARAMETERS:
0733  1253 ;
0733  1254 ;       ACF_LIST -
0733  1255 ;
0733  1256 ;       ACF$L_ADAPTER(AP) = ADDRESS OF ADAPTER CONTROL BLOCK
0733  1257 ;       ACF$L_CONFIGREG(AP) = ADDRESS OF CONFIGURATION STATUS REGISTER
0733  1258 ;       ACF$W_AVECTOR(AP) = OFFSET TO ADAPTER INTERRUPT VECTOR (SCB)
0733  1259 ;       ACF$B_AUNIT(AP) = ADAPTER UNIT NUMBER
0733  1260 ;       ACF$B_AFLAG(AP) = ADAPTER GENERATION CONTROL FLAGS
0733  1261 ;       ACF$L_CONTRLREG(AP) = ADDRESS OF CONTROL REGISTER
0733  1262 ;       ACF$W_CVECTOR(AP) = OFFSET TO CONTROLLER INTERRUPT VECTOR (TABLE)
0733  1263 ;       ACF$B_CUNIT(AP) = CONTROLLER UNIT NUMBER
0733  1264 ;       ACF$B_CNUMVEC(AP) = NUMBER OF CONTROLLER VECTORS
0733  1265 ;       ACF$L_DEVNAME(AP) = ADDRESS OF DEVICE NAME COUNTED STRING
0733  1266 ;       ACF$L_DRVNAME(AP) = ADDRESS OF DRIVER NAME COUNTED STRING
0733  1267 ;
0733  1268 ; IMPLICIT INPUTS:
0733  1269 ;
0733  1270 ;       NONE
0733  1271 ;
0733  1272 ; OUTPUT PARAMETERS:
0733  1273 ;
0733  1274 ;       NONE
0733  1275 ;
0733  1276 ; IMPLICIT OUTPUTS:
0733  1277 ;
0733  1278 ;       DATABASE LOADED
0733  1279 ;
0733  1280 ; COMPLETION CODES:
0733  1281 ;
0733  1282 ;       R0 = STATUS OF OPERATION
0733  1283 ;
0733  1284 ; SIDE EFFECTS:
0733  1285 ;
0733  1286 ;       NONE
0733  1287 ;
0733  1288 ;--
```

```
                        0733  1290 QIO$LOAD_DB:
                   0FFC 0733  1291         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0735  1292
          00000005'EF   94 0735  1293         CLRB    L^LOAD_FLAGS            ;CLEAR LOADER FLAGS                     [11]
                030D    30 0738  1294         BSBW    IOGEN$LOCK_IODB         ;LOCK THE I/O DATABASE
                        073B  1295 ;
                        073B  1296 ; LOCATE THE DPT FOR THIS DRIVER
                        073B  1297 ;
           53   18 AC   D0 073B  1298         MOVL    ACF$L_DRVNAME(AP),R3    ; Address of driver name
              52   83   9A 0742  1299         MOVZBL  (R3)+,R2                ; Get length of driver
                 FA19    30 0745  1300         BSBW    LOC$DPT                 ; Locate driver dispatch table
                21 50   E8 0748  1301         BLBS    R0,15$                  ; Branch if found
                        074B  1302
           51   14 AC   9E 074B  1303         MOVAB   ACF$L_DEVNAME(AP),R1    ; ADDRESS OF ASCIC DEVICE NAME
              50   81   9A 074F  1304         MOVZBL  (R1)+,R0                ; GET LENGTH
                   03   BB 0752  1305         PUSHR   #^M<R0,R1>              ; STACK LENGTH,ADDRESS
          00000046'EF   9F 0754  1306         PUSHAB  L^T_NODRIVER            ; ADDRESS OF ASCIC EDT TEXT             [11]
                7E   01 9A 075A  1307         movzbl  #ds$k_printf,-(sp)      ; Use PRINTF                            [13]
                7E   22 98 075D  1308         cvtbl   #ds$k_type_qio_nodriver,-(sp)   ; code                          [13]
     00000000'GF   05   FB 0760  1309         CALLS   #5, G^DSX$PRINT         ; DRIVER FOR DEVICE NOT FOUND           [13]
                   50   D4 0767  1310         CLRL    R0                      ; ERROR
                026D    31 0769  1311         BRW     DB_ERROR
                        076C  1312 ;
                        076C  1313 ; CHECK IF DEVICE DATABASE ALREADY LOADED
                        076C  1314 ;
           55   14 AC   D0 076C  1315 15$:    MOVL    ACF$L_DEVNAME(AP),R5    ;GET ADDR OF DEVICE NAME
              54   85   9A 0770  1316         MOVZBL  (R5)+,R4                ;GET SIZE OF DEVICE NAME
        5A    00000000'GF   DE 0773  1317         MOVAL   G^IOC$GL_DEVLIST,R10    ;GET ADDR OF DEVICE LISTHEAD
                   6A   D5 077A  1318 20$:    TSTL    DDB$L_LINK(R10)         ;IS THERE A NEXT DDB?
                   31   13 077C  1319         BEQL    CREATE_DDB              ;BR IF NOT - CREATE A NEW DDB
                5A   6A   D0 077E  1320         MOVL    DDB$L_LINK(R10),R10     ;GET ADDR OF NEXT DDB
           51   14 AA   9E 0781  1321         MOVAB   DDB$T_NAME(R10),R1      ;GET ADDR OF DEVICE NAME
              50   81   9A 0785  1322         MOVZBL  (R1)+,R0                ;GET SIZE OF DEVICE NAME
65   54   00   61   50   2D 0788  1323         CMPC5   R0,(R1),#0,R4,(R5)      ;COMPARE DEVICE NAMES
                   EA   12 078E  1324         BNEQ    20$                     ;BR IF NOT EQUAL
                        0790  1325
           50   04 AA   D0 0790  1326         MOVL    DDB$L_UCB(R10),R0       ;GET ADDR OF FIRST UCB
           59   20 A0   D0 0794  1327         MOVL    UCB$L_CRB(R0),R9        ;GET ADDR OF CRB
           58   1C A9   D0 0798  1328         MOVL    CRB$L_INTD+VEC$L_IDB(R9),R8 ;GET ADDR OF IDB
        48 A0   12 AC   91 079C  1329 30$:    CMPB    ACF$B_CUNIT(AP),UCB$W_UNIT(R0) ;IS UCB ALREADY LOADED?
                   03   12 07A1  1330         BNEQ    40$                     ;BR IF NOT
                022C    31 07A3  1331         BRW     DB_EXIT                 ;ELSE NOTHING TO DO - EXIT
           50   2C A0   D0 07A6  1332 40$:    MOVL    UCB$L_LINK(R0),R0       ;GET ADDR OF NEXT UCB
                   F0   12 07AA  1333         BNEQ    30$                     ;BR IF THERE IS ONE
                 0129    31 07AC  1334         BRW     CREATE_UCB              ;CREATE THE NEW UCB
                        07AF  1335 ;
                        07AF  1336 ; CREATE THE DDB
                        07AF  1337 ;
                        07AF  1338 CREATE_DDB:
           51   34   9A 07AF  1339         MOVZBL  #DDB$K_LENGTH,R1        ;GET SIZE OF DDB
                030C    30 07B2  1340         BSBW    QIO$ALLOCATE            ;CREATE THE DDB
              03 50   E8 07B5  1341         BLBS    R0,10$                  ;BR IF SUCCESS
                021E    31 07B8  1342         BRW     DB_ERROR                ;ELSE - ERROR
     00000005'EF   01   88 07BB  1343 10$:    BISB    #LOAD_M_DDB,L^LOAD_FLAGS ;SET DDB LOADED FLAG                   [11]
     00000006'EF   6A   DE 07C2  1344         MOVAL   DDB$L_LINK(R10),L^DDB_BLINK ;SAVE DDB BACKWARD LINK             [11]
                6A   52   D0 07C9  1345         MOVL    R2,DDB$L_LINK(R10)      ;SET LINK TO THIS DDB
                5A   52   D0 07CC  1346         MOVL    R2,R10                  ;GET ADDR OF DDB
```

```
                08 AA    51    B0  07CF  1347          MOVW    R1,DDB$W_SIZE(R10)          ;SET SIZE
                0A AA    06    90  07D3  1348          MOVB    #DYN$C_DDB,DDB$B_TYPE(R10) ;SET TYPE
                     0B AB    96  07D7  1349          INCB    DPT$B_REFC(R11)             ;INCREMENT DPT REFERENCE COUNT
                50    14 BC    9A  07DA  1350          MOVZBL  @ACF$L_DEVNAME(AP),R0       ;GET SIZE OF DEVICE NAME
                         50    D6  07DE  1351          INCL    R0                          ;ADD 1 BYTE FOR COUNT
          14 AA 14 BC    50    28  07E0  1352          MOVC    R0,@ACF$L_DEVNAME(AP),DDB$T_NAME(R10) ;SET DEVICE NAME
                50    18 BC    9A  07E6  1353          MOVZBL  @ACF$L_DRVNAME(AP),R0       ;GET SIZE OF DRIVER NAME
                         50    D6  07EA  1354          INCL    R0                          ;ADD 1 BYTE FOR COUNT
          24 AA 18 BC    50    28  07EC  1355          MOVC    R0,@ACF$L_DRVNAME(AP),DDB$T_DRVNAME(R10) ;SET DRIVER NAME
                                   07F2  1356 ;
                                   07F2  1357 ; CHECK IF CRB/IDB NEED TO BE CREATED
                                   07F2  1358 ;
             14 0D AB    00    E0  07F2  1359          BBS     #DPT$V_SUBCNTRL,DPT$B_FLAGS(R11),CREATE_CRB ;BR IF SUBCONTROLLER
                0C AB    01    91  07F7  1360          CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
                         0E    13  07FB  1361          BEQL    CREATE_CRB                  ;BR IF YES
                50    6C    D0  07FD  1362          MOVL    ACF$L_ADAPTER(AP),R0        ;GET ADDR OF ADP
                59    10 A0    D0  0800  1363          MOVL    ADP$L_CRB(R0),R9            ;GET ADDR OF EXISTING CRB
                58    1C A9    D0  0804  1364          MOVL    CRB$L_INTD+VEC$L_IDB(R9),R8 ;GET ADDR OF IDB
                      00CD     31  0808  1365          BRW     CREATE_UCB                  ;CREATE A NEW UCB
                                   080B  1366 ;
                                   080B  1367 ; CREATE CRB
                                   080B  1368 ;
                                   080B  1369 CREATE_CRB:
                51    13 AC    9A  080B  1370          MOVZBL  ACF$B_CNUMVEC(AP),R1        ;GET NUMBER OF INT VECTORS
                         51    D7  080F  1371          DECL    R1                          ;ONE IS ALWAYS ASSUMED
                   51    24    A4  0811  1372          MULW    #VEC$K_LENGTH,R1            ;COMPUTE SIZE OF EXTRA DISPATCHERS
                   51    38    A0  0814  1373          ADDW    #CRB$K_LENGTH,R1            ;COMPUTE TOTAL SIZE OF CRB
                      02A7     30  0817  1374          BSBW    QIO$ALLOCATE                ;ALLOCATE THE CRB
                   03 50       E8  081A  1375          BLBS    R0,10$                      ;BR IF SUCCESS
                      01B9     31  081D  1376          BRW     DB_ERROR                    ;ELSE - ERROR
00000005'EF       02    88  0820  1377 10$:     BISB    #LOAD_M_CRB,L^LOAD_FLAGS    ;SET CRB LOADED FLAG                    [11]
                59    52    D0  0827  1378          MOVL    R2,R9                       ;GET ADDR OF CRB
                08 A9    51    B0  082A  1379          MOVW    R1,CRB$W_SIZE(R9)           ;SET SIZE
                0A A9    05    90  082E  1380          MOVB    #DYN$C_CRB,CRB$B_TYPE(R9) ;SET TYPE
                69    69    DE  0832  1381          MOVAL   CRB$L_WQFL(R9),CRB$L_WQFL(R9) ;SET WAIT QUEUE LISTHEAD
                04 A9    69    DE  0835  1382          MOVAL   CRB$L_WQFL(R9),CRB$L_WQBL(R9) ;..
             19 0D AB    00    E1  0839  1383          BBC     #DPT$V_SUBCNTRL,DPT$B_FLAGS(R11),CREATE_IDB ;BR IF NOT SUBCONTROLLER
                50    6C    D0  083E  1384          MOVL    ACF$L_ADAPTER(AP),R0        ;GET ADDR OF ADAPTER CONTROL BLOCK
                51    10 A0    D0  0841  1385          MOVL    ADP$L_CRB(R0),R1            ;GET ADDR OF SECONDARY CRB
                10 A9    51    D0  0845  1386          MOVL    R1,CRB$L_LINK(R9)           ;SET ADDR OF SECONDARY CRB
                52    1C A1    D0  0849  1387          MOVL    CRB$L_INTD+VEC$L_IDB(R1),R2 ;GET ADDR OF SECONDARY IDB
                53    0A AC    9A  084D  1388          MOVZBL  ACF$B_AUNIT(AP),R3          ;GET ADAPTER UNIT NUMBER
             14 A243  15 A9    9E  0851  1389          MOVAB   CRB$L_INTD+1(R9),IDB$L_UCBLST(R2)[R3] ;SET ADDR OF DISPATCHER
                                   0857  1390 ;
                                   0857  1391 ; CREATE IDB
                                   0857  1392 ;
                                   0857  1393 CREATE_IDB:
                   51    34    9A  0857  1394          MOVZBL  #IDB$K_LENGTH,R1            ;SET LENGTH OF IDB
                      0264     30  085A  1395          BSBW    QIO$ALLOCATE                ;ALLOCATE THE IDB
                   03 50       E8  085D  1396          BLBS    R0,10$                      ;BR IF SUCCESS
                      0176     31  0860  1397          BRW     DB_ERROR                    ;ELSE - ERROR
00000005'EF       04    88  0863  1398 10$:     BISB    #LOAD_M_IDB,L^LOAD_FLAGS    ;SET IDB LOADED FLAG                    [11]
                58    52    D0  086A  1399          MOVL    R2,R8                       ;GET ADDR OF IDB
                08 A8    51    B0  086D  1400          MOVW    R1,IDB$W_SIZE(R8)           ;SET SIZE
                0A A8    09    90  0871  1401          MOVB    #DYN$C_IDB,IDB$B_TYPE(R8) ;SET TYPE
                10 A8    6C    D0  0875  1402          MOVL    ACF$L_ADAPTER(AP),IDB$L_ADP(R8) ;SET ADDR OF ADP
                68    0C AC    D0  0879  1403          MOVL    ACF$L_CONTRLREG(AP),IDB$L_CSR(R8) ;SET ADDR OF CSR
```

```
                0C A8   08   9B  087D  1404          MOVZBW  #8,IDB$W_UNITS(R8)      ; Always set up for 8 units
                                  0881  1405 ;
                                  0881  1406 ; CREATE INTERRUPT VECTOR DISPATCHER(S)
                                  0881  1407 ;
                                  0881  1408 CREATE_VEC:
                56   10 AC   32   0881  1409          CVTWL   ACF$W_CVECTOR(AP),R6    ;GET VECTOR TABLE OFFSET
                              0A   14   0885  1410          BGTR    5$                      ;BR IF VECTOR SPECIFIED
             50   007C802A 8F   D0   0887  1411          MOVL    #SYSG$_INVVEC,R0        ;SET INVALID VECTOR ERROR
                        0148   31   088E  1412          BRW     DB_ERROR                ;...EXIT
                55   13 AC   9A   0891  1413 5$:       MOVZBL  ACF$B_CNUMVEC(AP),R5    ;GET NUMBER OF INTERRUPT VECTORS
                54   14 A9   DE   0895  1414          MOVAL   CRB$L_INTD(R9),R4       ;GET ADDR OF FIRST DISPATCHER
             64   0000003C'EF   D0   0899  1415 10$:      MOVL    L^MBINT_DISP,VEC$Q_DISPATCH(R4) ;SET "PUSHR #^M<R2,R3,R4,R5>"   [11]
                                  08A0  1416                                          ; AND "JSB @#"
                14 A4   6C   D0   08A0  1417          MOVL    ACF$L_ADAPTER(AP),VEC$L_ADP(R4) ;SET ADDR OF ADP
                08 A4   58   D0   08A4  1418          MOVL    R8,VEC$L_IDB(R4)        ;SET ADDR OF IDB
                0C AB   01   91   08A8  1419          CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
                              21   12   08AC  1420          BNEQ    30$                     ;BR IF NOT
                   50   6C   D0   08AE  1421          MOVL    ACF$L_ADAPTER(AP),R0    ;GET ADDR OF ADP
                50   56   10 A0   C1   08B1  1422          ADDL3   ADP$L_VECTOR(R0),R6,R0  ;GET ADDR OF VECTOR TABLE ENTRY
             60   00000000'8F   D1   08B6  1423          CMPL    #UBA$UNEXINT,(R0)       ;IS VECTOR IN USE?
                              0A   13   08BD  1424          BEQL    20$                     ;BR IF NOT
             50   007C801A 8F   D0   08BF  1425          MOVL    #SYSG$_VECINUSE,R0      ;SET ERROR STATUS
                        0110   31   08C6  1426          BRW     DB_ERROR                ;...EXIT
                                  08C9  1427 20$:
                00000000'EF   16   08C9  1428          JSB     IOGEN$CONN_VEC          ; CONNECT UB DEVICE VECTOR      ; 19
                   56   04   C0   08CF  1429 30$:       ADDL    #4,R6                   ;INCREMENT VECTOR OFFSET
                   54   24   C0   08D2  1430          ADDL    #VEC$K_LENGTH,R4        ;INCREMENT DISPATCH ADDR
                     C1 55   F5   08D5  1431          SOBGTR  R5,10$                  ;DECREMENT VECTOR COUNT - BR IF MORE
                                  08D8  1432 ;
                                  08D8  1433 ; CREATE A UCB
                                  08D8  1434 ;
                                  08D8  1435 CREATE_UCB:
                51   0E AB   3C   08D8  1436          MOVZWL  DPT$W_UCBSIZE(R11),R1   ;GET SIZE OF UCB
                      01E2   30   08DC  1437          BSBW    QIO$ALLOCATE            ;ALLOCATE THE UCB
                     03 50   E8   08DF  1438          BLBS    R0,5$                   ;BR IF SUCCESS
                      00F4   31   08E2  1439          BRW     DB_ERROR                ;ELSE - ERROR
          00000005'EF   08   88   08E5  1440 5$:       BISB    #LOAD_M_UCB,L^LOAD_FLAGS ;SET UCB LOADED FLAG                   [11]
                57   52   D0   08EC  1441          MOVL    R2,R7                   ;GET ADDR OF UCB
                08 A7   51   B0   08EF  1442          MOVW    R1,UCB$W_SIZE(R7)       ;SET SIZE
                0A A7   10   90   08F3  1443          MOVB    #DYN$C_UCB,UCB$B_TYPE(R7) ;SET TYPE
             0C A7   0C A7   DE   08F7  1444          MOVAL   UCB$L_ASTQFL(R7),UCB$L_ASTQFL(R7) ;SET AST QUEUE LISTHEAD
             10 A7   0C A7   DE   08FC  1445          MOVAL   UCB$L_ASTQFL(R7),UCB$L_ASTQBL(R7) ;...
                20 A7   59   D0   0901  1446          MOVL    R9,UCB$L_CRB(R7)        ;SET ADDR OF CRB
                     0C A9   B6   0905  1447          INCW    CRB$W_REFC(R9)          ;INCREMENT CRB REFERENCE COUNT
                50   12 AC   9A   0908  1448          MOVZBL  ACF$B_CUNIT(AP),R0      ;GET CONTROLLER UNIT NUMBER
              14 A840   57   D0   090C  1449          MOVL    R7,IDB$L_UCBLST(R8)[R0] ;SET ADDR OF UCB IN IDB
                24 A7   5A   D0   0911  1450          MOVL    R10,UCB$L_DDB(R7)       ;SET ADDR OF DDB
             40 A7   40 A7   DE   0915  1451          MOVAL   UCB$L_IOQFL(R7),UCB$L_IOQFL(R7) ;SET I/O QUEUE LISTHEAD
             44 A7   40 A7   DE   091A  1452          MOVAL   UCB$L_IOQFL(R7),UCB$L_IOQBL(R7) ;...
             48 A7   12 AC   9B   091F  1453          MOVZBW  ACF$B_CUNIT(AP),UCB$W_UNIT(R7) ;SET UNIT NUMBER
             74 A7   0A AC   90   0924  1454          MOVB    ACF$B_AUNIT(AP),UCB$B_SLAVE(R7) ;SET SLAVE CNTRLER NUMBER
                                  0929  1455
                51   04 AA   DE   0929  1456          MOVAL   DDB$L_UCB(R10),R1       ;GET ADDR OF ADDR OF FIRST UCB
                   50   61   D0   092D  1457          MOVL    (R1),R0                 ;GET ADDR OF FIRST UCB
                              09   13   0930  1458          BEQL    20$                     ;BR IF NONE
                51   2C A0   DE   0932  1459 10$:       MOVAL   UCB$L_LINK(R0),R1       ;GET ADDR OF ADDR OF NEXT UCB
                   50   61   D0   0936  1460          MOVL    (R1),R0                 ;GET ADDR OF NEXT UCB
```

```
                F7   12   0939  1461          BNEQ    10$                             ;BR IF IT EXISTS
0000000A'EF     51   D0   093B  1462 20$:     MOVL    R1,L^UCB_BLINK                  ;SAVE UCB BACKWARD LINK                 [11]
          61    57   D0   0942  1463          MOVL    R7,(R1)                         ;SET FORWARD LINK TO UCB
                           0945  1464
                           0945  1465 ;
                           0945  1466 ; INITIALIZE ALL THE CONTROL BLOCKS
                           0945  1467 ;
                           0945  1468 INIT_DB:
            54   5B   D0   0945  1469          MOVL    R11,R4                          ;SET ADDR OF DRIVER PROLOGUE
            55   57   D0   0948  1470          MOVL    R7,R5                           ;SET ADDR OF UCB
      00000000'EF   16   094B  1471          JSB     IOC$INITDRV                             ; Init the control blocks       [11]
 58 A5    0800 8F   A8   0951  1472          BISW    #UCB$M_VALID,UCB$W_STS(R5);FORCE SOFTWARE VOLUME VALID
           0A 50   E8   0957  1473          BLBS    R0,INIT_CNTRL                   ;BR IF SUCCESS
      50   007C800A 8F   D0   095A  1474          MOVL    #SYSG$_INVDPTINI,R0             ;SET ERROR STATUS
             0075   31   0961  1475          BRW     DB_ERROR                        ;...EXIT
                           0964  1476 ;
                           0964  1477 ; CALL THE CONTROLLER AND DRIVE INITIALIZATION ROUTINES
                           0964  1478 ;
                           0964  1479 INIT_CNTRL:
                           0964  1480          DSBINT                          ;DISABLE INTERRUPTS
            7E   12   DB   0964                  MFPR    S^#PR$_IPL,-(SP)
            12   1F   DA   0967                  MTPR    #31,S^#PR$_IPL
 06 00000005'EF   01   E1   096A  1481          BBC     #LOAD_V_CRB,L^LOAD_FLAGS,INIT_UNIT ;BR IF CRB NOT JUST CREATED  [11]
            56   5A   D0   0972  1482          MOVL    R10,R6                          ;SET ADDR OF DDB
              00DB   30   0975  1483          BSBW    IOGEN$CNTRL_INI                 ;INITIALIZE THE CONTROLLER
                           0978  1484
                           0978  1485 INIT_UNIT:
            55   57   D0   0978  1486          MOVL    R7,R5                           ;GET ADDR OF UCB
            54   68   D0   097B  1487          MOVL    IDB$L_CSR(R8),R4                ;GET ADDR OF CSR
            53   54   D0   097E  1488          MOVL    R4,R3                           ;ASSUME ONLY ONE CSR
      50    10 A9   D0   0981  1489          MOVL    CRB$L_LINK(R9),R0               ;GET ADDR OF SECONDARY CRB
                07   13   0985  1490          BEQL    10$                             ;BR IF NONE
      50    1C A0   D0   0987  1491          MOVL    CRB$L_INTD+VEC$L_IDB(R0),R0 ;GET ADDR OF SECONDARY IDB
            54   60   D0   098B  1492          MOVL    IDB$L_CSR(R0),R4                ;GET SECONDARY CSR ADDR
      0C AB    01   91   098E  1493 10$:     CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
                18   12   0992  1494          BNEQ    20$                             ;BR IF NOT
            56   6C   D0   0994  1495          MOVL    ACF$L_ADAPTER(AP),R6            ;GET ADDR OF ADP
            56   66   D0   0997  1496          MOVL    ADP$L_CSR(R6),R6                ;GET ADDR OF adapter CSR
            50   56   D0   099A  1497          MOVL    R6,R0
      00000000'EF   16   099D  1498          JSB     IO$TESTCSR_L                    ; CHECK FOR NONEX CSR   ; 19
            06 50   E8   09A3  1499          BLBS    R0,20$                          ; BRANCH IF CSR EXISTS
      58 A7    10   AA   09A6  1500          BICW    #UCB$M_ONLINE,UCB$W_STS(R7) ;SET DEVICE OFFLINE
                23   11   09AA  1501          BRB     30$                             ;
      50    0C AA   D0   09AC  1502 20$:     MOVL    DDB$L_DDT(R10),R0               ; GET ADDRESS OF DDT
      51    18 A0   D0   09B0  1503          MOVL    DDT$L_UNITINIT(R0),R1           ; GET OFFSET OF UNIT CODE
      03 51    10   E0   09B4  1504          BBS     #16,R1,25$                      ; BRANCH IF ABSOLUTE SUPERVISOR ADDR
            51   50   C0   09B8  1505          ADDL    R0,R1                           ; ADD IN BASE ADDRESS
00000000'8F     51   D1   09BB  1506 25$:     CMPL    R1,#IOC$RTN                     ; IF EQUAL TO IOC$RETURN THEN
                06   12   09C2  1507          BNEQ    27$                             ; UNIT INIT WASN'T SPECIFIED
      51    2C A9   D0   09C4  1508          MOVL    CRB$L_INTD+VEC$L_UNITINIT(R9),R1   ; SO GET IT FROM CRB
                05   13   09C8  1509          BEQL    30$                             ; NOT SPECIFIED IN CRB EITHER
                61   16   09CA  1510 27$:     JSB     (R1)                            ; CALL UNIT INITIALIZATION
            56   5A   D0   09CC  1511          MOVL    R10,R6                          ; SET UP ADDRESS OF DDB
                           09CF  1512 30$:
                           09CF  1513          ENBINT                          ;RE-ENABLE INTERRUPTS
            12   8E   DA   09CF                  MTPR    (SP)+,S^#PR$_IPL
                           09D2  1514 ;
```

```
                    09D2  1515 ; OPERATION COMPLETED - UNLOCK I/O DATABASE AND EXIT
                    09D2  1516 ;
                    09D2  1517 DB_EXIT:
          007A   30 09D2  1518          BSBW    IOGEN$UNLK_IODB         ;UNLOCK I/O DATABASE
                    09D5  1519                                          ; AND LOWER IPL
     50    01   D0 09D5  1520          MOVL    #1,R0                   ;SET SUCCESS
                 04 09D8  1521          RET                             ;...EXIT
```

```
                                09D9  1523          .SBTTL  DB_ERROR        ERROR LOADING DATABASE
                                09D9  1524 ;++
                                09D9  1525 ;
                                09D9  1526 ; DB_ERROR - LOCAL ROUTINE TO UNLOAD A PARTIALLY LOADED DATABASE
                                09D9  1527 ;
                                09D9  1528 ;       This routine checks the loading flags (LOAD_FLAGS) and
                                09D9  1529 ;       unload and unlinks any control blocks that were just created.
                                09D9  1530 ;
                                09D9  1531 ; INPUTS:
                                09D9  1532 ;
                                09D9  1533 ;       LOAD_FLAGS = FLAGS INDICATING CONTROL BLOCKS THAT HAVE JUST
                                09D9  1534 ;                    BEEN CREATED
                                09D9  1535 ;
                                09D9  1536 ; OUTPUTS:
                                09D9  1537 ;
                                09D9  1538 ;       R0 = ERROR STATUS THAT CAUSED UNLOADING
                                09D9  1539 ;
                                09D9  1540 ;--
                                09D9  1541 DB_ERROR:
                     50   DD    09D9  1542          PUSHL   R0                      ;SAVE ERROR STATUS
                                09DB  1543 ;
                                09DB  1544 ; UCB UNLOADING
                                09DB  1545 ;
      1C 00000005'EF    03   E1  09DB  1546          BBC     #LOAD_V_UCB,L^LOAD_FLAGS,20$ ;BR IF UCB NOT CREATED          [11]
              0000000A'FF   D4  09E3  1547          CLRL    @L^UCB_BLINK            ;CLEAR POINTER TO UCB                [11]
                     0C A9  B7  09E9  1548          DECW    CRB$W_REFC(R9)          ;DECREMENT CRB REFERENCE COUNT
                50   12 AC  9A  09EC  1549          MOVZBL  ACF$B_CUNIT(AP),R0      ;GET CONTROLLER UNIT NUMBER
                   14 A840  D4  09F0  1550          CLRL    IDB$L_UCBLST(R8)[R0]    ;CLEAR IDB POINTER TO UCB
                50   57    D0  09F4  1551          MOVL    R7,R0                   ;SET ADDR OF UCB
                     02    10  09F7  1552          BSBB    10$                     ;DEALLOCATE THE UCB
                     04    11  09F9  1553          BRB     20$                     ;
                                09FB  1554 ;
                                09FB  1555 ; LOCAL SUBROUTINE TO DEALLOCATE A CONTROL BLOCK
                                09FB  1556 ;
                                09FB  1557 ;       INPUT - R0 = ADDRESS OF BLOCK TO DEALLOCATE
                                09FB  1558 ;
                   00E6   30    09FB  1559 10$:     BSBW    QIO$DEALLOCATE          ;DEALLOCATE TO NON-PAGED POOL
                          05    09FE  1560          RSB
                                09FF  1561
                                09FF  1562 ;
                                09FF  1563 ; IDB UNLOADING
                                09FF  1564 ;
      05 00000005'EF    02   E1  09FF  1565 20$:     BBC     #LOAD_V_IDB,L^LOAD_FLAGS,30$ ;BR IF IDB NOT CREATED          [11]
                50    58   D0  0A07  1566          MOVL    R8,R0                   ;SET ADDR OF IDB
                      EF   10  0A0A  1567          BSBB    10$                     ;DEALLOCATE THE IDB
                                0A0C  1568 ;
                                0A0C  1569 ; CRB UNLOADING
                                0A0C  1570 ;
      1A 00000005'EF    01   E1  0A0C  1571 30$:     BBC     #LOAD_V_CRB,L^LOAD_FLAGS,50$ ;BR IF CRB NOT CREATED          [11]
            10 0D AB    00   E1  0A14  1572          BBC     #DPT$V_SUBCNTRL,DPT$B_FLAGS(R11),40$ ;BR IF NOT SUBCONTROLLER
                51    10 A9  D0  0A19  1573          MOVL    CRB$L_LINK(R9),R1       ;GET ADDR OF SECONDARY CRB
                51    1C A1  D0  0A1D  1574          MOVL    CRB$L_INTD+VEC$L_IDB(R1),R1 ;GET ADDR OF SECONDARY IDB
                50    0A AC  9A  0A21  1575          MOVZBL  ACF$B_AUNIT(AP),R0      ;GET ADAPTER UNIT NUMBER
                   14 A140  D4  0A25  1576          CLRL    IDB$L_UCBLST(R1)[R0]    ;CLEAR DISPATCHER POINTER
                50   59    D0  0A29  1577 40$:     MOVL    R9,R0                   ;SET ADDR OF CRB
                     CD    10  0A2C  1578          BSBB    10$                     ;DEALLOCATE THE CRB
                                0A2E  1579 ;
```

```
                                0A2E  1580 ; DDB UNLOADING
                                0A2E  1581 ;
      0E 00000005'EF    00      E1  0A2E  1582 50$:     BBC     #LOAD_V_DDB,L^LOAD_FLAGS,60$ ;BR IF DDB NOT CREATED          [11]
                     0B AB      97  0A36  1583          DECB    DPT$B_REFC(R11)              ;DECREMENT DPT REFERENCE COUNT
                00000006'FF     D4  0A39  1584          CLRL    @L^DDB_BLINK                 ;CLEAR LINK TO THE DDB               [11]
                     50   5A    D0  0A3F  1585          MOVL    R10,R0                       ;SET ADDR OF DDB
                          B7    10  0A42  1586          BSBB    10$                          ;DEALLOCATE THE DDB
                                0A44  1587 60$:
                        0008    30  0A44  1588          BSBW    IOGEN$UNLK_IODB              ;UNLOCK THE I/O DATABASE
                                0A47  1589                                                   ; AND LOWER IPL
                          50 8ED0   0A47  1590          POPL    R0                           ;RESTORE STATUS
                             04     0A4A  1591          RET                                  ;...EXIT
                                0A4B  1592
                                0A4B  1593
                                0A4B  1594
                                0A4B  1595          .SBTTL                  I/O DATABASE INTERLOCKS
                                0A4B  1596 IOGEN$LOCK_IODB:
                                0A4B  1597          SETIPL  #31
                     12   1F    DA  0A4B                      MTPR    #31,S^#PR$_IPL
                             05     0A4E  1598          RSB
                                0A4F  1599 IOGEN$UNLK_IODB:
                                0A4F  1600          SETIPL  #0
                     12   00    DA  0A4F                      MTPR    #0,S^#PR$_IPL
                             05     0A52  1601          RSB
```

```
                                0A53  1603          .SBTTL  IOGEN$CNTRL_INI Initialize a controller
                                0A53  1604 ;++
                                0A53  1605 ;
                                0A53  1606 ; IOGEN$CNTRL_INI - INITIALIZE THE DEVICE CONTROLLER
                                0A53  1607 ;
                                0A53  1608 ; INPUTS - R6 = ADDR OF DDB
                                0A53  1609 ;          R11 = ADDR OF DRIVER PROLOGUE TABLE
                                0A53  1610 ;
                                0A53  1611 ;--
                                0A53  1612 IOGEN$CNTRL_INI::
 0C AB    01    91    0A53  1613          CMPB    #AT$_UBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
          42    12    0A57  1614          BNEQ    20$                         ;BR IF NOT
     0FFC 8F    BB    0A59  1615          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE REGISTERS
 50    04 A6    D0    0A5D  1616          MOVL    DDB$L_UCB(R6),R0            ;GET ADDR OF FIRST UCB
 58    20 A0    D0    0A61  1617          MOVL    UCB$L_CRB(R0),R8            ;GET ADDR OF CRB
 55    1C A8    D0    0A65  1618          MOVL    CRB$L_INTD+VEC$L_IDB(R8),R5 ;SET ADDR OF IDB
    54    65    D0    0A69  1619          MOVL    IDB$L_CSR(R5),R4            ;SET ADDR OF CSR
 50    10 A5    D0    0A6C  1620          MOVL    IDB$L_ADP(R5),R0            ;GET ADDR OF ADP
    50    60    D0    0A70  1621          MOVL    ADP$L_CSR(R0),R0            ;GET ADDR OF UBA CSR
     0040 8F    BB    0A73  1622          PUSHR   #^M<R6>                     ;SAVE DDB ADDRESS
 56    10 A5    D0    0A77  1623          MOVL    IDB$L_ADP(R5),R6            ;GET ADDRESS OF ADP
    56    66    D0    0A7B  1624          MOVL    ADP$L_CSR(R6),R6            ;GET ADDRESS OF CSR
    50    54    D0    0A7E  1625          MOVL    R4,R0                       ;Copy address of Controller CSR
00000000'EF    16    0A81  1626          JSB     IO$TESTCSR_W                ;TEST IF CONTROLLER EXISTS      ; 19
     0040 8F    BA    0A87  1627          POPR    #^M<R6>                     ;RESTORE DDB ADDRESS
       08 50    E9    0A8B  1628          BLBC    R0,10$                      ;BRANCH IF NON-EXISTENT CSR
 50    20 A8    D0    0A8E  1629          MOVL    CRB$L_INTD+VEC$L_INITIAL(R8),R0 ;GET ADDR OF INIT ROUTINE
          02    13    0A92  1630          BEQL    10$                         ;BR IF NONE
                      0A94  1631                                  ;R4 = ADDR OF CSR
                      0A94  1632                                  ;R5 = ADDR OF IDB
                      0A94  1633                                  ;R8 = ADDR OF CRB
          60    16    0A94  1634          JSB     (R0)                        ;INIT CONTROLLER
                      0A96  1635 10$:
     0FFC 8F    BA    0A96  1636          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;RESTORE REGISTERS
                05    0A9A  1637          RSB
                      0A9B  1638 20$:
 0C AB    00    91    0A9B  1639          CMPB    #AT$_MBA,DPT$B_ADPTYPE(R11) ;UBA DEVICE?
          1F    12    0A9F  1640          BNEQ    40$                         ; Branch if not MBA
     0FFC 8F    BB    0AA1  1641          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE REGISTERS
 50    04 A6    D0    0AA5  1642          MOVL    DDB$L_UCB(R6),R0            ;GET ADDR OF FIRST UCB
 58    20 A0    D0    0AA9  1643          MOVL    UCB$L_CRB(R0),R8            ;GET ADDR OF CRB
 55    1C A8    D0    0AAD  1644          MOVL    CRB$L_INTD+VEC$L_IDB(R8),R5 ;SET ADDR OF IDB
    54    65    D0    0AB1  1645          MOVL    IDB$L_CSR(R5),R4            ;SET ADDR OF CSR
 50    20 A8    D0    0AB4  1646          MOVL    CRB$L_INTD+VEC$L_INITIAL(R8),R0 ;GET ADDR OF INIT ROUTINE
          02    13    0AB8  1647          BEQL    30$                         ;BR IF NONE
                      0ABA  1648                                  ;R4 = ADDR OF CSR
                      0ABA  1649                                  ;R5 = ADDR OF IDB
                      0ABA  1650                                  ;R8 = ADDR OF CRB
          60    16    0ABA  1651          JSB     (R0)                        ;INIT CONTROLLER
                      0ABC  1652 30$:
     0FFC 8F    BA    0ABC  1653          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;RESTORE REGISTERS
                      0AC0  1654 40$:
                05    0AC0  1655          RSB
```

```
                                0AC1  1657          .SBTTL                  MEMORY ALLOCATION SUBROUTINES
                                0AC1  1658 ;+
                                0AC1  1659 ; ALONONPAGED CALLS EXE$ALONONPAGED TO ALLOCATE NON-PAGED POOL.
                                0AC1  1660 ; ZERO'S THE BLOCK WHEN ALLOCATED
                                0AC1  1661 ; ALSO THE LENGTH OF THE ALLOCATED BLOCK IS ADDED TO QIO$L_ALLOCATED.
                                0AC1  1662 ;
                                0AC1  1663 ; INPUT:
                                0AC1  1664 ;       R1 - SIZE OF REQUESTED BLOCK IN BYTES
                                0AC1  1665 ;
                                0AC1  1666 ; OUTPUTS:
                                0AC1  1667 ;       R0 - SUCCESS/FAILURE INDICATION
                                0AC1  1668 ;       R1 - ALLOCATED SIZE OF BLOCK
                                0AC1  1669 ;       R2 - ADDRESS OF BLOCK
                                0AC1  1670 ;-
                                0AC1  1671 QIO$ALLOCATE:
                          08    BB  0AC1  1672          PUSHR   #^M<R3>                 ; SAVE R3
              00000000'EF       16  0AC3  1673          JSB     EXE$ALONONPAGED         ; ATTEMPT TO ALLOCATE
                          08    BA  0AC9  1674          POPR    #^M<R3>                 ; RESTORE R3
                       15 50    E9  0ACB  1675          BLBC    R0,20$                  ; RETURN W/ ERROR
                          06    BB  0ACE  1676          PUSHR   #^M<R1,R2>              ; SAVE REGS
           51   51       FD 8F  78  0AD0  1677          ASHL    #-3,R1,R1               ; DIVIDE BY 8
                          82    7C  0AD5  1678 10$:     CLRQ    (R2)+                   ; CLEAR
                       FB 51    F5  0AD7  1679          SOBGTR  R1,10$                  ; COUNT QUADWORD AND CONTINUE
                          06    BA  0ADA  1680          POPR    #^M<R1,R2>              ; RESTORE
        00000000'EF       51    C0  0ADC  1681          ADDL    R1,L^QIO$L_ALLOCATE     ; REMEMBER IT WAS ALLOCATE          [11]
                                05  0AE3  1682 20$:     RSB                             ; RETURN
                                0AE4  1683
                                0AE4  1684
                                0AE4  1685
                                0AE4  1686 ;+
                                0AE4  1687 ; QIO$DEALLOCATE CALLS EXE$DEANONPAGED TO RELEASE THE MEMORY
                                0AE4  1688 ; THE SIZE OF THE BLOCK IS SUBTRACTED FROM QIO$L_ALLOCATE
                                0AE4  1689 ; INPUT:        R0 Address of block
                                0AE4  1690 ;-
                                0AE4  1691 QIO$DEALLOCATE::
                          0E    BB  0AE4  1692          PUSHR   #^M<R1,R2,R3>           ; SAVE REGS
                    51    08 A0   3C  0AE6  1693          MOVZWL  8(R0),R1                ; GET LENGH OF BLOCK
        00000000'EF       51    C2  0AEA  1694          SUBL    R1,L^QIO$L_ALLOCATE     ; SUBTRACT FROM SPACE ALLOCATE      [11]
              00000000'EF   16    16  0AF1  1695          JSB     EXE$DEANONPAGED         ; RELEASE IT
                          0E    BA  0AF7  1696          POPR    #^M<R1,R2,R3>           ; RESTORE
                                05  0AF9  1697          RSB                             ; RETURN
```

```
                        0AFA  1699          .SBTTL  ADPLINK         Link adapter to end of ADP list
                        0AFA  1700 ;+
                        0AFA  1701 ; ADPLINK LINKS THE ADAPTER CONTROL BLOCK  TO THE END OF THE ADP LIST
                        0AFA  1702 ;
                        0AFA  1703 ; INPUT:
                        0AFA  1704 ;       R6 - ADDRESS OF NEW ADP
                        0AFA  1705 ; OUTPUTS:
                        0AFA  1706 ;       ADP IS LINK TO THE END OF THE ADPLIST LOCATED BY IOC$GL_ADPLIST.
                        0AFA  1707 ;-
                        0AFA  1708 ADPLINK:                                             ;
    50  FFFFFFFC'EF  9E    0AFA  1709          MOVAB   <IOC$GL_ADPLIST-ADP$L_LINK>,R0  ; START OF LIST
     51  04 A0  D0    0B01  1710 10$:    MOVL    ADP$L_LINK(R0),R1       ; FLINK TO FIRST ENTRY
            05  13    0B05  1711          BEQL    20$                     ; AT END
        50  51  D0    0B07  1712          MOVL    R1,R0                   ; TRY AGAIN
            F5  11    0B0A  1713          BRB     10$                     ;
     04 A0  56  D0    0B0C  1714 20$:    MOVL    R6,ADP$L_LINK(R0)       ; CHAIN NEW ADP TO END OF LIST
            05  0B10  1715          RSB
                0B11  1716
                0B11  1717
                0B11  1718 ;+
                0B11  1719 ; FIND_ADP LOCATES THE ADAPTER CONTROL BLOCK WITH THE MATCHING TR NUMBER
                0B11  1720 ; INPUT:        R4=CSR                                                  [18]
                0B11  1721 ; OUTPUT:       R6=ADP
                0B11  1722 ;-
                0B11  1723 FIND_ADP:
            50  D4    0B11  1724          CLRL    R0                      ; INITIALLY A FAILURE
    56  FFFFFFFC'EF  9E    0B13  1725          MOVAB   L^IOC$GL_ADPLIST-ADP$L_LINK,R6  ; HEAD OF ADAPTER LIST      [11]
     56  04 A6  D0    0B1A  1726 10$:    MOVL    ADP$L_LINK(R6),R6       ; ADVANCE TO NEXT IN LIST
            08  13    0B1E  1727          BEQL    20$                     ; BRANCH IF END OF LIST
        54  66  D1    0B20  1728          CMPL    ADP$L_CSR(R6),R4        ; CSR NUMBER MATCH?             [18]
            F5  12    0B23  1729          BNEQ    10$                     ; BRANCH IF NO MATCH
        50  00'  D0    0B25  1730          MOVL    S^#SS$_NORMAL,R0        ; SUCCESS
            05  0B28  1731 20$:    RSB
```

```
                     0B29  1733          .Sbttl  Exe$PwrTimChk - Check reasonable interval since power recovery ;[14]
                     0B29  1734 ;++                                                                               [14]
                     0B29  1735 ; Functional Description:                                                         [14]
                     0B29  1736 ;       EXE$PWRTIMCHK is called by driver initialization code to check for        [14]
                     0B29  1737 ;       a sufficient interval since power recovery to expect devices to be        [14]
                     0B29  1738 ;       ready again.  If the return from EXE$PWRTIMCHK indicates that the         [14]
                     0B29  1739 ;       reasonable interval has not yet elapsed, the device driver may elect      [14]
                     0B29  1740 ;       to wait for a while using EXE$PWRTIMCHK check the time.                   [14]
                     0B29  1741 ;                                                                                [14]
                     0B29  1742 ; Calling Sequence:                                                               [14]
                     0B29  1743 ;       BSB/JSB EXE$PWRTIMCHK                                                     [14]
                     0B29  1744 ;                                                                                [14]
                     0B29  1745 ; Output Parameters:                                                              [14]
                     0B29  1746 ;       R0 - Low bit clear if interval expired.                                   [14]
                     0B29  1747 ;--                                                                               [14]
```

```
                                0B29  1749 Exe$PwrTimChk::                                  ;                                                  [14]
                      50   D4   0B29  1750          CLRL    R0                              ; Assume interval expired                          [14]
      06   0000FE17'EF   91   0B2B  1751          CMPB    DSA$GL_SID+3,#PR$_SID_TYP8NN ; is this the nautilus cpu?                  [24]
                      09   13   0B32  1752          BEQL    3999$                           ; yes                                              [28]
      11   0000FE17'EF   91   0B34  1753          CMPB    DSA$GL_SID+3,#pr$_sid_typ8rr ; is this the RRR cpu?          [28] ;G:8
                      52   12   0B3B  1754          BNEQ    4020$                           ; if not then continue                             [24]
                                0B3D  1755 ;+
                                0B3D  1756 ;  TODR is console resident for nautilus and RRR [28]                                    ;G:7
                                0B3D  1757 ;-
                                0B3D  1758 3999$:                                           ;                                                  [28]
                      07   BB   0B3D  1759          PUSHR   #^M<R0,R1,R2>                   ; register save mask                               [24]
                      52   D4   0B3F  1760          CLRL    R2                              ; init R2                                          [24]
            00000000'EF   D4   0B41  1761          CLRL    TODR_BYTE_CNT                   ; init byte/loop count                             [24]
                  00'8F   88   0B47  1762          BISB2   #DS$M_TODR_ACTIVE,-             ; set TODR read processing flag                    [24]
            00000000'EF        0B4A  1763                  CONSOLEAAA_FLAGS
                                0B4F  1764 4000$:
                 50   22   DB   0B4F  1765          MFPR    #PR$_TXCS,R0                    ; store tx status                                  [24]
   F5 50   00000000'8F   E1   0B52  1766          BBC     #TXCS$V_RDY,R0,4000$            ; branch if not ready to transmit                  [24]
      23   00000F08 8F   DA   0B5A  1767          MTPR    #^X00000F08,#PR$_TXDB           ; TOY read request                                 [24]
                                0B61  1768 4005$:
                                0B61  1769          $DS_BREAK                               ; get binary data                                  [24]
      00000000'9F   6E   FA   0B61                          CALLG   (SP), @#DS$BREAK
      04   00000000'EF   D1   0B68  1770          CMPL    TODR_BYTE_CNT,#4                ; do we have all four bytes of toy data            [24]
                      08   13   0B6F  1771          BEQL    4010$                           ; yes then continue                                [24]
   E8 52   00002710 8F   F2   0B71  1772          AOBLSS  #10000,R2,4005$                 ; else keep looping                                [24]
                                0B79  1773 4010$:
            00000000'8F   CA   0B79  1774          BICL2   #DS$M_TODR_ACTIVE,-             ; clear todr read processing flag                  [24]
            00000000'EF        0B7F  1775                  CONSOLEAAA_FLAGS
                      07   BA   0B84  1776          POPR    #^M<R0,R1,R2>                   ; register restore                                 [24]
      7E   00000000'EF   D0   0B86  1777          MOVL    TODR_STORE,-(SP)                ; get toy data                                     [24]
                      03   11   0B8D  1778          BRB     4030$
                                0B8F  1779 4020$:
                 7E   1B   DB   0B8F  1780          MFPR    #PR$_TODR,-(SP)                 ; Get current time of day                          [14]
                                0B92  1781 4030$:
      8E   00000004'EF   D1   0B92  1782          CMPL    L^EXE$GL_PWRDONE,(SP)+          ; Check for time expired                           [14]
                      02   1B   0B99  1783          BLEQU   10$                             ; Exit with low bit clear if expired               [14]
                      50   D6   0B9B  1784          INCL    R0                              ; Else set low bit of R0                           [14]
                           05   0B9D  1785 10$:     RSB                                     ;                                                  [14]
                                0B9E  1786          .END
```

```
$$ARGS                          = 00000007     D
$$T1                            = 00000020     D
$ER                             = 00000006     D
$MODULE                           00000000 R   D   02
$_KNOWN                         = 00000007     D
ACF                               FFFFFFC0     D
ACF$B_AFLAG                       0000000B     D
ACF$B_AUNIT                       0000000A     D
ACF$B_CNUMVEC                     00000013     D
ACF$B_CUNIT                       00000012     D
ACF$C_LENGTH                      00000020     D
ACF$K_LENGTH                      00000020     D
ACF$L_ADAPTER                     00000000     D
ACF$L_CONFIGREG                   00000004     D
ACF$L_CONTRLREG                   0000000C     D
ACF$L_DEVNAME                     00000014     D
ACF$L_DRVNAME                     00000018     D
ACF$W_AVECTOR                     00000008     D
ACF$W_CVECTOR                     00000010     D
ACF$W_MAXUNITS                    0000001C     D
ADP$B_NUMBER                      0000000B     D
ADP$B_PORT                        00000020     D
ADP$B_TYPE                        0000000A     D
ADP$C_DRADPLEN                    00000014     D
ADP$C_MBAADPLEN                   00000014     D
ADP$C_MPMADPLEN                   00000070     D
ADP$C_UBAADPLEN                   00000070     D
ADP$K_DRADPLEN                    00000014     D
ADP$K_MBAADPLEN                   00000014     D
ADP$K_MPMADPLEN                   00000070     D
ADP$K_UBAADPLEN                   00000070     D
ADP$L_CRB                         00000010     D
ADP$L_CSR                         00000000     D
ADP$L_DPQBL                       00000018     D
ADP$L_DPQFL                       00000014     D
ADP$L_INTD                        00000064     D
ADP$L_LINK                        00000004     D
ADP$L_MRQBL                       00000020     D
ADP$L_MRQFL                       0000001C     D
ADP$L_PRQQBL                      00000018     D
ADP$L_PRQQFL                      00000014     D
ADP$L_SHB                         0000001C     D
ADP$L_VECTOR                      00000010     D
ADP$W_ADPTYPE                     0000000E     D
ADP$W_DPBITMAP                    00000024     D
ADP$W_MRBITMAP                    00000026     D
ADP$W_SIZE                        00000008     D
ADP$W_TR                          0000000C     D
ADPLINK                           00000AFA R   D   04
ALLOC$_ACMODE                   = 00000010     D
ALLOC$_DEVNAM                   = 00000004     D
ALLOC$_NARGS                    = 00000004     D
ALLOC$_PHYBUF                   = 0000000C     D
ALLOC$_PHYLEN                   = 00000008     D
AT$_DR                          = 00000002     D
AT$_MBA                         = 00000000     D
AT$_UBA                         = 00000001     D
AUNKFUNC                          00000008 R   D   02
BIT...                          = 00000004     D
BUG$CHECK                         00000716 RG  D   04
CCB$B_AMOD                        00000009     D
CCB$B_STS                         00000008     D
CCB$C_LENGTH                      00000010     D
CCB$K_LENGTH                      00000010     D
CCB$L_DIRP                        0000000C     D
CCB$L_UCB                         00000000     D
CCB$L_WIND                        00000004     D
CCB$W_IOC                         0000000A     D
CMK$_APBOOT                     = 00000004     D
CMK$_ASTEXIT                    = 00000000     D
CMK$_CANTIM                     = 0000000B     D
CMK$_HIBER                      = 00000007     D
CMK$_INITSCB                    = 00000008     D
CMK$_LAST                       = 0000000E     D
CMK$_MMENABLE                   = 00000003     D
CMK$_RCONSOLE                   = 00000001     D
CMK$_REFRESH                    = 00000005     D
CMK$_SCHDWK                     = 00000006     D
CMK$_SETIMR                     = 0000000C     D
CMK$_SETIPL                     = 00000009     D
CMK$_SETPRT                     = 0000000D     D
CMK$_SGIPR                      = 0000000A     D
CMK$_TCONSOLE                   = 00000002     D
CONSOLEAAA_FLAGS                  ********W G    00
CR                              = 0000000D     D
CRB$B_MASK                        0000000E     D
CRB$B_TYPE                        0000000A     D
CRB$C_LENGTH                      00000038     D
CRB$K_LENGTH                      00000038     D
CRB$L_INTD                        00000014     D
CRB$L_INTD2                       00000038     D
CRB$L_LINK                        00000010     D
CRB$L_WQBL                        00000004     D
CRB$L_WQFL                        00000000     D
CRB$W_REFC                        0000000C     D
CRB$W_SIZE                        00000008     D
CREATE_CRB                        0000080B R   D   04
CREATE_DDB                        000007AF R   D   04
CREATE_IDB                        00000857 R   D   04
CREATE_UCB                        000008D8 R   D   04
CREATE_VEC                        00000881 R   D   04
CRLF_LENGTH                     = 00000002     D
CTL$GW_CHINDX                     0000003A RG  D   02
CTL$GW_NMIOCH                     00000038 RG  D   02
DB_ERROR                          000009D9 R   D   04
DB_EXIT                           000009D2 R   D   04
DDB$B_ACPCLASS                    00000013     D
DDB$B_TYPE                        0000000A     D
DDB$C_LENGTH                      00000034     D
DDB$K_LENGTH                      00000034     D
DDB$L_ACPD                        00000010     D
DDB$L_DDT                         0000000C     D
DDB$L_LINK                        00000000     D
DDB$L_UCB                         00000004     D
```

```
DDB$T_DRVNAME                   00000024    D
DDB$T_NAME                      00000014    D
DDB$W_SIZE                      00000008    D
DDB_BLINK                       00000006 R  D  03
DDT$L_ALTSTART                  0000001C    D
DDT$L_CANCEL                    0000000C    D
DDT$L_FDT                       00000008    D
DDT$L_REGDUMP                   00000010    D
DDT$L_START                     00000000    D
DDT$L_UNITINIT                  00000018    D
DDT$L_UNSOLINT                  00000004    D
DDT$W_DIAGBUF                   00000014    D
DDT$W_ERRORBUF                  00000016    D
DIR...                        = FFFFFFFF    D
DPT$B_ADPTYPE                   0000000C    D
DPT$B_FLAGS                     0000000D    D
DPT$B_REFC                      0000000B    D
DPT$B_TYPE                      0000000A    D
DPT$C_LENGTH                    0000002A    D
DPT$K_LENGTH                    0000002A    D
DPT$L_BLINK                     00000004    D
DPT$L_FLINK                     00000000    D
DPT$T_NAME                      0000001E    D
DPT$V_SUBCNTRL                = 00000000    D
DPT$W_INITTAB                   00000010    D
DPT$W_MAXUNITS                  00000016    D
DPT$W_REINITTAB                 00000012    D
DPT$W_SIZE                      00000008    D
DPT$W_UCBSIZE                   0000000E    D
DPT$W_UNLOAD                    00000014    D
DPT$W_VERSION                   00000018    D
DR$INITIAL                      ********    X  00
DR$INT                          ********    X  00
DR780$B_BR                      00000033      D
DR780$B_CONFIG                  00000035      D
DR780$B_SELF                    00000034      D
DR780$B_TR                      00000032      D
DR780$B_UUT                     00000036      D
DR780$K_LEN                     00000037      D
DRACRB                          000006F7 R  D  04
DRACRBLEN                     = 00000010      D
DS$BREAK                        ********    X  00
DS$GK_SID                       ********    X  00
DS$GL_FLAGS                     ********    X  00
DS$GPHARD                       ********    X  00
DS$K_CCB                        ********    X  00
DS$K_ERROR                    = 00000002      D
DS$K_NORMAL                   = 00000001      D
DS$K_PRINTB                   = 00000002      D
DS$K_PRINTF                   = 00000001      D
DS$K_PRINTI                   = 00000000      D
DS$K_PRINTX                   = 00000003      D
DS$K_SEVERE                   = 00000004      D
DS$K_SUBSYS                   = 00000066      D
DS$K_TYPE_ABORT_PROGRAM       = 00000014      D
DS$K_TYPE_ABORT_TEST          = 00000013      D
DS$K_TYPE_COMMAND_ERR         = 00000015      D
DS$K_TYPE_COMMAND_OUT         = 00000016      D
DS$K_TYPE_CRD_AUTOTEST        = 0000001A      D
DS$K_TYPE_DS_PROMPT           = 00000001      D
DS$K_TYPE_DS_START            = 0000001D      D
DS$K_TYPE_ERRDEV              = 00000008      D
DS$K_TYPE_ERRHARD             = 00000006      D
DS$K_TYPE_ERROR_BODY          = 00000009      D
DS$K_TYPE_ERROR_END           = 0000000A      D
DS$K_TYPE_ERRPREP             = 0000001B      D
DS$K_TYPE_ERRSOFT             = 00000007      D
DS$K_TYPE_ERRSUP              = 00000004      D
DS$K_TYPE_ERRSYS              = 00000005      D
DS$K_TYPE_ERR_HALT            = 0000000D      D
DS$K_TYPE_EXCEPTION           = 0000000C      D
DS$K_TYPE_EXCEPTION_HEAD      = 0000000B      D
DS$K_TYPE_FIRST_PASS          = 00000011      D
DS$K_TYPE_GENERAL             = 00000000      D
DS$K_TYPE_GENERAL_ERROR       = 00000003      D
DS$K_TYPE_NO_TESTS            = 00000012      D
DS$K_TYPE_PARAM_ERROR         = 0000001C      D
DS$K_TYPE_PROGRAM_END         = 00000010      D
DS$K_TYPE_PROGRAM_INFO        = 00000017      D
DS$K_TYPE_PROGRAM_START       = 0000000F      D
DS$K_TYPE_QIO_INVADP          = 00000024      D
DS$K_TYPE_QIO_NODRIVER        = 00000022      D
DS$K_TYPE_QIO_WRONGVER        = 00000023      D
DS$K_TYPE_SCRIPT_ECHO         = 00000021      D
DS$K_TYPE_SCRIPT_PNF          = 0000001E      D
DS$K_TYPE_SCRIPT_PROMPT       = 00000020      D
DS$K_TYPE_SCRIPT_SKIP         = 0000001F      D
DS$K_TYPE_SEQUENCE_ERROR      = 00000019      D
DS$K_TYPE_START_ERR           = 00000018      D
DS$K_TYPE_START_LIST          = 00000025      D
DS$K_TYPE_SUMMARY             = 0000000E      D
DS$K_TYPE_USER_PROMPT         = 00000002      D
DS$K_WARNING                  = 00000000      D
DS$LOAD                         ********    X  00
DS$M_ABRTFLG                  = 00000040      D
DS$M_BADTIME                  = 00100000      D
DS$M_BATCH                    = 00400000      D
DS$M_BRKCLR                   = 00001000      D
DS$M_BRKPT                    = 00000800      D
DS$M_CHARFLG                  = 00000100      D
DS$M_CMDFLG                   = 00000080      D
DS$M_CTRLC                    = 00000001      D
DS$M_CTRLO                    = 00010000      D
DS$M_DEVFLG                   = 00000200      D
DS$M_DISABLCC                 = 01000000      D
DS$M_DONFLG                   = 00002000      D
DS$M_ERRFLG                   = 00000010      D
DS$M_EXCEPT                   = 00080000      D
DS$M_EXETST                   = 00040000      D
DS$M_HLTFLG                   = 00000008      D
DS$M_LODFLG                   = 00000002      D
DS$M_MEMMGT                   = 00008000      D
DS$M_NI                       = 04000000      D
DS$M_OUTPUT                   = 00800000      D
```

```
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$M_TODR_ACTIVE                  ********W G    00
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DS$_AP_NORMAL_BREAK             = 00660150    D
DS$_ARITH                       = 006600D0    D
DS$_ASBE                        = 00660118    D
DS$_BADLINK                     = 006600F0    D
DS$_BADTYPE                     = 006600E8    D
DS$_BIIC                        = 00660120    D
DS$_CHME                        = 006600A8    D
DS$_CHMK                        = 006600E0    D
DS$_DEVNAME                     = 00660108    D
DS$_ERROR                       = 00660002    D
DS$_FHWE                        = 00660068    D
DS$_FRAGBUF                     = 00660080    D
DS$_ICBUSY                      = 006600C8    D
DS$_ICERR                       = 006600C0    D
DS$_IHWE                        = 00660060    D
DS$_ILLCHAR                     = 00660018    D
DS$_ILLPAGCNT                   = 00660078    D
DS$_ILLUNIT                     = 00660100    D
DS$_INIT_FAIL                   = 00660140    D
DS$_INSFMEM                     = 00660050    D
DS$_INVCPU                      = 00660130    D
DS$_IPL2HI                      = 006600B8    D
DS$_IVADDR                      = 00660040    D
DS$_IVVECT                      = 00660038    D
DS$_KRNLSTK                     = 00660090    D
DS$_LOGIC                       = 00660070    D
DS$_MCHK                        = 00660088    D
DS$_MEM_ALLOC_ERR               = 00660138    D
DS$_MMOFF                       = 00660058    D
DS$_NEEDUNIT                    = 006600F8    D
DS$_NODE                        = 00660128    D
DS$_NOPCS                       = 00660110    D
DS$_NORMAL                      = 00660001    D
DS$_NOSUPPORT                   = 006600B1    D
DS$_NOTALLOWMP                  = 00660148    D
DS$_NOTDON                      = 00660030    D
DS$_NOTIMP                      = 006600B0    D
DS$_NULLSTR                     = 00660010    D
DS$_OVERFLOW                    = 00660008    D
DS$_POWER                       = 00660098    D
DS$_PROGERR                     = 00660020    D
DS$_SEVERE                      = 00660004    D
DS$_TRANSL                      = 006600A0    D
DS$_TRUNCATE                    = 00660028    D
DS$_UNEXPINT                    = 006600D8    D
DS$_UNSUPPDEV                   = 00660158    D
DS$_VASFULL                     = 00660048    D
DS$_WARNING                     = 00660000    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSA$M_HALT                      = 00000001    D
DSR$COMPLETION                    ********    X    00
DSX$PRINT                         ********    X    00
DS_ERRSUP                         ********    X    00
DVR$B_CNUMVEC                     0000000B     D
DVR$B_TYPE                        0000000A     D
DVR$C_LEN                         00000014     D
DVR$L_IDENT                       00000010     D
DVR$W_DPT                         0000000E     D
DVR$W_GENERIC                     0000000C     D
```

```
DVR$W_SIZE                      00000008     D
DVR$_IDENT                    = FFFF0001     D
DW750$K_LEN                     00000032     D
DW780$B_BR                      00000033     D
DW780$B_TR                      00000032     D
DW780$K_LEN                     00000034     D
DYN$C_ACB                     = 00000002   G D
DYN$C_ADP                     = 00000001   G D
DYN$C_AQB                     = 00000003   G D
DYN$C_BRDCST                  = 0000001A   G D
DYN$C_BUFIO                   = 00000013   G D
DYN$C_CEB                     = 00000004   G D
DYN$C_CRB                     = 00000005   G D
DYN$C_CXB                     = 0000001B   G D
DYN$C_DDB                     = 00000006   G D
DYN$C_DPT                     = 0000001E   G D
DYN$C_EXTGSD                  = 00000028   G D
DYN$C_FCB                     = 00000007   G D
DYN$C_FRK                     = 00000008   G D
DYN$C_GSD                     = 00000015   G D
DYN$C_IDB                     = 00000009   G D
DYN$C_IRP                     = 0000000A   G D
DYN$C_IRPE                    = 0000002C   G D
DYN$C_JIB                     = 0000002F   G D
DYN$C_JPB                     = 0000001F   G D
DYN$C_KFH                     = 00000026   G D
DYN$C_KFI                     = 00000018   G D
DYN$C_LOG                     = 0000000B   G D
DYN$C_MBX                     = 0000002B   G D
DYN$C_MTL                     = 00000019   G D
DYN$C_MVL                     = 00000016   G D
DYN$C_NDB                     = 0000001C   G D
DYN$C_NET                     = 00000017   G D
DYN$C_PBH                     = 00000020   G D
DYN$C_PCB                     = 0000000C   G D
DYN$C_PDB                     = 00000021   G D
DYN$C_PFL                     = 00000023   G D
DYN$C_PIB                     = 00000022   G D
DYN$C_PQB                     = 0000000D   G D
DYN$C_PTR                     = 00000025   G D
DYN$C_RBM                     = 00000031   G D
DYN$C_RVT                     = 0000000E   G D
DYN$C_RVX                     = 00000027   G D
DYN$C_SFT                     = 00000024   G D
DYN$C_SHB                     = 0000002A   G D
DYN$C_SHMCEB                  = 0000002E   G D
DYN$C_SHMGSD                  = 00000029   G D
DYN$C_SHRBUFIO                = 00000080   G D
DYN$C_SLAVCEB                 = 0000002D   G D
DYN$C_SPECIAL                 = 00000080   G D
DYN$C_SSB                     = 0000001D   G D
DYN$C_TQE                     = 0000000F   G D
DYN$C_TWP                     = 00000030   G D
DYN$C_TYPAHD                  = 00000014   G D
DYN$C_UCB                     = 00000010   G D
DYN$C_VCA                     = 00000032   G D
DYN$C_VCB                     = 00000011   G D
DYN$C_WCB                     = 00000012   G D
EXE$ALONONPAGED                 ********     X   00
EXE$DEANONPAGED                 ********     X   00
EXE$FORKDSPTH                   ********     X   00
EXE$GL_PWRDONE                  00000004 RG  D   02
EXE$MAXACMODE                   00000707 RG  D   04
EXE$PWRTIMCHK                   00000B29 RG  D   04
EXE$QIO                         00000000 RG  D   04
FILE                            FFFFFFF4     D
FILEDEF                         FFFFFFE4     D
FILEDESC                        FFFFFFEC     D
FILL_ACF                        000003A6 R   D   04
FIND_ADP                        00000B11 R   D   04
HP$A_DEPENDENT                  00000032     D
HP$A_DEVICE                     00000018     D
HP$A_DVA                        0000001C     D
HP$A_LINK                       00000020     D
HP$B_DRIVE                      0000000B     D
HP$B_FLAGS                      0000000A     D
HP$B_RH780_BR                   00000033     D
HP$B_RH780_TR                   00000032     D
HP$K_RH780_LEN                  00000034     D
HP$Q_DEVICE                     00000000     D
HP$T_DEVICE                     0000000C     D
HP$T_TYPE                       00000026     D
HP$W_SIZE                       00000008     D
HP$W_VECTOR                     00000024     D
IDB$B_TYPE                      0000000A     D
IDB$C_LENGTH                    00000034     D
IDB$K_LENGTH                    00000034     D
IDB$L_ADP                       00000010     D
IDB$L_CSR                       00000000     D
IDB$L_OWNER                     00000004     D
IDB$L_UCBLST                    00000014     D
IDB$W_SIZE                      00000008     D
IDB$W_UNITS                     0000000C     D
INIT_CNTRL                      00000964 R   D   04
INIT_DB                         00000945 R   D   04
INIT_UNIT                       00000978 R   D   04
INI_DRADP                       00000616 R   D   04
INI_MBADP                       000004A0 R   D   04
INI_UBADP                       00000594 R   D   04
IO$S_FCODE                    = 00000006     D
IO$TESTCSR_L                    ********     X   00
IO$TESTCSR_W                    ********     X   00
IO$V_FCODE                    = 00000000     D
IO$_READPROMPT                = 00000037     D
IO$_WRITEVBLK                 = 00000030     D
IOC$GL_ADPLIST                  ********     X   00
IOC$GL_DEVLIST                  ********     X   00
IOC$GL_DPTLIST                  ********     X   00
IOC$INITDRV                     ********     X   00
IOC$IOPOST                      ********     X   00
IOC$REINITDRV                   ********     X   00
IOC$RTN                         ********     X   00
IOGEN$CNTRL_INI                 00000A53 RG  D   04
IOGEN$CONN_VEC                  ********     X   00
```

```
IOGEN$LOCK_IODB                 00000A4B R   D   04
IOGEN$UNLK_IODB                 00000A4F R   D   04
LASTCHAR                        00000004 R   D   03
LENGTH                          FFFFFFE0     D
LF                            = 0000000A     D
LOAD$_ADDRESS                 = 00000010     D
LOAD$_DEFAULT                 = 00000008     D
LOAD$_FILE                    = 00000004     D
LOAD$_LENGTH                  = 0000000C     D
LOAD$_NARGS                   = 00000007     D
LOAD$_RETLEN                  = 00000014     D
LOAD$_RETREC                  = 00000018     D
LOAD$_VBN                     = 0000001C     D
LOAD_DRIVER                     000002D8 R   D   04
LOAD_FLAGS                      00000005 R   D   03
LOAD_M_CRB                    = 00000002     D
LOAD_M_DDB                    = 00000001     D
LOAD_M_IDB                    = 00000004     D
LOAD_M_UCB                    = 00000008     D
LOAD_V_CRB                    = 00000001     D
LOAD_V_DDB                    = 00000000     D
LOAD_V_IDB                    = 00000002     D
LOAD_V_UCB                    = 00000003     D
LOC$DPT                         00000161 R   D   04
LOC$PTDESC                      ********     X   00
LOCAL                           FFFFFFC0     D
MBA$INITIAL                     ********     X   00
MBA$INT                         ********     X   00
MBACRB                          00000584 R   D   04
MBACRBLEN                     = 00000010     D
MBINT_DISP                      0000003C R   D   02
NUMUBAVEC                     = 00000080     D
PR$_IPL                       = 00000012     D
PR$_SID_TYP8NN                = 00000006     D
PR$_SID_TYP8RR                = 00000011     D
PR$_TODR                      = 0000001B     D
PR$_TXCS                      = 00000022     D
PR$_TXDB                      = 00000023     D
PREADVBLK                       ********     X   00
PSL$S_PRVMOD                  = 00000002     D
PSL$V_PRVMOD                  = 00000016     D
QIO$ADDUNIT                     00000259 RG  D   04
QIO$ALLOCATE                    00000AC1 R   D   04
QIO$CLEANUP                     00000192 RG  D   04
QIO$DEALLOCATE                  00000AE4 RG  D   04
QIO$INITIALIZE                  0000013E RG  D   04
QIO$LOAD_DB                     00000733 R   D   04
QIO$L_ALLOCATE                  00000000 R   D   03
QIO$_ASTADR                   = 00000014     D
QIO$_ASTPRM                   = 00000018     D
QIO$_CHAN                     = 00000008     D
QIO$_EFN                      = 00000004     D
QIO$_FUNC                     = 0000000C     D
QIO$_IOSB                     = 00000010     D
QIO$_NARGS                    = 0000000C     D
QIO$_P1                       = 0000001C     D
QIO$_P2                       = 00000020     D
QIO$_P3                       = 00000024     D
QIO$_P4                       = 00000028     D
QIO$_P5                       = 0000002C     D
QIO$_P6                       = 00000030     D
QIOCLEAN_CPU                    ********     X   00
READVBLK                        ********     X   00
REMOTE_NODE_CONTROL_LEVEL       ********W GX   00
RH780$B_BR                      00000033     D
RH780$B_TR                      00000032     D
RH780$K_LEN                     00000034     D
SAVABS...                     = FFFFFFC0     D
SCAN$ALPHA                      ********     X   00
SCAN$SEARCHLIST                 ********     X   00
SCB$L_ACCESS                    00000020     D
SCB$L_ARITH                     00000034     D
SCB$L_BREAK                     0000002C     D
SCB$L_CHME                      00000044     D
SCB$L_CHMK                      00000040     D
SCB$L_CHMS                      00000048     D
SCB$L_CHMU                      0000004C     D
SCB$L_COMPAT                    00000030     D
SCB$L_KNLSTK                    00000008     D
SCB$L_MACHCHK                   00000004     D
SCB$L_OPCCUS                    00000014     D
SCB$L_OPCDEC                    00000010     D
SCB$L_POWER                     0000000C     D
SCB$L_RADRMOD                   0000001C     D
SCB$L_ROPRAND                   00000018     D
SCB$L_RXDB                      000000F8     D
SCB$L_SFTLVL1                   00000084     D
SCB$L_SFTLVL10                  000000A8     D
SCB$L_SFTLVL11                  000000AC     D
SCB$L_SFTLVL12                  000000B0     D
SCB$L_SFTLVL13                  000000B4     D
SCB$L_SFTLVL14                  000000B8     D
SCB$L_SFTLVL15                  000000BC     D
SCB$L_SFTLVL2                   00000088     D
SCB$L_SFTLVL3                   0000008C     D
SCB$L_SFTLVL4                   00000090     D
SCB$L_SFTLVL5                   00000094     D
SCB$L_SFTLVL6                   00000098     D
SCB$L_SFTLVL7                   0000009C     D
SCB$L_SFTLVL8                   000000A0     D
SCB$L_SFTLVL9                   000000A4     D
SCB$L_TBIT                      00000028     D
SCB$L_TIMER                     000000C0     D
SCB$L_TRANSL                    00000024     D
SCB$L_TXDB                      000000FC     D
SCB$L_VM                        00000038     D
SCB$L_ZERO                      00000000     D
SCB_BASE                        ********     X   00
SCB_IMAGE                       ********     X   00
SEND_CONSOLE_RESPONSE           ********W GX   00
SIZ...                        = 00000001     D
SS$_INSFMEM                     ********     X   00
SS$_IVCHAN                      ********     X   00
SS$_NORMAL                      ********     X   00
```

```
SYSG$_INVDPTINI                 = 007C800A     D
SYSG$_INVVEC                    = 007C802A     D
SYSG$_VECINUSE                  = 007C801A     D
TODR_BYTE_CNT                     ********W G    00
TODR_STORE                        ********W G    00
TXCS$V_RDY                        ********W G    00
T_ADAPTER_TYPE                    000000C2 R   D  02
T_CRLF                            000000C0 R   D  02
T_INVADP                          00000095 R   D  02
T_KNOWN_DEV                       000000C9 R   D  02
T_NODRIVER                        00000046 R   D  02
T_WRONGVER                        00000068 R   D  02
UBA$INITIAL                       ********W G    00
UBA$UNEXINT                       ********     X 00
UBADP_CPU                         ********     X 00
UBINTSZ                           ********W G    00
UBINT_DISP                        00000044 RG  D  02
UCB$B_AMOD                        00000053     D
UCB$B_CEX                         00000077     D
UCB$B_CM1                         0000004A     D
UCB$B_CM2                         0000004B     D
UCB$B_DEVCLASS                    00000038     D
UCB$B_DEVTYPE                     00000039     D
UCB$B_DIPL                        00000052     D
UCB$B_DX_SCTCNT                   000000A6     D
UCB$B_ERTCNT                      00000070     D
UCB$B_ERTMAX                      00000071     D
UCB$B_FEX                         00000076     D
UCB$B_FIPL                        0000000B     D
UCB$B_LOCSRV                      0000003C     D
UCB$B_OFFNDX                      00000094     D
UCB$B_OFFRTC                      00000095     D
UCB$B_REMSRV                      0000003D     D
UCB$B_SECTORS                     0000003C     D
UCB$B_SLAVE                       00000074     D
UCB$B_SPR                         00000075     D
UCB$B_STATE                       00000052     D
UCB$B_TRACKS                      0000003D     D
UCB$B_TT_CRFILL                   0000009D     D
UCB$B_TT_DECRF                    000000A1     D
UCB$B_TT_DELFF                    000000A2     D
UCB$B_TT_DESPEE                   000000A0     D
UCB$B_TT_DETYPE                   000000A4     D
UCB$B_TT_LFFILL                   0000009E     D
UCB$B_TT_SPEED                    0000009C     D
UCB$B_TYPE                        0000000A     D
UCB$B_VERTSZ                      0000003F     D
UCB$C_LENGTH                      00000074     D
UCB$C_MB_LENGTH                   00000090     D
UCB$C_TT_LENGTH                   000000BC     D
UCB$K_LENGTH                      00000074     D
UCB$K_MB_LENGTH                   00000090     D
UCB$K_TT_LENGTH                   000000BC     D
UCB$L_AMB                         00000054     D
UCB$L_ASTQBL                      00000010     D
UCB$L_ASTQFL                      0000000C     D
UCB$L_CPID                        0000005C     D
UCB$L_CRB                         00000020     D
UCB$L_DDB                         00000024     D
UCB$L_DEVCHAR                     00000034     D
UCB$L_DEVDEPEND                   0000003C     D
UCB$L_DPC                         00000080     D
UCB$L_DUETIM                      0000005C     D
UCB$L_DX_BFPNT                    0000009C     D
UCB$L_DX_BUF                      00000098     D
UCB$L_DX_RXDB                     000000A0     D
UCB$L_EMB                         00000078     D
UCB$L_FIRST                       00000014     D
UCB$L_FPC                         0000000C     D
UCB$L_FQBL                        00000004     D
UCB$L_FQFL                        00000000     D
UCB$L_FR3                         00000010     D
UCB$L_FR4                         00000014     D
UCB$L_IOQBL                       00000044     D
UCB$L_IOQFL                       00000040     D
UCB$L_IRP                         0000004C     D
UCB$L_LINK                        0000002C     D
UCB$L_LOGADR                      00000064     D
UCB$L_MAXBLOCK                    00000084     D
UCB$L_MB_MBX                      0000007C     D
UCB$L_MB_PORT                     0000008C     D
UCB$L_MB_RAST                     00000078     D
UCB$L_MB_SHB                      00000080     D
UCB$L_MB_WAST                     00000074     D
UCB$L_MB_WIOQBL                   00000088     D
UCB$L_MB_WIOQFL                   00000084     D
UCB$L_MEDIA                       0000008C     D
UCB$L_NT_DATSSB                   00000074     D
UCB$L_NT_INTSSB                   00000078     D
UCB$L_OPCNT                       00000060     D
UCB$L_OWNUIC                      0000001C     D
UCB$L_PID                         00000028     D
UCB$L_RQBL                        00000004     D
UCB$L_RQFL                        00000000     D
UCB$L_SVAPTE                      00000068     D
UCB$L_SVPN                        00000064     D
UCB$L_TT_DECHAR                   000000A8     D
UCB$L_TT_RDUE                     0000008C     D
UCB$L_TT_RTIMOU                   000000B8     D
UCB$L_VCB                         00000030     D
UCB$M_ONLINE                    = 00000010     D
UCB$M_VALID                     = 00000800     D
UCB$T_PARTNER                     0000000C     D
UCB$W_BCNT                        0000006E     D
UCB$W_BCR                         00000096     D
UCB$W_BOFF                        0000006C     D
UCB$W_BUFQUO                      00000018     D
UCB$W_BYTESTOGO                   0000003E     D
UCB$W_CHARGE                      0000004A     D
UCB$W_CYLINDERS                   0000003E     D
UCB$W_DA                          0000008C     D
UCB$W_DC                          0000008E     D
UCB$W_DEVBUFSIZ                   0000003A     D
UCB$W_DEVSTS                      0000005A     D
```

```
UCB$W_DIRSEQ                    00000088    D
UCB$W_DSTADDR                   00000018    D
UCB$W_DX_BCR                    000000A4    D
UCB$W_EC1                       00000090    D
UCB$W_EC2                       00000092    D
UCB$W_ERRCNT                    00000072    D
UCB$W_FUNC                      0000007E    D
UCB$W_MB_SEED                   00000000    D
UCB$W_MSGCNT                    00000016    D
UCB$W_MSGMAX                    00000014    D
UCB$W_NT_CHAN                   0000007C    D
UCB$W_OFFSET                    0000008A    D
UCB$W_REFC                      00000050    D
UCB$W_SIZE                      00000008    D
UCB$W_SRCADDR                   0000001A    D
UCB$W_STS                       00000058    D
UCB$W_TT_DESIZE                 000000A5    D
UCB$W_UNIT                      00000048    D
UCB$W_VPROT                     0000001A    D
UCB_BLINK                       0000000A R  D   03
VEC$B_DATAPATH                  00000013    D
VEC$B_NUMREG                    00000012    D
VEC$C_LENGTH                    00000024    D
VEC$K_LENGTH                    00000024    D
VEC$L_ADP                       00000014    D
VEC$L_IDB                       00000008    D
VEC$L_INITIAL                   0000000C    D
VEC$L_START                     0000001C    D
VEC$L_UNITDISC                  00000020    D
VEC$L_UNITINIT                  00000018    D
VEC$Q_DISPATCH                  00000000    D
VEC$W_MAPREG                    00000010    D
VMS$QIO                         ********  X   00
WRITEVBLK                       ********  X   00
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFF4 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000103 (259.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
WORK	0000000E (14.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	00000B9E (2974.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$ARGS	=00000007	285 (1)	259 (1) 271 (1) 285 (1)
$$T1	=00000020	285 (1)	259 (1) 271 (1) 285 (1)
$ER	=00000006	1238 (13)	#-1238 (13) #-458 (1) #-666 (4) #-894 (6)
			#-925 (6)
$MODULE	00000000-R	325 (1)	1238 (13) 458 (1) 666 (4) 894 (6)
			925 (6)
$_KNOWN	=00000007	380 (1)	372 (1)
ACF	FFFFFFC0	733 (5)	738 (6)
ACF$B_AFLAG	0000000B		#-918 (6)
ACF$B_AUNIT	0000000A		#-1388 (15) #-1454 (15) #-1575 (16) #-901 (6)
ACF$B_CNUMVEC	00000013		#-1370 (15) #-1413 (15) #-915 (6) #-917 (6)
ACF$B_CUNIT	00000012		#-1329 (15) #-1448 (15) #-1453 (15) #-1549 (16)
			#-910 (6)
ACF$C_LENGTH	00000020		733 (5)
ACF$L_ADAPTER	00000000		#-1362 (15) #-1384 (15) #-1402 (15) #-1417 (15)
			#-1421 (15) #-1495 (15) #-896 (6)
ACF$L_CONFIGREG	00000004		#-897 (6)
ACF$L_CONTRLREG	0000000C		#-1403 (15) #-903 (6)
ACF$L_DEVNAME	00000014		1303 (15) #-1315 (15) #-1350 (15) 1352 (15)
			#-752 (6)
ACF$L_DRVNAME	00000018		#-1298 (15) #-1353 (15) 1355 (15) #-854 (6)
ACF$W_AVECTOR	00000008		#-899 (6)
ACF$W_CVECTOR	00000010		#-1409 (15) #-905 (6) #-908 (6)
ACF$W_MAXUNITS	0000001C		#-912 (6)
ADP$B_TYPE	0000000A		#-1003 (8) #-1056 (9) #-1157 (11)
ADP$C_DRADPLEN	00000014		#-1114 (10)
ADP$C_MBAADPLEN	00000014		#-960 (7)
ADP$C_UBAADPLEN	00000070		#-1047 (9)
ADP$L_CRB	00000010		#-1007 (8) #-1161 (11) #-1363 (15) #-1385 (15)
ADP$L_CSR	00000000		#-1004 (8) #-1058 (9) #-1158 (11) #-1496 (15)
			#-1621 (17) #-1624 (17) #-1728 (19)
ADP$L_DPQFL	00000014		1060 (9)
ADP$L_LINK	00000004		#-1069 (9) 1709 (19) #-1710 (19) #-1714 (19)
			1725 (19) #-1726 (19)
ADP$L_MRQFL	0000001C		1063 (9)
ADP$L_VECTOR	00000010		#-1073 (9) #-1422 (15)
ADP$W_ADPTYPE	0000000E		#-1006 (8) #-1057 (9) #-1160 (11)
ADP$W_MRBITMAP	00000026		#-1067 (9)
ADP$W_SIZE	00000008		#-1055 (9) #-1118 (10) #-964 (7)
ADP$W_TR	0000000C		#-1005 (8) #-1059 (9) #-1159 (11)
ADPLINK	00000AFA-R	1708 (19)	#-1011 (8) #-1071 (9) #-1165 (11)
ALLOC$_ACMODE	=00000010	259 (1)	
ALLOC$_DEVNAM	=00000004	259 (1)	
ALLOC$_NARGS	=00000004	259 (1)	
ALLOC$_PHYBUF	=0000000C	259 (1)	
ALLOC$_PHYLEN	=00000008	259 (1)	
AT$_DR	=00000002		#-1160 (11) 368 (1)
AT$_MBA	=00000000		#-1006 (8) #-1639 (17) 364 (1) 366 (1)
			#-893 (6)
AT$_UBA	=00000001		#-1057 (9) #-1360 (15) #-1419 (15) #-1493 (15)

```
                                                        #-1613  (17)    363  (1)    365  (1)    367  (1)
                                                          369   (1)
AUNKFUNC                    00000008-R   330  (1)        458   (1)
BIT...                     =00000004     402  (1)        276   (1)    277  (1)    287  (1)    290  (1)
                                                          402   (1)
BUG$CHECK                   00000716-R  1235  (13)     #-1239  (13)
CMK$_APBOOT                =00000004     277  (1)
CMK$_ASTEXIT               =00000000     277  (1)
CMK$_CANTIM                =0000000B     277  (1)
CMK$_HIBER                 =00000007     277  (1)
CMK$_INITSCB               =00000008     277  (1)
CMK$_LAST                  =0000000E     277  (1)
CMK$_MMENABLE              =00000003     277  (1)
CMK$_RCONSOLE              =00000001     277  (1)
CMK$_REFRESH               =00000005     277  (1)
CMK$_SCHDWK                =00000006     277  (1)
CMK$_SETIMR                =0000000C     277  (1)
CMK$_SETIPL                =00000009     277  (1)
CMK$_SETPRT                =0000000D     277  (1)
CMK$_SGIPR                 =0000000A     277  (1)
CMK$_TCONSOLE              =00000002     277  (1)
CONSOLEAAA_FLAGS            00000000-XR             #-1763  (19)  #-1775  (19)    318  (1)
CR                         =0000000D     252  (1)     358   (1)
CRB$B_TYPE                  0000000A                 #-1146  (11)  #-1380  (15)  #-992  (8)
CRB$C_LENGTH                00000038                 #-1102  (10)  #-948   (7)
CRB$K_LENGTH                00000038                 #-1373  (15)
CRB$L_INTD                  00000014                 #-1000  (8)   #-1001  (8)   #-1008  (8)    1133  (11)
                                                       1142  (11)    1143  (11)    1144  (11)    1145  (11)
                                                     #-1154  (11)  #-1155  (11)  #-1162  (11)  #-1328  (15)
                                                     #-1364  (15)  #-1387  (15)    1389  (15)    1414  (15)
                                                     #-1491  (15)  #-1508  (15)  #-1574  (16)  #-1618  (17)
                                                     #-1629  (17)  #-1644  (17)  #-1646  (17)  #-648   (4)
                                                       979   (8)     988   (8)     989   (8)     990   (8)
                                                       991   (8)
CRB$L_LINK                  00000010                 #-1386  (15)  #-1489  (15)  #-1573  (16)  #-631   (4)
CRB$L_WQBL                  00000004                 #-1148  (11)  #-1382  (15)  #-994   (8)
CRB$L_WQFL                  00000000                   1147  (11)    1148  (11)    1381  (15)    1382  (15)
                                                       993   (8)     994   (8)
CRB$W_REFC                  0000000C                 #-1149  (11)  #-1447  (15)  #-1548  (16)  #-995   (8)
CRB$W_SIZE                  00000008                 #-1106  (10)  #-1379  (15)  #-952   (7)
CREATE_CRB                  0000080B-R  1369  (15)  #-1359  (15)  #-1361  (15)
CREATE_DDB                  000007AF-R  1338  (15)  #-1319  (15)
CREATE_IDB                  00000857-R  1393  (15)  #-1383  (15)
CREATE_UCB                  000008D8-R  1435  (15)  #-1334  (15)  #-1365  (15)
CREATE_VEC                  00000881-R  1408  (15)
CRLF_LENGTH                =00000002     253  (1)   #-491   (1)
CTL$GW_CHINDX               0000003A-R  337  (1)
CTL$GW_NMIOCH               00000038-R  334  (1)
DB_ERROR                    000009D9-R  1541  (16)  #-1311  (15)  #-1342  (15)  #-1376  (15)  #-1397  (15)
                                                     #-1412  (15)  #-1426  (15)  #-1439  (15)  #-1475  (15)
DB_EXIT                     000009D2-R  1517  (15)  #-1331  (15)
DDB$B_TYPE                  0000000A                 #-1348  (15)
DDB$K_LENGTH                00000034                 #-1339  (15)
DDB$L_DDT                   0000000C                 #-1502  (15)
DDB$L_LINK                  00000000                 #-1318  (15)  #-1320  (15)    1344  (15)  #-1345  (15)
                                                     #-642   (4)
DDB$L_UCB                   00000004                 #-1326  (15)    1456  (15)  #-1616  (17)  #-1642  (17)
```

```
                                                #-620    (4)      634    (4)
DDB$T_DRVNAME                    00000024                 1355   (15)
DDB$T_NAME                       00000014                 1321   (15)     1352   (15)
DDB$W_SIZE                       00000008                #-1347  (15)
DDB_BLINK                        00000006-R   403   (1)  #-1344  (15)   #-1584  (16)
DDT$L_UNITINIT                   00000018                #-1503  (15)
DIR...                          =FFFFFFFF     733   (5)    733   (5)
DPT$B_ADPTYPE                    0000000C                #-1360  (15)   #-1419  (15)   #-1493  (15)   #-1613  (17)
                                                         #-1639  (17)   #-876   (6)
DPT$B_FLAGS                      0000000D                 1359   (15)     1383   (15)     1572   (16)
DPT$B_REFC                       0000000B                #-1349  (15)   #-1583  (16)   #-660    (4)   #-921    (6)
                                                         #-923   (6)
DPT$L_FLINK                      00000000                #-558   (3)    #-657   (4)    #-774    (6)
DPT$T_NAME                       0000001E                 563    (3)    #-777   (6)     853     (6)
DPT$V_SUBCNTRL                  =00000000                #-1359  (15)   #-1383  (15)   #-1572   (16)
DPT$W_MAXUNITS                   00000016                #-911   (6)
DPT$W_SIZE                       00000008                #-846   (6)
DPT$W_UCBSIZE                    0000000E                #-1436  (15)
DR$INITIAL                       00000000-XR              1155   (11)     1166   (11)     1177   (11)     303    (1)
DR$INT                           00000000-XR              1175   (11)     303    (1)
DRACRB                           000006F7-R   1173  (11)  1133   (11)     1178   (11)
DRACRBLEN                       =00000010     1178  (11) #-1133  (11)
DS$BREAK                         00000000-XR              1769   (19)     296    (1)
DS$GK_SID                        00000000-XR              312    (1)    #-439   (1)    #-460    (1)
DS$GL_FLAGS                      00000000-XR              312    (1)      434   (1)      464    (1)      469    (1)
DS$GPHARD                        00000000-XR              300    (1)
DS$K_CCB                         00000000-XR              305    (1)      335   (1)      338    (1)
DS$K_ERROR                      =00000002     287   (1)
DS$K_NORMAL                     =00000001     287   (1)
DS$K_PRINTB                     =00000002     276   (1)
DS$K_PRINTF                     =00000001     276   (1)  #-1307  (15)   #-806   (6)    #-834    (6)    #-881    (6)
DS$K_PRINTI                     =00000000     276   (1)
DS$K_PRINTX                     =00000003     276   (1)
DS$K_SEVERE                     =00000004     287   (1)
DS$K_SUBSYS                     =00000066     287   (1)    287    (1)
DS$K_TYPE_ABORT_PROGRAM         =00000014     276   (1)
DS$K_TYPE_ABORT_TEST            =00000013     276   (1)
DS$K_TYPE_COMMAND_ERR           =00000015     276   (1)
DS$K_TYPE_COMMAND_OUT           =00000016     276   (1)
DS$K_TYPE_CRD_AUTOTEST          =0000001A     276   (1)
DS$K_TYPE_DS_PROMPT             =00000001     276   (1)
DS$K_TYPE_DS_START              =0000001D     276   (1)
DS$K_TYPE_ERRDEV                =00000008     276   (1)
DS$K_TYPE_ERRHARD               =00000006     276   (1)
DS$K_TYPE_ERROR_BODY            =00000009     276   (1)
DS$K_TYPE_ERROR_END             =0000000A     276   (1)
DS$K_TYPE_ERRPREP               =0000001B     276   (1)
DS$K_TYPE_ERRSOFT               =00000007     276   (1)
DS$K_TYPE_ERRSUP                =00000004     276   (1)
DS$K_TYPE_ERRSYS                =00000005     276   (1)
DS$K_TYPE_ERR_HALT              =0000000D     276   (1)
DS$K_TYPE_EXCEPTION             =0000000C     276   (1)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B     276   (1)
DS$K_TYPE_FIRST_PASS            =00000011     276   (1)
DS$K_TYPE_GENERAL               =00000000     276   (1)
DS$K_TYPE_GENERAL_ERROR         =00000003     276   (1)
DS$K_TYPE_NO_TESTS              =00000012     276   (1)
```

```
DS$K_TYPE_PARAM_ERROR           =0000001C     276     (1)
DS$K_TYPE_PROGRAM_END           =00000010     276     (1)
DS$K_TYPE_PROGRAM_INFO          =00000017     276     (1)
DS$K_TYPE_PROGRAM_START         =0000000F     276     (1)
DS$K_TYPE_QIO_INVADP            =00000024     276     (1)    #-882    (6)
DS$K_TYPE_QIO_NODRIVER          =00000022     276     (1)    #-1308   (15)   #-807    (6)
DS$K_TYPE_QIO_WRONGVER          =00000023     276     (1)    #-835    (6)
DS$K_TYPE_SCRIPT_ECHO           =00000021     276     (1)
DS$K_TYPE_SCRIPT_PNF            =0000001E     276     (1)
DS$K_TYPE_SCRIPT_PROMPT         =00000020     276     (1)
DS$K_TYPE_SCRIPT_SKIP           =0000001F     276     (1)
DS$K_TYPE_SEQUENCE_ERROR        =00000019     276     (1)
DS$K_TYPE_START_ERR             =00000018     276     (1)
DS$K_TYPE_START_LIST            =00000025     276     (1)
DS$K_TYPE_SUMMARY               =0000000E     276     (1)
DS$K_TYPE_USER_PROMPT           =00000002     276     (1)
DS$K_WARNING                    =00000000     287     (1)
DS$LOAD                          00000000-XR                  301    (1)      800    (6)      821    (6)
DS$M_ABRTFLG                    =00000040     290     (1)
DS$M_BADTIME                    =00100000     290     (1)
DS$M_BATCH                      =00400000     290     (1)
DS$M_BRKCLR                     =00001000     290     (1)
DS$M_BRKPT                      =00000800     290     (1)
DS$M_CHARFLG                    =00000100     290     (1)
DS$M_CMDFLG                     =00000080     290     (1)
DS$M_CTRLC                      =00000001     290     (1)
DS$M_CTRLO                      =00010000     290     (1)
DS$M_DEVFLG                     =00000200     290     (1)
DS$M_DISABLCC                   =01000000     290     (1)
DS$M_DONFLG                     =00002000     290     (1)
DS$M_ERRFLG                     =00000010     290     (1)
DS$M_EXCEPT                     =00080000     290     (1)
DS$M_EXETST                     =00040000     290     (1)
DS$M_HLTFLG                     =00000008     290     (1)
DS$M_LODFLG                     =00000002     290     (1)
DS$M_MEMMGT                     =00008000     290     (1)
DS$M_NI                         =04000000     290     (1)
DS$M_OUTPUT                     =00800000     290     (1)
DS$M_RUBFLG                     =00000020     290     (1)
DS$M_SCRIPT                     =00200000     290     (1)
DS$M_SETIMR                     =02000000     290     (1)
DS$M_STRFLG                     =00000004     290     (1)
DS$M_SUBT                       =00004000     290     (1)
DS$M_SYSFLG                     =00000400     290     (1)
DS$M_TIMED                      =02000000     290     (1)
DS$M_TIMED_OUT                  =08000000     290     (1)
DS$M_TIMRON                     =00020000     290     (1)
DS$M_TODR_ACTIVE                 00000000-XR                  #-1762   (19)   #-1774   (19)     318    (1)
DS$V_ABRTFLG                    =00000006     290     (1)
DS$V_BADTIME                    =00000014     290     (1)
DS$V_BATCH                      =00000016     290     (1)
DS$V_BRKCLR                     =0000000C     290     (1)
DS$V_BRKPT                      =0000000B     290     (1)
DS$V_CHARFLG                    =00000008     290     (1)
DS$V_CMDFLG                     =00000007     290     (1)
DS$V_CTRLC                      =00000000     290     (1)
DS$V_CTRLO                      =00000010     290     (1)
```

```
DS$V_DEVFLG                     =00000009      290     (1)
DS$V_DISABLCC                   =00000018      290     (1)
DS$V_DONFLG                     =0000000D      290     (1)
DS$V_ERRFLG                     =00000004      290     (1)
DS$V_EXCEPT                     =00000013      290     (1)
DS$V_EXETST                     =00000012      290     (1)
DS$V_HLTFLG                     =00000003      290     (1)
DS$V_LODFLG                     =00000001      290     (1)
DS$V_MEMMGT                     =0000000F      290     (1)
DS$V_NI                         =0000001A      290     (1)    #-434    (1)    #-464    (1)    #-469    (1)
DS$V_OUTPUT                     =00000017      290     (1)
DS$V_RUBFLG                     =00000005      290     (1)
DS$V_SCRIPT                     =00000015      290     (1)
DS$V_SETIMR                     =00000019      290     (1)
DS$V_STRFLG                     =00000002      290     (1)
DS$V_SUBT                       =0000000E      290     (1)
DS$V_SYSFLG                     =0000000A      290     (1)
DS$V_TIMED                      =00000019      290     (1)
DS$V_TIMED_OUT                  =0000001B      290     (1)
DS$V_TIMRON                     =00000011      290     (1)
DS$_AP_NORMAL_BREAK             =00660150      287     (1)
DS$_ARITH                       =006600D0      287     (1)
DS$_ASBE                        =00660118      287     (1)
DS$_BADLINK                     =006600F0      287     (1)
DS$_BADTYPE                     =006600E8      287     (1)
DS$_BIIC                        =00660120      287     (1)
DS$_CHME                        =006600A8      287     (1)
DS$_CHMK                        =006600E0      287     (1)
DS$_DEVNAME                     =00660108      287     (1)
DS$_ERROR                       =00660002      287     (1)
DS$_FHWE                        =00660068      287     (1)
DS$_FRAGBUF                     =00660080      287     (1)
DS$_ICBUSY                      =006600C8      287     (1)
DS$_ICERR                       =006600C0      287     (1)
DS$_IHWE                        =00660060      287     (1)
DS$_ILLCHAR                     =00660018      287     (1)
DS$_ILLPAGCNT                   =00660078      287     (1)
DS$_ILLUNIT                     =00660100      287     (1)
DS$_INIT_FAIL                   =00660140      287     (1)
DS$_INSFMEM                     =00660C50      287     (1)
DS$_INVCPU                      =00660130      287     (1)
DS$_IPL2HI                      =006600B8      287     (1)
DS$_IVADDR                      =00660040      287     (1)
DS$_IVVECT                      =00660038      287     (1)
DS$_KRNLSTK                     =00660090      287     (1)
DS$_LOGIC                       =00660070      287     (1)
DS$_MCHK                        =00660088      287     (1)
DS$_MEM_ALLOC_ERR               =00660138      287     (1)
DS$_MMOFF                       =00660058      287     (1)
DS$_NEEDUNIT                    =006600F8      287     (1)
DS$_NODE                        =00660128      287     (1)
DS$_NOPCS                       =00660110      287     (1)
DS$_NORMAL                      =00660001      287     (1)
DS$_NOSUPPORT                   =006600B1      287     (1)    #-766    (6)
DS$_NOTALLOWMP                  =00660148      287     (1)
DS$_NOTDON                      =00660030      287     (1)
DS$_NOTIMP                      =006600B0      287     (1)      287    (1)
```

```
DS$_NULLSTR                     =00660010      287     (1)
DS$_OVERFLOW                    =00660008      287     (1)
DS$_POWER                       =00660098      287     (1)
DS$_PROGERR                     =00660020      287     (1)
DS$_SEVERE                      =00660004      287     (1)
DS$_TRANSL                      =006600A0      287     (1)
DS$_TRUNCATE                    =00660028      287     (1)
DS$_UNEXPINT                    =006600D8      287     (1)
DS$_UNSUPPDEV                   =00660158      287     (1)
DS$_VASFULL                     =00660048      287     (1)
DS$_WARNING                     =00660000      287     (1)       287     (1)
DSA$GL_FLAGS                     0000FE00                       #-1237  (13)
DSA$GL_SID                       0000FE14                       #-1751  (19)    #-1753  (19)
DSA$M_HALT                      =00000001                       #-1237  (13)
DSR$COMPLETION                   00000000-XR                     301     (1)       815   (6)       823   (6)
DSX$PRINT                        00000000-XR                    1309     (15)      300   (1)       808   (6)       836   (6)
                                                                883     (6)
DS_ERRSUP                        00000000-XR                    1238     (13)      300   (1)       458   (1)       666   (4)
                                                                894     (6)       925   (6)
DVR$B_CNUMVEC                    0000000B                       #-914    (6)
DVR$C_LEN                        00000014                        845     (6)       913   (6)
DVR$L_IDENT                      00000010                       #-828    (6)     #-831   (6)
DVR$W_SIZE                       00000008                       #-826    (6)
DVR$_IDENT                      =FFFF0001                       #-827    (6)     #-830   (6)
DYN$C_ADP                       =00000001                       #-1003   (8)     #-1056  (9)     #-1157  (11)
DYN$C_CRB                       =00000005                       #-1146   (11)    #-1380  (15)    #-992   (8)
DYN$C_DDB                       =00000006                       #-1348   (15)
DYN$C_IDB                       =00000009                       #-1151   (11)    #-1401  (15)    #-997   (8)
DYN$C_UCB                       =00000010                       #-1443   (15)
EXE$ALONONPAGED                  00000000-XR                     1673    (18)      298   (1)       813   (6)
EXE$DEANONPAGED                  00000000-XR                     1695    (18)      298   (1)       841   (6)
EXE$FORKDSPTH                    00000000-XR                     309     (1)
EXE$GL_PWRDONE                   00000004-R     327     (1)      #-1782  (19)
EXE$MAXACMODE                    00000707-R     1199    (12)
EXE$PWRTIMCHK                    00000B29-R     1749    (19)
EXE$QIO                          00000000-R     432     (1)
FILE                             FFFFFFF4       733     (5)      #-787   (6)     #-788   (6)     #-789   (6)       791   (6)
FILEDEF                          FFFFFFE4       733     (5)      #-792   (6)      797    (6)
FILEDESC                         FFFFFFEC       733     (5)      #-790   (6)     #-791   (6)       798   (6)       804   (6)
                                                                832     (6)
FILL_ACF                         000003A6-R     852     (6)      #-779   (6)
FIND_ADP                         00000B11-R     1723    (19)     #-889   (6)
HP$A_DEVICE                      00000018                       #-887    (6)     #-902   (6)
HP$A_LINK                        00000020                       #-764    (6)     #-857   (6)     #-860   (6)
HP$B_DRIVE                       0000000B                       #-900    (6)     #-909   (6)
HP$Q_DEVICE                      00000000                       #-743    (6)     #-744   (6)       803   (6)       878   (6)
HP$T_TYPE                        00000026                        755     (6)     #-865   (6)       870   (6)       879   (6)
HP$W_VECTOR                      00000024                       #-898    (6)     #-904   (6)     #-907   (6)
IDB$B_TYPE                       0000000A                       #-1151   (11)    #-1401  (15)    #-997   (8)
IDB$C_LENGTH                     00000034                       #-1108   (10)    #-954   (7)
IDB$K_LENGTH                     00000034                       #-1394   (15)
IDB$L_ADP                        00000010                       #-1009   (8)     #-1163  (11)    #-1402  (15)    #-1620  (17)
                                                                #-1623   (17)
IDB$L_CSR                        00000000                       #-1153   (11)    #-1403  (15)    #-1487  (15)    #-1492  (15)
                                                                #-1619   (17)    #-1645  (17)    #-999   (8)
IDB$L_UCBLST                     00000014                       #-1389   (15)    #-1449  (15)    #-1550  (16)    #-1576  (16)
IDB$W_SIZE                       00000008                       #-1112   (10)    #-1400  (15)    #-958   (7)
```

```
IDB$W_UNITS                     0000000C                        #-1152  (11)   #-1404  (15)   #-998   (8)
INIT_CNTRL                      00000964-R  1479    (15)      #-1473  (15)
INIT_DB                         00000945-R  1468    (15)
INIT_UNIT                       00000978-R  1485    (15)      #-1481  (15)
INI_DRADP                       00000616-R  1098    (10)        893   (6)
INI_MBADP                       000004A0-R   944    (7)         893   (6)
INI_UBADP                       00000594-R  1044    (9)         893   (6)
IO$S_FCODE                     =00000006                      #-446   (1)
IO$TESTCSR_L                    00000000-XR                     1498  (15)      299   (1)
IO$TESTCSR_W                    00000000-XR                     1626  (17)      299   (1)
IO$V_FCODE                     =00000000                      #-446   (1)
IO$_READPROMPT                 =00000037                      #-485   (1)
IO$_WRITEVBLK                  =00000030                      #-456   (1)
IOC$GL_ADPLIST                  00000000-XR                     1709  (19)     1725  (19)      297   (1)   #-545   (2)
                                                                607   (4)
IOC$GL_DEVLIST                  00000000-XR                     1317  (15)      297   (1)   #-544   (2)   #-617   (4)
                                                              #-642   (4)
IOC$GL_DPTLIST                  00000000-XR                      297   (1)      540   (2)      555   (3)     655   (4)
                                                                772   (6)      847   (6)
IOC$INITDRV                     00000000-XR                     1471  (15)      308   (1)
IOC$IOPOST                      00000000-XR                      309   (1)
IOC$REINITDRV                   00000000-XR                      309   (1)
IOC$RTN                         00000000-XR                    #-1506  (15)      311   (1)
IOGEN$CNTRL_INI                 00000A53-R  1612    (17)      #-1483  (15)
IOGEN$CONN_VEC                  00000000-XR                     1428  (15)      305   (1)
IOGEN$LOCK_IODB                 00000A4B-R  1596    (16)      #-1294  (15)
IOGEN$UNLK_IODB                 00000A4F-R  1599    (16)      #-1518  (15)   #-1588  (16)
LASTCHAR                        00000004-R   393    (1)       #-481   (1)    #-488   (1)
LENGTH                          FFFFFFE0     733    (5)         794   (6)    #-812   (6)
LF                             =0000000A     251    (1)         358   (1)    #-488   (1)
LOAD$_ADDRESS                  =00000010     285    (1)       #-820   (6)
LOAD$_DEFAULT                  =00000008     285    (1)
LOAD$_FILE                     =00000004     285    (1)
LOAD$_LENGTH                   =0000000C     285    (1)       #-819   (6)
LOAD$_NARGS                    =00000007     285    (1)
LOAD$_RETLEN                   =00000014     285    (1)
LOAD$_RETREC                   =00000018     285    (1)
LOAD$_VBN                      =0000001C     285    (1)
LOAD_DRIVER                     000002D8-R   786    (6)       #-776   (6)
LOAD_FLAGS                      00000005-R   395    (1)       #-1293  (15)   #-1343  (15)   #-1377  (15)   #-1398  (15)
                                                              #-1440  (15)     1481  (15)     1546  (16)     1565  (16)
                                                                1571  (16)     1582  (16)
LOAD_M_CRB                     =00000002     402    (1)       #-1377  (15)
LOAD_M_DDB                     =00000001     402    (1)       #-1343  (15)
LOAD_M_IDB                     =00000004     402    (1)       #-1398  (15)
LOAD_M_UCB                     =00000008     402    (1)       #-1440  (15)
LOAD_V_CRB                     =00000001     402    (1)       #-1481  (15)   #-1571  (16)
LOAD_V_DDB                     =00000000     402    (1)       #-1582  (16)
LOAD_V_IDB                     =00000002     402    (1)       #-1565  (16)
LOAD_V_UCB                     =00000003     402    (1)       #-1546  (16)
LOC$DPT                         00000161-R   553    (3)       #-1300  (15)
LOC$PTDESC                      00000000-XR                     310   (1)      757   (6)
LOCAL                           FFFFFFC0     733    (5)         737   (6)
MBA$INITIAL                     00000000-XR                     1001  (8)      1012  (8)      1023  (8)      302   (1)
MBA$INT                         00000000-XR                     1021  (8)      302   (1)
MBACRB                          00000584-R  1019    (8)         1024  (8)      979   (8)
MBACRBLEN                      =00000010    1024    (8)       #-979   (8)
```

```
MBINT_DISP                       0000003C-R    340   (1)     #-1415  (15)
NUMUBAVEC                       =00000080      1042  (9)     #-1047  (9)     #-1074  (9)
PR$_IPL                         =00000012                    #-1013  (8)     #-1053  (9)     #-1077  (9)     #-1131  (11)
                                                             #-1167  (11)    #-1480  (15)    #-1513  (15)    #-1597  (16)
                                                             #-1600  (16)    #-605   (4)     #-663   (4)     #-977   (8)
PR$_SID_TYP8NN                  =00000006                    #-1751  (19)
PR$_SID_TYP8RR                  =00000011                    #-1753  (19)
PR$_TODR                        =0000001B                    #-1780  (19)
PR$_TXCS                        =00000022                    #-1765  (19)
PR$_TXDB                        =00000023                    #-1767  (19)
PREADVBLK                        00000000-XR                   306  (1)
PSL$S_PRVMOD                    =00000002                    #-1201  (12)    #-1203  (12)
PSL$V_IS                        =0000001A      277   (1)
PSL$V_PRVMOD                    =00000016                    #-1201  (12)    #-1203  (12)
QIO$ADDUNIT                      00000259-R    735   (6)
QIO$ALLOCATE                     00000AC1-R    1671  (18)    #-1049  (9)     #-1103  (10)    #-1109  (10)    #-1115  (10)
                                                             #-1340  (15)    #-1374  (15)    #-1395  (15)    #-1437  (15)
                                                             #-949   (7)     #-955   (7)     #-961   (7)
QIO$CLEANUP                      00000192-R    601   (4)
QIO$DEALLOCATE                   00000AE4-R    1691  (18)    #-1123  (10)    #-1125  (10)    #-1559  (16)    #-638   (4)
                                                             #-643   (4)     #-649   (4)     #-651   (4)     #-969   (7)
                                                             #-971   (7)
QIO$INITIALIZE                   0000013E-R    538   (2)
QIO$LOAD_DB                      00000733-R    1290  (15)      920   (6)
QIO$L_ALLOCATE                   00000000-R    390   (1)     #-1681  (18)    #-1694  (18)    #-546   (2)     #-664   (4)
QIO$_ASTADR                     =00000014      271   (1)
QIO$_ASTPRM                     =00000018      271   (1)
QIO$_CHAN                       =00000008      271   (1)     #-436   (1)
QIO$_EFN                        =00000004      271   (1)
QIO$_FUNC                       =0000000C      271   (1)       447   (1)     #-486   (1)
QIO$_IOSB                       =00000010      271   (1)     #-499   (1)       504   (1)
QIO$_NARGS                      =0000000C      271   (1)
QIO$_P1                         =0000001C      271   (1)     #-476   (1)     #-480   (1)     #-502   (1)
QIO$_P2                         =00000020      271   (1)     #-475   (1)     #-479   (1)
QIO$_P3                         =00000024      271   (1)
QIO$_P4                         =00000028      271   (1)
QIO$_P5                         =0000002C      271   (1)     #-496   (1)
QIO$_P6                         =00000030      271   (1)
QIOCLEAN_CPU                     00000000-XR                   310   (1)       609   (4)
READVBLK                         00000000-XR                   306   (1)       472   (1)
REMOTE_NODE_CONTROL_LEVEL        00000000-XR                   313   (1)       319   (1)       503   (1)
SAVABS...                       =FFFFFFC0      733   (5)
SCAN$ALPHA                       00000000-XR                   308   (1)       745   (6)
SCAN$SEARCHLIST                  00000000-XR                   308   (1)       873   (6)
SCB$L_SFTLVL4                    00000090                    #-612   (4)     #-613   (4)
SCB$L_SFTLVL8                    000000A0                    #-610   (4)     #-611   (4)
SCB_BASE                         00000000-XR                  1139   (11)      1141   (11)      304   (1)       603   (4)
                                                              739   (6)       985   (8)       987   (8)
SCB_IMAGE                        00000000-XR                   304   (1)       604   (4)
SEND_CONSOLE_RESPONSE            00000000-XR                   313   (1)       319   (1)       477   (1)       493   (1)
                                                              497   (1)
SIZ...                          =00000001      402   (1)       290   (1)       402   (1)
SS$_INSFMEM                      00000000-XR                 #-1127  (10)      307   (1)     #-973   (7)
SS$_IVCHAN                       00000000-XR                   307   (1)
SS$_NORMAL                       00000000-XR                 #-1014  (8)     #-1168  (11)    #-1730  (19)      307   (1)
                                                             #-567   (3)
SYSG$_INVDPTINI                 =007C800A                    #-1474  (15)
```

```
SYSG$_INVVEC                    =007C802A                       #-1411    (15)
SYSG$_VECINUSE                  =007C801A                       #-1425    (15)
TODR_BYTE_CNT                    00000000-XR                    #-1761    (19)   #-1770   (19)     318    (1)
TODR_STORE                       00000000-XR                    #-1777    (19)     318    (1)
TXCS$V_RDY                       00000000-XR                    #-1766    (19)     321    (1)
T_ADAPTER_TYPE                   000000C2-R    362    (1)       #-875     (6)
T_CRLF                           000000C0-R    357    (1)         492     (1)
T_INVADP                         00000095-R    353    (1)         880     (6)
T_KNOWN_DEV                      000000C9-R    371    (1)         380     (1)      872    (6)
T_NODRIVER                       00000046-R    347    (1)        1306    (15)      805    (6)
T_WRONGVER                       00000068-R    350    (1)         833     (6)
UBA$INITIAL                      00000000-XR                      1070    (9)      320    (1)
UBA$UNEXINT                      00000000-XR                      1075    (9)    #-1423   (15)     302    (1)
UBADP_CPU                        00000000-XR                      1072    (9)      304    (1)
UBINTSZ                          00000000-XR                    #-1047    (9)      320    (1)
UBINT_DISP                       00000044-R    344    (1)
UCB$B_SLAVE                      00000074                       #-1454    (15)
UCB$B_TYPE                       0000000A                       #-1443    (15)
UCB$L_ASTQBL                     00000010                       #-1445    (15)
UCB$L_ASTQFL                     0000000C                        1444     (15)     1445   (15)
UCB$L_CRB                        00000020                       #-1327    (15)   #-1446   (15)   #-1617   (17)   #-1643   (17)
                                                                #-621     (4)
UCB$L_DDB                        00000024                       #-1450    (15)
UCB$L_IOQBL                      00000044                       #-1452    (15)
UCB$L_IOQFL                      00000040                        1451     (15)     1452   (15)
UCB$L_LINK                       0000002C                       #-1332    (15)     1459   (15)     634    (4)    #-635    (4)
UCB$M_ONLINE                    =00000010                       #-1500    (15)
UCB$M_VALID                     =00000800                       #-1472    (15)
UCB$W_SIZE                       00000008                       #-1442    (15)
UCB$W_STS                        00000058                       #-1472    (15)   #-1500   (15)
UCB$W_UNIT                       00000048                       #-1329    (15)   #-1453   (15)
UCB_BLINK                        0000000A-R    406    (1)       #-1462    (15)   #-1547   (16)
VEC$K_LENGTH                     00000024                       #-1372    (15)   #-1430   (15)
VEC$L_ADP                        00000014                       #-1008    (8)    #-1162   (11)   #-1417   (15)
VEC$L_IDB                        00000008                       #-1000    (8)    #-1154   (11)   #-1328   (15)   #-1364   (15)
                                                                #-1387    (15)   #-1418   (15)   #-1491   (15)   #-1574   (16)
                                                                #-1618    (17)   #-1644   (17)   #-648    (4)
VEC$L_INITIAL                    0000000C                       #-1001    (8)    #-1155   (11)   #-1629   (17)   #-1646   (17)
VEC$L_UNITINIT                   00000018                       #-1508    (15)
VEC$Q_DISPATCH                   00000000                       #-1415    (15)
VMS$QIO                          00000000-XR                      305     (1)      438    (1)
WRITEVBLK                        00000000-XR                      306     (1)      467    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACFDEF	2	257 (1)	257 (1)
$ADPDEF	2	258 (1)	258 (1)
$ALLOCDEF	1	259 (1)	259 (1)
$CCBDEF	1	260 (1)	260 (1)
$CRBDEF	1	261 (1)	261 (1)
$DCDEF	9	262 (1)	262 (1)
$DDBDEF	1	263 (1)	263 (1)
$DDTDEF	1	264 (1)	264 (1)
$DEF	1	290 (1)	
$DEFINI	1	246 (1)	246 (1) 257 (1) 258 (1) 260 (1) 261 (1)
			262 (1) 263 (1) 264 (1) 265 (1) 266 (1)
			267 (1) 268 (1) 270 (1) 272 (1) 273 (1)
			274 (1) 275 (1) 279 (1) 280 (1) 281 (1)
			282 (1) 283 (1) 284 (1) 286 (1) 288 (1)
$DPTDEF	2	265 (1)	265 (1)
$DS_BREAK	1	1769 (19)	1769 (19)
$DS_DR780_DEF	1	283 (1)	283 (1)
$DS_DSADEF	5	288 (1)	288 (1)
$DS_DSDEF	3	287 (1)	287 (1)
$DS_DW750_DEF	1	286 (1)	286 (1)
$DS_DW780_DEF	1	282 (1)	282 (1)
$DS_HPODEF	2	280 (1)	280 (1)
$DS_LOAD_DEF	1	285 (1)	285 (1)
$DS_QIODVR_DEF	2	279 (1)	279 (1)
$DS_RH780_DEF	1	281 (1)	281 (1)
$DS_SCBDEF	3	284 (1)	284 (1)
$DS_TYPEDEF	4	276 (1)	276 (1)
$DYNDEF	2	266 (1)	266 (1)
$EQU	1	290 (1)	276 (1) 277 (1) 287 (1)
$EQULS1	1	287 (1)	276 (1) 277 (1) 287 (1)
$EQULST	1		276 (1) 277 (1) 287 (1)
$GBLINI	2		276 (1) 277 (1) 287 (1) 290 (1)
$IDBDEF	1	267 (1)	267 (1)
$IODEF	15	268 (1)	268 (1)
$OFFDEF	1	259 (1)	259 (1) 271 (1) 285 (1)
$OFFSET	2	727 (5)	727 (5)
$OFFST1	1	733 (5)	733 (5)
$PRDEF	1	158 (1)	246 (1) 269 (1)
$PSLDEF	2	270 (1)	270 (1)
$PUSHADR	1	458 (1)	1238 (13) 458 (1) 666 (4) 894 (6) 925 (6)
$QIODEF	1	271 (1)	271 (1)
$SYSGMSGDEF	5	272 (1)	272 (1)
$UBADEF	6	273 (1)	273 (1)
$UCBDEF	10	274 (1)	274 (1)
$VECDEF	2	275 (1)	275 (1)
$VIELD	1		290 (1)
$VIELD1	1	290 (1)	290 (1) 402 (1)
BR_IF_MP	1	439 (1)	439 (1) 460 (1)
BR_IF_NI	1	434 (1)	434 (1) 464 (1) 469 (1)
CASE	1	448 (1)	448 (1) 892 (6)

CMKDEF	1	277	(1)	277	(1)								
DSBINT	1	605	(4)	1053	(9)	1131	(11)	1480	(15)	605	(4)	977	(8)
DSFDEF	3	290	(1)	290	(1)								
ENBINT	1	663	(4)	1013	(8)	1077	(9)	1167	(11)	1513	(15)	663	(4)
ERRSUP_S	1	458	(1)	1238	(13)	458	(1)	666	(4)	894	(6)	925	(6)
MODNAM	1	325	(1)	325	(1)								
SETIPL	1	1597	(16)	1597	(16)	1600	(16)						
_VIELD	1	397	(1)	397	(1)								

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	1429	(15)	1430	(15)	1505	(15)	1681	(18)	507	(1)	747	(6)		
ADDL2	00C0	480	(1)												
ADDL3	00C1	1422	(15)	743	(6)										
ADDW	00A0	1373	(15)												
AOBLSS	00F2	1772	(19)												
ASHL	0078	1677	(18)	505	(1)										
BBC	00E1	1383	(15)	1481	(15)	1546	(16)	1565	(16)	1571	(16)	1572	(16)	1582	(16)
		1766	(19)												
BBS	00E0	1138	(11)	1359	(15)	1504	(15)	434	(1)	464	(1)	469	(1)	984	(8)
BEQL	0013	1319	(15)	1361	(15)	1424	(15)	1458	(15)	1490	(15)	1509	(15)	1630	(17)
		1647	(17)	1711	(19)	1727	(19)	1752	(19)	1771	(19)	437	(1)	439	(1)
		460	(1)	489	(1)	560	(3)	618	(4)	626	(4)	636	(4)	647	(4)
		659	(4)	665	(4)	758	(6)	776	(6)	829	(6)	877	(6)		
BGTR	0014	1410	(15)												
BICL2	00CA	1774	(19)												
BICW	00AA	1500	(15)												
BISB	0088	1343	(15)	1377	(15)	1398	(15)	1440	(15)						
BISB2	0088	1762	(19)												
BISL	00C8	1237	(13)												
BISW	00A8	1472	(15)												
BLBC	00E9	1104	(10)	1110	(10)	1116	(10)	1628	(17)	1675	(18)	950	(7)	956	(7)
		962	(7)												
BLBS	00E8	1050	(9)	1301	(15)	1341	(15)	1375	(15)	1396	(15)	1438	(15)	1473	(15)
		1499	(15)	801	(6)	814	(6)	822	(6)	890	(6)	922	(6)		
BLEQ	0015	1202	(12)												
BLEQU	001B	1783	(19)												
BNEQ	0012	1324	(15)	1330	(15)	1333	(15)	1420	(15)	1461	(15)	1494	(15)	1507	(15)
		1614	(17)	1640	(17)	1729	(19)	1754	(19)	487	(1)	566	(3)	628	(4)
		632	(4)	763	(6)	765	(6)	778	(6)	858	(6)	861	(6)	866	(6)
		874	(6)	906	(6)	916	(6)								
BRB	0011	1119	(10)	1140	(11)	1239	(13)	1501	(15)	1553	(16)	1713	(19)	1778	(19)
		639	(4)	644	(4)	652	(4)	661	(4)	824	(6)	965	(7)	986	(8)
BRW	0031	1051	(9)	1128	(10)	1311	(15)	1331	(15)	1334	(15)	1342	(15)	1365	(15)
		1376	(15)	1397	(15)	1412	(15)	1426	(15)	1439	(15)	1475	(15)	779	(6)
		974	(7)												
BSBB	0010	1552	(16)	1567	(16)	1578	(16)	1586	(16)						
BSBW	0030	1011	(8)	1049	(9)	1071	(9)	1103	(10)	1109	(10)	1115	(10)	1123	(10)
		1125	(10)	1165	(11)	1294	(15)	1300	(15)	1340	(15)	1374	(15)	1395	(15)
		1437	(15)	1483	(15)	1518	(15)	1559	(16)	1588	(16)	638	(4)	643	(4)
		649	(4)	651	(4)	889	(6)	949	(7)	955	(7)	961	(7)	969	(7)
		971	(7)												
CALLG	00FA	1769	(19)	438	(1)	800	(6)	920	(6)						
CALLS	00FB	1238	(13)	1309	(15)	458	(1)	477	(1)	493	(1)	497	(1)	503	(1)
		666	(4)	808	(6)	821	(6)	836	(6)	883	(6)	894	(6)	925	(6)
CASEL	00CF	456	(1)	893	(6)										
CLRB	0094	1293	(15)	660	(4)	918	(6)								
CLRL	00D4	1069	(9)	1310	(15)	1547	(16)	1550	(16)	1576	(16)	1584	(16)	1724	(19)
		1750	(19)	1760	(19)	1761	(19)	459	(1)	474	(1)	490	(1)	495	(1)
		544	(2)	545	(2)	546	(2)	615	(4)	796	(6)	842	(6)	884	(6)
CLRQ	007C	1678	(18)	792	(6)										

```
CLRW            00B4    846     (6)
CMPB            0091    1329    (15)    1360    (15)    1419    (15)    1493    (15)    1613    (17)    1639    (17)    1751    (19)
                        1753    (19)    485     (1)     488     (1)     876     (6)
CMPC5           002D    1323    (15)    565     (3)
CMPL            00D1    1423    (15)    1506    (15)    1728    (19)    1770    (19)    1782    (19)    439     (1)     460     (1)
                        627     (4)     658     (4)     775     (6)     827     (6)     865     (6)
CMPW            00B1    777     (6)
CMPZV           00ED    1201    (12)
CVTBL           0098    1308    (15)    807     (6)     835     (6)     882     (6)
CVTWL           0032    1409    (15)
DECB            0097    1583    (16)    923     (6)
DECL            00D7    1371    (15)
DECW            00B7    1548    (16)
EXTZV           00EF    1203    (12)    446     (1)     888     (6)
INCB            0096    1349    (15)    917     (6)     921     (6)
INCL            00D6    1351    (15)    1354    (15)    1784    (19)
INCW            00B6    1447    (15)
INSQUE          000E    847     (6)
JMP             0017    467     (1)     472     (1)
JSB             0016    1012    (8)     1021    (8)     1070    (9)     1072    (9)     1166    (11)    1175    (11)    1428    (15)
                        1471    (15)    1498    (15)    1510    (15)    1626    (17)    1634    (17)    1651    (17)    1673    (18)
                        1695    (18)    342     (1)     609     (4)     745     (6)     757     (6)     813     (6)     815     (6)
                        823     (6)     841     (6)     873     (6)
MFPR            00DB    1053    (9)     1131    (11)    1480    (15)    1765    (19)    1780    (19)    605     (4)     977     (8)
MNEGW           00AE    1067    (9)
MOVAB           009E    1001    (8)     1075    (9)     1155    (11)    1303    (15)    1321    (15)    1389    (15)    1709    (19)
                        1725    (19)    555     (3)     563     (3)     623     (4)     737     (6)     738     (6)     752     (6)
                        755     (6)     761     (6)     791     (6)     845     (6)     853     (6)     870     (6)     872     (6)
                        913     (6)
MOVAL           00DE    1060    (9)     1063    (9)     1139    (11)    1141    (11)    1142    (11)    1143    (11)    1144    (11)
                        1145    (11)    1147    (11)    1148    (11)    1317    (15)    1344    (15)    1381    (15)    1382    (15)
                        1414    (15)    1444    (15)    1445    (15)    1451    (15)    1452    (15)    1456    (15)    1459    (15)
                        504     (1)     540     (2)     603     (4)     604     (4)     607     (4)     634     (4)     655     (4)
                        739     (6)     985     (8)     987     (8)     988     (8)     989     (8)     990     (8)     991     (8)
                        993     (8)     994     (8)
MOVAQ           007E    772     (6)
MOVB            0090    1003    (8)     1056    (9)     1146    (11)    1151    (11)    1157    (11)    1348    (15)    1380    (15)
                        1401    (15)    1443    (15)    1454    (15)    481     (1)     748     (6)     750     (6)     900     (6)
                        909     (6)     914     (6)     992     (8)     997     (8)
MOVC            0028    1352    (15)    1355    (15)
MOVC3           0028    1133    (11)    979     (8)
MOVL            00D0    1000    (8)     1004    (8)     1007    (8)     1008    (8)     1009    (8)     1010    (8)     1014    (8)
                        1046    (9)     1054    (9)     1058    (9)     1061    (9)     1062    (9)     1064    (9)     1065    (9)
                        1066    (9)     1073    (9)     1078    (9)     1100    (10)    1105    (10)    1111    (10)    1117    (10)
                        1122    (10)    1124    (10)    1153    (11)    1154    (11)    1158    (11)    1161    (11)    1162    (11)
                        1163    (11)    1164    (11)    1168    (11)    1236    (13)    1298    (15)    1315    (15)    1320    (15)
                        1326    (15)    1327    (15)    1328    (15)    1332    (15)    1345    (15)    1346    (15)    1362    (15)
                        1363    (15)    1364    (15)    1378    (15)    1384    (15)    1385    (15)    1386    (15)    1387    (15)
                        1399    (15)    1402    (15)    1403    (15)    1411    (15)    1415    (15)    1417    (15)    1418    (15)
                        1421    (15)    1425    (15)    1441    (15)    1446    (15)    1449    (15)    1450    (15)    1457    (15)
                        1460    (15)    1462    (15)    1463    (15)    1469    (15)    1470    (15)    1474    (15)    1482    (15)
                        1486    (15)    1487    (15)    1488    (15)    1489    (15)    1491    (15)    1492    (15)    1495    (15)
                        1496    (15)    1497    (15)    1502    (15)    1503    (15)    1508    (15)    1511    (15)    1520    (15)
                        1551    (16)    1566    (16)    1573    (16)    1574    (16)    1577    (16)    1585    (16)    1616    (17)
                        1617    (17)    1618    (17)    1619    (17)    1620    (17)    1621    (17)    1623    (17)    1624    (17)
                        1625    (17)    1629    (17)    1642    (17)    1643    (17)    1644    (17)    1645    (17)    1646    (17)
                        1710    (19)    1712    (19)    1714    (19)    1726    (19)    1730    (19)    1777    (19)    541     (2)
```

```
                    542  (2)   556  (3)   558  (3)   567  (3)   608  (4)   610  (4)   612  (4)
                    617  (4)   620  (4)   621  (4)   631  (4)   635  (4)   637  (4)   641  (4)
                    642  (4)   646  (4)   648  (4)   650  (4)   656  (4)   657  (4)   664  (4)
                    741  (6)   746  (6)   754  (6)   764  (6)   766  (6)   773  (6)   774  (6)
                    787  (6)   789  (6)   790  (6)   812  (6)   820  (6)   840  (6)   857  (6)
                    859  (6)   860  (6)   862  (6)   863  (6)   867  (6)   868  (6)   887  (6)
                    896  (6)   897  (6)   902  (6)   946  (7)   951  (7)   957  (7)   963  (7)
                    968  (7)   970  (7)   999  (8)
MOVPSL      00DC    1200 (12)
MOVQ        007D    496  (1)   562  (3)
MOVW        00B0    1005 (8)   1006 (8)   1055 (9)   1057 (9)   1059 (9)   1106 (10)  1112 (10)
                    1118 (10)  1149 (11)  1159 (11)  1160 (11)  1347 (15)  1379 (15)  1400 (15)
                    1442 (15)  506  (1)   508  (1)   509  (1)   788  (6)   826  (6)   898  (6)
                    904  (6)   907  (6)   911  (6)   952  (7)   958  (7)   964  (7)   995  (8)
MOVZBL      009A    1102 (10)  1108 (10)  1114 (10)  1299 (15)  1304 (15)  1307 (15)  1316 (15)
                    1322 (15)  1339 (15)  1350 (15)  1353 (15)  1370 (15)  1388 (15)  1394 (15)
                    1413 (15)  1448 (15)  1549 (16)  1575 (16)  564  (3)   756  (6)   760  (6)
                    806  (6)   834  (6)   871  (6)   875  (6)   881  (6)   948  (7)   954  (7)
                    960  (7)
MOVZBW      009B    1152 (11)  1404 (15)  1453 (15)  998  (8)
MOVZWL      003C    1047 (9)   1074 (9)   1127 (10)  1436 (15)  1693 (18)  762  (6)   818  (6)
                    819  (6)   973  (7)
MTPR        00DA    1013 (8)   1053 (9)   1077 (9)   1131 (11)  1167 (11)  1480 (15)  1513 (15)
                    1597 (16)  1600 (16)  1767 (19)  605  (4)   663  (4)   977  (8)
MULW        00A4    1372 (15)
POPL        8ED0    1590 (16)  926  (6)
POPR        00BA    1016 (8)   1080 (9)   1134 (11)  1170 (11)  1627 (17)  1636 (17)  1653 (17)
                    1674 (18)  1680 (18)  1696 (18)  1776 (19)  568  (3)   809  (6)   980  (8)
PUSHAB      009F    1306 (15)  492  (1)   803  (6)   804  (6)   805  (6)   833  (6)   878  (6)
                    879  (6)   880  (6)   891  (6)
PUSHAL      00DF    1238 (13)  458  (1)   666  (4)   794  (6)   894  (6)   925  (6)
PUSHAQ      007F    797  (6)   798  (6)   832  (6)
PUSHL       00DD    1238 (13)  1542 (16)  458  (1)   475  (1)   476  (1)   491  (1)   499  (1)
                    502  (1)   666  (4)   795  (6)   799  (6)   830  (6)   831  (6)   894  (6)
                    924  (6)   925  (6)
PUSHR       00BB    1020 (8)   1045 (9)   1099 (10)  1132 (11)  1174 (11)  1305 (15)  1615 (17)
                    1622 (17)  1641 (17)  1672 (18)  1676 (18)  1692 (18)  1759 (19)  341  (1)
                    345  (1)   554  (3)   624  (4)   802  (6)   945  (7)   978  (8)
RET         0004    1521 (15)  1591 (16)  441  (1)   462  (1)   483  (1)   510  (1)   548  (2)
                    668  (4)   767  (6)   810  (6)   816  (6)   843  (6)   885  (6)   927  (6)
RSB         0005    1017 (8)   1081 (9)   1171 (11)  1204 (12)  1560 (16)  1598 (16)  1601 (16)
                    1637 (17)  1655 (17)  1682 (18)  1697 (18)  1715 (19)  1731 (19)  1785 (19)
                    440  (1)   461  (1)   569  (3)
SOBGEQ      00F4    1068 (9)
SOBGTR      00F5    1076 (9)   1431 (15)  1679 (18)  749  (6)
SUBL        00C2    1694 (18)
SUBL3       00C3    479  (1)   559  (3)   744  (6)
TSTL        00D5    1318 (15)  625  (4)   629  (4)
TSTW        00B5    436  (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...
.ALIGN	333 (1)
.ASCIC	325 (1) 331 (1) 348 (1) 351 (1) 354 (1) 381 (1) 382 (1) 383 (1)
	384 (1) 385 (1) 386 (1) 387 (1)
.ASCII	358 (1)
.BLKB	396 (1) 733 (5)
.BLKL	404 (1) 407 (1)
.BYTE	363 (1) 364 (1) 365 (1) 366 (1) 367 (1) 368 (1) 369 (1) 393 (1)
.DSABL	14 (1)
.END	1786 (19)
.ENDC	1013 (8) 1053 (9) 1077 (9) 1131 (11) 1167 (11) 1238 (13) 1480 (15) 1513 (15)
	1597 (16) 1600 (16) 276 (1) 277 (1) 287 (1) 290 (1) 402 (1) 458 (1)
	605 (4) 663 (4) 666 (4) 733 (5) 894 (6) 925 (6) 977 (8)
.ENDM	1597 (16) 1769 (19) 245 (1) 246 (1) 257 (1) 258 (1) 259 (1) 260 (1)
	261 (1) 262 (1) 263 (1) 264 (1) 265 (1) 266 (1) 267 (1) 268 (1)
	270 (1) 271 (1) 272 (1) 273 (1) 274 (1) 275 (1) 276 (1) 277 (1)
	279 (1) 280 (1) 281 (1) 282 (1) 283 (1) 284 (1) 285 (1) 286 (1)
	287 (1) 288 (1) 290 (1) 325 (1) 397 (1) 434 (1) 439 (1) 448 (1)
	458 (1) 605 (4) 663 (4) 727 (5) 733 (5)
.ENDR	276 (1) 277 (1) 287 (1) 290 (1) 402 (1) 456 (1) 733 (5) 893 (6)
.ENTRY	432 (1) 538 (2) 601 (4) 735 (6)
.EXTRN	296 (1) 297 (1) 298 (1) 299 (1) 300 (1) 301 (1) 302 (1) 303 (1)
	304 (1) 305 (1) 306 (1) 307 (1) 308 (1) 309 (1) 310 (1) 311 (1)
	312 (1) 313 (1)
.IDENT	12 (1)
.IF	1013 (8) 1053 (9) 1077 (9) 1131 (11) 1167 (11) 1238 (13) 1480 (15) 1513 (15)
	1597 (16) 1600 (16) 276 (1) 277 (1) 287 (1) 290 (1) 402 (1) 458 (1)
	605 (4) 663 (4) 666 (4) 733 (5) 894 (6) 925 (6) 977 (8)
.IFF	1013 (8) 1053 (9) 1077 (9) 1131 (11) 1167 (11) 1238 (13) 1480 (15) 1513 (15)
	1597 (16) 1600 (16) 276 (1) 277 (1) 287 (1) 290 (1) 458 (1) 605 (4)
	663 (4) 666 (4) 733 (5) 894 (6) 925 (6) 977 (8)
.IIF	1238 (13) 276 (1) 277 (1) 287 (1) 290 (1) 402 (1) 458 (1) 666 (4)
	894 (6) 925 (6)
.IRP	259 (1) 271 (1) 276 (1) 277 (1) 285 (1) 287 (1) 290 (1) 402 (1)
	456 (1) 733 (5) 893 (6)
.LIBRARY	151 (1) 152 (1) 153 (1)
.LIST	259 (1) 271 (1) 284 (1) 285 (1) 288 (1) 292 (1) 733 (5)
.LONG	1022 (8) 1023 (8) 1176 (11) 1177 (11) 328 (1) 391 (1)
.MACRO	158 (1) 276 (1) 277 (1) 287 (1) 290 (1)
.MDELETE	277 (1) 287 (1)
.NLIST	256 (1) 259 (1) 271 (1) 284 (1) 285 (1) 288 (1) 733 (5)
.NOCROSS	246 (1) 257 (1) 258 (1) 260 (1) 261 (1) 262 (1) 263 (1) 264 (1)
	265 (1) 266 (1) 267 (1) 268 (1) 270 (1) 272 (1) 273 (1) 274 (1)
	275 (1) 279 (1) 280 (1) 281 (1) 282 (1) 283 (1) 284 (1) 286 (1)
	288 (1)
.NOSHOW	13 (1)
.PAGE	145 (1) 1732 (19) 1748 (19) 293 (1) 322 (1) 388 (1) 408 (1) 614 (4)
	780 (6) 848 (6)
.PSECT	323 (1) 389 (1) 410 (1) 733 (5)
.RESTORE	246 (1) 257 (1) 258 (1) 260 (1) 261 (1) 262 (1) 263 (1) 264 (1)
	265 (1) 266 (1) 267 (1) 268 (1) 270 (1) 272 (1) 273 (1) 274 (1)

```
                275     (1)     279     (1)     280     (1)     281     (1)     282     (1)     283     (1)     284     (1)     286     (1)
                288     (1)     733     (5)
.SAVE           246     (1)     257     (1)     258     (1)     260     (1)     261     (1)     262     (1)     263     (1)     264     (1)
                265     (1)     266     (1)     267     (1)     268     (1)     270     (1)     272     (1)     273     (1)     274     (1)
                275     (1)     279     (1)     280     (1)     281     (1)     282     (1)     283     (1)     284     (1)     286     (1)
                288     (1)     733     (5)
.SBTTL          1026    (9)     1083    (10)    1180    (12)    1206    (13)    1241    (14)    146     (1)     1523    (16)    1595    (16)
                1603    (17)    1657    (18)    1699    (19)    1733    (19)    409     (1)     514     (2)     571     (4)     670     (5)
                929     (7)
.SIGNED_WORD    456     (1)     893     (6)
.SUBTITLE       294     (1)
.TITLE          11      (1)
.WEAK           318     (1)     319     (1)     320     (1)     321     (1)
.WORD           1291    (15)    335     (1)     338     (1)     372     (1)     373     (1)     374     (1)     375     (1)     376     (1)
                377     (1)     378     (1)     379     (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	48	#-1298 (15), 1303 (15), #-1315 (15), #-1329 (15), #-1350 (15), 1352 (15)
		#-1353 (15), 1355 (15), #-1362 (15), #-1370 (15), #-1384 (15), #-1388 (15)
		#-1402 (15), #-1403 (15), #-1409 (15), #-1413 (15), #-1417 (15), #-1421 (15)
		#-1448 (15), #-1453 (15), #-1454 (15), #-1495 (15), #-1549 (16), #-1575 (16)
		#-436 (1), 438 (1), 447 (1), #-475 (1), #-476 (1), #-479 (1)
		#-480 (1), #-486 (1), #-496 (1), #-499 (1), #-502 (1), 504 (1)
		#-741 (6), #-743 (6), #-744 (6), #-754 (6), 803 (6), #-857 (6)
		#-859 (6), #-863 (6), #-868 (6), 878 (6), #-909 (6)
FP	15	737 (6), 738 (6), #-787 (6), #-788 (6), #-789 (6), #-790 (6)
		791 (6), #-792 (6), 794 (6), 797 (6), 798 (6), 804 (6)
		#-812 (6), 832 (6)
R0	182	#-1014 (8), #-1050 (9), #-1060 (9), #-1061 (9), #-1062 (9), #-1063 (9)
		#-1064 (9), #-1065 (9), #-1066 (9), #-1067 (9), #-1068 (9), #-1073 (9)
		#-1075 (9), #-1078 (9), #-1104 (10), #-1110 (10), #-1116 (10), #-1122 (10)
		#-1124 (10), #-1127 (10), #-1139 (11), #-1141 (11), #-1142 (11), #-1143 (11)
		#-1144 (11), #-1145 (11), #-1168 (11), #-1201 (12), #-1203 (12), #-1236 (13)
		1238 (13), #-1301 (15), #-1304 (15), #-1305 (15), #-1310 (15), #-1322 (15)
		#-1323 (15), #-1326 (15), #-1327 (15), #-1329 (15), #-1332 (15), #-1341 (15)
		#-1350 (15), #-1351 (15), #-1352 (15), #-1353 (15), #-1354 (15), #-1355 (15)
		#-1362 (15), #-1363 (15), #-1375 (15), #-1384 (15), #-1385 (15), #-1396 (15)
		#-1411 (15), #-1421 (15), #-1422 (15), #-1423 (15), #-1425 (15), #-1438 (15)
		#-1448 (15), #-1449 (15), #-1457 (15), 1459 (15), #-1460 (15), #-1473 (15)
		#-1474 (15), #-1489 (15), #-1491 (15), #-1492 (15), #-1497 (15), #-1499 (15)
		#-1502 (15), #-1503 (15), #-1505 (15), #-1520 (15), #-1542 (16), #-1549 (16)
		#-1550 (16), #-1551 (16), #-1566 (16), #-1575 (16), #-1576 (16), #-1577 (16)
		#-1585 (16), #-1590 (16), #-1616 (17), #-1617 (17), #-1620 (17), #-1621 (17)
		#-1625 (17), #-1628 (17), #-1629 (17), 1634 (17), #-1642 (17), #-1643 (17)
		#-1646 (17), 1651 (17), #-1675 (18), #-1693 (18), #-1709 (19), #-1710 (19)
		#-1712 (19), #-1714 (19), #-1724 (19), #-1730 (19), #-1750 (19), #-1759 (19)
		#-1765 (19), 1766 (19), #-1776 (19), #-1784 (19), #-345 (1), #-447 (1)
		#-456 (1), #-459 (1), #-506 (1), #-559 (3), #-564 (3), #-565 (3)
		#-567 (3), #-608 (4), #-620 (4), #-621 (4), #-637 (4), #-641 (4)
		#-648 (4), #-650 (4), #-656 (4), #-657 (4), #-658 (4), #-660 (4)
		#-739 (6), #-746 (6), #-747 (6), #-749 (6), #-756 (6), #-760 (6)
		761 (6), #-762 (6), #-766 (6), #-772 (6), #-773 (6), #-775 (6)
		#-801 (6), #-802 (6), #-809 (6), #-814 (6), #-822 (6), #-840 (6)
		#-842 (6), #-871 (6), #-875 (6), #-884 (6), #-890 (6), #-913 (6)
		#-914 (6), #-922 (6), #-924 (6), #-926 (6), #-950 (7), #-956 (7)
		#-962 (7), #-968 (7), #-970 (7), #-973 (7), #-985 (8), #-987 (8)
		#-988 (8), #-989 (8), #-990 (8), #-991 (8)
R1	102	#-1047 (9), #-1055 (9), #-1074 (9), #-1076 (9), #-1102 (10), #-1106 (10)
		#-1108 (10), #-1112 (10), #-1114 (10), #-1118 (10), #-1200 (12), 1201 (12)
		1203 (12), #-1303 (15), #-1304 (15), #-1305 (15), #-1321 (15), #-1322 (15)
		1323 (15), #-1339 (15), #-1347 (15), #-1370 (15), #-1371 (15), #-1372 (15)
		#-1373 (15), #-1379 (15), #-1385 (15), #-1386 (15), #-1387 (15), #-1394 (15)
		#-1400 (15), #-1436 (15), #-1442 (15), #-1456 (15), #-1457 (15), #-1459 (15)
		#-1460 (15), #-1462 (15), #-1463 (15), #-1503 (15), 1504 (15), #-1505 (15)
		#-1506 (15), #-1508 (15), 1510 (15), #-1573 (16), #-1574 (16), #-1576 (16)
		#-1676 (18), #-1677 (18), #-1679 (18), #-1680 (18), #-1681 (18), #-1692 (18)
		#-1693 (18), #-1694 (18), #-1696 (18), #-1710 (19), #-1712 (19), #-1759 (19)

		#-1776	(19)	#-345	(1)	#-504	(1)	#-505	(1)	#-506	(1)	#-507	(1)
		#-508	(1)	#-509	(1)	#-540	(2)	#-541	(2)	#-542	(2)	#-563	(3)
		#-564	(3)	565	(3)	#-623	(4)	#-625	(4)	#-627	(4)	#-655	(4)
		#-656	(4)	#-658	(4)	#-747	(6)	#-748	(6)	#-755	(6)	#-756	(6)
		#-760	(6)	761	(6)	#-812	(6)	#-818	(6)	#-819	(6)	#-870	(6)
		#-871	(6)	#-948	(7)	#-952	(7)	#-954	(7)	#-958	(7)	#-960	(7)
		#-964	(7)										
R10	75	#-1000	(8)	#-1001	(8)	#-1007	(8)	#-1008	(8)	#-1016	(8)	#-1099	(10)
		#-1105	(10)	#-1106	(10)	#-1124	(10)	1133	(11)	1142	(11)	1143	(11)
		1144	(11)	1145	(11)	#-1146	(11)	1147	(11)	1148	(11)	#-1149	(11)
		#-1154	(11)	#-1155	(11)	#-1161	(11)	#-1162	(11)	#-1170	(11)	1291	(15)
		#-1317	(15)	#-1318	(15)	#-1320	(15)	1321	(15)	#-1326	(15)	1344	(15)
		#-1345	(15)	#-1346	(15)	#-1347	(15)	#-1348	(15)	1352	(15)	1355	(15)
		#-1450	(15)	1456	(15)	#-1482	(15)	#-1502	(15)	#-1511	(15)	#-1585	(16)
		#-1615	(17)	#-1636	(17)	#-1641	(17)	#-1653	(17)	432	(1)	735	(6)
		#-860	(6)	#-862	(6)	#-865	(6)	#-867	(6)	870	(6)	879	(6)
		#-887	(6)	#-898	(6)	#-907	(6)	#-945	(7)	#-951	(7)	#-952	(7)
		#-970	(7)	979	(8)	988	(8)	989	(8)	990	(8)	991	(8)
		#-992	(8)	993	(8)	994	(8)	#-995	(8)				
R11	35	1291	(15)	#-1349	(15)	1359	(15)	#-1360	(15)	1383	(15)	#-1419	(15)
		#-1436	(15)	#-1469	(15)	#-1493	(15)	1572	(16)	#-1583	(16)	#-1613	(17)
		#-1615	(17)	#-1636	(17)	#-1639	(17)	#-1641	(17)	#-1653	(17)	432	(1)
		#-555	(3)	#-556	(3)	#-558	(3)	#-559	(3)	563	(3)	735	(6)
		#-857	(6)	#-859	(6)	#-860	(6)	#-862	(6)	#-863	(6)	#-867	(6)
		#-868	(6)	#-900	(6)	#-902	(6)	#-904	(6)				
R2	68	#-1016	(8)	#-1020	(8)	#-1045	(9)	#-1054	(9)	#-1080	(9)	#-1099	(10)
		#-1105	(10)	#-1111	(10)	#-1117	(10)	#-1132	(11)	#-1134	(11)	#-1170	(11)
		#-1174	(11)	1291	(15)	#-1299	(15)	#-1345	(15)	#-1346	(15)	#-1378	(15)
		#-1387	(15)	#-1389	(15)	#-1399	(15)	#-1441	(15)	#-1615	(17)	#-1636	(17)
		#-1641	(17)	#-1653	(17)	#-1676	(18)	#-1678	(18)	#-1680	(18)	#-1692	(18)
		#-1696	(18)	#-1759	(19)	#-1760	(19)	#-1772	(19)	#-1776	(19)	#-341	(1)
		#-345	(1)	432	(1)	#-479	(1)	#-480	(1)	#-481	(1)	#-554	(3)
		#-562	(3)	#-565	(3)	#-568	(3)	601	(4)	#-664	(4)	735	(6)
		#-746	(6)	#-750	(6)	#-762	(6)	#-777	(6)	#-788	(6)	#-820	(6)
		#-826	(6)	#-828	(6)	#-831	(6)	#-840	(6)	845	(6)	#-875	(6)
		#-876	(6)	#-893	(6)	#-945	(7)	#-951	(7)	#-957	(7)	#-963	(7)
		#-978	(8)	#-980	(8)								
R3	43	#-1016	(8)	#-1020	(8)	#-1045	(9)	#-1046	(9)	#-1080	(9)	#-1099	(10)
		#-1100	(10)	#-1132	(11)	#-1134	(11)	#-1170	(11)	#-1174	(11)	1291	(15)
		#-1298	(15)	#-1299	(15)	#-1388	(15)	#-1389	(15)	#-1488	(15)	#-1615	(17)
		#-1636	(17)	#-1641	(17)	#-1653	(17)	#-1672	(18)	#-1674	(18)	#-1692	(18)
		#-1696	(18)	#-341	(1)	#-345	(1)	432	(1)	#-554	(3)	565	(3)
		#-568	(3)	601	(4)	#-607	(4)	#-608	(4)	735	(6)	#-818	(6)
		#-826	(6)	#-872	(6)	#-888	(6)	#-945	(7)	#-946	(7)	#-978	(8)
		#-980	(8)										
R4	53	#-1004	(8)	#-1016	(8)	#-1020	(8)	#-1045	(9)	#-1058	(9)	#-1080	(9)
		#-1099	(10)	#-1132	(11)	#-1134	(11)	1138	(11)	#-1153	(11)	#-1158	(11)
		#-1170	(11)	#-1174	(11)	1291	(15)	#-1316	(15)	#-1323	(15)	#-1414	(15)
		#-1415	(15)	#-1417	(15)	#-1418	(15)	#-1430	(15)	#-1469	(15)	#-1487	(15)
		#-1488	(15)	#-1492	(15)	#-1615	(17)	#-1619	(17)	#-1625	(17)	#-1636	(17)
		#-1641	(17)	#-1645	(17)	#-1653	(17)	#-1728	(19)	#-341	(1)	#-345	(1)
		432	(1)	#-554	(3)	#-568	(3)	601	(4)	#-604	(4)	#-610	(4)
		#-612	(4)	735	(6)	#-744	(6)	#-887	(6)	888	(6)	#-897	(6)
		#-945	(7)	#-978	(8)	#-980	(8)	984	(8)	#-999	(8)		
R5	66	#-1010	(8)	#-1016	(8)	#-1020	(8)	#-1045	(9)	#-1080	(9)	#-1099	(10)
		#-1132	(11)	#-1134	(11)	#-1164	(11)	#-1170	(11)	#-1174	(11)	1291	(15)
		#-1315	(15)	#-1316	(15)	1323	(15)	#-1413	(15)	#-1431	(15)	#-1470	(15)

		#-1472	(15)	#-1486	(15)	#-1615	(17)	#-1618	(17)	#-1619	(17)	#-1620	(17)
		#-1623	(17)	#-1636	(17)	#-1641	(17)	#-1644	(17)	#-1645	(17)	#-1653	(17)
		#-341	(1)	#-345	(1)	#-554	(3)	#-568	(3)	601	(4)	#-603	(4)
		#-611	(4)	#-613	(4)	#-621	(4)	#-624	(4)	#-627	(4)	#-631	(4)
		#-634	(4)	#-635	(4)	#-637	(4)	735	(6)	#-743	(6)	#-773	(6)
		#-774	(6)	#-775	(6)	#-777	(6)	#-845	(6)	#-846	(6)	847	(6)
		853	(6)	#-876	(6)	#-911	(6)	913	(6)	#-921	(6)	#-923	(6)
		#-945	(7)	#-978	(8)	#-980	(8)						
R6	75	#-1003	(8)	#-1004	(8)	#-1005	(8)	#-1006	(8)	#-1007	(8)	#-1008	(8)
		#-1009	(8)	#-1054	(9)	#-1055	(9)	#-1056	(9)	#-1057	(9)	#-1058	(9)
		#-1059	(9)	1060	(9)	1063	(9)	#-1067	(9)	#-1069	(9)	#-1073	(9)
		#-1117	(10)	#-1118	(10)	#-1157	(11)	#-1158	(11)	#-1159	(11)	#-1160	(11)
		#-1161	(11)	#-1162	(11)	#-1163	(11)	1291	(15)	#-1409	(15)	#-1422	(15)
		#-1429	(15)	#-1482	(15)	#-1495	(15)	#-1496	(15)	#-1497	(15)	#-1511	(15)
		#-1615	(17)	#-1616	(17)	#-1622	(17)	#-1623	(17)	#-1624	(17)	#-1627	(17)
		#-1636	(17)	#-1641	(17)	#-1642	(17)	#-1653	(17)	#-1714	(19)	#-1725	(19)
		#-1726	(19)	#-1728	(19)	432	(1)	#-554	(3)	#-556	(3)	#-559	(3)
		#-568	(3)	601	(4)	#-617	(4)	#-620	(4)	634	(4)	#-641	(4)
		#-642	(4)	#-646	(4)	#-648	(4)	#-650	(4)	735	(6)	#-754	(6)
		755	(6)	#-764	(6)	#-896	(6)	#-963	(7)	#-964	(7)		
R7	60	#-1005	(8)	#-1016	(8)	#-1045	(9)	#-1046	(9)	#-1059	(9)	#-1080	(9)
		#-1099	(10)	#-1100	(10)	1139	(11)	1141	(11)	#-1159	(11)	#-1170	(11)
		1291	(15)	#-1441	(15)	#-1442	(15)	#-1443	(15)	1444	(15)	1445	(15)
		#-1446	(15)	#-1449	(15)	#-1450	(15)	1451	(15)	1452	(15)	#-1453	(15)
		#-1454	(15)	#-1463	(15)	#-1470	(15)	#-1486	(15)	#-1500	(15)	#-1551	(16)
		#-1615	(17)	#-1636	(17)	#-1641	(17)	#-1653	(17)	432	(1)	735	(6)
		#-738	(6)	#-752	(6)	#-854	(6)	#-896	(6)	#-897	(6)	#-899	(6)
		#-901	(6)	#-903	(6)	#-905	(6)	#-908	(6)	#-910	(6)	#-912	(6)
		#-915	(6)	#-917	(6)	#-918	(6)	920	(6)	#-945	(7)	#-946	(7)
		985	(8)	987	(8)								
R8	30	#-1016	(8)	#-1099	(10)	#-1170	(11)	1291	(15)	#-1328	(15)	#-1364	(15)
		#-1399	(15)	#-1400	(15)	#-1401	(15)	#-1402	(15)	#-1403	(15)	#-1404	(15)
		#-1418	(15)	#-1449	(15)	#-1487	(15)	#-1550	(16)	#-1566	(16)	#-1615	(17)
		#-1617	(17)	#-1618	(17)	#-1629	(17)	#-1636	(17)	#-1641	(17)	#-1643	(17)
		#-1644	(17)	#-1646	(17)	#-1653	(17)	432	(1)	735	(6)	#-945	(7)
R9	50	#-1000	(8)	#-1009	(8)	#-1010	(8)	#-1016	(8)	#-1099	(10)	#-1111	(10)
		#-1112	(10)	#-1122	(10)	#-1151	(11)	#-1152	(11)	#-1153	(11)	#-1154	(11)
		#-1163	(11)	#-1164	(11)	#-1170	(11)	1291	(15)	#-1327	(15)	#-1328	(15)
		#-1363	(15)	#-1364	(15)	#-1378	(15)	#-1379	(15)	#-1380	(15)	1381	(15)
		1382	(15)	#-1386	(15)	1389	(15)	1414	(15)	#-1446	(15)	#-1447	(15)
		#-1489	(15)	#-1508	(15)	#-1548	(16)	#-1573	(16)	#-1577	(16)	#-1615	(17)
		#-1636	(17)	#-1641	(17)	#-1653	(17)	432	(1)	735	(6)	#-945	(7)
		#-957	(7)	#-958	(7)	#-968	(7)	#-997	(8)	#-998	(8)	#-999	(8)
SP	41	#-1013	(8)	#-1053	(9)	#-1077	(9)	#-1131	(11)	#-1167	(11)	#-1236	(13)
		#-1307	(15)	#-1308	(15)	#-1480	(15)	#-1513	(15)	1769	(19)	#-1777	(19)
		#-1780	(19)	#-1782	(19)	#-474	(1)	#-490	(1)	#-495	(1)	#-496	(1)
		#-562	(3)	#-605	(4)	#-615	(4)	623	(4)	#-629	(4)	#-646	(4)
		#-663	(4)	#-737	(6)	#-748	(6)	#-750	(6)	752	(6)	#-796	(6)
		800	(6)	#-806	(6)	#-807	(6)	#-819	(6)	#-820	(6)	#-821	(6)
		#-834	(6)	#-835	(6)	#-881	(6)	#-882	(6)	#-977	(8)		

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.24
Command processing	882	00:00:00.19	00:00:00.64
Pass 1	1740	00:00:06.49	00:00:09.93
Symbol table sort	8	00:00:00.50	00:00:00.49
Pass 2	622	00:00:01.72	00:00:03.65
Symbol table output	2	00:00:00.11	00:00:00.35
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	69	00:00:00.79	00:00:01.29
Assembler run totals	3348	00:00:09.83	00:00:16.60

The working set limit was 3400 pages.
348441 bytes (681 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1693 non-local and 119 local symbols.
1786 source lines were read in Pass 1, producing 126 object records in Pass 2.
225 pages of virtual memory were used to define 60 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	25
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	6
SYS$COMMON:[SYSLIB]LIB.MLB;2	5
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	50

2035 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:QIO.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DIA
```

Table of contents

(1) 93 ONE PARAMETER FUNCTION PROCESSING
(1) 126 ZERO PARAMETER FUNCTION PROCESSING
(1) 159 READ AND WRITE FUNCTION PROCESSING
(1) 212 READ AND WRITE FUNCTION BUFFER CHECK AND LOCK ROUTINES
(1) 255 READ AND WRITE BUFFER CHECK AND LOCK AND RETURN ROUTINES
(1) 356 CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
(1) 392 CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
(1) 431 CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
(1) 478 CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
(1) 531 SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
(1) 571 SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS
(1) 610 SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
(1) 648 CARRIAGE CONTROL INTERPRETATION

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element QIOFDT.MAR
                        0000     2 ; *1     6-FEB-1986 15:22:10 DS ""
                        0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element QIOFDT.MAR
                        0000     4         .TITLE  QIOFDT  *** QIOFDT function dispatch routines
                        0000     5         .IDENT  /05-03/
                    00000000     6         .PSECT  SEP, EXE, SHR, WRT, LONG
                        0000     7
                    00000014    0000     8 EXE$C_CMSTKSZ   = ^X14          ; Utterly unused Change mode call size
                        0000     9 ;
                        0000    10 ;*****************************************************************************
                        0000    11 ;*                                                                           *
                        0000    12 ; COPYRIGHT (C) 1977, 1980
                        0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000    14 ;*                                                                           *
                        0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        0000    20 ;*  TRANSFERRED.                                                            *
                        0000    21 ;*                                                                           *
                        0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000    24 ;*  CORPORATION.                                                            *
                        0000    25 ;*                                                                           *
                        0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000    28 ;*                                                                           *
                        0000    29 ;*****************************************************************************
                        0000    30 ;
                        0000    31 ; D. N. CUTLER 15-SEP-76
                        0000    32 ;
                        0000    33 ; MODIFIED BY:
                        0000    34 ;
                        0000    35 ;       DS5.4   John Ciukaj     5-May-1980      Version 5.4
                        0000    36 ;                 Used Vax/Vms version 2.0 SYSQIOFDT.MAR and created
                        0000    37 ;                 a UPD file with the edits necessary to reflect the
                        0000    38 ;                 changes exhibited in Supervisor version 5.3 module
                        0000    39 ;                 QIOFDT.
                        0000    40 ;
                        0000    41 ;                Dave Butenhof   30-jun-1980,    version 5.5
                        0000    42 ;       02         Alter two BSBW's to longword displacment JSB's since
                        0000    43 ;                  the supervisor's gotten fatter.
                        0000    44 ;
                        0000    45 ;       03       M. Baggett      8-Dec-1982      Version 6.10
                        0000    46 ;                Change BSBW to JSB to fixed truncation error.
                        0000    47 ;
                        0000    48 ;
                        0000    49 ;       SRB0003          Steve Beckhardt         23-Oct-1979
                        0000    50 ;       Fixed bug to deallocate diagnostic buffer if QIO is being
                        0000    51 ;       backed out to fault a page in.
                        0000    52 ;
                        0000    53 ;       MHB0029          M. H. Bramhall          28-Sep-1979
                        0000    54 ;       Added EXE$CARRIAGE for carriage control processing.
                        0000    55 ;
                        0000    56 ;       SRB0002          STEVE BECKHARDT         30-MAR-1979
                        0000    57 ;       ADDED ENTRY POINTS EXE$MODIFY, EXE$MODIFYLOCK, AND EXE$MODIFYLOCKR
```

```
                        0000     58 ;       FOR FUNCTIONS THAT READ AND WRITE MEMORY.
                        0000     59 ;
                        0000     60 ;       SRB0001          STEVE BECKHARDT         29-MAR-1979
                        0000     61 ;       ADDED ROUTINES EXE$READLOCKR, EXE$WRITELOCKR, EXE$READCHKR, AND
                        0000     62 ;       EXE$WRITECHKR
                        0000     63 ;
                        0000     64 ; SYSTEM SERVICE QUEUE I/O FUNCTION DECISION TABLE SUBROUTINES
                        0000     65 ;
                        0000     66 ; MACRO LIBRARY CALLS
                        0000     67 ;
                        0000     68
                        0000     69          $ACBDEF                             ;DEFINE ACB OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .IIF    NB,ACB$L_ASTQFL,        ACB$L_ASTQFL:
                00000004 0000                       .BLKL   1
                        0004                .IIF    NB,ACB$L_ASTQBL,        ACB$L_ASTQBL:
                00000008 0004                       .BLKL   1
                        0008                .IIF    NB,ACB$W_SIZE,  ACB$W_SIZE:
                0000000A 0008                       .BLKW   1
                        000A                .IIF    NB,ACB$B_TYPE,  ACB$B_TYPE:
                0000000B 000A                       .BLKB   1
                        000B                .IIF    NB,ACB$B_RMOD,  ACB$B_RMOD:
                0000000C 000B                       .BLKB   1
                        000C                .IIF    NB,ACB$L_PID,   ACB$L_PID:
                00000010 000C                       .BLKL   1
                        0010                .IIF    NB,ACB$L_AST,   ACB$L_AST:
                00000014 0010                       .BLKL   1
                        0014                .IIF    NB,ACB$L_ASTPRM,        ACB$L_ASTPRM:
                00000018 0014                       .BLKL   1
                        0018                .IIF    NB,ACB$C_LENGTH,        ACB$C_LENGTH:
                        0018                .IIF    NB,ACB$K_LENGTH,        ACB$K_LENGTH:
                        0018                .IIF    NB,ACB$L_KAST,  ACB$L_KAST:
                0000001C 0018                       .BLKL   1
                00000000                    .RESTORE
                        0000     70         $CCBDEF                             ;DEFINE CCB OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 001C               .=0
                        0000                .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
                00000004 0000                       .BLKL   1
                        0004                .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
                00000008 0004                       .BLKL   1
                        0008                .IIF    NB,CCB$B_STS,   CCB$B_STS:
                00000009 0008                       .BLKB   1
                        0009                .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
                0000000A 0009                       .BLKB   1
                        000A                .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
                0000000C 000A                       .BLKW   1
                        000C                .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
                00000010 000C                       .BLKL   1
                        0010                .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                        0010                .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
                00000000                    .RESTORE
                        0000     71         $DEVDEF                             ;DEFINE DEVICE CHARACTERISTICS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 001C               .=0
```

```
                    00000000                .RESTORE
                        0000    72          $IODEF                          ;DEFINE I/O FUNCTION CODES
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                    00000000  001C          .=0
                    00000000                .RESTORE
                        0000    73          $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                    00000000  001C          .=0
                        0000                .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
                    00000004  0000                      .BLKL   1
                        0004                .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
                    00000008  0004                      .BLKL   1
                        0008                .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
                    0000000A  0008                      .BLKW   1
                        000A                .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
                    0000000B  000A                      .BLKB   1
                        000B                .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
                    0000000C  000B                      .BLKB   1
                        000C                .IIF    NB,IRP$L_PID,   IRP$L_PID:
                    00000010  000C                      .BLKL   1
                        0010                .IIF    NB,IRP$L_AST,   IRP$L_AST:
                    00000014  0010                      .BLKL   1
                        0014                .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
                    00000018  0014                      .BLKL   1
                        0018                .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
                    0000001C  0018                      .BLKL   1
                        001C                .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
                    00000020  001C                      .BLKL   1
                        0020                .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
                    00000022  0020                      .BLKW   1
                        0022                .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
                    00000023  0022                      .BLKB   1
                        0023                .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
                    00000024  0023                      .BLKB   1
                        0024                .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
                    00000028  0024                      .BLKL   1
                        0028                .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
                    0000002A  0028                      .BLKW   1
                        002A                .IIF    NB,IRP$W_STS,   IRP$W_STS:
                    0000002C  002A                      .BLKW   1
                        002C                .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
                    00000030  002C                      .BLKL   1
                        0030                .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
                    00000032  0030                      .BLKW   1
                        0032                .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
                    00000034  0032                      .BLKW   1
                        0034                .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
                        0034                .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
                    00000038  0034                      .BLKL   1
                        0038                .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
                        0038                .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
                        0038                .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
                    00000039  0038                      .BLKB   1
                    0000003C  0039                      .BLKB   3
                        003C                .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
```

```
                    003C                .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
                    003C                .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
          0000003E  003C                        .BLKW   1
                    003E                .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
          00000040  003E                        .BLKW   1
                    0040                .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
          00000044  0040                        .BLKL   1
                    0044                .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
          00000048  0044                        .BLKL   1
                    0048                .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
          0000004C  0048                        .BLKL   1
                    004C                .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
          00000050  004C                        .BLKL   1
                    0050                .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
          00000054  0050                        .BLKL   1
          00000058  0054                        .BLKL   1
          0000005C  0058                        .BLKL   1
                    005C                .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
                    005C                .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
                00000000                .RESTORE
                    0000     74         $PCBDEF                         ;DEFINE PCB VALUES
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
          00000000  005C                .=0
                    0000                .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
          00000004  0000                        .BLKL   1
                    0004                .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
          00000008  0004                        .BLKL   1
                    0008                .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
          0000000A  0008                        .BLKW   1
                    000A                .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
          0000000B  000A                        .BLKB   1
                    000B                .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
          0000000C  000B                        .BLKB   1
                    000C                .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
          0000000D  000C                        .BLKB   1
                    000D                .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
          0000000E  000D                        .BLKB   1
                    000E                .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
          00000010  000E                        .BLKW   1
                    0010                .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
          00000014  0010                        .BLKL   1
                    0014                .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
          00000018  0014                        .BLKL   1
                    0018                .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
          0000001C  0018                        .BLKL   1
                    001C                .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
          00000020  001C                        .BLKL   1
                    0020                .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
          00000024  0020                        .BLKL   1
                    0024                .IIF    NB,PCB$L_STS,   PCB$L_STS:
          00000028  0024                        .BLKL   1
                    0028                .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
          0000002C  0028                        .BLKL   1
                    002C                .IIF    NB,PCB$W_STATE, PCB$W_STATE:
          0000002E  002C                        .BLKW   1
                    002E                .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
```

```
0000002F  002E                                  .BLKB   1
          002F                  .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                                  .BLKB   1
          0030                  .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                                  .BLKW   1
          0032                  .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                                  .BLKW   1
          0034                  .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                                  .BLKW   1
          0036                  .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                                  .BLKW   1
          0038                  .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                                  .BLKW   1
          003A                  .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                                  .BLKW   1
          003C                  .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                                  .BLKW   1
          003E                  .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                                  .BLKW   1
          0040                  .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                                  .BLKW   1
          0042                  .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                                  .BLKW   1
          0044                  .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                                  .BLKB   8
          004C                  .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
          004C                  .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                                  .BLKL   1
          0054                  .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                                  .BLKL   1
          005C                  .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                                  .BLKL   1
          0060                  .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                                  .BLKL   1
          0064                  .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                                  .BLKL   1
          0068                  .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                                  .BLKB   16
          0078                  .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                                  .BLKL   1
          007C                  .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084  007C                                  .BLKQ   1
          0084                  .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                                  .BLKL   1
          0088                  .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
          0088                  .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                                  .BLKW   1
          008A                  .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                                  .BLKW   1
          008C                  .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
          008C                  .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
      00000000                  .RESTORE
          0000    75            $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
```

```
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000  008C                .=0
                        00000000            .RESTORE
                        0000    76          $UCBDEF                             ;DEFINE UCB OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000  008C                .=0
                        0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                        0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                00000004  0000                        .BLKL   1
                        0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                        0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                00000008  0004                        .BLKL   1
                        0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                0000000A  0008                        .BLKW   1
                        000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                0000000B  000A                        .BLKB   1
                        000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                0000000C  000B                        .BLKB   1
                        000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                        000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                        000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                0000000D  000C                        .BLKB   1
                00000010  000D                        .BLKB   3
                        0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                        0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
                00000014  0010                        .BLKL   1
                        0014                .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                        0014                .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                        0014                .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
                00000016  0014                        .BLKW   1
                        0016                .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
                00000018  0016                        .BLKW   1
                        0018                .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                        0018                .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
                0000001A  0018                        .BLKW   1
                        001A                .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                        001A                .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
                0000001C  001A                        .BLKW   1
                        001C                .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
                00000020  001C                        .BLKL   1
                        0020                .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
                00000024  0020                        .BLKL   1
                        0024                .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
                00000028  0024                        .BLKL   1
                        0028                .IIF    NB,UCB$L_PID,   UCB$L_PID:
                0000002C  0028                        .BLKL   1
                        002C                .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
                00000030  002C                        .BLKL   1
                        0030                .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
                00000034  0030                        .BLKL   1
                        0034                .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
                00000038  0034                        .BLKL   1
                        0038                .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
                00000039  0038                        .BLKB   1
                        0039                .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
```

```
                    0000003A  0039                        .BLKB   1
                              003A              .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
                    0000003C  003A                        .BLKW   1
                              003C              .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                              003C              .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                              003C              .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
                    0000003D  003C                        .BLKB   1
                              003D              .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                              003D              .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
                    0000003E  003D                        .BLKB   1
                              003E              .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                              003E              .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
                    0000003F  003E                        .BLKB   1
                              003F              .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
                    00000040  003F                        .BLKB   1
                              0040              .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
                    00000044  0040                        .BLKL   1
                              0044              .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
                    00000048  0044                        .BLKL   1
                              0048              .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
                    0000004A  0048                        .BLKW   1
                              004A              .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                              004A              .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
                    0000004B  004A                        .BLKB   1
                              004B              .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
                    0000004C  004B                        .BLKB   1
                              004C              .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
                    00000050  004C                        .BLKL   1
                              0050              .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
                    00000052  0050                        .BLKW   1
                              0052              .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                              0052              .IIF    NB,UCB$B_STATE, UCB$B_STATE:
                    00000053  0052                        .BLKB   1
                              0053              .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
                    00000054  0053                        .BLKB   1
                              0054              .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
                    00000058  0054                        .BLKL   1
                              0058              .IIF    NB,UCB$W_STS,   UCB$W_STS:
                    0000005A  0058                        .BLKW   1
                              005A              .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
                    0000005C  005A                        .BLKW   1
                              005C              .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                              005C              .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
                    00000060  005C                        .BLKL   1
                              0060              .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
                    00000064  0060                        .BLKL   1
                              0064              .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                              0064              .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
                    00000068  0064                        .BLKL   1
                              0068              .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
                    0000006C  0068                        .BLKL   1
                              006C              .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
                    0000006E  006C                        .BLKW   1
                              006E              .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
                    00000070  006E                        .BLKW   1
                              0070              .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
                    00000071  0070                        .BLKB   1
```

```
                     0071          .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072             0071                          .BLKB   1
                     0072          .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074             0072                          .BLKW   1
                     0074          .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                     0074          .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000             0074          . = 0
                     0000          .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002             0000                          .BLKW   1
00000074             0002          . = 116
                     0074          .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078             0074                          .BLKL   1
                     0078          .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C             0078                          .BLKL   1
                     007C          .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080             007C                          .BLKL   1
                     0080          .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084             0080                          .BLKL   1
                     0084          .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088             0084                          .BLKL   1
                     0088          .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C             0088                          .BLKL   1
                     008C          .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090             008C                          .BLKL   1
                     0090          .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                     0090          .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074             0090          . = 116
                     0074          .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075             0074                          .BLKB   1
                     0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076             0075                          .BLKB   1
                     0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077             0076                          .BLKB   1
                     0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078             0077                          .BLKB   1
                     0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C             0078                          .BLKL   1
0000007E             007C                          .BLKW   1
                     007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080             007E                          .BLKW   1
                     0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084             0080                          .BLKL   1
00000080             0084          . = 128
00000084             0080                          .BLKL   1
                     0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088             0084                          .BLKL   1
                     0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A             0088                          .BLKW   1
                     008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C             008A                          .BLKW   1
                     008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                     008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E             008C                          .BLKW   1
                     008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090             008E                          .BLKW   1
                     0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092             0090                          .BLKW   1
```

```
                        0092                .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094                0092                        .BLKW   1
                        0094                .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095                0094                        .BLKB   1
                        0095                .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096                0095                        .BLKB   1
                        0096                .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098                0096                        .BLKW   1
00000096                0098                . = 150
00000098                0096                        .BLKW   1
                        0098                .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C                0098                        .BLKL   1
                        009C                .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0                009C                        .BLKL   1
                        00A0                .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4                00A0                        .BLKL   1
                        00A4                .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6                00A4                        .BLKW   1
                        00A6                .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7                00A6                        .BLKB   1
000000A8                00A7                        .BLKB   1
00000074                00A8                . = 116
00000078                0074                        .BLKL   1
0000007C                0078                        .BLKL   1
00000080                007C                        .BLKL   1
00000084                0080                        .BLKL   1
00000088                0084                        .BLKL   1
0000008C                0088                        .BLKL   1
                        008C                .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090                008C                        .BLKL   1
00000094                0090                        .BLKL   1
00000098                0094                        .BLKL   1
0000009C                0098                        .BLKL   1
                        009C                .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D                009C                        .BLKB   1
                        009D                .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E                009D                        .BLKB   1
                        009E                .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F                009E                        .BLKB   1
000000A0                009F                        .BLKB   1
                        00A0                .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1                00A0                        .BLKB   1
                        00A1                .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2                00A1                        .BLKB   1
                        00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3                00A2                        .BLKB   1
000000A4                00A3                        .BLKB   1
                        00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5                00A4                        .BLKB   1
                        00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7                00A5                        .BLKW   1
000000A8                00A7                        .BLKB   1
                        00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC                00A8                        .BLKL   1
000000B0                00AC                        .BLKL   1
000000B4                00B0                        .BLKL   1
000000B8                00B4                        .BLKL   1
```

```
                     00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC             00B8                                .BLKL   1
                     00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                     00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074             00BC                . = 116
                     0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078             0074                                .BLKL   1
                     0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C             0078                                .BLKL   1
                     007C                .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E             007C                                .BLKW   1
00000080             007E                                .BLKW   1
                00000000                .RESTORE
                     0000    77         $VADEF                                   ;DEFINE VIRTUAL ADDRESS FIELDS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000             00BC                .=0
                00000000                .RESTORE
                     0000    78
                     0000    79 ;
                     0000    80 ; LOCAL SYMBOLS
                     0000    81 ;
                     0000    82 ; ARGUMENT LIST OFFSET DEFINITIONS
                     0000    83 ;
                     0000    84
00000000             0000    85 P1=0                                             ;FIRST FUNCTION DEPENDENT PARAMETER
00000004             0000    86 P2=4                                             ;SECOND FUNCTION DEPENDENT PARAMETER
00000008             0000    87 P3=8                                             ;THIRD FUNCTION DEPENDENT PARAMETER
0000000C             0000    88 P4=12                                            ;FOURTH FUNCTION DEPENDENT PARAMETER
00000010             0000    89 P5=16                                            ;FIFTH FUNCTION DEPENDENT PARAMETER
00000014             0000    90 P6=20                                            ;SIXTH FUNCTION DEPENDENT PARAMETER
```

```
                     0000     93          .SBTTL  ONE PARAMETER FUNCTION PROCESSING
                     0000     94 ;+
                     0000     95 ; EXE$ONEPARM - ONE PARAMETER FUNCTION PROCESSING
                     0000     96 ;
                     0000     97 ; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
                     0000     98 ; PROCESS A ONE PARAMETER FUNCTION THAT REQUIRES NO SPECIAL CHECKING.
                     0000     99 ;
                     0000    100 ; INPUTS:
                     0000    101 ;
                     0000    102 ;       R0 = SCRATCH.
                     0000    103 ;       R1 = SCRATCH.
                     0000    104 ;       R2 = SCRATCH.
                     0000    105 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     0000    106 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0000    107 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     0000    108 ;       R6 = ADDRESS OF CCB.
                     0000    109 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     0000    110 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     0000    111 ;       R9 = SCRATCH.
                     0000    112 ;       R10 = SCRATCH.
                     0000    113 ;       R11 = SCRATCH.
                     0000    114 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     0000    115 ;
                     0000    116 ; OUTPUTS:
                     0000    117 ;
                     0000    118 ; ***TBS***
                     0000    119 ;-
                     0000    120
                     0000    121          .ENABL  LSB
                     0000    122 EXE$ONEPARM::                               ;ONE PARAMETER FUNCTION PROCESSING
 34 A3   6C   D0     0000    123          MOVL    P1(AP),IRP$L_MEDIA(R3)     ;STORE PARAMETER IN MEDIA ADDRESS
         03   11     0004    124          BRB     10$                        ;
```

```
                        0006   126          .SBTTL  ZERO PARAMETER FUNCTION PROCESSING
                        0006   127 ;+
                        0006   128 ; EXE$ZEROPARAM - ZERO PARAMETER FUNCTION PROCESSING
                        0006   129 ;
                        0006   130 ; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
                        0006   131 ; PROCESS A ZERO PARAMETER FUNCTION THAT REQUIRES NO ADDITION CHECKING.
                        0006   132 ;
                        0006   133 ; INPUTS:
                        0006   134 ;
                        0006   135 ;       R0 = SCRATCH.
                        0006   136 ;       R1 = SCRATCH.
                        0006   137 ;       R2 = SCRATCH.
                        0006   138 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        0006   139 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        0006   140 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                        0006   141 ;       R6 = ADDRESS OF CCB.
                        0006   142 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                        0006   143 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                        0006   144 ;       R9 = SCRATCH.
                        0006   145 ;       R10 = SCRATCH.
                        0006   146 ;       R11 = SCRATCH.
                        0006   147 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                        0006   148 ;
                        0006   149 ; OUTPUTS:
                        0006   150 ;
                        0006   151 ; ***TBS***
                        0006   152 ;-
                        0006   153
                        0006   154 EXE$ZEROPARM::                               ;ZERO PARAMETER FUNCTION PROCESSING
34 A3   D4              000    155          CLRL    IRP$L_MEDIA(R3)            ;CLEAR PARAMETER
 FFF4'  31              0009   156 10$:     BRW     EXE$QIODRVPKT              ;QUEUE I/O PACKET TO DRIVER
                        000C   157          .DSABL  LSB
```

```
                        000C   159          .SBTTL  READ AND WRITE FUNCTION PROCESSING
                        000C   160 ;+
                        000C   161 ; EXE$READ - READ FUNCTION PROCESSING
                        000C   162 ; EXE$WRITE - WRITE FUNCTION PROCESSING
                        000C   163 ; EXE$MODIFY - MODIFY FUNCTION PROCESSING
                        000C   164 ;
                        000C   165 ; THESE ROUTINES ARE CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
                        000C   166 ; PROCESS A READ OR WRITE PHYSICAL OR LOGICAL FUNCTION.
                        000C   167 ; EXE$MODIFY IS USED FOR FUNCTIONS THAT READ AND WRITE MEMORY.
                        000C   168 ;
                        000C   169 ; INPUTS:
                        000C   170 ;
                        000C   171 ;       R0 = SCRATCH.
                        000C   172 ;       R1 = SCRATCH.
                        000C   173 ;       R2 = SCRATCH.
                        000C   174 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        000C   175 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        000C   176 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                        000C   177 ;       R6 = ADDRESS OF CCB.
                        000C   178 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                        000C   179 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                        000C   180 ;       R9 = SCRATCH.
                        000C   181 ;       R10 = SCRATCH.
                        000C   182 ;       R11 = SCRATCH.
                        000C   183 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                        000C   184 ;
                        000C   185 ; OUTPUTS:
                        000C   186 ;
                        000C   187 ; ***TBS***
                        000C   188 ;-
                        000C   189
                        000C   190          .ENABL  LSB
                        000C   191 EXE$MODIFY::                                 ;MODIFY FUNCTION PROCESSING
   52   44'AF    DE     000C   192          MOVAL   B^EXE$MODIFYLOCK,R2        ;SET ADDRESS OF BUFFER CHECK ROUTINE
        04    11       0010   193          BRB     5$
                        0012   194 EXE$READ::                                   ;READ FUNCTION PROCESSING
   52   3E'AF    DE     0012   195          MOVAL   B^EXE$READLOCK,R2          ;SET ADDRESS OF BUFFER CHECK ROUTINE
04 2A A3    01    E3    0016   196 5$:      BBCS    #IRP$V_FUNC,IRP$W_STS(R3),10$ ;SET READ FUNCTION STATUS
                        001B   197 EXE$WRITE::                                  ;WRITE FUNCTION PROCESSING
   52   41'AF    DE     001B   198          MOVAL   B^EXE$WRITELOCK,R2         ;SET ADDRESS OF BUFFER CHECK ROUTINE
38 A3   0C AC    D0     001F   199 10$:     MOVL    P4(AP),IRP$B_CARCON(R3)    ;INSERT CARRIAGE CONTROL BYTE
      06   00    ED     0024   200          CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;PHYSICAL I/O FUNCTION?
   1F   20 A3           0027   201                  IRP$W_FUNC(R3),#IO$_PHYSICAL ;
        04    15       002A   202          BLEQ    20$                        ;IF LEQ YES
        15    A2       002C   203          SUBW    #IO$_READLBLK-IO$_READPBLK,- ;CONVERT TO PHYSICAL FUNCTION
      20 A3           002E   204                  IRP$W_FUNC(R3)             ;
   51   04 AC    3C     0030   205 20$:     MOVZWL  P2(AP),R1                  ;GET NUMBER OF BYTES TO TRANSFER
        05    13       0034   206          BEQL    30$                        ;IF EQL NONE
      50   6C    D0     0036   207          MOVL    P1(AP),R0                  ;GET STARTING VIRTUAL ADDRESS OF TRANSFER
        62    16       0039   208          JSB     (R2)                       ;CHECK BUFFER AND LOCK IN MEMORY
         FFC2'  31     003B   209 30$:     BRW     EXE$QIODRVPKT              ;QUEUE I/O PACKET TO DRIVER
                        003E   210          .DSABL  LSB
```

```
                     003E    212          .SBTTL  READ AND WRITE FUNCTION BUFFER CHECK AND LOCK ROUTINES
                     003E    213 ;+
                     003E    214 ; EXE$READLOCK - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK
                     003E    215 ; EXE$WRITELOCK - CHECK BUFFER FOR WRITE ACCESSIBILITY AND LOCK
                     003E    216 ; EXE$MODIFYLOCK - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK
                     003E    217 ;
                     003E    218 ; THESE ROUTINES ARE CALLED TO CHECK THE ACCESSIBILITY OF AN I/O BUFFER AND
                     003E    219 ; TO LOCK THE BUFFER IN MEMORY FOR A DIRECT MEMORY TRANSFER.
                     003E    220 ;
                     003E    221 ; INPUTS:
                     003E    222 ;
                     003E    223 ;       R0 = STARTING ADDRESS OF I/O BUFFER.
                     003E    224 ;       R1 = LENGTH OF TRANSFER IN BYTES.
                     003E    225 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     003E    226 ;       R6 = ADDRESS OF CCB.
                     003E    227 ;
                     003E    228 ; OUTPUTS:
                     003E    229 ;
                     003E    230 ;       THE I/O BUFFER IS CHECKED FOR THE PROPER ACCESSIBILITY. IF THE
                     003E    231 ;       CHECK SUCCEEDS, THEN THE BUFFER IS LOCKED IN MEMORY AND THE STARTING
                     003E    232 ;       ADDRESS OF THE PAGE TABLE ENTRIES THAT MAP THE TRANSFER IS STORED
                     003E    233 ;       IN THE I/O PACKET. ELSE THE I/O IS COMPLETED WITH A STATUS OF
                     003E    234 ;       ACCESS VIOLATION.
                     003E    235 ;-
                     003E    236
                     003E    237 EXE$READLOCK::                          ;CHECK BUFFER FOR READ FUNCTION AND LOCK
          11   10    003E    238          BSBB    EXE$READLOCKR          ;EXE$READLOCKR RETURNS NORMALLY ON
                     0040    239                                         ;SUCCESS, VIA COROUTINE CALL ON FAILURE
               05    0040    240          RSB                            ;RETURNS TO CALLER ON SUCCESS, TO
                     0041    241                                         ;EXE$READLOCKR ON FAILURE
                     0041    242
                     0041    243 EXE$WRITELOCK::                         ;CHECK BUFFER FOR WRITE FUNCTION AND LOCK
          15   10    0041    244          BSBB    EXE$WRITELOCKR         ;EXE$WRITELOCKR RETURNS NORMALLY ON
                     0043    245                                         ;SUCCESS, VIA COROUTINE CALL ON FAILURE
               05    0043    246          RSB                            ;RETURNS TO CALLER ON SUCCESS, TO
                     0044    247                                         ;EXE$WRITELOCKR ON FAILURE
                     0044    248
                     0044    249 EXE$MODIFYLOCK::                        ;CHECK BUFFER FOR MODIFY FUNCTION AND LOCK
          01   10    0044    250          BSBB    EXE$MODIFYLOCKR        ;EXE$MODIFYLOCKR RETURNS NORMALLY ON
                     0046    251                                         ;SUCCESS, VIA COROUTINE CALL ON FAILURE
               05    0046    252          RSB                            ;RETURNS TO CALLER ON SUCCESS, TO
                     0047    253                                         ;EXE$MODIFYLOCKR ON FAILURE
```

```
                     0047   255          .SBTTL  READ AND WRITE BUFFER CHECK AND LOCK AND RETURN ROUTINES
                     0047   256 ;+
                     0047   257 ; EXE$READLOCKR - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK AND RETURN
                     0047   258 ;                 ON ERROR
                     0047   259 ; EXE$WRITELOCKR - CHECK BUFFER FOR WRITE ACCESSIBILITY AND LOCK AND RETURN
                     0047   260 ;                  ON ERROR
                     0047   261 ; EXE$MODIFYLOCKR - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK AND RETURN
                     0047   262 ;                  ON ERROR
                     0047   263 ;
                     0047   264 ; THESE ROUTINES ARE CALLED TO CHECK THE ACCESSIBILITY OF AN I/O BUFFER
                     0047   265 ; AND TO LOCK THE BUFFER IN MEMORY FOR A DIRECT MEMORY TRANSFER.  IN
                     0047   266 ; ADDITION, THESE ROUTINES PERFORM A COROUTINE CALL IF THERE IS AN ERROR
                     0047   267 ; OR ANY PAGES HAVE TO BE FAULTED IN.  THE PURPOSE OF THE COROUTINE
                     0047   268 ; CALL IS TO ALLOW THE CALLER TO PERFORM ANY NECESSARY CLEANUP BEFORE
                     0047   269 ; THE QIO IS BACKED UP OR ABORTED.  THESE ROUTINES ARE TYPICALLY CALLED
                     0047   270 ; BY DRIVERS THAT MUST LOCK MULTIPLE AREAS INTO MEMORY.  SINCE THESE
                     0047   271 ; ROUTINES CANNOT UNLOCK AREAS PREVIOUSLY LOCKED, THE COROUTINE CALL ALLOWS
                     0047   272 ; THE CALLER (THE DRIVER) TO UNLOCK PREVIOUSLY LOCKED AREAS (AND PERFORM
                     0047   273 ; ANY OTHER CLEANUP) AND THEN RETURN HERE TO BACK UP OR ABORT THE I/O.
                     0047   274 ;
                     0047   275 ; EXE$MODIFYLOCKR IS USED WHEN THE BUFFER WILL BE READ AND WRITTEN BY THE
                     0047   276 ; I/O DEVICE.  IT DISABLES AN OPTIMIZATION IN MMG$IOLOCK WHICH IS USED
                     0047   277 ; WHEN THE BUFFER IS ONLY WRITTEN.
                     0047   278 ;
                     0047   279 ; INPUTS:
                     0047   280 ;
                     0047   281 ;       R0 = STARTING ADDRESS OF I/O BUFFER.
                     0047   282 ;       R1 = LENGTH OF BUFFER IN BYTES.
                     0047   283 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0047   284 ;       R6 = ADDRESS OF CCB.
                     0047   285 ;
                     0047   286 ; OUTPUTS:
                     0047   287 ;
                     0047   288 ;       THE I/O BUFFER IS CHECKED FOR THE PROPER ACCESSIBILITY.  IF THE
                     0047   289 ;       CHECK SUCCEEDS, THEN THE BUFFER IS LOCKED IN MEMORY AND THE STARTING
                     0047   290 ;       ADDRESS OF THE PAGE TABLE ENTRIES THAT MAP THE TRANSFER IS STORED
                     0047   291 ;       IN THE I/O PACKET.
                     0047   292 ;
                     0047   293 ;       R0 = RETURN CODE
                     0047   294 ;
                     0047   295 ;       NOTE THAT IF THERE ARE NO ERRORS AND NO PAGES HAVE TO BE FAULTED
                     0047   296 ;       IN, THEN THESE ROUTINES RETURN NORMALLY.  HOWEVER, IF THERE IS AN
                     0047   297 ;       ERROR OR A PAGE HAS TO BE FAULTED IN, THEN THE CALLER IS CALLED
                     0047   298 ;       BY A COROUTINE CALL.  THE CALLER'S RSB THEN RETURNS HERE WHERE
                     0047   299 ;       THE QIO IS EITHER BACKED UP OR ABORTED.  NOTE THAT IN THIS CASE
                     0047   300 ;       THE CALLER'S ERROR HANDLING CODE MUST PRESERVE ALL REGISTERS,
                     0047   301 ;       INCLUDING R0 AND R1.
                     0047   302 ;-
                     0047   303          .ENABL  LSB
                     0047   304 EXE$MODIFYLOCKR::                          ;CHECK BUFFER FOR MODIFY FUNCTION AND LOCK
          50   DD   0047   305          PUSHL   R0                         ;SAVE STARTING ADDRESS OF BUFFER
        0096   30   0049   306          BSBW    EXE$READCHKR               ;CHECK BUFFER FOR READ FUNCTION
    52    04   C8   004C   307          BISL    #4,R2                      ;DISABLE OPTIMIZATION IN MMG$IOLOCK
          0C   11   004F   308          BRB     10$
                     0051   309
                     0051   310 EXE$READLOCKR::                           ;CHECK BUFFER FOR READ FUNCTION AND LOCK
          50   DD   0051   311          PUSHL   R0                         ;SAVE STARTING ADDRESS OF BUFFER
```

```
                 008C   30   0053   312          BSBW    EXE$READCHKR                ;CHECK BUFFER FOR READ FUNCTION
                   05   11   0056   313          BRB     10$
                                0058   314
                                0058   315 EXE$WRITELOCKR::                          ;CHECK BUFFER FOR WRITE FUNCTION AND LOCK
                   50   DD   0058   316          PUSHL   R0                          ;SAVE STARTING ADDRESS OF BUFFER
                 00AF   30   005A   317          BSBW    EXE$WRITECHKR               ;CHECK BUFFER FOR WRITE FUNCTION
               1D 50    E9   005D   318 10$:     BLBC    R0,15$                      ;BRANCH IF ERROR
                  50 8ED0   0060   319          POPL    R0                          ;RESTORE STARTING ADDRESS OF BUFFER
      30 A3    50   FE00 8F   AB   0063   320          BICW3   #^C<VA$M_BYTE>,R0,IRP$W_BOFF(R3) ;SET BYTE OFFSET IN PAGE
                  53   DD   006A   321          PUSHL   R3                          ;SAVE ADDRESS OF I/O PACKET
          00000000'EF   16   006C   322          JSB     MMG$IOLOCK                  ;LOCK PAGES FOR I/O                     [03]
               53   8E   D0   0072   323          MOVL    (SP)+,R3                    ;RETRIEVE ADDRESS OF I/O PACKET
                 08 50    E9   0075   324          BLBC    R0,20$                      ;IF LBC LOCK FAILURE
            2C A3    51   D0   0078   325          MOVL    R1,IRP$L_SVAPTE(R3)         ;INSERT ADDRESS OF FIRST PTE IN PACKET
                  05   007C   326          RSB                                 ;
               5E   04   C0   007D   327 15$:     ADDL    #4,SP                       ;THROW AWAY OLD R0
                  9E   16   0080   328 20$:     JSB     @(SP)+                      ;COROUTINE CALL TO CLEANUP
                  50   D5   0082   329          TSTL    R0                          ;ERRORS ENCOUNTERED?
                  45   12   0084   330          BNEQ    50$                         ;IF NEQ YES
                  51   DD   0086   331          PUSHL   R1                          ;SAVE VIRTUAL ADDRESS OF PAGE TO FAULT
               3E A4   B6   0088   332          INCW    PCB$W_DIOCNT(R4)            ;ADJUST DIRECT I/O COUNT
               0A A6   B7   008B   333          DECW    CCB$W_IOC(R6)               ;DECREMENT CHANNEL I/O COUNT
         0F 2A A3    07   E1   008E   334          BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),25$  ;BR. IF NO DIAGNOSTIC BUFFER
            50   44 A3   D0   0093   335          MOVL    IRP$L_DIAGBUF(R3),R0        ;GET ADDRESS OF DIAGNOSTIC BUFFER
                  53   DD   0097   336          PUSHL   R3                          ;SAVE R3
          00000000'EF   16   0099   337          jsb     l^exe$deanonpaged                   ; Deallocate diagnostic buffer
                  53 8ED0   009F   338          POPL    R3                          ;RESTORE R3
                                00A2   339 25$:
   03 0B A3    06   E1   00A2   340          BBC     #ACB$V_QUOTA,IRP$B_RMOD(R3),30$ ;IF CLR, NO AST SPECIFIED
               38 A4   B6   00A7   341          INCW    PCB$W_ASTCNT(R4)            ;ADJUST AST COUNT
            50   53   D0   00AA   342 30$:     MOVL    R3,R0                       ;SET ADDRESS OF BLOCK TO DEALLOCATE
          00000000'EF   16   00AD   343          jsb     l^exe$deanonpaged                   ; Deallocate I/O packet
                  02   BA   00B3   344          POPR    #^M<R1>                     ;RETRIEVE VIRTUAL ADDRESS OF PAGE TO FAULT
            5E   5D   D0   00B5   345          MOVL    FP,SP                       ;TRIM STACK BACK TO CHANGE MODE FRAME
         5C    08 AD   7D   00B8   346          MOVQ    8(FP),AP                    ;RESTORE USER ARGUMENT AND FRAME POINTERS
            5E   14   C0   00BC   347          ADDL    S^#EXE$C_CMSTKSZ,SP         ;REMOVE CHANGE MODE CALL FRAME FROM STACK
      50    8E   04   C3   00BF   348          SUBL3   #4,(SP)+,R0                 ;CALCULATE RESTART ADDRESS
               C7'AF   9F   00C3   349          PUSHAB  B^40$                       ;SET NEW RETURN ADDRESS
                  02   00C6   350          REI                                 ;
                  61   95   00C7   351 40$:     TSTB    (R1)                        ;FAULT USER BUFFER AGAIN
                  60   17   00C9   352          JMP     (R0)                        ;REPEAT SYSTEM SERVICE
                FF32'  31   00CB   353 50$:     BRW     EXE$ABORTIO                 ;ABORT I/O REQUEST
                                00CE   354          .DSABL  LSB
```

```
                        00CE    356          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
                        00CE    357 ;+
                        00CE    358 ; EXE$READCHK - CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
                        00CE    359 ;
                        00CE    360 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A READ I/O
                        00CE    361 ; FUNCTION.
                        00CE    362 ;
                        00CE    363 ; INPUTS:
                        00CE    364 ;
                        00CE    365 ;       R0 = ADDRESS OF BUFFER.
                        00CE    366 ;       R1 = SIZE OF TRANSFER IN BYTES.
                        00CE    367 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        00CE    368 ;
                        00CE    369 ; OUTPUTS:
                        00CE    370 ;
                        00CE    371 ;       IF BUFFER IS NOT WRITE ACCESSIBLE, THEN THE I/O REQUEST IS TERM-
                        00CE    372 ;       INATED VIA EXE$IOFINISH WITH A STATUS OF SS$_ACCVIO.
                        00CE    373 ;
                        00CE    374 ;       IF BUFFER IS WRITE ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                        00CE    375 ;       TURNED:
                        00CE    376 ;
                        00CE    377 ;               R0 = ADDRESS OF BUFFER.
                        00CE    378 ;               R1 = SIZE OF TRANSFER IN BYTES.
                        00CE    379 ;               R2 = READ FUNCTION INDICATOR (1).
                        00CE    380 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                        00CE    381 ;
                        00CE    382 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                        00CE    383 ;               IRP$W_FUNC(R3) = READ.
                        00CE    384 ;-
                        00CE    385
                        00CE    386          .ENABL  LSB
                        00CE    387 EXE$READCHK::                               ;CHECK BUFFER FOR READ FUNCTION
 50     DD              00CE    388          PUSHL   R0                         ;SAVE ADDRESS OF BUFFER
 10     10              00D0    389          BSBB    EXE$READCHKR               ;CHECK BUFFER
 04     11              00D2    390          BRB     10$
```

```
                        00D4    392          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
                        00D4    393 ;+
                        00D4    394 ; EXE$WRITECHK - CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
                        00D4    395 ;
                        00D4    396 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A WRITE I/O
                        00D4    397 ; FUNCTION.
                        00D4    398 ;
                        00D4    399 ; INPUTS:
                        00D4    400 ;
                        00D4    401 ;       R0 = ADDRESS OF BUFFER.
                        00D4    402 ;       R1 = SIZE OF TRANSFER IN BYTES.
                        00D4    403 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        00D4    404 ;
                        00D4    405 ; OUTPUTS:
                        00D4    406 ;
                        00D4    407 ;       IF BUFFER IS NOT READ ACCESSIBLE, THEN THE I/O REQUEST IS TERM-
                        00D4    408 ;       INATED VIA EXE$IOFINISH WITH A STATUS OF SS$_ACCVIO.
                        00D4    409 ;
                        00D4    410 ;       IF BUFFER IS READ ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                        00D4    411 ;       TURNED:
                        00D4    412 ;
                        00D4    413 ;               R0 = ADDRESS OF BUFFER.
                        00D4    414 ;               R1 = SIZE OF TRANSFER IN BYTES.
                        00D4    415 ;               R2 = WRITE FUNCTION INDICATOR (0).
                        00D4    416 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                        00D4    417 ;
                        00D4    418 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                        00D4    419 ;               IRP$W_FUNC(R3) = WRITE.
                        00D4    420 ;-
                        00D4    421
                        00D4    422 EXE$WRITECHK::                              ;CHECK BUFFER FOR WRITE FUNCTION
   50    DD               00D4    423          PUSHL   R0                      ;SAVE ADDRESS OF BUFFER
   34    10               00D6    424          BSBB    EXE$WRITECHKR           ;CHECK BUFFER
03 50    E8               00D8    425 10$:     BLBS    R0,20$                  ;BRANCH IF SUCCESS
 FF22'   31               00DB    426          BRW     EXE$ABORTIO             ;ABORT I/O
   50 8ED0                00DE    427 20$:     POPL    R0                      ;RESTORE ADDRESS OF BUFFER
      05                  00E1    428          RSB
                          00E2    429          .DSABL  LSB
```

```
                                00E2    431          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
                                00E2    432 ;+
                                00E2    433 ; EXE$READCHKR - CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
                                00E2    434 ;
                                00E2    435 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A READ I/O
                                00E2    436 ; FUNCTION.  STATUS IS RETURNED IN R0.
                                00E2    437 ;
                                00E2    438 ; INPUTS:
                                00E2    439 ;
                                00E2    440 ;       R0 = ADDRESS OF BUFFER.
                                00E2    441 ;       R1 = SIZE OF TRANSFER IN BYTES.
                                00E2    442 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                00E2    443 ;
                                00E2    444 ; OUTPUTS:
                                00E2    445 ;
                                00E2    446 ;       IF THE BUFFER IS NOT WRITE ACCESSIBLE, THEN THE FOLLOWING
                                00E2    447 ;       VALUE IS RETURNED:
                                00E2    448 ;
                                00E2    449 ;       R0 = SS$_ACCVIO
                                00E2    450 ;
                                00E2    451 ;       IF BUFFER IS WRITE ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                                00E2    452 ;       TURNED:
                                00E2    453 ;
                                00E2    454 ;               R0 = SS$_NORMAL
                                00E2    455 ;               R1 = SIZE OF TRANSFER IN BYTES.
                                00E2    456 ;               R2 = READ FUNCTION INDICATOR (1).
                                00E2    457 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                                00E2    458 ;
                                00E2    459 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                                00E2    460 ;               IRP$W_FUNC(R3) = READ.
                                00E2    461 ;-
                                00E2    462
                                00E2    463          .ENABL  LSB
                                00E2    464 EXE$READCHKR::                              ;CHECK BUFFER FOR READ FUNCTION
          51    DD              00E2    465          PUSHL   R1                         ;SAVE R1
     51   50    C0              00E4    466          ADDL    R0,R1                      ;CALCULATE ENDING BUFFER ADDRESS
50   01FF 8F    AA              00E7    467          BICW    #VA$M_BYTE,R0              ;TRUNCATE TO START OF PAGE
     51   50    C2              00EC    468          SUBL    R0,R1                      ;CALCULATE LENGTH OF BUFFER TO PROBE
52   FE00 8F    32              00EF    469          CVTWL   #-^X200,R2                 ;SET ADDRESS ADJUSTMENT CONSTANT
                                00F4    470 10$:     IFNOWRT R1,(R0),ACCESS             ;CAN ENDS OF BUFFER BE WRITTEN?
60   51    00    0D              00F4                PROBEW  #0,R1,(R0)
          40    13              00F8                BEQL    ACCESS
     50   52    C2              00FA    471          SUBL    R2,R0                      ;CALCULATE VIRTUAL ADDRESS OF NEXT PAGE
   51    6142    3E              00FD    472          MOVAW   (R1)[R2],R1                ;CALCULATE NEW LENGTH
          F1    14              0101    473          BGTR    10$                        ;IF GTR MORE TO TEST
  2A A3    02    A8              0103    474          BISW    #IRP$M_FUNC,IRP$W_STS(R3)  ;SET READ FUNCTION
     52    01    D0              0107    475          MOVL    #1,R2                      ;SET READ FUNCTION INDICATOR
          23    11              010A    476          BRB     30$                        ;
```

```
                                010C    478          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
                                010C    479 ;+
                                010C    480 ; EXE$WRITECHKR - CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
                                010C    481 ;
                                010C    482 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A WRITE I/O
                                010C    483 ; FUNCTION.  STATUS IS RETURNED IN R0
                                010C    484 ;
                                010C    485 ; INPUTS:
                                010C    486 ;
                                010C    487 ;       R0 = ADDRESS OF BUFFER.
                                010C    488 ;       R1 = SIZE OF TRANSFER IN BYTES.
                                010C    489 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                010C    490 ;
                                010C    491 ; OUTPUTS:
                                010C    492 ;
                                010C    493 ;       IF BUFFER IS NOT READ ACCESSIBLE, THEN THE FOLLOWING VALUE IS
                                010C    494 ;       RETURNED:
                                010C    495 ;
                                010C    496 ;       R0 = SS$_ACCVIO
                                010C    497 ;
                                010C    498 ;       IF BUFFER IS READ ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
                                010C    499 ;       TURNED:
                                010C    500 ;
                                010C    501 ;               R0 = SS$_NORMAL
                                010C    502 ;               R1 = SIZE OF TRANSFER IN BYTES.
                                010C    503 ;               R2 = WRITE FUNCTION INDICATOR (0).
                                010C    504 ;               R3 = ADDRESS OF I/O REQUEST PACKET.
                                010C    505 ;
                                010C    506 ;               IRP$W_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
                                010C    507 ;               IRP$W_FUNC(R3) = WRITE.
                                010C    508 ;-
                                010C    509
                                010C    510 EXE$WRITECHKR::                             ;CHECK BUFFER FOR WRITE FUNCTION
          51    DD              010C    511          PUSHL   R1                         ;SAVE R1
     51   50    C0              010E    512          ADDL    R0,R1                      ;CALCULATE ENDING BUFFER ADDRESS
50   01FF 8F    AA              0111    513          BICW    #VA$M_BYTE,R0              ;TRUNCATE TO START OF PAGE
     51   50    C2              0116    514          SUBL    R0,R1                      ;CALCULATE LENGTH OF BUFFER TO PROBE
52   FE00 8F    32              0119    515          CVTWL   #-^X200,R2                 ;GET ADDRESS ADJUSTMENT CONSTANT
                                011E    516 20$:     IFNORD  R1,(R0),ACCESS             ;CAN ENDS OF BUFFER BE READ?
60   51   00    0C              011E                 PROBER  #0,R1,(R0)
          16    13              0122                 BEQL    ACCESS
     50   52    C2              0124    517          SUBL    R2,R0                      ;CALCULATE VIRTUAL ADDRESS OF NEXT PAGE
   51   6142    3E              0127    518          MOVAW   (R1)[R2],R1                ;CALCULATE NEW LENGTH
          F1    14              012B    519          BGTR    20$                        ;IF GTR MORE TO CHECK
          52    D4              012D    520          CLRL    R2                         ;SET WRITE FUNCTION INDICATOR
          51 8ED0               012F    521 30$:     POPL    R1                         ;RESTORE R1
  32 A3   51    B0              0132    522          MOVW    R1,IRP$W_BCNT(R3)          ;INSERT TRANSFER BYTE COUNT IN PACKET
     50   01    3C              0136    523          MOVZWL  #SS$_NORMAL,R0             ;SUCCESS RETURN
                05              0139    524          RSB                                ;
                                013A    525 ACCESS:                                     ;SET ACCESS VIOLATION
          51 8ED0               013A    526          POPL    R1                         ;RESTORE R1
     50   0C    3C              013D    527          MOVZWL  #SS$_ACCVIO,R0             ;
                05              0140    528          RSB
                                0141    529          .DSABL  LSB
```

```
                     0141   531          .SBTTL  SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
                     0141   532 ;+
                     0141   533 ; EXE$SETCHAR - SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
                     0141   534 ;
                     0141   535 ; THIS ROUTINE PLACES THE NEW CHARACTERISTICS SPECIFIED BY THE QUADWORD POINTED
                     0141   536 ; TO BY P1 INTO THE SECOND AND THIRD LONGWORDS OF THE DEVICE UCB.
                     0141   537 ;
                     0141   538 ; INPUTS:
                     0141   539 ;
                     0141   540 ;       R0 = SCRATCH.
                     0141   541 ;       R1 = SCRATCH.
                     0141   542 ;       R2 = SCRATCH.
                     0141   543 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     0141   544 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0141   545 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     0141   546 ;       R6 = ADDRESS OF CCB.
                     0141   547 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     0141   548 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     0141   549 ;       R9 = SCRATCH.
                     0141   550 ;       R10 = SCRATCH.
                     0141   551 ;       R11 = SCRATCH.
                     0141   552 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     0141   553 ;
                     0141   554 ; OUTPUTS:
                     0141   555 ;
                     0141   556 ;       THE CHARACTERISTICS SPECIFIED BY THE QUADWORD POINTER TO BY P1 ARE STORED
                     0141   557 ;       IN THE SECOND AND THIRD LONGWORDS OF THE DEVICE UCB.
                     0141   558 ;-
                     0141   559
                     0141   560          .ENABL  LSB
                     0141   561 EXE$SETCHAR::                                ;SET DEVICE MODE AND CHARACTERISTICS
          51   6C    D0     0141   562          MOVL    P1(AP),R1            ;GET ADDRESS OF CHARACTERISTICS
                     0144   563          IFNORD  #8,(R1),ACCESS               ;CAN CHARACTERISTICS QUADWORD BE READ?
    61    08   00    0C     0144                  PROBER  #0,#8,(R1)
               F0   13     0148                  BEQL    ACCESS
          57   23    D1     014A   564          CMPL    #IO$_SETMODE,R7      ;SET MODE FUNCTION?
               04   13     014D   565          BEQL    10$                  ;IF EQL YES
       38  A5   61   B0     014F   566          MOVW    (R1),UCB$B_DEVCLASS(R5) ;SET DEVICE TYPE AND CLASS
  3A   A5   02   A1   B0     0153   567 10$:     MOVW    2(R1),UCB$W_DEVBUFSIZ(R5) ;SET DEFAULT BUFFER SIZE
  3C   A5   04   A1   D0     0158   568          MOVL    4(R1),UCB$L_DEVDEPEND(R5) ;SET DEVICE CHARACTERISTICS
               14   11     015D   569          BRB     20$                  ;
```

```
                     015F    571          .SBTTL  SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS
                     015F    572 ;+
                     015F    573 ; EXE$SETMODE - SET DEVICE CHARACTERISTICS AND MODE
                     015F    574 ;
                     015F    575 ; FUNCTIONAL DESCRIPTION:
                     015F    576 ;
                     015F    577 ; THIS ROUTINE PLACES THE NEW CHARACTERISTICS SPECIFIED BY P1 INTO
                     015F    578 ; THE I/O PACKET FOR INSERTION INTO THE UCB WHEN THE UNIT IS IDLE.
                     015F    579 ; THE INPUT DATA IS IN THE FORM RETURNED BY $GTCHAN. THE SPECIFIED BUFFER
                     015F    580 ; IS ASSUMED TO BE 12 BYTES IN LENGTH. THE P2 LENGTH SPECIFIER IS IGNORED.
                     015F    581 ;
                     015F    582 ; THE NEW CHARACTERISTICS ARE PLACED IN IRP$L_MEDIA/MEDIA+4 AND THE
                     015F    583 ; PACKET IS QUEUED VIA EXE$QIODRVPKT.
                     015F    584 ;
                     015F    585 ; INPUTS:
                     015F    586 ;
                     015F    587 ;       R3 = I/O PACKET ADDRESS
                     015F    588 ;       R4 = CURRENT PCB
                     015F    589 ;       R5 = ACB ADDRESS
                     015F    590 ;       R6 = ASSIGNED CCB ADDRESS
                     015F    591 ;       AP = ADDRESS OF THE QIO ARGUMENT P1
                     015F    592 ;
                     015F    593 ; OUTPUTS:
                     015F    594 ;
                     015F    595 ;       R0 = STATUS OF THE OPERATION
                     015F    596 ;       R3+ ARE PRESERVED.
                     015F    597 ;
                     015F    598 ; COMPLETION CODES:
                     015F    599 ;
                     015F    600 ;       SS$_NORMAL - SUCCESSFUL
                     015F    601 ;       SS$_ACCVIO - BUFFER ACCESS VIOLATION
                     015F    602 ;-
                     015F    603
                     015F    604 EXE$SETMODE::                                   ;SET DEVICE MODE AND CHARACTERISTICS
          51    6C   D0   015F    605          MOVL    P1(AP),R1                 ;ADDRESS BUFFER
                           0162    606          IFNORD  #8,(R1),ACCESS            ;BR IF NO ACCESS
     61   08    00   0C   0162                  PROBER  #0,#8,(R1)
                D2   13   0166                  BEQL    ACCESS
       34 A3    61   7D   0168    607          MOVQ    (R1),IRP$L_MEDIA(R3)      ;INSERT CHARACTERISTICS IN I/O PACKET
             FE91'  31   016C    608          BRW     EXE$QIODRVPKT             ;QUEUE THE PACKET
```

```
                        016F    610          .SBTTL  SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
                        016F    611 ;+
                        016F    612 ; EXE$SENSEMODE - SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
                        016F    613 ;
                        016F    614 ; THIS ROUTINE OBTAINS THE CURRENT DEVICE MODE/CHARACTERISTICS FROM THE DEVICE
                        016F    615 ; DEPENDENT CHARACTERISTICS LONGWORD IN THE UCB AND IMMEDIATELY COMPLETES THE
                        016F    616 ; I/O OPERATION WITH THE SECOND LONGWORD OF THE FINAL I/O STATUS EQUAL TO THE
                        016F    617 ; DEVICE DEPENDENT CHARACTERSITICS.
                        016F    618 ;
                        016F    619 ; INPUTS:
                        016F    620 ;
                        016F    621 ;       R0 = SCRATCH.
                        016F    622 ;       R1 = SCRATCH.
                        016F    623 ;       R2 = SCRATCH.
                        016F    624 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        016F    625 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        016F    626 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                        016F    627 ;       R6 = ADDRESS OF CCB.
                        016F    628 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                        016F    629 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                        016F    630 ;       R9 = SCRATCH.
                        016F    631 ;       R10 = SCRATCH.
                        016F    632 ;       R11 = SCRATCH.
                        016F    633 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                        016F    634 ;
                        016F    635 ; OUTPUTS:
                        016F    636 ;
                        016F    637 ;       THE DEVICE DEPENDENT CHARACTERISTICS ARE OBTAINED FROM THE UCB AND
                        016F    638 ;       THE I/O IS COMPLETED WITH THE SECOND I/O STATUS LONGWORD EQUAL TO THE
                        016F    639 ;       DEVICE CHARACTERISTICS.
                        016F    640 ;-
                        016F    641
                        016F    642 EXE$SENSEMODE::                                 ;SENSE DEVICE MODE/CHARACTERISTICS
51    3C A5    D0      016F    643          MOVL    UCB$L_DEVDEPEND(R5),R1         ;GET DEVICE DEPENDENT CHARACTERSITICS
   50    01    3C      0173    644 20$:     MOVZWL  #SS$_NORMAL,R0                 ;SET NORMAL COMPLETION STATUS
      FE87'    31      0176    645          BRW     EXE$FINISHIO                   ;FINISH I/O OPERATION
                        0179    646          .DSABL  LSB
```

```
                     0179    648          .SBTTL  CARRIAGE CONTROL INTERPRETATION
                     0179    649 ;+
                     0179    650 ; EXE$CARRIAGE - INTERPRET CARRIAGE CONTROL SPECIFIER
                     0179    651 ;
                     0179    652 ; FUNCTIONAL DESCRIPTION:
                     0179    653 ;
                     0179    654 ; THIS ROUTINE IS USED BY THE LINE PRINTER DRIVER AND THE TERMINAL
                     0179    655 ; DRIVER TO INTERPRET THE CARRIAGE CONTROL SPECIFIER IN IRP$B_CARCON .
                     0179    656 ; NOTE THAT IRP$B_CARCON IS USED AS A LONGWORD!
                     0179    657 ;
                     0179    658 ; THE SPECIFIER IS AS FOLLOWS:
                     0179    659 ;
                     0179    660 ;       .BYTE 1 -- FORTRAN CARRIAGE CONTROL CHARACTER IF NOT 0
                     0179    661 ;       .BYTE 2 -- ***** IGNORED *****
                     0179    662 ;       .BYTE 3 -- PREFIX CARRIAGE CONTROL
                     0179    663 ;       .BYTE 4 -- SUFFIX CARRIAGE CONTROL
                     0179    664 ;
                     0179    665 ; THE PRE/SUF FIELDS ARE AS FOLLOWS
                     0179    666 ;
                     0179    667 ;       IF BIT 7=0 THEN BITS 6-0 ARE THE NUMBER OF NEWLINES TO INSERT.
                     0179    668 ;       IF BIT 7=1 AND BIT 6=0 THEN BITS 4-0 ARE THE ASCII CHARACTER TO
                     0179    669 ;                                  OUTPUT. ASCII SET C0 OR C1 IS SPECIFIED BY BIT 5.
                     0179    670 ;       IF BIT 7=1 AND BIT 6=1 THEN BITS 5-0 ARE THE PRINTER CHANNEL NUMBER
                     0179    671 ;
                     0179    672 ; ASCII SET C0 IS ASSUMED AND BIT 6 IS IGNORED IF BIT 7=0.
                     0179    673 ;
                     0179    674 ; INPUTS:
                     0179    675 ;
                     0179    676 ;       R3 = ADDRESS OF THE I/O PACKET
                     0179    677 ;       R5 = ADDRESS OF THE UCB
                     0179    678 ;
                     0179    679 ; OUTPUTS:
                     0179    680 ;
                     0179    681 ;       IRP$B_CARCON IS SET UP TO REFLECT THE PRE/SUF CHARACTERS TO SEND.
                     0179    682 ;
                     0179    683 ;       BYTE 0 = NUMBER OF CHARACTERS TO SEND
                     0179    684 ;       BYTE 1 = CHARACTER, IF 0 THEN NEWLINE
                     0179    685 ;
                     0179    686 ;       IRP$B_CARCON+2 HAS THE SUFFIX CONTROL.
                     0179    687 ;
                     0179    688 ;       R0,R1 ARE USED.
                     0179    689 ;
                     0179    690 ;-
                     0179    691 ;
                     0179    692 ; LOCAL DATA TABLE
                     0179    693 ;
                     0179    694 CCTABLE:                                    ; CARRIAGE CONTROL TO FORTRAN MATCH TABLE
0D 01 00 01          0179    695          .BYTE   1,0,1,13                   ; SPACE => 1 NL, 1 CR
       20            017D    696          .ASCII  / /
0D 01 00 02          017E    697          .BYTE   2,0,1,13                   ; "0" => 2 NL, 1 CR
       30            0182    698          .ASCII  /0/
0D 01 0C 01          0183    699          .BYTE   1,12,1,13                  ; "1" => 1 FF, 1 CR
       31            0187    700          .ASCII  /1/
0D 01 00 00          0188    701          .BYTE   0,0,1,13                   ; "+" => NOTHING, 1 CR
       2B            018C    702          .ASCII  /+/
00 00 00 01          018D    703          .BYTE   1,0,0,0                    ; "$" => 1 NL, NOTHING
       24            0191    704          .ASCII  /$/
```

```
                     0D 01 00 01  0192   705          .BYTE   1,0,1,13                ; DEFAULT => 1 NL, 1 CR
                        00        0196   706          .BYTE   0                       ; TABLE END
                                  0197   707 ;
                                  0197   708 ;
                                  0197   709 ;
                                  0197   710 EXE$CARRIAGE::                          ; INTERPRET CARRIAGE CONTROL
                  51    38 A3     9A   0197   711          MOVZBL  IRP$B_CARCON(R3),R1    ; GET FORTRAN SPECIFIER
                           12       13   019B   712          BEQL    20$                    ; IF EQL THEN TRY PRE/SUF
                  50    D9 AF     9E   019D   713          MOVAB   B^CCTABLE,R0           ; ADDRESS MATCH TABLE
                  38 A3    80     D0   01A1   714 10$:     MOVL    (R0)+,IRP$B_CARCON(R3) ; ASSUME MATCH
                           60     95   01A5   715          TSTB    (R0)                   ; END OF TABLE?
                           05     13   01A7   716          BEQL    15$                    ; IF EQL THEN YES
                        51 80     91   01A9   717          CMPB    (R0)+,R1               ; MATCH?
                           F3     12   01AC   718          BNEQ    10$                    ; NO THEN SEARCH
                           05     10   01AE   719 15$:     RSB                            ; ELSE RETURN
                                  01AF   720 ;
                                  01AF   721 ; PRE/SUF CARRIAGE CONTROL
                                  01AF   722 ;
                  51    3A A3     9A   01AF   723 20$:     MOVZBL  IRP$B_CARCON+2(R3),R1  ; GET PREFIX SPECIFIER
                           02       13   01B3   724          BEQL    30$                    ; IF EQL THEN NONE
                           19       10   01B5   725          BSBB    100$                   ; INTERPRET THE SPECIFIER
                  38 A3    51     90   01B7   726 30$:     MOVB    R1,IRP$B_CARCON(R3)    ; INSERT NUMBER
                  39 A3    50     90   01BB   727          MOVB    R0,IRP$B_CARCON+1(R3)  ; INSERT CHARACTER
                  51    3B A3     9A   01BF   728          MOVZBL  IRP$B_CARCON+3(R3),R1  ; GET SUFFIX SPECIFIER
                           02       13   01C3   729          BEQL    40$                    ; IF EQL THEN NONE
                           09       10   01C5   730          BSBB    100$                   ; CONVERT THE SPECIFIER
                  3A A3    51     90   01C7   731 40$:     MOVB    R1,IRP$B_CARCON+2(R3)  ; INSERT NUMBER
                  3B A3    50     90   01CB   732          MOVB    R0,IRP$B_CARCON+3(R3)  ; INSERT CHARACTER
                           05       10   01CF   733          RSB                            ; RETURN
                                  01D0   734 ;
                                  01D0   735 ; SUBROUTINE TO INTERPRET PRE/SUF SPECIFIER
                                  01D0   736 ;
                        50        D4   01D0   737 100$:     CLRL    R0                     ; ASSUME NEWLINE
                     08 51        07   E1   01D2   738          BBC     #7,R1,110$             ; IF BIT 7 CLEAR THEN DONE
                  50    51 E0     8F   8B   01D6   739          BICB3   #^X0E0,R1,R0           ; REMOVE OTHER BITS
                        51        01   9A   01DB   740          MOVZBL  #1,R1                  ; SET ONE CHARACTER
                                  05   01DE   741 110$:     RSB                            ; RETURN
                                  01DF   742
                                  01DF   743          .END
```

```
ACB$B_RMOD                      0000000B    D
ACB$B_TYPE                      0000000A    D
ACB$C_LENGTH                    00000018    D
ACB$K_LENGTH                    00000018    D
ACB$L_AST                       00000010    D
ACB$L_ASTPRM                    00000014    D
ACB$L_ASTQBL                    00000004    D
ACB$L_ASTQFL                    00000000    D
ACB$L_KAST                      00000018    D
ACB$L_PID                       0000000C    D
ACB$V_QUOTA                   = 00000006    D
ACB$W_SIZE                      00000008    D
ACCESS                          0000013A R  D   01
CCB$B_AMOD                      00000009    D
CCB$B_STS                       00000008    D
CCB$C_LENGTH                    00000010    D
CCB$K_LENGTH                    00000010    D
CCB$L_DIRP                      0000000C    D
CCB$L_UCB                       00000000    D
CCB$L_WIND                      00000004    D
CCB$W_IOC                       0000000A    D
CCTABLE                         00000179 R  D   01
EXE$ABORTIO                     ********  X     01
EXE$CARRIAGE                    00000197 RG D   01
EXE$C_CMSTKSZ                 = 00000014    D
EXE$DEANONPAGED                 ********  X     01
EXE$FINISHIO                    ********  X     01
EXE$MODIFY                      0000000C RG D   01
EXE$MODIFYLOCK                  00000044 RG D   01
EXE$MODIFYLOCKR                 00000047 RG D   01
EXE$ONEPARM                     00000000 RG D   01
EXE$QIODRVPKT                   ********    X   01
EXE$READ                        00000012 RG D   01
EXE$READCHK                     000000CE RG D   01
EXE$READCHKR                    000000E2 RG D   01
EXE$READLOCK                    0000003E RG D   01
EXE$READLOCKR                   00000051 RG D   01
EXE$SENSEMODE                   0000016F RG D   01
EXE$SETCHAR                     00000141 RG D   01
EXE$SETMODE                     0000015F RG D   01
EXE$WRITE                       0000001B RG D   01
EXE$WRITECHK                    000000D4 RG D   01
EXE$WRITECHKR                   0000010C RG D   01
EXE$WRITELOCK                   00000041 RG D   01
EXE$WRITELOCKR                  00000058 RG D   01
EXE$ZEROPARM                    00000006 RG D   01
IO$_PHYSICAL                  = 0000001F    D
IO$_READLBLK                  = 00000021    D
IO$_READPBLK                  = 0000000C    D
IO$_SETMODE                   = 00000023    D
IRP$B_CARCON                    00000038    D
IRP$B_EFN                       00000022    D
IRP$B_PRI                       00000023    D
IRP$B_RMOD                      0000000B    D
IRP$B_TYPE                      0000000A    D
IRP$C_LENGTH                    0000005C    D
IRP$K_LENGTH                    0000005C    D

IRP$L_ARB                       00000050    D
IRP$L_AST                       00000010    D
IRP$L_ASTPRM                    00000014    D
IRP$L_DIAGBUF                   00000044    D
IRP$L_EXTEND                    0000004C    D
IRP$L_IOQBL                     00000004    D
IRP$L_IOQFL                     00000000    D
IRP$L_IOSB                      00000024    D
IRP$L_IOST1                     00000034    D
IRP$L_IOST2                     00000038    D
IRP$L_MEDIA                     00000034    D
IRP$L_PID                       0000000C    D
IRP$L_SEGVBN                    00000040    D
IRP$L_SEQNUM                    00000048    D
IRP$L_SVAPTE                    0000002C    D
IRP$L_TT_TERM                   00000038    D
IRP$L_UCB                       0000001C    D
IRP$L_WIND                      00000018    D
IRP$M_FUNC                    = 00000002    D
IRP$Q_NT_PRVMSK                 0000003C    D
IRP$S_FCODE                   = 00000006    D
IRP$V_DIAGBUF                 = 00000007    D
IRP$V_FCODE                   = 00000000    D
IRP$V_FUNC                    = 00000001    D
IRP$W_ABCNT                     0000003C    D
IRP$W_BCNT                      00000032    D
IRP$W_BOFF                      00000030    D
IRP$W_CHAN                      09000028    D
IRP$W_FUNC                      00000020    D
IRP$W_OBCNT                     0000003E    D
IRP$W_SIZE                      00000008    D
IRP$W_STS                       0000002A    D
IRP$W_TT_PRMPT                  0000003C    D
MMG$IOLOCK                      ********  X     01
P1                            = 00000000    D
P2                            = 00000004    D
P3                            = 00000008    D
P4                            = 0000000C    D
P5                            = 00000010    D
P6                            = 00000014    D
PCB$B_ASTACT                    0000000C    D
PCB$B_ASTEN                     0000000D    D
PCB$B_PRI                       0000000B    D
PCB$B_PRIB                      0000002F    D
PCB$B_TYPE                      0000000A    D
PCB$B_WEFC                      0000002E    D
PCB$C_LENGTH                    0000008C    D
PCB$K_LENGTH                    0000008C    D
PCB$L_ARB                       00000084    D
PCB$L_ASTQBL                    00000014    D
PCB$L_ASTQFL                    00000010    D
PCB$L_EFC2P                     00000058    D
PCB$L_EFC3P                     0000005C    D
PCB$L_EFCS                      00000050    D
PCB$L_EFCU                      00000054    D
PCB$L_EFWM                      0000004C    D
PCB$L_JIB                       00000078    D
```

PCB$L_OWNER		0000001C	D	UCB$B_TT_DETYPE		000000A4	D
PCB$L_PHD		00000064	D	UCB$B_TT_LFFILL		0000009E	D
PCB$L_PHYPCB		00000018	D	UCB$B_TT_SPEED		0000009C	D
PCB$L_PID		00000060	D	UCB$B_TYPE		0000000A	D
PCB$L_PQB		0000004C	D	UCB$B_VERTSZ		0000003F	D
PCB$L_SQBL		00000004	D	UCB$C_LENGTH		00000074	D
PCB$L_SQFL		00000000	D	UCB$C_MB_LENGTH		00000090	D
PCB$L_STS		00000024	D	UCB$C_TT_LENGTH		000000BC	D
PCB$L_UIC		00000088	D	UCB$K_LENGTH		00000074	D
PCB$L_WSSWP		00000020	D	UCB$K_MB_LENGTH		00000090	D
PCB$L_WTIME		00000028	D	UCB$K_TT_LENGTH		000000BC	D
PCB$Q_PRIV		0000007C	D	UCB$L_AMB		00000054	D
PCB$T_LNAME		00000068	D	UCB$L_ASTQBL		00000010	D
PCB$T_TERMINAL		00000044	D	UCB$L_ASTQFL		0000000C	D
PCB$W_APTCNT		00000030	D	UCB$L_CPID		0000005C	D
PCB$W_ASTCNT		00000038	D	UCB$L_CRB		00000020	D
PCB$W_BIOCNT		0000003A	D	UCB$L_DDB		00000024	D
PCB$W_BIOLM		0000003C	D	UCB$L_DEVCHAR		00000034	D
PCB$W_DIOCNT		0000003E	D	UCB$L_DEVDEPEND		0000003C	D
PCB$W_DIOLM		00000040	D	UCB$L_DPC		00000080	D
PCB$W_GPGCNT		00000034	D	UCB$L_DUETIM		0000005C	D
PCB$W_GRP		0000008A	D	UCB$L_DX_BFPNT		0000009C	D
PCB$W_MEM		00000088	D	UCB$L_DX_BUF		00000098	D
PCB$W_MTXCNT		0000000E	D	UCB$L_DX_RXDB		000000A0	D
PCB$W_PPGCNT		00000036	D	UCB$L_EMB		00000078	D
PCB$W_PRCCNT		00000042	D	UCB$L_FIRST		00000014	D
PCB$W_SIZE		00000008	D	UCB$L_FPC		0000000C	D
PCB$W_STATE		0000002C	D	UCB$L_FQBL		00000004	D
PCB$W_TMBU		00000032	D	UCB$L_FQFL		00000000	D
SIZ...	=	00000002	D	UCB$L_FR3		00000010	D
SS$_ACCVIO	=	0000000C	D	UCB$L_FR4		00000014	D
SS$_NORMAL	=	00000001	D	UCB$L_IOQBL		00000044	D
UCB$B_AMOD		00000053	D	UCB$L_IOQFL		00000040	D
UCB$B_CEX		00000077	D	UCB$L_IRP		0000004C	D
UCB$B_CM1		0000004A	D	UCB$L_LINK		0000002C	D
UCB$B_CM2		0000004B	D	UCB$L_LOGADR		00000064	D
UCB$B_DEVCLASS		00000038	D	UCB$L_MAXBLOCK		00000084	D
UCB$B_DEVTYPE		00000039	D	UCB$L_MB_MBX		0000007C	D
UCB$B_DIPL		00000052	D	UCB$L_MB_PORT		0000008C	D
UCB$B_DX_SCTCNT		000000A6	D	UCB$L_MB_RAST		00000078	D
UCB$B_ERTCNT		00000070	D	UCB$L_MB_SMB		00000080	D
UCB$B_ERTMAX		00000071	D	UCB$L_MB_WAST		00000074	D
UCB$B_FEX		00000076	D	UCB$L_MB_WIOQBL		00000088	D
UCB$B_FIPL		0000000B	D	UCB$L_MB_WIOQFL		00000084	D
UCB$B_LOCSRV		0000003C	D	UCB$L_MEDIA		0000008C	D
UCB$B_OFFNDX		00000094	D	UCB$L_MT_DATSSB		00000074	D
UCB$B_OFFRTC		00000095	D	UCB$L_MT_INTSSB		00000078	D
UCB$B_REMSRV		0000003D	D	UCB$L_OPCNT		00000060	D
UCB$B_SECTORS		0000003C	D	UCB$L_OWNUIC		0000001C	D
UCB$B_SLAVE		00000074	D	UCB$L_PID		00000028	D
UCB$B_SPR		00000075	D	UCB$L_RQBL		00000004	D
UCB$B_STATE		00000052	D	UCB$L_RQFL		00000000	D
UCB$B_TRACKS		0000003D	D	UCB$L_SVAPTE		00000068	D
UCB$B_TT_CRFILL		0000009D	D	UCB$L_SVPN		00000064	D
UCB$B_TT_DECRF		000000A1	D	UCB$L_TT_DECHAR		000000A8	D
UCB$B_TT_DELFF		000000A2	D	UCB$L_TT_RDUE		0000008C	D
UCB$B_TT_DESPEE		000000A0	D	UCB$L_TT_RTIMOU		000000B8	D

```
UCB$L_VCB                    00000030    D
UCB$T_PARTNER                0000000C    D
UCB$W_BCNT                   0000006E    D
UCB$W_BCR                    00000096    D
UCB$W_BOFF                   0000006C    D
UCB$W_BUFQUO                 00000018    D
UCB$W_BYTESTOGO              0000003E    D
UCB$W_CHARGE                 0000004A    D
UCB$W_CYLINDERS              0000003E    D
UCB$W_DA                     0000008C    D
UCB$W_DC                     0000008E    D
UCB$W_DEVBUFSIZ              0000003A    D
UCB$W_DEVSTS                 0000005A    D
UCB$W_DIRSEQ                 00000088    D
UCB$W_DSTADDR                00000018    D
UCB$W_DX_BCR                 000000A4    D
UCB$W_EC1                    00000090    D
UCB$W_EC2                    00000092    D
UCB$W_ERRCNT                 00000072    D
UCB$W_FUNC                   0000007E    D
UCB$W_MB_SEED                00000000    D
UCB$W_MSGCNT                 00000016    D
UCB$W_MSGMAX                 00000014    D
UCB$W_NT_CHAN                0000007C    D
UCB$W_OFFSET                 0000008A    D
UCB$W_REFC                   0C000050    D
UCB$W_SIZE                   00000008    D
UCB$W_SRCADDR                0000001A    D
UCB$W_STS                    00000058    D
UCB$W_TT_DESIZE              000000A5    D
UCB$W_UNIT                   00000048    D
UCB$W_VPROT                  0000001A    D
VA$M_BYTE                  = 000001FF    D
```

```
                    +-----------------+
                    ! Psect synopsis !
                    +-----------------+

PSECT name                  Allocation          PSECT No.  Attributes
----------                  ----------          ---------  ----------
.  ABS  .                   00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
SEP                         000001DF  (  479.)  01 (  1.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD    WRT NOVEC LONG
$ABS$                       000000BC  (  188.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
ACB$V_QUOTA	=00000006		#-340 (1)
ACCESS	0000013A-R	525 (1)	#-470 (1) #-516 (1) #-563 (1) #-606 (1)
CCB$W_IOC	0000000A		#-333 (1)
CCTABLE	00000179-R	694 (1)	713 (1)
EXE$ABORTIO	00000000-XR		#-353 (1) #-426 (1)
EXE$CARRIAGE	00000197-R	710 (1)	
EXE$C_CMSTKSZ	=00000014	8 (1)	#-347 (1)
EXE$DEANONPAGED	00000000-XR		337 (1) 343 (1)
EXE$FINISHIO	00000000-XR		#-645 (1)
EXE$MODIFY	0000000C-R	191 (1)	
EXE$MODIFYLOCK	00000044-R	249 (1)	192 (1)
EXE$MODIFYLOCKR	00000047-R	304 (1)	#-250 (1)
EXE$ONEPARM	00000000-R	122 (1)	
EXE$QIODRVPKT	00000000-XR		#-156 (1) #-209 (1) #-608 (1)
EXE$READ	00000012-R	194 (1)	
EXE$READCHK	000000CE-R	387 (1)	
EXE$READCHKR	000000E2-R	464 (1)	#-306 (1) #-312 (1) #-389 (1)
EXE$READLOCK	0000003E-R	237 (1)	195 (1)
EXE$READLOCKR	00000051-R	310 (1)	#-238 (1)
EXE$SENSEMODE	0000016F-R	642 (1)	
EXE$SETCHAR	00000141-R	561 (1)	
EXE$SETMODE	0000015F-R	604 (1)	
EXE$WRITE	0000001B-R	197 (1)	
EXE$WRITECHK	000000D4-R	422 (1)	
EXE$WRITECHKR	0000010C-R	510 (1)	#-317 (1) #-424 (1)
EXE$WRITELOCK	00000041-R	243 (1)	198 (1)
EXE$WRITELOCKR	00000058-R	315 (1)	#-244 (1)
EXE$ZEROPARM	00000006-R	154 (1)	
IO$_PHYSICAL	=0000001F		#-201 (1)
IO$_READLBLK	=00000021		#-203 (1)
IO$_READPBLK	=0000000C		#-203 (1)
IO$_SETMODE	=00000023		#-564 (1)
IRP$B_CARCON	00000038		#-199 (1) #-711 (1) #-714 (1) #-723 (1)
			#-726 (1) #-727 (1) #-728 (1) #-731 (1)
			#-732 (1)
IRP$B_RMOD	0000000B		340 (1)
IRP$L_DIAGBUF	00000044		#-335 (1)
IRP$L_MEDIA	00000034		#-123 (1) #-155 (1) #-607 (1)
IRP$L_SVAPTE	0000002C		#-325 (1)
IRP$M_FUNC	=00000002		#-474 (1)
IRP$S_FCODE	=00000006		#-200 (1)
IRP$V_DIAGBUF	=00000007		#-334 (1)
IRP$V_FCODE	=00000000		#-200 (1)
IRP$V_FUNC	=00000001		#-196 (1)
IRP$W_BCNT	00000032		#-522 (1)
IRP$W_BOFF	00000030		#-320 (1)
IRP$W_FUNC	00000020		201 (1) #-204 (1)
IRP$W_STS	0000002A		196 (1) 334 (1) #-474 (1)
MMG$IOLOCK	00000000-XR		322 (1)
P1	=00000000	85 (1)	#-123 (1) #-207 (1) #-562 (1) #-605 (1)

```
P2                              =00000004     86      (1)     #-205     (1)
P3                              =00000008     87      (1)
P4                              =0000000C     88      (1)     #-199     (1)
P5                              =00000010     89      (1)
P6                              =00000014     90      (1)
PCB$W_ASTCNT                     00000038                     #-341     (1)
PCB$W_DIOCNT                     0000003E                     #-332     (1)
SS$_ACCVIO                      =0000000C                     #-527     (1)
SS$_NORMAL                      =00000001                     #-523     (1)     #-644     (1)
UCB$B_DEVCLASS                   00000038                     #-566     (1)
UCB$L_DEVDEPEND                  0000003C                     #-568     (1)     #-643     (1)
UCB$W_DEVBUFSIZ                  0000003A                     #-567     (1)
VA$M_BYTE                       =000001FF                     #-320     (1)     #-467     (1)     #-513     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	69 (1)	69 (1)
$CCBDEF	1	70 (1)	70 (1)
$DEFINI	1	69 (1)	69 (1) 70 (1) 71 (1) 72 (1) 73 (1) 74 (1) 75 (1) 76 (1) 77 (1)
$DEVDEF	1	71 (1)	71 (1)
$IODEF	15	72 (1)	72 (1)
$IRPDEF	4	73 (1)	73 (1)
$PCBDEF	4	74 (1)	74 (1)
$SSDEF	21	75 (1)	75 (1)
$UCBDEF	10	76 (1)	76 (1)
$VADEF	1	77 (1)	77 (1)
IFNORD	1	516 (1)	516 (1) 563 (1) 606 (1)
IFNOWRT	1	470 (1)	470 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	327 (1) 347 (1) 466 (1) 512 (1)
BBC	00E1	334 (1) 340 (1) 738 (1)
BBCS	00E3	196 (1)
BEQL	0013	206 (1) 470 (1) 516 (1) 563 (1) 565 (1) 606 (1) 712 (1)
		716 (1) 724 (1) 729 (1)
BGTR	0014	473 (1) 519 (1)
BICB3	008B	739 (1)
BICW	00AA	467 (1) 513 (1)
BICW3	00AB	320 (1)
BISL	00C8	307 (1)
BISW	00A8	474 (1)
BLBC	00E9	318 (1) 324 (1)
BLBS	00E8	425 (1)
BLEQ	0015	202 (1)
BNEQ	0012	330 (1) 718 (1)
BRB	0011	124 (1) 193 (1) 308 (1) 313 (1) 390 (1) 476 (1) 569 (1)
BRW	0031	156 (1) 209 (1) 353 (1) 426 (1) 608 (1) 645 (1)
BSBB	0010	238 (1) 244 (1) 250 (1) 389 (1) 424 (1) 725 (1) 730 (1)
BSBW	0030	306 (1) 312 (1) 317 (1)
CLRL	00D4	155 (1) 520 (1) 737 (1)
CMPB	0091	717 (1)
CMPL	00D1	564 (1)
CMPZV	00ED	200 (1)
CVTWL	0032	469 (1) 515 (1)
DECW	00B7	333 (1)
INCW	00B6	332 (1) 341 (1)
JMP	0017	352 (1)
JSB	0016	208 (1) 322 (1) 328 (1) 337 (1) 343 (1)
MOVAB	009E	713 (1)
MOVAL	00DE	192 (1) 195 (1) 198 (1)
MOVAW	003E	472 (1) 518 (1)
MOVB	0090	726 (1) 727 (1) 731 (1) 732 (1)
MOVL	00D0	123 (1) 199 (1) 207 (1) 323 (1) 325 (1) 335 (1) 342 (1)
		345 (1) 475 (1) 562 (1) 568 (1) 605 (1) 643 (1) 714 (1)
MOVQ	007D	346 (1) 607 (1)
MOVW	00B0	522 (1) 566 (1) 567 (1)
MOVZBL	009A	711 (1) 723 (1) 728 (1) 740 (1)
MOVZWL	003C	205 (1) 523 (1) 527 (1) 644 (1)
POPL	8ED0	319 (1) 338 (1) 427 (1) 521 (1) 526 (1)
POPR	00BA	344 (1)
PROBER	000C	516 (1) 563 (1) 606 (1)
PROBEW	000D	470 (1)
PUSHAB	009F	349 (1)
PUSHL	00DD	305 (1) 311 (1) 316 (1) 321 (1) 331 (1) 336 (1) 388 (1)
		423 (1) 465 (1) 511 (1)
REI	0002	350 (1)
RSB	0005	240 (1) 246 (1) 252 (1) 326 (1) 428 (1) 524 (1) 528 (1)
		719 (1) 733 (1) 741 (1)
SUBL	00C2	468 (1) 471 (1) 514 (1) 517 (1)
SUBL3	00C3	348 (1)

```
SUBW            00A2      203    (1)
TSTB            0095      351    (1)    715    (1)
TSTL            00D5      329    (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCII	696	(1)	698	(1)	700	(1)	702	(1)	704	(1)						
.BYTE	695	(1)	697	(1)	699	(1)	701	(1)	703	(1)	705	(1)	706	(1)		
.DSABL	157	(1)	210	(1)	354	(1)	429	(1)	529	(1)	646	(1)				
.ENABL	121	(1)	190	(1)	303	(1)	386	(1)	463	(1)	560	(1)				
.END	743	(1)														
.ENDM	470	(1)	516	(1)	[illegible]9	(1)	70	(1)	71	(1)	72	(1)	73	(1)	74	(1)
	75	(1)	76	(1)	77	(1)										
.IDENT	5	(1)														
.NOCROSS	69	(1)	70	(1)	71	(1)	72	(1)	73	(1)	74	(1)	75	(1)	76	(1)
	77	(1)														
.PAGE	125	(1)	158	(1)	211	(1)	254	(1)	355	(1)	391	(1)	430	(1)	477	(1)
	530	(1)	570	(1)	609	(1)	647	(1)	91	(1)	92	(1)				
.PSECT	6	(1)														
.RESTORE	69	(1)	70	(1)	71	(1)	72	(1)	73	(1)	74	(1)	75	(1)	76	(1)
	77	(1)														
.SAVE	69	(1)	70	(1)	71	(1)	72	(1)	73	(1)	74	(1)	75	(1)	76	(1)
	77	(1)														
.SBTTL	126	(1)	159	(1)	212	(1)	255	(1)	356	(1)	392	(1)	431	(1)	478	(1)
	531	(1)	571	(1)	610	(1)	648	(1)	93	(1)						
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	7	#-123	(1)	#-199	(1)	#-205	(1)	#-207	(1)	#-346	(1)	#-562	(1)
		#-605	(1)										
FP	2	#-345	(1)	#-346	(1)								
R0	38	#-207	(1)	#-305	(1)	#-311	(1)	#-316	(1)	#-318	(1)	#-319	(1)
		#-320	(1)	#-324	(1)	#-329	(1)	#-335	(1)	#-342	(1)	#-348	(1)
		352	(1)	#-388	(1)	#-423	(1)	#-425	(1)	#-427	(1)	#-466	(1)
		#-467	(1)	#-468	(1)	470	(1)	#-471	(1)	#-512	(1)	#-513	(1)
		#-514	(1)	516	(1)	#-517	(1)	#-523	(1)	#-527	(1)	#-644	(1)
		#-713	(1)	#-714	(1)	#-715	(1)	#-717	(1)	#-727	(1)	#-732	(1)
		#-737	(1)	#-739	(1)								
R1	38	#-205	(1)	#-325	(1)	#-331	(1)	#-344	(1)	#-351	(1)	#-465	(1)
		#-466	(1)	#-468	(1)	#-470	(1)	472	(1)	#-511	(1)	#-512	(1)
		#-514	(1)	#-516	(1)	518	(1)	#-521	(1)	#-522	(1)	#-526	(1)
		#-562	(1)	563	(1)	#-566	(1)	#-567	(1)	#-568	(1)	#-605	(1)
		606	(1)	#-607	(1)	#-643	(1)	#-711	(1)	#-717	(1)	#-723	(1)
		#-726	(1)	#-728	(1)	#-731	(1)	738	(1)	#-739	(1)	#-740	(1)
R2	13	#-192	(1)	#-195	(1)	#-198	(1)	208	(1)	#-307	(1)	#-469	(1)
		#-471	(1)	472	(1)	#-475	(1)	#-515	(1)	#-517	(1)	518	(1)
		#-520	(1)										
R3	27	#-123	(1)	#-155	(1)	196	(1)	#-199	(1)	201	(1)	#-204	(1)
		#-320	(1)	#-321	(1)	#-323	(1)	#-325	(1)	334	(1)	#-335	(1)
		#-336	(1)	#-338	(1)	340	(1)	#-342	(1)	#-474	(1)	#-522	(1)
		#-607	(1)	#-711	(1)	#-714	(1)	#-723	(1)	#-726	(1)	#-727	(1)
		#-728	(1)	#-731	(1)	#-732	(1)						
R4	2	#-332	(1)	#-341	(1)								
R5	4	#-566	(1)	#-567	(1)	#-568	(1)	#-643	(1)				
R6	1	#-333	(1)										
R7	1	#-564	(1)										
SP	6	#-323	(1)	#-327	(1)	328	(1)	#-345	(1)	#-347	(1)	#-348	(1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	26	00:00:00.02	00:00:00.20
Command processing	878	00:00:00.23	00:00:00.72
Pass 1	763	00:00:02.93	00:00:04.23
Symbol table sort	0	00:00:00.34	00:00:00.33
Pass 2	73	00:00:00.88	00:00:01.83
Symbol table output	2	00:00:00.03	00:00:00.18
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	6	00:00:00.11	00:00:00.22
Assembler run totals	1751	00:00:04.55	00:00:07.72

The working set limit was 2900 pages.
166703 bytes (326 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1171 non-local and 26 local symbols.
743 source lines were read in Pass 1, producing 76 object records in Pass 2.

57 pages of virtual memory were used to define 20 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	6
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	3
SYS$COMMON:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	16

1437 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:QIOFDT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 81 DECLARATIONS
(1) 137 QUEUE I/O REQUEST
(1) 385 COMPLETE I/O OPERATION
(1) 437 QUEUE I/O PACKET TO DRIVER
(1) 460 EXE$ALTQUEPKT - Call driver ALTSTART entry point
(1) 505 QUEUE I/O PACKET TO ACP
(1) 535 INSERT I/O PACKET IN UNIT QUEUE
(1) 558 INSERT I/O PACKET IN QUEUE BY PRIORITY

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element QIOREQ.MAR
0000     2 ; *1     6-FEB-1986 15:22:13 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element QIOREQ.MAR
0000     4         .TITLE  QIOREQ  *** QIOREQ handle QIO request
0000     5         .IDENT  /01-02/
0000     6         .NLIST  CND
0000     7
0000     8 ;
0000     9 ;*****************************************************************************
0000    10 ;*                                                                           *
0000    11 ; COPYRIGHT (C) 1977, 1980
0000    12 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
0000    13 ;*                                                                           *
0000    14 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
0000    15 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
0000    16 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
0000    17 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    18 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
0000    19 ;*  TRANSFERRED.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
0000    22 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    23 ;*  CORPORATION.                                                             *
0000    24 ;*                                                                           *
0000    25 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
0000    26 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    27 ;*                                                                           *
0000    28 ;*****************************************************************************
0000    29 ;
0000    30 ;++
0000    31 ;
0000    32 ; AUTHOR:
0000    33 ;
0000    34 ;       D. N. CUTLER,   14-JUN-76
0000    35 ;
0000    36 ; FACILITY:
0000    37 ;
0000    38 ;       SYSTEM SERVICE QUEUE I/O REQUEST
0000    39 ;
0000    40 ; MODIFIED BY:
0000    41 ;
0000    42 ;       DS5.4   John Ciukaj     5-May-1980      Version 5.4
0000    43 ;                 Used Vax/Vms version 2.0 SYSQIOREQ.MAR and created
0000    44 ;                 a UPD file with the edits necessary to reflect the
0000    45 ;                 changes exhibited in Supervisor version 5.3 module
0000    46 ;                 QIOREQ.
0000    47 ;                Dave Butenhof   30-jun-1980,    version 5.5
0000    48 ;       02          Alter some word displacement BSB's to longword displacement
0000    49 ;                   JSB's, since supervisor has expanded.
0000    50 ;
0000    51 ;
0000    52 ;       V0208   LMK0001         LEN KAWELL              03-OCT-1979
0000    53 ;                DISALLOW LOGICAL/PHYSICAL I/O TO A SPOOLED DEVICE UNLESS
0000    54 ;                THE PROCESS HAS LOG_IO PRIVILEGE.
0000    55 ;
0000    56 ;       V0207   KDM0053         Kathleen D. Morse       31-Aug-1979
0000    57 ;                ADD TWO NEW ENTRY POINTS FOR SHARED MEMORY GLOBAL SECTION
```

```
                        0000     58 ;                  I/O, EXE$BLDPKTGSR AND EXE$BLDPKTGSW.
                        0000     59 ;
                        0000     60 ;          V0006   CHP0002         CAROL PETERS            14-AUG-1979
                        0000     61 ;                  IN EXE$ALTQUEPKT, IF NO ALTERNATE START I/O ADDRESS
                        0000     62 ;                  IS SPECIFIED IN THE DDR, JUST RETURN TO CALLER.
                        0000     63 ;
                        0000     64 ;          V0005   ACG0047         ANDREW C. GOLDSTEIN     09-AUG-1979
                        0000     65 ;                  ADD ACCESS RIGHTS BLOCK, PROTECTION INTERFACE CHANGES.
                        0000     66 ;
                        0000     67 ;          V04     CHP0001         CAROL PETERS            03-AUG-1979
                        0000     68 ;                  ADDED NEW ENTRY POINT FOR QUEUING A PACKET TO A DRIVER.
                        0000     69 ;                  ENTRY POINT IS STORED IN DDT AND CALLED ALTSTART.  ADDED
                        0000     70 ;                  SYSQIOREQ ENTRY POINT, EXE$ALTQUEPKT, WHICH JSBS TO
                        0000     71 ;                  THE ALTSTART ENTRY POIN, AND THEN RSBS TO ITS CALLER.
                        0000     72 ;
                        0000     73 ;          V03     RIH23909        R. I. HUSTVEDT          14-MAY-1979
                        0000     74 ;                  REMOVE SUPERFLUOUS BRANCH FOLLOWING CALL TO SCH$CLREF.
                        0000     75 ;
                        0000     76 ;          V02     PHL0002         P. H. LIPMAN            01-MAY-1978
                        0000     77 ;                  REWROTE BUILDPKT FOR SEGMENTED VIRTUAL I/O.
                        0000     78 ;
                        0000     79 ;--
```

```
                        0000     81          .SBTTL  DECLARATIONS
                        0000     82
                        0000     83 ;
                        0000     84 ; MACRO LIBRARY CALLS
                        0000     85 ;
                        0000     86
                        0000     87          $ACBDEF                             ;DEFINE ACB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .IIF    NB,ACB$L_ASTQFL,        ACB$L_ASTQFL:
00000004                0000                         .BLKL   1
                        0004                 .IIF    NB,ACB$L_ASTQBL,        ACB$L_ASTQBL:
00000008                0004                         .BLKL   1
                        0008                 .IIF    NB,ACB$W_SIZE,  ACB$W_SIZE:
0000000A                0008                         .BLKW   1
                        000A                 .IIF    NB,ACB$B_TYPE,  ACB$B_TYPE:
0000000B                000A                         .BLKB   1
                        000B                 .IIF    NB,ACB$B_RMOD,  ACB$B_RMOD:
0000000C                000B                         .BLKB   1
                        000C                 .IIF    NB,ACB$L_PID,   ACB$L_PID:
00000010                000C                         .BLKL   1
                        0010                 .IIF    NB,ACB$L_AST,   ACB$L_AST:
00000014                0010                         .BLKL   1
                        0014                 .IIF    NB,ACB$L_ASTPRM,        ACB$L_ASTPRM:
00000018                0014                         .BLKL   1
                        0018                 .IIF    NB,ACB$C_LENGTH,        ACB$C_LENGTH:
                        0018                 .IIF    NB,ACB$K_LENGTH,        ACB$K_LENGTH:
                        0018                 .IIF    NB,ACB$L_KAST,  ACB$L_KAST:
0000001C                0018                         .BLKL   1
                        0000                 .RESTORE
                        0000     88          $AQBDEF                             ;DEFINE AQB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                001C                 .=0
                        0000                 .RESTORE
                        0000     89          $CADEF                              ;DEFINE CONDITIONAL ASSEMBLY PARAMETERS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                001C                 .=0
                        0000                 .RESTORE
                        0000     90          $CCBDEF                             ;DEFINE CCB OFFSETS
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
00000000                001C                 .=0
                        0000                 .IIF    NB,CCB$L_UCB,   CCB$L_UCB:
00000004                0000                         .BLKL   1
                        0004                 .IIF    NB,CCB$L_WIND,  CCB$L_WIND:
00000008                0004                         .BLKL   1
                        0008                 .IIF    NB,CCB$B_STS,   CCB$B_STS:
00000009                0008                         .BLKB   1
                        0009                 .IIF    NB,CCB$B_AMOD,  CCB$B_AMOD:
0000000A                0009                         .BLKB   1
                        000A                 .IIF    NB,CCB$W_IOC,   CCB$W_IOC:
0000000C                000A                         .BLKW   1
                        000C                 .IIF    NB,CCB$L_DIRP,  CCB$L_DIRP:
00000010                000C                         .BLKL   1
                        0010                 .IIF    NB,CCB$C_LENGTH,        CCB$C_LENGTH:
                        0010                 .IIF    NB,CCB$K_LENGTH,        CCB$K_LENGTH:
```

```
                  0000          .RESTORE
                  0000    91    $DDBDEF                          ;DEFINE DDB OFFSETS
                  0000          .SAVE   LOCAL_BLOCK
                  0000          .PSECT  $ABS$,ABS
        00000000  001C          .=0
                  0000          .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
        00000004  0000                          .BLKL   1
                  0004          .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
        00000008  0004                          .BLKL   1
                  0008          .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
        0000000A  0008                          .BLKW   1
                  000A          .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
        0000000B  000A                          .BLKB   1
        0000000C  000B                          .BLKB   1
                  000C          .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
        00000010  000C                          .BLKL   1
                  0010          .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
        00000013  0010                          .BLKB   3
                  0013          .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
        00000014  0013                          .BLKB   1
                  0014          .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
        00000015  0014                          .BLKB   1
        00000024  0015                          .BLKB   15
                  0024          .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
        00000025  0024                          .BLKB   1
        00000034  0025                          .BLKB   15
                  0034          .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                  0034          .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
                  0000          .RESTORE
                  0000    92    $DDTDEF                          ;DEFINE DDT OFFSETS
                  0000          .SAVE   LOCAL_BLOCK
                  0000          .PSECT  $ABS$,ABS
        00000000  0034          .=0
                  0000          .IIF    NB,DDT$L_START, DDT$L_START:
        00000004  0000                          .BLKL   1
                  0004          .IIF    NB,DDT$L_UNSOLINT,      DDT$L_UNSOLINT:
        00000008  0004                          .BLKL   1
                  0008          .IIF    NB,DDT$L_FDT,   DDT$L_FDT:
        0000000C  0008                          .BLKL   1
                  000C          .IIF    NB,DDT$L_CANCEL,        DDT$L_CANCEL:
        00000010  000C                          .BLKL   1
                  0010          .IIF    NB,DDT$L_REGDUMP,       DDT$L_REGDUMP:
        00000014  0010                          .BLKL   1
                  0014          .IIF    NB,DDT$W_DIAGBUF,       DDT$W_DIAGBUF:
        00000016  0014                          .BLKW   1
                  0016          .IIF    NB,DDT$W_ERRORBUF,      DDT$W_ERRORBUF:
        00000018  0016                          .BLKW   1
                  0018          .IIF    NB,DDT$L_UNITINIT,      DDT$L_UNITINIT:
        0000001C  0018                          .BLKL   1
                  001C          .IIF    NB,DDT$L_ALTSTART,      DDT$L_ALTSTART:
        00000020  001C                          .BLKL   1
                  0000          .RESTORE
                  0000    93    $DYNDEF                          ;DEFINE DATA STRUCTURE TYPE CODES
                  0000          .SAVE   LOCAL_BLOCK
                  0000          .PSECT  $ABS$,ABS
        00000000  0034          .=0
                  0000          .RESTORE
```

```
                        0000    94          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 0034               .=0
                        0000                .RESTORE
                        0000    95          $IRPDEF                         ;DEFINE IRP OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 0034               .=0
                        0000                .IIF    NB,IRP$L_IOQFL, IRP$L_IOQFL:
                00000004 0000                       .BLKL   1
                        0004                .IIF    NB,IRP$L_IOQBL, IRP$L_IOQBL:
                00000008 0004                       .BLKL   1
                        0008                .IIF    NB,IRP$W_SIZE,  IRP$W_SIZE:
                0000000A 0008                       .BLKW   1
                        000A                .IIF    NB,IRP$B_TYPE,  IRP$B_TYPE:
                0000000B 000A                       .BLKB   1
                        000B                .IIF    NB,IRP$B_RMOD,  IRP$B_RMOD:
                0000000C 000B                       .BLKB   1
                        000C                .IIF    NB,IRP$L_PID,   IRP$L_PID:
                00000010 000C                       .BLKL   1
                        0010                .IIF    NB,IRP$L_AST,   IRP$L_AST:
                00000014 0010                       .BLKL   1
                        0014                .IIF    NB,IRP$L_ASTPRM,        IRP$L_ASTPRM:
                00000018 0014                       .BLKL   1
                        0018                .IIF    NB,IRP$L_WIND,  IRP$L_WIND:
                0000001C 0018                       .BLKL   1
                        001C                .IIF    NB,IRP$L_UCB,   IRP$L_UCB:
                00000020 001C                       .BLKL   1
                        0020                .IIF    NB,IRP$W_FUNC,  IRP$W_FUNC:
                00000022 0020                       .BLKW   1
                        0022                .IIF    NB,IRP$B_EFN,   IRP$B_EFN:
                00000023 0022                       .BLKB   1
                        0023                .IIF    NB,IRP$B_PRI,   IRP$B_PRI:
                00000024 0023                       .BLKB   1
                        0024                .IIF    NB,IRP$L_IOSB,  IRP$L_IOSB:
                00000028 0024                       .BLKL   1
                        0028                .IIF    NB,IRP$W_CHAN,  IRP$W_CHAN:
                0000002A 0028                       .BLKW   1
                        002A                .IIF    NB,IRP$W_STS,   IRP$W_STS:
                0000002C 002A                       .BLKW   1
                        002C                .IIF    NB,IRP$L_SVAPTE,        IRP$L_SVAPTE:
                00000030 002C                       .BLKL   1
                        0030                .IIF    NB,IRP$W_BOFF,  IRP$W_BOFF:
                00000032 0030                       .BLKW   1
                        0032                .IIF    NB,IRP$W_BCNT,  IRP$W_BCNT:
                00000034 0032                       .BLKW   1
                        0034                .IIF    NB,IRP$L_IOST1, IRP$L_IOST1:
                        0034                .IIF    NB,IRP$L_MEDIA, IRP$L_MEDIA:
                00000038 0034                       .BLKL   1
                        0038                .IIF    NB,IRP$L_IOST2, IRP$L_IOST2:
                        0038                .IIF    NB,IRP$L_TT_TERM,       IRP$L_TT_TERM:
                        0038                .IIF    NB,IRP$B_CARCON,        IRP$B_CARCON:
                00000039 0038                       .BLKB   1
                0000003C 0039                       .BLKB   3
                        003C                .IIF    NB,IRP$Q_NT_PRVMSK,     IRP$Q_NT_PRVMSK:
                        003C                .IIF    NB,IRP$W_ABCNT, IRP$W_ABCNT:
```

```
                    003C                .IIF    NB,IRP$W_TT_PRMPT,      IRP$W_TT_PRMPT:
0000003E  003C                                      .BLKW   1
                    003E                .IIF    NB,IRP$W_OBCNT, IRP$W_OBCNT:
00000040  003E                                      .BLKW   1
                    0040                .IIF    NB,IRP$L_SEGVBN,        IRP$L_SEGVBN:
00000044  0040                                      .BLKL   1
                    0044                .IIF    NB,IRP$L_DIAGBUF,       IRP$L_DIAGBUF:
00000048  0044                                      .BLKL   1
                    0048                .IIF    NB,IRP$L_SEQNUM,        IRP$L_SEQNUM:
0000004C  0048                                      .BLKL   1
                    004C                .IIF    NB,IRP$L_EXTEND,        IRP$L_EXTEND:
00000050  004C                                      .BLKL   1
                    0050                .IIF    NB,IRP$L_ARB,   IRP$L_ARB:
00000054  0050                                      .BLKL   1
00000058  0054                                      .BLKL   1
0000005C  0058                                      .BLKL   1
                    005C                .IIF    NB,IRP$C_LENGTH,        IRP$C_LENGTH:
                    005C                .IIF    NB,IRP$K_LENGTH,        IRP$K_LENGTH:
                    0000                .RESTORE
                    0000    96          $PCBDEF                         ;DEFINE PCB OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
00000000  005C                          .=0
                    0000                .IIF    NB,PCB$L_SQFL,  PCB$L_SQFL:
00000004  0000                                      .BLKL   1
                    0004                .IIF    NB,PCB$L_SQBL,  PCB$L_SQBL:
00000008  0004                                      .BLKL   1
                    0008                .IIF    NB,PCB$W_SIZE,  PCB$W_SIZE:
0000000A  0008                                      .BLKW   1
                    000A                .IIF    NB,PCB$B_TYPE,  PCB$B_TYPE:
0000000B  000A                                      .BLKB   1
                    000B                .IIF    NB,PCB$B_PRI,   PCB$B_PRI:
0000000C  000B                                      .BLKB   1
                    000C                .IIF    NB,PCB$B_ASTACT,        PCB$B_ASTACT:
0000000D  000C                                      .BLKB   1
                    000D                .IIF    NB,PCB$B_ASTEN, PCB$B_ASTEN:
0000000E  000D                                      .BLKB   1
                    000E                .IIF    NB,PCB$W_MTXCNT,        PCB$W_MTXCNT:
00000010  000E                                      .BLKW   1
                    0010                .IIF    NB,PCB$L_ASTQFL,        PCB$L_ASTQFL:
00000014  0010                                      .BLKL   1
                    0014                .IIF    NB,PCB$L_ASTQBL,        PCB$L_ASTQBL:
00000018  0014                                      .BLKL   1
                    0018                .IIF    NB,PCB$L_PHYPCB,        PCB$L_PHYPCB:
0000001C  0018                                      .BLKL   1
                    001C                .IIF    NB,PCB$L_OWNER, PCB$L_OWNER:
00000020  001C                                      .BLKL   1
                    0020                .IIF    NB,PCB$L_WSSWP, PCB$L_WSSWP:
00000024  0020                                      .BLKL   1
                    0024                .IIF    NB,PCB$L_STS,   PCB$L_STS:
00000028  0024                                      .BLKL   1
                    0028                .IIF    NB,PCB$L_WTIME, PCB$L_WTIME:
0000002C  0028                                      .BLKL   1
                    002C                .IIF    NB,PCB$W_STATE, PCB$W_STATE:
0000002E  002C                                      .BLKW   1
                    002E                .IIF    NB,PCB$B_WEFC,  PCB$B_WEFC:
0000002F  002E                                      .BLKB   1
```

```
                    002F            .IIF    NB,PCB$B_PRIB,  PCB$B_PRIB:
00000030  002F                              .BLKB   1
                    0030            .IIF    NB,PCB$W_APTCNT,        PCB$W_APTCNT:
00000032  0030                              .BLKW   1
                    0032            .IIF    NB,PCB$W_TMBU,  PCB$W_TMBU:
00000034  0032                              .BLKW   1
                    0034            .IIF    NB,PCB$W_GPGCNT,        PCB$W_GPGCNT:
00000036  0034                              .BLKW   1
                    0036            .IIF    NB,PCB$W_PPGCNT,        PCB$W_PPGCNT:
00000038  0036                              .BLKW   1
                    0038            .IIF    NB,PCB$W_ASTCNT,        PCB$W_ASTCNT:
0000003A  0038                              .BLKW   1
                    003A            .IIF    NB,PCB$W_BIOCNT,        PCB$W_BIOCNT:
0000003C  003A                              .BLKW   1
                    003C            .IIF    NB,PCB$W_BIOLM, PCB$W_BIOLM:
0000003E  003C                              .BLKW   1
                    003E            .IIF    NB,PCB$W_DIOCNT,        PCB$W_DIOCNT:
00000040  003E                              .BLKW   1
                    0040            .IIF    NB,PCB$W_DIOLM, PCB$W_DIOLM:
00000042  0040                              .BLKW   1
                    0042            .IIF    NB,PCB$W_PRCCNT,        PCB$W_PRCCNT:
00000044  0042                              .BLKW   1
                    0044            .IIF    NB,PCB$T_TERMINAL,      PCB$T_TERMINAL:
0000004C  0044                              .BLKB   8
                    004C            .IIF    NB,PCB$L_PQB,   PCB$L_PQB:
                    004C            .IIF    NB,PCB$L_EFWM,  PCB$L_EFWM:
00000050  004C                              .BLKL   1
                    0050            .IIF    NB,PCB$L_EFCS,  PCB$L_EFCS:
00000054  0050                              .BLKL   1
                    0054            .IIF    NB,PCB$L_EFCU,  PCB$L_EFCU:
00000058  0054                              .BLKL   1
                    0058            .IIF    NB,PCB$L_EFC2P, PCB$L_EFC2P:
0000005C  0058                              .BLKL   1
                    005C            .IIF    NB,PCB$L_EFC3P, PCB$L_EFC3P:
00000060  005C                              .BLKL   1
                    0060            .IIF    NB,PCB$L_PID,   PCB$L_PID:
00000064  0060                              .BLKL   1
                    0064            .IIF    NB,PCB$L_PHD,   PCB$L_PHD:
00000068  0064                              .BLKL   1
                    0068            .IIF    NB,PCB$T_LNAME, PCB$T_LNAME:
00000078  0068                              .BLKB   16
                    0078            .IIF    NB,PCB$L_JIB,   PCB$L_JIB:
0000007C  0078                              .BLKL   1
                    007C            .IIF    NB,PCB$Q_PRIV,  PCB$Q_PRIV:
00000084  007C                              .BLKQ   1
                    0084            .IIF    NB,PCB$L_ARB,   PCB$L_ARB:
00000088  0084                              .BLKL   1
                    0088            .IIF    NB,PCB$L_UIC,   PCB$L_UIC:
                    0088            .IIF    NB,PCB$W_MEM,   PCB$W_MEM:
0000008A  0088                              .BLKW   1
                    008A            .IIF    NB,PCB$W_GRP,   PCB$W_GRP:
0000008C  008A                              .BLKW   1
                    008C            .IIF    NB,PCB$C_LENGTH,        PCB$C_LENGTH:
                    008C            .IIF    NB,PCB$K_LENGTH,        PCB$K_LENGTH:
                    0000            .RESTORE
                    0000     97     $PHDDEF                         ;DEFINE PHD OFFSETS
                    0000            .SAVE   LOCAL_BLOCK
```

```
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .RESTORE
                        0000     98         $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .RESTORE
                        0000     99         $PRIDEF                         ;DEFINE PRIORITY CLASS INCREMENTS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .RESTORE
                        0000    100         $PRVDEF                         ;DEFINE PRIVILEGE BITS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .RESTORE
                        0000    101         $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .RESTORE
                        0000    102         $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .RESTORE
                        0000    103         $SECDEF                         ;DEFINE SEC OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .RESTORE
                        0000    104         $UCBDEF                         ;DEFINE UCB OFFSETS
                        0000                .SAVE   LOCAL_BLOCK
                        0000                .PSECT  $ABS$,ABS
                00000000 008C               .=0
                        0000                .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
                        0000                .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
                00000004 0000                       .BLKL   1
                        0004                .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
                        0004                .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
                00000008 0004                       .BLKL   1
                        0008                .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
                0000000A 0008                       .BLKW   1
                        000A                .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
                0000000B 000A                       .BLKB   1
                        000B                .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
                0000000C 000B                       .BLKB   1
                        000C                .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                        000C                .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                        000C                .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
                0000000D 000C                       .BLKB   1
                00000010 000D                       .BLKB   3
                        0010                .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                        0010                .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
                00000014 0010                       .BLKL   1
```

```
                        0014            .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                        0014            .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                        0014            .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
            00000016    0014                            .BLKW   1
                        0016            .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
            00000018    0016                            .BLKW   1
                        0018            .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                        0018            .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
            0000001A    0018                            .BLKW   1
                        001A            .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                        001A            .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
            0000001C    001A                            .BLKW   1
                        001C            .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
            00000020    001C                            .BLKL   1
                        0020            .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
            00000024    0020                            .BLKL   1
                        0024            .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
            00000028    0024                            .BLKL   1
                        0028            .IIF    NB,UCB$L_PID,   UCB$L_PID:
            0000002C    0028                            .BLKL   1
                        002C            .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
            00000030    002C                            .BLKL   1
                        0030            .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
            00000034    0030                            .BLKL   1
                        0034            .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
            00000038    0034                            .BLKL   1
                        0038            .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
            00000039    0038                            .BLKB   1
                        0039            .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
            0000003A    0039                            .BLKB   1
                        003A            .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
            0000003C    003A                            .BLKW   1
                        003C            .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                        003C            .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                        003C            .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
            0000003D    003C                            .BLKB   1
                        003D            .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                        003D            .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
            0000003E    003D                            .BLKB   1
                        003E            .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                        003E            .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
            0000003F    003E                            .BLKB   1
                        003F            .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
            00000040    003F                            .BLKB   1
                        0040            .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
            00000044    0040                            .BLKL   1
                        0044            .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
            00000048    0044                            .BLKL   1
                        0048            .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
            0000004A    0048                            .BLKW   1
                        004A            .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
                        004A            .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
            0000004B    004A                            .BLKB   1
                        004B            .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
            0000004C    004B                            .BLKB   1
                        004C            .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
            00000050    004C                            .BLKL   1
```

```
                        0050                .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052                0050                        .BLKW   1
                        0052                .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
                        0052                .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053                0052                        .BLKB   1
                        0053                .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054                0053                        .BLKB   1
                        0054                .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058                0054                        .BLKL   1
                        0058                .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A                0058                        .BLKW   1
                        005A                .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C                005A                        .BLKW   1
                        005C                .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
                        005C                .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060                005C                        .BLKL   1
                        0060                .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064                0060                        .BLKL   1
                        0064                .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
                        0064                .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068                0064                        .BLKL   1
                        0068                .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C                0068                        .BLKL   1
                        006C                .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E                006C                        .BLKW   1
                        006E                .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070                006E                        .BLKW   1
                        0070                .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071                0070                        .BLKB   1
                        0071                .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072                0071                        .BLKB   1
                        0072                .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074                0072                        .BLKW   1
                        0074                .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
                        0074                .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000                0074                . = 0
                        0000                .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002                0000                        .BLKW   1
00000074                0002                . = 116
                        0074                .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078                0074                        .BLKL   1
                        0078                .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C                0078                        .BLKL   1
                        007C                .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080                007C                        .BLKL   1
                        0080                .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084                0080                        .BLKL   1
                        0084                .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088                0084                        .BLKL   1
                        0088                .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C                0088                        .BLKL   1
                        008C                .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090                008C                        .BLKL   1
                        0090                .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                        0090                .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074                0090                . = 116
                        0074                .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
```

```
00000075  0074                          .BLKB   1
          0075          .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                          .BLKB   1
          0076          .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                          .BLKB   1
          0077          .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                          .BLKB   1
          0078          .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                          .BLKL   1
0000007E  007C                          .BLKW   1
          007E          .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                          .BLKW   1
          0080          .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                          .BLKL   1
00000080  0084          . = 128
00000084  0080                          .BLKL   1
          0084          .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                          .BLKL   1
          0088          .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                          .BLKW   1
          008A          .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                          .BLKW   1
          008C          .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
          008C          .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                          .BLKW   1
          008E          .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                          .BLKW   1
          0090          .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                          .BLKW   1
          0092          .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                          .BLKW   1
          0094          .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                          .BLKB   1
          0095          .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                          .BLKB   1
          0096          .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                          .BLKW   1
00000096  0098          . = 150
00000098  0096                          .BLKW   1
          0098          .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                          .BLKL   1
          009C          .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                          .BLKL   1
          00A0          .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4  00A0                          .BLKL   1
          00A4          .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6  00A4                          .BLKW   1
          00A6          .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7  00A6                          .BLKB   1
000000A8  00A7                          .BLKB   1
00000074  00A8          . = 116
00000078  0074                          .BLKL   1
0000007C  0078                          .BLKL   1
00000080  007C                          .BLKL   1
00000084  0080                          .BLKL   1
00000088  0084                          .BLKL   1
0000008C  0088                          .BLKL   1
```

```
                    008C                .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090  008C                                        .BLKL   1
00000094  0090                                        .BLKL   1
00000098  0094                                        .BLKL   1
0000009C  0098                                        .BLKL   1
                    009C                .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D  009C                                        .BLKB   1
                    009D                .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E  009D                                        .BLKB   1
                    009E                .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F  009E                                        .BLKB   1
000000A0  009F                                        .BLKB   1
                    00A0                .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1  00A0                                        .BLKB   1
                    00A1                .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2  00A1                                        .BLKB   1
                    00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3  00A2                                        .BLKB   1
000000A4  00A3                                        .BLKB   1
                    00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5  00A4                                        .BLKB   1
                    00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7  00A5                                        .BLKW   1
000000A8  00A7                                        .BLKB   1
                    00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC  00A8                                        .BLKL   1
000000B0  00AC                                        .BLKL   1
000000B4  00B0                                        .BLKL   1
000000B8  00B4                                        .BLKL   1
                    00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC  00B8                                        .BLKL   1
                    00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                    00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074  00BC                          . = 116
                    0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078  0074                                        .BLKL   1
                    0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C  0078                                        .BLKL   1
                    007C                .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E  007C                                        .BLKW   1
00000080  007E                                        .BLKW   1
                    0000                .RESTORE
                    0000    105         $VCBDEF                                 ;DEFINE VCB OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
00000000  00BC                          .=0
                    0000                .IIF    NB,VCB$L_FCBFL, VCB$L_FCBFL:
00000004  0000                                        .BLKL   1
                    0004                .IIF    NB,VCB$L_FCBBL, VCB$L_FCBBL:
00000008  0004                                        .BLKL   1
                    0008                .IIF    NB,VCB$W_SIZE,  VCB$W_SIZE:
0000000A  0008                                        .BLKW   1
                    000A                .IIF    NB,VCB$B_TYPE,  VCB$B_TYPE:
0000000B  000A                                        .BLKB   1
                    000B                .IIF    NB,VCB$C_MRKLEN,        VCB$C_MRKLEN:
                    000B                .IIF    NB,VCB$K_MRKLEN,        VCB$K_MRKLEN:
                    000B                .IIF    NB,VCB$B_STATUS,        VCB$B_STATUS:
```

```
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,VCB$W_TRANS, VCB$W_TRANS:
0000000E  000C                                  .BLKW   1
          000E                  .IIF    NB,VCB$W_RVN,   VCB$W_RVN:
00000010  000E                                  .BLKW   1
          0010                  .IIF    NB,VCB$L_AQB,   VCB$L_AQB:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,VCB$T_VOLNAME,       VCB$T_VOLNAME:
00000020  0014                                  .BLKB   12
          0020                  .IIF    NB,VCB$L_RVT,   VCB$L_RVT:
00000024  0020                                  .BLKL   1
          0024                  .IIF    NB,VCB$C_COMLEN,        VCB$C_COMLEN:
          0024                  .IIF    NB,VCB$K_COMLEN,        VCB$K_COMLEN:
          0024                  .IIF    NB,VCB$L_HOMELBN,       VCB$L_HOMELBN:
00000028  0024                                  .BLKL   1
          0028                  .IIF    NB,VCB$L_HOME2LBN,      VCB$L_HOME2LBN:
0000002C  0028                                  .BLKL   1
          002C                  .IIF    NB,VCB$L_IXHDR2LBN,     VCB$L_IXHDR2LBN:
00000030  002C                                  .BLKL   1
          0030                  .IIF    NB,VCB$L_IBMAPLBN,      VCB$L_IBMAPLBN:
00000034  0030                                  .BLKL   1
          0034                  .IIF    NB,VCB$L_SBMAPLBN,      VCB$L_SBMAPLBN:
00000038  0034                                  .BLKL   1
          0038                  .IIF    NB,VCB$B_IBMAPSIZE,     VCB$B_IBMAPSIZE:
00000039  0038                                  .BLKB   1
          0039                  .IIF    NB,VCB$B_SBMAPSIZE,     VCB$B_SBMAPSIZE:
0000003A  0039                                  .BLKB   1
          003A                  .IIF    NB,VCB$B_IBMAPVBN,      VCB$B_IBMAPVBN:
0000003B  003A                                  .BLKB   1
          003B                  .IIF    NB,VCB$B_SBMAPVBN,      VCB$B_SBMAPVBN:
0000003C  003B                                  .BLKB   1
          003C                  .IIF    NB,VCB$W_CLUSTER,       VCB$W_CLUSTER:
0000003E  003C                                  .BLKW   1
          003E                  .IIF    NB,VCB$W_EXTEND,        VCB$W_EXTEND:
00000040  003E                                  .BLKW   1
          0040                  .IIF    NB,VCB$L_FREE,  VCB$L_FREE:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,VCB$L_MAXFILES,      VCB$L_MAXFILES:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,VCB$B_WINDOW,        VCB$B_WINDOW:
00000049  0048                                  .BLKB   1
          0049                  .IIF    NB,VCB$B_LRU_LIM,       VCB$B_LRU_LIM:
0000004A  0049                                  .BLKB   1
          004A                  .IIF    NB,VCB$W_FILEPROT,      VCB$W_FILEPROT:
0000004C  004A                                  .BLKW   1
          004C                  .IIF    NB,VCB$W_MCOUNT,        VCB$W_MCOUNT:
0000004E  004C                                  .BLKW   1
          004E                  .IIF    NB,VCB$B_EOFDELTA,      VCB$B_EOFDELTA:
0000004F  004E                                  .BLKB   1
          004F                  .IIF    NB,VCB$B_RESFILES,      VCB$B_RESFILES:
00000050  004F                                  .BLKB   1
          0050                  .IIF    NB,VCB$W_RECORDSZ,      VCB$W_RECORDSZ:
00000052  0050                                  .BLKW   1
          0052                  .IIF    NB,VCB$B_BLOCKFACT,     VCB$B_BLOCKFACT:
00000053  0052                                  .BLKB   1
          0053                  .IIF    NB,VCB$B_STATUS2,       VCB$B_STATUS2:
00000054  0053                                  .BLKB   1
```

```
                    0054                .IIF    NB,VCB$L_QUOTAFCB,      VCB$L_QUOTAFCB:
00000058            0054                                .BLKL   1
                    0058                .IIF    NB,VCB$L_CACHE, VCB$L_CACHE:
0000005C            0058                                .BLKL   1
                    005C                .IIF    NB,VCB$L_QUOCACHE,      VCB$L_QUOCACHE:
00000060            005C                                .BLKL   1
                    0060                .IIF    NB,VCB$W_QUOSIZE,       VCB$W_QUOSIZE:
00000062            0060                                .BLKW   1
                    0062                .IIF    NB,VCB$W_PENDERR,       VCB$W_PENDERR:
00000064            0062                                .BLKW   1
                    0064                .IIF    NB,VCB$C_LENGTH,        VCB$C_LENGTH:
                    0064                .IIF    NB,VCB$K_LENGTH,        VCB$K_LENGTH:
00000000            0064                . = 0
                    0000                .IIF    NB,VCB$L_BLOCKFL,       VCB$L_BLOCKFL:
00000004            0000                                .BLKL   1
                    0004                .IIF    NB,VCB$L_BLOCKBL,       VCB$L_BLOCKBL:
00000008            0004                                .BLKL   1
0000000B            0008                . = 11
00000024            000B                . = 36
                    0024                .IIF    NB,VCB$L_CUR_FID,       VCB$L_CUR_FID:
                    0024                .IIF    NB,VCB$W_CUR_NUM,       VCB$W_CUR_NUM:
00000026            0024                                .BLKW   1
                    0026                .IIF    NB,VCB$W_CUR_SEQ,       VCB$W_CUR_SEQ:
00000028            0026                                .BLKW   1
                    0028                .IIF    NB,VCB$L_START_FID,     VCB$L_START_FID:
                    0028                .IIF    NB,VCB$W_START_NUM,     VCB$W_START_NUM:
0000002A            0028                                .BLKW   1
                    002A                .IIF    NB,VCB$W_START_SEQ,     VCB$W_START_SEQ:
0000002C            002A                                .BLKW   1
                    002C                .IIF    NB,VCB$W_MODE,  VCB$W_MODE:
0000002E            002C                                .BLKW   1
                    002E                .IIF    NB,VCB$B_TM,    VCB$B_TM:
0000002F            002E                                .BLKB   1
                    002F                .IIF    NB,VCB$B_CUR_RVN,       VCB$B_CUR_RVN:
00000030            002F                                .BLKB   1
                    0030                .IIF    NB,VCB$L_ST_RECORD,     VCB$L_ST_RECORD:
00000034            0030                                .BLKL   1
                    0034                .IIF    NB,VCB$L_MVL,   VCB$L_MVL:
00000038            0034                                .BLKL   1
                    0038                .IIF    NB,VCB$L_WCB,   VCB$L_WCB:
0000003C            0038                                .BLKL   1
                    003C                .IIF    NB,VCB$L_VPFL,  VCB$L_VPFL:
00000040            003C                                .BLKL   1
                    0040                .IIF    NB,VCB$L_VPBL,  VCB$L_VPBL:
00000044            0040                                .BLKL   1
                    0044                .IIF    NB,VCB$L_USRLBLAST,     VCB$L_USRLBLAST:
00000048            0044                                .BLKL   1
0000000B            0048                . = 11
                    000B                .IIF    NB,VCB$B_QNAMECNT,      VCB$B_QNAMECNT:
0000000C            000B                                .BLKB   1
                    000C                .IIF    NB,VCB$T_QNAME, VCB$T_QNAME:
00000020            000C                                .BLKB   20
                    0000                .RESTORE
                    0000    106         $WCBDEF                                 ;DEFINE WINDOW CONTROL BLOCK OFFSETS
                    0000                .SAVE   LOCAL_BLOCK
                    0000                .PSECT  $ABS$,ABS
00000000            00BC                .=0
```

```
                    0000                .IIF    NB,WCB$L_WLFL,  WCB$L_WLFL:
00000004            0000                        .BLKL   1
                    0004                .IIF    NB,WCB$L_WLBL,  WCB$L_WLBL:
00000008            0004                        .BLKL   1
                    0008                .IIF    NB,WCB$W_SIZE,  WCB$W_SIZE:
0000000A            0008                        .BLKW   1
                    000A                .IIF    NB,WCB$B_TYPE,  WCB$B_TYPE:
0000000B            000A                        .BLKB   1
                    000B                .IIF    NB,WCB$B_ACCESS,        WCB$B_ACCESS:
0000000C            000B                        .BLKB   1
                    000C                .IIF    NB,WCB$L_PID,   WCB$L_PID:
0000000E            000C                        .BLKB   2
                    000E                .IIF    NB,WCB$W_REFCNT,        WCB$W_REFCNT:
00000010            000E                        .BLKW   1
                    0010                .IIF    NB,WCB$L_ORGUCB,        WCB$L_ORGUCB:
00000014            0010                        .BLKL   1
                    0014                .IIF    NB,WCB$W_ACON,  WCB$W_ACON:
00000016            0014                        .BLKW   1
                    0016                .IIF    NB,WCB$W_NMAP,  WCB$W_NMAP:
00000018            0016                        .BLKW   1
                    0018                .IIF    NB,WCB$L_FCB,   WCB$L_FCB:
0000001C            0018                        .BLKL   1
                    001C                .IIF    NB,WCB$L_RVT,   WCB$L_RVT:
00000020            001C                        .BLKL   1
                    0020                .IIF    NB,WCB$L_STVBN, WCB$L_STVBN:
00000024            0020                        .BLKL   1
                    0024                .IIF    NB,WCB$C_MAP,   WCB$C_MAP:
                    0024                .IIF    NB,WCB$K_MAP,   WCB$K_MAP:
                    0024                .IIF    NB,WCB$C_LENGTH,        WCB$C_LENGTH:
                    0024                .IIF    NB,WCB$K_LENGTH,        WCB$K_LENGTH:
                    0024                .IIF    NB,WCB$W_P1_COUNT,      WCB$W_P1_COUNT:
00000026            0024                        .BLKW   1
                    0026                .IIF    NB,WCB$L_P1_LBN,        WCB$L_P1_LBN:
0000002A            0026                        .BLKL   1
                    002A                .IIF    NB,WCB$W_P2_COUNT,      WCB$W_P2_COUNT:
0000002C            002A                        .BLKW   1
                    002C                .IIF    NB,WCB$L_P2_LBN,        WCB$L_P2_LBN:
00000030            002C                        .BLKL   1
00000000            0030                . = 0
                    0000                .IIF    NB,WCB$W_COUNT, WCB$W_COUNT:
00000002            0000                        .BLKW   1
                    0002                .IIF    NB,WCB$L_LBN,   WCB$L_LBN:
00000006            0002                        .BLKL   1
00000000            0006                . = 0
FFFFFFFA            0000                        .BLKB   -6
                    FFFA                .IIF    NB,WCB$W_PREVCOUNT,     WCB$W_PREVCOUNT:
FFFFFFFC            FFFA                        .BLKW   1
                    FFFC                .IIF    NB,WCB$L_PREVLBN,       WCB$L_PREVLBN:
00000000            FFFC                        .BLKL   1
                    0000                .RESTORE
                    0000    107
                    0000    108 ;
                    0000    109 ; LOCAL SYMBOLS
                    0000    110 ;
                    0000    111 ; ARGUMENT LIST OFFSET DEFINITIONS
                    0000    112 ;
                    0000    113
```

```
                    00000004  0000    114 EFN=4                                   ;EVENT FLAG NUMBER
                    00000008  0000    115 CHAN=8                                  ;I/O CHANNEL NUMBER
                    0000000C  0000    116 FUNC=12                                 ;I/O FUNCTION CODE
                    00000010  0000    117 IOSB=16                                 ;ADDRESS OF I/O STATUS BLOCK
                    00000014  0000    118 ASTADR=20                               ;ADDRESS OF AST SERVICE ROUTINE
                    00000018  0000    119 ASTPRM=24                               ;AST SERVICE ROUTINE PARAMETER
                    0000001C  0000    120 P1=28                                   ;FIRST FUNCTION DEPENDENT PARAMETER
                    00000020  0000    121 P2=32                                   ;SECOND FUNCTION DEPENDENT PARAMETER
                    00000024  0000    122 P3=36                                   ;THIRD FUNCTION DEPENDENT PARAMETER
                    00000028  0000    123 P4=40                                   ;FOURTH FUNCTION DEPENDENT PARAMETER
                    0000002C  0000    124 P5=44                                   ;FIFTH FUNCTION DEPENDENT PARAMETER
                    00000030  0000    125 P6=48                                   ;SIXTH FUNCTION DEPENDENT PARAMETER
                              0000    126
                              0000    127 ;
                              0000    128 ; FUNCTION DECISION TABLE OFFSET DEFINITIONS
                              0000    129 ;
                              0000    130
                    00000000  0000    131 LEGAL=0                                 ;LEGAL FUNCTION MASK
                    00000008  0000    132 IOTYPE=8                                ;I/O FUNCTION TYPE MASK
                    00000010  0000    133 FDTACT=16                               ;ACTION ROUTINE MASKS
                          00000000    134         .PSECT  SEP, SHR, EXE, WRT, LONG
                              0000    135         MODNAM  QIOREQ
 51 45 52 4F 49 51 00'        0000        $MODULE:        .ASCIC  \QIOREQ\
                    06        0000
```

```
0007   137          .SBTTL  QUEUE I/O REQUEST
0007   138 ;*
0007   139 ; EXE$QIOREQ - QUEUE I/O REQUEST
0007   140 ;
0007   141 ; THIS SERVICE PROVIDES THE CAPABILITY TO INITIATE AN I/O OPERATION
0007   142 ; BY QUEUEING A REQUEST TO A DEVICE'S ASSOCIATED DRIVER.  ONCE THE
0007   143 ; OPERATION HAS BEEN INITIATED, CONTROL WILL RETURN TO THE CALLER
0007   144 ; WHO CAN SYNCHRONIZE I/O COMPLETION IN ONE OF THREE WAYS:
0007   145 ;
0007   146 ;       1) SPECIFY THE ADDRESS OF AN AST ROUTINE THAT WILL BE
0007   147 ;          EXECUTED WHEN THE I/O COMPLETES.
0007   148 ;
0007   149 ;       2) WAIT FOR THE SPECIFIED EVENT FLAG TO BE SET.
0007   150 ;
0007   151 ;       3) POLL THE SPECIFIED I/O STATUS BLOCK FOR A COMPLETION
0007   152 ;          STATUS.
0007   153 ;
0007   154 ; THIS ROUTINE VERIFYS THE FUNCTION INDEPENDENT PARAMETERS, ALLOCATES
0007   155 ; AN I/O REQUEST PACKET, COPIES THE FUNCTION INDEPENDENT PARAMETERS AND
0007   156 ; PROCESS INFORMATION TO THE I/O PACKET, CHECKS ACCESS TO THE DEVICE,
0007   157 ; AND CALLS THE DRIVER'S FUNCTION DECISION TABLE ROUTINE(S) THAT CORRESPOND
0007   158 ; TO THE SPECIFIED FUNCTION.  IT IS THEN UP TO THE FDT ROUTINE TO EITHER
0007   159 ; COMPLETE THE REQUEST IMMEDIATELY (EXE$ABORTIO OR EXE$FINISHIO) OR TO
0007   160 ; QUEUE THE I/O REQUEST FOR FURTHER PROCESSING BY THE DRIVER'S STARTIO
0007   161 ; ROUTINE (EXE$QIODRVPKT).
0007   162 ;
0007   163 ; INPUTS:
0007   164 ;
0007   165 ;       EFN(AP) = EVENT FLAG NUMBER.
0007   166 ;       CHAN(AP) = I/O CHANNEL NUMBER.
0007   167 ;       FUNC(AP) = I/O FUNCTION CODE.
0007   168 ;       IOSB(AP) = ADDRESS OF I/O STATUS BLOCK.
0007   169 ;       ASTADR(AP) = ADDRESS OF AST SERVICE ROUTINE.
0007   170 ;       ASTPRM(AP) = AST SERVICE ROUTINE PARAMETER.
0007   171 ;       P1(AP) TO P6(AP) = FUNCTION DEPENDENT PARAMETERS.
0007   172 ;
0007   173 ;       R4 = CURRENT PROCESS PCB ADDRESS.
0007   174 ;
0007   175 ; OUTPUTS:
0007   176 ;
0007   177 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO INITIATE THE I/O REQUEST.
0007   178 ;
0007   179 ;               R0 = SS$_ABORT - A NETWORK LOGICAL LINK WAS BROKEN.
0007   180 ;
0007   181 ;               R0 = SS$_ACCVIO - THE I/O STATUS BLOCK CANNOT BE WRITTEN BY
0007   182 ;                    THE CALLER.
0007   183 ;
0007   184 ;               R0 = SS$_DEVOFFLINE - THE SPECIFIED DEVICE IS OFFLINE.
0007   185 ;
0007   186 ;               R0 = SS$_EXQUOTA - THE PROCESS HAS EXCEEDED ITS BUFFERED I/O
0007   187 ;                    QUOTA, DIRECT I/O QUOTA, OR BUFFERED I/O BYTE COUNT
0007   188 ;                    QUOTA AND HAS DISABLED RESOURCE WAIT MODE.  OR, THE
0007   189 ;                    PROCESS HAS EXCEEDED ITS AST LIMIT QUOTA.
0007   190 ;
0007   191 ;               R0 = SS$_ILLEFC - AN ILLEGAL EVENT FLAG NUMBER WAS SPECIFIED.
0007   192 ;
0007   193 ;               R0 = SS$_INSFMEM - INSUFFICIENT DYNAMIC MEMORY IS AVAILABLE
```

```
                                0007   194 ;                   TO ALLOCATE AN I/O REQUEST PACKET AND THE PROCESS HAS
                                0007   195 ;                   DISABLED RESOURCE WAIT MODE.
                                0007   196 ;
                                0007   197 ;           R0 = SS$_IVCHAN - AN INVALID CHANNEL NUMBER WAS SPECIFIED.
                                0007   198 ;
                                0007   199 ;           R0 = SS$_NOPRIV - THE SPECIFIED CHANNEL DOES NOT EXIST OR WAS
                                0007   200 ;                ASSIGNED FROM A MORE PRIVILEGED ACCESS MODE.  OR, THE
                                0007   201 ;                PROCESS DOES NOT HAVE THE PRIVILEGE TO PERFORM THE
                                0007   202 ;                SPECIFIED TYPE OF I/O FUNCTION ON THE DEVICE.
                                0007   203 ;
                                0007   204 ;           R0 = SS$_UNASEFC - UNASSOCIATED EVENT FLAG CLUSTER SPECIFIED.
                                0007   205 ;
                                0007   206 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0007   207 ;
                                0007   208 ;           R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0007   209 ;-
                                0007   210
                                0007   211          .ENABL  LSB
                           0FFC 0007   212          .ENTRY  VMS$QIO,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
    54  00000000'EF     DE     0009   213          MOVAL   DS$AX_SOFTPCB, R4       ; Get PCB address into R4
      7E    04 AC     9A     0010   214 10$:     MOVZBL  EFN(AP),-(SP)           ;GET EVENT FLAG #
00000000'9F    01     FB     0014   215          CALLS   #1,@#SYS$CLREF          ;CLEAR SPECIFIED EVENT FLAG
          03 50     E8     001B   216          BLBS    R0, 15$                 ; Event flag cleared OK
           0080     31     001E   217          BRW     ERROR                   ; Didn't work
                    0021   218 15$:
      50    08 AC     3C     0021   219          MOVZWL  CHAN(AP),R0             ;GET CHANNEL NUMBER
            FFD8'     30     0025   220          BSBW    IOC$VERIFYCHAN          ;VERIFY CHANNEL NUMBER
         56   51     D0     0028   221          MOVL    R1,R6                   ;SAVE ADDRESS OF CCB
           73 50     E9     002B   222          BLBC    R0,ERROR                ;IF LBC INVALID CHANNEL
         55   66     D0     002E   223          MOVL    CCB$L_UCB(R6),R5        ;GET ASSIGNED DEVICE UCB ADDRESS
         59   52     D0     0031   224          MOVL    R2,R9                   ;SAVE CHANNEL INDEX
      5A    0C AC     3C     0034   225          MOVZWL  FUNC(AP),R10            ;GET I/O FUNCTION CODE AND MODIFIERS
57    5A  FFFFFFFF'8F     CB     0038   226          BICL3   #^C<IO$M_FCODE>,R10,R7  ;CLEAR ALL BUT I/O FUNCTION CODE
   09 34 A5    00'     E1     0040   227          BBC     S^#DEV$V_SPL,UCB$L_DEVCHAR(R5),17$ ;IF CLR, DEVICE NOT SPOOLED
         57    00'     D1     0045   228          CMPL    S^#IO$_LOGICAL,R7       ;VIRTUAL I/O FUNCTION?
               04     18     0048   229          BGEQ    17$                     ;IF GEQ NO
      55    54 A5     D0     004A   230          MOVL    UCB$L_AMB(R5),R5        ;GET INTERMEDIATE DEVICE UCB ADDRESS
      50    24 A5     D0     004E   231 17$:     MOVL    UCB$L_DDB(R5),R0        ;GET ADDRESS OF DDB
      50    0C A0     D0     0052   232          MOVL    DDB$L_DDT(R0),R0        ;GET ADDRESS OF DDT
      58    08 A0     D0     0056   233          MOVL    DDT$L_FDT(R0),R8        ;GET ADDRESS OF FDT
      03 58     10     E0     005A   234          BBS     #16,R8,20$              ;BRANCH IF SUPERVISOR ADDRESS
         58     50     C0     005E   235          ADDL    R0,R8                   ;CHANGE OFFSET TO ABSOLUTE
      2B 68     57     E1     0061   236 20$:     BBC     R7,LEGAL(R8),50$        ;IF CLR, ILLEGAL I/O FUNCTION
   2D 58 A5     04     E1     0065   237          BBC     #UCB$V_ONLINE,UCB$W_STS(R5),60$ ;IF CLR, DEVICE OFFLINE
      51    10 AC     D0     006A   238          MOVL    IOSB(AP),R1             ;GET ADDRESS OF I/O STATUS BLOCK
               08     13     006E   239          BEQL    30$                     ;IF EQL NONE SPECIFIED
                      0070   240          IFNOWRT #8,(R1),70$             ;CAN I/O STATUS BLOCK BE WRITTEN?
 61    08    00     0D     0070                   PROBEW  #0,#8,(R1)
               28     13     0074                   BEQL    70$
               61     7C     0076   241          CLRQ    (R1)                    ;CLEAR I/O STATUS BLOCK
               7E     DC     0078   242 30$:     MOVPSL  -(SP)                   ;READ CURRENT PSL
      5B    3A A4     DE     007A   243          MOVAL   PCB$W_BIOCNT(R4),R11    ;SET FOR BUFFERED I/O FUNCTION
   04 08 A8     57     E0     007E   244          BBS     R7,IOTYPE(R8),40$       ;IF SET, BUFFERED I/O FUNCTION
      5B    3E A4     DE     0083   245          MOVAL   PCB$W_DIOCNT(R4),R11    ;SET FOR DIRECT I/O FUNCTION
         52    5B     D0     0087   246 40$:     MOVL    R11,R2                  ;SET ADDRESS OF USAGE COUNT WORD
        31 04 A6     E9     008A   247          BLBC    CCB$L_WIND(R6),GTPKT    ;IF LBC ACCESS/DEACCESS NOT PENDING
               80     11     008E   248          BRB     10$                     ;
```

```
                50    0000'8F    3C   0090    249 50$:    MOVZWL  #SS$_ILLIOFUNC,R0       ;SET ILLEGAL I/O FUNCTION STATUS
                                0A   11   0095    250         BRB     ERROR               ;
                50    0000'8F    3C   0097    251 60$:    MOVZWL  #SS$_DEVOFFLINE,R0      ;SET DEVICE OFFLINE STATUS
                                03   11   009C    252         BRB     ERROR               ;
                      50    00'  3C   009E    253 70$:    MOVZWL  S^#SS$_ACCVIO,R0        ;SET ACCESS VIOLATION STATUS
                                   00A1    254 ERROR:  SETIPL  #0                      ;ALLOW INTERRUPTS
                      12    00   DA   00A1                MTPR    #0,S^#PR$_IPL
                      50    DD   00A4    255         PUSHL   R0                      ;SAVE FINAL STATUS
                   7E   04 AC   9A   00A6    256         MOVZBL  EFN(AP),-(SP)           ;GET SPECIFIED EVENT FLAG #
          00000000'9F    01    FB   00AA    257         CALLS   #1,@#SYS$SETEF          ;SET SPECIFIED EVENT FLAG
                         01    BA   00B1    258         POPR    #^M<R0>                 ;RESTORE FINAL STATUS
                         04    00B3    259         RET                             ;
                               00B4    260
                               00B4    261 ;
                               00B4    262 ; ALLOCATE REQUEST I/O PACKET
                               00B4    263 ;
                               00B4    264
          00000000'EF          16   00B4    265 ALLOC:  JSB     L^EXE$ALLOCIRP          ; Allocate I/O request packet
                   0B 50       E8   00BA    266         BLBS    R0,SUCCES               ;IF LBS SUCCESSFUL ALLOCATION
                      E2       11   00BD    267         BRB     ERROR                   ;
          52    00000000'FF    0F   00BF    268 GTPKT:  REMQUE  @L^IOC$GL_IRPFL, R2     ; Get I/O packet from lookaside list
                         EC    1D   00C6    269         BVS     ALLOC                   ;IF VS EMPTY LIST
                               00C8    270         .DSABL  LSB
                               00C8    271
                               00C8    272 ;
                               00C8    273 ; BUILD DEVICE INDEPENDENT PART OF I/O PACKET
                               00C8    274 ;
                               00C8    275
                         6B    B7   00C8    276 SUCCES: DECW    (R11)                   ;UPDATE I/O COUNT FOR FUNCTION TYPE
                      0A A6    B6   00CA    277         INCW    CCB$W_IOC(R6)           ;INCREMENT OUTSTANDING I/O ON CHANNEL
                   53    52    D0   00CD    278         MOVL    R2,R3                   ;COPY ADDRESS OF ALLOCATED I/O PACKET
                   52    08    C0   00D0    279         ADDL    #IRP$W_SIZE,R2          ;POINT TO SIZE FIELD
                82 005C 8F    B0   00D3    280         MOVW    #IRP$C_LENGTH,(R2)+     ;INSERT LENGTH OF I/O PACKET
                   82    0A    90   00D8    281         MOVB    #DYN$C_IRP,(R2)+        ;INSERT DATA STRUCTURE TYPE
          62    6E    02    16   EF   00DB    282         EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,(SP),(R2) ;INSERT ACCESS MODE
                         52    D6   00E0    283         INCL    R2                      ;ADJUST PAST ACCESS MODE FIELD
                   82    60 A4    D0   00E2    284         MOVL    PCB$L_PID(R4),(R2)+     ;INSERT PROCESS ID OF CURRENT PROCESS
                   82    14 AC    7D   00E6    285         MOVQ    ASTADR(AP),(R2)+        ;INSERT AST ADDRESS AND PARAMETER
                   82    04 A6    D0   00EA    286         MOVL    CCB$L_WIND(R6),(R2)+    ;Insert Window address
                      82    55    D0   00EE    287         MOVL    R5,(R2)+                ;INSERT DEVICE UCB ADDRESS
                      82    5A    B0   00F1    288         MOVW    R10,(R2)+               ;INSERT I/O FUNCTION CODE
                   82    04 AC    90   00F4    289         MOVB    EFN(AP),(R2)+           ;INSERT EVENT FLAG NUMBER
                         82    94   00F8    290         CLRB    (R2)+                   ;INSERT PROCESS BASE PRIORITY
                   82    10 AC    D0   00FA    291         MOVL    IOSB(AP),(R2)+          ;INSERT I/O STATUS BLOCK ADDRESS
                      82    59    3C   00FE    292         MOVZWL  R9,(R2)+                ;INSERT CHANNEL INDEX AND ZERO STATUS
          04 08 A8    57    E1   0101    293         BBC     R7,IOTYPE(R8),20$       ;IF CLR, DIRECT I/O FUNCTION
             FE A2    01    A8   0106    294         BISW    #IRP$M_BUFIO,-2(R2)     ;SET TO BUFFERED I/O FUNCTION
                         62    7C   010A    295 20$:    CLRQ    (R2)                    ;CLEAR PTE ADDRESS, BYTE OFFSET, AND BYTE CO
                               010C    296
                               010C    297 ;
                               010C    298 ; CHECK IF REQUESTING PROCESS HAS PRIVILEGE TO ACCESS DEVICE
                               010C    299 ;
                               010C    300
                      57    00'  D1   010C    301 ACCESS: CMPL    S^#IO$_WRITEVBLK,R7     ;POSSIBLY VIRTUAL READ OR WRITE?
                         20    1A   010F    302         BGTRU   15$                     ;IF GTRU NO
                      57    00'  D1   0111    303         CMPL    S^#IO$_READVBLK,R7      ;VIRTUAL READ OR WRITE?
                         1B    1F   0114    304         BLSSU   15$                     ;IF LSSU NO
```

```
                OF 34 A5    00'  E1   0116   305          BBC     S^#DEV$V_FOD,UCB$L_DEVCHAR(R5),5$ ;IF CLR, NOT FILE DEVICE
                                      011B   306
                                      011B   307 ;
                                      011B   308 ; THE FOLLOWING TEST IS NECESSITATED BY THE SYSTEM INITIALIZATION SEQUENCE
                                      011B   309 ;
                                      011B   310
                      18 A3    D5     011B   311          TSTL    IRP$L_WIND(R3)          ;WINDOW ADDRESS SPECIFIED?
                         11    12     011E   312          BNEQ    15$                     ;IF NEQ YES
                OE 34 A5    00'  E1   0120   313          BBC     S^#DEV$V_MNT,UCB$L_DEVCHAR(R5),60$ ;IF CLR, DEVICE NOT MOUNTED
                0C 34 A5    00'  E1   0125   314          BBC     S^#DEV$V_FOR,UCB$L_DEVCHAR(R5),80$ ;IF CLR, MOUNTED STRUCTURED
                                      012A   315
                                      012A   316 ;
                                      012A   317 ; CONVERT VIRTUAL READ/WRITE FUNCTION TO ITS LOGICAL COUNTERPART
                                      012A   318 ;
                                      012A   319
                      57    00'  C2   012A   320 5$:     SUBL    S^#IO$_READVBLK-IO$_READLBLK,R7 ;CONVERT TO LOGICAL FUNCTION
                   20 A3    00'  A2   012D   321          SUBW    S^#IO$_READVBLK-IO$_READLBLK,IRP$W_FUNC(R3) ;
                         03    11     0131   322 15$:     BRB     80$                     ; Don't care about VIRTUAL/LOG/PHYS rights
                                      0133   323 60$:
                            0064    31 0133   324 70$:     BRW     EXE$ABORTIO             ;
                                      0136   325
                                      0136   326 ;
                                      0136   327 ; PROCESS HAS ACCESS TO DEVICE
                                      0136   328 ;
                                      0136   329
                      57    00'  D1   0136   330 80$:     CMPL    S^#IO$_PHYSICAL,R7      ;LOGICAL OR VIRTUAL I/O FUNCTION?
                         39    19     0139   331          BLSS    90$                     ;IF LSS YES
                2A A3    0100 8F    A8 013B   332          BISW    #IRP$M_PHYSIO,IRP$W_STS(R3) ;SET PHYSICAL I/O FLAG
                   59    30 AC    D0 0141   333          MOVL    P6(AP),R9               ;GET ADDRESS OF DIAGNOSTIC BUFFER
                         2D    13     0145   334          BEQL    90$                     ;IF EQL NONE SPECIFIED
                   51    24 A5    D0 0147   335          MOVL    UCB$L_DDB(R5),R1        ;GET ADDRESS OF DDB
                   51    0C A1    D0 014B   336          MOVL    DDB$L_DDT(R1),R1        ;GET ADDRESS OF DDT
                   51    14 A1    3C 014F   337          MOVZWL  DDT$W_DIAGBUF(R1),R1    ;GET SIZE OF DIAGNOSTIC BUFFER
                         1F    13     0153   338          BEQL    90$                     ;IF EQL NO DIAGNOSTIC FUNCTIONS
                         53    DD     0155   339          PUSHL   R3                      ;SAVE I/O PACKET ADDRESS
                00000000'EF    16     0157   340          JSB     L^EXE$ALLOCBUF          ; Allocate diagnostic buffer
                         53 8ED0     015D   341          POPL    R3                      ;RETRIEVE I/O PACKET ADDRESS
                      D0 50    E9     0160   342          BLBC    R0,70$                  ;IF LBC ALLOCATION FAILURE
                   44 A3    52    D0 0163   343          MOVL    R2,IRP$L_DIAGBUF(R3)    ;SAVE ADDRESS OF DIAGNOSTIC BUFFER
                   82    0C A2    9E 0167   344          MOVAB   12(R2),(R2)+            ;SET POINTER TO DATA AREA
                      62    59    D0 016B   345          MOVL    R9,(R2)                 ;SAVE USER ADDRESS OF DIAGNOSTIC BUFFER
                2A A3    0080 8F    A8 016E   346          BISW    #IRP$M_DIAGBUF,IRP$W_STS(R3) ;SET DIAGNOSTIC BUFFER PRESENT
                                      0174   347
                                      0174   348 ;
                                      0174   349 ; CHECK IF AST IS SPECIFIED
                                      0174   350 ;
                                      0174   351
                      10 A3    D5     0174   352 90$:     TSTL    IRP$L_AST(R3)           ;AST ADDRESS SPECIFIED?
                         05    13     0177   353          BEQL    100$                    ;IF EQL NO
                 0B A3    40 8F    88 0179   354          BISB    #ACB$M_QUOTA,IRP$B_RMOD(R3) ;SET AST QUOTA UPDATE FLAG
                                      017E   355
                                      017E   356 ;
                                      017E   357 ; SCAN FUNCTION DECISION TABLE CALLING EACH SELECTED ACTION ROUTINE WITH:
                                      017E   358 ;
                                      017E   359 ;       R0 = SCRATCH.
                                      017E   360 ;       R1 = SCRATCH.
                                      017E   361 ;       R2 = SCRATCH.
```

```
                                017E    362 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                017E    363 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                017E    364 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                017E    365 ;       R6 = ADDRESS OF CCB.
                                017E    366 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                017E    367 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                017E    368 ;       R9 = SCRATCH.
                                017E    369 ;       R10 = SCRATCH.
                                017E    370 ;       R11 = SCRATCH.
                                017E    371 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                017E    372 ;
                                017E    373
    58    04    C0    017E    374 100$:   ADDL    #FDTACT-12,R8           ;POINT TO ACTION ROUTINE MASKS
    5C    1C    C0    0181    375         ADDL    #P1,AP                  ;POINT TO FIRST FUNCTION DEPENDENT PARAMETER
    58    0C    C0    0184    376 110$:   ADDL    #12,R8                  ;POINT TO NEXT FUNCTION MASK
F9  68    57    E1    0187    377         BBC     R7,(R8),110$            ;IF CLR, THEN ACTION NOT SELECTED
50    08  A8    D0    018B    378         MOVL    8(R8),R0                ;GET ADDRESS OF ACTION ROUTINE
03  50    10    E0    018F    379         BBS     #16,R0,120$             ;BRANCH IF ABSOLUTE SUPERVISOR ADDR
    50    58    C0    0193    380         ADDL    R8,R0                   ;CHANGE OFFSET TO ABSOLUTE
          60    16    0196    381 120$:   JSB     (R0)                    ;CALL ACTION ROUTINE
          EA    11    0198    382         BRB     110$                    ;
```

```
                     019A   385          .SBTTL  COMPLETE I/O OPERATION
                     019A   386 ;+
                     019A   387 ; EXE$ABORTIO - ABORT I/O OPERATION
                     019A   388 ;
                     019A   389 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                     019A   390 ; TO FINISH AN I/O OPERATION WITHOUT RETURNING THE FINAL I/O STATUS.
                     019A   391 ;
                     019A   392 ; EXE$FINISHIO - FINISH I/O OPERATION
                     019A   393 ;
                     019A   394 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                     019A   395 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS.
                     019A   396 ;
                     019A   397 ; EXE$FINISHIOC - FINISH I/O OPERATION WITH SECOND I/O STATUS LONGWORD CLEARED
                     019A   398 ;
                     019A   399 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DESCISION TABLE ACTION ROUTINE
                     019A   400 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS WITH THE
                     019A   401 ; SECOND I/O STATUS LONGWORD CLEARED.
                     019A   402 ;
                     019A   403 ; INPUTS:
                     019A   404 ;
                     019A   405 ;       R0 = FIRST LONGWORD OF FINAL I/O STATUS.
                     019A   406 ;       R1 = SECOND LONGWORD OF FINAL I/O STATUS.
                     019A   407 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     019A   408 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     019A   409 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     019A   410 ;
                     019A   411 ; OUTPUTS:
                     019A   412 ;
                     019A   413 ;       THE FINAL I/O STATUS IS STORED IN THE I/O PACKET AND THE PACKET IS
                     019A   414 ;       INSERTED IN THE I/O POST PROCESSING QUEUE. A SOFTWARE INTERRUPT
                     019A   415 ;       IS GENERATED TO INITIATE I/O POST PROCESSING AND THE FIRST WORD
                     019A   416 ;       OF THE FINAL I/O STATUS IS RETURNED AS THE SERVICE STATUS.
                     019A   417 ;-
                     019A   418
                     019A   419          .ENABL  LSB
                     019A   420 EXE$ABORTIO::                                ;ABORT I/O OPERATION
          24 A3   D4 019A   421          CLRL    IRP$L_IOSB(R3)              ;CLEAR ADDRESS OF I/O STATUS BLOCK
    11 0B A3   06 E5 019D   422          BBCC    #ACB$V_QUOTA,IRP$B_RMOD(R3),10$ ;IF CLR, NO AST SPECIFIED
          38 A4   B6 01A2   423          INCW    PCB$W_ASTCNT(R4)            ;UPDATE AVAILABLE AST QUEUE ENTRIES
             0C   11 01A5   424          BRB     10$                         ;
                     01A7   425 EXE$FINISHIOC::                              ;FINISH I/O OPERATION CLEAR SECOND LONGWORD
             51   D4 01A7   426          CLRL    R1                          ;CLEAR SECOND I/O STATUS LONGWORD
                     01A9   427 EXE$FINISHIO::                               ;FINISH I/O OPERATION
          60 A5   D6 01A9   428          INCL    UCB$L_OPCNT(R5)             ;INCREMENT OPERATIONS COMPLETED
       34 A3   50   7D 01AC   429          MOVQ    R0,IRP$L_MEDIA(R3)          ;STORE FINAL I/O STATUS
          50   00' 3C 01B0   430          MOVZWL  S^#SS$_NORMAL,R0            ;SET NORMAL COMPLETION STATUS
00000000'FF   63   0E 01B3   431 10$:     INSQUE  (R3),@IOC$GL_PSBL           ;INSERT I/O PACKET IN POST PROCESS QUEUE
             42   12 01BA   432          BNEQ    QIORETURN                   ;IF NEQ NOT FIRST ENTRY IN QUEUE
                     01BC   433          SOFTINT #IPL$_IOPOST                ;INITIATE SOFTWARE INTERRUPT
          14   04   DA 01BC                      MTPR    #IPL$_IOPOST,S^#PR$_SIRR
             3D   11 01BF   434          BRB     QIORETURN                   ;
                     01C1   435          .DSABL  LSB
```

```
                     01C1   437          .SBTTL  QUEUE I/O PACKET TO DRIVER
                     01C1   438 ;+
                     01C1   439 ; EXE$QIODRVPKT - QUEUE I/O PACKET TO DRIVER
                     01C1   440 ;
                     01C1   441 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                     01C1   442 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE DRIVER.
                     01C1   443 ;
                     01C1   444 ; INPUTS:
                     01C1   445 ;
                     01C1   446 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     01C1   447 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     01C1   448 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     01C1   449 ;
                     01C1   450 ; OUTPUTS:
                     01C1   451 ;
                     01C1   452 ;       THE I/O PACKET IS QUEUED BY PRIORITY IN THE APPROPRIATE DEVICE
                     01C1   453 ;       QUEUE AND A NORMAL COMPLETION STATUS IS RETURNED.
                     01C1   454 ;-
                     01C1   455
                     01C1   456 EXE$QIODRVPKT::                          ;QUEUE I/O PACKET
3F   10              01C1   457          BSBB    EXE$INSIOQ             ;INSERT I/O PACKET IN DEVICE QUEUE
34   11              01C3   458          BRB     EXE$QIORETURN          ;
```

```
                        01C5    460          .SBTTL  EXE$ALTQUEPKT - Call driver ALTSTART entry point
                        01C5    461
                        01C5    462 ;
                        01C5    463 ; EXE$ALTQUEPKT - activates a driver at its ALTSTART entry point
                        01C5    464 ;
                        01C5    465 ; Routine description:
                        01C5    466 ;
                        01C5    467 ;       Locates and calls a driver entry point supplied as an alternate
                        01C5    468 ;       START I/O entry point. Does not test for unit busy before the
                        01C5    469 ;       call. Exits by returning to caller.
                        01C5    470 ;
                        01C5    471 ;       The routine expects to gain control at or below driver fork
                        01C5    472 ;       level. The routine raises to driver fork IPL before the call,
                        01C5    473 ;       and restores the previous IPL before returning to its caller.
                        01C5    474 ;
                        01C5    475 ; Inputs:
                        01C5    476 ;
                        01C5    477 ;       R3      - address of packet or buffer
                        01C5    478 ;       R5      - address of UCB
                        01C5    479 ;
                        01C5    480 ; Outputs:
                        01C5    481 ;
                        01C5    482 ;       Control returns to the requesting process.
                        01C5    483 ;
                        01C5    484 ;       The routine destroys R0-R1.
                        01C5    485 ;
                        01C5    486 ;--
                        01C5    487
                        01C5    488 EXE$ALTQUEPKT::                             ; Start I/O in driver.
                        01C5    489          DSBINT  UCB$B_FIPL(R5)             ; Raise to fork IPL.
    7E    12    DB      01C5                  MFPR    S^#PR$_IPL,-(SP)
12    0B  A5    DA      01C8                  MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
50    24  A5    D0      01CC    490          MOVL    UCB$L_DDB(R5),R0           ; Get address of unit's DDB.
50    0C  A0    D0      01D0    491          MOVL    DDB$L_DDT(R0),R0           ; Get address of unit's DDT.
51    1C  A0    D0      01D4    492          MOVL    DDT$L_ALTSTART(R0),R1      ; Get start offset
          09    13      01D8    493          BEQL    20$                        ; Branch if no alternate entry point
03 51     10    E0      01DA    494          BBS     #16,R1,10$                 ; Branch if supervisor absolute
   51     50    C0      01DE    495          ADDL    R0,R1                      ; Make relative to DDT
                        01E1    496                                             ; address.
                        01E1    497
                        01E1    498 10$:
          61    16      01E1    499          JSB     (R1)                       ; Go to the driver entry point.
                        01E3    500
                        01E3    501 20$:
                        01E3    502          ENBINT                             ; Reenable interrupts.
   12     8E    DA      01E3                  MTPR    (SP)+,S^#PR$_IPL
                05      01E6    503          RSB                                ; Return to caller.
```

```
                                01E7    505          .SBTTL  QUEUE I/O PACKET TO ACP
                                01E7    506 ;+
                                01E7    507 ; EXE$QIOACPPKT - QUEUE I/O PACKET TO ACP
                                01E7    508 ;
                                01E7    509 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
                                01E7    510 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE ACP.
                                01E7    511 ;
                                01E7    512 ; INPUTS:
                                01E7    513 ;
                                01E7    514 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                01E7    515 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                01E7    516 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                01E7    517 ;
                                01E7    518 ;       CURRENT IPL MUST BE AT SYNCH OR HIGHER LEVEL.
                                01E7    519 ;
                                01E7    520 ; OUTPUTS:
                                01E7    521 ;
                                01E7    522 ;       R4 ALTERED
                                01E7    523 ;       THE I/O PACKET IS QUEUED AT THE END OF THE APPROPRIATE ACP QUEUE
                                01E7    524 ;       AND A NORMAL COMPLETION STATUS IS RETURNED.
                                01E7    525 ;-
                                01E7    526
                                01E7    527 EXE$QIOACPPKT::                             ;QUEUE I/O PACKET TO ACP
                                01E7    528          BUG_CHECK NONEXSTACP               ;NONEXISTENT ACP PROCESS
                     00000000'9F  16    01E7                 JSB     @#BUG$CHECK
2C 50 43 41 54 53 58 45 4E 4F 4E 00'    01ED                 .ASCIC  "NONEXSTACP,"
                                0B      01ED
                                01F9    529 EXE$QIORETURN::                             ;QUEUE I/O REQUEST COMPLETION STATUS RETURN
                   50   0000'8F   3C    01F9    530          MOVZWL  #SS$_NORMAL,R0     ;SET NORMAL COMPLETION STATUS
                                01FE    531 QIORETURN:                                  ;RETURN SPECIFIED STATUS
                                01FE    532          SETIPL  #0                         ;ALLOW ALL INTERRUPTS
                12   00   DA    01FE                 MTPR    #0,S^#PR$_IPL
                                04      0201    533          RET                        ;
```

```
                                0202    535          .SBTTL  INSERT I/O PACKET IN UNIT QUEUE
                                0202    536 ;+
                                0202    537 ; EXE$INSIOQ - INSERT I/O PACKET IN UNIT QUEUE
                                0202    538 ;
                                0202    539 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A UNIT QUEUE AND CALL
                                0202    540 ; THE APPROPRIATE I/O DRIVER IF THE UNIT IS NOT BUSY.
                                0202    541 ;
                                0202    542 ; INPUTS:
                                0202    543 ;
                                0202    544 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                0202    545 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                                0202    546 ;-
                                0202    547
                                0202    548 EXE$INSIOQ::                                ;INSERT IN I/O QUEUE
                                0202    549          DSBINT  UCB$B_FIPL(R5)             ;RAISE IPL TO FORK LEVEL
          7E    12    DB      0202                   MFPR    S^#PR$_IPL,-(SP)
      12    0B A5     DA      0205                   MTPR    UCB$B_FIPL(R5),S^#PR$_IPL
   05 58 A5     08    E2      0209    550          BBSS    #UCB$V_BSY,UCB$W_STS(R5),10$ ;IF SET, THEN DEVICE IS BUSY
             FDEF'     30      020E    551          BSBW    IOC$INITIATE               ;INITIATE I/O FUNCTION
                06     11      0211    552          BRB     20$                        ;
      52    40 A5     DE      0213    553 10$:     MOVAL   UCB$L_IOQFL(R5),R2         ;GET ADDRESS OF I/O QUEUE LISTHEAD
                04     10      0217    554          BSBB    EXE$INSERTIRP              ;INSERT I/O PACKET IN DEVICE QUEUE
                               0219    555 20$:     ENBINT                             ;ENABLE INTERRUPTS
          12    8E    DA      0219                   MTPR    (SP)+,S^#PR$_IPL
                05     0221    556          RSB                                        ;
```

```
                     021D   558          .SBTTL  INSERT I/O PACKET IN QUEUE BY PRIORITY
                     021D   559 ;+
                     021D   560 ; EXE$INSERTIRP - INSERT I/O PACKET IN QUEUE BY PRIORITY
                     021D   561 ;
                     021D   562 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A SPECIFIED QUEUE BY
                     021D   563 ; PRIORITY.
                     021D   564 ;
                     021D   565 ; INPUTS:
                     021D   566 ;
                     021D   567 ;       R2 = ADDRESS OF QUEUE LISTHEAD.
                     021D   568 ;       R3 = ADDRESS OF I/O PACKET.
                     021D   569 ;
                     021D   570 ;       CURRENT IPL MUST BE THE FORK LEVEL OF THE RESPECTIVE DRIVER PROCESS
                     021D   571 ;       OR HIGHER.
                     021D   572 ;
                     021D   573 ; OUTPUTS:
                     021D   574 ;
                     021D   575 ;       THE I/O PACKET IS INSERTED IN THE SPECIFIED QUEUE BY PRIORITY AND
                     021D   576 ;       THE 'Z' CONDITION CODE IS RETURNED TO THE CALLER.
                     021D   577 ;
                     021D   578 ;               'Z' = 1 = ENTRY WAS FIRST ENTRY IN THE QUEUE.
                     021D   579 ;
                     021D   580 ;               'Z' = 0 = ENTRIES WERE ALREADY IN THE QUEUE.
                     021D   581 ;
                     021D   582 ;       R2 AND R3 ARE PRESERVED ACROSS THE CALL.
                     021D   583 ;-
                     021D   584
                     021D   585 EXE$INSERTIRP::                          ;INSERT I/O PACKET IN QUEUE BY PRIORITY
     51   52   D0     021D   586          MOVL    R2,R1                   ;COPY LISTHEAD ADDRESS
51   04   A1   D0     0220   587 10$:     MOVL    IRP$L_IOQBL(R1),R1      ;GET ADDRESS OF NEXT ENTRY
     51   52   D1     0224   588          CMPL    R2,R1                   ;END OF QUEUE?
          07   13     0227   589          BEQL    20$                     ;IF EQL YES
23 A1   23   A3   91   0229   590          CMPB    IRP$B_PRI(R3),IRP$B_PRI(R1) ;NEW ENTRY PRIORITY GREATER?
          F0   1F     022E   591          BLSSU   10$                     ;IF LSS YES
     61   63   0E     0230   592 20$:     INSQUE  IRP$L_IOQFL(R3),IRP$L_IOQFL(R1) ;INSERT PACKET IN I/O QUEUE
          05   0233   593          RSB                                     ;
                     0234   594
                     0234   595          .END
```

```
$ER                             = 00000001     D
$MODULE                           00000000 R   D   02
ACB$B_RMOD                        0000000B     D
ACB$B_TYPE                        0000000A     D
ACB$C_LENGTH                      00000018     D
ACB$K_LENGTH                      00000018     D
ACB$L_AST                         00000010     D
ACB$L_ASTPRM                      00000014     D
ACB$L_ASTQBL                      00000004     D
ACB$L_ASTQFL                      00000000     D
ACB$L_KAST                        00000018     D
ACB$L_PID                         0000000C     D
ACB$M_QUOTA                     = 00000040     D
ACB$V_QUOTA                     = 00000006     D
ACB$W_SIZE                        00000008     D
ACCESS                            0000010C R   D   02
ALLOC                             000000B4 R   D   02
ASTADR                          = 00000014     D
ASTPRM                          = 00000018     D
BUG$CHECK                         ********   X   02
CCB$B_AMOD                        00000009     D
CCB$B_STS                         00000008     D
CCB$C_LENGTH                      00000010     D
CCB$K_LENGTH                      00000010     D
CCB$L_DIRP                        0000000C     D
CCB$L_UCB                         00000000     D
CCB$L_WIND                        00000004     D
CCB$W_IOC                         0000000A     D
CHAN                            = 00000008     D
DDB$B_ACPCLASS                    00000013     D
DDB$B_TYPE                        0000000A     D
DDB$C_LENGTH                      00000034     D
DDB$K_LENGTH                      00000034     D
DDB$L_ACPD                        00000010     D
DDB$L_DDT                         0000000C     D
DDB$L_LINK                        00000000     D
DDB$L_UCB                         00000004     D
DDB$T_DRVNAME                     00000024     D
DDB$T_NAME                        00000014     D
DDB$W_SIZE                        00000008     D
DDT$L_ALTSTART                    0000001C     D
DDT$L_CANCEL                      0000000C     D
DDT$L_FDT                         00000008     D
DDT$L_REGDUMP                     00000010     D
DDT$L_START                       00000000     D
DDT$L_UNITINIT                    00000018     D
DDT$L_UNSOLINT                    00000004     D
DDT$W_DIAGBUF                     00000014     D
DDT$W_ERRORBUF                    00000016     D
DEV$V_FOD                         ********   X   02
DEV$V_FOR                         ********   X   02
DEV$V_MNT                         ********   X   02
DEV$V_SPL                         ********   X   02
DS$AX_SOFTPCB                     ********   X   02
DYN$C_IRP                       = 0000000A     D
EFN                             = 00000004     D
ERROR                             000000A1 R   D   02
EXE$ABORTIO                       0000019A RG  D   02
EXE$ALLOCBUF                      ********     X   02
EXE$ALLOCIRP                      ********     X   02
EXE$ALTQUEPKT                     000001C5 RG  D   02
EXE$FINISHIO                      000001A9 RG  D   02
EXE$FINISHIOC                     000001A7 RG  D   02
EXE$INSERTIRP                     0000021D RG  D   02
EXE$INSIOQ                        00000202 RG  D   02
EXE$QIOACPPKT                     000001E7 RG  D   02
EXE$QIODRVPKT                     000001C1 RG  D   02
EXE$QIORETURN                     000001F9 RG  D   02
FDTACT                          = 00000010     D
FUNC                            = 0000000C     D
GTPKT                             000000BF R   D   02
IO$M_FCODE                        ********     X   02
IO$_LOGICAL                       ********     X   02
IO$_PHYSICAL                      ********     X   02
IO$_READLBLK                      ********     X   02
IO$_READVBLK                      ********     X   02
IO$_WRITEVBLK                     ********     X   02
IOC$GL_IRPFL                      ********     X   02
IOC$GL_PSBL                       ********     X   02
IOC$INITIATE                      ********     X   02
IOC$VERIFYCHAN                    ********     X   02
IOSB                            = 00000010     D
IOTYPE                          = 00000008     D
IPL$_IOPOST                     = 00000004     D
IRP$B_CARCON                      00000038     D
IRP$B_EFN                         00000022     D
IRP$B_PRI                         00000023     D
IRP$B_RMOD                        0000000B     D
IRP$B_TYPE                        0000000A     D
IRP$C_LENGTH                      0000005C     D
IRP$K_LENGTH                      0000005C     D
IRP$L_ARB                         00000050     D
IRP$L_AST                         00000010     D
IRP$L_ASTPRM                      00000014     D
IRP$L_DIAGBUF                     00000044     D
IRP$L_EXTEND                      0000004C     D
IRP$L_IOQBL                       00000004     D
IRP$L_IOQFL                       00000000     D
IRP$L_IOSB                        00000024     D
IRP$L_IOST1                       00000034     D
IRP$L_IOST2                       00000038     D
IRP$L_MEDIA                       00000034     D
IRP$L_PID                         0000000C     D
IRP$L_SEGVBN                      00000040     D
IRP$L_SEQNUM                      00000048     D
IRP$L_SVAPTE                      0000002C     D
IRP$L_TT_TERM                     00000038     D
IRP$L_UCB                         0000001C     D
IRP$L_WIND                        00000018     D
IRP$M_BUFIO                     = 00000001     D
IRP$M_DIAGBUF                   = 00000080     D
IRP$M_PHYSIO                    = 00000100     D
IRP$Q_NT_PRVMSK                   0000003C     D
IRP$W_ABCNT                       0000003C     D
```

IRP$W_BCNT		00000032	D	PCB$W_PRCCNT		00000042	D			
IRP$W_BOFF		00000030	D	PCB$W_SIZE		00000008	D			
IRP$W_CHAN		00000028	D	PCB$W_STATE		0000002C	D			
IRP$W_FUNC		00000020	D	PCB$W_TMBU		00000032	D			
IRP$W_OBCNT		0000003E	D	PR$_IPL	=	00000012	D			
IRP$W_SIZE		00000008	D	PR$_SIRR	=	00000014	D			
IRP$W_STS		0000002A	D	PSL$S_PRVMOD	=	00000002	D			
IRP$W_TT_PRMPT		0000003C	D	PSL$V_PRVMOD	=	00000016	D			
LEGAL	=	00000000	D	QIORETURN		000001FE	R	D	02	
P1	=	0000001C	D	SIZ...	=	00000001		D		
P2	=	00000020	D	SS$_ACCVIO		********		X	02	
P3	=	00000024	D	SS$_DEVOFFLINE		********		X	02	
P4	=	00000028	D	SS$_ILLIOFUNC		********		X	02	
P5	=	0000002C	D	SS$_NORMAL		********		X	02	
P6	=	00000030	D	SUCCES		000000C8	R	D	02	
PCB$B_ASTACT		0000000C	D	SYS$CLREF		********		X	02	
PCB$B_ASTEN		0000000D	D	SYS$SETEF		********		X	02	
PCB$B_PRI		0000000B	D	UCB$B_AMOD		00000053		D		
PCB$B_PRIB		0000002F	D	UCB$B_CEX		00000077		D		
PCB$B_TYPE		0000000A	D	UCB$B_CM1		0000004A		D		
PCB$B_WEFC		0000002E	D	UCB$B_CM2		0000004B		D		
PCB$C_LENGTH		0000008C	D	UCB$B_DEVCLASS		00000038		D		
PCB$K_LENGTH		0000008C	D	UCB$B_DEVTYPE		00000039		D		
PCB$L_ARB		00000084	D	UCB$B_DIPL		00000052		D		
PCB$L_ASTQBL		00000014	D	UCB$B_DX_SCTCNT		000000A6		D		
PCB$L_ASTQFL		00000010	D	UCB$B_ERTCNT		00000070		D		
PCB$L_EFC2P		00000058	D	UCB$B_ERTMAX		00000071		D		
PCB$L_EFC3P		0000005C	D	UCB$B_FEX		00000076		D		
PCB$L_EFCS		00000050	D	UCB$B_FIPL		0000000B		D		
PCB$L_EFCU		00000054	D	UCB$B_LOCSRV		0000003C		D		
PCB$L_EFWM		0000004C	D	UCB$B_OFFNDX		00000094		D		
PCB$L_JIB		00000078	D	UCB$B_OFFRTC		00000095		D		
PCB$L_OWNER		0000001C	D	UCB$B_REMSRV		0000003D		D		
PCB$L_PHD		00000064	D	UCB$B_SECTORS		0000003C		D		
PCB$L_PHYPCB		00000018	D	UCB$B_SLAVE		00000074		D		
PCB$L_PID		00000060	D	UCB$B_SPR		00000075		D		
PCB$L_PQB		0000004C	D	UCB$B_STATE		00000052		D		
PCB$L_SQBL		00000004	D	UCB$B_TRACKS		0000003D		D		
PCB$L_SQFL		00000000	D	UCB$B_TT_CRFILL		0000009D		D		
PCB$L_STS		00000024	D	UCB$B_TT_DECRF		000000A1		D		
PCB$L_UIC		00000088	D	UCB$B_TT_DELFF		000000A2		D		
PCB$L_WSSWP		00000020	D	UCB$B_TT_DESPEE		000000A0		D		
PCB$L_WTIME		00000028	D	UCB$B_TT_DETYPE		000000A4		D		
PCB$Q_PRIV		0000007C	D	UCB$B_TT_LFFILL		0000009E		D		
PCB$T_LNAME		00000068	D	UCB$B_TT_SPEED		0000009C		D		
PCB$T_TERMINAL		00000044	D	UCB$B_TYPE		0000000A		D		
PCB$W_APTCNT		00000030	D	UCB$B_VERTSZ		0000003F		D		
PCB$W_ASTCNT		00000038	D	UCB$C_LENGTH		00000074		D		
PCB$W_BIOCNT		0000003A	D	UCB$C_MB_LENGTH		00000090		D		
PCB$W_BIOLM		0000003C	D	UCB$C_TT_LENGTH		000000BC		D		
PCB$W_DIOCNT		0000003E	D	UCB$K_LENGTH		00000074		D		
PCB$W_DIOLM		00000040	D	UCB$K_MB_LENGTH		00000090		D		
PCB$W_GPGCNT		00000034	D	UCB$K_TT_LENGTH		000000BC		D		
PCB$W_GRP		0000008A	D	UCB$L_AMB		00000054		D		
PCB$W_MEM		00000088	D	UCB$L_ASTQBL		00000010		D		
PCB$W_MTXCNT		0000000E	D	UCB$L_ASTQFL		0000000C		D		
PCB$W_PPGCNT		00000036	D	UCB$L_CPID		0000005C		D		

UCB$L_CRB		00000020	D	UCB$W_DIRSEQ		00000088	D
UCB$L_DDB		00000024	D	UCB$W_DSTADDR		00000018	D
UCB$L_DEVCHAR		00000034	D	UCB$W_DX_BCR		000000A4	D
UCB$L_DEVDEPEND		0000003C	D	UCB$W_EC1		00000090	D
UCB$L_DPC		00000080	D	UCB$W_EC2		00000092	D
UCB$L_DUETIM		0000005C	D	UCB$W_ERRCNT		00000072	D
UCB$L_DX_BFPNT		0000009C	D	UCB$W_FUNC		0000007E	D
UCB$L_DX_BUF		00000098	D	UCB$W_MB_SEED		00000000	D
UCB$L_DX_RXDB		000000A0	D	UCB$W_MSGCNT		00000016	D
UCB$L_EMB		00000078	D	UCB$W_MSGMAX		00000014	D
UCB$L_FIRST		00000014	D	UCB$W_NT_CHAN		0000007C	D
UCB$L_FPC		0000000C	D	UCB$W_OFFSET		0000008A	D
UCB$L_FQBL		00000004	D	UCB$W_REFC		00000050	D
UCB$L_FQFL		00000000	D	UCB$W_SIZE		00000008	D
UCB$L_FR3		00000010	D	UCB$W_SRCADDR		0000001A	D
UCB$L_FR4		00000014	D	UCB$W_STS		00000058	D
UCB$L_IOQBL		00000044	D	UCB$W_TT_DESIZE		000000A5	D
UCB$L_IOQFL		00000040	D	UCB$W_UNIT		00000048	D
UCB$L_IRP		0000004C	D	UCB$W_VPROT		0000001A	D
UCB$L_LINK		0000002C	D	VCB$B_BLOCKFACT		00000052	D
UCB$L_LOGADR		00000064	D	VCB$B_CUR_RVN		0000002F	D
UCB$L_MAXBLOCK		00000084	D	VCB$B_EOFDELTA		0000004E	D
UCB$L_MB_MBX		0000007C	D	VCB$B_IBMAPSIZE		00000038	D
UCB$L_MB_PORT		0000008C	D	VCB$B_IBMAPVBN		0000003A	D
UCB$L_MB_RAST		00000078	D	VCB$B_LRU_LIM		00000049	D
UCB$L_MB_SHB		00000080	D	VCB$B_QNAMECNT		0000000B	D
UCB$L_MB_WAST		00000074	D	VCB$B_RESFILES		0000004F	D
UCB$L_MB_WIOQBL		00000088	D	VCB$B_SBMAPSIZE		00000039	D
UCB$L_MB_WIOQFL		00000084	D	VCB$B_SBMAPVBN		0000003B	D
UCB$L_MEDIA		0000008C	D	VCB$B_STATUS		0000000B	D
UCB$L_NT_DATSSB		00000074	D	VCB$B_STATUS2		00000053	D
UCB$L_NT_INTSSB		00000078	D	VCB$B_TM		0000002E	D
UCB$L_OPCNT		00000060	D	VCB$B_TYPE		0000000A	D
UCB$L_OWNUIC		0000001C	D	VCB$B_WINDOW		00000048	D
UCB$L_PID		00000028	D	VCB$C_COMLEN		00000024	D
UCB$L_RQBL		00000004	D	VCB$C_LENGTH		00000064	D
UCB$L_RQFL		00000000	D	VCB$C_MRKLEN		0000000B	D
UCB$L_SVAPTE		00000068	D	VCB$K_COMLEN		00000024	D
UCB$L_SVPN		00000064	D	VCB$K_LENGTH		00000064	D
UCB$L_TT_DECHAR		000000A8	D	VCB$K_MRKLEN		0000000B	D
UCB$L_TT_RDUE		0000008C	D	VCB$L_AQB		00000010	D
UCB$L_TT_RTIMOU		000000B8	D	VCB$L_BLOCKBL		00000004	D
UCB$L_VCB		00000030	D	VCB$L_BLOCKFL		00000000	D
UCB$T_PARTNER		0000000C	D	VCB$L_CACHE		00000058	D
UCB$V_BSY	=	00000008	D	VCB$L_CUR_FID		00000024	D
UCB$V_ONLINE	=	00000004	D	VCB$L_FCBBL		00000004	D
UCB$W_BCNT		0000006E	D	VCB$L_FCBFL		00000000	D
UCB$W_BCR		00000096	D	VCB$L_FREE		00000040	D
UCB$W_BOFF		0000006C	D	VCB$L_HOME2LBN		00000028	D
UCB$W_BUFQUO		00000018	D	VCB$L_HOMELBN		00000024	D
UCB$W_BYTESTOGO		0000003E	D	VCB$L_IBMAPLBN		00000030	D
UCB$W_CHARGE		0000004A	D	VCB$L_IXHDR2LBN		0000002C	D
UCB$W_CYLINDERS		0000003E	D	VCB$L_MAXFILES		00000044	D
UCB$W_DA		0000008C	D	VCB$L_MVL		00000034	D
UCB$W_DC		0000008E	D	VCB$L_QUOCACHE		0000005C	D
UCB$W_DEVBUFSIZ		0000003A	D	VCB$L_QUOTAFCB		00000054	D
UCB$W_DEVSTS		0000005A	D	VCB$L_RVT		00000020	D

```
VCB$L_SBMAPLBN                  00000034    D
VCB$L_START_FID                 00000028    D
VCB$L_ST_RECORD                 00000030    D
VCB$L_USRLBLAST                 00000044    D
VCB$L_VPBL                      00000040    D
VCB$L_VPFL                      0000003C    D
VCB$L_WCB                       00000038    D
VCB$T_QNAME                     0000000C    D
VCB$T_VOLNAME                   00000014    D
VCB$W_CLUSTER                   0000003C    D
VCB$W_CUR_NUM                   00000024    D
VCB$W_CUR_SEQ                   00000026    D
VCB$W_EXTEND                    0000003E    D
VCB$W_FILEPROT                  0000004A    D
VCB$W_MCOUNT                    0000004C    D
VCB$W_MODE                      0000002C    D
VCB$W_PENDERR                   00000062    D
VCB$W_QUOSIZE                   00000060    D
VCB$W_RECORDSZ                  00000050    D
VCB$W_RVN                       0000000E    D
VCB$W_SIZE                      00000008    D
VCB$W_START_NUM                 00000028    D
VCB$W_START_SEQ                 0000002A    D
VCB$W_TRANS                     0000000C    D
VMS$QIO                         00000007 RG D  02
WCB$B_ACCESS                    0000000B    D
WCB$B_TYPE                      0000000A    D
WCB$C_LENGTH                    00000024    D
WCB$C_MAP                       00000024    D
WCB$K_LENGTH                    00000024    D
WCB$K_MAP                       00000024    D
WCB$L_FCB                       00000018    D
WCB$L_LBN                       00000002    D
WCB$L_ORGUCB                    00000010    D
WCB$L_P1_LBN                    00000026    D
WCB$L_P2_LBN                    0000002C    D
WCB$L_PID                       0000000C    D
WCB$L_PREVLBN                   FFFFFFFC    D
WCB$L_RVT                       0000001C    D
WCB$L_STVBN                     00000020    D
WCB$L_WLBL                      00000004    D
WCB$L_WLFL                      00000000    D
WCB$W_ACON                      00000014    D
WCB$W_COUNT                     00000000    D
WCB$W_NMAP                      00000016    D
WCB$W_P1_COUNT                  00000024    D
WCB$W_P2_COUNT                  0000002A    D
WCB$W_PREVCOUNT                 FFFFFFFA    D
WCB$W_REFCNT                    0000000E    D
WCB$W_SIZE                      00000008    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000234 (564.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$ER	=00000001	135 (1)	
$MODULE	00000000-R	135 (1)	
ACB$M_QUOTA	=00000040		#-354 (1)
ACB$V_QUOTA	=00000006		#-422 (1)
ACCESS	0000010C-R	301 (1)	
ALLOC	000000B4-R	265 (1)	#-269 (1)
ASTADR	=00000014	118 (1)	#-285 (1)
ASTPRM	=00000018	119 (1)	
BUG$CHECK	00000000-XR		528 (1)
CCB$L_UCB	00000000		#-223 (1)
CCB$L_WIND	00000004		#-247 (1) #-286 (1)
CCB$W_IOC	0000000A		#-277 (1)
CHAN	=00000008	115 (1)	#-219 (1)
DDB$L_DDT	0000000C		#-232 (1) #-336 (1) #-491 (1)
DDT$L_ALTSTART	0000001C		#-492 (1)
DDT$L_FDT	00000008		#-233 (1)
DDT$W_DIAGBUF	00000014		#-337 (1)
DEV$V_FOD	00000000-XR		#-305 (1)
DEV$V_FOR	00000000-XR		#-314 (1)
DEV$V_MNT	00000000-XR		#-313 (1)
DEV$V_SPL	00000000-XR		#-227 (1)
DS$AX_SOFTPCB	00000000-XR		213 (1)
DYN$C_IRP	=0000000A		#-281 (1)
EFN	=00000004	114 (1)	#-214 (1) #-256 (1) #-289 (1)
ERROR	000000A1-R	254 (1)	#-217 (1) #-222 (1) #-250 (1) #-252 (1)
			#-267 (1)
EXE$ABORTIO	0000019A-R	420 (1)	#-324 (1)
EXE$ALLOCBUF	00000000-XR		340 (1)
EXE$ALLOCIRP	00000000-XR		265 (1)
EXE$ALTQUEPKT	000001C5-R	488 (1)	
EXE$FINISHIO	000001A9-R	427 (1)	
EXE$FINISHIOC	000001A7-R	425 (1)	
EXE$INSERTIRP	0000021D-R	585 (1)	#-554 (1)
EXE$INSIOQ	00000202-R	548 (1)	#-457 (1)
EXE$QIOACPPKT	000001E7-R	527 (1)	
EXE$QIODRVPKT	000001C1-R	456 (1)	
EXE$QIORETURN	000001F9-R	529 (1)	#-458 (1)
FDTACT	=00000010	133 (1)	#-374 (1)
FUNC	=0000000C	116 (1)	#-225 (1)
GTPKT	000000BF-R	268 (1)	#-247 (1)
IO$M_FCODE	00000000-XR		#-226 (1)
IO$_LOGICAL	00000000-XR		#-228 (1)
IO$_PHYSICAL	00000000-XR		#-330 (1)
IO$_READLBLK	00000000-XR		#-320 (1) #-321 (1)
IO$_READVBLK	00000000-XR		#-303 (1) #-320 (1) #-321 (1)
IO$_WRITEVBLK	00000000-XR		#-301 (1)
IOC$GL_IRPFL	00000000-XR		268 (1)
IOC$GL_PSBL	00000000-XR		431 (1)
IOC$INITIATE	00000000-XR		#-551 (1)
IOC$VERIFYCHAN	00000000-XR		#-220 (1)

```
IOSB                            =00000010      117     (1)     #-238    (1)     #-291    (1)
IOTYPE                          =00000008      132     (1)       244    (1)       293    (1)
IPL$_IOPOST                     =00000004                      #-433    (1)
IRP$B_PRI                        00000023                      #-590    (1)
IRP$B_RMOD                       0000000B                      #-354    (1)       422    (1)
IRP$C_LENGTH                     0000005C                      #-280    (1)
IRP$L_AST                        00000010                      #-352    (1)
IRP$L_DIAGBUF                    00000044                      #-343    (1)
IRP$L_IOQBL                      00000004                      #-587    (1)
IRP$L_IOQFL                      00000000                       592     (1)
IRP$L_IOSB                       00000024                      #-421    (1)
IRP$L_MEDIA                      00000034                      #-429    (1)
IRP$L_WIND                       00000018                      #-311    (1)
IRP$M_BUFIO                     =00000001                      #-294    (1)
IRP$M_DIAGBUF                   =00000080                      #-346    (1)
IRP$M_PHYSIO                    =00000100                      #-332    (1)
IRP$W_FUNC                       00000020                      #-321    (1)
IRP$W_SIZE                       00000008                      #-279    (1)
IRP$W_STS                        0000002A                      #-332    (1)     #-346    (1)
LEGAL                           =00000000      131     (1)       236    (1)
P1                              =0000001C      120     (1)     #-375    (1)
P2                              =00000020      121     (1)
P3                              =00000024      122     (1)
P4                              =00000028      123     (1)
P5                              =0000002C      124     (1)
P6                              =00000030      125     (1)     #-333    (1)
PCB$L_PID                        00000060                      #-284    (1)
PCB$W_ASTCNT                     00000038                      #-423    (1)
PCB$W_BIOCNT                     0000003A                       243     (1)
PCB$W_DIOCNT                     0000003E                       245     (1)
PR$_IPL                         =00000012                      #-254    (1)     #-489    (1)     #-502    (1)     #-532    (1)
                                                               #-549    (1)     #-555    (1)
PR$_SIRR                        =00000014                      #-433    (1)
PSL$S_PRVMOD                    =00000002                      #-282    (1)
PSL$V_PRVMOD                    =00000016                      #-282    (1)
QIORETURN                        000001FE-R     531     (1)     #-432    (1)     #-434    (1)
SS$_ACCVIO                       00000000-XR                   #-253    (1)
SS$_DEVOFFLINE                   00000000-XR                   #-251    (1)
SS$_ILLIOFUNC                    00000000-XR                   #-249    (1)
SS$_NORMAL                       00000000-XR                   #-430    (1)     #-530    (1)
SUCCES                           000000C8-R     276     (1)     #-266    (1)
SYS$CLREF                        00000000-XR                    215     (1)
SYS$SETEF                        00000000-XR                    257     (1)
UCB$B_FIPL                       0000000B                      #-489    (1)     #-549    (1)
UCB$L_AMB                        00000054                      #-230    (1)
UCB$L_DDB                        00000024                      #-231    (1)     #-335    (1)     #-490    (1)
UCB$L_DEVCHAR                    00000034                       227     (1)       305    (1)       313    (1)       314    (1)
UCB$L_IOQFL                      00000040                       553     (1)
UCB$L_OPCNT                      00000060                      #-428    (1)
UCB$V_BSY                       =00000008                      #-550    (1)
UCB$V_ONLINE                    =00000004                      #-237    (1)
UCB$W_STS                        00000058                       237     (1)       550    (1)
VMS$QIO                          00000007-R     212     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$ACBDEF	2	87 (1)	87 (1)
$AQBDEF	2	88 (1)	88 (1)
$CADEF	1	89 (1)	89 (1)
$CCBDEF	1	90 (1)	90 (1)
$DDBDEF	1	91 (1)	91 (1)
$DDTDEF	1	92 (1)	92 (1)
$DEFINI	1	87 (1)	100 (1) 101 (1) 102 (1) 103 (1) 104 (1)
			105 (1) 106 (1) 87 (1) 88 (1) 89 (1)
			90 (1) 91 (1) 92 (1) 93 (1) 94 (1)
			95 (1) 96 (1) 97 (1) 98 (1) 99 (1)
$DYNDEF	2	93 (1)	93 (1)
$IPLDEF	1	94 (1)	94 (1)
$IRPDEF	4	95 (1)	95 (1)
$PCBDEF	4	96 (1)	96 (1)
$PHDDEF	6	97 (1)	97 (1)
$PRDEF	5	98 (1)	98 (1)
$PRIDEF	1	99 (1)	99 (1)
$PRVDEF	4	100 (1)	100 (1)
$PSLDEF	2	101 (1)	101 (1)
$RSNDEF	1	102 (1)	102 (1)
$SECDEF	3	103 (1)	103 (1)
$UCBDEF	10	104 (1)	104 (1)
$VCBDEF	6	105 (1)	105 (1)
$WCBDEF	3	106 (1)	106 (1)
BUG_CHECK	1	528 (1)	528 (1)
DSBINT	1	489 (1)	489 (1) 549 (1)
ENBINT	1	502 (1)	502 (1) 555 (1)
IFNOWRT	1	240 (1)	240 (1)
MODNAM	1	135 (1)	135 (1)
SETIPL	1	254 (1)	254 (1) 532 (1)
SOFTINT	1	433 (1)	433 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	235	(1)	279	(1)	374	(1)	375	(1)	376	(1)	380	(1)	495	(1)
BBC	00E1	227	(1)	236	(1)	237	(1)	293	(1)	305	(1)	313	(1)	314	(1)
		377	(1)												
BBCC	00E5	422	(1)												
BBS	00E0	234	(1)	244	(1)	379	(1)	494	(1)						
BBSS	00E2	550	(1)												
BEQL	0013	239	(1)	240	(1)	334	(1)	338	(1)	353	(1)	493	(1)	589	(1)
BGEQ	0018	229	(1)												
BGTRU	001A	302	(1)												
BICL3	00CB	226	(1)												
BISB	0088	354	(1)												
BISW	00A8	294	(1)	332	(1)	346	(1)								
BLBC	00E9	222	(1)	247	(1)	342	(1)								
BLBS	00E8	216	(1)	266	(1)										
BLSS	0019	331	(1)												
BLSSU	001F	304	(1)	591	(1)										
BNEQ	0012	312	(1)	432	(1)										
BRB	0011	248	(1)	250	(1)	252	(1)	267	(1)	322	(1)	382	(1)	424	(1)
		434	(1)	458	(1)	552	(1)								
BRW	0031	217	(1)	324	(1)										
BSBB	0010	457	(1)	554	(1)										
BSBW	0030	220	(1)	551	(1)										
BVS	001D	269	(1)												
CALLS	00FB	215	(1)	257	(1)										
CLRB	0094	290	(1)												
CLRL	00D4	421	(1)	426	(1)										
CLRQ	007C	241	(1)	295	(1)										
CMPB	0091	590	(1)												
CMPL	00D1	228	(1)	301	(1)	303	(1)	330	(1)	588	(1)				
DECW	00B7	276	(1)												
EXTZV	00EF	282	(1)												
INCL	00D6	283	(1)	428	(1)										
INCW	00B6	277	(1)	423	(1)										
INSQUE	000E	431	(1)	592	(1)										
JSB	0016	265	(1)	340	(1)	381	(1)	499	(1)	528	(1)				
MFPR	00DB	489	(1)	549	(1)										
MOVAB	009E	344	(1)												
MOVAL	00DE	213	(1)	243	(1)	245	(1)	553	(1)						
MOVB	0090	281	(1)	289	(1)										
MOVL	00D0	221	(1)	223	(1)	224	(1)	230	(1)	231	(1)	232	(1)	233	(1)
		238	(1)	246	(1)	278	(1)	284	(1)	286	(1)	287	(1)	291	(1)
		333	(1)	335	(1)	336	(1)	343	(1)	345	(1)	378	(1)	490	(1)
		491	(1)	492	(1)	586	(1)	587	(1)						
MOVPSL	00DC	242	(1)												
MOVQ	007D	285	(1)	429	(1)										
MOVW	00B0	280	(1)	288	(1)										
MOVZBL	009A	214	(1)	256	(1)										
MOVZWL	003C	219	(1)	225	(1)	249	(1)	251	(1)	253	(1)	292	(1)	337	(1)
		430	(1)	530	(1)										
MTPR	00DA	254	(1)	433	(1)	489	(1)	502	(1)	532	(1)	549	(1)	555	(1)

```
POPL            8ED0      341    (1)
POPR            00BA      258    (1)
PROBEW          000D      240    (1)
PUSHL           00DD      255    (1)    339    (1)
REMQUE          000F      268    (1)
RET             0004      259    (1)    533    (1)
RSB             0005      503    (1)    556    (1)    593    (1)
SUBL            00C2      320    (1)
SUBW            00A2      321    (1)
TSTL            00D5      311    (1)    352    (1)
```

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	135	(1)	528	(1)												
.DSABL	270	(1)	435	(1)												
.ENABL	211	(1)	419	(1)												
.END	595	(1)														
.ENDC	254	(1)	489	(1)	502	(1)	532	(1)	549	(1)	555	(1)				
.ENDM	100	(1)	101	(1)	102	(1)	103	(1)	104	(1)	105	(1)	106	(1)	135	(1)
	240	(1)	254	(1)	433	(1)	489	(1)	502	(1)	528	(1)	87	(1)	88	(1)
	89	(1)	90	(1)	91	(1)	92	(1)	93	(1)	94	(1)	95	(1)	96	(1)
	97	(1)	98	(1)	99	(1)										
.ENTRY	212	(1)														
.IDENT	5	(1)														
.IF	254	(1)	489	(1)	502	(1)	532	(1)	549	(1)	555	(1)				
.IFF	254	(1)	489	(1)	502	(1)	532	(1)	549	(1)	555	(1)				
.NLIST	6	(1)														
.NOCROSS	100	(1)	101	(1)	102	(1)	103	(1)	104	(1)	105	(1)	106	(1)	87	(1)
	88	(1)	89	(1)	90	(1)	91	(1)	92	(1)	93	(1)	94	(1)	95	(1)
	96	(1)	97	(1)	98	(1)	99	(1)								
.PAGE	136	(1)	383	(1)	384	(1)	436	(1)	459	(1)	504	(1)	534	(1)	557	(1)
	80	(1)														
.PSECT	134	(1)														
.RESTORE	100	(1)	101	(1)	102	(1)	103	(1)	104	(1)	105	(1)	106	(1)	87	(1)
	88	(1)	89	(1)	90	(1)	91	(1)	92	(1)	93	(1)	94	(1)	95	(1)
	96	(1)	97	(1)	98	(1)	99	(1)								
.SAVE	100	(1)	101	(1)	102	(1)	103	(1)	104	(1)	105	(1)	106	(1)	87	(1)
	88	(1)	89	(1)	90	(1)	91	(1)	92	(1)	93	(1)	94	(1)	95	(1)
	96	(1)	97	(1)	98	(1)	99	(1)								
.SBTTL	137	(1)	385	(1)	437	(1)	460	(1)	505	(1)	535	(1)	558	(1)	81	(1)
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	10	#-214 (1) #-219 (1) #-225 (1) #-238 (1) #-256 (1) #-285 (1)
		#-289 (1) #-291 (1) #-333 (1) #-375 (1)
R0	27	#-216 (1) #-219 (1) #-222 (1) #-231 (1) #-232 (1) #-233 (1)
		#-235 (1) #-249 (1) #-251 (1) #-253 (1) #-255 (1) #-258 (1)
		#-266 (1) #-342 (1) #-378 (1) 379 (1) #-380 (1) 381 (1)
		#-429 (1) #-430 (1) #-490 (1) #-491 (1) #-492 (1) #-495 (1)
		#-530 (1)
R1	20	#-221 (1) #-238 (1) 240 (1) #-241 (1) #-335 (1) #-336 (1)
		#-337 (1) #-426 (1) #-492 (1) 494 (1) #-495 (1) 499 (1)
		#-586 (1) #-587 (1) #-588 (1) #-590 (1) 592 (1)
R10	4	212 (1) #-225 (1) #-226 (1) #-288 (1)
R11	5	212 (1) #-243 (1) #-245 (1) #-246 (1) #-276 (1)
R2	28	212 (1) #-224 (1) #-246 (1) #-268 (1) #-278 (1) #-279 (1)
		#-280 (1) #-281 (1) #-282 (1) #-283 (1) #-284 (1) #-285 (1)
		#-286 (1) #-287 (1) #-288 (1) #-289 (1) #-290 (1) #-291 (1)
		#-292 (1) #-294 (1) #-295 (1) #-343 (1) 344 (1) #-345 (1)
		#-553 (1) #-586 (1) #-588 (1)
R3	17	212 (1) #-278 (1) #-311 (1) #-321 (1) #-332 (1) #-339 (1)
		#-341 (1) #-343 (1) #-346 (1) #-352 (1) #-354 (1) #-421 (1)
		422 (1) #-429 (1) 431 (1) #-590 (1) 592 (1)
R4	6	212 (1) #-213 (1) 243 (1) 245 (1) #-284 (1) #-423 (1)
R5	18	212 (1) #-223 (1) 227 (1) #-230 (1) #-231 (1) 237 (1)
		#-287 (1) 305 (1) 313 (1) 314 (1) #-335 (1) #-428 (1)
		#-489 (1) #-490 (1) #-549 (1) 550 (1) 553 (1)
R6	6	212 (1) #-221 (1) #-223 (1) #-247 (1) #-277 (1) #-286 (1)
R7	11	212 (1) #-226 (1) #-228 (1) #-236 (1) #-244 (1) #-293 (1)
		#-301 (1) #-303 (1) #-320 (1) #-330 (1) #-377 (1)
R8	12	212 (1) #-233 (1) 234 (1) #-235 (1) 236 (1) 244 (1)
		293 (1) #-374 (1) #-376 (1) 377 (1) #-378 (1) #-380 (1)
R9	5	212 (1) #-224 (1) #-292 (1) #-333 (1) #-345 (1)
SP	8	#-214 (1) #-242 (1) #-256 (1) 282 (1) #-489 (1) #-502 (1)
		#-549 (1) #-555 (1)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.19
Command processing	890	00:00:00.24	00:00:00.77
Pass 1	973	00:00:03.55	00:00:05.43
Symbol table sort	0	00:00:00.28	00:00:00.28
Pass 2	56	00:00:00.97	00:00:01.97
Symbol table output	2	00:00:00.05	00:00:00.25
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	6	00:00:00.15	00:00:00.24
Assembler run totals	1953	00:00:05.28	00:00:09.14

The working set limit was 2900 pages.

220656 bytes (431 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1072 non-local and 26 local symbols.
595 source lines were read in Pass 1, producing 72 object records in Pass 2.
141 pages of virtual memory were used to define 36 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	11
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	2
SYS$COMMON:[SYSLIB]LIB.MLB;2	10
SYS$COMMON:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	32

1549 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

```
MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:QIOREQ.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 61 INITIALIZE DRIVER DATA BASE
(1) 90 REINITIALIZE DRIVER DATA BASE
(1) 125 LOCAL SUBROUTINE TO EXECUTE OPERATION CODE
(1) 213 LOCAL SUBROUTINE TO LOCATE DATA STRUCTURE TYPE

```
                     0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element RELOCDRV.MAR
                     0000     2 ; *1     6-FEB-1986 15:22:15 DS ""
                     0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element RELOCDRV.MAR
                     0000     4         .TITLE  RELOCDRV        *** RELOCDRV relocate I/O driver
                     0000     5         .IDENT  /01/
                00000000     6         .PSECT  SEP,SHR,EXE,WRT,LONG
                     0000     7
                     0000     8
                     0000     9 ;
                     0000    10 ;*****************************************************************************
                     0000    11 ;*                                                                           *
                     0000    12 ; COPYRIGHT (C) 1978, 1980
                     0000    13 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                     0000    14 ;*                                                                           *
                     0000    15 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                     0000    16 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                     0000    17 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                     0000    18 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                     0000    19 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                     0000    20 ;*  TRANSFERRED.                                                            *
                     0000    21 ;*                                                                           *
                     0000    22 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                     0000    23 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
                     0000    24 ;*  CORPORATION.                                                            *
                     0000    25 ;*                                                                           *
                     0000    26 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                     0000    27 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                     0000    28 ;*                                                                           *
                     0000    29 ;*****************************************************************************
                     0000    30 ;
                     0000    31 ; D. N. CUTLER 16-JUN-78
                     0000    32 ;
                     0000    33 ;       DS5.4   John Ciukaj     5-May-1980      Version 5.4
                     0000    34 ;                 Used Vax/Vms version 2.0 RELOCDRV.MAR and created
                     0000    35 ;                 a UPD file with the edits necessary to reflect the
                     0000    36 ;                 changes exhibited in Supervisor version 5.3 module
                     0000    37 ;                 RELOCDRV.
                     0000    38 ;
                     0000    39 ;
                     0000    40 ; RELOCATE DRIVER DATABASE
                     0000    41 ;
                     0000    42 ; MACRO LIBRARY CALLS
                     0000    43 ;
                     0000    44
                     0000    45         $CRBDEF                             ;DEFINE CRB OFFSETS
                     0000               .SAVE   LOCAL_BLOCK
                     0000               .IIF    NB,CRB$L_WQFL,  CRB$L_WQFL:
          00000004   0000                       .BLKL   1
                     0004               .IIF    NB,CRB$L_WQBL,  CRB$L_WQBL:
          00000008   0004                       .BLKL   1
                     0008               .IIF    NB,CRB$W_SIZE,  CRB$W_SIZE:
          0000000A   0008                       .BLKW   1
                     000A               .IIF    NB,CRB$B_TYPE,  CRB$B_TYPE:
          0000000B   000A                       .BLKB   1
          0000000C   000B                       .BLKB   1
                     000C               .IIF    NB,CRB$W_REFC,  CRB$W_REFC:
          0000000E   000C                       .BLKW   1
```

```
                     000E                .IIF    NB,CRB$B_MASK,  CRB$B_MASK:
0000000F  000E                                   .BLKB   1
00000010  000F                                   .BLKB   1
                     0010                .IIF    NB,CRB$L_LINK,  CRB$L_LINK:
00000014  0010                                   .BLKL   1
                     0014                .IIF    NB,CRB$L_INTD,  CRB$L_INTD:
00000038  0014                                   .BLKL   9
                     0038                .IIF    NB,CRB$C_LENGTH,        CRB$C_LENGTH:
                     0038                .IIF    NB,CRB$K_LENGTH,        CRB$K_LENGTH:
                     0038                .IIF    NB,CRB$L_INTD2, CRB$L_INTD2:
0000005C  0038                                   .BLKL   9
        00000000                         .RESTORE
                     0000    46          $DDBDEF                                 ;DEFINE DDB OFFSETS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000  005C                           .=0
                     0000                .IIF    NB,DDB$L_LINK,  DDB$L_LINK:
00000004  0000                                   .BLKL   1
                     0004                .IIF    NB,DDB$L_UCB,   DDB$L_UCB:
00000008  0004                                   .BLKL   1
                     0008                .IIF    NB,DDB$W_SIZE,  DDB$W_SIZE:
0000000A  0008                                   .BLKW   1
                     000A                .IIF    NB,DDB$B_TYPE,  DDB$B_TYPE:
0000000B  000A                                   .BLKB   1
0000000C  000B                                   .BLKB   1
                     000C                .IIF    NB,DDB$L_DDT,   DDB$L_DDT:
00000010  000C                                   .BLKL   1
                     0010                .IIF    NB,DDB$L_ACPD,  DDB$L_ACPD:
00000013  0010                                   .BLKB   3
                     0013                .IIF    NB,DDB$B_ACPCLASS,      DDB$B_ACPCLASS:
00000014  0013                                   .BLKB   1
                     0014                .IIF    NB,DDB$T_NAME,  DDB$T_NAME:
00000015  0014                                   .BLKB   1
00000024  0015                                   .BLKB   15
                     0024                .IIF    NB,DDB$T_DRVNAME,       DDB$T_DRVNAME:
00000025  0024                                   .BLKB   1
00000034  0025                                   .BLKB   15
                     0034                .IIF    NB,DDB$C_LENGTH,        DDB$C_LENGTH:
                     0034                .IIF    NB,DDB$K_LENGTH,        DDB$K_LENGTH:
        00000000                         .RESTORE
                     0000    47          $DPTDEF                                 ;DEFINE DPT OFFSETS
                     0000                .SAVE   LOCAL_BLOCK
                     0000                .PSECT  $ABS$,ABS
00000000  005C                           .=0
                     0000                .IIF    NB,DPT$L_FLINK, DPT$L_FLINK:
00000004  0000                                   .BLKL   1
                     0004                .IIF    NB,DPT$L_BLINK, DPT$L_BLINK:
00000008  0004                                   .BLKL   1
                     0008                .IIF    NB,DPT$W_SIZE,  DPT$W_SIZE:
0000000A  0008                                   .BLKW   1
                     000A                .IIF    NB,DPT$B_TYPE,  DPT$B_TYPE:
0000000B  000A                                   .BLKB   1
                     000B                .IIF    NB,DPT$B_REFC,  DPT$B_REFC:
0000000C  000B                                   .BLKB   1
                     000C                .IIF    NB,DPT$B_ADPTYPE,       DPT$B_ADPTYPE:
0000000D  000C                                   .BLKB   1
                     000D                .IIF    NB,DPT$B_FLAGS, DPT$B_FLAGS:
```

```
0000000E  000D                                  .BLKB   1
          000E                  .IIF    NB,DPT$W_UCBSIZE,       DPT$W_UCBSIZE:
00000010  000E                                  .BLKW   1
          0010                  .IIF    NB,DPT$W_INITTAB,       DPT$W_INITTAB:
00000012  0010                                  .BLKW   1
          0012                  .IIF    NB,DPT$W_REINITTAB,     DPT$W_REINITTAB:
00000014  0012                                  .BLKW   1
          0014                  .IIF    NB,DPT$W_UNLOAD,        DPT$W_UNLOAD:
00000016  0014                                  .BLKW   1
          0016                  .IIF    NB,DPT$W_MAXUNITS,      DPT$W_MAXUNITS:
00000018  0016                                  .BLKW   1
          0018                  .IIF    NB,DPT$W_VERSION,       DPT$W_VERSION:
0000001A  0018                                  .BLKW   1
0000001E  001A                                  .BLKL   1
          001E                  .IIF    NB,DPT$T_NAME,  DPT$T_NAME:
0000002A  001E                                  .BLKB   12
          002A                  .IIF    NB,DPT$C_LENGTH,        DPT$C_LENGTH:
          002A                  .IIF    NB,DPT$K_LENGTH,        DPT$K_LENGTH:
      00000000                  .RESTORE
          0000     48           $DYNDEF                         ;DEFINE DATA STRUCTURE TYPES
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  005C                  .=0
      00000000                  .RESTORE
          0000     49           $IDBDEF                         ;DEFINE IDB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  005C                  .=0
          0000                  .IIF    NB,IDB$L_CSR,   IDB$L_CSR:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,IDB$L_OWNER, IDB$L_OWNER:
00000008  0004                                  .BLKL   1
          0008                  .IIF    NB,IDB$W_SIZE,  IDB$W_SIZE:
0000000A  0008                                  .BLKW   1
          000A                  .IIF    NB,IDB$B_TYPE,  IDB$B_TYPE:
0000000B  000A                                  .BLKB   1
0000000C  000B                                  .BLKB   1
          000C                  .IIF    NB,IDB$W_UNITS, IDB$W_UNITS:
0000000E  000C                                  .BLKW   1
00000010  000E                                  .BLKW   1
          0010                  .IIF    NB,IDB$L_ADP,   IDB$L_ADP:
00000014  0010                                  .BLKL   1
          0014                  .IIF    NB,IDB$L_UCBLST,        IDB$L_UCBLST:
00000034  0014                                  .BLKL   8
          0034                  .IIF    NB,IDB$C_LENGTH,        IDB$C_LENGTH:
          0034                  .IIF    NB,IDB$K_LENGTH,        IDB$K_LENGTH:
      00000000                  .RESTORE
          0000     50           $UCBDEF                         ;DEFINE UCB OFFSETS
          0000                  .SAVE   LOCAL_BLOCK
          0000                  .PSECT  $ABS$,ABS
00000000  005C                  .=0
          0000                  .IIF    NB,UCB$L_FQFL,  UCB$L_FQFL:
          0000                  .IIF    NB,UCB$L_RQFL,  UCB$L_RQFL:
00000004  0000                                  .BLKL   1
          0004                  .IIF    NB,UCB$L_FQBL,  UCB$L_FQBL:
          0004                  .IIF    NB,UCB$L_RQBL,  UCB$L_RQBL:
00000008  0004                                  .BLKL   1
```

```
                    0008            .IIF    NB,UCB$W_SIZE,  UCB$W_SIZE:
          0000000A  0008                    .BLKW   1
                    000A            .IIF    NB,UCB$B_TYPE,  UCB$B_TYPE:
          0000000B  000A                    .BLKB   1
                    000B            .IIF    NB,UCB$B_FIPL,  UCB$B_FIPL:
          0000000C  000B                    .BLKB   1
                    000C            .IIF    NB,UCB$L_ASTQFL,        UCB$L_ASTQFL:
                    000C            .IIF    NB,UCB$L_FPC,   UCB$L_FPC:
                    000C            .IIF    NB,UCB$T_PARTNER,       UCB$T_PARTNER:
          0000000D  000C                    .BLKB   1
          00000010  000D                    .BLKB   3
                    0010            .IIF    NB,UCB$L_ASTQBL,        UCB$L_ASTQBL:
                    0010            .IIF    NB,UCB$L_FR3,   UCB$L_FR3:
          00000014  0010                    .BLKL   1
                    0014            .IIF    NB,UCB$L_FR4,   UCB$L_FR4:
                    0014            .IIF    NB,UCB$L_FIRST, UCB$L_FIRST:
                    0014            .IIF    NB,UCB$W_MSGMAX,        UCB$W_MSGMAX:
          00000016  0014                    .BLKW   1
                    0016            .IIF    NB,UCB$W_MSGCNT,        UCB$W_MSGCNT:
          00000018  0016                    .BLKW   1
                    0018            .IIF    NB,UCB$W_BUFQUO,        UCB$W_BUFQUO:
                    0018            .IIF    NB,UCB$W_DSTADDR,       UCB$W_DSTADDR:
          0000001A  0018                    .BLKW   1
                    001A            .IIF    NB,UCB$W_VPROT, UCB$W_VPROT:
                    001A            .IIF    NB,UCB$W_SRCADDR,       UCB$W_SRCADDR:
          0000001C  001A                    .BLKW   1
                    001C            .IIF    NB,UCB$L_OWNUIC,        UCB$L_OWNUIC:
          00000020  001C                    .BLKL   1
                    0020            .IIF    NB,UCB$L_CRB,   UCB$L_CRB:
          00000024  0020                    .BLKL   1
                    0024            .IIF    NB,UCB$L_DDB,   UCB$L_DDB:
          00000028  0024                    .BLKL   1
                    0028            .IIF    NB,UCB$L_PID,   UCB$L_PID:
          0000002C  0028                    .BLKL   1
                    002C            .IIF    NB,UCB$L_LINK,  UCB$L_LINK:
          00000030  002C                    .BLKL   1
                    0030            .IIF    NB,UCB$L_VCB,   UCB$L_VCB:
          00000034  0030                    .BLKL   1
                    0034            .IIF    NB,UCB$L_DEVCHAR,       UCB$L_DEVCHAR:
          00000038  0034                    .BLKL   1
                    0038            .IIF    NB,UCB$B_DEVCLASS,      UCB$B_DEVCLASS:
          00000039  0038                    .BLKB   1
                    0039            .IIF    NB,UCB$B_DEVTYPE,       UCB$B_DEVTYPE:
          0000003A  0039                    .BLKB   1
                    003A            .IIF    NB,UCB$W_DEVBUFSIZ,     UCB$W_DEVBUFSIZ:
          0000003C  003A                    .BLKW   1
                    003C            .IIF    NB,UCB$L_DEVDEPEND,     UCB$L_DEVDEPEND:
                    003C            .IIF    NB,UCB$B_SECTORS,       UCB$B_SECTORS:
                    003C            .IIF    NB,UCB$B_LOCSRV,        UCB$B_LOCSRV:
          0000003D  003C                    .BLKB   1
                    003D            .IIF    NB,UCB$B_REMSRV,        UCB$B_REMSRV:
                    003D            .IIF    NB,UCB$B_TRACKS,        UCB$B_TRACKS:
          0000003E  003D                    .BLKB   1
                    003E            .IIF    NB,UCB$W_CYLINDERS,     UCB$W_CYLINDERS:
                    003E            .IIF    NB,UCB$W_BYTESTOGO,     UCB$W_BYTESTOGO:
          0000003F  003E                    .BLKB   1
                    003F            .IIF    NB,UCB$B_VERTSZ,        UCB$B_VERTSZ:
```

```
00000040  003F                                  .BLKB   1
          0040                  .IIF    NB,UCB$L_IOQFL, UCB$L_IOQFL:
00000044  0040                                  .BLKL   1
          0044                  .IIF    NB,UCB$L_IOQBL, UCB$L_IOQBL:
00000048  0044                                  .BLKL   1
          0048                  .IIF    NB,UCB$W_UNIT,  UCB$W_UNIT:
0000004A  0048                                  .BLKW   1
          004A                  .IIF    NB,UCB$W_CHARGE,        UCB$W_CHARGE:
          004A                  .IIF    NB,UCB$B_CM1,   UCB$B_CM1:
0000004B  004A                                  .BLKB   1
          004B                  .IIF    NB,UCB$B_CM2,   UCB$B_CM2:
0000004C  004B                                  .BLKB   1
          004C                  .IIF    NB,UCB$L_IRP,   UCB$L_IRP:
00000050  004C                                  .BLKL   1
          0050                  .IIF    NB,UCB$W_REFC,  UCB$W_REFC:
00000052  0050                                  .BLKW   1
          0052                  .IIF    NB,UCB$B_DIPL,  UCB$B_DIPL:
          0052                  .IIF    NB,UCB$B_STATE, UCB$B_STATE:
00000053  0052                                  .BLKB   1
          0053                  .IIF    NB,UCB$B_AMOD,  UCB$B_AMOD:
00000054  0053                                  .BLKB   1
          0054                  .IIF    NB,UCB$L_AMB,   UCB$L_AMB:
00000058  0054                                  .BLKL   1
          0058                  .IIF    NB,UCB$W_STS,   UCB$W_STS:
0000005A  0058                                  .BLKW   1
          005A                  .IIF    NB,UCB$W_DEVSTS,        UCB$W_DEVSTS:
0000005C  005A                                  .BLKW   1
          005C                  .IIF    NB,UCB$L_CPID,  UCB$L_CPID:
          005C                  .IIF    NB,UCB$L_DUETIM,        UCB$L_DUETIM:
00000060  005C                                  .BLKL   1
          0060                  .IIF    NB,UCB$L_OPCNT, UCB$L_OPCNT:
00000064  0060                                  .BLKL   1
          0064                  .IIF    NB,UCB$L_LOGADR,        UCB$L_LOGADR:
          0064                  .IIF    NB,UCB$L_SVPN,  UCB$L_SVPN:
00000068  0064                                  .BLKL   1
          0068                  .IIF    NB,UCB$L_SVAPTE,        UCB$L_SVAPTE:
0000006C  0068                                  .BLKL   1
          006C                  .IIF    NB,UCB$W_BOFF,  UCB$W_BOFF:
0000006E  006C                                  .BLKW   1
          006E                  .IIF    NB,UCB$W_BCNT,  UCB$W_BCNT:
00000070  006E                                  .BLKW   1
          0070                  .IIF    NB,UCB$B_ERTCNT,        UCB$B_ERTCNT:
00000071  0070                                  .BLKB   1
          0071                  .IIF    NB,UCB$B_ERTMAX,        UCB$B_ERTMAX:
00000072  0071                                  .BLKB   1
          0072                  .IIF    NB,UCB$W_ERRCNT,        UCB$W_ERRCNT:
00000074  0072                                  .BLKW   1
          0074                  .IIF    NB,UCB$C_LENGTH,        UCB$C_LENGTH:
          0074                  .IIF    NB,UCB$K_LENGTH,        UCB$K_LENGTH:
00000000  0074                  . = 0
          0000                  .IIF    NB,UCB$W_MB_SEED,       UCB$W_MB_SEED:
00000002  0000                                  .BLKW   1
00000074  0002                  . = 116
          0074                  .IIF    NB,UCB$L_MB_WAST,       UCB$L_MB_WAST:
00000078  0074                                  .BLKL   1
          0078                  .IIF    NB,UCB$L_MB_RAST,       UCB$L_MB_RAST:
0000007C  0078                                  .BLKL   1
```

```
                        007C            .IIF    NB,UCB$L_MB_MBX,        UCB$L_MB_MBX:
00000080  007C                                  .BLKL   1
                        0080            .IIF    NB,UCB$L_MB_SHB,        UCB$L_MB_SHB:
00000084  0080                                  .BLKL   1
                        0084            .IIF    NB,UCB$L_MB_WIOQFL,     UCB$L_MB_WIOQFL:
00000088  0084                                  .BLKL   1
                        0088            .IIF    NB,UCB$L_MB_WIOQBL,     UCB$L_MB_WIOQBL:
0000008C  0088                                  .BLKL   1
                        008C            .IIF    NB,UCB$L_MB_PORT,       UCB$L_MB_PORT:
00000090  008C                                  .BLKL   1
                        0090            .IIF    NB,UCB$C_MB_LENGTH,     UCB$C_MB_LENGTH:
                        0090            .IIF    NB,UCB$K_MB_LENGTH,     UCB$K_MB_LENGTH:
00000074  0090          . = 116
                        0074            .IIF    NB,UCB$B_SLAVE, UCB$B_SLAVE:
00000075  0074                                  .BLKB   1
                        0075            .IIF    NB,UCB$B_SPR,   UCB$B_SPR:
00000076  0075                                  .BLKB   1
                        0076            .IIF    NB,UCB$B_FEX,   UCB$B_FEX:
00000077  0076                                  .BLKB   1
                        0077            .IIF    NB,UCB$B_CEX,   UCB$B_CEX:
00000078  0077                                  .BLKB   1
                        0078            .IIF    NB,UCB$L_EMB,   UCB$L_EMB:
0000007C  0078                                  .BLKL   1
0000007E  007C                                  .BLKW   1
                        007E            .IIF    NB,UCB$W_FUNC,  UCB$W_FUNC:
00000080  007E                                  .BLKW   1
                        0080            .IIF    NB,UCB$L_DPC,   UCB$L_DPC:
00000084  0080                                  .BLKL   1
00000080  0084          . = 128
00000084  0080                                  .BLKL   1
                        0084            .IIF    NB,UCB$L_MAXBLOCK,      UCB$L_MAXBLOCK:
00000088  0084                                  .BLKL   1
                        0088            .IIF    NB,UCB$W_DIRSEQ,        UCB$W_DIRSEQ:
0000008A  0088                                  .BLKW   1
                        008A            .IIF    NB,UCB$W_OFFSET,        UCB$W_OFFSET:
0000008C  008A                                  .BLKW   1
                        008C            .IIF    NB,UCB$L_MEDIA, UCB$L_MEDIA:
                        008C            .IIF    NB,UCB$W_DA,    UCB$W_DA:
0000008E  008C                                  .BLKW   1
                        008E            .IIF    NB,UCB$W_DC,    UCB$W_DC:
00000090  008E                                  .BLKW   1
                        0090            .IIF    NB,UCB$W_EC1,   UCB$W_EC1:
00000092  0090                                  .BLKW   1
                        0092            .IIF    NB,UCB$W_EC2,   UCB$W_EC2:
00000094  0092                                  .BLKW   1
                        0094            .IIF    NB,UCB$B_OFFNDX,        UCB$B_OFFNDX:
00000095  0094                                  .BLKB   1
                        0095            .IIF    NB,UCB$B_OFFRTC,        UCB$B_OFFRTC:
00000096  0095                                  .BLKB   1
                        0096            .IIF    NB,UCB$W_BCR,   UCB$W_BCR:
00000098  0096                                  .BLKW   1
00000096  0098          . = 150
00000098  0096                                  .BLKW   1
                        0098            .IIF    NB,UCB$L_DX_BUF,        UCB$L_DX_BUF:
0000009C  0098                                  .BLKL   1
                        009C            .IIF    NB,UCB$L_DX_BFPNT,      UCB$L_DX_BFPNT:
000000A0  009C                                  .BLKL   1
```

```
                    00A0                .IIF    NB,UCB$L_DX_RXDB,       UCB$L_DX_RXDB:
000000A4            00A0                        .BLKL   1
                    00A4                .IIF    NB,UCB$W_DX_BCR,        UCB$W_DX_BCR:
000000A6            00A4                        .BLKW   1
                    00A6                .IIF    NB,UCB$B_DX_SCTCNT,     UCB$B_DX_SCTCNT:
000000A7            00A6                        .BLKB   1
000000A8            00A7                        .BLKB   1
00000074            00A8                . = 116
00000078            0074                        .BLKL   1
0000007C            0078                        .BLKL   1
00000080            007C                        .BLKL   1
00000084            0080                        .BLKL   1
00000088            0084                        .BLKL   1
0000008C            0088                        .BLKL   1
                    008C                .IIF    NB,UCB$L_TT_RDUE,       UCB$L_TT_RDUE:
00000090            008C                        .BLKL   1
00000094            0090                        .BLKL   1
00000098            0094                        .BLKL   1
0000009C            0098                        .BLKL   1
                    009C                .IIF    NB,UCB$B_TT_SPEED,      UCB$B_TT_SPEED:
0000009D            009C                        .BLKB   1
                    009D                .IIF    NB,UCB$B_TT_CRFILL,     UCB$B_TT_CRFILL:
0000009E            009D                        .BLKB   1
                    009E                .IIF    NB,UCB$B_TT_LFFILL,     UCB$B_TT_LFFILL:
0000009F            009E                        .BLKB   1
000000A0            009F                        .BLKB   1
                    00A0                .IIF    NB,UCB$B_TT_DESPEE,     UCB$B_TT_DESPEE:
000000A1            00A0                        .BLKB   1
                    00A1                .IIF    NB,UCB$B_TT_DECRF,      UCB$B_TT_DECRF:
000000A2            00A1                        .BLKB   1
                    00A2                .IIF    NB,UCB$B_TT_DELFF,      UCB$B_TT_DELFF:
000000A3            00A2                        .BLKB   1
000000A4            00A3                        .BLKB   1
                    00A4                .IIF    NB,UCB$B_TT_DETYPE,     UCB$B_TT_DETYPE:
000000A5            00A4                        .BLKB   1
                    00A5                .IIF    NB,UCB$W_TT_DESIZE,     UCB$W_TT_DESIZE:
000000A7            00A5                        .BLKW   1
000000A8            00A7                        .BLKB   1
                    00A8                .IIF    NB,UCB$L_TT_DECHAR,     UCB$L_TT_DECHAR:
000000AC            00A8                        .BLKL   1
000000B0            00AC                        .BLKL   1
000000B4            00B0                        .BLKL   1
000000B8            00B4                        .BLKL   1
                    00B8                .IIF    NB,UCB$L_TT_RTIMOU,     UCB$L_TT_RTIMOU:
000000BC            00B8                        .BLKL   1
                    00BC                .IIF    NB,UCB$C_TT_LENGTH,     UCB$C_TT_LENGTH:
                    00BC                .IIF    NB,UCB$K_TT_LENGTH,     UCB$K_TT_LENGTH:
00000074            00BC                . = 116
                    0074                .IIF    NB,UCB$L_NT_DATSSB,     UCB$L_NT_DATSSB:
00000078            0074                        .BLKL   1
                    0078                .IIF    NB,UCB$L_NT_INTSSB,     UCB$L_NT_INTSSB:
0000007C            0078                        .BLKL   1
                    007C                .IIF    NB,UCB$W_NT_CHAN,       UCB$W_NT_CHAN:
0000007E            007C                        .BLKW   1
00000080            007E                        .BLKW   1
        00000000                        .RESTORE
              0000    51                $VECDEF                         ;DEFINE VEC OFFSETS
```

```
                        0000              .SAVE   LOCAL_BLOCK
                        0000              .PSECT  $ABS$,ABS
                00000000  00BC              .=0
                        0000              .IIF    NB,VEC$Q_DISPATCH,      VEC$Q_DISPATCH:
                00000008  0000                      .BLKQ   1
                        0008              .IIF    NB,VEC$L_IDB,   VEC$L_IDB:
                0000000C  0008                      .BLKL   1
                        000C              .IIF    NB,VEC$L_INITIAL,       VEC$L_INITIAL:
                00000010  000C                      .BLKL   1
                        0010              .IIF    NB,VEC$W_MAPREG,        VEC$W_MAPREG:
                00000012  0010                      .BLKW   1
                        0012              .IIF    NB,VEC$B_NUMREG,        VEC$B_NUMREG:
                00000013  0012                      .BLKB   1
                        0013              .IIF    NB,VEC$B_DATAPATH,      VEC$B_DATAPATH:
                00000014  0013                      .BLKB   1
                        0014              .IIF    NB,VEC$L_ADP,   VEC$L_ADP:
                00000018  0014                      .BLKL   1
                        0018              .IIF    NB,VEC$L_UNITINIT,      VEC$L_UNITINIT:
                0000001C  0018                      .BLKL   1
                        001C              .IIF    NB,VEC$L_START, VEC$L_START:
                00000020  001C                      .BLKL   1
                        0020              .IIF    NB,VEC$L_UNITDISC,      VEC$L_UNITDISC:
                00000024  0020                      .BLKL   1
                        0024              .IIF    NB,VEC$C_LENGTH,        VEC$C_LENGTH:
                        0024              .IIF    NB,VEC$K_LENGTH,        VEC$K_LENGTH:
                00000000                  .RESTORE
                        0000    52
                        0000    53 ;
                        0000    54 ; LOCAL DATA
                        0000    55 ;
                        0000    56 ; STRUCTURE TYPE BYTE ARRAY
                        0000    57 ;
                        0000    58
          10 09 06 05   0000    59 STYPE:  .BYTE   DYN$C_CRB,DYN$C_DDB,DYN$C_IDB,DYN$C_UCB ;
```

ZRC R-R R R R RRS P-ICPSPSPSPCA T8T27 M-DDSST 6

```
                        0004     61          .SBTTL  INITIALIZE DRIVER DATA BASE
                        0004     62 ;+
                        0004     63 ; IOC$INITDRV - INITIALIZE DRIVER DATA BASE
                        0004     64 ;
                        0004     65 ; THIS ROUTINE IS CALLED AFTER HAVING LOADED ALL OR PART OF A DEVICE DATA BASE.
                        0004     66 ; IT MUST BE CALLED FOR EACH UNIT THAT IS DEFINED. ITS FUNCTION IS TO USE THE
                        0004     67 ; INITIALIZATION TABLE IN THE DRIVER PROLOGUE TO INITIALIZE STATIC INFORMATION
                        0004     68 ; IN THE DRIVER DATA BASE.
                        0004     69 ;
                        0004     70 ; INPUTS:
                        0004     71 ;
                        0004     72 ;       R4 = ADDRESS OF DRIVER PROLOGUE TABLE.
                        0004     73 ;       R5 = ADDRESS OF DEVICE UCB.
                        0004     74 ;
                        0004     75 ; OUTPUTS:
                        0004     76 ;
                        0004     77 ;       THE DRIVER PROLOGUE TABLE IS INTERPRETED AND THE RESPECTIVE DATA BASE
                        0004     78 ;       IS INITIALIZED.
                        0004     79 ;
                        0004     80 ;       R0 LOW BIT CLEAR INDICATES A BAD DATA STRUCTURE.
                        0004     81 ;
                        0004     82 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                        0004     83 ;-
                        0004     84
                        0004     85          .ENABL  LSB
                        0004     86 IOC$INITDRV::                                ;INITIALIZE DRIVER DATA BASE
    53    10 A4   3C    0004     87          MOVZWL  DPT$W_INITTAB(R4),R3        ;GET OFFSET TO INITIALIZATION TABLE
          04   11   0008     88          BRB     10$                             ;BR TO COMMON CODE
```

```
                        000A     90          .SBTTL  REINITIALIZE DRIVER DATA BASE
                        000A     91 ;+
                        000A     92 ; IOC$REINITDRV - REINITIALIZE DRIVER DATA BASE
                        000A     93 ;
                        000A     94 ; THIS ROUTINE IS CALLED AFTER HAVING LOADED A DRIVER WHOSE DEVICE DATA BASE IS
                        000A     95 ; ALREADY RESIDENT IN MEMORY. IT MUST BE CALLED FOR EACH UNIT THAT IS DEFINED.
                        000A     96 ; ITS FUNCTION IS TO INITIALIZE ONLY THOSE FIELDS IN THE DATA BASE THAT ARE NOT
                        000A     97 ; DYNAMIC.
                        000A     98 ;
                        000A     99 ; INPUTS:
                        000A    100 ;
                        000A    101 ;       R4 = ADDRESS OF DRIVER PROLOGUE TABLE.
                        000A    102 ;       R5 = ADDRESS OF DEVICE UCB.
                        000A    103 ;
                        000A    104 ; OUTPUTS:
                        000A    105 ;
                        000A    106 ;       THE DRIVER PROLOGUE TABLE IS INTERPRETED AND THE RESPECTIVE DATA BASE
                        000A    107 ;       IS REINITIALIZED.
                        000A    108 ;
                        000A    109 ;       R0 LOW BIT CLEAR INDICATES A BAD DATA STRUCTURE.
                        000A    110 ;
                        000A    111 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                        000A    112 ;-
                        000A    113
                        000A    114 IOC$REINITDRV::                             ;REINITIALIZE DRIVER DATA BASE
          53   12 A4   3C  000A    115         MOVZWL  DPT$W_REINITTAB(R4),R3  ;GET OFFSET TO REINITIALIZATION TABLE
             53   54   C0  000E    116 10$:    ADDL    R4,R3                   ;CALCULATE ADDRESS OF INITIALIZE TABLE
                  5F   10  0011    117 20$:    BSBB    LOCATE                  ;LOCATE STRUCTURE TYPE
             51   83   9A  0013    118         MOVZBL  (R3)+,R1                ;GET OFFSET IN STRUCTURE
             52   51   C0  0016    119         ADDL    R1,R2                   ;CALCULATE BASE ADDRESS OF FIELD
             51   83   9A  0019    120         MOVZBL  (R3)+,R1                ;GET OPERATION CODE
                  02   10  001C    121         BSBB    EXECUTE                 ;EXECUTE OPERATION CODE
                  F1   11  001E    122         BRB     20$                     ;
                       0020    123         .DSABL  LSB
```

```
                     0020   125          .SBTTL  LOCAL SUBROUTINE TO EXECUTE OPERATION CODE
                     0020   126 ;
                     0020   127 ; EXECUTE - EXECUTE OPERATION CODE
                     0020   128 ;
                     0020   129 ; THIS ROUTINE IS CALLED TO EXECUTE A SINGLE RELOCATION OPERATION CODE.
                     0020   130 ;
                     0020   131 ; INPUTS:
                     0020   132 ;
                     0020   133 ;       R1 = OPERATION CODE.
                     0020   134 ;       R2 = STRUCTURE FIELD BASE ADDRESS.
                     0020   135 ;       R3 = ADDRESS OF NEXT RELOCATION BYTE.
                     0020   136 ;       R4 = BASE ADDRESS OF DRIVER PROLOGUE TABLE.
                     0020   137 ;       R5 = ADDRESS OF DEVICE UCB.
                     0020   138 ;
                     0020   139 ; OUTPUTS:
                     0020   140 ;
                     0020   141 ;       IF AN INVALID OPERATION CODE IS SPECIFIED, A FAILURE IS RETURNED TO THE
                     0020   142 ;       CALLERS CALLER. ELSE THE OPERAND VALUE IS STORE IN THE SPECIFIED FIELD
                     0020   143 ;       IN THE SPECIFIED STRUCTURE.
                     0020   144 ;
                     0020   145
                     0020   146 EXECUTE:                                      ;EXECUTE RELOCATION OPERATION
                     0020   147          CASE    R1,<-                       ;DISPATCH TO FETCH ROUTINE
                     0020   148                  10$,-                       ;BYTE IMMEDIATE
                     0020   149                  30$,-                       ;WORD IMMEDIATE
                     0020   150                  50$,-                       ;WORD DISPLACED
                     0020   151                  70$,-                       ;LONGWORD IMMEDIATE
                     0020   152                  90$,-                       ;FIELD IMMEDIATE
                     0020   153                  >
04'  00  51  AF      0020                   CASEW   R1,#0,S^#<<30001$-30000$>/2>-1
                     0024           30000$:
                0022' 0024                  .SIGNED_WORD    10$-30000$
                0029' 0026                  .SIGNED_WORD    30$-30000$
                0030' 0028                  .SIGNED_WORD    50$-30000$
                0038' 002A                  .SIGNED_WORD    70$-30000$
                003F' 002C                  .SIGNED_WORD    90$-30000$
                      002E           30001$:
                      002E   154
                      002E   155 ;
                      002E   156 ; INDIRECT OPERAND
                      002E   157 ;
                      002E   158
     50  93  D0     002E   159          MOVL    @(R3)+,R0                   ;GET DATA LONGWORD
                      0031   160          CASE    R1,<-                       ;DISPATCH TO STORE ROUTINE
                      0031   161                  20$,-                       ;STORE BYTE
                      0031   162                  40$,-                       ;STORE WORD
                      0031   163                  60$,-                       ;STORE WORD DISPLACED
                      0031   164                  80$,-                       ;STORE LONGWORD
                      0031   165                  100$,-                      ;STORE FILED
                      0031   166                  >,LIMIT=#^X80               ;
04'  0080 8F  51  AF  0031                   CASEW   R1,#^X80,S^#<<30003$-30002$>/2>-1
                      0037           30002$:
                0012' 0037                  .SIGNED_WORD    20$-30002$
                0019' 0039                  .SIGNED_WORD    40$-30002$
                0020' 003B                  .SIGNED_WORD    60$-30002$
                0028' 003D                  .SIGNED_WORD    80$-30002$
                002F' 003F                  .SIGNED_WORD    100$-30002$
```

```
                                0041        30003$:
                    8E    D5    0041   167          TSTL    (SP)+                  ;REMOVE RETURN FROM STACK
                    50    D4    0043   168          CLRL    R0                     ;SET FAILURE INDICATION
                          05    0045   169          RSB                            ;
                                0046   170
                                0046   171 ;
                                0046   172 ; BYTE IMMEDIATE
                                0046   173 ;
                                0046   174
              50    83    9A    0046   175 10$:     MOVZBL  (R3)+,R0               ;GET DATA BYTE
              62    50    90    0049   176 20$:     MOVB    R0,(R2)                ;STORE BYTE
                          05    004C   177          RSB                            ;
                                004D   178
                                004D   179 ;
                                004D   180 ; WORD IMMEDIATE
                                004D   181 ;
                                004D   182
              50    83    3C    004D   183 30$:     MOVZWL  (R3)+,R0               ;GET DATA WORD
              62    50    B0    0050   184 40$:     MOVW    R0,(R2)                ;STORE WORD
                          05    0053   185          RSB                            ;
                                0054   186
                                0054   187 ;
                                0054   188 ; WORD DISPLACED
                                0054   189 ;
                                0054   190
              50    83    32    0054   191 50$:     CVTWL   (R3)+,R0               ;GET WORD DISPLACEMENT
              50    54    C0    0057   192 60$:     ADDL    R4,R0                  ;CALCULATE ACTUAL ADDRESS
                    03    11    005A   193          BRB     80$                    ;
                                005C   194
                                005C   195 ;
                                005C   196 ; LONGWORD IMMEDIATE
                                005C   197 ;
                                005C   198
              50    83    D0    005C   199 70$:     MOVL    (R3)+,R0               ;GET DATA LONGWORD
              62    50    D0    005F   200 80$:     MOVL    R0,(R2)                ;STORE DATA LONGWORD
                          05    0062   201          RSB                            ;
                                0063   202
                                0063   203 ;
                                0063   204 ; FIELD STRUCTURE
                                0063   205 ;
                                0063   206
              50    83    D0    0063   207 90$:     MOVL    (R3)+,R0               ;GET DATA LONGWORD
              7E    83    9A    0066   208 100$:    MOVZBL  (R3)+,-(SP)            ;GET STARTING BIT NUMBER
              51    83    9A    0069   209          MOVZBL  (R3)+,R1               ;GET SIZE OF FIELD
    62    51    8E    50    F0    006C   210          INSV    R0,(SP)+,R1,(R2)       ;STORE FIELD
                          05    0071   211          RSB                            ;
```

```
                                0072          .SBTTL  LOCAL SUBROUTINE TO LOCATE DATA STRUCTURE TYPE
                                0072  ;
                                0072  ; LOCATE - LOCATE DATA STRUCTURE TYPE
                                0072  ;
                                0072  ; INPUTS:
                                0072  ;
                                0072  ;       R5 = ADDRESS OF DEVICE UCB.
                                0072  ;
                                0072  ; OUTPUTS:
                                0072  ;
                                0072  ;       IF THE STRUCTURE TYPE CODE IS ZERO, THEN SUCCESS IS RETURNED TO THE CALLERS
                                0072  ;       CALLER. IF THE STRUCTURE TYPE CODE IS INVALID, THEN AN ERROR IS RETURNED
                                0072  ;       TO THE CALLERS CALLER. ELSE THE APPROPRIATE STRUCTURE BASE ADDRESS IS
                                0072  ;       RETURNED IN R2.
                                0072  ;
                                0072
                                0072  LOCATE:                                   ;LOCATE DATA STRUCTURE TYPE
          50    01    88        0072          BISB    #1,R0                     ;ASSUME END OF RELOCATION TABLE
                63    95        0075          TSTB    (R3)                      ;END OF TABLE?
                11    13        0077          BEQL    10$                       ;IF EQL YES
   82 AF  04    83    3A        0079          LOCC    (R3)+,#4,STYPE            ;LOCATE STRUCTURE CODE
                                007E          CASE    R0,<-                     ;DISPATCH TO STRUCTURE ROUTINE
                                007E                  20$,-                     ;UCB
                                007E                  30$,-                     ;IDB
                                007E                  40$,-                     ;DDB
                                007E                  50$,-                     ;CRB
                                007E                  >,LIMIT=#1                ;
      03'  01    50    AF        007E          CASEW   R0,#1,S^#<<30005$-30004$>/2>-1
                                0082  30004$:
                 000B'          0082          .SIGNED_WORD    20$-30004$
                 000F'          0084          .SIGNED_WORD    30$-30004$
                 0018'          0086          .SIGNED_WORD    40$-30004$
                 001D'          0088          .SIGNED_WORD    50$-30004$
                                008A  30005$:
                8E    D5        008A  10$:    TSTL    (SP)+                     ;REMOVE RETURN FROM STACK
                      05        008C          RSB                               ;
                                008D
                                008D  ;
                                008D  ; UCB STRUCTURE
                                008D  ;
                                008D
          52    55    D0        008D  20$:    MOVL    R5,R2                     ;SET UCB ADDRESS
                      05        0090          RSB                               ;
                                0091
                                0091  ;
                                0091  ; IDB STRUCTURE
                                0091  ;
                                0091
      52    20 A5    D0        0091  30$:    MOVL    UCB$L_CRB(R5),R2          ;GET ADDRESS OF CRB
      52    1C A2    D0        0095          MOVL    CRB$L_INTD+VEC$L_IDB(R2),R2 ;GET ADDRESS OF IDB
                      05        0099          RSB                               ;
                                009A
                                009A  ;
                                009A  ; DDB STRUCTURE
                                009A  ;
                                009A
      52    24 A5    D0        009A  40$:    MOVL    UCB$L_DDB(R5),R2          ;GET ADDRESS OF DDB
```

```
                     05  009E   263          RSB                                 ;
                         009F   264
                         009F   265 ;
                         009F   266 ; CRB STRUCTURE
                         009F   267 ;
                         009F   268
           52  20 A5    D0  009F   269 50$:     MOVL    UCB$L_CRB(R5),R2        ;GET ADDRESS OF CRB
                     05  00A3   270          RSB                                 ;
                         00A4   271
                         00A4   272          .END
```

CRB$B_MASK	0000000E	D		UCB$B_AMOD	00000053	D		UCB$L_IOQFL	00000040	D					
CRB$B_TYPE	0000000A	D		UCB$B_CEX	00000077	D		UCB$L_IRP	0000004C	D					
CRB$C_LENGTH	00000038	D		UCB$B_CM1	0000004A	D		UCB$L_LINK	0000002C	D					
CRB$K_LENGTH	00000038	D		UCB$B_CM2	0000004B	D		UCB$L_LOGADR	00000064	D					
CRB$L_INTD	00000014	D		UCB$B_DEVCLASS	00000038	D		UCB$L_MAXBLOCK	00000084	D					
CRB$L_INTD2	00000038	D		UCB$B_DEVTYPE	00000039	D		UCB$L_MB_MBX	0000007C	D					
CRB$L_LINK	00000010	D		UCB$B_DIPL	00000052	D		UCB$L_MB_PORT	0000008C	D					
CRB$L_WQBL	00000004	D		UCB$B_DX_SCTCNT	000000A6	D		UCB$L_MB_RAST	00000078	D					
CRB$L_WQFL	00000000	D		UCB$B_ERTCNT	00000070	D		UCB$L_MB_SHB	00000080	D					
CRB$W_REFC	0000000C	D		UCB$B_ERTMAX	00000071	D		UCB$L_MB_WAST	00000074	D					
CRB$W_SIZE	00000008	D		UCB$B_FEX	00000076	D		UCB$L_MB_WIOQBL	00000088	D					
DDB$B_ACPCLASS	00000013	D		UCB$B_FIPL	0000000B	D		UCB$L_MB_WIOQFL	00000084	D					
DDB$B_TYPE	0000000A	D		UCB$B_LOCSRV	0000003C	D		UCB$L_MEDIA	0000008C	D					
DDB$C_LENGTH	00000034	D		UCB$B_OFFNDX	00000094	D		UCB$L_NT_DATSSB	00000074	D					
DDB$K_LENGTH	00000034	D		UCB$B_OFFRTC	00000095	D		UCB$L_NT_INTSSB	00000078	D					
DDB$L_ACPD	00000010	D		UCB$B_REMSRV	0000003D	D		UCB$L_OPCNT	00000060	D					
DDB$L_DDT	0000000C	D		UCB$B_SECTORS	0000003C	D		UCB$L_OWNUIC	0000001C	D					
DDB$L_LINK	00000000	D		UCB$B_SLAVE	00000074	D		UCB$L_PID	00000028	D					
DDB$L_UCB	00000004	D		UCB$B_SPR	00000075	D		UCB$L_RQBL	00000004	D					
DDB$T_DRVNAME	00000024	D		UCB$B_STATE	00000052	D		UCB$L_RQFL	00000000	D					
DDB$T_NAME	00000014	D		UCB$B_TRACKS	0000003D	D		UCB$L_SVAPTE	00000068	D					
DDB$W_SIZE	00000008	D		UCB$B_TT_CRFILL	0000009D	D		UCB$L_SVPN	00000064	D					
DPT$B_ADPTYPE	0000000C	D		UCB$B_TT_DECRF	000000A1	D		UCB$L_TT_DECHAR	000000A8	D					
DPT$B_FLAGS	0000000D	D		UCB$B_TT_DELFF	000000A2	D		UCB$L_TT_RDUE	0000008C	D					
DPT$B_REFC	0000000B	D		UCB$B_TT_DESPEE	000000A0	D		UCB$L_TT_RTIMOU	000000B8	D					
DPT$B_TYPE	0000000A	D		UCB$B_TT_DETYPE	000000A4	D		UCB$L_VCB	00000030	D					
DPT$C_LENGTH	0000002A	D		UCB$B_TT_LFFILL	0000009E	D		UCB$T_PARTNER	0000000C	D					
DPT$K_LENGTH	0000002A	D		UCB$B_TT_SPEED	0000009C	D		UCB$W_BCNT	0000006E	D					
DPT$L_BLINK	00000004	D		UCB$B_TYPE	0000000A	D		UCB$W_BCR	00000096	D					
DPT$L_FLINK	00000000	D		UCB$B_VERTSZ	0000003F	D		UCB$W_BOFF	0000006C	D					
DPT$T_NAME	0000001E	D		UCB$C_LENGTH	00000074	D		UCB$W_BUFQUO	00000018	D					
DPT$W_INITTAB	00000010	D		UCB$C_MB_LENGTH	00000090	D		UCB$W_BYTESTOGO	0000003E	D					
DPT$W_MAXUNITS	00000016	D		UCB$C_TT_LENGTH	000000BC	D		UCB$W_CHARGE	0000004A	D					
DPT$W_REINITTAB	00000012	D		UCB$K_LENGTH	00000074	D		UCB$W_CYLINDERS	0000003E	D					
DPT$W_SIZE	00000008	D		UCB$K_MB_LENGTH	00000090	D		UCB$W_DA	0000008C	D					
DPT$W_UCBSIZE	0000000E	D		UCB$K_TT_LENGTH	000000BC	D		UCB$W_DC	0000008E	D					
DPT$W_UNLOAD	00000014	D		UCB$L_AMB	00000054	D		UCB$W_DEVBUFSIZ	0000003A	D					
DPT$W_VERSION	00000018	D		UCB$L_ASTQBL	00000010	D		UCB$W_DEVSTS	0000005A	D					
DYN$C_CRB =	00000005	D		UCB$L_ASTQFL	0000000C	D		UCB$W_DIRSEQ	00000088	D					
DYN$C_DDB =	00000006	D		UCB$L_CPID	0000005C	D		UCB$W_DSTADDR	00000018	D					
DYN$C_IDB =	00000009	D		UCB$L_CRB	00000020	D		UCB$W_DX_BCR	000000A4	D					
DYN$C_UCB =	00000010	D		UCB$L_DDB	00000024	D		UCB$W_EC1	00000090	D					
EXECUTE	00000020	R	D	01 UCB$L_DEVCHAR	00000034	D		UCB$W_EC2	00000092	D					
IDB$B_TYPE	0000000A		D	UCB$L_DEVDEPEND	0000003C	D		UCB$W_ERRCNT	00000072	D					
IDB$C_LENGTH	00000034		D	UCB$L_DPC	00000080	D		UCB$W_FUNC	0000007E	D					
IDB$K_LENGTH	00000034		D	UCB$L_DUETIM	0000005C	D		UCB$W_MB_SEED	00000000	D					
IDB$L_ADP	00000010		D	UCB$L_DX_BFPNT	0000009C	D		UCB$W_MSGCNT	00000016	D					
IDB$L_CSR	00000000		D	UCB$L_DX_BUF	00000098	D		UCB$W_MSGMAX	00000014	D					
IDB$L_OWNER	00000004		D	UCB$L_DX_RXDB	000000A0	D		UCB$W_NT_CHAN	0000007C	D					
IDB$L_UCBLST	00000014		D	UCB$L_EMB	00000078	D		UCB$W_OFFSET	0000008A	D					
IDB$W_SIZE	00000008		D	UCB$L_FIRST	00000014	D		UCB$W_REFC	00000050	D					
IDB$W_UNITS	0000000C		D	UCB$L_FPC	0000000C	D		UCB$W_SIZE	00000008	D					
IOC$INITDRV	00000004	RG	D	01 UCB$L_FQBL	00000004	D		UCB$W_SRCADDR	0000001A	D					
IOC$REINITDRV	0000000A	RG	D	01 UCB$L_FQFL	00000000	D		UCB$W_STS	00000058	D					
LOCATE	00000072	R	D	01 UCB$L_FR3	00000010	D		UCB$W_TT_DESIZE	000000A5	D					
SIZ... =	00000001		D	UCB$L_FR4	00000014	D		UCB$W_UNIT	00000048	D					
STYPE	00000000	R	D	01 UCB$L_IOQBL	00000044	D		UCB$W_VPROT	0000001A	D					

```
VEC$B_DATAPATH    00000013    D
VEC$B_NUMREG      00000012    D
VEC$C_LENGTH      00000024    D
VEC$K_LENGTH      00000024    D
VEC$L_ADP         00000014    D
VEC$L_IDB         00000008    D
VEC$L_INITIAL     0000000C    D
VEC$L_START       0000001C    D
VEC$L_UNITDISC    00000020    D
VEC$L_UNITINIT    00000018    D
VEC$Q_DISPATCH    00000000    D
VEC$W_MAPREG      00000010    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
$EP	000000A4 (164.)	01 (1.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG
ABS	000000BC (188.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
CRB$L_INTD	00000014		#-255 (1)
DPT$W_INITTAB	00000010		#-87 (1)
DPT$W_REINITTAB	00000012		#-115 (1)
DYN$C_CRB	=00000005		59 (1)
DYN$C_DDB	=00000006		59 (1)
DYN$C_IDB	=00000009		59 (1)
DYN$C_UCB	=00000010		59 (1)
EXECUTE	00000020-R	146 (1)	#-121 (1)
IOC$INITDRV	00000004-R	86 (1)	
IOC$REINITDRV	0000000A-R	114 (1)	
LOCATE	00000072-R	229 (1)	#-117 (1)
STYPE	00000000-R	59 (1)	233 (1)
UCB$L_CRB	00000020		#-254 (1) #-269 (1)
UCB$L_DDB	00000024		#-262 (1)
VEC$L_IDB	00000008		#-255 (1)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CRBDEF	1	45 (1)	45 (1)
$DDBDEF	1	46 (1)	46 (1)
$DEFINI	1	45 (1)	45 (1) 46 (1) 47 (1) 48 (1) 49 (1)
			50 (1) 51 (1)
$DPTDEF	2	47 (1)	47 (1)
$DYNDEF	2	48 (1)	48 (1)
$IDBDEF	1	49 (1)	49 (1)
$UCBDEF	10	50 (1)	50 (1)
$VECDEF	2	51 (1)	51 (1)
CASE	1	147 (1)	147 (1) 160 (1) 234 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	116	(1)	119	(1)	192	(1)								
BEQL	0013	232	(1)												
BISB	0088	230	(1)												
BRB	0011	122	(1)	193	(1)	88	(1)								
BSBB	0010	117	(1)	121	(1)										
CASEW	00AF	153	(1)	166	(1)	239	(1)								
CLRL	00D4	168	(1)												
CVTWL	0032	191	(1)												
INSV	00F0	210	(1)												
LOCC	003A	233	(1)												
MOVB	0090	176	(1)												
MOVL	00D0	159	(1)	199	(1)	200	(1)	207	(1)	247	(1)	254	(1)	255	(1)
		262	(1)	269	(1)										
MOVW	00B0	184	(1)												
MOVZBL	009A	118	(1)	120	(1)	175	(1)	208	(1)	209	(1)				
MOVZWL	003C	115	(1)	183	(1)	87	(1)								
RSB	0005	169	(1)	177	(1)	185	(1)	201	(1)	211	(1)	241	(1)	248	(1)
		256	(1)	263	(1)	270	(1)								
TSTB	0095	231	(1)												
TSTL	00D5	167	(1)	240	(1)										

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.BYTE	59	(1)														
.DSABL	123	(1)														
.ENABL	85	(1)														
.END	272	(1)														
.ENDM	147	(1)	45	(1)	46	(1)	47	(1)	48	(1)	49	(1)	50	(1)	51	(1)
.ENDR	153	(1)	166	(1)	239	(1)										
.IDENT	5	(1)														
.IRP	153	(1)	166	(1)	239	(1)										
.NOCROSS	45	(1)	46	(1)	47	(1)	48	(1)	49	(1)	50	(1)	51	(1)		
.PAGE	124	(1)	212	(1)	60	(1)	89	(1)								
.PSECT	6	(1)														
.RESTORE	45	(1)	46	(1)	47	(1)	48	(1)	49	(1)	50	(1)	51	(1)		
.SAVE	45	(1)	46	(1)	47	(1)	48	(1)	49	(1)	50	(1)	51	(1)		
.SBTTL	125	(1)	213	(1)	61	(1)	90	(1)								
.SIGNED_WORD	153	(1)	166	(1)	239	(1)										
.TITLE	4	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
R0	14	#-159	(1)	#-168	(1)	#-175	(1)	#-176	(1)	#-183	(1)	#-184	(1)
		#-191	(1)	#-192	(1)	#-199	(1)	#-200	(1)	#-207	(1)	#-210	(1)
		#-230	(1)	#-239	(1)								
R1	7	#-118	(1)	#-119	(1)	#-120	(1)	#-153	(1)	#-166	(1)	#-209	(1)
		#-210	(1)										
R2	11	#-119	(1)	#-176	(1)	#-184	(1)	#-200	(1)	210	(1)	#-247	(1)
		#-254	(1)	#-255	(1)	#-262	(1)	#-269	(1)				
R3	15	#-115	(1)	#-116	(1)	#-118	(1)	#-120	(1)	#-159	(1)	#-175	(1)
		#-183	(1)	#-191	(1)	#-199	(1)	#-207	(1)	#-208	(1)	#-209	(1)
		#-231	(1)	#-233	(1)	#-87	(1)						
R4	4	#-115	(1)	#-116	(1)	#-192	(1)	#-87	(1)				
R5	4	#-247	(1)	#-254	(1)	#-262	(1)	#-269	(1)				
SP	4	#-167	(1)	#-208	(1)	#-210	(1)	#-240	(1)				

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.01	00:00:00.19
Command processing	893	00:00:00.22	00:00:00.70
Pass 1	565	00:00:01.48	00:00:02.40
Symbol table sort	0	00:00:00.08	00:00:00.08
Pass 2	28	00:00:00.38	00:00:00.83
Symbol table output	2	00:00:00.03	00:00:00.08
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	4	00:00:00.05	00:00:00.08
Assembler run totals	1518	00:00:02.25	00:00:04.36

The working set limit was 2400 pages.
82358 bytes (161 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 317 non-local and 23 local symbols.
272 source lines were read in Pass 1, producing 30 object records in Pass 2.
70 pages of virtual memory were used to define 17 macros.

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	8
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	13

618 GETS were required to define 13 macros.

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 18 Frame B5          Sequence 3555
RELOCDRV                    *** RELOCDRV relocate I/O driver     11-NOV-1987 17:50:53  VAX/VMS Macro V04-00        Page  22
VAX-11 Macro Run Statistics                                      3-JAN-1985 16:55:04  RELOCDRV.MAR;1                    (1)

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:RELOCDRV.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$
```

```
0     ! DEC/CMS REPLACEMENT HISTORY, Element RMS.B32
0     ! *20   11-NOV-1987 08:53:35 GEARIN "ADDED SUPPORT FOR EXTENDED P-TABLES"
0     ! *19   22-MAY-1987 14:12:32 BAGGETT "FINISHED"
0     ! *18   14-APR-1987 14:54:38 MARTIN "FIXED RMS"
0     ! *17   14-APR-1987 14:34:06 BAGGETT "FINISHED"
0     ! *16   30-MAR-1987 15:58:10 MARTIN "UPDATES FOR RRR; TESTED SOMEWHAT ONLINE, STEPPING VIA DEBUGGER"
0     ! *15   26-MAR-1987 11:48:25 BARANSKI "RESTORING POST RELEASE CHANGES"
0     ! *14   26-MAR-1987 11:26:23 BARANSKI "FREEZING RELEASE CHANGE"
0     ! *13   25-MAR-1987 09:49:41 MEIER "EXTEND EXIT 47 TO ONE MORE OCCURANCE"
0     ! *12   16-MAR-1987 10:27:17 BERGAZZI "RECOMPILE"
0     ! *11   16-MAR-1987 10:20:29 BERGAZZI "ADD RA70 SUPPORT"
0     ! *10    4-MAR-1987 15:28:07 BAGGETT "TK50 ADDED AS BOOT/LOAD DEVICE"
0     ! *9    19-JAN-1987 13:48:32 MEIER "ADDED LLL CASE TO THE ADAPTER-FINDING ALGORITHM IN DSX$CHECK_SUPPORT"
0     ! *8     4-DEC-1986 16:28:35 SILVER "VOLUME SHADOWING"
0     ! *7    29-OCT-1986 15:56:14 BARANSKI "Change Addressing Mode."
0     ! *6    22-JUL-1986 09:33:32 MUSE "RMS TEST."
0     ! *5    27-MAY-1986 15:14:15 SALEM "ALL DONE"
0     ! *4     2-APR-1986 14:29:29 SALEM "<no reason given>"
0     ! *3    19-FEB-1986 20:23:41 BARANSKI "Fix CMS error."
0     ! *2    14-FEB-1986 10:25:40 BAGGETT "<no reason given>"
0     ! *1     6-FEB-1986 15:22:17 DS ""
0     ! DEC/CMS REPLACEMENT HISTORY, Element RMS.B32
0     %title '*** RMS Master RMS routines'
0     module rms (                                    ! Emulate RMS for VAX Supervisor
0                     ident = '10.10-20'                                                    !G:20
0                     ) =
0     !
0     ! Copyright (c) 1980, 1982, 1983, 1984, 1985, 1986, 1987                                !G:9
0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0     !
0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0     ! REMAIN IN DEC.
0     !
0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0     ! CORPORATION.
0     !
0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0
0     !++
0     ! FACILITY: Diagnostic Supervisor
0     !
0     ! ABSTRACT:
0     !
0     !       This module emulates RMS (read-only functions) for the
0     !       Supervisor.  It additionally supports RT-11 format files
0     !       on console floppy.
0     !
0     ! AUTHOR:
0     !       Dave Butenhof,          4-aug-1980
0     !
```

```
0     ! MODIFIED BY:
0     !
0     !                Dave Butenhof, 02-oct-1980, Version 6.1
0     !       1        Eliminate possiblility of accessing console storage device
0     !                as "CS:" by explicitly checking for controller and unit.
0     !       2        Make "last-chance" backup filespec global as "DEF$Q_BACKSTOP"
0     !       3        <DEBUG> to make user mode debug easier, conditionalize MFPR
0     !                to execute only in standalone.
0     !       4        Add TS11 rpb setup to RMS$OPEN.
0     !                Dave Butenhof, 9-apr-1981, Version 6.3
0     !       05       Fix up some context problems in RMS base routines.  Namely,
0     !                Return EOF if VBN = eofvbn and ffb = 0 ($READ).  Also,
0     !                start RSZ and RBF as 0!!!
0     !                - Dave butenhof, 02-Jun-1981, version 6.4
0     !       06       Alter directory spec on libraries, use logical names $DS,
0     !                $DIAG for flexibility.
0     !                - Dave Butenhof, 24-Jun-1981, version 6.4                                  [07]
0     !       07       Setup RPB for console device, to allow using the console                   [07]
0     !                BOOTDRIVR routines.  Also, start cleaning up some of the                   [07]
0     !                Macro JSB routine linkages to use Bliss V2.1's register output             [07]
0     !                parameter capability, instead of global registers.
0     !       08       - Dave Butenhof, 31-aug-1981, version 6.5
0     !                Support DQ device (IDC) for Nebula, as ODS disk
0     !
0     !       09       - Dave Butenhof, 26-Oct-1981, version 6.-
0     !                Alter call to LOC$PTABLE to use now argument, link address
0     !
0     !       10       - Dave Butenhof, 25-Jan-1982, version 6.6
0     !                Fix calls to Exe$AloNonPaged to use output arguments instead
0     !                of global registers.  Also, add new routines to allocate and
0     !                init pseudo RPB, and to deallocate same.  Generally clean up
0     !                RMS$OPEN somewhat.
0     !
0     !       11       - Dave Butenhof, 05-Mar-1982, version 6.6
0     !                Fix edit 10...can't have size field of data structure be
0     !                direct output parameter of AloNonPaged, since pointer hasn't
0     !                been set yet!
0     !
0     !       [12]     Dave Butenhof, 15-Apr-1982, version 6.7                                    !G:20
0     !                Add support for UDA-50 disks (DUan)                                        !G:20
0     !       [13]     Dave Butenhof, 24-May-1982, version 6.8                                    !G:20
0     !                Generalize checking for device support with DS$GA_SUPPORT_TABLE            !G:20
0     !                data structure and DSX$CHECK_SUPPORT routine.  This is only for            !G:20
0     !                VDS file services, and does not/should not reflect all devices.            !G:20
0     !
0     !       14       Bob Bergazzi    7-Aug-1982      version 6.9
0     !                Edit 13 had a typo - RL211 in device support table would not
0     !                allow file services for an RL02 connected to an RL11.
0     !
0     !       15       M. Baggett      1-Dec-1982      version 6.10
0     !                Added support for TU80 magtape as load device.
0     !
0     !       16       M. Baggett      7-Dec-1982      version 6.10
0     !                Added support for TU81 magtape as load device.
0     !
0     !       17       M. Baggett      23-Feb-1983     Version 6.11
0     !                Added support for RC25 booting and file operations.
```

```
;   0115  0     !
;   0116  0     !       18      M. Baggett      1-March-1983    Version 6.11
;   0117  0     !               Added LESI,RCF25 and support for booting CI-HSC50.
;   0118  0     !
;   0119  0     !       19      M. Baggett      17-May-1983     Version 6.11
;   0120  0     !               Added changes to support CI-HSC50 port station #.
;   0121  0     !       20      M. Baggett      22-Sep-1983     Version 6.13
;   0122  0     !               Changed name LESI to RC11
;   0123  0     !
;   0124  0     !       21      M. Baggett      11-Oct-1983     Version 6.13
;   0125  0     !               Added support for CI-HSC50 tape and TS05
;   0126  0     !
;   0127  0     !       22      Domenic Andella 1-Mar-1984      Version 6.13
;   0128  0     !               Added code in routines DSX$CHECK_SUPPORT and
;   0129  0     !               RMS$OPEN for 8600 CPU. Each routine uses code
;   0130  0     !               to 'backtrack' through ptables till _LINK field
;   0131  0     !               equals zero, when this occurs the present Ptable
;   0132  0     !               is linked to HUB. Not so for a VENUS, when ptable
;   0133  0     !               _LINK field equals zero it means were linked to
;   0134  0     !               to an SBIA since the HUB is ABUS.  Therfore the
;   0135  0     !               modifications have to do with saving and reloading
;   0136  0     !               the previous adapter and controller address if running on a VENUS.
;   0137  0     !
;   0138  0     !       23      Bob Bergazzi    April 13, 1984  Version 7.0
;   0139  0     !               Changed DSX$Check_support to support ODS format on
;   0140  0     !               the console storage device for the ZZZ cpu.
;   0141  0     !
;   0142  0     !       24      M. Baggett      28-aug-1984     Version 7.1
;   0143  0     !               Changed controller for RC25/RCF25 from RC11 to LESI
;   0144  0     !               so devices could be accessed.
;   0145  0     !
;   0146  0     !       25      M. Baggett      5-Oct-1984      Version 7.1
;   0147  0     !               Change TAPE in support_table to allow operaton on HSC50 tape.
;   0148  0     !
;   0149  0     !       26      Bob Bergazzi    Oct. 18,1984    Version 7.1
;   0150  0     !               Added recognition of DWBUA in the DSX$CHECK_SUPPORT routine.
;   0151  0     !
;   0152  0     !       27      M. Baggett      18-Oct-1984     Version 7.1
;   0153  0     !               Add changes for HSC50 port for magtape operation.
;   0154  0     !
;   0155  0     !       28      M. BAGGETT      2-NOV-1984      VERSION 7.1
;   0156  0     !               Support TK50.  Change TU81 to have init flag.
;   0157  0     !               Change order of clearing init flag for all applicable
;   0158  0     !               devices so a CLI command will only clear it once per command.
;   0159  0     !
;   0160  0     !       29      M. BAGGETT      26-NOV-1984     Version 7.1
;   0161  0     !               Allow file operations from RUX50.
;   0162  0     !
;   0163  0     !       30      M. Baggett      22-FEB-1985     Version 8.1
;   0164  0     !               Start support for ULTRIX file structure.
;   0165  0     !
;   0166  0     !       31      Bob Bergazzi    7-May-1985      Version 8.2
;   0167  0     !               Add support to DSX$CHECK_SUPPORT for BCI750.
;   0168  0     !
;   0169  0     !       32      M. Baggett      21-May-1985     Version 8.2
;   0170  0     !               Added support for ULTRIX files.
;   0171  0     !
```

```
;   0172  0     !       33      R.E. MUSE       28-MAY-1985     Version 8.2
;   0173  0     !               Changed DSX$Check_support to support ODS format on
;   0174  0     !               the console storage device for the AAA cpu.
;   0175  0     !
;   0176  0     !       34      Bob Bergazzi    5-Jun-1985      Version 8.2
;   0177  0     !               Added support for KDB50.
;   0178  0     !
;   0179  0     !       35      Bob Bergazzi    11-June-1985    Version 8.3
;   0180  0     !               Changed BCI750 to CIBCI.
;   0181  0     !
;   0182  0     !       36      M. Baggett      12-Jun-1985     Version 8.3
;   0183  0     !               No default name in Ultrix.
;   0184  0     !
;   0185  0     !       37      M. Baggett      14-Aug-1985     Version 8.3
;   0186  0     !               Added KFBTA.
;   0187  0     !
;   0188  0     !       38      Bob Bergazzi    20-Aug-1985     Version 8.3
;   0189  0     !               Fix edit 33.
;   0190  0     !
;   0191  0     !       39      M. Baggett      22-Aug-1985     Version 9.0
;   0192  0     !               Ultrix updates.
;   0193  0     !
;   0194  0     !       40      Richard E. Muse  2-Oct-1985     Version 9.0
;   0195  0     !               Added code in routines DSX$CHECK_SUPPORT and
;   0196  0     !               RMS$OPEN for the NAUTILUS CPU. Each routine uses code
;   0197  0     !               to 'backtrack' through ptables till the _LINK field
;   0198  0     !               equals zero, when this occurs the present Ptable
;   0199  0     !               is linked to HUB and thus the adapter/controller's
;   0200  0     !               address is available.  Not so for a NAUTILUS,when
;   0201  0     !               the ptable _LINK field equals zero it means were linked to
;   0202  0     !               to an NBIA which in turn has an NBIB linked to it. Therfore
;   0203  0     !               the modifications have to do with saving and reloading
;   0204  0     !               adapter/controller address back two levels
;   0205  0     !               if running on a NAUTILUS.
;   0206  0     !
;   0207  0     !       41      J. Baranski     3-Feb-1986      Version 9.1
;   0208  0     !               Add RA82 to KBD50 and UDA50 Device Support Tables.
;   0209  0     !
;   0210  0     !       42      M. Baggett      11-Feb-1986     Version 10.0                    !G:4
;   0211  0     !               Change device type code for KFTBA to BTD$K_BVPSSP.              !G:2
;   0212  0     !                                                                               !G:4
;   0213  0     !       43      Ted Salem       4-March-1986    Version 10.0                    !G:4
;   0214  0     !               Modified $OPEN $CLOSE $READ $GET and DSX$CHECK_SUPPORT          !G:4
;   0215  0     !               to support the ANDROMEDA console using the PPA console          !G:4
;   0216  0     !               interface (PPIFACE.MAR) instead of a device driver.             !G:4
;   0217  0     !                                                                               !G:4
;   0218  0     !       44      J. Baranski     7-Mar-1986      Version 10.0                    !G:4
;   0219  0     !               Fix RA82 Device Support Entry.                                  !G:4
;   0220  0     !                                                                               !G:7
;   0221  0     !                                                                               !G:7
;   0222  0     !       45      J. Baranski     7-Oct-1986      Version 10.3                    !G:7
;   0223  0     !               Make DS$GB_SYSTYPE Addressing Mode Long Relative.               !G:7
;   0224  0     !                                                                               !G:8
;   0225  0     !       46      Ted Salem                                                       !G:19
;   0226  0     !               Changed the value of the device unit number from a              !G:8
;   0227  0     !               byte of xx, to a word of 80xx, IF the device is a shadowed set. !G:8
;   0228  0     !                                                                               !G:9
```

```
;    0229  0     !       47      Steve Meier     27-Aug-1986     Version 10.3                                    !G:9
;    0230  0     !               The same edit as 22 except for the XBI on the LLL                               !G:9
;    0231  0     !               machine. Since the BI is not directly linked to the XMI                         !G:9
;    0232  0     !               (the XBI adapter is stuck in the middle) To find the P-table                    !G:9
;    0233  0     !               of the BI we must follow the links back to zero (hub) and                       !G:9
;    0234  0     !               then take one step forward.                                                     !G:9
;    0235  0     !                                                                                               !G:10
;    0236  0     !       48      M. Baggett      25-Feb-1987     Version 10.5                                    !G:10
;    0237  0     !               Modification for TK50 booting.                                                  !G:10
;    0238  0     !                                                                                               !G:11
;    0239  0     !       49      Bob Bergazzi    4-Mar-1987      Version 10.7                                    !G:11
;    0240  0     !               Added load support for RA70.                                                    !G:11
;    0241  0     !                                                                                               !G:16
;    0242  0     !       50      Ingrid Martin   25-Mar-1987     Version 10.7                                    !G:16
;    0243  0     !               Updates for RRR - similar to edits 22, 47                                       !G:16
;    0244  0     !                                                                                               !G:16
;    0245  0     !       51      M. Baggett      7-Apr-1987      Version 10.7                                    !G:17
;    0246  0     !               No intialization flag for TK50. Add TK70                                        !G:17
;    0247  0     !                                                                                               !G:19
;    0248  0     !       52      M. Baggett      22-May-1987     Version 10.9                                    !G:19
;    0249  0     !               Add I/O support for RD54                                                        !G:19
;    0250  0     !                                                                                               !G:20
;    0251  0     !               David Gearin    26-Oct-1987     Version 10.10                                   !G:20
;    0252  0     !               Added code for the extended p-table format.  Drive                              !G:20
;    0253  0     !               unit #'s will now be extracted from HPE$W_EXT_DRIVE                             !G:20
;    0254  0     !               instead of HP$B_DRIVE.                                                          !G:20
;    0255  0     !--                                                                                             !G:19
;    0256  0                                                                                                     !G:19
;    0257  1     begin                                                                                           !G:19
;    0258  1     !
;    0259  1     ! include files:
;    0260  1     !                                                                                               !G:19
;    0261  1
;    0262  1     library '$diag';                !
;    0263  1
;    0264  1     library '$ds';                  !
;    0265  1
;    0266  1     library 'sys$library:lib.l32';
;    0267  1
;    0268  1     !
;    0269  1     ! table of contents:
;    0270  1     !
;    0271  1
;    0272  1     forward routine
;    0273  1         rms$error : jsb_rms,                        ! Load error status
;    0274  1         rms$load_xab : call_rms novalue,            ! Load XAB data
;    0275  1         rms$alloc_cache : jsb_cblock,               ! Allocate cache
;    0276  1         rms$init : jsb_none novalue,                ! Initialize
;    0277  1         rms$cleanup : jsb_none novalue,             ! Clean up RMS allocations
;    0278  1         rab$error : jsb_rms,                        ! Store error code in RAB
;    0279  1         fab$error : jsb_rms,                        ! Store error code in FAB
;    0280  1         verify_cblock : jsb_rms,                    ! Be sure IFI and cblock pointer are OK
;    0281  1         verify_rab : jsb_rms,                       ! Verify rab
;    0282  1         verify_fab : jsb_rms,                       ! Verify fab
;    0283  1         rms$open,                                   ! RMS$OPEN emulator--open file
;    0284  1         rms$connect,                                ! RMS$CONNECT emulator
;    0285  1         rms$get,                                    ! RMS$GET emulator--record de-block
```

```
;    0286  1          rms$disconnect,                          ! RMS$DISCONNECT emulator
;    0287  1          rms$close;                               ! RMS$CLOSE emulator--close file
;    0288  1
;    0289  1      !
;    0290  1      ! Local values
;    0291  1      !
;    0292  1
;    0293  1      literal
;    0294  1          True = 1,                                    ! Literal TRUE
;    0295  1          False = 0,                                   ! literal FALSE
;    0296  1          rms$c_maxfiles = 3,                      ! Maximum files open at once
;    0297  1          device$_ansi = dev$m_sqd or dev$m_fod,    ! Describe magtape
;    0298  1          device$_ods = dev$m_fod or dev$m_dir or dev$m_rnd,  ! Describe disk
;    0299  1          device$_rt11 = dev$m_dir or dev$m_rnd or dev$m_sdi or dev$m_fod,    ! Describe floppy
;    0300  1          ! Re-define Restart parameter block length as 16-byte multiple,
;    0301  1          ! to make allocation easier w/out requiring assumptions or
;    0302  1          ! explicitly storing the actual length.
;    0303  1          rpb$_length = (rpb$c_length + 15) and not 15,                                          !G:16
;    0304  1          pr$_sid_typ8RR = 17;                     ! until this appears in a macro [50]          !G:16
;    0305  1
;    0306  1      !
;    0307  1      ! external references:
;    0308  1      !
;    0309  1      $ds_dsadef;
;    0310  1
;    0311  1      external
;    0312  1          Ultrix_flags     : Byte addressing_mode (long_relative),    ! Flag for ULTRIX disk structure. [30]
;    0313  1          Ds$Gb_Uda50_Init : Byte weak,                ! Global init flag in PUBTDRIVR
;    0314  1          Ds$Gb_CI_Init : Byte Weak,                   ! Global init flag in PABTDRIVR
;    0315  1          Ds$Gb_KDB50_Init : Byte Weak,                ! Global init flag in BDABTDRIVR       [34]
;    0316  1          Ds$Gb_PB_Init : Byte Weak,                   ! Global init flag in PBBTDRIVR        [37]
;    0317  1          DS$GB_SYSTYPE : BYTE addressing_mode(long_relative),! Global byte of system type       [43]    !G:7
;    0318  1          def$q_dir : bblock addressing_mode (long_relative),
;    0319  1          def$q_dev : bblock addressing_mode (long_relative),
;    0320  1          boo$select,                          ! Non-interrupt "QIO" package stuff                   !G:8
;    0321  1          boo$al_vector : addressing_mode (long_relative);
;    0322  1                                                                                                 !G:4
;    0323  1      external literal PR$_SYS_TYP900 ;            ! System type for Andromeda     [43]       !G:5
;    0324  1                                                                                                 !G:5
;    0325  1      literal
;    0326  1      ultrix_v_mode = 0;      !  ultrix flag                                    [32]            !G:5
;    0327  1                                                                                                 !G:19
;    0328  1      MAP                                          !                                             [23] !G:4
;    0329  1              DSA$GL_SID : BLOCK;                  !                                             [23] !G:4
;    0330  1                                                                                                 !G:4
;    0331  1      external routine
;    0332  1          exe$deanonpaged : jsb_deall addressing_mode (long_relative),       ! Deallocate dyn. pool space
;    0333  1          exe$alononpaged : jsb_alloc addressing_mode (long_relative),       ! Allocate from dynamic pool
;    0334  1          fil$cvtfilnam,                           ! Convert ascii to RAD50
;    0335  1          BreakUP_File,
;    0336  1          loc$ptable : jsb_loc$ptable novalue addressing_mode (long_relative),
;    0337  1          rt11$open : call_rms,                    ! Access console RT-11 file
;    0338  1          ods$open : call_rms,                     ! Access disk file
;    0339  1          ultrix$open : call_rms addressing_mode (long_relative),            ! Access ultrix disk file[30]
;    0340  1          ansi$open : call_rms addressing_mode (long_relative),      ! Access magtape file       !G:4
;    0341  1          ppa$lookup : call_rms addressing_mode (long_relative) weak, ! Open an Andromeda console file [43]   !G:4
;    0342  1          ppa$read   : call_rms addressing_mode (long_relative) weak, ! Read a data block for Andromeda[43]   !G:4
```

```
;    0343  1        ppa$close  : call_rms addressing_mode (long_relative) weak; ! Close a Andromeda console file [43]   !G:4
;    0344  1    !
;    0345  1    ! psect definitions:
;    0346  1    !
;    0347  1
;    0348  1    psect
;    0349  1        plit = data ( share, addressing_mode (long_relative));
;    0350  1
;    0351  1    psect
;    0352  1        own = work ( read, write, addressing_mode (long_relative));
;    0353  1
;    0354  1    psect
;    0355  1    global = work ( read, write, addressing_mode (long_relative));                                          !G:20
;    0356  1
;    0357  1    psect
;    0358  1        code = code ( read, execute, share);
;    0359  1
;    0360  1    !
;    0361  1    ! Own storage
;    0362  1    !
;    0363  1
;    0364  1    global
;    0365  1        rms$file_table : vector [4];                ! ifi-indexed pointers
;    0366  1    !
;    0367  1    ! Global data definitions:
;    0368  1    !
;    0369  1
;    0370  1    global bind
;    0371  1        def$c_bkstp_len = %charcount (%ascii'CSA0:[SYSMAINT];0'),
;    0372  1        def$q_backstop = uplit byte(%ascii'CSA0:[SYSMAINT];0'),
;    0373  1        def$c_ultrix_len = %charcount (%ascii'CSA0:(0,6)field'),!                                  [32]
;    0374  1        def$q_ultrix = uplit byte(%ascii'CSA0:(0,6)field'); !                                      [32]
;    0375  1
```

```
;    0377  1     %Sbttl 'Device support table'
;    0378  1
;    0379  1     Compiletime                                                                                                    [13]
;    0380  1         $$_Rows_$$ = 0;                                                                                            [13]
;    0381  1
;    0382  1     Global Bind                                                                                                    [13]
;    0383  1         Ds$GA_Support_Table = Uplit (                                                                              [13]
;    0384  1             Support_Entry ('console', HUB, Btd$K_Console, RT11, <>, 0, <>),                                    ! Console       [13]
;    0385  1             Support_Entry ('console', HUB, Btd$K_Console, ODS, <>, 0, <>),                                     ! ZZZ,AAA console
;    0386  1             Support_Entry ('RP04', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                            ! RP04          [13]
;    0387  1             Support_Entry ('RP05', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                            ! RP05          [13]
;    0388  1             Support_Entry ('RP06', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                            ! RP06          [13]
;    0389  1             Support_Entry ('RP07', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                            ! RP07          [13]
;    0390  1             Support_Entry ('RM03', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                            ! RM03          [13]
;    0391  1             Support_Entry ('RM05', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                            ! RM05          [13]
;    0392  1             Support_Entry ('RM80', MBA, Btd$K_Mb, ODS, <A>, 0, <>),                                            ! RM80          [13]
;    0393  1             Support_Entry ('TM03', MBA, Btd$K_Tm, ANSI, <A, C>, 0, <'TE16', 'TU45', 'TU77'>),                  ! TM03          [13]
;    0394  1             Support_Entry ('TM78', MBA, Btd$K_Tf, ANSI, <A, C>, 0, <'TU78'>),                                  ! TM78          [13]
;    0395  1             Support_Entry ('RB730',UBA, Btd$K_Dq, ODS, <A, C>, 0, <'RL02', 'R80'>),                            ! IDC           [13]
;    0396  1             Support_Entry ('UDA50',UBA, Btd$K_UDA,ODS, <A, C>,Ds$GB_Uda50_Init,<'RA70','RA80','RA81','RA82','RA90','RA60
;    0397  1             Support_Entry ('RK611',UBA, Btd$K_Dm, ODS, <A, C>, 0, <'RK06', 'RK07'>),                           ! RK06/07       [13]
;    0398  1             Support_Entry ('RL11' ,UBA, Btd$K_Dl, ODS, <A, C>, 0, <'RL02'>),                                   ! RL02          [14]
;    0399  1             Support_Entry ('TS11', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                            ! TS04          [13]
;    0400  1             Support_Entry ('TS04', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                            ! TS04          [13]
;    0401  1             Support_Entry ('TS05', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                            ! TS05          [21]
;    0402  1             Support_Entry ('TU80', UBA, Btd$K_Ts, ANSI,<A>, 0, <>),                                            ! TU80          [15]
;    0403  1             Support_Entry ('TU81', UBA, Btd$K_Mu, ANSI,<A>, 0, <>),                                            ! TU81          [28]
;    0404  1             Support_Entry ('CI_NODE',CIA, Btd$K_HSCCI,ODS, <A, C>, DS$GB_CI_INIT, <'DISK'>),                   ! HSC50         [20]
;    0405  1             Support_Entry ('CI_NODE',CIA, Btd$K_HSCMU,ANSI, <A, C>, DS$GB_CI_INIT, <'TAPE'>),                  ! HSC50 TAPE    [25]
;    0406  1             Support_Entry ('LESI',UBA, Btd$K_UDA,ODS, <A, C>, Ds$GB_Uda50_Init, <'RC25','RCF25'>),             ! LESI          [12]
;    0407  1             Support_Entry ('RUX50',UBA, Btd$K_UDA,ODS, <A, C>, Ds$GB_Uda50_Init, <'RX31','RX32','RX50','RX180'>),!  [29]
;    0408  1             Support_Entry ('TK50', DEBNT,Btd$K_HSCCI, ANSI,<A>,0, <>),                                         ! TK50          [28]
;    0409  1             Support_Entry ('TK70', DEBNT,Btd$K_HSCCI, ANSI,<A>,0, <>),                                         ! TK70          [51]
;    0410  1             SUPPORT_ENTRY ('RA60',KDB50,BTD$K_BDA,ODS,<A>,DS$GB_KDB50_INIT,<>),                                ! KDB50 (BDA)   [34]
;    0411  1             SUPPORT_ENTRY ('RA70',KDB50,BTD$K_BDA,ODS,<A>,DS$GB_KDB50_INIT,<>),                                ! KDB50 (BDA)   [49]
;    0412  1             SUPPORT_ENTRY ('RA80',KDB50,BTD$K_BDA,ODS,<A>,DS$GB_KDB50_INIT,<>),                                ! KDB50 (BDA)   [34]
;    0413  1             SUPPORT_ENTRY ('RA81',KDB50,BTD$K_BDA,ODS,<A>,DS$GB_KDB50_INIT,<>),                                ! KDB50 (BDA)   [34]
;    0414  1             SUPPORT_ENTRY ('RA82',KDB50,BTD$K_BDA,ODS,<A>,DS$GB_KDB50_INIT,<>),                                ! KDB50 (BDA)   [41]
;    0415  1             SUPPORT_ENTRY ('RA90',KDB50,BTD$K_BDA,ODS,<A>,DS$GB_KDB50_INIT,<>),                                ! KDB50 (BDA)   [41]
;    0416  1             SUPPORT_ENTRY ('RX50',KFBTA,BTD$K_BVPSSP,ODS,<A>,DS$GB_PB_INIT,<>),                                ! KFBTA (BVP)   [37]
;    0417  1             SUPPORT_ENTRY ('RD52',KFBTA,BTD$K_BVPSSP,ODS,<A>,DS$GB_PB_INIT,<>),                                ! KFBTA (BVP)   [37]
;    0418  1             SUPPORT_ENTRY ('RD53',KFBTA,BTD$K_BVPSSP,ODS,<A>,DS$GB_PB_INIT,<>),                                ! KFBTA (BVP)   [37]
;    0419  1             SUPPORT_ENTRY ('RD54',KFBTA,BTD$K_BVPSSP,ODS,<A>,DS$GB_PB_INIT,<>)                                 ! KFBTA (BVP)   [52]
;    0420  1             );                                                                                                 [13]
;    0421  1
;    0422  1     Global Literal                                                                                                 [13]
;    0423  1         Ds$GK_Support_Table_Rows = $$_Rows_$$;                                                                     [13]
```

```
;    0425  1      %Sbttl 'Allocate pseudo RPB'
;    0426  1
;    0427  1      !++
;    0428  1      ! Functional description:
;    0429  1      !       Allocate a block of nonpaged pool and initialize it to resemble an
;    0430  1      !       RPB data structure.  Note that it is NOT an RPB for a number of
;    0431  1      !       reasons, including the fact that it may not be page-aligned.  It is
;    0432  1      !       close enough to an RPB to make the VMS standalone BOOTDRIVR I/O
;    0433  1      !       mechanism function correctly.
;    0434  1      !                                                                          !G:19
;    0435  1      ! Input Parameters:
;    0436  1      !       04(Ap)          DeviceType (type code for device [DM, DL, MB,...])
;    0437  1      !       08(Ap)          ADPType (type code for adapter [MBA, UBA])
;    0438  1      !       12(Ap)          ADPPhy (physical/virtual address of adapter)
;    0439  1      !       16(Ap)          CSRPhy (physical/virtual address of controller)
;    0440  1      !       20(Ap)          Drive (Unit number of drive)
;    0441  1      !       24(AP)          Slave unit number
;    0442  1      !       28(AP)          Port number on CI network for HSC50.                [19]
;    0443  1      !       32(AP)          BI node number                                      [37]
;    0444  1      !
;    0445  1      ! Outputs:
;    0446  1      !       R0              0 if couldn't allocate RPB; else address of RPB
;    0447  1      !--
;    0448  1
;    0449  1      Global Routine Dsr$Allocate_Pseudo_RPB (        !                          [10]
;    0450  1          DeviceType, ADPType, CSRPhy, ADPPhy, Drive, Slave, Ciport, binode) =    !  [10]
;    0451  2          Begin                          !                                      [10]
;    0452  2
;    0453  2          Builtin                        !                                      [10]
;    0454  2              Mfpr;                      !                                      [10]
;    0455  2                                                                                !G:19
;    0456  2          Local                          !                                      [10]
;    0457  2              RPB : Ref Block [, Byte];      !                                  [10]
;    0458  2                                                                                !G:19
;    0459  2          If Not Exe$AloNonPaged (RPB$_Length;, RPB)  !                         [10]
;    0460  2          Then                           !                                      [10]
;    0461  2              Return 0;                  !                                      [10]
;    0462  2
;    0463  2          Ch$Fill (0, RPB$_Length, .RPB);     ! Zero it for good measure        [10]
;    0464  2          .RPB = .RPB;                   ! First longword is own address       [10]
;    0465  2          RPB [RPB$B_DevTyp] = .DeviceType;   !                                 [10]
;    0466  2          RPB [RPB$L_BootR1] = .binode;       !                                 [37]
;    0467  2          RPB [RPB$L_IOVec] = Boo$AL_Vector;  !                                 [10]
;    0468  2          (RPB [RPB$B_ConfReg])<0, 8> = .ADPType;     !                         [10]
;    0469  2          RPB [RPB$L_CSRPhy] = RPB [RPB$L_CSRVir] = .CSRPhy;  !                 [10]
;    0470  2          RPB [RPB$L_ADPPhy] = RPB [RPB$L_ADPVir] = .ADPPhy;  !                 [10]
;    0471  2          RPB [RPB$W_Unit] = .Drive;  !                                         [10]
;    0472  2          RPB [RPB$B_Slave] = .Slave; !                                         [10]
;    0473  2          RPB [RPB$L_BOOTR2] = .Ciport; ! HSC50 CI port number.                 [19]
;    0474  2          Boo$AL_Vector + 8 = Boo$Select - Boo$AL_Vector;     ! Initialize the entry.  [10]
;    0475  2
;    0476  2          If Not .Dsa$V_User Then Mfpr (Pr$_Sbr, RPB [RPB$L_Svaspt]); ! Get sys page table addr.  [10]
;    0477  2
;    0478  2          .RPB                           !                                      [10]
;    0479  1          End;                           !                                      [10]
```

```
                                                                                .TITLE  RMS *** RMS Master RMS routines
                                                                                .IDENT  \10.10-20\

                                                                                .PSECT  WORK,NOEXE,2

                                                                  00000 RMS$FILE_TABLE::
                                                                                .BLKB   16

                                                                                .PSECT  DATA,NOWRT,NOEXE,  SHR,2

5D  54  4E  49  41  4D  53  59  53  5B  3A  30  41  53  43  00000 P.AAA:  .ASCII  \CSA0:[SYSMAINT];0\                        ;
                                                        30  3B  0000F                                                        ;
64  6C  65  69  66  29  36  2C  30  28  3A  30  41  53  43  00011 P.AAB:  .ASCII  \CSA0:(0,6)field\                          ;
                            65  6C  6F  73  6E  6F  63  07  00020 P.AAD:  .ASCII  <7>\console\                                ;
                            65  6C  6F  73  6E  6F  63  07  00028 P.AAE:  .ASCII  <7>\console\                                ;
                                        34  30  50  52  04  00030 P.AAF:  .ASCII  <4>\RP04\                                   ;
                                        35  30  50  52  04  00035 P.AAG:  .ASCII  <4>\RP05\                                   ;
                                        36  30  50  52  04  0003A P.AAH:  .ASCII  <4>\RP06\                                   ;
                                        37  30  50  52  04  0003F P.AAI:  .ASCII  <4>\RP07\                                   ;
                                        33  30  4D  52  04  00044 P.AAJ:  .ASCII  <4>\RM03\                                   ;
                                        35  30  4D  52  04  00049 P.AAK:  .ASCII  <4>\RM05\                                   ;
                                        30  38  4D  52  04  0004E P.AAL:  .ASCII  <4>\RM80\                                   ;
                                        33  30  4D  54  04  00053 P.AAM:  .ASCII  <4>\TM03\                                   ;
                                        36  31  45  54  04  00058 P.AAO:  .ASCII  <4>\TE16\                                   ;
                                        35  34  55  54  04  0005D P.AAP:  .ASCII  <4>\TU45\                                   ;
                                        37  37  55  54  04  00062 P.AAQ:  .ASCII  <4>\TU77\                                   ;
                                                            00067         .BLKB   1
                        00000000' 00000000' 00000000'       00068 P.AAN:  .ADDRESS P.AAO, P.AAP, P.AAQ                        ;
                                            00000000        00074         .LONG   0                                           ;
                                        38  37  4D  54  04  00078 P.AAR:  .ASCII  <4>\TM78\                                   ;
                                        38  37  55  54  04  0007D P.AAT:  .ASCII  <4>\TU78\                                   ;
                                                            00082         .BLKB   2
                                            00000000'       00084 P.AAS:  .ADDRESS P.AAT                                      ;
                                            00000000        00088         .LONG   0                                           ;
                                    30  33  37  42  52  05  0008C P.AAU:  .ASCII  <5>\RB730\                                  ;
                                        32  30  4C  52  04  00092 P.AAW:  .ASCII  <4>\RL02\                                   ;
                                            30  38  52  03  00097 P.AAX:  .ASCII  <3>\R80\                                    ;
                                                            0009B         .BLKB   1
                                        00000000' 00000000' 0009C P.AAV:  .ADDRESS P.AAW, P.AAX                               ;
                                            00000000        000A4         .LONG   0                                           ;
                                    30  35  41  44  55  05  000A8 P.AAY:  .ASCII  <5>\UDA50\                                  ;
                                        30  37  41  52  04  000AE P.ABA:  .ASCII  <4>\RA70\                                   ;
                                        30  38  41  52  04  000B3 P.ABB:  .ASCII  <4>\RA80\                                   ;
                                        31  38  41  52  04  000B8 P.ABC:  .ASCII  <4>\RA81\                                   ;
                                        32  38  41  52  04  000BD P.ABD:  .ASCII  <4>\RA82\                                   ;
                                        30  39  41  52  04  000C2 P.ABE:  .ASCII  <4>\RA90\                                   ;
                                        30  36  41  52  04  000C7 P.ABF:  .ASCII  <4>\RA60\                                   ;
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 000CC P.AAZ:  .ADDRESS P.ABA, P.ABB, P.ABC, P.ABD, P.ABE, P.ABF   ;
                                            00000000        000E4         .LONG   0                                           ;
                                    31  31  36  4B  52  05  000E8 P.ABG:  .ASCII  <5>\RK611\                                  ;
                                        36  30  4B  52  04  000EE P.ABI:  .ASCII  <4>\RK06\                                   ;
                                        37  30  4B  52  04  000F3 P.ABJ:  .ASCII  <4>\RK07\                                   ;
                                        00000000' 00000000' 000F8 P.ABH:  .ADDRESS P.ABI, P.ABJ                               ;
                                            00000000        00100         .LONG   0                                           ;
                                        31  31  4C  52  04  00104 P.ABK:  .ASCII  <4>\RL11\                                   ;
                                        32  30  4C  52  04  00109 P.ABM:  .ASCII  <4>\RL02\                                   ;
                                                            0010E         .BLKB   2
```

```
                                  00000000'  00110 P.ABL:  .ADDRESS P.ABM                                  ;
                                  00000000   00114         .LONG   0                                       ;
                        31  31  53  54  04   00118 P.ABN:  .ASCII  <4>\TS11\                               ;
                        34  30  53  54  04   0011D P.ABO:  .ASCII  <4>\TS04\                               ;
                        35  30  53  54  04   00122 P.ABP:  .ASCII  <4>\TS05\                               ;
                        30  38  55  54  04   00127 P.ABQ:  .ASCII  <4>\TU80\                               ;
                        31  38  55  54  04   0012C P.ABR:  .ASCII  <4>\TU81\                               ;
                45  44  4F  4E  5F  49  43  07   00131 P.ABS:  .ASCII  <7>\CI_NODE\                           ;
                        4B  53  49  44  04   00139 P.ABU:  .ASCII  <4>\DISK\                               ;
                                             0013E         .BLKB   2
                                  00000000'  00140 P.ABT:  .ADDRESS P.ABU                                  ;
                                  00000000   00144         .LONG   0                                       ;
                45  44  4F  4E  5F  49  43  07   00148 P.ABV:  .ASCII  <7>\CI_NODE\                           ;
                        45  50  41  54  04   00150 P.ABX:  .ASCII  <4>\TAPE\                               ;
                                             00155         .BLKB   3
                                  00000000'  00158 P.ABW:  .ADDRESS P.ABX                                  ;
                                  00000000   0015C         .LONG   0                                       ;
                        49  53  45  4C  04   00160 P.ABY:  .ASCII  <4>\LESI\                               ;
                        35  32  43  52  04   00165 P.ACA:  .ASCII  <4>\RC25\                               ;
                    35  32  46  43  52  05   0016A P.ACB:  .ASCII  <5>\RCF25\                              ;
                        00000000' 00000000'  00170 P.ABZ:  .ADDRESS P.ACA, P.ACB                          ;
                                  00000000   00178         .LONG   0                                       ;
                    30  35  58  55  52  05   0017C P.ACC:  .ASCII  <5>\RUX50\                              ;
                        31  33  58  52  04   00182 P.ACE:  .ASCII  <4>\RX31\                               ;
                        32  33  58  52  04   00187 P.ACF:  .ASCII  <4>\RX32\                               ;
                        30  35  58  52  04   0018C P.ACG:  .ASCII  <4>\RX50\                               ;
                    30  38  31  58  52  05   00191 P.ACH:  .ASCII  <5>\RX180\                              ;
                                             00197         .BLKB   1
        00000000' 00000000' 00000000' 00000000'  00198 P.ACD:  .ADDRESS P.ACE, P.ACF, P.ACG, P.ACH          ;
                                  00000000   001A8         .LONG   0                                       ;
                        30  35  4B  54  04   001AC P.ACI:  .ASCII  <4>\TK50\                               ;
                        30  37  4B  54  04   001B1 P.ACJ:  .ASCII  <4>\TK70\                               ;
                        30  36  41  52  04   001B6 P.ACK:  .ASCII  <4>\RA60\                               ;
                        30  37  41  52  04   001BB P.ACL:  .ASCII  <4>\RA70\                               ;
                        30  38  41  52  04   001C0 P.ACM:  .ASCII  <4>\RA80\                               ;
                        31  38  41  52  04   001C5 P.ACN:  .ASCII  <4>\RA81\                               ;
                        32  38  41  52  04   001CA P.ACO:  .ASCII  <4>\RA82\                               ;
                        30  39  41  52  04   001CF P.ACP:  .ASCII  <4>\RA90\                               ;
                        30  35  58  52  04   001D4 P.ACQ:  .ASCII  <4>\RX50\                               ;
                        32  35  44  52  04   001D9 P.ACR:  .ASCII  <4>\RD52\                               ;
                        33  35  44  52  04   001DE P.ACS:  .ASCII  <4>\RD53\                               ;
                        34  35  44  52  04   001E3 P.ACT:  .ASCII  <4>\RD54\                               ;
                                  00000000'  001E8 P.AAC:  .ADDRESS P.AAD                                  ;
                                        00   001EC         .BYTE   0                                       ;
                                        40   001ED         .BYTE   64                                      ;
                                        03   001EE         .BYTE   3                                       ;
                                        00   001EF         .BYTE   0                                       ;
                                  00000000   001F0         .LONG   0                                       ;
                                  00000000   001F4         .LONG   0                                       ;
                                  00000000'  001F8         .ADDRESS P.AAE                                  ;
                                        00   001FC         .BYTE   0                                       ;
                                        40   001FD         .BYTE   64                                      ;
                                        02   001FE         .BYTE   2                                       ;
                                        00   001FF         .BYTE   0                                       ;
                                  00000000   00200         .LONG   0                                       ;
                                  00000000   00204         .LONG   0                                       ;
                                  00000000'  00208         .ADDRESS P.AAF                                  ;
```

```
                20  0020C        .BYTE   32                                  ;
                00  0020D        .BYTE   0                                   ;
                02  0020E        .BYTE   2                                   ;
                01  0020F        .BYTE   1                                   ;
          00000000  00210        .LONG   0                                   ;
          00000000  00214        .LONG   0                                   ;
          00000000' 00218        .ADDRESS P.AAG                              ;
                20  0021C        .BYTE   32                                  ;
                00  0021D        .BYTE   0                                   ;
                02  0021E        .BYTE   2                                   ;
                01  0021F        .BYTE   1                                   ;
          00000000  00220        .LONG   0                                   ;
          00000000  00224        .LONG   0                                   ;
          00000000' 00228        .ADDRESS P.AAH                              ;
                20  0022C        .BYTE   32                                  ;
                00  0022D        .BYTE   0                                   ;
                02  0022E        .BYTE   2                                   ;
                01  0022F        .BYTE   1                                   ;
          00000000  00230        .LONG   0                                   ;
          00000000  00234        .LONG   0                                   ;
          00000000' 00238        .ADDRESS P.AAI                              ;
                20  0023C        .BYTE   32                                  ;
                00  0023D        .BYTE   0                                   ;
                02  0023E        .BYTE   2                                   ;
                01  0023F        .BYTE   1                                   ;
          00000000  00240        .LONG   0                                   ;
          00000000  00244        .LONG   0                                   ;
          00000000' 00248        .ADDRESS P.AAJ                              ;
                20  0024C        .BYTE   32                                  ;
                00  0024D        .BYTE   0                                   ;
                02  0024E        .BYTE   2                                   ;
                01  0024F        .BYTE   1                                   ;
          00000000  00250        .LONG   0                                   ;
          00000000  00254        .LONG   0                                   ;
          00000000' 00258        .ADDRESS P.AAK                              ;
                20  0025C        .BYTE   32                                  ;
                00  0025D        .BYTE   0                                   ;
                02  0025E        .BYTE   2                                   ;
                01  0025F        .BYTE   1                                   ;
          00000000  00260        .LONG   0                                   ;
          00000000  00264        .LONG   0                                   ;
          00000000' 00268        .ADDRESS P.AAL                              ;
                20  0026C        .BYTE   32                                  ;
                00  0026D        .BYTE   0                                   ;
                02  0026E        .BYTE   2                                   ;
                01  0026F        .BYTE   1                                   ;
          00000000  00270        .LONG   0                                   ;
          00000000  00274        .LONG   0                                   ;
          00000000' 00278        .ADDRESS P.AAM                              ;
                20  0027C        .BYTE   32                                  ;
                B1  0027D        .BYTE   -79                                 ;
                01  0027E        .BYTE   1                                   ;
                03  0027F        .BYTE   3                                   ;
          00000000  00280        .LONG   0                                   ;
          00000000' 00284        .ADDRESS P.AAN                              ;
          00000000' 00288        .ADDRESS P.AAR                              ;
                20  0028C        .BYTE   32                                  ;
```

```
                B3  0028D       .BYTE   -77                                     ;
                01  0028E       .BYTE   1                                       ;
                03  0028F       .BYTE   3                                       ;
          00000000  00290       .LONG   0                                       ;
          00000000' 00294       .ADDRESS P.AAS                                  ;
          00000000' 00298       .ADDRESS P.AAU                                  ;
                00  0029C       .BYTE   0                                       ;
                03  0029D       .BYTE   3                                       ;
                02  0029E       .BYTE   2                                       ;
                03  0029F       .BYTE   3                                       ;
          00000000  002A0       .LONG   0                                       ;
          00000000' 002A4       .ADDRESS P.AAV                                  ;
          00000000' 002A8       .ADDRESS P.AAY                                  ;
                00  002AC       .BYTE   0                                       ;
                11  002AD       .BYTE   17                                      ;
                02  002AE       .BYTE   2                                       ;
                03  002AF       .BYTE   3                                       ;
          00000000G 002B0       .ADDRESS DS$GB_UDA50_INIT                       ;
          00000000' 002B4       .ADDRESS P.AAZ                                  ;
          00000000' 002B8       .ADDRESS P.ABG                                  ;
                00  002BC       .BYTE   0                                       ;
                01  002BD       .BYTE   1                                       ;
                02  002BE       .BYTE   2                                       ;
                03  002BF       .BYTE   3                                       ;
          00000000  002C0       .LONG   0                                       ;
          00000000' 002C4       .ADDRESS P.ABH                                  ;
          00000000' 002C8       .ADDRESS P.ABK                                  ;
                00  002CC       .BYTE   0                                       ;
                02  002CD       .BYTE   2                                       ;
                02  002CE       .BYTE   2                                       ;
                03  002CF       .BYTE   3                                       ;
          00000000  002D0       .LONG   0                                       ;
          00000000' 002D4       .ADDRESS P.ABL                                  ;
          00000000' 002D8       .ADDRESS P.ABN                                  ;
                00  002DC       .BYTE   0                                       ;
                B2  002DD       .BYTE   -78                                     ;
                01  002DE       .BYTE   1                                       ;
                01  002DF       .BYTE   1                                       ;
          00000000  002E0       .LONG   0                                       ;
          00000000  002E4       .LONG   0                                       ;
          00000000' 002E8       .ADDRESS P.ABO                                  ;
                00  002EC       .BYTE   0                                       ;
                B2  002ED       .BYTE   -78                                     ;
                01  002EE       .BYTE   1                                       ;
                01  002EF       .BYTE   1                                       ;
          00000000  002F0       .LONG   0                                       ;
          00000000  002F4       .LONG   0                                       ;
          00000000' 002F8       .ADDRESS P.ABP                                  ;
                00  002FC       .BYTE   0                                       ;
                B2  002FD       .BYTE   -78                                     ;
                01  002FE       .BYTE   1                                       ;
                01  002FF       .BYTE   1                                       ;
          00000000  00300       .LONG   0                                       ;
          00000000  00304       .LONG   0                                       ;
          00000000' 00308       .ADDRESS P.ABQ                                  ;
                00  0030C       .BYTE   0                                       ;
                B2  0030D       .BYTE   -78                                     ;
```

```
                01  0030E          .BYTE   1                                       ;
                01  0030F          .BYTE   1                                       ;
          00000000  00310          .LONG   0                                       ;
          00000000  00314          .LONG   0                                       ;
          00000000' 00318          .ADDRESS P.ABR                                  ;
                00  0031C          .BYTE   0                                       ;
                B4  0031D          .BYTE   -76                                     ;
                01  0031E          .BYTE   1                                       ;
                01  0031F          .BYTE   1                                       ;
          00000000  00320          .LONG   0                                       ;
          00000000  00324          .LONG   0                                       ;
          00000000' 00328          .ADDRESS P.ABS                                  ;
                00  0032C          .BYTE   0                                       ;
                20  0032D          .BYTE   32                                      ;
                02  0032E          .BYTE   2                                       ;
                03  0032F          .BYTE   3                                       ;
          00000000G 00330          .ADDRESS DS$GB_CI_INIT                          ;
          00000000' 00334          .ADDRESS P.ABT                                  ;
          00000000' 00338          .ADDRESS P.ABV                                  ;
                00  0033C          .BYTE   0                                       ;
                B5  0033D          .BYTE   -75                                     ;
                01  0033E          .BYTE   1                                       ;
                03  0033F          .BYTE   3                                       ;
          00000000G 00340          .ADDRESS DS$GB_CI_INIT                          ;
          00000000' 00344          .ADDRESS P.ABW                                  ;
          00000000' 00348          .ADDRESS P.ABY                                  ;
                00  0034C          .BYTE   0                                       ;
                11  0034D          .BYTE   17                                      ;
                02  0034E          .BYTE   2                                       ;
                03  0034F          .BYTE   3                                       ;
          00000000G 00350          .ADDRESS DS$GB_UDA50_INIT                       ;
          00000000' 00354          .ADDRESS P.ABZ                                  ;
          00000000' 00358          .ADDRESS P.ACC                                  ;
                00  0035C          .BYTE   0                                       ;
                11  0035D          .BYTE   17                                      ;
                02  0035E          .BYTE   2                                       ;
                03  0035F          .BYTE   3                                       ;
          00000000G 00360          .ADDRESS DS$GB_UDA50_INIT                       ;
          00000000' 00364          .ADDRESS P.ACD                                  ;
          00000000' 00368          .ADDRESS P.ACI                                  ;
                00  0036C          .BYTE   0                                       ;
                20  0036D          .BYTE   32                                      ;
                01  0036E          .BYTE   1                                       ;
                01  0036F          .BYTE   1                                       ;
          00000000  00370          .LONG   0                                       ;
          00000000  00374          .LONG   0                                       ;
          00000000' 00378          .ADDRESS P.ACJ                                  ;
                00  0037C          .BYTE   0                                       ;
                20  0037D          .BYTE   32                                      ;
                01  0037E          .BYTE   1                                       ;
                01  0037F          .BYTE   1                                       ;
          00000000  00380          .LONG   0                                       ;
          00000000  00384          .LONG   0                                       ;
          00000000' 00388          .ADDRESS P.ACK                                  ;
                00  0038C          .BYTE   0                                       ;
                21  0038D          .BYTE   33                                      ;
                02  0038E          .BYTE   2                                       ;
```

```
                01  0038F        .BYTE   1                                   ;
        00000000G  00390        .ADDRESS DS$GB_KDB50_INIT                   ;
        00000000   00394        .LONG   0                                   ;
        00000000'  00398        .ADDRESS P.ACL                              ;
                00  0039C        .BYTE   0                                   ;
                21  0039D        .BYTE   33                                  ;
                02  0039E        .BYTE   2                                   ;
                01  0039F        .BYTE   1                                   ;
        00000000G  003A0        .ADDRESS DS$GB_KDB50_INIT                   ;
        00000000   003A4        .LONG   0                                   ;
        00000000'  003A8        .ADDRESS P.ACM                              ;
                00  003AC        .BYTE   0                                   ;
                21  003AD        .BYTE   33                                  ;
                02  003AE        .BYTE   2                                   ;
                01  003AF        .BYTE   1                                   ;
        00000000G  003B0        .ADDRESS DS$GB_KDB50_INIT                   ;
        00000000   003B4        .LONG   0                                   ;
        00000000'  003B8        .ADDRESS P.ACN                              ;
                00  003BC        .BYTE   0                                   ;
                21  003BD        .BYTE   33                                  ;
                02  003BE        .BYTE   2                                   ;
                01  003BF        .BYTE   1                                   ;
        00000000G  003C0        .ADDRESS DS$GB_KDB50_INIT                   ;
        00000000   003C4        .LONG   0                                   ;
        00000000'  003C8        .ADDRESS P.ACO                              ;
                00  003CC        .BYTE   0                                   ;
                21  003CD        .BYTE   33                                  ;
                02  003CE        .BYTE   2                                   ;
                01  003CF        .BYTE   1                                   ;
        00000000G  003D0        .ADDRESS DS$GB_KDB50_INIT                   ;
        00000000   003D4        .LONG   0                                   ;
        00000000'  003D8        .ADDRESS P.ACP                              ;
                00  003DC        .BYTE   0                                   ;
                21  003DD        .BYTE   33                                  ;
                02  003DE        .BYTE   2                                   ;
                01  003DF        .BYTE   1                                   ;
        00000000G  003E0        .ADDRESS DS$GB_KDB50_INIT                   ;
        00000000   003E4        .LONG   0                                   ;
        00000000'  003E8        .ADDRESS P.ACQ                              ;
                00  003EC        .BYTE   0                                   ;
                22  003ED        .BYTE   34                                  ;
                02  003EE        .BYTE   2                                   ;
                01  003EF        .BYTE   1                                   ;
        00000000G  003F0        .ADDRESS DS$GB_PB_INIT                      ;
        00000000   003F4        .LONG   0                                   ;
        00000000'  003F8        .ADDRESS P.ACR                              ;
                00  003FC        .BYTE   0                                   ;
                22  003FD        .BYTE   34                                  ;
                02  003FE        .BYTE   2                                   ;
                01  003FF        .BYTE   1                                   ;
        00000000G  00400        .ADDRESS DS$GB_PB_INIT                      ;
        00000000   00404        .LONG   0                                   ;
        00000000'  00408        .ADDRESS P.ACS                              ;
                00  0040C        .BYTE   0                                   ;
                22  0040D        .BYTE   34                                  ;
                02  0040E        .BYTE   2                                   ;
                01  0040F        .BYTE   1                                   ;
```

```
00000000G 00410          .ADDRESS DS$GB_PB_INIT                                 ;
00000000  00414          .LONG    0                                             ;
00000000' 00418          .ADDRESS P.ACT                                         ;
       00 0041C          .BYTE    0                                             ;
       22 0041D          .BYTE    34                                            ;
       02 0041E          .BYTE    2                                             ;
       01 0041F          .BYTE    1                                             ;
00000000G 00420          .ADDRESS DS$GB_PB_INIT                                 ;
00000000  00424          .LONG    0                                             ;

                         DSA$AL_APTMAIL=     65024
                         DSA$GL_FLAGS=       65024
                         DSA$GL_APTCOM=      65028
                         DSA$GL_PASSES=      65032
                         DSA$GL_UNITS=       65036
                         DSA$GL_SECTNO=      65040
                         DSA$GL_SID=         65044
                         DSA$GL_MSGTYP=      65088
                         DSA$GL_ERRNO=       65092
                         DSA$GL_EVENT=       65096
                         DSA$GL_SUBTNO=      65100
                         DSA$GL_TESTNO=      65104
                         DSA$GL_PASSNO=      65108
                         DSA$GL_DEVLEN=      65112
                         DSA$GL_DEVNAM=      65116
                         DSA$GL_MSGPTR=      65128
                         DEF$C_BKSTP_LEN==   17
                         DEF$Q_BACKSTOP==    P.AAA
                         DEF$C_ULTRIX_LEN==  15
                         DEF$Q_ULTRIX==      P.AAB
                         DS$GA_SUPPORT_TABLE==
                                             P.AAC
                         DS$GK_SUPPORT_TABLE_ROWS==
                                             36
                                 .EXTRN  ULTRIX_FLAGS, DS$GB_SYSTYPE
                                 .EXTRN  DEF$Q_DIR, DEF$Q_DEV
                                 .EXTRN  BOO$SELECT, BOO$AL_VECTOR
                                 .EXTRN  PR$_SYS_TYP900, EXE$DEANONPAGED
                                 .EXTRN  EXE$ALONONPAGED
                                 .EXTRN  FIL$CVTFILNAM, BREAKUP_FILE
                                 .EXTRN  LOC$PTABLE, RT11$OPEN
                                 .EXTRN  ODS$OPEN, ULTRIX$OPEN
                                 .EXTRN  ANSI$OPEN
                                 .WEAK   DS$GB_UDA50_INIT
                                 .WEAK   DS$GB_CI_INIT, DS$GB_KDB50_INIT
                                 .WEAK   DS$GB_PB_INIT, PPA$LOOKUP
                                 .WEAK   PPA$READ, PPA$CLOSE

                                 .PSECT  CODE,NOWRT,  SHR,2

                    0FFC 00000   .ENTRY  DSR$ALLOCATE_PSEUDO_RPB, Save R2,R3,R4,R5,- ; 0449
                                         R6,R7,R8,R9,R10,R11                        ;
51      0110   8F  3C 00002     MOVZWL  #272, R1                                    ; 0459
   00000000G   EF  16 00007     JSB     EXE$ALONONPAGED                             ;
56              52  D0 0000D     MOVL    R2, R6                                      ;
03              50  E8 00010     BLBS    R0, 1$                                      ;
                50  D4 00013     CLRL    R0                                          ; 0461
```

```
                                                    04 00015          RET                                                             ;
    0110   8F          00          6E                00 2C 00016 1$:   MOVC5   #0, (SP), #0, #272, (RPB)                       ; 0463
                                                    66     0001D                                                                    ;
                                    66                56 D0 0001E       MOVL    RPB, (RPB)                                        ; 0464
                              66    A6        04      AC 90 00021       MOVB    DEVICETYPE, 102(RPB)                              ; 0465
                              20    A6        20      AC D0 00026       MOVL    BINODE, 32(RPB)                                   ; 0466
                              34    A6 00000000G      EF 9E 0002B       MOVAB   BOO$AL_VECTOR, 52(RPB)                            ; 0467
                            0090    C6        08      AC 90 00033       MOVB    ADPTYPE, 144(RPB)                                 ; 0468
                                    50        0C      AC D0 00039       MOVL    CSRPHY, R0                                        ; 0469
                              58    A6                50 D0 0003D       MOVL    R0, 88(RPB)                                       ;
                              54    A6                50 D0 00041       MOVL    R0, 84(RPB)                                       ;
                                    50        10      AC D0 00045       MOVL    ADPPHY, R0                                        ; 0470
                              60    A6                50 D0 00049       MOVL    R0, 96(RPB)                                       ;
                              5C    A6                50 D0 0004D       MOVL    R0, 92(RPB)                                       ;
                              64    A6        14      AC B0 00051       MOVW    DRIVE, 100(RPB)                                   ; 0471
                              67    A6        18      AC 90 00056       MOVB    SLAVE, 103(RPB)                                   ; 0472
                              24    A6        1C      AC D0 0005B       MOVL    CIPORT, 36(RPB)                                   ; 0473
                      00000000G   EF 00000000*  8F    D0 00060          MOVL    #<BOO$SELECT-BOO$AL_VECTOR>, -                    ; 0474
                                                                                 BOO$AL_VECTOR+8                                   ;
                    04 0000FE03   9F                04 E0 0006B       BBS     #4, @#^X0000FE03, 2$                              ; 0476
                              50    A6                0C DB 00073       MFPR    #12, 80(RPB)                                      ;
                                    50                56 D0 00077 2$:   MOVL    RPB, R0                                           ; 0479
                                                    04 0007A          RET                                                             ;

; Routine Size:  123 bytes,    Routine Base:  CODE + 0000
```

```
;   0481  1     %Sbttl 'Deallocate RPB'
;   0482  1
;   0483  1     Global Routine Dsr$Deallocate_Pseudo_RPB (RPB) =        !                                                [10]
;   0484  2         Begin                          !                                                                    [10]
;   0485  2         (.RPB + 8)<0, 16> = RPB$_Length;    !                                                               [10]
;   0486  2         Exe$DeaNonPaged (.RPB)         !                                                                    [10]
;   0487  1         End;                           !                                                                    [10]


                                                  0FFC 00000        .ENTRY  DSR$DEALLOCATE_PSEUDO_RPB, Save R2,R3,R4,-   ; 0483
                                                                            R5,R6,R7,R8,R9,R10,R11                       ;
                                  50       04   AC  D0 00002        MOVL    RPB, R0                                      ; 0485
                              08  A0     0110   8F  B0 00006        MOVW    #272, 8(R0)                                  ;
                                   00000000G   EF  16 0000C        JSB     EXE$DEANONPAGED                              ; 0486
                                                   04 00012        RET                                                  ; 0487

; Routine Size:  19 bytes,    Routine Base:  CODE + 007B
```

```
;    0489  1      %Sbttl 'Check device support tables'
;    0490  1
;    0491  1      !+
;    0492  1      ! Function:
;    0493  1      !       Search device support tables for the controller/device specified: return various
;    0494  1      !       information on the pair.
;    0495  1      !
;    0496  1      ! Calling sequence:
;    0497  1      !       status = DSX$CHECK_SUPPORT (Device, Adapter, Btd, File, Init)
;    0498  1      !
;    0499  1      ! Inputs:
;    0500  1      !       Device          => Address of device Ptable
;    0501  1      !
;    0502  1      ! Outputs:
;    0503  1      !       Adc             => Address to receive adapter code (BOOTDRIVR)
;    0504  1      !       Btd             => Address to receive boot device type code
;    0505  1      !       File            => Address to receive file structure type code
;    0506  1      !       Init            => Address to recieve address of driver init flag byte
;    0507  1      !
;    0508  1      ! Status:
;    0509  1      !       SS$_Normal      => OK
;    0510  1      !       DS$_Error       => Device is not supported by file services
;    0511  1      !       DS$_ProgErr     => Error in support table (SPT$V_CONTROLLER set but no drive name list, or bad file structur
;    0512  1      !-
;    0513  1
;    0514  1      Global Routine DSX$Check_Support (Device, Adc, Btd, File, Init) =       !                                          [13]
;    0515  2          Begin                                                               !                                          [13]
;    0516  2
;    0517  2          Builtin                                                             !                                          [13]
;    0518  2              NullParameter;                                                  ! Determine if parameter was specified     [13]
;    0519  2
;    0520  2          Map                                                                 !                                          [13]
;    0521  2              Device : Ref $Ds_Hpo_Decl ();                                                                  !G:20
;    0522  2
;    0523  2          Local                                                                                              !G:20
;    0524  2              Adapter_epb: Ref $Ds_Hpeo_decl (),                                                             !G:20
;    0525  2              Controller_Epb: Ref $Ds_Hpeo_decl (),                                          !
;    0526  2              Device_epb : Ref $Ds_Hpeo_decl (),                                  !
;    0527  2              Controller : Ref $Ds_Hpo_Decl (),                                  ! Pointer to controller Ptable          [13]
;    0528  2              Adapter : Ref $DS_Hpo_Decl (),                                     ! Pointer to adapter Ptable             [13]
;    0529  2              Check_Device : Ref Vector [, Byte],                                ! Pointer for controller device name    [13]
;    0530  2              Support_Row : Ref Support_Entry_Def,                               ! Pointer to support list vector        [13]
;    0531  2              Found_Device : Byte Initial (0),                                   ! flag for search
;    0532  2              PREV_ADAPTER : REF $DS_HPO_DECL (),                                ! Storage for saved copy of adapter ptable - venus      [22]
;    0533  2              PREV_CONTROLLER : REF $DS_HPO_DECL (),                             ! Storage for saved copy of controller ptable - venus   [22]
;    0534  2              PREV_PREV_CONTROLLER : REF $DS_HPO_DECL (),                        ! Storage for saved copy of controller ptable - nautilus [40
;    0535  2              PREV_PREV_ADAPTER : REF $DS_HPO_DECL ();                           ! Storage for saved copy of adapter ptable - nautilus   [40]
;    0536  2              Adapter = Controller = PREV_ADAPTER = PREV_CONTROLLER = PREV_PREV_ADAPTER = PREV_PREV_CONTROLLER =  .Device;
;    0537  2                                                                                 ! Copy Ptable address                   [22] [40]
;    0538  2
;    0539  2          While .Adapter [Hp$A_Link] NeqA 0 Do                                ! Locate link Ptables                      [13]
;    0540  3              Begin                                                           !                                          [13]
;    0541  3              PREV_PREV_CONTROLLER = .PREV_CONTROLLER;                        ! save copy                                [40]
;    0542  3              PREV_CONTROLLER = .CONTROLLER;                                  ! Save copy                                [22]
;    0543  3              Controller = .Adapter;                                          ! Shift links down                         [13]
;    0544  3              PREV_PREV_ADAPTER = .PREV_ADAPTER;                              ! save copy                                [40]
;    0545  3              PREV_ADAPTER =.ADAPTER;                                         ! Save copy                                [22]
```

```
;   0546  3             Adapter = .Adapter [Hp$A_Link]                  !                                                   [13]
;   0547  2             End;                                            !                                                   [13]
;   0548  3       BEGIN                                                 !                                                   [22]
;   0549  3         BIND
;   0550  3             DEV_TYPE = ADAPTER [HP$T_TYPE] : VECTOR [, BYTE];       ! Points to ptable _LINK field                [22]
;   0551  3
;   0552  3     ! The following IF statement checks if the ptable _TYPE field is an SBIA, if so the previously saved adapter    [22]
;   0553  3     ! address is loaded into ADAPTER. This is necessary for a VENUS CPU since HUB is equal to ABUS , and a _LINK    [22]
;   0554  3     ! field of zero would indicate that the present ptable is for an SBIA.                                        [22]
;   0555  3
;   0556  4             IF (.DEV_TYPE [0] EQL 4                         ! Check for length                                    [47]
;   0557  4                AND (.(DEV_TYPE [1]) <0,32> EQL %ASCII 'SBIA'))! Check for _TYPE filed of SBIA
;   0558  4               OR (.DEV_TYPE [0] EQL 3                       ! Check length                                        [47]
;   0559  4                AND (.(DEV_TYPE [1]) <0,24> EQL %ASCII 'XBI'))! Check for _type filed of XBIA
;   0560  3             THEN                                            !                                                   [22]
;   0561  4               BEGIN                                         !                                                   [22]
;   0562  4                     CONTROLLER = .PREV_CONTROLLER;          ! Load 'previous' controller ptable address           [22]
;   0563  4                     ADAPTER = .PREV_ADAPTER;                ! Load 'prevous' adapter ptable address               [22]
;   0564  4               END                                           !                                                   [22]
;   0565  2       END;                                                  !                                                   [22]
;   0566  3       BEGIN                                                 !                                                   [40]
;   0567  3         BIND
;   0568  3             DEV_TYPE = ADAPTER [HP$T_TYPE] : VECTOR [, BYTE];       ! Points to ptable _LINK field                [40]
;   0569  3
;   0570  3     ! The following IF statement checks if the ptable _TYPE field is an NBIA, if so the PREV_PREV saved adapter     [40]
;   0571  3     ! address is loaded into ADAPTER. This is necessary for NAUTILUS and RRR CPU since the _LINK                  [40]
;   0572  3     ! field of zero would indicate that the present ptable is for an NBIA.                                        [40]
;   0573  3
;   0574  3             IF .DEV_TYPE [0] EQL 4                          ! Check for correct ASCII count of 4.                 [40]
;   0575  4             AND (.(DEV_TYPE [1]) <0,32> EQL %ASCII 'NBIA')  ! Check for _TYPE filed of NBIA                       [40]
;   0576  3             THEN                                            !                                                   [40]
;   0577  4               BEGIN                                         !                                                   [40]
;   0578  4                     CONTROLLER = .PREV_PREV_CONTROLLER;     ! Load  controller ptable address                     [40]
;   0579  4                     ADAPTER = .PREV_PREV_ADAPTER;           ! Load adapter ptable address                         [40]
;   0580  4               END                                           !                                                   [40]
;   0581  2       END;                                                  !                                                   [40]
;   0582  2         Check_Device = Controller [Hp$T_Type];              ! Point to device type of controller                  [13]
;   0583  2         Support_Row = Ds$GA_Support_Table;                  ! Point to beginning of table                         [13]
;   0584  2
;   0585  2         Incr I From 0 to Ds$GK_Support_Table_Rows - 1 Do    ! Loop through the table                              [13]
;   0586  3             Begin                                           !                                                   [13]
;   0587  3
;   0588  3             Bind                                            !                                                   [13]
;   0589  3                 Device_Type = .Support_Row [Spt$L_Type] : Vector [, Byte];          !                           [13]
;   0590  3
;   0591  3             If .Device_Type [0] Eql .Check_Device [0]       ! If device type lengths are same                     [13]
;   0592  3              And Ch$Eql (                                   ! .. and names match                                  [13]
;   0593  3                 .Device_Type [0],                           !                                                   [13]
;   0594  3                 Device_Type [1],                            !                                                   [13]
;   0595  3                 .Device_Type [0],                           !                                                   [13]
;   0596  3                 Check_Device [1]                            !                                                   [13]
;   0597  3                 )                                           !                                                   [13]
;   0598  3             Then                                            !                                                   [13]
;   0599  4                 Begin                                       ! Tentative match...verify it                         [13]
;   0600  4
;   0601  4                 Local                                       !                                                   [13]
;   0602  4                     Found_Adapter : Byte Initial (0),       ! Flag for matching adapter                           [13]
```

```
;   0603  4                 Found_Unit : Byte Initial (0);          ! Flag for matching unit                                [13]
;   0604  4
;   0605  4               If .Support_Row [Spt$V_Adapter]           ! If there should be an adapter                         [13]
;   0606  4               Then                                      !                                                       [13]
;   0607  5                   Begin                                 !                                                       [13]
;   0608  5
;   0609  5                   Bind                                  ! Bind to the adapter device type                       [13]
;   0610  5                       Adapter_Type = Adapter [Hp$T_Type] : Vector [, Byte];      !                            [13]
;   0611  5
;   0612  5                   If .Support_Row [Spt$B_Adc] Eql 0     ! If it should be a UNIBUS                              [13]
;   0613  5                    And .Adapter_Type [0] Eql 5          ! .. and string has the right length for 'DW%%%'        [26]
;   0614  5                    And .(Adapter_Type [1])<0, 16> Eql %Asci'DW'  ! .. and it starts out right                   [26]
;   0615  5                   Then                                  !                                                       [13]
;   0616  5                       Found_Adapter = True              ! .. then mark that it's OK                             [13]
;   0617  5                   Else If .Support_Row [Spt$B_Adc] Eql 32 ! .. if it should be MASSbus                          [13]
;   0618  5                    And .Adapter_Type [0] Eql 5          ! .. and it has right length for 'RH7%0'                [13]
;   0619  5                    And .(Adapter_Type [1])<0, 24> Eql %Asci'RH7' ! .. and it starts out right                   [13]
;   0620  5                    And .Adapter_Type [5] Eql %C'0'      ! .. and it ends right                                  [13]
;   0621  5                   Then                                  !                                                       [13]
;   0622  5                       Found_Adapter = True              ! .. then mark it as OK                                 [13]
;   0623  5
;   0624  5                   ELSE If .Support_Row [Spt$B_Adc] Eql 0  ! .. if it should be CIbus                            [18]
;   0625  5                    And .Adapter_Type [0] Eql 5          ! .. and it has right length for 'CI7%0'                [18]
;   0626  5                    And .(Adapter_Type [1])<0, 24> Eql %Asci'CI7' ! .. and it starts out right                   [18]
;   0627  5                    And .Adapter_Type [5] Eql %C'0'      ! .. and it ends right                                  [18]
;   0628  5                   Then                                  !                                                       [18]
;   0629  5                       Found_Adapter = True              ! .. then mark it as OK                                 [18]
;   0630  5
;   0631  5                   ELSE If .Support_Row [Spt$B_Adc] Eql 0  ! .. if it should be CIbus                            [31]
;   0632  5                    And .Adapter_Type [0] Eql 5          ! .. and it has right length for 'CIBCI'                [35]
;   0633  5                    And .(Adapter_Type [1])<0, 32> Eql %Asci'CIBC'       ! .. and it starts out right          [35]
;   0634  5                   Then                                  !                                                       [31]
;   0635  5                       Found_Adapter = True              ! .. then mark it as OK                                 [31]
;   0636  5
;   0637  5                   ELSE If .Support_Row [Spt$B_Adc] Eql 0  ! .. if it should be KDB50                            [34]
;   0638  5                    And .Adapter_Type [0] Eql 5          ! .. and it has right length for 'KDB50'                [34]
;   0639  5                    And .(Adapter_Type [1])<0, 32> Eql %Asci'KDB5'       ! .. and it starts out right          [34]
;   0640  5                   Then                                  !                                                       [34]
;   0641  5                       Found_Adapter = True              ! .. then mark it as OK                                 [34]
;   0642  5
;   0643  5                   ELSE If .Support_Row [Spt$B_Adc] Eql 0  ! .. if it should be KFBTA                            [37]
;   0644  5                    And .Adapter_Type [0] Eql 5          ! .. and it has right length for 'KFBTA'                [37]
;   0645  5                    And .(Adapter_Type [1])<0, 32> Eql %Asci'KFBT'       ! .. and it starts out right          [37]
;   0646  5                   Then                                  !                                                       [37]
;   0647  5                       Found_Adapter = True              ! .. then mark it as OK                                 [37]
;   0648  5
;   0649  5                   ELSE If .Support_Row [Spt$B_Adc] Eql 0  ! .. if it should be DEBNT                            [48]
;   0650  5                    And .Adapter_Type [0] Eql 5          ! .. and it has right length for 'DEBNT'                [48]
;   0651  5                    And .(Adapter_Type [1])<0, 32> Eql %Asci'DEBN'       ! .. and it starts out right          [48]
;   0652  5                   Then                                  !                                                       [48]
;   0653  5                       Found_Adapter = True              ! .. then mark it as OK                                 [48]
;   0654  5                                                                                                        !G:10
;   0655  5                   End                                   !                                                       [13]
;   0656  4               Else                                      !                                                       [13]
;   0657  4                   Found_Adapter = True;                 ! Pretend we found it, if there shouldn't be any        [13]
;   0658  4
;   0659  4               If .Support_Row [Spt$V_Controller]        ! If there should be a controller, check drives         [13]
```

```
;   0660  4                 Then                                    !                                                       [13]
;   0661  5                     Begin                               !                                                       [13]
;   0662  5
;   0663  5                     Local                               !                                                       [13]
;   0664  5                         Index : Initial (0);            ! Index in list of drive names                          [13]
;   0665  5
;   0666  5                     Bind                                ! Bind to vector of names                               [13]
;   0667  5                         Drive_Vector = .Support_Row [Spt$L_Drives] : Vector;        !                           [13]
;   0668  5
;   0669  5                     If Drive_Vector Eql 0               ! If no support list                                    [13]
;   0670  5                     Then                                !                                                       [13]
;   0671  5                         Return DS$_ProgErr;             ! Return with an error                                  [13]
;   0672  5
;   0673  5                     While Not .Found_Unit And .Drive_Vector [.Index] Neq 0 Do       ! Loop through drive list   [13]
;   0674  6                         Begin                           !                                                       [13]
;   0675  6
;   0676  6                         Bind                            ! Map to ASCIC string                                   [13]
;   0677  6                             Unit_Name = .Drive_Vector [.Index] : Vector [, Byte],   ! Map string to check       [13]
;   0678  6                             Device_Name = Device [Hp$T_Type] : Vector [, Byte];     ! Map device Ptable type    [13]
;   0679  6
;   0680  6                         If .Unit_Name [0] Eql .Device_Name [0]      ! If sizes are equal                        [13]
;   0681  6                          And Ch$Eql (                   ! .. and names themselves match OK                      [13]
;   0682  6                             .Unit_Name [0],             !                                                       [13]
;   0683  6                             Unit_Name [1],              !                                                       [13]
;   0684  6                             .Unit_Name [0],             !                                                       [13]
;   0685  6                             Device_Name [1]             !                                                       [13]
;   0686  6                             )                           !                                                       [13]
;   0687  6                         Then                            !                                                       [13]
;   0688  7                             Begin                       !                                                       [13]
;   0689  7                             Found_Unit = True;          ! OK, this drive is supported                           [13]
;   0690  7                             ExitLoop                    ! Don't check the others                                [13]
;   0691  6                             End;                        !                                                       [13]
;   0692  6
;   0693  6                         Index = .Index + 1              ! Increment table index                                 [13]
;   0694  6                         End                             ! End of WHILE loop                                     [13]
;   0695  6
;   0696  5                     End                                 !                                                       [13]
;   0697  4                 Else                                    !                                                       [13]
;   0698  4                     Found_Unit = True;                  ! Mark as OK if shouldn't be a drive                    [13]
;   0699  4
;   0700  4                 Found_Device = (.Found_Adapter And .Found_Unit);     ! Check both search bits                   [13]
;   0701  4
;   0702  4                 If .Found_Device
;   0703  5                     THEN IF ((.SUPPORT_ROW [SPT$B_BTD] NEQ BTD$K_CONSOLE) OR                    ! Not console or    [23]
;   0704  5
;   0705  6                             ((.SUPPORT_ROW [SPT$B_BTD] EQL BTD$K_CONSOLE) AND                   ! (console and      [23]
;   0706  6                              (.DSA$GL_SID [PR$V_SID_TYPE] NEQ PR$_SID_TYP8SS) AND               !  not ZZZ cpu and  [23]
;   0707  6                              (.DSA$GL_SID [PR$V_SID_TYPE] NEQ PR$_SID_TYP8NN) AND               !  not AAA cpu and  [38]
;   0708  6                              (.DSA$GL_SID [PR$V_SID_TYPE] NEQ PR$_SID_TYP8RR) AND               !  not RRR cpu and  [50]
;   0709  5                              (.SUPPORT_ROW [SPT$B_FILE] EQL 3))          OR                     !  RT11)            [23]
;   0710  5
;   0711  6                             ((.SUPPORT_ROW [SPT$B_BTD] EQL BTD$K_CONSOLE) AND                   ! ((console and     [23]
;   0712  7                              ((.DSA$GL_SID [PR$V_SID_TYPE] EQL PR$_SID_TYP8SS) OR               ! ( ZZZ cpu or      [23]
;   0713  7                              (.DSA$GL_SID [PR$V_SID_TYPE] EQL PR$_SID_TYP8NN) OR                !  AAA cpu) and     [38]
;   0714  6                              (.DSA$GL_SID [PR$V_SID_TYPE] EQL PR$_SID_TYP8RR)) AND              !  RRR cpu) and     [50]
;   0715  5                              (.SUPPORT_ROW [SPT$B_FILE] EQL 2)))                                !  ODS)             [23]
;   0716  5
```

```
;   0717  4                         THEN EXITLOOP
;   0718  4
;   0719  3                 End;                                    !                                                   [13]
;   0720  3
;   0721  3             Support_Row = .Support_Row + Spt$K_Row_Size     ! Advance to next row                               [13]
;   0722  2             End;                                        !                                                   [13]
;   0723  2
;   0724  2         If Not .Found_Device Then Return Ds$_Error;         ! Return with error, if not found                   [13]
;   0725  2
;   0726  3                     IF      ((.SUPPORT_ROW [SPT$B_BTD] NEQ BTD$K_CONSOLE) AND           ! (not console and      [43]
;   0727  3                              (.DS$GB_SYSTYPE EQL PR$_SYS_TYP900))                       !  ANRROMEDA)           [43]
;   0728  2                         then Return DS$_ERROR ;                                                         !G:4
;   0729  2                                                                                                         !G:4
;   0730  2         If Not NullParameter (2)                            ! If output params specified, copy values           [13]
;   0731  2         Then                                                !                                                   [13]
;   0732  2             .Adc = .Support_Row [Spt$B_Adc];                ! .. copy adapter code                              [13]
;   0733  2
;   0734  2         If Not NullParameter (3)                            !                                                   [13]
;   0735  2         Then                                                !                                                   [13]
;   0736  2             .Btd = .Support_Row [Spt$B_Btd];                ! .. copy device code                               [13]
;   0737  2
;   0738  2         If Not NullParameter (4)                            !                                                   [13]
;   0739  2         Then                                                !                                                   [13]
;   0740  2             .File = .Support_Row [Spt$B_File];              ! .. copy file code                                 [13]
;   0741  2
;   0742  2         If Not NullParameter (5)                            !                                                   [13]
;   0743  2         Then                                                !                                                   [13]
;   0744  2             .Init = .Support_Row [Spt$L_Init];              ! .. copy init address                              [13]
;   0745  2
;   0746  2         SS$_Normal                                          ! Success                                           [13]
;   0747  1         End;                                                ! End of routine                                    [13]


                                    0FFC 00000          .ENTRY  DSX$CHECK_SUPPORT, Save R2,R3,R4,R5,R6,R7,-  ; 0514
                                                                R8,R9,R10,R11                                ;
                  5E            04  C2 00002          SUBL2   #4, SP                                       ;
                                6E  94 00005          CLRB    FOUND_DEVICE                                 ; 0515
                  52      04    AC  D0 00007          MOVL    DEVICE, PREV_PREV_CONTROLLER                 ; 0536
                  55            52  D0 0000B          MOVL    PREV_PREV_CONTROLLER, PREV_PREV_ADAPTER      ;
                  53            52  D0 0000E          MOVL    PREV_PREV_CONTROLLER, PREV_CONTROLLER        ;
                  54            52  D0 00011          MOVL    PREV_PREV_CONTROLLER, PREV_ADAPTER           ;
                  50            52  D0 00014          MOVL    PREV_PREV_CONTROLLER, CONTROLLER             ;
                  57            52  D0 00017          MOVL    PREV_PREV_CONTROLLER, ADAPTER                ;
                          20    A7  D5 0001A 1$:      TSTL    32(ADAPTER)                                  ; 0539
                                15  13 0001D          BEQL    2$                                           ;
                  52            53  D0 0001F          MOVL    PREV_CONTROLLER, PREV_PREV_CONTROLLER        ; 0541
                  53            50  D0 00022          MOVL    CONTROLLER, PREV_CONTROLLER                  ; 0542
                  50            57  D0 00025          MOVL    ADAPTER, CONTROLLER                          ; 0543
                  55            54  D0 00028          MOVL    PREV_ADAPTER, PREV_PREV_ADAPTER              ; 0544
                  54            57  D0 0002B          MOVL    ADAPTER, PREV_ADAPTER                        ; 0545
                  57      20    A7  D0 0002E          MOVL    32(ADAPTER), ADAPTER                         ; 0546
                                E6  11 00032          BRB     1$                                           ;
                  51      26    A7  9E 00034 2$:      MOVAB   38(ADAPTER), R1                              ; 0550
                  04            61  91 00038          CMPB    (R1), #4                                     ; 0556
```

```
                                                    0A  12 0003B          BNEQ    3$                                                      ;
                                41494253    8F      01  A1  D1 0003D          CMPL    1(R1), #1095320147                                      ; 0557
                                                    11  13 00045          BEQL    4$                                                      ;
                                        03          61  91 00047 3$:      CMPB    (R1), #3                                                ; 0558
                                                    12  12 0004A          BNEQ    5$                                                      ;
00494258    8F      01  A1              18          00  ED 0004C          CMPZV   #0, #24, 1(R1), #4801112                                ; 0559
                                                    06  12 00056          BNEQ    5$                                                      ;
                                        50          53  D0 00058 4$:      MOVL    PREV_CONTROLLER, CONTROLLER                             ; 0562
                                        57          54  D0 0005B          MOVL    PREV_ADAPTER, ADAPTER                                   ; 0563
                                        51      26  A7  9E 0005E 5$:      MOVAB   38(ADAPTER), R1                                         ; 0568
                                        04          61  91 00062          CMPB    (R1), #4                                                ; 0574
                                                    10  12 00065          BNEQ    6$                                                      ;
                                4149424E    8F      01  A1  D1 00067          CMPL    1(R1), #1095320142                                      ; 0575
                                                    06  12 0006F          BNEQ    6$                                                      ;
                                        50          52  D0 00071          MOVL    PREV_PREV_CONTROLLER, CONTROLLER                        ; 0578
                                        57          55  D0 00074          MOVL    PREV_PREV_ADAPTER, ADAPTER                              ; 0579
                                        54      26  A0  9E 00077 6$:      MOVAB   38(R0), CHECK_DEVICE                                    ; 0582
                                        55 00000000'    EF  9E 0007B          MOVAB   DS$GA_SUPPORT_TABLE, SUPPORT_ROW                        ; 0583
                                                    5B  D4 00082          CLRL    I                                                       ; 0591
                                        50          65  D0 00084 7$:      MOVL    (SUPPORT_ROW), R0                                       ; 0589
                                        64          60  91 00087          CMPB    (R0), (CHECK_DEVICE)                                    ; 0591
                                                    0E  12 0008A          BNEQ    8$                                                      ;
                                        52          60  9A 0008C          MOVZBL  (R0), R2                                                ; 0593
                                        51          60  9A 0008F          MOVZBL  (R0), R1                                                ; 0595
            51              00      01  A0          52  2D 00092          CMPC5   R2, 1(R0), #0, R1, 1(CHECK_DEVICE)                      ; 0596
                                                01  A4  00098                                                                            ;
                                                    03  13 0009A 8$:      BEQL    9$                                                      ;
                                                  0142  31 0009C          BRW     26$                                                     ;
                                                    59  94 0009F 9$:      CLRB    FOUND_ADAPTER                                           ; 0599
                                                    5A  94 000A1          CLRB    FOUND_UNIT                                              ;
                                        03      07  A5  E8 000A3          BLBS    7(SUPPORT_ROW), 10$                                     ; 0605
                                                  0099  31 000A7          BRW     17$                                                     ;
                                        50      26  A7  9E 000AA 10$:     MOVAB   38(ADAPTER), R0                                         ; 0610
                                                    51  D4 000AE          CLRL    R1                                                      ; 0612
                                                04  A5  95 000B0          TSTB    4(SUPPORT_ROW)                                          ;
                                                    0F  12 000B3          BNEQ    11$                                                     ;
                                                    51  D6 000B5          INCL    R1                                                      ;
                                        05          60  91 000B7          CMPB    (R0), #5                                                ; 0613
                                                    08  12 000BA          BNEQ    11$                                                     ;
                                5744    8F      01  A0  B1 000BC          CMPW    1(R0), #22340                                           ; 0614
                                                    7F  13 000C2          BEQL    17$                                                     ;
                                        20      04  A5  91 000C4 11$:     CMPB    4(SUPPORT_ROW), #32                                     ; 0617
                                                    17  12 000C8          BNEQ    12$                                                     ;
                                        05          60  91 000CA          CMPB    (R0), #5                                                ; 0618
                                                    12  12 000CD          BNEQ    12$                                                     ;
00374852    8F      01  A0              18          00  ED 000CF          CMPZV   #0, #24, 1(R0), #3622994                                ; 0619
                                                    06  12 000D9          BNEQ    12$                                                     ;
                                        30      05  A0  91 000DB          CMPB    5(R0), #48                                              ; 0620
                                                    62  13 000DF          BEQL    17$                                                     ;
                                        62          51  E9 000E1 12$:     BLBC    R1, 18$                                                 ; 0624
                                        05          60  91 000E4          CMPB    (R0), #5                                                ; 0625
                                                    12  12 000E7          BNEQ    13$                                                     ;
00374943    8F      01  A0              18          00  ED 000E9          CMPZV   #0, #24, 1(R0), #3623235                                ; 0626
                                                    06  12 000F3          BNEQ    13$                                                     ;
                                        30      05  A0  91 000F5          CMPB    5(R0), #48                                              ; 0627
                                                    48  13 000F9          BEQL    17$                                                     ;
                                        48          51  E9 000FB 13$:     BLBC    R1, 18$                                                 ; 0631
```

```
                        05          60  91 000FE        CMPB    (R0), #5                                  ; 0632
                                    0A  12 00101        BNEQ    14$                                       ;
                43424943  8F    01  A0  D1 00103        CMPL    1(R0), #1128417603                        ; 0633
                                    36  13 0010B        BEQL    17$                                       ;
                        36          51  E9 0010D 14$:   BLBC    R1, 18$                                   ; 0637
                        05          60  91 00110        CMPB    (R0), #5                                  ; 0638
                                    0A  12 00113        BNEQ    15$                                       ;
                3542444B  8F    01  A0  D1 00115        CMPL    1(R0), #893535307                         ; 0639
                                    24  13 0011D        BEQL    17$                                       ;
                        24          51  E9 0011F 15$:   BLBC    R1, 18$                                   ; 0643
                        05          60  91 00122        CMPB    (R0), #5                                  ; 0644
                                    0A  12 00125        BNEQ    16$                                       ;
                5442464B  8F    01  A0  D1 00127        CMPL    1(R0), #1413629515                        ; 0645
                                    12  13 0012F        BEQL    17$                                       ;
                        12          51  E9 00131 16$:   BLBC    R1, 18$                                   ; 0649
                        05          60  91 00134        CMPB    (R0), #5                                  ; 0650
                                    0D  12 00137        BNEQ    18$                                       ;
                4E424544  8F    01  A0  D1 00139        CMPL    1(R0), #1312965956                        ; 0651
                                    03  12 00141        BNEQ    18$                                       ;
                        59          01  90 00143 17$:   MOVB    #1, FOUND_ADAPTER                         ; 0657
            3E      07  A5          01  E1 00146 18$:   BBC     #1, 7(SUPPORT_ROW), 22$                   ; 0659
                                    56  D4 0014B        CLRL    INDEX                                     ; 0661
                            0C      A5  D5 0014D        TSTL    12(SUPPORT_ROW)                           ; 0669
                                    08  12 00150        BNEQ    19$                                       ;
                        50 00660020 8F  D0 00152        MOVL    #6684704, R0                              ; 0671
                                        04 00159        RET                                               ;
                        50      04  AC  D0 0015A 19$:   MOVL    DEVICE, R0                                ; 0678
                        58      26  A0  9E 0015E        MOVAB   38(R0), R8                                ;
                        27          5A  E8 00162 20$:   BLBS    FOUND_UNIT, 23$                           ; 0673
                            0C B546     D5 00165        TSTL    @12(SUPPORT_ROW)[INDEX]                   ;
                                    21  13 00169        BEQL    23$                                       ;
                        50  0C B546     D0 0016B        MOVL    @12(SUPPORT_ROW)[INDEX], R0               ; 0677
                        68          60  91 00170        CMPB    (R0), (R8)                                ; 0680
                                    10  12 00173        BNEQ    21$                                       ;
                        52          60  9A 00175        MOVZBL  (R0), R2                                  ; 0682
                        51          60  9A 00178        MOVZBL  (R0), R1                                  ; 0684
    51      00      01  A0          52  2D 0017B        CMPC5   R2, 1(R0), #0, R1, 1(R8)                  ; 0685
                            01      A8     00181                                                          ;
                                    04  13 00183        BEQL    22$                                       ;
                                    56  D6 00185 21$:   INCL    INDEX                                     ; 0693
                                    D9  11 00187        BRB     20$                                       ;
                        5A          01  90 00189 22$:   MOVB    #1, FOUND_UNIT                            ; 0698
                        50          5A  92 0018C 23$:   MCOMB   FOUND_UNIT, R0                            ; 0700
            6E          59          50  8B 0018F        BICB3   R0, FOUND_ADAPTER, FOUND_DEVICE           ;
                        4B          6E  E9 00193        BLBC    FOUND_DEVICE, 26$                         ; 0702
                    40  8F      05  A5  91 00196        CMPB    5(SUPPORT_ROW), #64                       ; 0703
                                    4D  12 0019B        BNEQ    27$                                       ;
                        05 0000FE17 9F  91 0019D        CMPB    @#^X0000FE17, #5                          ; 0706
                                    18  13 001A4        BEQL    24$                                       ;
                        06 0000FE17 9F  91 001A6        CMPB    @#^X0000FE17, #6                          ; 0707
                                    0F  13 001AD        BEQL    24$                                       ;
                        11 0000FE17 9F  91 001AF        CMPB    @#^X0000FE17, #17                         ; 0708
                                    06  13 001B6        BEQL    24$                                       ;
                        03      06  A5  91 001B8        CMPB    6(SUPPORT_ROW), #3                        ; 0709
                                    2C  13 001BC        BEQL    27$                                       ;
                    40  8F      05  A5  91 001BE 24$:   CMPB    5(SUPPORT_ROW), #64                       ; 0711
                                    1C  12 001C3        BNEQ    26$                                       ;
```

```
                                50 0000FE17   9F  9A 001C5          MOVZBL  a#^X0000FE17, R0                                ; 0712
                                05            50  91 001CC          CMPB    R0, #5                                          ;
                                              0A  13 001CF          BEQL    25$                                             ;
                                06            50  91 001D1          CMPB    R0, #6                                          ; 0713
                                              05  13 001D4          BEQL    25$                                             ;
                                11            50  91 001D6          CMPB    R0, #17                                         ; 0714
                                              06  12 001D9          BNEQ    26$                                             ;
                                02       06   A5  91 001DB 25$:     CMPB    6(SUPPORT_ROW), #2                              ; 0715
                                              09  13 001DF          BEQL    27$                                             ;
                                55            10  C0 001E1 26$:     ADDL2   #16, SUPPORT_ROW                                ; 0721
      FE9A            5B        01            23  F1 001E4          ACBL    #35, #1, I, 7$                                  ;
                                16            6E  E9 001EA 27$:     BLBC    FOUND_DEVICE, 28$                               ; 0724
                          40    8F       05   A5  91 001ED          CMPB    5(SUPPORT_ROW), #64                             ; 0726
                                17            13  91 001F2          BEQL    29$                                             ;
00000000G  8F 00000000G   EF    08            00  ED 001F4          CMPZV   #0, #8, DS$GB_SYSTYPE, #PR$_SYS_TYP900          ; 0727
                                              08  12 00201          BNEQ    29$                                             ;
                                50 00660002   8F  D0 00203 28$:     MOVL    #6684674, R0                                    ; 0728
                                              04  00 0020A          RET                                                     ;
                                02            6C  91 0020B 29$:     CMPB    (AP), #2                                        ; 0730
                                              0A  1F 0020E          BLSSU   30$                                             ;
                                         08   AC  D5 00210          TSTL    8(AP)                                           ;
                                              05  13 00213          BEQL    30$                                             ;
                          08    BC       04   A5  9A 00215          MOVZBL  4(SUPPORT_ROW), aADC                            ; 0732
                                03            6C  91 0021A 30$:     CMPB    (AP), #3                                        ; 0734
                                              0A  1F 0021D          BLSSU   31$                                             ;
                                         0C   AC  D5 0021F          TSTL    12(AP)                                          ;
                                              05  13 00222          BEQL    31$                                             ;
                          0C    BC       05   A5  9A 00224          MOVZBL  5(SUPPORT_ROW), aBTD                            ; 0736
                                04            6C  91 00229 31$:     CMPB    (AP), #4                                        ; 0738
                                              0A  1F 0022C          BLSSU   32$                                             ;
                                         10   AC  D5 0022E          TSTL    16(AP)                                          ;
                                              05  13 00231          BEQL    32$                                             ;
                          10    BC       06   A5  9A 00233          MOVZBL  6(SUPPORT_ROW), aFILE                           ; 0740
                                05            6C  91 00238 32$:     CMPB    (AP), #5                                        ; 0742
                                              0A  1F 0023B          BLSSU   33$                                             ;
                                         14   AC  D5 0023D          TSTL    20(AP)                                          ;
                                              05  13 00240          BEQL    33$                                             ;
                          14    BC       08   A5  D0 00242          MOVL    8(SUPPORT_ROW), aINIT                           ; 0744
                                50            01  D0 00247 33$:     MOVL    #1, R0                                          ; 0747
                                              04  00 0024A          RET                                                     ;

; Routine Size:  587 bytes,    Routine Base:  CODE + 008E
```

```
;    0749  1      %sbttl 'load error status'
;    0750  1
;    0751  1      !++
;    0752  1      ! Functional description:
;    0753  1      !       Set status in FAB field STS and clean up static storage for
;    0754  1      !       return.
;    0755  1      !
;    0756  1      ! Inputs:
;    0757  1      !       R0      error code
;    0758  1      ! Implicit inputs:
;    0759  1      !       R11     Pointer to FAB
;    0760  1      !       also some CBLOCK fields if the CBLOCK has been allocated
;    0761  1      ! Outputs:
;    0762  1      !       none
;    0763  1      ! Implicit outputs:
;    0764  1      !       loads error code into FAB [fab$l_sts]
;    0765  1      ! Side effects:
;    0766  1      !       It will deallocate all nonpaged pool space allocated and
;    0767  1      !       clear the RMS$FILE_TABLE entry used to point to it, as well
;    0768  1      !       as the FAB's IFI which is an index into this table.
;    0769  1      ! Return codes: R0 unaffected
;    0770  1      !--
;    0771  1
;    0772  1      global routine rms$error (code) : jsb_rms =
;    0773  2          begin
;    0774  2
;    0775  2          external register
;    0776  2              cblock = 9 : ref bblock,
;    0777  2              fab = 11 : ref bblock;
;    0778  2
;    0779  2          bind
;    0780  2              cache = cblock [ctl$l_cache1] : vector; ! Make it easy to look at
;    0781  2
;    0782  2          fab [fab$l_sts] = .code;                    ! Load the return code
;    0783  2
;    0784  2          if .cblock eql 0 then return .code;         ! Can't go further if no cblock
;    0785  2
;    0786  2          if not .code                                ! If it was an error
;    0787  2              or .cblock [ctl$b_op] eql ctl$c_close   !   or this is a close
;    0788  2          then
;    0789  3              begin                                   ! Deallocate the world!!!!!
;    0790  3
;    0791  3              if .cblock [ctl$b_op] eql ctl$c_open or .cblock [ctl$b_op] eql ctl$c_close
;    0792  3                                                      ! If we should deallocate
;    0793  3              then
;    0794  4                  begin
;    0795  4
;    0796  4                  if .cblock [ctl$l_filhdr] neq 0     ! If file header space was allocated
;    0797  4                  then
;    0798  5                      begin
;    0799  5                      (.cblock [ctl$l_filhdr] + 8)<0, 16> = ctl$c_filhdr_siz; ! Set length of filhdr
;    0800  5                      exe$deanonpaged (.cblock [ctl$l_filhdr])        ! Deallocate it
;    0801  4                      end;
;    0802  4
;    0803  4                  if .cblock [ctl$l_rpb] neq 0        ! If we allocated an RPB
;    0804  4                  then
;    0805  4                      Dsr$Deallocate_Pseudo_RPB (.cblock [ctl$l_rpb]);       ! Deallocate it                [10]
```

```
;   0806  4
;   0807  4                 if .cblock [ctl$w_file_len] neq 0   ! If we allocated a string
;   0808  4                 then
;   0809  5                     begin                           ! De-allocate it
;   0810  5                     (.cblock [ctl$l_file_ptr] + 8)<0, 16> = .cblock [ctl$w_file_alloc];
;   0811  5                                                     ! Load IRP$W_SIZE field
;   0812  5                     exe$deanonpaged (.cblock [ctl$l_file_ptr])      ! De-allocate the string
;   0813  4                     end;
;   0814  4
;   0815  4     !
;   0816  4     ! Now deallocate cache buffers if any were allocated
;   0817  4     !
;   0818  4
;   0819  4                 incr i from 0 to 2 do
;   0820  4
;   0821  4                     if .cache [.i] neq 0            ! If this buffer was allocated
;   0822  4                     then
;   0823  5                         begin
;   0824  5                         (.cache [.i] + 8)<0, 16> = .cblock [ctl$w_cache_size]*512;  ! Set size of buffer
;   0825  5                         exe$deanonpaged (.cache [.i])       ! Deallocate it
;   0826  4                         end;
;   0827  4
;   0828  4                 exe$deanonpaged (.cblock);          ! De-allocate the block
;   0829  4                 rms$file_table [.fab [fab$w_ifi]] = 0;      ! Clear table entry
;   0830  4                 cblock = 0;                         ! Clear cblock pointer
;   0831  4                 fab [fab$w_ifi] = 0                 ! Clear IFI
;   0832  4                 end
;   0833  4
;   0834  2             end;
;   0835  2
;   0836  2         if .cblock neq 0 then cblock [ctl$b_op] = 0;        ! Clear operation code
;   0837  2
;   0838  2         .code                                       ! Return what we came with
;   0839  1         end;                                        ! of rms$error



                                        3E  BB 00000 RMS$ERROR::
                                                          PUSHR   #^M<R1,R2,R3,R4,R5>                     ; 0772
                        55              50  D0 00002      MOVL    R0, R5                                  ;
                    08  AB              55  D0 00005      MOVL    CODE, 8(FAB)                            ; 0782
                                        59  D5 00009      TSTL    CBLOCK                                  ; 0784
                                        7F  13 0000B      BEQL    9$                                      ;
                        05              55  E9 0000D      BLBC    CODE, 1$                                ; 0786
                        04              69  91 00010      CMPB    (CBLOCK), #4                            ; 0787
                                        71  12 00013      BNEQ    8$                                      ;
                        01              69  91 00015 1$:  CMPB    (CBLOCK), #1                            ; 0791
                                        05  13 00018      BEQL    2$                                      ;
                        04              69  91 0001A      CMPB    (CBLOCK), #4                            ;
                                        67  12 0001D      BNEQ    8$                                      ;
                        50        54    A9  D0 0001F 2$:  MOVL    84(CBLOCK), R0                          ; 0796
                                        0C  13 00023      BEQL    3$                                      ;
                    08  A0      0200    8F  B0 00025      MOVW    #512, 8(R0)                             ; 0799
                           00000000G    EF  16 0002B      JSB     EXE$DEANONPAGED                         ; 0800
                                  38    A9  D5 00031 3$:  TSTL    56(CBLOCK)                              ; 0803
```

```
                                        08  13 00034          BEQL    4$                                          ;
                                  38    A9  DD 00036          PUSHL   56(CBLOCK)                                  ; 0805
                     FD64   CF          01  FB 00039          CALLS   #1, DSR$DEALLOCATE_PSEUDO_RPB               ;
                                  48    A9  B5 0003E 4$:      TSTW    72(CBLOCK)                                  ; 0807
                                        0F  13 00041          BEQL    5$                                          ;
                          50      4C    A9  D0 00043          MOVL    76(CBLOCK), R0                              ; 0810
                       08   A0      4A    A9  B0 00047          MOVW    74(CBLOCK), 8(R0)                           ;
                             00000000G   EF  16 0004C          JSB     EXE$DEANONPAGED                             ; 0812
                                        54  D4 00052 5$:      CLRL    I                                           ; 0819
                          50      2C A944   D0 00054 6$:      MOVL    44(CBLOCK)[I], R0                           ; 0821
                                        0E  13 00059          BEQL    7$                                          ;
          08   A0      0A   A9    0200    8F  A5 0005B          MULW3   #512, 10(CBLOCK), 8(R0)                     ; 0824
                             00000000G   EF  16 00063          JSB     EXE$DEANONPAGED                             ; 0825
               E7         54            02  F3 00069 7$:      AOBLEQ  #2, I, 6$                                   ;
                          50            59  D0 0006D          MOVL    CBLOCK, R0                                  ; 0828
                             00000000G   EF  16 00070          JSB     EXE$DEANONPAGED                             ;
                          50      02    AB  3C 00076          MOVZWL  2(FAB), R0                                  ; 0829
                             00000000'EF40   D4 0007A          CLRL    RMS$FILE_TABLE[R0]                          ;
                                        59  D4 00081          CLRL    CBLOCK                                      ; 0830
                                  02    AB  B4 00083          CLRW    2(FAB)                                      ; 0831
                                        59  D5 00086 8$:      TSTL    CBLOCK                                      ; 0836
                                        02  13 00088          BEQL    9$                                          ;
                                        69  94 0008A          CLRB    (CBLOCK)                                    ;
                          50            55  D0 0008C 9$:      MOVL    CODE, R0                                    ; 0839
                                        3E  BA 0008F          POPR    #^M<R1,R2,R3,R4,R5>                         ;
                                            05 00091          RSB                                                 ;

; Routine Size:  146 bytes,    Routine Base:  CODE + 02D9

;   0840  1
```

```
;    0842  1     %sbttl 'load XAB'
;    0843  1
;    0844  1     global routine rms$load_xab : call_rms novalue =
;    0845  1
;    0846  1     !++
;    0847  1     ! Functional description:
;    0848  1     !       set the fields of the FHC XAB.
;    0849  1     !
;    0850  1     ! Inputs:
;    0851  1     !       none
;    0852  1     !
;    0853  1     ! Implicit inputs:
;    0854  1     !       R9              Pointer to the RMS control block
;    0855  1     !       R11             Pointer to the FAB block
;    0856  1     !
;    0857  1     !       all other fields are taken from the FAB or the RMS internal
;    0858  1     !       control block.
;    0859  1     !
;    0860  1     ! Outputs:
;    0861  1     !       none
;    0862  1     !
;    0863  1     ! Implicit Outputs:
;    0864  1     !       If the XAB pointer of the FAB is not zero, the XAB block is
;    0865  1     !       filled in.
;    0866  1     !
;    0867  1     ! Side Effects:
;    0868  1     !       none
;    0869  1     !
;    0870  1     ! Return codes:
;    0871  1     !       none
;    0872  1     !--
;    0873  1
;    0874  2         begin
;    0875  2
;    0876  2         external register
;    0877  2             cblock = 9 : ref bblock,                ! Pointer to control block
;    0878  2             fab = 11 : ref bblock;                  ! Pointer to FAB
;    0879  2
;    0880  2         local
;    0881  2             xab : ref bblock;
;    0882  2
;    0883  2         xab = .fab [fab$l_xab];                     ! Get first XAB address
;    0884  2
;    0885  2         while .xab neq 0                            ! while there are more XABs
;    0886  2             and .xab [xab$b_cod] neq xab$c_fhc do   !  and it's not a FHCXAB
;    0887  2             xab = .xab [xab$l_nxt];                 ! link to next XAB in chain
;    0888  2
;    0889  2         if .xab eql 0                               ! If we ran out of XABs
;    0890  2             or .xab [xab$b_bln] neq xab$c_fhclen    !  or this isn't really an FHC
;    0891  2         then
;    0892  2             return;                                 ! return to caller
;    0893  2
;    0894  2         xab [xab$b_atr] = .fab [fab$b_rat];         ! Copy attributes
;    0895  2         xab [xab$l_ebk] = .cblock [ctl$l_eofvbn];   ! Copy end-of-file VBN
;    0896  2         xab [xab$w_ffb] = .cblock [ctl$w_offbyte];  ! Copy end-of-file byte number
;    0897  2         xab [xab$b_hsz] = .fab [fab$b_fsz];         ! Copy fixed header size
;    0898  2         xab [xab$w_lrl] =                           ! Longest is maximum
```

```
;   0899  2        xab [xab$w_mrz] = .fab [fab$w_mrs];          ! Copy max record length
;   0900  2        (xab [xab$b_rfo])<0, 4> = .fab [fab$b_rfm]; ! Copy record format
;   0901  2        (xab [xab$b_rfo])<4, 4> = fab$c_seq;         ! Always sequential
;   0902  2        xab [xab$l_sbn] = .cblock [ctl$l_startlbn]  ! Copy starting block number
;   0903  1        end;                                         ! of rms$load_xab



                                                 0000 00000          .ENTRY  RMS$LOAD_XAB, Save nothing                  ; 0844
                                 50       24   AB  D0 00002          MOVL    36(FAB), XAB                                ; 0883
                                              0B  13 00006 1$:      BEQL    2$                                          ; 0885
                                 1D           60  91 00008          CMPB    (XAB), #29                                  ; 0886
                                              06  13 0000B          BEQL    2$                                          ;
                                 50       04   A0  D0 0000D          MOVL    4(XAB), XAB                                 ; 0887
                                              F3  11 00011          BRB     1$                                          ;
                                              50  D5 00013 2$:      TSTL    XAB                                         ; 0889
                                              37  13 00015          BEQL    3$                                          ;
                                 2C       01   A0  91 00017          CMPB    1(XAB), #44                                 ; 0890
                                              31  12 0001B          BNEQ    3$                                          ;
                          09     A0       1E   AB  90 0001D          MOVB    30(FAB), 9(XAB)                             ; 0894
                          10     A0       44   A9  D0 00022          MOVL    68(CBLOCK), 16(XAB)                         ; 0895
                          14     A0       42   A9  B0 00027          MOVW    66(CBLOCK), 20(XAB)                         ; 0896
                          17     A0       3F   AB  90 0002C          MOVB    63(FAB), 23(XAB)                            ; 0897
                                 51       36   AB  3C 00031          MOVZWL  54(FAB), R1                                 ; 0899
                          18     A0           51  B0 00035          MOVW    R1, 24(XAB)                                 ;
                          0A     A0           51  B0 00039          MOVW    R1, 10(XAB)                                 ;
     08   A0          04          00       1F   AB  F0 0003D          INSV    31(FAB), #0, #4, 8(XAB)                     ; 0900
                          08     A0       F0   8F  8A 00044          BICB2   #240, 8(XAB)                                ; 0901
                          28     A0       50   A9  D0 00049          MOVL    80(CBLOCK), 40(XAB)                         ; 0902
                                                  04 0004E 3$:      RET                                                 ; 0903

; Routine Size:  79 bytes,    Routine Base:  CODE + 036B


;   0904  1
```

```
;    0906  1     %sbttl 'allocate cache blocks'
;    0907  1
;    0908  1     global routine rms$alloc_cache (bufsiz) : jsb_cblock =
;    0909  1
;    0910  1     !++
;    0911  1     ! Functional description:
;    0912  1     !       Allocate cache blocks.  Since Magtape requires larger cache
;    0913  1     !       buffers than other devices, this routine allows each format-
;    0914  1     !       specific OPEN routine to allocate whatever buffer size is
;    0915  1     !       necessary.
;    0916  1     !
;    0917  1     ! Inputs:
;    0918  1     !       bufsiz          size of cache buffer needed in pages
;    0919  1     !
;    0920  1     ! Implicit inputs:
;    0921  1     !       R9              pointer to control block
;    0922  1     !
;    0923  1     ! Outputs:
;    0924  1     !       none
;    0925  1     !
;    0926  1     ! Implicit outputs:
;    0927  1     !       none
;    0928  1     !
;    0929  1     ! Side effects:
;    0930  1     !       'non-paged' pool space is allocated for each cache buffer (3).
;    0931  1     !       The cache buffer pointers in the control block are set to point
;    0932  1     !       to them, and the CACHE_SIZE word is set to the buffer length
;    0933  1     !       in pages
;    0934  1     !
;    0935  1     ! Return codes
;    0936  1     !       RMS$_NORMAL             if everything was OK
;    0937  1     !       RMS$_DME                if couldn't allocate memory
;    0938  1     !-
;    0939  2         begin
;    0940  2
;    0941  2         external register
;    0942  2             cblock = 9 : ref bblock;                ! Map to control block
;    0943  2
;    0944  2         Local                       !                                       [10]
;    0945  2             Size;                   !                                       [10]
;    0946  2
;    0947  2         cblock [ctl$w_cache_size] = size = .bufsiz; ! Set buffer size to alloc.
;    0948  2         size = .size*512;                           ! Convert to bytes
;    0949  2
;    0950  2         If Not Exe$Alononpaged (.Size;, CBlock [Ctl$L_Cache1])      !       [10]
;    0951  2         Then                        !                                       [10]
;    0952  2             Return Rms$_Dme;        ! Allocate A Buffer                     [10]
;    0953  2
;    0954  2         If Not Exe$Alononpaged (.Size;, CBlock [Ctl$L_Cache2])      !       [10]
;    0955  2         Then                        !                                       [10]
;    0956  2             Return Rms$_Dme;        !                                       [10]
;    0957  2
;    0958  2         If Not Exe$Alononpaged (.Size;, CBlock [Ctl$L_Cache3])      !       [10]
;    0959  2         Then                        !                                       [10]
;    0960  2             Return Rms$_Dme;        !                                       [10]
;    0961  2
;    0962  2         rms$_normal
```

```
;   0963  1          end;                                    ! of rms$alloc_cache



                                        1C  BB 00000 RMS$ALLOC_CACHE::
                                                          PUSHR   #^M<R2,R3,R4>                          ; 0908
                          54            50  D0 00002        MOVL    BUFSIZ, SIZE                           ; 0947
                     0A   A9            50  B0 00005        MOVW    BUFSIZ, 10(CBLOCK)                     ;
                54        54            09  78 00009        ASHL    #9, SIZE, SIZE                         ; 0948
                          51            54  D0 0000D        MOVL    SIZE, R1                               ; 0950
                             00000000G  EF  16 00010        JSB     EXE$ALONONPAGED                        ;
                     2C   A9            52  D0 00016        MOVL    R2, 44(CBLOCK)                         ;
                          20            50  E9 0001A        BLBC    R0, 1$                                 ;
                          51            54  D0 0001D        MOVL    SIZE, R1                               ; 0954
                             00000000G  EF  16 00020        JSB     EXE$ALONONPAGED                        ;
                     30   A9            52  D0 00026        MOVL    R2, 48(CBLOCK)                         ;
                          10            50  E9 0002A        BLBC    R0, 1$                                 ;
                          51            54  D0 0002D        MOVL    SIZE, R1                               ; 0958
                             00000000G  EF  16 00030        JSB     EXE$ALONONPAGED                        ;
                     34   A9            52  D0 00036        MOVL    R2, 52(CBLOCK)                         ;
                          09            50  E8 0003A        BLBS    R0, 2$                                 ;
                          50 000184D4   8F  D0 0003D 1$:    MOVL    #99540, R0                             ; 0960
                                        07  11 00044        BRB     3$                                     ;
                          50 00010001   8F  D0 00046 2$:    MOVL    #65537, R0                             ; 0963
                                        1C  BA 0004D 3$:    POPR    #^M<R2,R3,R4>                          ;
                                            05 0004F        RSB                                            ;

; Routine Size:  80 bytes,    Routine Base:  CODE + 03BA

;   0964  1
```

```
;    0966  1     %sbttl 'initialize static table'
;    0967  1
;    0968  1     !++
;    0969  1     ! Functional description:
;    0970  1     !       Set the file table pointer locations to 0.
;    0971  1     !
;    0972  1     ! Inputs: none
;    0973  1     ! Implicit inputs: none
;    0974  1     ! Outputs: none
;    0975  1     ! Implicit outputs: none
;    0976  1     ! Side effects: none
;    0977  1     ! Status return: none
;    0978  1     !--
;    0979  1
;    0980  1     global routine rms$init : jsb_none novalue =
;    0981  1
;    0982  1         incr i from 1 to 3 do
;    0983  1             rms$file_table [.i] = 0;



                                50            01  D0 00000 RMS$INIT::
                                                            MOVL    #1, I                                   ; 0982
                           00000000'EF40      D4  00003 1$:    CLRL    RMS$FILE_TABLE[I]                       ; 0983
                     F5         50            03  F3 0000A        AOBLEQ  #3, I, 1$                       ;
                                                  05 0000E        RSB                                     ;

; Routine Size:  15 bytes,    Routine Base:  CODE + 040A

;    0984  1
```

```
;    0986  1     %sbttl 'clean up code'
;    0987  1
;    0988  1     global routine rms$cleanup : jsb_none novalue =
;    0989  1
;    0990  1     !++
;    0991  1     ! Functional description:
;    0992  1     !       Called from CLEANUP to de-allocate any RMS static storage from
;    0993  1     !       files not properly closed due to some error.
;    0994  1     !
;    0995  1     ! Inputs : none
;    0996  1     ! Implicit inputs:
;    0997  1     !       rms$file_table  pointers to control blocks
;    0998  1     ! Outputs: None
;    0999  1     ! Implicit outputs: none
;    1000  1     ! Side effects:
;    1001  1     !       deallocates any RMS static storage
;    1002  1     !--
;    1003  1
;    1004  1         incr i from 1 to rms$c_maxfiles do
;    1005  1
;    1006  1             if .rms$file_table [.i] neq 0
;    1007  1             then
;    1008  2                 begin
;    1009  2
;    1010  2                 local
;    1011  2                     x : $fab_decl;                  ! Allocate fake FAB
;    1012  2
;    1013  2                 global register
;    1014  2                     cblock = 9 : ref bblock ,
;    1015  2                     rab = 10,
;    1016  2                     fab = 11 : ref bblock;
;    1017  2
;    1018  2                 cblock = .rms$file_table [.i];      ! Map the control block
;    1019  2                 fab = x;                            ! Map the fake FAB
;    1020  2                 rab = 0;                            ! No RAB
;    1021  2                 fab [fab$w_ifi] = .i;               ! Set the file_table index
;    1022  2                 cblock [ctl$b_op] = ctl$c_close;    ! Force de-allocate
;    1023  2                 rms$error (0)                       ! De-allocate it all
;    1024  1                 end;


                        0E00   8F   BB 00000 RMS$CLEANUP::
                                             PUSHR   #^M<R9,R10,R11>                 ; 0988
                     5E   B0   AE   9E 00004         MOVAB   -80(SP), SP             ;
                     51        01   D0 00008         MOVL    #1, I                   ; 1004
                     50 00000000'EF41   D0 0000B 1$:  MOVL    RMS$FILE_TABLE[I], R0   ; 1006
                               14   13 00013         BEQL    2$                      ;
                     59        50   D0 00015         MOVL    R0, CBLOCK              ; 1018
                     5B        6E   9E 00018         MOVAB   X, FAB                  ; 1019
                               5A   D4 0001B         CLRL    RAB                     ; 1020
                  02 AB        51   B0 0001D         MOVW    I, 2(FAB)               ; 1021
                     69        04   90 00021         MOVB    #4, (CBLOCK)            ; 1022
                               50   D4 00024         CLRL    R0                      ; 1023
                             FE97   30 00026         BSBW    RMS$ERROR               ;
```

```
                DE          51          03  F3 00029 2$:     AOBLEQ  #3, I, 1$                          ;
                            5E      50  AE  9E 0002D         MOVAB   80(SP), SP                         ; 1024
                                  0E00  8F  BA 00031         POPR    #^M<R9,R10,R11>                    ;
                                            05 00035         RSB                                        ;

; Routine Size:  54 bytes,    Routine Base:  CODE + 0419


;   1025  1
```

```
;   1027  1     %sbttl 'load error code in RAB'
;   1028  1     routine rab$error (code) : jsb_rms =
;   1029  1
;   1030  1     !++
;   1031  1     ! Functional description:
;   1032  1     !       Sets error code into RAB
;   1033  1     ! Inputs:
;   1034  1     !       code            the error code
;   1035  1     ! Implicit inputs:
;   1036  1     !       R10             pointer to RAB
;   1037  1     ! Outputs: none
;   1038  1     ! Implicit outputs: none
;   1039  1     ! Side effects: none
;   1040  1     ! Return status:
;   1041  1     !       same as code input
;   1042  1     !--
;   1043  1
;   1044  2         begin
;   1045  2
;   1046  2         external register
;   1047  2             rab = 10 : ref bblock;
;   1048  2
;   1049  3         (rab [rab$l_sts] = .code)
;   1050  1         end;                                    ! of rab$error


                        08   AA          50  D0 00000 RAB$ERROR:
                                                      MOVL    CODE, 8(RAB)                    ; 1049
                                             05 00004 RSB                                     ; 1050

; Routine Size:  5 bytes,    Routine Base:  CODE + 044F
```

```
;   1052  1     %sbttl 'load FAB error code'
;   1053  1     routine fab$error (code) : jsb_rms =
;   1054  1
;   1055  1     !++
;   1056  1     ! Functional description:
;   1057  1     !       Sets error code into FAB
;   1058  1     ! Inputs:
;   1059  1     !       code            the error code
;   1060  1     ! Implicit inputs:
;   1061  1     !       R11             pointer to FAB
;   1062  1     ! Outputs: none
;   1063  1     ! Implicit outputs: none
;   1064  1     ! Side effects: none
;   1065  1     ! Return status:
;   1066  1     !       same as code input
;   1067  1     !--
;   1068  1
;   1069  2         begin
;   1070  2
;   1071  2         external register
;   1072  2             fab = 11 : ref bblock;
;   1073  2
;   1074  3         (fab [fab$l_sts] = .code)
;   1075  1         end;                                    ! of fab$error


                    08   AB          50  D0 00000 FAB$ERROR:
                                                 MOVL    CODE, 8(FAB)                    ; 1074
                                         05 00004 RSB                                    ; 1075

; Routine Size:  5 bytes,    Routine Base:  CODE + 0454
```

```
;    1077  1     %sbttl 'verify validity of FAB'
;    1078  1     routine verify_fab : jsb_rms =
;    1079  1
;    1080  1     !++
;    1081  1     ! Functional description:
;    1082  1     !       Check the ID fields of the FAB to be sure it's really a FAB
;    1083  1     ! Inputs: none
;    1084  1     ! Implicit inputs:
;    1085  1     !       R11             pointer to the FAB to verify
;    1086  1     ! Outputs: none
;    1087  1     ! Implicit outputs: none
;    1088  1     ! Side effects: none
;    1089  1     ! Return codes:
;    1090  1     !       0               not a FAB
;    1091  1     !       1               A good FAB
;    1092  1     !--
;    1093  1
;    1094  2         begin
;    1095  2
;    1096  2         external register
;    1097  2             fab = 11 : ref bblock;
;    1098  2
;    1099  2         if .fab [fab$b_bid] neq fab$c_bid           ! If the ID byte is wrong,
;    1100  2             or .fab [fab$b_bln] neq fab$c_bln       !  or the length is wrong
;    1101  2         then
;    1102  2             return 0;                               ! it's no good
;    1103  2
;    1104  2         1
;    1105  1         end;                                        ! of verify_fab



                                    03          6B  91 00000 VERIFY_FAB:
                                                             CMPB    (FAB), #3                       ; 1099
                                                07  12 00003         BNEQ    1$                      ;
                                50  8F      01  AB  91 00005         CMPB    1(FAB), #80             ; 1100
                                                03  13 0000A         BEQL    2$                      ;
                                                50  D4 0000C 1$:     CLRL    R0                      ; 1102
                                                    05 0000E         RSB                             ;
                                    50          01  D0 0000F 2$:     MOVL    #1, R0                  ; 1105
                                                    05 00012         RSB                             ;

; Routine Size:  19 bytes,    Routine Base:  CODE + 0459
```

```
;    1107  1     %sbttl 'verify cblock'
;    1108  1     routine verify_cblock : jsb_rms =
;    1109  1
;    1110  1     !++
;    1111  1     ! Functional description:
;    1112  1     !       Initialize the global CBLOCK pointer (R9) by using a FAB's
;    1113  1     !       IFI field as index into the RMS$FILE_TABLE array of cblock
;    1114  1     !       pointers.
;    1115  1     ! Inputs: none
;    1116  1     ! Implicit inputs:
;    1117  1     !       R11             pointer to FAB
;    1118  1     ! Outputs: none
;    1119  1     ! Implicit outputs:
;    1120  1     !       R9              pointer to CBLOCK
;    1121  1     ! Side Effects: none
;    1122  1     ! Return codes:
;    1123  1     !       0               IFI or RMS$FILE_TABLE entry is invalid
;    1124  1     !       1               Alright
;    1125  1     !--
;    1126  1
;    1127  2         begin
;    1128  2
;    1129  2         external register
;    1130  2             cblock = 9,                             ! Global pointer to Cblock
;    1131  2             fab = 11 : ref bblock;                  ! Global pointer to FAB
;    1132  2
;    1133  2         local
;    1134  2             ifi;
;    1135  2
;    1136  2         ifi = .fab [fab$w_ifi];                     ! Get the IFI index
;    1137  2
;    1138  2         if .ifi leq 0 or .ifi gtr rms$c_maxfiles    ! If the ifi is bad
;    1139  2             or (cblock = .rms$file_table [.ifi]) eql 0      ! or the entry is bad
;    1140  2         then
;    1141  2             return (cblock = 0);                    ! Clear cblock and return
;    1142  2
;    1143  2         1
;    1144  1         end;                                        ! of verify_cblock



                                50       02   AB  3C 00000 VERIFY_CBLOCK:
                                                            MOVZWL  2(FAB), IFI                     ; 1136
                                              0F  15 00004          BLEQ    1$                      ; 1138
                                03            50  D1 00006          CMPL    IFI, #3                 ;
                                              0A  14 00009          BGTR    1$                      ;
                                59 00000000'EF40  D0 0000B          MOVL    RMS$FILE_TABLE[IFI], CBLOCK     ; 1139
                                              05  12 00013          BNEQ    2$                      ;
                                              59  D4 00015 1$:      CLRL    CBLOCK                  ; 1141
                                              50  D4 00017          CLRL    R0                      ;
                                                  05 00019          RSB                             ;
                                50            01  D0 0001A 2$:      MOVL    #1, R0                  ; 1144
                                                  05 0001D          RSB                             ;

; Routine Size:  30 bytes,    Routine Base:  CODE + 046C
```

```
;   1146  1     %sbttl 'verify validity of RAB'
;   1147  1     routine verify_rab : jsb_rms =
;   1148  1
;   1149  1     !++
;   1150  1     ! Functional description:
;   1151  1     !       Check the ID fields of the RAB to be sure it's really a RAB
;   1152  1     ! Inputs: none
;   1153  1     ! Implicit inputs:
;   1154  1     !       R10             pointer to the RAB to verify
;   1155  1     ! Outputs: none
;   1156  1     ! Implicit outputs: none
;   1157  1     ! Side effects: none
;   1158  1     ! Return codes:
;   1159  1     !       0               not a RAB
;   1160  1     !       1               A good RAB
;   1161  1     !--
;   1162  1
;   1163  2         begin
;   1164  2
;   1165  2         external register
;   1166  2             rab = 10 : ref bblock;
;   1167  2
;   1168  2         if .rab [rab$b_bid] neq rab$c_bid           ! If the ID byte is wrong,
;   1169  2             or .rab [rab$b_bln] neq rab$c_bln       !  or the length is wrong
;   1170  2         then
;   1171  2             return 0;                               ! it's no good
;   1172  2
;   1173  2         1
;   1174  1         end;                                        ! of verify_rab



                        01          6A  91 00000 VERIFY_RAB:
                                                     CMPB    (RAB), #1                       ; 1168
                                    07  12 00003         BNEQ    1$                              ;
                    44  8F      01  AA  91 00005         CMPB    1(RAB), #68                     ; 1169
                                    03  13 0000A         BEQL    2$                              ;
                                    50  D4 0000C 1$:     CLRL    R0                              ; 1171
                                        05 0000E         RSB                                     ;
                        50          01  D0 0000F 2$:     MOVL    #1, R0                          ; 1174
                                        05 00012         RSB                                     ;

; Routine Size:  19 bytes,    Routine Base:  CODE + 048A
```

```
;    1176  1      %sbttl 'RMS$OPEN routine'
;    1177  1
;    1178  1      global routine rms$open (fabptr) =
;    1179  1
;    1180  1      !++
;    1181  1      ! Functional Description:
;    1182  1      !       Search for the file designated by the FAB filename field (with
;    1183  1      !       appropriate defaults added).  If file is found, set fields in
;    1184  1      !       FAB, NAM and XAB blocks to values retrieved from the file header
;    1185  1      !       or other media areas.
;    1186  1      !
;    1187  1      ! Inputs:
;    1188  1      !       fabptr          Address of the FAB block to control access to
;    1189  1      !                       this file.
;    1190  1      !
;    1191  1      ! Implicit inputs:
;    1192  1      !       fab$l_nam       points to the NAM block associated with this FAB.
;    1193  1      !       fab$l_xab       points to the XAB chain associated with this FAB.
;    1194  1      !       several other pointers with the FAB, NAM and XAB blocks which
;    1195  1      !       determine the type of file manipulations, what values are to be
;    1196  1      !       returned, etc.
;    1197  1      !
;    1198  1      ! Outputs:
;    1199  1      !       R0              status code
;    1200  1      !
;    1201  1      ! Implicit Outputs:
;    1202  1      !       many fields in the FAB, NAM and XAB are set to describe the file
;    1203  1      !       just opened.
;    1204  1      !
;    1205  1      ! Side Effects:
;    1206  1      !       Dynamic pool space is allocated and initialized.  This includes
;    1207  1      !       a copy of the file header, and several blocks for file caching.
;    1208  1      !
;    1209  1      ! Return codes:
;    1210  1      !       RMS$_NORMAL                     Everything's cool
;    1211  1      !       RMS$_ACC                        Error accessing file
;    1212  1      !       RMS$_FAB                        Bad FAB
;    1213  1      !       RMS$_DME                        insufficient memory for internal
;    1214  1      !                                       data structures
;    1215  1      !       RMS$_ORG                        Invalid file organization (only
;    1216  1      !                                       Sequential supported)
;    1217  1      !--
;    1218  1
;    1219  2          begin
;    1220  2
;    1221  2          global register
;    1222  2              cblock = 9 : ref bblock ,               ! Pointer to control block
;    1223  2              rab = 10 : ref bblock ,                 ! RMS package globals
;    1224  2              fab = 11 : ref bblock;
;    1225  2
;    1226  2          local
;    1227  2              CBlockSize : Word,                      ! Return from AloNonPaged
;    1228  2              file_index,                             ! Potential IFI
;    1229  2              filedesc : bblock [dsc$c_s_bln*5],      ! Descriptors for filename parts
;    1230  2              filename : bblock [nam$c_maxrss],       ! Final filename buffer
;    1231  2              filenamedesc : bblock [dsc$c_s_bln];    ! Descriptor for filename
;    1232  2
```

```
;    1233  2      !
;    1234  2      ! first, check consistency.  Specifically, be sure that the
;    1235  2      ! 'FAB' passed really IS a FAB block.  Accept no substitutes
;    1236  2      !
;    1237  2          rab = 0;                                    ! No RAB for OPEN service
;    1238  2          cblock = 0;                                 ! No cblock yet
;    1239  2          fab = .fabptr;                              ! Set FAB pointer
;    1240  2
;    1241  2          if not verify_fab () then return (rms$_fab);        ! Invalid FAB
;    1242  2
;    1243  2          if .fab [fab$b_org] neq fab$c_seq then return rms$error (rms$_org); ! DS only supports sequential
;    1244  2
;    1245  2          fab [fab$w_ifi] = 0;                        ! Pre-clear ifi
;    1246  3          file_index = (incr i from 1 to 3 do         ! Find un-used IFI
;    1247  2              if .rms$file_table [.i] eql 0 then exitloop .i);
;    1248  2
;    1249  2          if .file_index eql -1                       ! If no open slot,
;    1250  2          then
;    1251  2              return (fab [fab$l_sts] = rms$_dme);    ! Error
;    1252  2
;    1253  2          If Not Exe$Alononpaged (Ctl$C_Length; CblockSize, CBlock)   !                    [11]
;    1254  2          Then                                        !                                    [10]
;    1255  2              Return (Fab [Fab$L_Sts] = Rms$_Dme);    ! Error If Can'T Allocate           [10]
;    1256  2
;    1257  2          Rms$File_Table [.File_Index] = .CBlock;     ! Set file pointer
;    1258  2          ch$fill (0, ctl$c_length, .cblock);         ! Clear the table
;    1259  2          CBlock [Ctl$W_Size] = .CblockSize;          ! set structure size                [11]
;    1260  2          cblock [ctl$b_op] = ctl$c_open;             ! Set operation code
;    1261  2          fab [fab$w_ifi] = .file_index;              ! Store pointer to it
;    1262  2
;    1263  2      !
;    1264  2      ! Initialize FAB fields
;    1265  2      !
;    1266  2          fab [fab$w_mrs] = fab [fab$b_bks] =         ! Clear a few fields
;    1267  2          fab [fab$l_stv] = fab [fab$l_mrn] = fab [fab$l_sdc] = 0;
;    1268  2          fab [fab$l_sts] = rms$_suc;                 ! Pre-set success code
;    1269  2          ch$fill (0, dsc$c_s_bln*5, filedesc);       ! Initialize filespec parts
;    1270  2      !
;    1271  2      ! Set file name for proper disk structure
;    1272  2      !
;    1273  2           If (.ultrix_flags)<ultrix_v_mode,1>        !                                   [32]
;    1274  2          Then                                        !                                   [32]
;    1275  3           begin                                      !                                   [32]
;    1276  3          breakup_file (def$c_ultrix_len, def$q_ultrix, filedesc);      ! Ultimate system default[32]
;    1277  3          breakup_file (.def$q_dev [dsc$w_length],    ! Load DS default device            [32]
;    1278  3              .def$q_dev [dsc$a_pointer], filedesc);  !                                   [32]
;    1279  3            breakup_file (.def$q_dir [dsc$w_length],  ! Load DS default filespec          [32]
;    1280  3              .def$q_dir [dsc$a_pointer], filedesc);  !                                   [32]
;    1281  3            End                                       !                                   [32]
;    1282  2            Else                                      !                                   [32]
;    1283  3            begin                                     !                                   [32]
;    1284  3            breakup_file (def$c_bkstp_len, def$q_backstop, filedesc);   ! Ultimate system default
;    1285  3            breakup_file (.def$q_dev [dsc$w_length],  ! Load DS default device
;    1286  3              .def$q_dev [dsc$a_pointer], filedesc);
;    1287  3            breakup_file (.def$q_dir [dsc$w_length],  ! Load DS default filespec
;    1288  3              .def$q_dir [dsc$a_pointer], filedesc);
;    1289  3            breakup_file (.fab [fab$b_dns],           ! Load user's default filespec
```

```
;   1290  3             .fab [fab$l_dna], filedesc);
;   1291  2         End;                                    !                                   [32]
;   1292  2         breakup_file (.fab [fab$b_fns],         ! Load filespec
;   1293  2             .fab [fab$l_fna], filedesc);
;   1294  3         begin
;   1295  4         begin
;   1296  4
;   1297  4         local
;   1298  4             pointer,
;   1299  4             length;
;   1300  4
;   1301  4         pointer = filename;                     ! Point to filename buffer
;   1302  4         length = nam$c_maxrss;
;   1303  4         filenamedesc [dsc$a_pointer] = .pointer;    ! And set final descriptor
;   1304  4
;   1305  4         incr i from 0 to 4 do                   ! For each field
;   1306  5             begin
;   1307  5
;   1308  5             bind
;   1309  5                 d = filedesc + .i*dsc$c_s_bln : bblock;     ! Isolate field to load
;   1310  5
;   1311  5             local
;   1312  5                 len,
;   1313  5                 ptr;
;   1314  5
;   1315  5             len = .d [dsc$w_length];
;   1316  5             ptr = .d [dsc$a_pointer];
;   1317  5
;   1318  5             while (len = .len - 1) geq 0 and (length = .length - 1) geq 0 do
;   1319  5                 ch$wchar_a (ch$rchar_a (ptr), pointer)
;   1320  5
;   1321  4             end;
;   1322  4
;   1323  4         filenamedesc [dsc$w_length] = .pointer - filename;  ! Get final length
;   1324  3         end;
;   1325  3     !
;   1326  3     ! Now, copy filename into system pool space, and set up file control
;   1327  3     ! block descriptor to it.
;   1328  3     !
;   1329  3
;   1330  3         If Not Exe$Alononpaged (    !                                                   [10]
;   1331  3             Cblock [Ctl$W_File_Len] = .Filenamedesc [Dsc$W_Length]; !                   [10]
;   1332  3             CBlock [Ctl$W_File_Alloc],      !                                           [10]
;   1333  3             CBlock [Ctl$L_File_Ptr])        !                                           [10]
;   1334  3         Then                        !                                                   [10]
;   1335  3             Return Rms$Error (Rms$_Dme);    ! Fatal If Can'T Allocate                   [10]
;   1336  3
;   1337  3         ch$move (.filenamedesc [dsc$w_length], .filenamedesc [dsc$a_pointer], .cblock [ctl$l_file_ptr]);
;   1338  3                                                 ! Copy the string itself
;   1339  2         end;                                    ! of file merge
;   1340  2     !
;   1341  2     ! Now, access the file and set up static pointers, FAB fields, etc.
;   1342  2     !
;   1343  3         begin
;   1344  3
;   1345  3         local
;   1346  3             devlen,
```

```
;    1347  3              devptr : ref vector [, word],           ! Map to check fst 2 chars
;    1348  3              stat;
;    1349  3
;    1350  3          linkage
;    1351  3              jsb_device = jsb (register = 4, register = 5;   !                                          [07]
;    1352  3                  register = 0, register  = 1);       !                                                  [07]
;    1353  3
;    1354  3          external routine
;    1355  3              scan$device : jsb_device novalue addressing_mode (long_relative);
;    1356  3
;    1357  3          scan$device (.filenamedesc [dsc$w_length],  !                                                  [07]
;    1358  3                  .filenamedesc [dsc$a_pointer];      !                                                  [07]
;    1359  3                  devlen, devptr);    ! Check for device name                                        [07]
;    1360  3          cblock [ctl$w_unit] = .devlen<16, 16, 1>;   ! Store unit number returned
;    1361  3          stat =                      ! Do the device dependent part                                 [07]
;    1362  4              begin                   !                                                              [07]
;    1363  4
;    1364  4              local
;    1365  4                  adapter_epb: ref $ds_hpeo_decl (),   ! Extended ptable for adapter                 !G:20
;    1366  4                  controller_epb: ref $ds_hpeo_decl (),! Extended ptable for controllel              !G:20
;    1367  4                  device_epb: ref $ds_hpeo_decl(),     ! Extended ptable for device                  !G:20
;    1368  4                  adapter_addr_ext: long,              ! temp address of end of EPB                  !G:20
;    1369  4                  controller_addr_ext: long,           ! temp address of end of EPB                  !G:20
;    1370  4                  device_addr_ext: long,               ! temp address of end of EPB                  !G:20
;    1371  4                                                                                                     !G:20
;    1372  4                  AdapterType,        !                                                              [10]
;    1373  4                  DeviceType,         !                                                              [10]
;    1374  4                  DeviceUnit,         !                                                              [10]
;    1375  4                  SlaveNumber,        !                                                              [10]
;    1376  4                  Cinumber,           !                                                              [19]
;    1377  4                  binode,             !                                                              [37]
;    1378  4                  OpenRoutine,        !                                                              [10]
;    1379  4                  FileFormat,         ! Fileformat code                                              [13]
;    1380  4                  InitByte,           ! Address of byte to clear to cause driver init on file open   [28]
;    1381  4                  Status,             ! Status return                                                [13]
;    1382  4                  device : ref $ds_hpo_decl (),       ! Pointer to device PTABLE
;    1383  4                  controller : ref $ds_hpo_decl (),   ! Pointer to controller PTABLE
;    1384  4                  adapter : ref $ds_hpo_decl (),      ! Pointer to adapter PTABLE
;    1385  4                  PREV_CONTROLLER : REF $DS_HPO_DECL (),!Storage for previous controller ptable address  [22]
;    1386  4                  PREV_ADAPTER : REF $DS_HPO_DECL (), ! Storage for previous adapter ptable address     [22]
;    1387  4              PREV_PREV_CONTROLLER : REF $DS_HPO_DECL (),     ! Storage for saved copy of controller ptable - nautilus [40
;    1388  4              PREV_PREV_ADAPTER : REF $DS_HPO_DECL ();        ! Storage for saved copy of adapter ptable - nautilus   [40]
;    1389  4
;    1390  4              SlaveNumber = 0;        !                                                              [10]
;    1391  4              OpenRoutine = 0;        !                                                              [10]
;    1392  4              Cinumber    = 0;        !                                                              [19]
;    1393  4              binode      = 0;        !                                                              [37]
;    1394  4              Loc$Ptable (.DevLen<0, 16>, .DevPtr, -1; Device);       ! Find the ptable             [09]
;    1395  4
;    1396  4              If .Device Eql 0 Then Return Rms$Error (Rms$_Dev);      ! If none, error
;    1397  4              Controller = Adapter = CBlock [Ctl$L_Ptable] = PREV_ADAPTER = PREV_CONTROLLER = PREV_PREV_ADAPTER
;    1398  4                      = PREV_PREV_CONTROLLER =  .Device;! Output R1 = ptable pointer                  [22] [40]
;    1399  4
;    1400  4          While .Adapter [Hp$A_Link] NeqA 0 Do                ! Locate link Ptables
;    1401  5              Begin                                           !                                          [13]
;    1402  5              PREV_PREV_CONTROLLER = .PREV_CONTROLLER;        ! save copy                                [40]
;    1403  5              PREV_CONTROLLER = .CONTROLLER;                  ! Save copy                                [22]
```

```
;  1404  5            Controller = .Adapter;                          ! Shift links down                                  [13]
;  1405  5            PREV_PREV_ADAPTER = .PREV_ADAPTER;              ! save copy                                         [40]
;  1406  5            PREV_ADAPTER =.ADAPTER;                         ! Save copy                                         [22]
;  1407  5            Adapter = .Adapter [Hp$A_Link]                  !                                                   [13]
;  1408  4            End;                                            !                                                   [13]
;  1409  5      BEGIN                                                 !                                                   [22]
;  1410  5        BIND
;  1411  5            DEV_TYPE = ADAPTER [HP$T_TYPE] : VECTOR [, BYTE];       ! Points to ptable _LINK field                [22]
;  1412  5
;  1413  5    ! The following IF statement checks if the ptable _TYPE field is an SBIA, if so the previously saved adapter    [22]
;  1414  5    ! address is loaded into ADAPTER. This is necessary for a VENUS CPU since HUB is equal to ABUS , and a _LINK    [22]
;  1415  5    ! filed of zero would indicate that the present ptable is for an SBIA.                                      [22]
;  1416  5
;  1417  6            IF (.DEV_TYPE [0] EQL 4                         ! Check for length                                  [47]
;  1418  6               AND (.(DEV_TYPE [1]) <0,32> EQL %ASCII 'SBIA'))! Check for _TYPE filed of SBIA
;  1419  6              OR (.DEV_TYPE [0] EQL 3                       ! Check length                                      [47]
;  1420  6               AND (.(DEV_TYPE [1]) <0,24> EQL %ASCII 'XBI'))! Check for _type filed of XBIA
;  1421  5            THEN                                            !                                                   [22]
;  1422  6              BEGIN                                         !                                                   [22]
;  1423  6                    CONTROLLER = .PREV_CONTROLLER;          ! Load 'previous' controller ptable address          [22]
;  1424  6                    ADAPTER = .PREV_ADAPTER;                ! Load 'prevous' adapter ptable address              [22]
;  1425  6              END                                           !                                                   [22]
;  1426  4        END;                                                !                                                   [22]
;  1427  5      BEGIN                                                 !                                                   [40]
;  1428  5        BIND
;  1429  5            DEV_TYPE = ADAPTER [HP$T_TYPE] : VECTOR [, BYTE];       ! Points to ptable _LINK field                [40]
;  1430  5                                                                                                        !G:20
;  1431  5    ! The following IF statement checks if the ptable _TYPE field is an NBIA, if so the PREV_PREV saved adapter    [40]
;  1432  5    ! address is loaded into ADAPTER. This is necessary for NAUTILUS and RRR CPU since the _LINK                  [40]
;  1433  5    ! field of zero would indicate that the present ptable is for an NBIA.                                      [40]
;  1434  5
;  1435  5            IF .DEV_TYPE [0] EQL 4                          ! Check for correct ASCII count of 4.                [40]
;  1436  5            AND .(DEV_TYPE [1]) <0,32> EQL %ASCII 'NBIA'    ! Check for _TYPE filed of NBIA                      [40]
;  1437  5            THEN                                            !                                                   [40]
;  1438  6              BEGIN                                         !                                                   [40]
;  1439  6                    CONTROLLER = .PREV_PREV_CONTROLLER;     ! Load  controller ptable address                    [40]
;  1440  6                    ADAPTER = .PREV_PREV_ADAPTER;           ! Load adapter ptable address                        [40]
;  1441  6              END                                           !                                                   [40]
;  1442  4      END;                                                  !                                                   [40]
;  1443  4            device_addr_ext =  (.device + .device[Hp$w_size]) - 4;                                        !G:20
;  1444  4            Device_Epb      =  ..device_addr_ext;                                                         !G:20
;  1445  4            If .(Device [Hp$T_device] ) <0,32> EQL %ASCII '_DUS'   ! Check if device is a shadow member          [46]
;  1446  4            Then                                            !                                                   [46]
;  1447  4              deviceUnit      =  (.device_epb [Hpe$w_ext_drive]) or 32768                                 !G:20
;  1448  4            Else                                                                                          !G:20
;  1449  4              deviceUnit      =  .device_epb [Hpe$w_ext_drive];                                           !G:20
;  1450  4                                                                                                          !G:19
;  1451  4            Status = DSX$Check_Support (.Device, AdapterType, DeviceType, FileFormat, InitByte);    !      [13]
;  1452  4
;  1453  4            If Not .Status                                  ! If device not supported                      [13]
;  1454  4            Then                                            !                                              [13]
;  1455  5                Begin                                       !                                              [13]
;  1456  5                Fab [Fab$L_Stv] = .Status;                  ! set secondary status                         [13]
;  1457  5                Return RMS$Error (RMS$_Dev)                 ! Device error                                 [13]
;  1458  4                End;                                        !                                              [13]
;  1459  4                                                                                                          !G:4
;  1460  5            If  (.DS$GB_SYSTYPE EQL PR$_SYS_TYP900)         !  if Andromeda cpu                            [43] !G:5
```

```
;  4          Then                                          ! then call the PPA interface to                [43] !G:4
;  5              Begin                                     ! open the file                                 [43] !G:4
;  5              Status = PPA$LOOKUP(FILENAMEDESC);        !                                               [43] !G:4
;  5              Return  .Status                           !                                               [43] !G:4
;  4              End;                                      !                                               [43] !G:4
;  4                                                                                                        !G:4
;  4          If .InitByte NeqA 0                           ! If init byte was specified in table           [13]
;  4          Then                                          !                                               [13]
;  4              (.InitByte)<0, 8> = 0;                    ! ... clear the byte                            [13]
;  4
;  4          Case .FileFormat From 1 To 3 Of               ! Case on file format                           [13]
;  4              Set                                       !                                               [13]
;  4              [RMS$K_Ansi] :                            ! .. ANSI file (magtape)                        [13]
;  5                  Begin                                 !                                               [13]
;  5                    adapter_addr_ext =  (.adapter + .adapter[Hp$w_size]) - 4;                           !G:20
;  5                    Adapter_Epb      =  ..adapter_addr_ext;                                             !G:20
;  5                    binode           =   .adapter_epb [Hpe$w_ext_drive];                                !G:20
;  5                    Cinumber = .Controller [Hp$B_Ci_tem] ;        ! Set CI port number for HSC50                [25]
;  5                                                                                                        !G:20
;  5                    controller_addr_ext =  (.controller + .controller[Hp$w_size]) - 4;                  !G:20
;  5                    Controller_Epb          =  ..controller_addr_ext;                                   !G:20
;  5                    deviceUnit       =  .controller_epb [Hpe$w_ext_drive];                              !G:20
;  5                    binode           =   .controller_epb [Hpe$w_ext_drive];                             !G:20
;  5                                                                                                        !G:20
;  5                                                                                                        !G:20
;  5                    device_addr_ext =  (.device + .device[Hp$w_size]) - 4;                              !G:20
;  5                    Device_Epb       =  ..device_addr_ext;                                              !G:20
;  5                    SlaveNumber      =  .device_epb [Hpe$w_ext_drive];                                  !G:20
;  5                                                                                                        !G:20
;  5                  Fab [Fab$L_Dev] = Device$_Ansi;         ! Set device characteristics                  [13]
;  5                  CBlock [Ctl$V_Ansi] = True;             ! Set ANSI bit                                [13]
;  5                  OpenRoutine = Ansi$Open                 ! Set OPEN routine address                    [13]
;  4                  End;                                    !                                               [13]
;  4              [RMS$K_Ods] :                               ! .. ODS file (normal disk)                   [13]
;  5                  Begin                                   !                                               [13]
;  5                    adapter_addr_ext =  (.adapter + .adapter[Hp$w_size]) - 4;                           !G:20
;  5                    Adapter_Epb      =  ..adapter_addr_ext;                                             !G:20
;  5                    binode           =  .adapter_epb[hpe$w_ext_drive];                                  !G:20
;  5                                                                                                        !G:20
;  5                    Cinumber = .Controller [Hp$B_Ci_tem] ;        ! Set CI port number for HSC50                [20]
;  5                    Fab [Fab$L_Dev] = Device$_Ods;                ! Set device characteristics                  [13]
;  5                    Cblock [Ctl$V_Ods] = True;            ! Set ODS bit                                 [13] !G:20
;  5              If (.ultrix_flags)<ultrix_v_mode,1>         !                                               [32]
;  5              Then                                        !                                               [32] !G:20
;  5                  OpenRoutine = Ultrix$Open               ! Set Open routine address                    [32]
;  5              Else                                        !                                               [32]
;  5                  OpenRoutine = Ods$Open;                 ! Set OPEN routine address                    [13]
;  4                  End;                                    !                                               [13]
;  4              [RMS$K_Rt11] :                              ! .. RT11 console media                       [13]
;  5                  Begin                                   !                                               [13]
;  5                  Fab [Fab$L_Dev] = Device$_Rt11;         ! .. set device characteristics               [13]
;  5                  CBlock [Ctl$V_Rt11] = True;             ! .. set RT-11 bit                            [13]
;  5                  OpenRoutine = Rt11$Open                 ! .. set OPEN routine address                 [13]
;  4                  End;                                    !                                               [13]
;  4              [OutRange] :                                ! Bad support table file code                 [13]
;  5                  Begin                                   !                                               [13]
;  5                  Fab [Fab$L_Stv] = DS$_ProgErr;          ! set secondary return status                 [13]
```

```
;   1518  5                 Return RMS$Error (RMS$_Dev)          ! error code                                  [13]
;   1519  5                 End                                  !                                             [13]
;   1520  4             Tes;                                     !                                             [13]
;   1521  4
;   1522  5         (If .OpenRoutine Neq 0                       ! Open file if good device                    [10]
;   1523  5         Then                                         !                                             [10]
;   1524  6             Begin                                    !                                             [10]
;   1525  6             CBlock [Ctl$L_RPB] = Dsr$Allocate_Pseudo_RPB (      !                                  [10]
;   1526  6                 .DeviceType,                         !                                             [10]
;   1527  6                 .AdapterType,                        !                                             [10]
;   1528  6                 .Controller [HP$A_Device],           !                                             [10]
;   1529  6                 .Adapter [HP$A_Device],              !                                             [10]
;   1530  6                 .DeviceUnit,                         !                                             [10]
;   1531  6                 .SlaveNumber,                        !                                             [10]
;   1532  6                 .Cinumber,                           !
;   1533  6                 .binode);                            !                                             [37]
;   1534  6             (.OpenRoutine) ()                        ! Call the open routine                       [10]
;   1535  6             End                                      !                                             [10]
;   1536  5         Else                                         !                                             [10]
;   1537  5             Rms$_Dev)                                !                                             [10] !G:20
;   1538  3         End;                                         !                                             [07]
;   1539  3     rms$load_xab ();                             ! Load the FHCXAB if any
;   1540  3     rms$error (.stat)                            ! Return status in FAB & R0
;   1541  3     end                                          ! of file access block
;   1542  1     end;                                         ! Of RMS$OPEN


                                                                .EXTRN  SCAN$DEVICE

                                        0FFC 00000              .ENTRY  RMS$OPEN, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1178
                                                                        R11                                          ;
                    5E      FEB4    CE  9E 00002                MOVAB   -332(SP), SP                                 ;
                                    59  7C 00007                CLRQ    CBLOCK                                       ; 1238
                    5B        04    AC  D0 00009                MOVL    FABPTR, FAB                                  ; 1239
                                    AD  10 0000D                BSBB    VERIFY_FAB                                   ; 1241
                    08              50  E8 0000F                BLBS    R0, 1$                                       ;
                    50 0001850C     8F  D0 00012                MOVL    #99596, R0                                   ;
                                        04 00019                RET                                                  ;
                              1D    AB  95 0001A 1$:            TSTB    29(FAB)                                      ; 1243
                                    0A  13 0001D                BEQL    2$                                           ;
                    50 0001860C     8F  D0 0001F                MOVL    #99852, R0                                   ;
                                  0370  31 00026                BRW     35$                                          ;
                              02    AB  B4 00029 2$:            CLRW    2(FAB)                                       ; 1245
                    50              01  D0 0002C                MOVL    #1, I                                        ; 1246
                       00000000'EF40    D5 0002F 3$:            TSTL    RMS$FILE_TABLE[I]                            ; 1247
                                    05  12 00036                BNEQ    4$                                           ;
                    56              50  D0 00038                MOVL    I, FILE_INDEX                                ;
                                    07  11 0003B                BRB     5$                                           ;
          EE        50              03  F3 0003D 4$:            AOBLEQ  #3, I, 3$                                    ;
                    56              01  CE 00041                MNEGL   #1, FILE_INDEX                               ; 1246
             FFFFFFFF 8F            56  D1 00044 5$:            CMPL    FILE_INDEX, #-1                              ; 1249
                                    13  13 0004B                BEQL    6$                                           ;
                    51        5C    8F  9A 0004D                MOVZBL  #92, R1                                      ; 1253
                       00000000G    EF  16 00051                JSB     EXE$ALONONPAGED                              ;
                    57              51  D0 00057                MOVL    R1, CBLOCKSIZE                               ;
                    59              52  D0 0005A                MOVL    R2, CBLOCK                                   ;
```

```
                                10          50  E8 0005D          BLBS    R0, 7$                                  ;
                          08    AB 000184D4   8F  D0 00060 6$:      MOVL    #99540, 8(FAB)                          ; 1255
                                50 000184D4   8F  D0 00068          MOVL    #99540, R0                              ;
                                                  04 0006F          RET                                             ;
                     00000000'EF46            59  D0 00070 7$:      MOVL    CBLOCK, RMS$FILE_TABLE[FILE_INDEX]      ; 1257
  005C   8F          00         6E            00  2C 00078          MOVC5   #0, (SP), #0, #92, (CBLOCK)             ; 1258
                                              69     0007F                                                          ;
                          08    A9            57  B0 00080          MOVW    CBLOCKSIZE, 8(CBLOCK)                   ; 1259
                                69            01  90 00084          MOVB    #1, (CBLOCK)                            ; 1260
                          02    AB            56  B0 00087          MOVW    FILE_INDEX, 2(FAB)                      ; 1261
                                      44      AB  D4 0008B          CLRL    68(FAB)                                 ; 1267
                                      38      AB  D4 0008E          CLRL    56(FAB)                                 ;
                                      0C      AB  D4 00091          CLRL    12(FAB)                                 ;
                                      3E      AB  94 00094          CLRB    62(FAB)                                 ;
                                      36      AB  B4 00097          CLRW    54(FAB)                                 ; 1266
                          08    AB 00010001   8F  D0 0009A          MOVL    #65537, 8(FAB)                          ; 1268
          28          00         6E            00  2C 000A2          MOVC5   #0, (SP), #0, #40, FILEDESC             ; 1269
                                      D8      AD     000A7                                                          ;
                                37 00000000G  EF  E9 000A9          BLBC    ULTRIX_FLAGS, 8$                        ; 1273
                                      D8      AD  9F 000B0          PUSHAB  FILEDESC                                ; 1276
                                   00000000'  EF  9F 000B3          PUSHAB  DEF$Q_ULTRIX                            ;
                                              0F  DD 000B9          PUSHL   #15                                     ;
                        0000G   CF            03  FB 000BB          CALLS   #3, BREAKUP_FILE                        ;
                                      D8      AD  9F 000C0          PUSHAB  FILEDESC                                ; 1277
                                   00000000G  EF  DD 000C3          PUSHL   DEF$Q_DEV+4                             ; 1278
                                7E 00000000G  EF  3C 000C9          MOVZWL  DEF$Q_DEV, -(SP)                        ; 1277
                        0000G   CF            03  FB 000D0          CALLS   #3, BREAKUP_FILE                        ;
                                      D8      AD  9F 000D5          PUSHAB  FILEDESC                                ; 1279
                                   00000000G  EF  DD 000D8          PUSHL   DEF$Q_DIR+4                             ; 1280
                                7E 00000000G  EF  3C 000DE          MOVZWL  DEF$Q_DIR, -(SP)                        ; 1279
                                              44  11 000E5          BRB     9$                                      ;
                                      D8      AD  9F 000E7 8$:      PUSHAB  FILEDESC                                ; 1284
                                   00000000'  EF  9F 000EA          PUSHAB  DEF$Q_BACKSTOP                          ;
                                              11  DD 000F0          PUSHL   #17                                     ;
                        0000G   CF            03  FB 000F2          CALLS   #3, BREAKUP_FILE                        ;
                                      D8      AD  9F 000F7          PUSHAB  FILEDESC                                ; 1285
                                   00000000G  EF  DD 000FA          PUSHL   DEF$Q_DEV+4                             ; 1286
                                7E 00000000G  EF  3C 00100          MOVZWL  DEF$Q_DEV, -(SP)                        ; 1285
                        0000G   CF            03  FB 00107          CALLS   #3, BREAKUP_FILE                        ;
                                      D8      AD  9F 0010C          PUSHAB  FILEDESC                                ; 1287
                                   00000000G  EF  DD 0010F          PUSHL   DEF$Q_DIR+4                             ; 1288
                                7E 00000000G  EF  3C 00115          MOVZWL  DEF$Q_DIR, -(SP)                        ; 1287
                        0000G   CF            03  FB 0011C          CALLS   #3, BREAKUP_FILE                        ;
                                      D8      AD  9F 00121          PUSHAB  FILEDESC                                ; 1289
                                      30      AB  DD 00124          PUSHL   48(FAB)                                 ; 1290
                                7E    35      AB  9A 00127          MOVZBL  53(FAB), -(SP)                          ; 1289
                        0000G   CF            03  FB 0012B 9$:      CALLS   #3, BREAKUP_FILE                        ;
                                      D8      AD  9F 00130          PUSHAB  FILEDESC                                ; 1292
                                      2C      AB  DD 00133          PUSHL   44(FAB)                                 ; 1293
                                7E    34      AB  9A 00136          MOVZBL  52(FAB), -(SP)                          ; 1292
                        0000G   CF            03  FB 0013A          CALLS   #3, BREAKUP_FILE                        ;
                                52    24      AE  9E 0013F          MOVAB   FILENAME, POINTER                       ; 1301
                                54    FF      8F  9A 00143          MOVZBL  #255, LENGTH                            ; 1302
                          20    AE            52  D0 00147          MOVL    POINTER, FILENAMEDESC+4                 ; 1303
                                              51  D4 0014B          CLRL    I                                       ; 1305
                                50    D8 AD41   7E 0014D 10$:      MOVAQ   FILEDESC[I], R0                         ; 1309
                                53            60  3C 00152          MOVZWL  (R0), LEN                               ; 1315
```

```
                        50        04   A0  D0 00155          MOVL    4(R0), PTR                                  ; 1316
                                       53  D7 00159 11$:     DECL    LEN                                         ; 1318
                                       09  19 0015B          BLSS    12$                                         ;
                                       54  D7 0015D          DECL    LENGTH                                      ;
                                       05  19 0015F          BLSS    12$                                         ;
                        82             80  90 00161          MOVB    (PTR)+, (POINTER)+                          ; 1319
                                       F3  11 00164          BRB     11$                                         ;
            E3          51             04  F3 00166 12$:     AOBLEQ  #4, I, 10$                                  ;
                        50        24   AE  9E 0016A          MOVAB   FILENAME, R0                                ; 1323
      1C    AE          52             50  A3 0016E          SUBW3   R0, POINTER, FILENAMEDESC                   ;
                        51        1C   AE  3C 00173          MOVZWL  FILENAMEDESC, R1                            ; 1331
                  48    A9             51  B0 00177          MOVW    R1, 72(CBLOCK)                              ;
                           00000000G   EF  16 0017B          JSB     EXE$ALONONPAGED                             ; 1333
                  4A    A9             51  B0 00181          MOVW    R1, 74(CBLOCK)                              ; 1332
                  4C    A9             52  D0 00185          MOVL    R2, 76(CBLOCK)                              ; 1333
                        0A             50  E8 00189          BLBS    R0, 13$                                     ;
                        50 000184D4    8F  D0 0018C          MOVL    #99540, R0                                  ; 1335
                                     0203  31 00193          BRW     35$                                         ;
      4C    B9    20    BE        1C   AE  28 00196 13$:     MOVC3   FILENAMEDESC, @FILENAMEDESC+4, @76(CBLOCK)  ; 1337
                        55        20   AE  D0 0019D          MOVL    FILENAMEDESC+4, R5                          ; 1357
                        54        1C   AE  3C 001A1          MOVZWL  FILENAMEDESC, R4                            ;
                           00000000G   EF  16 001A5          JSB     SCAN$DEVICE                                 ;
                        53             50  D0 001AB          MOVL    R0, R3                                      ;
            50          53        F0   8F  78 001AE          ASHL    #-16, DEVLEN, R0                            ; 1360
                  02    A9             50  B0 001B3          MOVW    R0, 2(CBLOCK)                               ;
                                       58  D4 001B7          CLRL    OPENROUTINE                                 ; 1391
                                  04   AE  7C 001B9          CLRQ    CINUMBER                                    ; 1392
                                       6E  D4 001BC          CLRL    BINODE                                      ; 1393
                        52             01  CE 001BE          MNEGL   #1, R2                                      ; 1394
                        50             53  3C 001C1          MOVZWL  DEVLEN, R0                                  ;
                           00000000G   EF  16 001C4          JSB     LOC$PTABLE                                  ;
                        53             51  D0 001CA          MOVL    R1, R3                                      ;
                                       03  12 001CD          BNEQ    14$                                         ; 1396
                                     00EF  31 001CF          BRW     27$                                         ;
                        57             53  D0 001D2 14$:     MOVL    DEVICE, PREV_PREV_CONTROLLER                ; 1398
                        56             53  D0 001D5          MOVL    DEVICE, PREV_PREV_ADAPTER                   ;
                        55             53  D0 001D8          MOVL    DEVICE, PREV_CONTROLLER                     ;
                        51             53  D0 001DB          MOVL    DEVICE, PREV_ADAPTER                        ; 1397
                  3C    A9             53  D0 001DE          MOVL    DEVICE, 60(CBLOCK)                          ;
                        52             53  D0 001E2          MOVL    DEVICE, ADAPTER                             ;
                        54             53  D0 001E5          MOVL    DEVICE, CONTROLLER                          ;
                                  20   A2  D5 001E8 15$:     TSTL    32(ADAPTER)                                 ; 1400
                                       15  13 001EB          BEQL    16$                                         ;
                        57             55  D0 001ED          MOVL    PREV_CONTROLLER, PREV_PREV_CONTROLLER       ; 1402
                        55             54  D0 001F0          MOVL    CONTROLLER, PREV_CONTROLLER                 ; 1403
                        54             52  D0 001F3          MOVL    ADAPTER, CONTROLLER                         ; 1404
                        56             51  D0 001F6          MOVL    PREV_ADAPTER, PREV_PREV_ADAPTER             ; 1405
                        51             52  D0 001F9          MOVL    ADAPTER, PREV_ADAPTER                       ; 1406
                        52        20   A2  D0 001FC          MOVL    32(ADAPTER), ADAPTER                        ; 1407
                                       E6  11 00200          BRB     15$                                         ;
                        50        26   A2  9E 00202 16$:     MOVAB   38(ADAPTER), R0                             ; 1411
                        04             60  91 00206          CMPB    (R0), #4                                    ; 1417
                                       0A  12 00209          BNEQ    17$                                         ;
            41494253    8F        01   A0  D1 0020B          CMPL    1(R0), #1095320147                          ; 1418
                                       11  13 00213          BEQL    18$                                         ;
                        03             60  91 00215 17$:     CMPB    (R0), #3                                    ; 1419
                                       12  12 00218          BNEQ    19$                                         ;
```

```
00494258   8F       01   A0            18          00  ED 0021A        CMPZV   #0, #24, 1(R0), #4801112                    ; 1420
                                                    06  12 00224        BNEQ    19$                                         ;
                                        54          55  D0 00226 18$:   MOVL    PREV_CONTROLLER, CONTROLLER                 ; 1423
                                        52          51  D0 00229        MOVL    PREV_ADAPTER, ADAPTER                       ; 1424
                                        51      26  A2  9E 0022C 19$:   MOVAB   38(ADAPTER), R1                            ; 1429
                                        04          61  91 00230        CMPB    (R1), #4                                    ; 1435
                                                    10  12 00233        BNEQ    20$                                         ;
                                4149424E    8F      01  A1  D1 00235        CMPL    1(R1), #1095320142                          ; 1436
                                                    06  12 0023D        BNEQ    20$                                         ;
                                        54          57  D0 0023F        MOVL    PREV_PREV_CONTROLLER, CONTROLLER            ; 1439
                                        52          56  D0 00242        MOVL    PREV_PREV_ADAPTER, ADAPTER                  ; 1440
                                        50      08  A3  3C 00245 20$:   MOVZWL  8(DEVICE), R0                              ; 1443
                                        57      FC A043 9E 00249        MOVAB   -4(R0)[DEVICE], DEVICE_ADDR_EXT             ;
                                        55          67  D0 0024E        MOVL    (DEVICE_ADDR_EXT), DEVICE_EPB               ; 1444
                                5355445F    8F      0C  A3  D1 00251        CMPL    12(DEVICE), #1398096991                     ; 1445
                                                    0B  12 00259        BNEQ    21$                                         ;
                                        56      14  A5  3C 0025B        MOVZWL  20(DEVICE_EPB), DEVICEUNIT                  ; 1447
        56              01          0F              01  F0 0025F        INSV    #1, #15, #1, DEVICEUNIT                     ;
                                                    04  11 00264        BRB     22$                                         ;
                                        56      14  A5  3C 00266 21$:   MOVZWL  20(DEVICE_EPB), DEVICEUNIT                  ; 1449
                                                0C  AE  9F 0026A 22$:   PUSHAB  INITBYTE                                    ; 1451
                                                14  AE  9F 0026D        PUSHAB  FILEFORMAT                                  ;
                                                1C  AE  9F 00270        PUSHAB  DEVICETYPE                                  ;
                                                24  AE  9F 00273        PUSHAB  ADAPTERTYPE                                 ;
                                                    53  DD 00276        PUSHL   DEVICE                                      ;
                            F974    CF              05  FB 00278        CALLS   #5, DSX$CHECK_SUPPORT                       ;
                                        51          50  D0 0027D        MOVL    R0, STATUS                                  ;
                                        06          51  E8 00280        BLBS    STATUS, 23$                                 ; 1453
                              0C    AB              51  D0 00283        MOVL    STATUS, 12(FAB)                             ; 1456
                                                    38  11 00287        BRB     27$                                         ; 1457
00000000G  8F 00000000G  EF             08          00  ED 00289 23$:   CMPZV   #0, #8, DS$GB_SYSTYPE, #PR$_SYS_TYP900    ; 1460
                                                    0E  12 00296        BNEQ    24$                                         ;
                                                1C  AE  9F 00298        PUSHAB  FILENAMEDESC                                ; 1463
                        00000000G   EF              01  FB 0029B        CALLS   #1, PPA$LOOKUP                              ;
                                        51          50  D0 002A2        MOVL    R0, STATUS                                  ;
                                                    04  002A5        RET                                                 ; 1464
                                                0C  AE  D5 002A6 24$:   TSTL    INITBYTE                                    ; 1467
                                                    03  13 002A9        BEQL    25$                                         ;
                                                0C  BE  94 002AB        CLRB    @INITBYTE                                   ; 1469
                    02              01          10  AE  CF 002AE 25$:   CASEL   FILEFORMAT, #1, #2                          ; 1471
                009D            0065            0018        002B3 26$:   .WORD   28$-26$,-                                   ;
                                                                                29$-26$,-                                   ;
                                                                                31$-26$                                     ;
                              0C    AB 00660020   8F  D0 002B9        MOVL    #6684704, 12(FAB)                           ; 1517
                                    50 000184C4   8F  D0 002C1 27$:   MOVL    #99524, R0                                  ; 1518
                                              00CE  31  002C8        BRW     35$                                         ;
                                        50      08  A2  3C 002CB 28$:   MOVZWL  8(ADAPTER), R0                             ; 1475
                                        50      FC A042 9E 002CF        MOVAB   -4(R0)[ADAPTER], ADAPTER_ADDR_EXT           ;
                                        50          60  D0 002D4        MOVL    (ADAPTER_ADDR_EXT), ADAPTER_EPB             ; 1476
                                        6E      14  A0  3C 002D7        MOVZWL  20(ADAPTER_EPB), BINODE                     ; 1477
                                04  AE      34  A4  9A 002DB        MOVZBL  52(CONTROLLER), CINUMBER                    ; 1478
                                        50      08  A4  3C 002E0        MOVZWL  8(CONTROLLER), R0                           ; 1480
                                        50      FC A044 9E 002E4        MOVAB   -4(R0)[CONTROLLER], CONTROLLER_ADDR_EXT     ;
                                        50          60  D0 002E9        MOVL    (CONTROLLER_ADDR_EXT), CONTROLLER_EPB       ; 1481
                                        56      14  A0  3C 002EC        MOVZWL  20(CONTROLLER_EPB), DEVICEUNIT              ; 1482
                                        6E      14  A0  3C 002F0        MOVZWL  20(CONTROLLER_EPB), BINODE                  ; 1483
                                        50      08  A3  3C 002F4        MOVZWL  8(DEVICE), R0                               ; 1486
```

```
                        57        FC A043  9E 002F8          MOVAB   -4(R0)[DEVICE], DEVICE_ADDR_EXT                  ;
                        55              67  D0 002FD          MOVL    (DEVICE_ADDR_EXT), DEVICE_EPB                   ; 1487
                    08  AE        14   A5  3C 00300          MOVZWL  20(DEVICE_EPB), SLAVENUMBER                     ; 1488
                    40  AB      4020   8F  3C 00305          MOVZWL  #16416, 64(FAB)                                 ; 1490
                    01  A9              08  88 0030B          BISB2   #8, 1(CBLOCK)                                   ; 1491
                        58 00000000G    EF  9E 0030F          MOVAB   ANSI$OPEN, OPENROUTINE                          ; 1492
                                        49  11 00316          BRB     32$                                             ;
                        50        08    A2  3C 00318 29$:     MOVZWL  8(ADAPTER), R0                                  ; 1496
                        50        FC A042  9E 0031C          MOVAB   -4(R0)[ADAPTER], ADAPTER_ADDR_EXT               ;
                        50              60  D0 00321          MOVL    (ADAPTER_ADDR_EXT), ADAPTER_EPB                 ; 1497
                        6E        14    A0  3C 00324          MOVZWL  20(ADAPTER_EPB), BINODE                         ; 1498
                    04  AE        34    A4  9A 00328          MOVZBL  52(CONTROLLER), CINUMBER                        ; 1500
                    40  AB  10004008    8F  D0 0032D          MOVL    #268451848, 64(FAB)                             ; 1501
                    01  A9              02  88 00335          BISB2   #2, 1(CBLOCK)                                   ; 1502
                        09 00000000G    EF  E9 00339          BLBC    ULTRIX_FLAGS, 30$                               ; 1503
                        58 00000000G    EF  9E 00340          MOVAB   ULTRIX$OPEN, OPENROUTINE                        ; 1505
                                        18  11 00347          BRB     32$                                             ;
                        58      0000G   CF  9E 00349 30$:     MOVAB   ODS$OPEN, OPENROUTINE                           ; 1507
                                        11  11 0034E          BRB     32$                                             ; 1503
                    40  AB  10004018    8F  D0 00350 31$:     MOVL    #268451864, 64(FAB)                             ; 1511
                    01  A9              01  88 00358          BISB2   #1, 1(CBLOCK)                                   ; 1512
                        58      0000G   CF  9E 0035C          MOVAB   RT11$OPEN, OPENROUTINE                          ; 1513
                                        27  13 00361 32$:     BEQL    33$                                             ; 1522
                                        6E  DD 00363          PUSHL   BINODE                                          ; 1533
                                    08  AE  DD 00365          PUSHL   CINUMBER                                        ; 1532
                                    10  AE  DD 00368          PUSHL   SLAVENUMBER                                     ; 1531
                                        56  DD 0036B          PUSHL   DEVICEUNIT                                      ; 1530
                                    18  A2  DD 0036D          PUSHL   24(ADAPTER)                                     ; 1529
                                    18  A4  DD 00370          PUSHL   24(CONTROLLER)                                  ; 1528
                                    30  AE  DD 00373          PUSHL   ADAPTERTYPE                                     ; 1527
                                    30  AE  DD 00376          PUSHL   DEVICETYPE                                      ; 1526
                  F7E5  CF                  08  FB 00379          CALLS   #8, DSR$ALLOCATE_PSEUDO_RPB                     ;
                    38  A9                  50  D0 0037E          MOVL    R0, 56(CBLOCK)                                  ;
                        68                  00  FB 00382          CALLS   #0, (OPENROUTINE)                               ; 1534
                        52                  50  D0 00385          MOVL    R0, STAT                                        ;
                                            07  11 00388          BRB     34$                                             ;
                        52 000184C4         8F  D0 0038A 33$:     MOVL    #99524, STAT                                    ; 1522
                  FB38  CF                  00  FB 00391 34$:     CALLS   #0, RMS$LOAD_XAB                                ; 1539
                        50                  52  D0 00396          MOVL    STAT, R0                                        ; 1540
                                        FAA0  30  00399 35$:     BSBW    RMS$ERROR                                       ;
                                              04  0039C          RET                                                     ; 1542

; Routine Size:  925 bytes,    Routine Base:  CODE + 049D

;   1543  1
;   1544  1                                                                                             !G:20
;   1545  1                                                                                             !G:20
```

```
;    1547  1      %sbttl 'RMS$CONNECT routine'
;    1548  1
;    1549  1      global routine rms$connect (rabptr) =
;    1550  1
;    1551  1      !++
;    1552  1      ! Functional description:
;    1553  1      !       Connect a RAB to a FAB.  This is relatively trivial; the validity
;    1554  1      !       of both RAB and FAB are verified, checks are made to be sure the
;    1555  1      !       file is open and no other RAB has been connected previously.
;    1556  1      !
;    1557  1      ! Inputs:
;    1558  1      !       rabptr          pointer to the RAB to connect
;    1559  1      !
;    1560  1      ! Implicit inputs:
;    1561  1      !       RAB [rab$l_fab] points to the FAB to connect this RAB to
;    1562  1      !       FAB [fab$w_ifi] indexes the rms$file_table to find the CBLOCK
;    1563  1      !
;    1564  1      ! Outputs:
;    1565  1      !       none
;    1566  1      !
;    1567  1      ! Implicit outputs:
;    1568  1      !       RAB [rab$l_sts] return status
;    1569  1      !       RAB [rab$l_stv] secondary status
;    1570  1      !
;    1571  1      ! Side Effects:
;    1572  1      !       some fields in the RAB and CBLOCK are modified
;    1573  1      !
;    1574  1      ! Return codes:
;    1575  1      !       RMS$_NORMAL     all OK
;    1576  1      !       RMS$_CCR        a RAB is already connected to this FAB
;    1577  1      !       RMS$_RAB        the RAB block is invalid
;    1578  1      !       RMS$_FAB        the FAB block is invalid
;    1579  1      !       RMS$_IFI        the FAB's IFI field is invalid.
;    1580  1      !       RMS$_RAC        Invalid record access mode--only SEQuential and
;    1581  1      !                       RFA access modes are supported.
;    1582  1      !--
;    1583  1
;    1584  2          begin
;    1585  2
;    1586  2          global register
;    1587  2              rab = 10 : ref bblock ,                 ! Pointer to RAB
;    1588  2              fab = 11 : ref bblock ,                 ! Pointer to FAB
;    1589  2              cblock = 9 : ref bblock;                ! Pointer to CBLOCK
;    1590  2
;    1591  2          cblock = 0;
;    1592  2          fab = 0;
;    1593  2          rab = .rabptr;                              ! Map to RAB
;    1594  2
;    1595  2          if not verify_rab ()                        ! If the RAB is bad
;    1596  2          then
;    1597  2              return (rms$_rab);                      !  return error
;    1598  2
;    1599  2          rab [rab$l_stv] = 0;
;    1600  2
;    1601  2          if .rab [rab$b_rac] neq rab$c_seq           ! If not sequential access
;    1602  2              and .rab [rab$b_rac] neq rab$c_rfa      !  and not RFA random
;    1603  2          then
```

```
;    1604  2          return rab$error (rms$_rac);          ! then bad access mode
;    1605  2
;    1606  2      fab = .rab [rab$l_fab];                   ! Map to FAB
;    1607  2
;    1608  2      if not verify_fab ()                      ! If the FAB is bad
;    1609  2      then
;    1610  2          return rab$error (rms$_fab);          !  return error
;    1611  2
;    1612  2      if not verify_cblock ()                   ! If the IFI is bad
;    1613  2      then
;    1614  2          return rab$error (rms$_ifi);          !  return with error
;    1615  2
;    1616  2      if .cblock [ctl$l_rab] neq 0              ! If there's already a RAB
;    1617  2      then
;    1618  2          return rab$error (rms$_ccr);          !  it's no good--only one allowed
;    1619  2
;    1620  2      cblock [ctl$l_rab] = .rab;                ! Now set point to RAB
;    1621  2      cblock [ctl$l_vbn] = 1;                   ! Set internal RFA
;    1622  2      cblock [ctl$w_byte] = 0;
;    1623  2      cblock [ctl$w_rec_size] = 0;              ! Clear last-record size
;    1624  2      ch$fill (0, 6, rab [rab$w_rfa]);          ! Set RFA to 0
;    1625  2      rab [rab$l_rbf] = 0;
;    1626  2      rab [rab$w_rsz] = 0;
;    1627  2      cblock [ctl$l_nbp] = 0;
;    1628  2      rab$error (rms$_normal)                   ! Set good return
;    1629  1      end;


                     0FFC 00000          .ENTRY  RMS$CONNECT, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 1549
                                                 R10,R11                                     ;
                       59  D4 00002              CLRL    CBLOCK                              ; 1591
                       5B  D4 00004              CLRL    FAB                                 ; 1592
5A          04         AC  D0 00006              MOVL    RABPTR, RAB                         ; 1593
                     FC43  30 00009              BSBW    VERIFY_RAB                          ; 1595
08                     50  E8 0000C              BLBS    R0, 1$                              ;
50  0001863C           8F  D0 00010              MOVL    #99900, R0                          ; 1597
                           04 00016              RET                                         ;
            0C         AA  D4 00017  1$:         CLRL    12(RAB)                             ; 1599
            1E         AA  95 0001A              TSTB    30(RAB)                             ; 1601
                       0F  13 0001D              BEQL    2$                                  ;
02          1E         AA  91 00020              CMPB    30(RAB), #2                         ; 1602
                       09  13 00023              BEQL    2$                                  ;
50  00018644           8F  D0 00025              MOVL    #99908, R0                          ; 1604
                       4F  11 00028              BRB     6$                                  ;
5B          3C         AA  D0 0002A  2$:         MOVL    60(RAB), FAB                        ; 1606
                     FBE9  30 0002E              BSBW    VERIFY_FAB                          ; 1608
09                     50  E8 00031              BLBS    R0, 3$                              ;
50  0001850C           8F  D0 00034              MOVL    #99596, R0                          ; 1610
                       3C  11 00037              BRB     6$                                  ;
                     FBED  30 00039  3$:         BSBW    VERIFY_CBLOCK                       ; 1612
09                     50  E8 0003C              BLBS    R0, 4$                              ;
50  00018564           8F  D0 0003F              MOVL    #99684, R0                          ; 1614
                       2D  11 00042              BRB     6$                                  ;
            20         A9  D5 00044  4$:         TSTL    32(CBLOCK)                          ; 1616
```

```
                                        09  13 00054          BEQL    5$                                          ;
                          50 00018494   8F  D0 00056          MOVL    #99476, R0                                  ; 1618
                                        1F  11 0005D          BRB     6$                                          ;
                     20   A9            5A  D0 0005F 5$:      MOVL    RAB, 32(CBLOCK)                             ; 1620
                     18   A9            01  7D 00063          MOVQ    #1, 24(CBLOCK)                              ; 1621
     06          00      6E            00  2C 00067          MOVC5   #0, (SP), #0, #6, 16(RAB)                   ; 1624
                                  10    AA     0006C                                                              ;
                                  28    AA  D4 0006E          CLRL    40(RAB)                                     ; 1625
                                  22    AA  B4 00071          CLRW    34(RAB)                                     ; 1626
                                  04    A9  D4 00074          CLRL    4(CBLOCK)                                   ; 1627
                          50 00010001   8F  D0 00077          MOVL    #65537, R0                                  ; 1628
                                      FB94  30 0007E 6$:      BSBW    RAB$ERROR                                   ;
                                            04 00081          RET                                                 ; 1629

; Routine Size:  130 bytes,    Routine Base:  CODE + 083A


;   1630  1
```

```
;    1632  1      %sbttl 'RMS routine'
;    1633  1
;    1634  1      global routine rms$get (rabptr) =
;    1635  1
;    1636  1      !++
;    1637  1      ! Functional description:
;    1638  1      !       Read the next record into a user-specified buffer, from the
;    1639  1      !       file currently open on the FAB to which the input RAB is
;    1640  1      !       connected.
;    1641  1      !
;    1642  1      ! Inputs:
;    1643  1      !       rabptr                  pointer to the RAB to use
;    1644  1      !
;    1645  1      ! Implicit inputs:
;    1646  1      !       RAB [rab$l_fab]         The FAB which controls the file
;    1647  1      !       RAB [rab$w_rfa]         Desired record's file address
;    1648  1      !       FAB [fab$w_ifi]         Index in RMS$FILE_TABLE for this file
;    1649  1      !       CBLOCK                  Internal control block
;    1650  1      !
;    1651  1      ! Outputs:
;    1652  1      !       none
;    1653  1      !
;    1654  1      ! Implicit outputs:
;    1655  1      !       The RBF and RSZ fields of the RAB are set to describe the
;    1656  1      !       record read.
;    1657  1      !
;    1658  1      !       RAB [rab$l_sts]         contains the status code
;    1659  1      !       RAB [rab$l_stv]         may contain a secondary status
;    1660  1      !       RAB [rab$w_rfa]         in sequential access mode ONLY, this
;    1661  1      !                               3-word field is updated prior to the actual
;    1662  1      !                               access.  Not updated for Andromeda.                    !G:4
;    1663  1      !
;    1664  1      ! Side effects:
;    1665  1      !       CBLOCK                  The next RFA, the cache control data, and
;    1666  1      !                               the tape position control are all subject
;    1667  1      !                               to change.
;    1668  1      !
;    1669  1      ! Return codes:
;    1670  1      !       RMS$_NORMAL             Everything's nice
;    1671  1      !       RMS$_RER                Some read error (RAB [rab$l_stv] contains
;    1672  1      !                               the actual SS$_* error code)
;    1673  1      !       RMS$_RFA                Invalid RFA was specified in random-by-RFA
;    1674  1      !                               mode.
;    1675  1      !       RMS$_EOF                Attempt to read record past end of file
;    1676  1      !       RMS$_RAB                bad RAB
;    1677  1      !       RMS$_FAB                bad FAB
;    1678  1      !       RMS$_IFI                bad IFI (or CBLOCK)
;    1679  1      !       RMS$_ISI                FAB and RAB not connected
;    1680  1      !       RMS$_RTB                Record too big for buffer (truncated)
;    1681  1      !--
;    1682  1
;    1683  2          begin
;    1684  2
;    1685  2          global register
;    1686  2              cblock = 9 : ref bblock ,           ! Map to CBLOCK
;    1687  2              rab = 10 : ref bblock ,             ! Map to RAB
;    1688  2              fab = 11 : ref bblock;              ! Map to FAB
```

```
;  1689  2
;  1690  2          local
;  1691  2              stat;
;  1692  2
;  1693  2          fab = cblock = 0;
;  1694  2          rab = .rabptr;                          ! Get pointer to RAB
;  1695  2
;  1696  2          if not verify_rab ()                    ! If it isn't a RAB
;  1697  2          then
;  1698  2              return (rms$_rab);                  !  no good at all
;  1699  2
;  1700  2          rab [rab$l_stv] = 0;                    ! Clear out STV field
;  1701  2          fab = .rab [rab$l_fab];                 ! Point to RAB's FAB
;  1702  2
;  1703  2          if not verify_fab ()                    ! If it isn't a FAB
;  1704  2          then
;  1705  2              return rab$error (rms$_fab);        !  then quit nicely
;  1706  2
;  1707  2          if not verify_cblock ()                 ! Load CBLOCK pointer,
;  1708  2          then
;  1709  2              return rab$error (rms$_ifi);        !  and quit if IFI's bad
;  1710  2
;  1711  2          if .cblock [ctl$l_rab] neq .rab         ! If this isn't the right RAB
;  1712  2          then
;  1713  2              return rab$error (rms$_isi);        !  say 'bad stream id'
;  1714  2
;  1715  2      !
;  1716  2      ! Now, all the control stuff is OK.  We can go to work
;  1717  2      !
;  1718  2                                                                                                  !G:4
;  1719  3         If  (.DS$GB_SYSTYPE EQL PR$_SYS_TYP900)          !  if Andromeda cpu                  [43]  !G:5
;  1720  2         then                                                                                      !G:4
;  1721  3              Begin                                                                                !G:4
;  1722  3              stat = PPA$READ                          ! Read block (into UBF) using interface [43]  !G:4
;  1723  3                   ( RAB[RAB$L_UBF], RAB[RAB$W_RSZ] );  ! and return record size read (RSZ)    [43]  !G:4
;  1724  3              RAB [rab$l_sts] = .stat ;                ! Fill the status in the RAB           [43]  !G:4
;  1725  3              end                                                                                  !G:4
;  1726  2          else                                                                                      !G:4
;  1727  2                                                                                                  !G:19
;  1728  3              Begin                                                                                !G:4
;  1729  3              cblock [ctl$b_op] = ctl$c_get;       ! We're on GET operation                        !G:4
;  1730  3                                                                                                  !G:4
;  1731  3              if .rab [rab$b_rac] eql rab$c_rfa        ! If random, copy user's                    !G:4
;  1732  3              then                                                                                 !G:4
;  1733  3                  ch$move (6, rab [rab$w_rfa], cblock [ctl$w_rfa]);   ! RFA                        !G:4
;  1734  3                                                                                                  !G:4
;  1735  3              rab [rab$w_rsz] = rab [rab$l_rbf] = 0;  ! Init context                              [05] !G:4
;  1736  3                                                                                                  !G:4
;  1737  3              if .cblock [ctl$l_vbn] gtr .cblock [ctl$l_eofvbn]       ! If RFA is past EOF         !G:4
;  1738  3                  or .cblock [ctl$l_vbn] eql .cblock [ctl$l_eofvbn]   !                           !G:4
;  1739  3                  and .cblock [ctl$w_byte] geq .cblock [ctl$w_offbyte]                            !G:4
;  1740  3              then                                                                                 !G:4
;  1741  3                  return rab$error (                                                               !G:4
;  1742  3                                                                                                  !G:4
;  1743  3                          if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof);     !  don't bother !G:4
;  1744  3                                                                                                  !G:4
;  1745  3              stat = (.cblock [ctl$l_get]) ();         ! Call actual GET routine                   !G:4
```

```
;   1746  3          ch$move (6, cblock [ctl$w_rfa], rab [rab$w_rfa])        ! Set user RFA                          !G:4
;   1747  2          END ;                                                                                            !G:4
;   1748  2                                                                                                           !G:4
;   1749  2      rab$error (.stat)                                  ! Return whatever status
;   1750  1      end;                                               ! Of RMS$GET



                                                  0FFC 00000          .ENTRY  RMS$GET, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-  ; 1634
                                                                              R11                                          ;
                                                 59  D4 00002          CLRL    CBLOCK                                       ; 1693
                                                 5B  D4 00004          CLRL    FAB                                          ;
                                 5A       04    AC  D0 00006          MOVL    RABPTR, RAB                                  ; 1694
                                             FBC1  30 0000A          BSBW    VERIFY_RAB                                   ; 1696
                                 08             50  E8 0000D          BLBS    R0, 1$                                       ;
                                 50 0001863C    8F  D0 00010          MOVL    #99900, R0                                   ; 1698
                                                    04 00017          RET                                                  ;
                                          0C    AA  D4 00018 1$:      CLRL    12(RAB)                                      ; 1700
                                 5B       3C    AA  D0 0001B          MOVL    60(RAB), FAB                                 ; 1701
                                             FB7B  30 0001F          BSBW    VERIFY_FAB                                   ; 1703
                                 09             50  E8 00022          BLBS    R0, 2$                                       ;
                                 50 0001850C    8F  D0 00025          MOVL    #99596, R0                                   ; 1705
                                                 7E  11 0002C          BRB     9$                                           ;
                                             FB7F  30 0002E 2$:      BSBW    VERIFY_CBLOCK                                ; 1707
                                 09             50  E8 00031          BLBS    R0, 3$                                       ;
                                 50 00018564    8F  D0 00034          MOVL    #99684, R0                                   ; 1709
                                                 6F  11 0003B          BRB     9$                                           ;
                                 5A       20    A9  D1 0003D 3$:      CMPL    32(CBLOCK), RAB                              ; 1711
                                                 09  13 00041          BEQL    4$                                           ;
                                 50 00018584    8F  D0 00043          MOVL    #99716, R0                                   ; 1713
                                                 72  11 0004A          BRB     12$                                          ;
00000000G  8F 00000000G  EF      08             00  ED 0004C 4$:      CMPZV   #0, #8, DS$GB_SYSTYPE, #PR$_SYS_TYP900       ; 1719
                                                 16  12 00059          BNEQ    5$                                           ;
                                          22    AA  9F 0005B          PUSHAB  34(RAB)                                      ; 1723
                                          24    AA  9F 0005E          PUSHAB  36(RAB)                                      ;
                         00000000G  EF          02  FB 00061          CALLS   #2, PPA$READ                                 ;
                                 56             50  D0 00068          MOVL    R0, STAT                                     ;
                             08  AA             56  D0 0006B          MOVL    STAT, 8(RAB)                                 ; 1724
                                                 4A  11 0006F          BRB     11$                                          ;
                                 69             02  90 00071 5$:      MOVB    #2, (CBLOCK)                                 ; 1729
                                 02       1E    AA  91 00074          CMPB    30(RAB), #2                                  ; 1731
                                                 06  12 00078          BNEQ    6$                                           ;
                 18   A9     10  AA             06  28 0007A          MOVC3   #6, 16(RAB), 24(CBLOCK)                      ; 1733
                                          28    AA  D4 00080 6$:      CLRL    40(RAB)                                      ; 1735
                                          22    AA  B4 00083          CLRW    34(RAB)                                      ;
                             44  A9       18    A9  D1 00086          CMPL    24(CBLOCK), 68(CBLOCK)                       ; 1737
                                                 09  14 0008B          BGTR    7$                                           ;
                                                 1F  12 0008D          BNEQ    10$                                          ; 1738
                             42  A9       1C    A9  B1 0008F          CMPW    28(CBLOCK), 66(CBLOCK)                       ; 1739
                                                 18  1F 00094          BLSSU   10$                                          ;
                                 02       1E    AA  91 00096 7$:      CMPB    30(RAB), #2                                  ; 1743
                                                 09  12 0009A          BNEQ    8$                                           ;
                                 50 0001865C    8F  D0 0009C          MOVL    #99932, R0                                   ;
                                                 19  11 000A3          BRB     12$                                          ;
                                 50 0001827A    8F  D0 000A5 8$:      MOVL    #98938, R0                                   ;
```

```
                                        10  11 000AC 9$:     BRB     12$                                        ;
                          0C   B9       00  FB 000AE 10$:    CALLS   #0, @12(CBLOCK)                            ; 1745
                               56       50  D0 000B2         MOVL    R0, STAT                                   ;
              10   AA     18   A9       06  28 000B5         MOVC3   #6, 24(CBLOCK), 16(RAB)                    ; 1746
                               50       56  D0 000BB 11$:    MOVL    STAT, R0                                   ; 1749
                                      FAD2  30 000BE 12$:    BSBW    RAB$ERROR                                  ;
                                            04 000C1         RET                                                ; 1750

; Routine Size:  194 bytes,    Routine Base:  CODE + 08BC


;   1751  1
```

```
%sbttl 'RMS$READ routine'

global routine rms$read (rabptr) =

!++
! Functional description:
!       Read from a file, beginning with either the next or some specified
!       VBN.  As many blocks are read as will fit in the user buffer
!
! Inputs:
!       rabptr                  pointer to the RAB to use
!
! Implicit inputs:
!       RAB [rab$l_fab]         The FAB which controls the file
!       RAB [rab$l_bkt]         VBN to read from
!       FAB [fab$w_ifi]         Index in RMS$FILE_TABLE for this file
!       CBLOCK                  Internal control block
!
! Outputs:
!       none
!
! Implicit outputs:
!       The RBF and RSZ fields of the RAB are set to describe the
!       record read.
!
!       RAB [rab$l_sts]         contains the status code
!       RAB [rab$l_stv]         may contain a secondary status
!
! Side effects:
!       CBLOCK                  The next block, the cache control data, and
!                               the tape position control are all subject
!                               to change.
!
! Return codes:
!       RMS$_NORMAL             Everything's nice
!       RMS$_RER                Some read error (RAB [rab$l_stv] contains
!                               the actual SS$_* error code)
!       RMS$_RAB                bad RAB
!       RMS$_FAB                bad FAB
!       RMS$_IFI                bad IFI (or CBLOCK)
!       RMS$_ISI                FAB and RAB not connected
!       RMS$_EOF                VBN specified is past EOF
!       RMS$_SEQ                VBN was specified for access to Andromeda console       [43]            !G:4
!--

    begin

    global register
        cblock = 9 : ref bblock ,                   ! Map to CBLOCK
        rab = 10 : ref bblock ,                     ! Map to RAB
        fab = 11 : ref bblock;                      ! Map to FAB

    local
        stat;

    fab = cblock = 0;
    rab = .rabptr;                                  ! Get pointer to RAB
```

```
;   1810  2
;   1811  2         if not verify_rab ()                        ! If it isn't a RAB
;   1812  2         then
;   1813  2             return (rms$_rab);                      !  no good at all
;   1814  2
;   1815  2         rab [rab$l_stv] = 0;                        ! Clear out STV field
;   1816  2         fab = .rab [rab$l_fab];                     ! Point to RAB's FAB
;   1817  2
;   1818  2         if not verify_fab ()                        ! If it isn't a FAB
;   1819  2         then
;   1820  2             return rab$error (rms$_fab);            !  then quit nicely
;   1821  2
;   1822  2         if not verify_cblock ()                     ! Load CBLOCK pointer,
;   1823  2         then
;   1824  2             return rab$error (rms$_ifi);            !  and quit if IFI's bad
;   1825  2
;   1826  2         if .cblock [ctl$l_rab] neq .rab             ! If this isn't the right RAB
;   1827  2         then
;   1828  2             return rab$error (rms$_isi);            !  say 'bad stream id'
;   1829  2
;   1830  2     !
;   1831  2     ! Now, all the control stuff is OK.  We can go to work
;   1832  2     !
;   1833  2                                                                                                  !G:4
;   1834  3      If  (.DS$GB_SYSTYPE EQL PR$_SYS_TYP900)                !  if Andromeda cpu                [43]   !G:5
;   1835  2          then                                                                                    !G:5
;   1836  3            Begin                                                                                 !G:4
;   1837  3            if .rab [rab$l_bkt] gtr 0                     ! If user specified VBN              [43]   !G:4
;   1838  3            then return rab$error (rms$_seq);             !  return error                      [43]   !G:4
;   1839  3                                                                                                  !G:4
;   1840  3            stat = PPA$READ                               ! Read block (into UBF) using interface [43]   !G:4
;   1841  3                  ( RAB[RAB$L_UBF], RAB[RAB$W_RSZ] );     ! and return record size read (RSZ)    [43]   !G:4
;   1842  3            end                                                                                   !G:4
;   1843  2         else                                                                                     !G:4
;   1844  3            Begin                                                                                 !G:4
;   1845  3            cblock [ctl$b_op] = ctl$c_read;          ! We're on READ operation                    !G:4
;   1846  3                                                                                                  !G:4
;   1847  3            if .cblock [ctl$l_nbp] eql 0             ! If this is first                           !G:4
;   1848  3            then                                                                                  !G:4
;   1849  3            cblock [ctl$l_nbp] = 1;                  ! Set to block 1                             !G:4
;   1850  3                                                                                                  !G:4
;   1851  3            if .rab [rab$l_bkt] gtr 0                     ! If user specified VBN                !G:4
;   1852  3            then                                                                                  !G:4
;   1853  3                cblock [ctl$l_nbp] = .rab [rab$l_bkt];    ! Copy to next block pointer           !G:4
;   1854  3                                                                                                  !G:4
;   1855  3            rab [rab$l_rfa0] = .cblock [ctl$l_nbp]; ! Set RFA to beginning of block              !G:4
;   1856  3            rab [rab$w_rfa4] = 0;                                                                 !G:4
;   1857  3            rab [rab$w_rsz] = rab [rab$l_rbf] = 0;  ! Init context stuff                         [05] !G:4
;   1858  3                                                                                                  !G:4
;   1859  3            if .cblock [ctl$l_nbp] gtr .cblock [ctl$l_eofvbn]       ! If past end of file         !G:4
;   1860  4                or (.cblock [ctl$l_nbp] eql .cblock [ctl$l_eofvbn]  !                           [05] !G:4
;   1861  4                 and .cblock [ctl$w_offbyte] eql 0) !  (check if on null block)                 [05] !G:4
;   1862  3            then                                                                                  !G:4
;   1863  3                return rab$error (rms$_eof);              !  don't bother                        !G:4
;   1864  3                                                                                                  !G:4
;   1865  3            stat = (.cblock [ctl$l_read]) ();             ! Call actual READ routine             !G:4
;   1866  2            END;                                                                                  !G:4
```

```
;   1867  2                                                                                                    !G:4
;   1868  2                                                                                                    !G:4
;   1869  2         rab$error (.stat)                          ! Return whatever status
;   1870  1         end;                                       ! Of RMS$READ



                                                  0FFC 00000          .ENTRY  RMS$READ, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1755
                                                                              R11                                         ;
                                                 59  D4 00002          CLRL    CBLOCK                                      ; 1808
                                                 5B  D4 00004          CLRL    FAB                                         ;
                                  5A       04   AC  D0 00006          MOVL    RABPTR, RAB                                 ; 1809
                                              FAFF  30 0000A          BSBW    VERIFY_RAB                                  ; 1811
                                  08           50  E8 0000D          BLBS    R0, 1$                                      ;
                                  50 0001863C  8F  D0 00010          MOVL    #99900, R0                                  ; 1813
                                                   04 00017          RET                                                 ;
                                           0C   AA  D4 00018 1$:     CLRL    12(RAB)                                     ; 1815
                                  5B       3C   AA  D0 0001B          MOVL    60(RAB), FAB                                ; 1816
                                              FAB9  30 0001F          BSBW    VERIFY_FAB                                  ; 1818
                                  09           50  E8 00022          BLBS    R0, 2$                                      ;
                                  50 0001850C  8F  D0 00025          MOVL    #99596, R0                                  ; 1820
                                               39  11 0002C          BRB     5$                                          ;
                                              FABD  30 0002E 2$:     BSBW    VERIFY_CBLOCK                               ; 1822
                                  09           50  E8 00031          BLBS    R0, 3$                                      ;
                                  50 00018564  8F  D0 00034          MOVL    #99684, R0                                  ; 1824
                                               7D  11 0003B          BRB     13$                                         ;
                                  5A       20   A9  D1 0003D 3$:     CMPL    32(CBLOCK), RAB                             ; 1826
                                               09  13 00041          BEQL    4$                                          ;
                                  50 00018584  8F  D0 00043          MOVL    #99716, R0                                  ; 1828
                                               6E  11 0004A          BRB     13$                                         ;
00000000G  8F 00000000G  EF       08           00  ED 0004C 4$:     CMPZV   #0, #8, DS$GB_SYSTYPE, #PR$_SYS_TYP900      ; 1834
                                               1D  12 00059          BNEQ    7$                                          ;
                                           38   AA  D5 0005B          TSTL    56(RAB)                                     ; 1837
                                               09  15 0005E          BLEQ    6$                                          ;
                                  50 000186AC  8F  D0 00060          MOVL    #100012, R0                                 ; 1838
                                               51  11 00067 5$:     BRB     13$                                         ;
                                           22   AA  9F 00069 6$:     PUSHAB  34(RAB)                                     ; 1841
                                           24   AA  9F 0006C          PUSHAB  36(RAB)                                     ;
                          00000000G  EF         02  FB 0006F          CALLS   #2, PPA$READ                                ;
                                               3F  11 00076          BRB     12$                                         ;
                                  69           03  90 00078 7$:     MOVB    #3, (CBLOCK)                                ; 1845
                                           04   A9  D5 0007B          TSTL    4(CBLOCK)                                   ; 1847
                                               04  12 0007E          BNEQ    8$                                          ;
                              04   A9          01  D0 00080          MOVL    #1, 4(CBLOCK)                               ; 1849
                                           38   AA  D5 00084 8$:     TSTL    56(RAB)                                     ; 1851
                                               05  15 00087          BLEQ    9$                                          ;
                              04   A9      38   AA  D0 00089          MOVL    56(RAB), 4(CBLOCK)                          ; 1853
                              10   AA      04   A9  D0 0008E 9$:     MOVL    4(CBLOCK), 16(RAB)                          ; 1855
                                           14   AA  B4 00093          CLRW    20(RAB)                                     ; 1856
                                           28   AA  D4 00096          CLRL    40(RAB)                                     ; 1857
                                           22   AA  B4 00099          CLRW    34(RAB)                                     ;
                              44   A9      04   A9  D1 0009C          CMPL    4(CBLOCK), 68(CBLOCK)                       ; 1859
                                               07  14 000A1          BGTR    10$                                         ;
                                               0E  12 000A3          BNEQ    11$                                         ; 1860
                                           42   A9  B5 000A5          TSTW    66(CBLOCK)                                  ; 1861
```

```
                                  09  12 000A8          BNEQ    11$                                  ;
                     50 0001827A  8F  D0 000AA 10$:     MOVL    #98938, R0                           ; 1863
                                  07  11 000B1          BRB     13$                                  ;
                 10  B9           00  FB 000B3 11$:     CALLS   #0, @16(CBLOCK)                      ; 1865
                     51           50  D0 000B7 12$:     MOVL    R0, STAT                             ;
                                FA14  30 000BA 13$:     BSBW    RAB$ERROR                            ; 1869
                                      04 000BD          RET                                          ; 1870

; Routine Size:  190 bytes,    Routine Base:  CODE + 097E


;   1871  1
```

```
;    1873  1      %sbttl 'RMS$DISCONNECT routine'
;    1874  1
;    1875  1      global routine rms$disconnect (rabptr) =
;    1876  1
;    1877  1      !++
;    1878  1      ! Functional description:
;    1879  1      !       Disconnect a RAB from a FAB.  This is relatively trivial; the validity
;    1880  1      !       of both RAB and FAB are verified, checks are made to be sure the
;    1881  1      !       file is open and the RAB was connected to that FAB
;    1882  1      !
;    1883  1      ! Inputs:
;    1884  1      !       rabptr          pointer to the RAB to disconnect
;    1885  1      !
;    1886  1      ! Implicit inputs:
;    1887  1      !       RAB [rab$l_fab] points to the FAB to disconnect this RAB from
;    1888  1      !       FAB [fab$w_ifi] indexes the rms$file_table to find the CBLOCK
;    1889  1      !
;    1890  1      ! Outputs:
;    1891  1      !       none
;    1892  1      !
;    1893  1      ! Implicit outputs:
;    1894  1      !       RAB [rab$l_sts] return status
;    1895  1      !       RAB [rab$l_stv] secondary status
;    1896  1      !
;    1897  1      ! Side Effects:
;    1898  1      !       some fields in the RAB and CBLOCK are modified
;    1899  1      !
;    1900  1      ! Return codes:
;    1901  1      !       RMS$_NORMAL     all OK
;    1902  1      !       RMS$_RAB        the RAB block is invalid
;    1903  1      !       RMS$_FAB        the FAB block is invalid
;    1904  1      !       RMS$_IFI        the FAB's IFI field is invalid.
;    1905  1      !       RMS$_ISI        RAB and FAB weren't connected
;    1906  1      !--
;    1907  1
;    1908  2          begin
;    1909  2
;    1910  2          global register
;    1911  2              rab = 10 : ref bblock ,                 ! Pointer to RAB
;    1912  2              fab = 11 : ref bblock ,                 ! Pointer to FAB
;    1913  2              cblock = 9 : ref bblock;                ! Pointer to CBLOCK
;    1914  2
;    1915  2          fab = cblock = 0;
;    1916  2          rab = .rabptr;                              ! Map to RAB
;    1917  2
;    1918  2          if not verify_rab ()                        ! If the RAB is bad
;    1919  2          then
;    1920  2              return (rms$_rab);                      !  return error
;    1921  2
;    1922  2          rab [rab$l_stv] = 0;
;    1923  2          fab = .rab [rab$l_fab];                     ! Map to FAB
;    1924  2
;    1925  2          if not verify_fab ()                        ! If the FAB is bad
;    1926  2          then
;    1927  2              return rab$error (rms$_fab);            !  return error
;    1928  2
;    1929  2          if not verify_cblock ()                     ! If bad IFI
```

```
;    1930  2          then
;    1931  2              return rab$error (rms$_ifi);          !  return with error
;    1932  2
;    1933  2          if .cblock [ctl$l_rab] neq .rab           ! If this isn't the right RAB
;    1934  2          then
;    1935  2              return rab$error (rms$_isi);          !  say 'bad stream id'
;    1936  2
;    1937  2          cblock [ctl$l_rab] = 0;                   ! Clear RAB pointer.
;    1938  2          rab$error (rms$_normal)                   ! Set good return
;    1939  1          end;


                                    0FFC 00000          .ENTRY  RMS$DISCONNECT, Save R2,R3,R4,R5,R6,R7,R8,- ; 1875
                                                                R9,R10,R11                                  ;
                                      59  D4 00002          CLRL    CBLOCK                                ; 1915
                                      5B  D4 00004          CLRL    FAB                                   ;
                          5A      04  AC  D0 00006          MOVL    RABPTR, RAB                           ; 1916
                                    FA41  30 0000A          BSBW    VERIFY_RAB                            ; 1918
                          08            50  E8 0000D          BLBS    R0, 1$                                ;
                          50 0001863C   8F  D0 00010          MOVL    #99900, R0                            ; 1920
                                        04 00017            RET                                           ;
                                    0C  AA  D4 00018 1$:    CLRL    12(RAB)                               ; 1922
                          5B        3C  AA  D0 0001B        MOVL    60(RAB), FAB                          ; 1923
                                    F9FB  30 0001F          BSBW    VERIFY_FAB                            ; 1925
                          09            50  E8 00022          BLBS    R0, 2$                                ;
                          50 0001850C   8F  D0 00025          MOVL    #99596, R0                            ; 1927
                                        28  11 0002C          BRB     5$                                    ;
                                    F9FF  30 0002E 2$:      BSBW    VERIFY_CBLOCK                         ; 1929
                          09            50  E8 00031          BLBS    R0, 3$                                ;
                          50 00018564   8F  D0 00034          MOVL    #99684, R0                            ; 1931
                                        19  11 0003B          BRB     5$                                    ;
                          5A        20  A9  D1 0003D 3$:      CMPL    32(CBLOCK), RAB                       ; 1933
                                        09  13 00041          BEQL    4$                                    ;
                          50 00018584   8F  D0 00043          MOVL    #99716, R0                            ; 1935
                                        0A  11 0004A          BRB     5$                                    ;
                                    20  A9  D4 0004C 4$:      CLRL    32(CBLOCK)                            ; 1937
                          50 00010001   8F  D0 0004F          MOVL    #65537, R0                            ; 1938
                                    F9BA  30 00056 5$:      BSBW    RAB$ERROR                             ;
                                        04 00059            RET                                           ; 1939

; Routine Size:  90 bytes,    Routine Base:  CODE + 0A3C

;    1940  1
```

```
;    1942  1      %sbttl 'RMS$CLOSE routine'
;    1943  1
;    1944  1      global routine rms$close (fabptr) =
;    1945  1
;    1946  1      !++
;    1947  1      ! Functional description
;    1948  1      !       Perform operations necessary to sever a file's connection to
;    1949  1      !       RMS.  This includes de-allocating all internal data blocks,
;    1950  1      !       re-positioning a magtape (if the file is on magtape), and
;    1951  1      !       clearing the file's slot in the RMS$FILE_TABLE.
;    1952  1      !
;    1953  1      ! Inputs:
;    1954  1      !       fabptr          Pointer to the FAB describing the file to be closed
;    1955  1      !
;    1956  1      ! Implicit inputs:
;    1957  1      !       The control block (CBLOCK) associated with the file.
;    1958  1      !
;    1959  1      ! Outputs:
;    1960  1      !       none
;    1961  1      !
;    1962  1      ! Implicit outputs:
;    1963  1      !       FAB [fab$l_sts]         contains return code
;    1964  1      !
;    1965  1      ! Side Effects:
;    1966  1      !       Deallocate file's CBLOCK and clear the file's slot in the file
;    1967  1      !       pointer table, and the FAB's IFI.
;    1968  1      !
;    1969  1      ! Return codes:
;    1970  1      !       RMS$_NORMAL             All OK
;    1971  1      !       possible magtape positioning error code
;    1972  1      !--
;    1973  1
;    1974  2          begin
;    1975  2
;    1976  2          global register
;    1977  2              cblock = 9 : ref bblock ,                  ! map control block
;    1978  2              rab = 10,                                  ! register for rab (not used)
;    1979  2              fab = 11 : ref bblock;                     ! To map FAB
;    1980  2                                                                                          !G:20
;    1981  2          local
;    1982  2              stat;
;    1983  2
;    1984  3       If  (.DS$GB_SYSTYPE EQL PR$_SYS_TYP900)               !  if Andromeda cpu          [43]    !G:5
;    1985  2           Then                                             ! then call the PPA interface to [43]    !G:5
;    1986  3               Begin                                        ! open the file              [43]    !G:4
;    1987  3               Stat = PPA$CLOSE();                          !                            [43]    !G:4
;    1988  3               Return  .Stat                                !                            [43]    !G:4
;    1989  2               End;                                         !                            [43]    !G:4
;    1990  2                                                                                                 !G:4
;    1991  2                                                                                                 !G:4
;    1992  2          cblock = rab = 0;                                  ! None
;    1993  2          fab = .fabptr;                                     ! Set up FAB pointer
;    1994  2
;    1995  2          if not verify_fab () then return (rms$_fab);           ! Error if not rea FAB
;    1996  2
;    1997  2          fab [fab$l_sts] = 0;                               ! Init some fields
;    1998  2          fab [fab$l_stv] = 0;
```

```
;    1999  2
;    2000  2          if not verify_cblock ()                    ! If bad IFI
;    2001  2          then
;    2002  2              return rms$error (rms$_ifi);           !  return error
;    2003  2
;    2004  2          cblock [ctl$b_op] = ctl$c_close;           ! Specify operation
;    2005  2
;    2006  2          if .cblock [ctl$l_rab] neq 0               ! If RAB wasn't disconnected
;    2007  2          then
;    2008  3              begin
;    2009  3
;    2010  3              local
;    2011  3                  stat;
;    2012  3
;    2013  3              stat = rms$disconnect (.cblock [ctl$l_rab]);    ! Disconnect it
;    2014  3
;    2015  3              if not .stat
;    2016  3              then
;    2017  3                  return rms$error (.stat)           ! Error?
;    2018  2              end;
;    2019  2
;    2020  3          stat = (if .cblock [ctl$l_close] neq 0     ! If there's a close routine
;    2021  3          then (.cblock [ctl$l_close]) ()            !  call it
;    2022  2          else rms$_normal);                         !  otherwise, set good status
;    2023  2          rms$error (.stat)                          ! De-allocate, return
;    2024  1          end;


                                          0FFC 00000          .ENTRY  RMS$CLOSE, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 1944
                                                                      R10,R11                                          ;
00000000G  8F 00000000G  EF         08          00  ED 00002          CMPZV   #0, #8, DS$GB_SYSTYPE, #PR$_SYS_TYP900   ; 1984
                                                0B  12 0000F          BNEQ    1$                                       ;
                      00000000G  EF             00  FB 00011          CALLS   #0, PPA$CLOSE                            ; 1987
                                 52             50  D0 00018          MOVL    R0, STAT                                 ;
                                                04  0001B          RET                                              ; 1988
                                                59  7C 0001C 1$:     CLRQ    CBLOCK                                   ; 1992
                                 5B       04    AC  D0 0001E          MOVL    FABPTR, FAB                              ; 1993
                                             F99E 30 00022          BSBW    VERIFY_FAB                               ; 1995
                                 08             50  E8 00025          BLBS    R0, 2$                                   ;
                                 50 0001850C    8F  D0 00028          MOVL    #99596, R0                               ;
                                                04  0002F          RET                                              ;
                                          08    AB  7C 00030 2$:     CLRQ    8(FAB)                                   ; 1997
                                             F9A0 30 00033          BSBW    VERIFY_CBLOCK                            ; 2000
                                 09             50  E8 00036          BLBS    R0, 3$                                   ;
                                 50 00018564    8F  D0 00039          MOVL    #99684, R0                               ; 2002
                                                2B  11 00040          BRB     7$                                       ;
                                 69             04  90 00042 3$:     MOVB    #4, (CBLOCK)                             ; 2004
                                          20    A9  D5 00045          TSTL    32(CBLOCK)                               ; 2006
                                                0B  13 00048          BEQL    4$                                       ;
                                          20    A9  DD 0004A          PUSHL   32(CBLOCK)                               ; 2013
                                    FF54  CF        01  FB 0004D          CALLS   #1, RMS$DISCONNECT                       ;
                                 18             50  E9 00052          BLBC    STAT, 7$                                 ; 2015
                                          14    A9  D5 00055 4$:     TSTL    20(CBLOCK)                               ; 2020
                                                09  13 00058          BEQL    5$                                       ;
```

```
                                14   B9            00  FB 0005A          CALLS   #0, a20(CBLOCK)                          ; 2021
                                     52            50  D0 0005E          MOVL    R0, STAT                                 ;
                                                   07  11 00061          BRB     6$                                       ;
                                     52 00010001   8F  D0 00063 5$:      MOVL    #65537, STAT                             ; 2020
                                     50            52  D0 0006A 6$:      MOVL    STAT, R0                                 ; 2023
                                                 F7D3  30 0006D 7$:      BSBW    RMS$ERROR                                ;
                                                       04 00070          RET                                              ; 2024

; Routine Size:  113 bytes,    Routine Base:  CODE + 0A96


;   2025  1
;   2026  1     end
;   2027  1
;   2028  0     eludom




;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  WORK                           16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  DATA                         1064  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                         2823  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                       0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)



Symbol                      Type  Defined  Referenced ...
--------------------------- ------- ----- ---------------
$$_ROWS_$$                  Comptime   380   384     385     386     387     388     389     390     391     392     393     394
                                          395     396     397     398     399     400     401     402     403     404     405
                                          406     407     408     409     410     411     412     413     414     415     416
                                          417     418     419     423
$ASCIC                      Macro    Lib01   384     385     386     387     388     389     390     391     392     393     394
                                          395     396     397     398     399     400     401     402     403     404     405
                                          406     407     408     409     410     411     412     413     414     415     416
                                          417     418     419
$DS_DSADEF                  Macro    Lib01   309
$DS_HPEO_DECL               Macro    Lib01   524     525     526    1365    1366    1367
$DS_HPEO_F                  Fieldset Lib01   524     525     526    1365    1366    1367
$DS_HPO_DECL                Macro    Lib01   521     527     528     532     533     534     535    1382    1383    1384    1385
                                         1386    1387    1388
$DS_HPO_F                   Fieldset Lib01   521     527     528     532     533     534     535    1382    1383    1384    1385
                                         1386    1387    1388
$FAB_DECL                   Macro    Lib03  1011
A                           Unbound          393     394     395     396     397     398     404     405     406     407
ADAPTER                     Local      528   536=    539a.   543.    545.    546=    546a.   550aa   563=    568aa   579=    610aa
                            Local     1384  1397=   1400a.  1404.   1406.   1407=   1407a.  1411aa  1424=   1429aa  1440=   1475.
                                         1475a.  1496.   1496a.  1529a.
ADAPTERTYPE                 Local     1372  1451a   1527.
```

Symbol	Type	Defined	Referenced ...
ADAPTER_ADDR_EXT	Local	1368	1475= 1476a. 1496= 1497a.
ADAPTER_EPB	Local	524	
	Local	1365	1476= 1477a. 1497= 1498a.
ADAPTER_TYPE	Bind	610	613. 614. 618. 619. 620. 625. 626. 627. 632. 633. 638. 639. 644. 645. 650. 651.
ADC	RoutForm	514	732a=
ADPPHY	RoutForm	450	470.
ADPTYPE	RoutForm	450	468.
ANSI$OPEN	ExtRout	340	1492a
AP	Builtin	730	730a. 730.
	Builtin	734	734a. 734.
	Builtin	738	738a. 738.
	Builtin	742	742a. 742.
BBLOCK	Structur	Lib02	318 319 776 777 877 878 881 942 1014 1016 1047 1072 1097 1131 1166 1222 1223 1224 1229 1230 1231 1309 1587 1588 1589 1686 1687 1688 1801 1802 1803 1911 1912 1913 1977 1979
BINODE	RoutForm	450	466.
	Local	1377	1393= 1477= 1483= 1498= 1533.
BOO$AL_VECTOR	External	321	467a 474= 474a
BOO$SELECT	External	320	474a
BREAKUP_FILE	ExtRout	335	1276c 1277c 1279c 1284c 1285c 1287c 1289c 1292c
BTD	RoutForm	514	736a=
BTD$K_BDA	Literal	Lib03	410 411 412 413 414 415
BTD$K_BVPSSP	Literal	Lib03	416 417 418 419
BTD$K_CONSOLE	Literal	Lib03	384 385 703 705 711 726
BTD$K_DL	Literal	Lib03	398
BTD$K_DM	Literal	Lib03	397
BTD$K_DQ	Literal	Lib03	395
BTD$K_HSCCI	Literal	Lib03	404 408 409
BTD$K_HSCMU	Literal	Lib02	405
BTD$K_MB	Literal	Lib03	386 387 388 389 390 391 392
BTD$K_MU	Literal	Lib02	403
BTD$K_TF	Literal	Lib02	394
BTD$K_TM	Literal	Lib02	393
BTD$K_TS	Literal	Lib02	399 400 401 402
BTD$K_UDA	Literal	Lib03	396 406 407
BUFSIZ	RoutForm	908	947.
CACHE	Bind	780	821. 824. 825.
CALL_RMS	Linkage	Lib02	274 337 338 339 340 341 342 343 844
CBLOCK	ExtReg	776	780aa 784. 787a. 791a. 796a. 799a. 800a. 803a. 805a. 807a. 810a. 812a. 824a. 828. 830= 836. 836a=
	ExtReg	877	895a. 896a. 902a.
	ExtReg	942	947a= 950a= 954a= 958a=
	GlobReg	1014	1018= 1022a=
	ExtReg	1130	1139= 1141=
	GlobReg	1222	1238= 1253= 1257. 1258a= 1259a= 1260a= 1331a= 1332a= 1333a= 1337a= 1360a= 1397a= 1491a= 1502a= 1512a= 1525a=
	GlobReg	1589	1591= 1616a. 1620a= 1621a= 1622a= 1623a= 1627a=
	GlobReg	1686	1693= 1711a. 1729a= 1733a= 1737a. 1738a. 1739a. 1745ac 1746a.
	GlobReg	1801	1808= 1826a. 1845a= 1847a. 1849a= 1853a= 1855a. 1859a. 1860a. 1861a. 1865ac
	GlobReg	1913	1915= 1933a. 1937a=
	GlobReg	1977	1992= 2004a= 2006a. 2013a. 2020a. 2021ac
CBLOCKSIZE	Local	1227	1253= 1259.
CHECK_DEVICE	Local	529	582= 591a. 596a.

Symbol	Type	Defined	Referenced ...							
CIA	Unbound		386	387	388	389	390	391	392	393	394
CINUMBER	Local	1376	1392=	1478=	1500=	1532.					
CIPORT	RoutForm	450	473.								
CODE	RoutForm	772	782.	784.	786.	838.					
	RoutForm	1028	1049.								
	RoutForm	1053	1074.								
CONTROLLER	Local	527	536=	542.	543=	562=	578=	582aa			
	Local	1383	1397=	1403.	1404=	1423=	1439=	1478a.	1480.	1480a.	1500a.	1528a.
CONTROLLER_ADDR_EXT	Local	1369	1480=	1481a.							
CONTROLLER_EPB	Local	525									
	Local	1366	1481=	1482a.	1483a.						
CSRPHY	RoutForm	450	469.								
CTL$B_OP	Macro	Lib02	787	791	836	1022	1260	1729	1845	2004	
CTL$C_CLOSE	Literal	Lib02	787	791	1022	2004					
CTL$C_FILHDR_SIZ	Literal	Lib02	799								
CTL$C_GET	Literal	Lib02	1729								
CTL$C_LENGTH	Literal	Lib02	1253	1258							
CTL$C_OPEN	Literal	Lib02	791	1260							
CTL$C_READ	Literal	Lib02	1845								
CTL$L_CACHE1	Macro	Lib02	780	950							
CTL$L_CACHE2	Macro	Lib02	954								
CTL$L_CACHE3	Macro	Lib02	958								
CTL$L_CLOSE	Macro	Lib02	2020	2021							
CTL$L_EOFVBN	Macro	Lib02	895	1737	1738	1859	1860				
CTL$L_FILE_PTR	Macro	Lib02	810	812	1333	1337					
CTL$L_FILHDR	Macro	Lib02	796	799	800						
CTL$L_GET	Macro	Lib02	1745								
CTL$L_NBP	Macro	Lib02	1627	1847	1849	1853	1855	1859	1860		
CTL$L_PTABLE	Macro	Lib02	1397								
CTL$L_RAB	Macro	Lib02	1616	1620	1711	1826	1933	1937	2006	2013	
CTL$L_READ	Macro	Lib02	1865								
CTL$L_RPB	Macro	Lib02	803	805	1525						
CTL$L_STARTLBN	Macro	Lib02	902								
CTL$L_VBN	Macro	Lib02	1621	1737	1738						
CTL$V_ANSI	Macro	Lib02	1491								
CTL$V_ODS	Macro	Lib02	1502								
CTL$V_RT11	Macro	Lib02	1512								
CTL$W_BYTE	Macro	Lib02	1622	1739							
CTL$W_CACHE_SIZE	Macro	Lib02	824	947							
CTL$W_FILE_ALLOC	Macro	Lib02	810	1332							
CTL$W_FILE_LEN	Macro	Lib02	807	1331							
CTL$W_OFFBYTE	Macro	Lib02	896	1739	1861						
CTL$W_REC_SIZE	Macro	Lib02	1623								
CTL$W_RFA	Macro	Lib02	1733	1746							
CTL$W_SIZE	Macro	Lib02	1259								
CTL$W_UNIT	Macro	Lib02	1360								
D	Bind	1309	1315.	1316.							
DEBNT	Unbound		386	387	388	389	390	391	392	393	394
DEF$C_BKSTP_LEN	GlobBind	371	1284								
DEF$C_ULTRIX_LEN	GlobBind	373	1276								
DEF$Q_BACKSTOP	GlobBind	372	1284a								
DEF$Q_DEV	External	319	1277.	1278.	1285.	1286.					
DEF$Q_DIR	External	318	1279.	1280.	1287.	1288.					
DEF$Q_ULTRIX	GlobBind	374	1276a								
DEV$M_DIR	Literal	Lib03	298	299							

Symbol	Type	Defined	Referenced ...									
DEV$M_FOD	Literal	Lib03	297	298	299							
DEV$M_RND	Literal	Lib03	298	299								
DEV$M_SDI	Literal	Lib03	299									
DEV$M_SQD	Literal	Lib03	297									
DEVICE	RoutForm	514	521m	536.	678aa							
	Local	1382	1394=	1396.	1398.	1443.	1443a.	1445a.	1451.	1486.	1486a.	
DEVICE$_ANSI	Literal	297	1490									
DEVICE$_ODS	Literal	298	1501									
DEVICE$_RT11	Literal	299	1511									
DEVICETYPE	RoutForm	450	465.									
	Local	1373	1451a	1526.								
DEVICEUNIT	Local	1374	1447=	1449=	1482=	1530.						
DEVICE_ADDR_EXT	Local	1370	1443=	1444a.	1486=	1487a.						
DEVICE_EPB	Local	526										
	Local	1367	1444=	1447a.	1449a.	1487=	1488a.					
DEVICE_NAME	Bind	678	680.	685.								
DEVICE_TYPE	Bind	589	591.	593.	594.	595.						
DEVLEN	Local	1346	1359=	1360.	1394.							
DEVPTR	Local	1347	1359=	1394.								
DEV_TYPE	Bind	550	556.	557.	558.	559.						
	Bind	568	574.	575.								
	Bind	1411	1417.	1418.	1419.	1420.						
	Bind	1429	1435.	1436.								
DRIVE	RoutForm	450	471.									
DRIVELIST	Unbound		384	385	386	387	388	389	390	391	392	399	400
			401	402	403	408	409	410	411	412	413	414	415
			416	417	418	419							
	Macro	Lib02	393	394	395	396	397	398	404	405	406	407	
DRIVE_VECTOR	Bind	667	669	673.	677.							
DS$GA_SUPPORT_TABLE	GlobBind	383	583a									
DS$GB_CI_INIT	External	314	404a	405a								
DS$GB_KDB50_INIT	External	315	410a	411a	412a	413a	414a	415a				
DS$GB_PB_INIT	External	316	416a	417a	418a	419a						
DS$GB_SYSTYPE	External	317	727.	1460.	1719.	1834.	1984.					
DS$GB_UDA50_INIT	External	313	396a	406a	407a							
DS$GK_SUPPORT_TABLE_ROWS	GlobLit	423	585									
DS$_ERROR	Literal	Lib01	724	728								
DS$_PROGERR	Literal	Lib01	671	1517								
DSA$AL_APTMAIL	Bind	309	309									
DSA$GL_APTCOM	Bind	309										
DSA$GL_DEVLEN	Bind	309										
DSA$GL_DEVNAM	Bind	309										
DSA$GL_ERRNO	Bind	309										
DSA$GL_EVENT	Bind	309										
DSA$GL_FLAGS	Bind	309	476									
DSA$GL_MSGPTR	Bind	309										
DSA$GL_MSGTYP	Bind	309										
DSA$GL_PASSES	Bind	309										
DSA$GL_PASSNO	Bind	309										
DSA$GL_SECTNO	Bind	309										
DSA$GL_SID	Bind	309	329m	706	707	708	712	713	714			
DSA$GL_SUBTNO	Bind	309										
DSA$GL_TESTNO	Bind	309										
DSA$GL_UNITS	Bind	309										
DSA$V_USER	Macro	Lib01	476									

Symbol	Type	Defined	Referenced ...								
DSC$A_POINTER	Macro	Lib03	1278	1280	1286	1288	1303	1316	1337	1358	
DSC$C_S_BLN	Literal	Lib03	1229	1231	1269	1309					
DSC$W_LENGTH	Macro	Lib03	1277	1279	1285	1287	1315	1323	1331	1337	1357
DSR$ALLOCATE_PSEUDO_RPB	GlobRout	449	1525c								
DSR$DEALLOCATE_PSEUDO_RPB	GlobRout	483	805c								
DSX$CHECK_SUPPORT	GlobRout	514	1451c								
EXE$ALONONPAGED	ExtRout	333	459c	950c	954c	958c	1253c	1330c			
EXE$DEANONPAGED	ExtRout	332	486c	800c	812c	825c	828c				
FAB	ExtReg	777	782a=	829a.	831a=						
	ExtReg	878	883a.	894a.	897a.	899a.	900a.				
	GlobReg	1016	1019=	1021a=							
	ExtReg	1072	1074a=								
	ExtReg	1097	1099a.	1100a.							
	ExtReg	1131	1136a.								
	GlobReg	1224	1239=	1243a.	1245a=	1251a=	1255a=	1261a=	1266a=	1267a=	1268a=	1289a.	1290a.
			1292a.	1293a.	1456a=	1490a=	1501a=	1511a=	1517a=				
	GlobReg	1588	1592=	1606=									
	GlobReg	1688	1693=	1701=									
	GlobReg	1803	1808=	1816=									
	GlobReg	1912	1915=	1923=									
	GlobReg	1979	1993=	1997a=	1998a=								
FAB$B_BID	Macro	Lib03	1099										
FAB$B_BKS	Macro	Lib03	1266										
FAB$B_BLN	Macro	Lib03	1100										
FAB$B_DNS	Macro	Lib03	1289										
FAB$B_FNS	Macro	Lib03	1292										
FAB$B_FSZ	Macro	Lib03	897										
FAB$B_ORG	Macro	Lib03	1243										
FAB$B_RAT	Macro	Lib03	894										
FAB$B_RFM	Macro	Lib03	900										
FAB$C_BID	Literal	Lib03	1099										
FAB$C_BLN	Literal	Lib03	1011	1100									
FAB$C_SEQ	Literal	Lib03	901	1243									
FAB$ERROR	Routine	1053	279f										
FAB$L_DEV	Macro	Lib03	1490	1501	1511								
FAB$L_DNA	Macro	Lib03	1290										
FAB$L_FNA	Macro	Lib03	1293										
FAB$L_MRN	Macro	Lib03	1267										
FAB$L_SDC	Macro	Lib03	1267										
FAB$L_STS	Macro	Lib03	782	1074	1251	1255	1268	1997					
FAB$L_STV	Macro	Lib03	1267	1456	1517	1998							
FAB$L_XAB	Macro	Lib03	883										
FAB$W_IFI	Macro	Lib03	829	831	1021	1136	1245	1261					
FAB$W_MRS	Macro	Lib03	899	1266									
FABPTR	RoutForm	1178	1239.										
	RoutForm	1944	1993.										
FALSE	Literal	295											
FIL$CVTFILNAM	ExtRout	334											
FILE	RoutForm	514	740a=										
FILEDESC	Local	1229	1269=	1276a	1278a	1280a	1284a	1286a	1288a	1290a	1293a	1309a	
FILEFORMAT	Local	1379	1451a	1471.									
FILENAME	Local	1230	1301a	1323a									
FILENAMEDESC	Local	1231	1303=	1323=	1331.	1337.	1337a.	1357.	1358.	1463a			
FILE_INDEX	Local	1228	1246=	1249.	1257.	1261.							
FOUND_ADAPTER	Local	602	602=	616=	622=	629=	635=	641=	647=	653=	657=	700.	

Symbol	Type	Defined	Referenced ...									
FOUND_DEVICE	Local	531	531=	700=	702.	724.						
FOUND_UNIT	Local	603	603=	673.	689=	698=	700.					
HP$A_DEVICE	Field	Lib01	1528.	1529.								
HP$A_LINK	Field	Lib01	539.	546.	1400.	1407.						
HP$B_CI_TEM	Field	Lib01	1478.	1500.								
HP$K_LENGTH	Literal	Lib01	521	527	528	532	533	534	535	1382	1383	1384	1385
			1386	1387	1388							
HP$T_DEVICE	Field	Lib01	1445.									
HP$T_TYPE	Field	Lib01	550a	568a	582a	610a	678a	1411a	1429a			
HP$W_SIZE	Field	Lib01	1443.	1475.	1480.	1486.	1496.					
HPE$K_LENGTH	Literal	Lib01	524	525	526	1365	1366	1367				
HPE$W_EXT_DRIVE	Field	Lib01	1447.	1449.	1477.	1482.	1483.	1488.	1498.			
HUB	Unbound		386	387	388	389	390	391	392	393	394	
I	Local	585										
	Local	819	821.	824.	825.							
	Local	982	983.									
	Local	1004	1006.	1018.	1021.							
	Local	1246	1247.									
	Local	1305	1309.									
IFI	Local	1134	1136=	1138.	1139.							
INDEX	Local	664	664=	673.	677.	693=	693.					
INIT	RoutForm	514	744a=									
INITBYTE	Local	1380	1451a	1467.	1469a=							
JSB_ALLOC	Linkage	Lib02	333									
JSB_CBLOCK	Linkage	Lib02	275	908								
JSB_DEALL	Linkage	Lib02	332									
JSB_DEVICE	Linkage	1351	1355									
JSB_LOC$PTABLE	Linkage	Lib02	336									
JSB_NONE	Linkage	Lib02	276	277	980	988						
JSB_RMS	Linkage	Lib02	273	278	279	280	281	282	772	1028	1053	1078	1108
			1147									
KDB50	Unbound		386	387	388	389	390	391	392	393	394	
KFBTA	Unbound		386	387	388	389	390	391	392	393	394	
LEN	Local	1312	1315=	1318=	1318.							
LENGTH	Local	1299	1302=	1318=	1318.							
LOC$PTABLE	ExtRout	336	1394c									
MFPR	Builtin	454	476									
NAM$C_MAXRSS	Literal	Lib03	1230	1302								
NULLPARAMETER	Builtin	518	730	734	738	742						
ODS	Unbound		393	394	399	400	401	402	403	405	408	409
ODS$OPEN	ExtRout	338	1507a									
OPENROUTINE	Local	1378	1391=	1492=	1505=	1507=	1513=	1522.	1534ac			
POINTER	Local	1298	1301=	1303.	1319a=	1319=	1319.	1323.				
PPA$CLOSE	ExtRout	343	1987c									
PPA$LOOKUP	ExtRout	341	1463c									
PPA$READ	ExtRout	342	1722c	1840c								
PR$V_SID_TYPE	Macro	Lib03	706	707	708	712	713	714				
PR$_SBR	Literal	Lib03	476									
PR$_SID_TYP8NN	Literal	Lib03	707	713								
PR$_SID_TYP8RR	Literal	304	708	714								
PR$_SID_TYP8SS	Literal	Lib03	706	712								
PR$_SYS_TYP900	ExtLit	323	727	1460	1719	1834	1984					
PREV_ADAPTER	Local	532	536=	544.	545=	563.						
	Local	1386	1397=	1405.	1406=	1424.						
PREV_CONTROLLER	Local	533	536=	541.	542=	562.						

Symbol	Type	Defined	Referenced ...									
	Local	1385	1397=	1402.	1403=	1423.						
PREV_PREV_ADAPTER	Local	535	536=	544=	579.							
	Local	1388	1397=	1405=	1440.							
PREV_PREV_CONTROLLER	Local	534	536=	541=	578.							
	Local	1387	1398=	1402=	1439.							
PTR	Local	1313	1316=	1319.	1319=							
RAB	GlobReg	1015	1020=									
	ExtReg	1047	1049a=									
	ExtReg	1166	1168a.	1169a.								
	GlobReg	1223	1237=									
	GlobReg	1587	1593=	1599a=	1601a.	1602a.	1606a.	1620.	1624a=	1625a=	1626a=	
	GlobReg	1687	1694=	1700a=	1701a.	1711.	1723aa	1724a=	1731a.	1733a.	1735a=	1743a.	1746a=
	GlobReg	1802	1809=	1815a=	1816a.	1826.	1837a.	1841aa	1851a.	1853a.	1855a=	1856a=	1857a=
	GlobReg	1911	1916=	1922a=	1923a.	1933.							
	GlobReg	1978	1992=									
RAB$B_BID	Macro	Lib03	1168									
RAB$B_BLN	Macro	Lib03	1169									
RAB$B_RAC	Macro	Lib03	1601	1602	1731	1743						
RAB$C_BID	Literal	Lib03	1168									
RAB$C_BLN	Literal	Lib03	1169									
RAB$C_RFA	Literal	Lib03	1602	1731	1743							
RAB$C_SEQ	Literal	Lib03	1601									
RAB$ERROR	Routine	1028	278f	1604c	1610c	1614c	1618c	1628c	1705c	1709c	1713c	1741c	1749c
			1820c	1824c	1828c	1838c	1863c	1869c	1927c	1931c	1935c	1938c	
RAB$L_BKT	Macro	Lib03	1837	1851	1853								
RAB$L_FAB	Macro	Lib03	1606	1701	1816	1923							
RAB$L_RBF	Macro	Lib03	1625	1735	1857								
RAB$L_RFA0	Macro	Lib03	1855										
RAB$L_STS	Macro	Lib03	1049	1724									
RAB$L_STV	Macro	Lib03	1599	1700	1815	1922							
RAB$L_UBF	Macro	Lib03	1723	1841									
RAB$W_RFA	Macro	Lib03	1624	1733	1746								
RAB$W_RFA4	Macro	Lib03	1856										
RAB$W_RSZ	Macro	Lib03	1626	1723	1735	1841	1857						
RABPTR	RoutForm	1549	1593.										
	RoutForm	1634	1694.										
	RoutForm	1755	1809.										
	RoutForm	1875	1916.										
RMS$ALLOC_CACHE	GlobRout	908	275f										
RMS$CLEANUP	GlobRout	988	277f										
RMS$CLOSE	GlobRout	1944	287f										
RMS$CONNECT	GlobRout	1549	284f										
RMS$C_MAXFILES	Literal	296	1004	1138									
RMS$DISCONNECT	GlobRout	1875	286f	2013c									
RMS$ERROR	GlobRout	772	273f	1023c	1243c	1335c	1396c	1457c	1518c	1540c	2002c	2017c	2023c
RMS$FILE_TABLE	Global	365	829=	983=	1006.	1018.	1139.	1247.	1257=				
RMS$GET	GlobRout	1634	285f										
RMS$INIT	GlobRout	980	276f										
RMS$K_ANSI	Unbound		384	385	386	387	388	389	390	391	392	395	396
			397	398	404	406	407	410	411	412	413	414	415
			416	417	418	419							
	Literal	Lib02	393	394	399	400	401	402	403	405	408	409	1473
RMS$K_ODS	Unbound		384	393	394	399	400	401	402	403	405	408	409
	Literal	Lib02	385	386	387	388	389	390	391	392	395	396	397
			398	404	406	407	410	411	412	413	414	415	416

Symbol	Type	Defined	Referenced ...										
			417	418	419	1494							
RMS$K_RT11	Unbound		385	386	387	388	389	390	391	392	393	394	395
			396	397	398	399	400	401	402	403	404	405	406
			407	408	409	410	411	412	413	414	415	416	417
			418	419									
	Literal	Lib02	384	1509									
RMS$LOAD_XAB	GlobRout	844	274f	1539c									
RMS$OPEN	GlobRout	1178	283f										
RMS$READ	GlobRout	1755											
RMS$_CCR	Literal	Lib03	1618										
RMS$_DEV	Literal	Lib03	1396	1457	1518	1537							
RMS$_DME	Literal	Lib03	952	956	960	1251	1255	1335					
RMS$_EOF	Literal	Lib03	1743	1863									
RMS$_FAB	Literal	Lib03	1241	1610	1705	1820	1927	1995					
RMS$_IFI	Literal	Lib03	1614	1709	1824	1931	2002						
RMS$_ISI	Literal	Lib03	1713	1828	1935								
RMS$_NORMAL	Literal	Lib03	962	1628	1938	2022							
RMS$_ORG	Literal	Lib03	1243										
RMS$_RAB	Literal	Lib03	1597	1698	1813	1920							
RMS$_RAC	Literal	Lib03	1604										
RMS$_RFA	Literal	Lib03	1743										
RMS$_SEQ	Literal	Lib03	1838										
RMS$_SUC	Literal	Lib03	1268										
RPB	Local	457	459=	463a=	464a=	464.	465a=	466a=	467a=	468a=	469a=	470a=	471a=
			472a=	473a=	476a=	478.							
	RoutForm	483	485.	486.									
RPB$B_CONFREG	Macro	Lib03	468										
RPB$B_DEVTYP	Macro	Lib03	465										
RPB$B_SLAVE	Macro	Lib03	472										
RPB$C_LENGTH	Literal	Lib03	303										
RPB$L_ADPPHY	Macro	Lib03	470										
RPB$L_ADPVIR	Macro	Lib03	470										
RPB$L_BOOTR1	Macro	Lib03	466										
RPB$L_BOOTR2	Macro	Lib03	473										
RPB$L_CSRPHY	Macro	Lib03	469										
RPB$L_CSRVIR	Macro	Lib03	469										
RPB$L_IOVEC	Macro	Lib03	467										
RPB$L_SVASPT	Macro	Lib03	476										
RPB$W_UNIT	Macro	Lib03	471										
RPB$_LENGTH	Literal	303	459	463	485								
RT11	Unbound		385	386	387	388	389	390	391	392	393	394	395
			396	397	398	399	400	401	402	403	404	405	406
			407	408	409	410	411	412	413	414	415	416	417
			418	419									
RT11$OPEN	ExtRout	337	1513a										
SCAN$DEVICE	ExtRout	1355	1357c										
SIZE	Local	945	947=	948=	948.	950.	954.	958.					
SLAVE	RoutForm	450	472.										
SLAVENUMBER	Local	1375	1390=	1488=	1531.								
SPT$B_ADC	Field	Lib02	612.	617.	624.	631.	637.	643.	649.	732.			
SPT$B_BTD	Field	Lib02	703.	705.	711.	726.	736.						
SPT$B_FILE	Field	Lib02	709.	715.	740.								
SPT$K_ROW_SIZE	Literal	Lib02	530	721									
SPT$L_DRIVES	Field	Lib02	667.										
SPT$L_INIT	Field	Lib02	744.										

Symbol	Type	Defined	Referenced ...
SPT$L_TYPE	Field	Lib02	589.
SPT$V_ADAPTER	Unbound		393 394 395 396 397 398 404 405 406 407
	Field	Lib02	386 387 388 389 390 391 392 393 394 395 396
			397 398 399 400 401 402 403 404 405 406 407
			408 409 410 411 412 413 414 415 416 417 418
			419 605.
SPT$V_CONTROLLER	Unbound		386 387 388 389 390 391 392 393 394 395 396
			397 398 399 400 401 402 403 404 405 406 407
			408 409 410 411 412 413 414 415 416 417 418
			419
	Field	Lib02	393 394 395 396 397 398 404 405 406 407 659.
SS$_NORMAL	Literal	Lib03	746
STAT	Local	1348	1361= 1540.
	Local	1691	1722= 1724. 1745= 1749.
	Local	1806	1840= 1865= 1869.
	Local	1982	1987= 1988. 2020= 2023.
	Local	2011	2013= 2015. 2017.
STATUS	Local	1381	1451= 1453. 1456. 1463= 1464.
SUPPORTFLAGS	Macro	Lib02	384 385 386 387 388 389 390 391 392 393 394
			395 396 397 398 399 400 401 402 403 404 405
			406 407 408 409 410 411 412 413 414 415 416
			417 418 419
SUPPORT_ENTRY	Macro	Lib02	384 385 386 387 388 389 390 391 392 393 394
			395 396 397 398 399 400 401 402 403 404 405
			406 407 408 409 410 411 412 413 414 415 416
			417 418 419
SUPPORT_ENTRY_DEF	Macro	Lib02	530
SUPPORT_ENTRY_FIELD	Fieldset	Lib02	530
SUPPORT_ROW	Local	530	583= 589a. 605a. 612a. 617a. 624a. 631a. 637a. 643a. 649a. 659a.
			667a. 703a. 705a. 709a. 711a. 715a. 721= 721. 726a. 732a. 736a.
			740a. 744a.
TRUE	Literal	294	616 622 629 635 641 647 653 657 689 698 1491
			1502 1512
UBA	Unbound		386 387 388 389 390 391 392 393 394
ULTRIX$OPEN	ExtRout	339	1505a
ULTRIX_FLAGS	External	312	1273. 1503.
ULTRIX_V_MODE	Literal	326	1273 1503
UNIT_NAME	Bind	677	680. 682. 683. 684.
VERIFY_CBLOCK	Routine	1108	280f 1612c 1707c 1822c 1929c 2000c
VERIFY_FAB	Routine	1078	282f 1241c 1608c 1703c 1818c 1925c 1995c
VERIFY_RAB	Routine	1147	281f 1595c 1696c 1811c 1918c
X	Local	1011	1019a
XAB	Local	881	883= 885. 886a. 887= 887a. 889. 890a. 894a= 895a= 896a= 897a=
			898a= 899a= 900a= 901a= 902a=
XAB$B_ATR	Macro	Lib03	894
XAB$B_BLN	Macro	Lib03	890
XAB$B_COD	Macro	Lib03	886
XAB$B_HSZ	Macro	Lib03	897
XAB$B_RFO	Macro	Lib03	900 901
XAB$C_FHC	Literal	Lib03	886
XAB$C_FHCLEN	Literal	Lib03	890
XAB$L_EBK	Macro	Lib03	895
XAB$L_NXT	Macro	Lib03	887
XAB$L_SBN	Macro	Lib03	902
XAB$W_FFB	Macro	Lib03	896

```
Symbol                        Type  Defined  Referenced ...
------------------------- -------- ----- ----------------
XAB$W_LRL                  Macro    Lib03   898
XAB$W_MRZ                  Macro    Lib03   899


                CROSS REFERENCE MAP

Line #      Event          File ...
-----  ---------------  ---------
    1  Source (start)   DISK$DS:[DS.LOCAL.RELEASE.WORK]RMS.B32;2
   24  Module RMS
  262  Library #1       DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
  264  Library #2       DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
  266  Library #3       SYS$COMMON:[SYSLIB]LIB.L32;2
 2028  Eludom RMS


        KEY TO REFERENCE TYPE FLAGS

            .   Fetch
            =   Store
            c   Routine call
            a   Address use
            @   Indirect use
            e   External, external routine, or external literal declaration
            f   Forward or forward routine declaration
            h   Condition handler enabling
            m   Map declaration
            u   Undeclare declaration


;                           Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                          940        27         2        96          00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                          675        66         9        47          00:00.1
;  SYS$COMMON:[SYSLIB]LIB.L32;2           20450       103         0      1101          00:00.8




;                                  COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:RMS.B32

; Size:          2823 code + 1080 data bytes
; Run Time:         00:13.9
; Elapsed Time:     00:17.5
```

```
; Lines/CPU Min:    8785
; Lexemes/CPU-Min:131532
; Memory Used:  396 pages
; Compilation Complete
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element RT11.B32
;    0002  0      ! *1     6-FEB-1986 15:22:22 DS ""
;    0003  0      ! DEC/CMS REPLACEMENT HISTORY, Element RT11.B32
;    0004  0      %title '*** RT11 Console media RMS routines'
;    0005  0      module RT11 (                                   ! RMS$OPEN, RMS$GET, RMS$READ, RMS$CLOSE services
;    0006  0                   ident = '01-04'
;    0007  0                   ) =
;    0008  0      !
;    0009  0      ! Copyright (c) 1980, 1981
;    0010  0      ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0011  0      !
;    0012  0      ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0013  0      ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0014  0      ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0015  0      ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0016  0      ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0017  0      ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0018  0      ! REMAIN IN DEC.
;    0019  0      !
;    0020  0      ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0021  0      ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0022  0      ! CORPORATION.
;    0023  0      !
;    0024  0      ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0025  0      ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0026  0
;    0027  0      !++
;    0028  0      ! FACILITY: Diagnostic Supervisor
;    0029  0      !
;    0030  0      ! ABSTRACT:
;    0031  0      !
;    0032  0      !       This module supports RMS operations on the VAX console media
;    0033  0      !       (RT-11 formatted RX-01/02 or TU-58).
;    0034  0      !
;    0035  0      ! AUTHOR:
;    0036  0      !       Dave Butenhof,          15-aug-1980
;    0037  0      !
;    0038  0      ! MODIFIED BY:
;    0039  0      !
;    0040  0      !               Dave Butenhof, 6-feb-1981, version 6.3
;    0041  0      !       01      Make $GET return full record size in RAB$L_STV on
;    0042  0      !               RMS$_RTB error.
;    0043  0      !               - Dave butenhof, 02-Jun-1981, version 6.4
;    0044  0      !       02      Fix spec. of libraries for flexibility
;    0045  0      !               - Dave Butenhof, 26-Jun-1981, version 6.4
;    0046  0      !       03      Delete useage of DSP$XX_READ routine; use bootdrivers
;    0047  0      !               (boo$qio) for console storage read.
;    0048  0      !
;    0049  0      !       04      Jack Stansbury, 1-June-1982, Version 6.8
;    0050  0      !               Fixed a truncation error.
;    0051  0      !--
;    0052  0
```

```
;    0054  0     %sbttl 'declarations'
;    0055  1     begin
;    0056  1     !
;    0057  1     ! include files:
;    0058  1     !
;    0059  1
;    0060  1     library '$diag';                !
;    0061  1
;    0062  1     library '$ds';                  !
;    0063  1
;    0064  1     library 'sys$library:lib.l32';
;    0065  1
;    0066  1     !
;    0067  1     ! Internal definitions
;    0068  1     !
;    0069  1
;    0070  1     linkage
;    0071  1         jsb_acc = jsb : global (block = 6, vbn = 7, cblock = 9, rab = 10),
;    0072  1         jsb_adv = jsb (register = 0) : global (vbn = 7, offset = 8);
;    0073  1
;    0074  1     !
;    0075  1     ! table of contents:
;    0076  1     !
;    0077  1
;    0078  1     forward routine
;    0079  1         advance     : Addressing_Mode (Long_Relative) jsb_adv novalue,      ! Advance internal 'RFA' pointers       [04]
;    0080  1         access      : Addressing_Mode (Long_Relative) jsb_acc,              ! Access block of file                  [04]
;    0081  1         rt11$open   : Addressing_Mode (Long_Relative) call_rms,             ! RMS$OPEN emulator--open file          [04]
;    0082  1         rt11$get    : Addressing_Mode (Long_Relative) call_rms,             ! RMS$GET emulator--read/de-block       [04]
;    0083  1         rt11$read   : Addressing_Mode (Long_Relative) call_rms;             ! RMS$READ emulator--read block-mode    [04]
;    0084  1
;    0085  1
;    0086  1     !
;    0087  1     ! External references
;    0088  1     !
;    0089  1
;    0090  1     external routine
;    0091  1         fil$cvtfilnam       : Addressing_Mode (Long_Relative),                  ! Convert ascii filename to RAD50
;    0092  1         rms$alloc_cache     : Addressing_Mode (Long_Relative) Jsb_cblock,       ! Allocate cache blocks
;    0093  1         boo$qio             : Addressing_Mode (Long_Relative);                  ! Logical block I/O
;    0094  1     !
;    0095  1     ! psect definitions:
;    0096  1     !
;    0097  1
;    0098  1     psect
;    0099  1         plit = data ( share, addressing_mode (long_relative));
;    0100  1
;    0101  1     psect
;    0102  1         own = work ( read, write);
;    0103  1
;    0104  1     psect
;    0105  1         global = work ( read, write);
;    0106  1
;    0107  1     psect
;    0108  1         code = code (share, addressing_mode (long_relative));       !                                           [04]
;    0109  1
```

```
;    0111  1     %sbttl 'advance record pointers'
;    0112  1     routine advance (length) : jsb_adv novalue =
;    0113  2         begin
;    0114  2
;    0115  2         external register
;    0116  2             vbn = 7,
;    0117  2             offset = 8;
;    0118  2
;    0119  2         offset = .offset + .length;                 ! Advance pointer
;    0120  2
;    0121  2         if .offset gtr 511                          ! If we overlapped block
;    0122  2         then
;    0123  3             begin
;    0124  3             vbn = .vbn + (.offset/512);             ! Increment VBN
;    0125  3             offset = .offset and 511                ! Slice down offset
;    0126  3             end
;    0127  3
;    0128  1         end;


                                                        .TITLE  RT11 *** RT11 Console media RMS routines
                                                        .IDENT  \01-04\

                                                        .EXTRN  FIL$CVTFILNAM, RMS$ALLOC_CACHE
                                                        .EXTRN  BOO$QIO

                                                        .PSECT  CODE,NOWRT,  SHR,2

                        58              50  C0 00000 ADVANCE:ADDL2   LENGTH, OFFSET                    ; 0119
                000001FF   8F           58  D1 00003         CMPL    OFFSET, #511                      ; 0121
                                        10  15 0000A         BLEQ    1$                                ;
              50            58 00000200   8F  C7 0000C         DIVL3   #512, OFFSET, R0                  ; 0124
                            57            50  C0 00014         ADDL2   R0, VBN                           ;
        58    58            09            00  EF 00017         EXTZV   #0, #9, OFFSET, OFFSET            ; 0125
                                            05  0001C 1$:      RSB                                       ; 0128

; Routine Size:  29 bytes,    Routine Base:  CODE + 0000
```

```
;    0130  1      %sbttl 'Access file block via cache'
;    0131  1      routine access : jsb_acc =
;    0132  1
;    0133  1      !++
;    0134  1      ! Functional description:
;    0135  1      !       If the desired VBN is not in cache, refresh cache.  Return
;    0136  1      !       pointer to appropriate cache buffer
;    0137  1      !
;    0138  1      ! Implicit inputs:
;    0139  1      !       R7              desired VBN
;    0140  1      !       R9              RMS control block
;    0141  1      !       R10             RAB
;    0142  1      !
;    0143  1      ! Implicit outputs:
;    0144  1      !       R6              pointer to buffer
;    0145  1      !
;    0146  1      ! Side effects:
;    0147  1      !       may alter cache and perform I/O to floppy.
;    0148  1      !--
;    0149  1
;    0150  2          begin
;    0151  2
;    0152  2          external register
;    0153  2              block = 6,
;    0154  2              vbn = 7,
;    0155  2              cblock = 9 : ref bblock,
;    0156  2              rab = 10 : ref bblock;
;    0157  2
;    0158  2          bind
;    0159  2              cache = cblock [ctl$l_cache1] : vector;
;    0160  2
;    0161  2          if .vbn geq .cblock [ctl$l_eofvbn]          ! If past end of file,
;    0162  2          then
;    0163  2              return (if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof); !  that's it.
;    0164  2
;    0165  2          if .vbn lss .cblock [ctl$l_lvbn]            ! If block's not in
;    0166  2              or .vbn geq .cblock [ctl$l_hvbn]        !  cache now, get it
;    0167  2          then
;    0168  3              begin
;    0169  3
;    0170  3              if .cblock [ctl$w_cache_size] eql 0     ! If no cache buffers
;    0171  3              then
;    0172  4                  begin                               ! make them happen
;    0173  4
;    0174  4                  local
;    0175  4                      stat;
;    0176  4
;    0177  4                  stat = rms$alloc_cache (1);         ! Allocate
;    0178  4
;    0179  4                  if not .stat                        ! If failure
;    0180  4                  then
;    0181  5                      begin
;    0182  5                      rab [rab$l_stv] = .stat;
;    0183  5                      return rms$_dme
;    0184  5                      end
;    0185  5
;    0186  3                  end;
```

```
;   0187  3
;   0188  3             block = .vbn + .cblock [ctl$l_startlbn] - 1;    ! First block to read
;   0189  3
;   0190  3             incr i from 0 to 2 do                          ! Read the blocks
;   0191  4                 begin
;   0192  4
;   0193  4                 local
;   0194  4                     stat;
;   0195  4
;   0196  5                 if (.block - .cblock [ctl$l_startlbn] - 1)  ! If the 3-block window
;   0197  4                     geq .cblock [ctl$l_eofvbn]             !  overlaps EOF, don't
;   0198  4                 then
;   0199  4                     exitloop;                              !  try to read any more
;   0200  4
;   0201  4                 stat = boo$qio (.cache [.i],               ! Read to cache buffer i                    [03]
;   0202  4                     512,                ! one block                                                    [03]
;   0203  4                     .block,             ! logical block number                                         [03]
;   0204  4                     io$_readlblk,       ! read it                                                      [03]
;   0205  4                     0,                  ! physical address                                             [03]
;   0206  4                     .cblock [ctl$l_rpb]);   ! address of RPB                                           [03]
;   0207  4
;   0208  4                 if not .stat
;   0209  4                 then
;   0210  5                     begin
;   0211  5                     rab [rab$l_stv] = .stat;               ! Set read code
;   0212  5                     return rms$_rer
;   0213  4                     end;
;   0214  4
;   0215  4                 block = .block + 1                         ! Advance block #
;   0216  3                 end;
;   0217  3
;   0218  3             cblock [ctl$l_lvbn] = .vbn;                    ! Window begins here
;   0219  3             cblock [ctl$l_hvbn] = min (.vbn + 3,           ! Ends at VBN +3 or
;   0220  3                 .cblock [ctl$l_eofvbn])                    !  at EOF
;   0221  2             end;
;   0222  2
;   0223  2         block = .cache [.vbn - .cblock [ctl$l_lvbn]];          ! Get pointer
;   0224  2         1
;   0225  1         end;                                               ! of access



                                        0C  BB 00000 ACCESS: PUSHR   #^M<R2,R3>                          ; 0131
                        53       2C     A9  9E 00002         MOVAB   44(CBLOCK), R3                      ; 0159
                 44     A9              57  D1 00006         CMPL    VBN, 68(CBLOCK)                     ; 0161
                                        18  19 0000A         BLSS    2$                                  ;
                        02       1E     AA  91 0000C         CMPB    30(RAB), #2                         ; 0163
                                        09  12 00010         BNEQ    1$                                  ;
                        50 0001865C     8F  D0 00012         MOVL    #99932, R0                          ;
                                        70  11 00019         BRB     6$                                  ;
                        50 0001827A     8F  D0 0001B 1$:     MOVL    #98938, R0                          ;
                                        67  11 00022         BRB     6$                                  ;
                 28     A9              57  D1 00024 2$:     CMPL    VBN, 40(CBLOCK)                     ; 0165
                                        06  19 00028         BLSS    3$                                  ;
                 24     A9              57  D1 0002A         CMPL    VBN, 36(CBLOCK)                     ; 0166
```

```
                                                79  19 0002E            BLSS    10$                                                     ;
                                            0A  A9  B5 00030 3$:        TSTW    10(CBLOCK)                                              ; 0170
                                                19  12 00033            BNEQ    4$                                                      ;
                                        50      01  D0 00035            MOVL    #1, R0                                                  ; 0177
                                    00000000G   EF  16 00038            JSB     RMS$ALLOC_CACHE                                         ;
                                        0D      50  E8 0003E            BLBS    STAT, 4$                                                ; 0179
                                    0C  AA      50  D0 00041            MOVL    STAT, 12(RAB)                                           ; 0182
                                        50 000184D4 8F  D0 00045        MOVL    #99540, R0                                              ; 0183
                                                67  11 0004C            BRB     11$                                                     ;
                    50                  57      50  A9  C1 0004E 4$:    ADDL3   80(CBLOCK), VBN, R0                                     ; 0188
                                        56      FF  A0  9E 00053        MOVAB   -1(R0), BLOCK                                           ;
                                                52  D4 00057            CLRL    I                                                       ; 0206
                    50                  56      50  A9  C3 00059 5$:    SUBL3   80(CBLOCK), BLOCK, R0                                   ; 0196
                                                50  D7 0005E            DECL    R0                                                      ;
                                    44  A9          50  D1 00060        CMPL    R0, 68(CBLOCK)                                          ; 0197
                                                2D  18 00064            BGEQ    8$                                                      ;
                                            38  A9  DD 00066            PUSHL   56(CBLOCK)                                              ; 0206
                                        7E          21  7D 00069        MOVQ    #33, -(SP)                                              ; 0201
                                                56  DD 0006C            PUSHL   BLOCK                                                   ; 0203
                                        7E      0200 8F  3C 0006E       MOVZWL  #512, -(SP)                                             ; 0201
                                              6342  DD 00073            PUSHL   (R3)[I]                                                 ;
                        00000000G   EF          06  FB 00076            CALLS   #6, BOO$QIO                                             ;
                                        0D      50  E8 0007D            BLBS    STAT, 7$                                                ; 0208
                                    0C  AA      50  D0 00080            MOVL    STAT, 12(RAB)                                           ; 0211
                                        50 0001C0F4 8F  D0 00084        MOVL    #114932, R0                                             ; 0212
                                                28  11 0008B 6$:        BRB     11$                                                     ;
                                                56  D6 0008D 7$:        INCL    BLOCK                                                   ; 0215
                    C6                  52      02  F3 0008F            AOBLEQ  #2, I, 5$                                               ;
                                    28  A9          57  D0 00093 8$:    MOVL    VBN, 40(CBLOCK)                                         ; 0218
                                        50      03  A7  9E 00097        MOVAB   3(R7), R0                                               ; 0219
                                    44  A9          50  D1 0009B        CMPL    R0, 68(CBLOCK)                                          ; 0220
                                                04  15 0009F            BLEQ    9$                                                      ;
                                        50      44  A9  D0 000A1        MOVL    68(CBLOCK), R0                                          ;
                                    24  A9          50  D0 000A5 9$:    MOVL    R0, 36(CBLOCK)                                          ; 0219
                        50              57      28  A9  C3 000A9 10$:   SUBL3   40(CBLOCK), VBN, R0                                    ; 0223
                                        56        6340  D0 000AE        MOVL    (R3)[R0], BLOCK                                         ;
                                        50          01  D0 000B2        MOVL    #1, R0                                                  ; 0225
                                                    0C  BA 000B5 11$:   POPR    #^M<R2,R3>                                              ;
                                                    05 000B7            RSB                                                             ;

; Routine Size:  184 bytes,    Routine Base:  CODE + 001D
```

```
;    0227  1      %sbttl 'RT-11 OPEN service'
;    0228  1
;    0229  1      global routine rt11$open : call_rms =
;    0230  1
;    0231  1      !++
;    0232  1      ! Functional description:
;    0233  1      !       Access a file on the console media (floppy or TU-58) to be
;    0234  1      !       sure it exists and to ascertain it's properties (such as
;    0235  1      !       size).  Return this information in the FAB block and in
;    0236  1      !       the RMS file control block, which is pointed to by the
;    0237  1      !       entry in RMS$FILE_TABLE indexed by the FAB's IFI field.
;    0238  1      !
;    0239  1      ! Inputs:
;    0240  1      !       none
;    0241  1      !
;    0242  1      ! Implicit inputs:
;    0243  1      !       R11     points to the FAB
;    0244  1      !       R9      points to the CBLOCK (RMS internal file control block)
;    0245  1      !
;    0246  1      ! Outputs:
;    0247  1      !       none
;    0248  1      !
;    0249  1      ! Implicit outputs:
;    0250  1      !       Several fields in the FAB and in the static file control block
;    0251  1      !       are altered to describe the file.  Also, if an XAB (XABFHC is
;    0252  1      !       the only XAB currently supported) exists, it will be filled in.
;    0253  1      !
;    0254  1      ! Return status
;    0255  1      !       RMS$_DME        insufficient memory for allocation of buffers
;    0256  1      !       RMS$_NEF        non existant file
;    0257  1      !       RMS$_ACC        can't access media's directory segment
;    0258  1      !--
;    0259  1
;    0260  2          begin
;    0261  2
;    0262  2          local
;    0263  2              ptr,
;    0264  2              len,
;    0265  2              segment,                            ! Directory segment number
;    0266  2              rad50nam : vector [3],              ! Hold RAD50 name
;    0267  2              rad50desc : bblock [dsc$c_s_bln],   ! Descriptor for RAD50 cvt.
;    0268  2              segblk : bblock [1024],             ! Buffer to read in segment
;    0269  2              address_first,                      ! First entry of segment
;    0270  2              extra_length,                       ! Extra len of entries in seg
;    0271  2              start_block,                        ! Starting block of file
;    0272  2              file_length;                        ! Length of file (blocks)
;    0273  2
;    0274  2          external register
;    0275  2              cblock = 9 : ref bblock,            ! Pointer to control block
;    0276  2              rab = 10,
;    0277  2              fab = 11 : ref bblock;              ! Define the FAB
;    0278  2
;    0279  2          bind
;    0280  2              unit = cblock [ctl$w_unit] : word;
;    0281  2
;    0282  2          cblock [ctl$l_get] = rt11$get;          ! Store GET routine address
;    0283  2          cblock [ctl$l_read] = rt11$read;        ! Store READ routine address
```

```
;    0284  2          ptr = .cblock [ctl$l_file_ptr];             ! Get filename descriptor
;    0285  2          len = .cblock [ctl$w_file_len];
;    0286  2
;    0287  2          while 1 do                                  ! Skip to end of directory
;    0288  3              begin
;    0289  3
;    0290  3              local
;    0291  3                  char : byte;
;    0292  3
;    0293  3              if (len = .len - 1) lss 0               ! If no more string
;    0294  3              then
;    0295  3                  return (rms$_fnm);                  !  invalid file name
;    0296  3
;    0297  3              char = ch$rchar_a (ptr);                ! Get next character
;    0298  3
;    0299  3              if .char eql %ascii']'                  ! If it's a directory
;    0300  3                  or .char eql %ascii'>'              !  terminator
;    0301  3              then
;    0302  3                  exitloop                            ! then leave scan loop
;    0303  2              end;
;    0304  2
;    0305  2          rad50desc [dsc$w_length] = .len;            ! Load filename descr.
;    0306  2          rad50desc [dsc$a_pointer] = .ptr;
;    0307  2
;    0308  2          if not fil$cvtfilnam (rad50desc, rad50nam)  ! Try to RAD50 the name
;    0309  2          then
;    0310  2              return (rms$_fnm);                      ! Bad filename if can't
;    0311  2
;    0312  2          segment = 1;                                ! Start with first segment
;    0313  2
;    0314  2          while 1 do                                  ! Find directory entry
;    0315  3              begin
;    0316  3
;    0317  3              local
;    0318  3                  stat;
;    0319  3
;    0320  3              if .segment eql 0 then return (rms$_fnf);       ! If no more, file not found
;    0321  3
;    0322  4              if not (stat = boo$qio (segblk, ! Read to buffer segblk                          [03]
;    0323  4                          1024,       !  read two blocks (a segment)                       [03]
;    0324  4                          4 * .segment*2,     !  logical block number                       [03]
;    0325  4                          io$_readlblk,       !  read operation                             [03]
;    0326  4                          0,          !  physical address                                   [03]
;    0327  4                          .cblock [ctl$l_rpb]))       !                                     [03]
;    0328  3              then
;    0329  4                  (fab [fab$l_stv] = .stat;           ! If can't read, file access
;    0330  3                  return (rms$_acc));                 !  error.
;    0331  3
;    0332  3              segment = .segblk [rt$w_segment];       ! Link to next segment
;    0333  3              start_block = .segblk [rt$w_startblock];        ! Where segment starts
;    0334  3              extra_length = .segblk [rt$w_extralen]; ! Variable part of entries
;    0335  3              address_first = segblk [rt$w_first] - rt$_fixed - .extra_length;        ! First entry address
;    0336  3
;    0337  3              if
;    0338  3
;    0339  3                  while 1 do                          ! Find file name in segment
;    0340  4                      begin
```

```
;   0341  4
;   0342  4                     bind
;   0343  4                         entry = (address_first = .address_first + rt$_fixed + .extra_length) : bblock;
;   0344  4
;   0345  4                     if .entry [rt$v_endseg] then exitloop 0;        ! End of segment
;   0346  4
;   0347  4                     file_length = .entry [rt$w_filelength]; ! Get length of file
;   0348  4                     start_block = .file_length + .start_block;      ! Start of next file
;   0349  4
;   0350  4                     if .entry [rt$v_perm]           ! If file is permanent
;   0351  4                         and .entry [rt$l_filename] eql .rad50nam [0]        ! and filename matches
;   0352  4                         and .entry [rt$w_filetype] eql .(rad50nam + 6)<0, 16, 0>
;   0353  4                     then
;   0354  4                         exitloop 1
;   0355  4
;   0356  4                     end                             ! ( value 1 if found, 0 if end of segment )
;   0357  3             then
;   0358  3
;   0359  3                 exitloop                            ! Access file if found
;   0360  2             end;                                    ! of segment loop
;   0361  2
;   0362  2         start_block = .start_block - .file_length;  ! Back up pointer
;   0363  2     !
;   0364  2     ! Initialize control block fields
;   0365  2     !
;   0366  2         cblock [ctl$w_offbyte] = 0;                 ! Set fst RFA past EOF
;   0367  2         cblock [ctl$l_eofvbn] = .file_length + 1;   !  VBN of said RFA
;   0368  2         cblock [ctl$l_startlbn] = .start_block;     ! First block of file
;   0369  2         fab [fab$l_alq] = .file_length;             ! Set # blocks in file
;   0370  2         fab [fab$l_fop] = fab$m_ctg;                ! RT-11 files are contiguous
;   0371  2         fab [fab$b_fsz] = 0;                        ! Clear fixed record size
;   0372  2         fab [fab$b_rat] = fab$m_cr;                 ! Treat as CR format
;   0373  2         fab [fab$b_rfm] = fab$c_rt11;               ! RT-11 format
;   0374  2         fab [fab$w_mrs] = 512;                      ! Max rec size of 1 block
;   0375  2         rms$_normal                                 ! Return OK
;   0376  1         end;                                        ! of RT11$OPEN


                                        007C 00000          .ENTRY  RT11$OPEN, Save R2,R3,R4,R5,R6          ; 0229
                    5E      FBEC    CE  9E 00002          MOVAB   -1044(SP), SP                           ;
                0C  A9 00000000V    EF  9E 00007          MOVAB   RT11$GET, 12(CBLOCK)                    ; 0282
                10  A9 00000000V    EF  9E 0000F          MOVAB   RT11$READ, 16(CBLOCK)                   ; 0283
                    51        4C    A9  D0 00017          MOVL    76(CBLOCK), PTR                         ; 0284
                    50        48    A9  3C 0001B          MOVZWL  72(CBLOCK), LEN                         ; 0285
                                    50  D7 0001F 1$:      DECL    LEN                                     ; 0293
                                    26  19 00021          BLSS    3$                                      ;
                    52              81  90 00023          MOVB    (PTR)+, CHAR                            ; 0297
                5D  8F              52  91 00026          CMPB    CHAR, #93                               ; 0299
                                    05  13 0002A          BEQL    2$                                      ;
                    3E              52  91 0002C          CMPB    CHAR, #62                               ; 0300
                                    EE  12 0002F          BNEQ    1$                                      ;
                EC  AD              50  B0 00031 2$:      MOVW    LEN, RAD50DESC                          ; 0305
                F0  AD              51  D0 00035          MOVL    PTR, RAD50DESC+4                        ; 0306
                            F4      AD  9F 00039          PUSHAB  RAD50NAM                                ; 0308
```

```
                                  EC   AD  9F 0003C          PUSHAB  RAD50DESC                                  ;
          00000000G  EF              02  FB 0003F          CALLS   #2, FIL$CVTFILNAM                          ;
                     08              50  E8 00046          BLBS    R0, 4$                                     ;
                     50 0001852C     8F  D0 00049 3$:      MOVL    #99628, R0                                 ; 0310
                                         04 00050          RET                                                ;
                     53              01  D0 00051 4$:      MOVL    #1, SEGMENT                                ; 0312
                                     53  D5 00054 5$:      TSTL    SEGMENT                                    ; 0320
                                     08  12 00056          BNEQ    6$                                         ;
                     50 00018292     8F  D0 00058          MOVL    #98962, R0                                 ;
                                         04 0005F          RET                                                ;
                                  38   A9  DD 00060 6$:      PUSHL   56(CBLOCK)                                 ; 0327
                     7E              21  7D 00063          MOVQ    #33, -(SP)                                 ; 0322
        7E           53              01  78 00066          ASHL    #1, SEGMENT, -(SP)                         ; 0324
                     6E              04  C0 0006A          ADDL2   #4, (SP)                                   ;
                     7E       0400   8F  3C 0006D          MOVZWL  #1024, -(SP)                               ; 0322
                                  14   AE  9F 00072          PUSHAB  SEGBLK                                     ;
          00000000G  EF              06  FB 00075          CALLS   #6, BOO$QIO                                ;
                     0C              50  E8 0007C          BLBS    STAT, 7$                                   ;
                0C   AB              50  D0 0007F          MOVL    STAT, 12(FAB)                              ; 0329
                     50 0001C002     8F  D0 00083          MOVL    #114690, R0                                ; 0330
                                         04 0008A          RET                                                ;
                     53         02   AE  3C 0008B 7$:      MOVZWL  SEGBLK+2, SEGMENT                          ; 0332
                     56         08   AE  3C 0008F          MOVZWL  SEGBLK+8, START_BLOCK                      ; 0333
                     54         06   AE  3C 00093          MOVZWL  SEGBLK+6, EXTRA_LENGTH                     ; 0334
                     50         FC   AE  9E 00097          MOVAB   SEGBLK-4, R0                               ; 0335
        52           50              54  C3 0009B          SUBL3   EXTRA_LENGTH, R0, ADDRESS_FIRST            ;
                     52         0E A442  9E 0009F 8$:      MOVAB   14(EXTRA_LENGTH)[ADDRESS_FIRST], -         ; 0343
                                                                   ADDRESS_FIRST                              ;
                     50              52  D0 000A4          MOVL    ADDRESS_FIRST, R0                          ;
        A9           60              0B  E0 000A7          BBS     #11, (R0), 5$                              ; 0345
                     55         08   A0  3C 000AB          MOVZWL  8(R0), FILE_LENGTH                         ; 0347
                     56              55  C0 000AF          ADDL2   FILE_LENGTH, START_BLOCK                   ; 0348
        E9           60              0A  E1 000B2          BBC     #10, (R0), 8$                              ; 0350
                F4   AD         02   A0  D1 000B6          CMPL    2(R0), RAD50NAM                            ; 0351
                                     E2  12 000BB          BNEQ    8$                                         ;
                FA   AD         06   A0  B1 000BD          CMPW    6(R0), RAD50NAM+6                          ; 0352
                                     DB  12 000C2          BNEQ    8$                                         ;
                     56              55  C2 000C4          SUBL2   FILE_LENGTH, START_BLOCK                   ; 0362
                                  42   A9  B4 000C7          CLRW    66(CBLOCK)                                 ; 0366
                44   A9         01   A5  9E 000CA          MOVAB   1(R5), 68(CBLOCK)                          ; 0367
                50   A9              56  D0 000CF          MOVL    START_BLOCK, 80(CBLOCK)                    ; 0368
                10   AB              55  D0 000D3          MOVL    FILE_LENGTH, 16(FAB)                       ; 0369
                04   AB 00100000     8F  D0 000D7          MOVL    #1048576, 4(FAB)                           ; 0370
                                  3F   AB  94 000DF          CLRB    63(FAB)                                    ; 0371
                1E   AB       0402   8F  B0 000E2          MOVW    #1026, 30(FAB)                             ; 0372
                36   AB       0200   8F  B0 000E8          MOVW    #512, 54(FAB)                              ; 0374
                     50 00010001     8F  D0 000EE          MOVL    #65537, R0                                 ; 0376
                                         04 000F5          RET                                                ;

; Routine Size:  246 bytes,    Routine Base:  CODE + 00D5

;   0377  1
```

```
;    0379  1      %sbttl 'RT11$GET routine'
;    0380  1
;    0381  1      global routine rt11$get : call_rms =
;    0382  1
;    0383  1      !++
;    0384  1      ! Functional description:
;    0385  1      !       Access a single record from the record stream, copying it to
;    0386  1      !       the user's buffer.
;    0387  1      !
;    0388  1      ! Inputs:
;    0389  1      !       none
;    0390  1      !
;    0391  1      ! Implicit inputs:
;    0392  1      !       R9              pointer to internal control CBLOCK
;    0393  1      !       R10             pointer to RAB
;    0394  1      !       various fields within all three control structures
;    0395  1      !
;    0396  1      ! Outputs:
;    0397  1      !       none
;    0398  1      !
;    0399  1      ! Implicit outputs:
;    0400  1      !       RAB [rab$l_rbf] Address where record is stored
;    0401  1      !       RAB [rab$w_rsz] Size of record
;    0402  1      !       RAB [rab$l_sts] Status return code
;    0403  1      !
;    0404  1      ! Side effects:
;    0405  1      !       CBLOCK          current RFA will be updated if access is sequential.
;    0406  1      !                       record size is set if read is successful.
;    0407  1      !                       Cache contents and control data may be affected
;    0408  1      ! Status codes:
;    0409  1      !       RMS$_NORMAL     good return
;    0410  1      !       RMS$_RTB        record too big (truncated)
;    0411  1      !       RMS$_EOF        RFA is beyond last record of file
;    0412  1      !       RMS$_RER        Read error on media
;    0413  1      !--
;    0414  1
;    0415  2          begin
;    0416  2
;    0417  2          external register
;    0418  2              cblock = 9 : ref bblock,                    ! Map CBLOCK
;    0419  2              rab = 10 : ref bblock;                      ! Map RAB
;    0420  2
;    0421  2          global register
;    0422  2              block = 6 : ref vector [, byte],            ! Access block
;    0423  2              vbn = 7,                                    ! Current file 'VBN'
;    0424  2              offset = 8;                                 ! Current block offset
;    0425  2
;    0426  2          local
;    0427  2              outlim,                                     ! Remaining length of buffer
;    0428  2              outptr,                                     ! Address to copy to
;    0429  2              rtb;                                        ! Flag for record truncated
;    0430  2
;    0431  2          vbn = .cblock [ctl$l_vbn];                      ! Grab the intended RFA
;    0432  2          offset = .cblock [ctl$w_byte];
;    0433  2
;    0434  2          if .rab [rab$b_rac] neq rab$c_rfa               ! If not random access
;    0435  2          then
```

```
;    0436  2              advance (.cblock [ctl$w_rec_size]);     ! advance RFA to next record
;    0437  2
;    0438  2          cblock [ctl$l_vbn] = .vbn;                  ! Save new RFA
;    0439  2          cblock [ctl$w_byte] = .offset;
;    0440  2          cblock [ctl$w_rec_size] = 0;                ! If bad error, repeat record
;    0441  2          rtb = 0;                                    ! Record's not too big yet
;    0442  2          rab [rab$w_rsz] = 0;                        ! Clear record size
;    0443  2          outlim = .rab [rab$w_usz];                  ! Limiting size for transfer
;    0444  2          rab [rab$l_rbf] = outptr = .rab [rab$l_ubf];        ! Set output pointer
;    0445  2
;    0446  2          while 1 do                                  ! Read record
;    0447  3              begin
;    0448  3
;    0449  3              while 1 do                              ! Read up to CR or EOF
;    0450  4                  begin
;    0451  4
;    0452  4                  local
;    0453  4                      cr,                             ! Flag
;    0454  4                      string_size,                    ! Size of partial string
;    0455  4                      bite,                           ! Current offset in block
;    0456  4                      len;                            ! Length to copy
;    0457  4
;    0458  4                  cr = 0;                             ! Clear CR flag
;    0459  4                  bite = .offset;                     ! Where to start in block
;    0460  5                  begin
;    0461  5
;    0462  5                  local
;    0463  5                      stat;
;    0464  5
;    0465  5                  stat = access ();                   ! Access the block
;    0466  5
;    0467  5                  if not .stat then return .stat
;    0468  5
;    0469  4                  end;
;    0470  4
;    0471  4                  if .block [.offset] eql 0           ! If reached EOF
;    0472  4                  then
;    0473  4                      return (if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof); !  get out
;    0474  4
;    0475  5                  while not (cr = .block [.bite] eql %x'd')   ! Find end of block or CR
;    0476  4                      and .bite leq 511 do
;    0477  4                      bite = .bite + 1;               ! Advance
;    0478  4
;    0479  4                  string_size = .bite - .offset;      ! Get length of this string
;    0480  4                  len = min (.string_size, .outlim);  ! Get length to transfer
;    0481  4
;    0482  4                  if .outlim lss .string_size         ! If not enough space,
;    0483  4                  then
;    0484  4                      rtb = 1;                        !  set flag for overflow
;    0485  4
;    0486  4                  outptr = ch$move (.len, block [.offset], .outptr);  ! Copy as much as we can
;    0487  4                  outlim = .outlim - .len;            ! Count size of copy
;    0488  4                  rab [rab$w_rsz] = .rab [rab$w_rsz] + .len;
;    0489  4                  advance (.string_size);             ! Advance past this block
;    0490  4
;    0491  4                  if .cr
;    0492  4                  then
```

```
;   0493  4                 exitloop                        ! Keep looping until end
;   0494  3             end;
;   0495  3
;   0496  3         advance (1);                            ! Skip the CR
;   0497  4         begin
;   0498  4
;   0499  4         local
;   0500  4             stat;
;   0501  4
;   0502  4         stat = access ();                       ! Access new RFA
;   0503  4
;   0504  4         if not .stat then return .stat
;   0505  4
;   0506  3         end;
;   0507  3
;   0508  3         if .block [.offset] eql %x'a'           ! If CR is followed by LF
;   0509  3         then
;   0510  4             begin
;   0511  4             cblock [ctl$w_rec_size] =           ! Set whole record size
;   0512  4             (.vbn - .cblock [ctl$l_vbn])*512 + .offset - .cblock [ctl$w_byte] + 1;
;   0513  4             return
;   0514  5                 (if .rtb
;   0515  5                 then
;   0516  6                     begin
;   0517  6                     rab [rab$l_stv] = .cblock [ctl$w_rec_size];
;   0518  6                     rms$_rtb
;   0519  6                     end
;   0520  5                 else rms$_normal)
;   0521  4             end
;   0522  3         else                                    ! Not end of record yet
;   0523  3
;   0524  3             if .outlim lss 1                    ! If there's no room
;   0525  3             then
;   0526  3                 rtb = 1
;   0527  3             else
;   0528  4                 begin
;   0529  4                 outlim = .outlim - 1;           ! Insert the CR we found
;   0530  4                 rab [rab$w_rsz] = .rab [rab$w_rsz] + 1;
;   0531  4                 ch$wchar_a (%x'd', outptr)
;   0532  4                 end
;   0533  4
;   0534  3         end
;   0535  3
;   0536  1     end;                                        ! of RT11$GET


                                  09FC 00000        .ENTRY  RT11$GET, Save R2,R3,R4,R5,R6,R7,R8,R11     ; 0381
                 5E          10   C2 00002        SUBL2   #16, SP                                     ;
                 57      18  A9   D0 00005        MOVL    24(CBLOCK), VBN                             ; 0431
                 58      1C  A9   3C 00009        MOVZWL  28(CBLOCK), OFFSET                          ; 0432
                 02      1E  AA   91 0000D        CMPB    30(RAB), #2                                 ; 0434
                             08   13 00011        BEQL    1$                                          ;
                 50      1E  A9   3C 00013        MOVZWL  30(CBLOCK), R0                              ; 0436
                        FE1A CF   16 00017        JSB     ADVANCE                                     ;
```

```
                18   A9          57  D0 0001B 1$:   MOVL    VBN, 24(CBLOCK)                  ; 0438
                1C   A9          58  3C 0001F       MOVZWL  OFFSET, 28(CBLOCK)               ; 0439
                          08     AE  D4 00023       CLRL    RTB                              ; 0441
                          22     AA  B4 00026       CLRW    34(RAB)                          ; 0442
                04   AE    20     AA  3C 00029       MOVZWL  32(RAB), OUTLIM                  ; 0443
                     53    24     AA  D0 0002E       MOVL    36(RAB), OUTPTR                  ; 0444
                28   AA          53  D0 00032       MOVL    OUTPTR, 40(RAB)                  ;
                          0C     AE  D4 00036 2$:   CLRL    CR                               ; 0458
                     52          58  D0 00039       MOVL    OFFSET, BITE                     ; 0459
                        FE12     CF  16 0003C       JSB     ACCESS                           ; 0465
                     01          50  E8 00040       BLBS    STAT, 3$                         ; 0467
                                     04 00043       RET                                      ;
                            6846     95 00044 3$:   TSTB    (OFFSET)[BLOCK]                  ; 0471
                                16   12 00047       BNEQ    5$                               ;
                     02    1E     AA  91 00049       CMPB    30(RAB), #2                      ; 0473
                                08   12 0004D       BNEQ    4$                               ;
                     50 0001865C 8F  D0 0004F       MOVL    #99932, R0                       ;
                                     04 00056       RET                                      ;
                     50 0001827A 8F  D0 00057 4$:   MOVL    #98938, R0                       ;
                                     04 0005E       RET                                      ;
                                50   D4 0005F 5$:   CLRL    R0                               ; 0475
                     0D        6246  91 00061       CMPB    (BITE)[BLOCK], #13               ;
                                02   12 00065       BNEQ    6$                               ;
                                50   D6 00067       INCL    R0                               ;
                0C   AE          50  D0 00069 6$:   MOVL    R0, CR                           ;
                     0D          50  E8 0006D       BLBS    R0, 7$                           ;
          000001FF   8F          52  D1 00070       CMPL    BITE, #511                       ; 0476
                                04   14 00077       BGTR    7$                               ;
                                52   D6 00079       INCL    BITE                             ; 0477
                                E2   11 0007B       BRB     5$                               ;
        6E           52          58  C3 0007D 7$:   SUBL3   OFFSET, BITE, STRING_SIZE        ; 0479
                     50          6E  D0 00081       MOVL    STRING_SIZE, R0                  ; 0480
                04   AE          50  D1 00084       CMPL    R0, OUTLIM                       ;
                                04   15 00088       BLEQ    8$                               ;
                     50    04     AE  D0 0008A       MOVL    OUTLIM, R0                       ;
                     5B          50  D0 0008E 8$:   MOVL    R0, LEN                          ;
                     6E    04     AE  D1 00091       CMPL    OUTLIM, STRING_SIZE              ; 0482
                                04   18 00095       BGEQ    9$                               ;
                08   AE          01  D0 00097       MOVL    #1, RTB                          ; 0484
        63        6846           5B  28 0009B 9$:   MOVC3   LEN, (OFFSET)[BLOCK], (OUTPTR)   ; 0486
                04   AE          5B  C2 000A0       SUBL2   LEN, OUTLIM                      ; 0487
                22   AA          5B  A0 000A4       ADDW2   LEN, 34(RAB)                     ; 0488
                     50          6E  D0 000A8       MOVL    STRING_SIZE, R0                  ; 0489
                        FD86     CF  16 000AB       JSB     ADVANCE                          ;
                     59    0C     AE  E9 000AF       BLBC    CR, 13$                          ; 0491
                     50          01  D0 000B3       MOVL    #1, R0                           ; 0496
                        FD7B     CF  16 000B6       JSB     ADVANCE                          ;
                        FD94     CF  16 000BA       JSB     ACCESS                           ; 0502
                     4E          50  E9 000BE       BLBC    STAT, 14$                        ; 0504
                     0A        6846  91 000C1       CMPB    (OFFSET)[BLOCK], #10             ; 0508
                                31   12 000C5       BNEQ    11$                              ;
        50           57    18     A9  C3 000C7       SUBL3   24(CBLOCK), VBN, R0              ; 0512
        50           50          09  78 000CC       ASHL    #9, R0, R0                       ;
                     50          58  C0 000D0       ADDL2   OFFSET, R0                       ;
                     51    1C     A9  3C 000D3       MOVZWL  28(CBLOCK), R1                   ;
                     50          51  C2 000D7       SUBL2   R1, R0                           ;
    1E   A9          50          01  A1 000DA       ADDW3   #1, R0, 30(CBLOCK)               ;
```

```
                    0D       08   AE  E9 000DF          BLBC    RTB, 10$                                  ; 0514
                0C  AA       1E   A9  3C 000E3          MOVZWL  30(CBLOCK), 12(RAB)                       ; 0517
                    50 000181A8   8F  D0 000E8          MOVL    #98728, R0                                ; 0516
                                      04 000EF          RET                                               ;
                    50 00010001   8F  D0 000F0 10$:     MOVL    #65537, R0                                ; 0514
                                      04 000F7          RET                                               ;
                             04   AE  D5 000F8 11$:     TSTL    OUTLIM                                    ; 0524
                                  06  14 000FB          BGTR    12$                                       ;
                08  AE           01  D0 000FD          MOVL    #1, RTB                                   ; 0526
                                  09  11 00101          BRB     13$                                       ;
                             04   AE  D7 00103 12$:     DECL    OUTLIM                                    ; 0529
                             22   AA  B6 00106          INCW    34(RAB)                                   ; 0530
                    83           0D  90 00109          MOVB    #13, (OUTPTR)+                            ; 0531
                                FF27  31 0010C 13$:     BRW     2$                                        ;
                                      04 0010F 14$:     RET                                               ; 0536

; Routine Size:  272 bytes,    Routine Base:  CODE + 01CB


;   0537  1
```

```
;    0539  1      %sbttl 'RT11$READ routine'
;    0540  1
;    0541  1      global routine rt11$read : call_rms =
;    0542  1
;    0543  1      !++
;    0544  1      ! Functional description:
;    0545  1      !       Read virtual blocks of file
;    0546  1      !
;    0547  1      ! Inputs:
;    0548  1      !       none
;    0549  1      !
;    0550  1      ! Implicit inputs:
;    0551  1      !       R9                      CBLOCK pointer
;    0552  1      !       R10                     RAB pointer
;    0553  1      !       various fields in the above-listed data structures
;    0554  1      !
;    0555  1      ! Outputs:
;    0556  1      !       none
;    0557  1      !
;    0558  1      ! Implicit outputs:
;    0559  1      !       RAB [rab$l_rbf]         points to buffer
;    0560  1      !       RAB [rab$w_rsz]         size of buffer
;    0561  1      !
;    0562  1      ! Side Effects:
;    0563  1      !       CBLOCK [ctl$l_nbp]      points to next VBN for sequential READ.
;    0564  1      !
;    0565  1      ! Return codes:
;    0566  1      !       RMS$_NORMAL             OK
;    0567  1      !       RMS$_RER                Read error
;    0568  1      !--
;    0569  1
;    0570  2          begin
;    0571  2
;    0572  2          external register
;    0573  2              cblock = 9 : ref bblock,                ! Map to control block
;    0574  2              rab = 10 : ref bblock;                  ! Map to RAB
;    0575  2
;    0576  2          local
;    0577  2              size,
;    0578  2              stat,
;    0579  2              next;
;    0580  2
;    0581  2          next = .cblock [ctl$l_nbp];                 ! VBN to read
;    0582  2          rab [rab$w_rsz] = size = min (.rab [rab$w_usz],     ! smaller of given size
;    0583  2              (.cblock [ctl$l_eofvbn] - .next)*512);  ! and rest of file size
;    0584  2          cblock [ctl$l_nbp] = .next + (.size + 511)/512;     ! Set next block pointer
;    0585  2          stat = boo$qio (                ! Read the virtual block                    [03]
;    0586  2                          rab [rab$l_rbf] = .rab [rab$l_ubf], !  to specified place    [03]
;    0587  2                          .size,      !  with specified size                         [03]
;    0588  2                          .cblock [ctl$l_startlbn] + .next - 1,       ! logical      [03]
;    0589  2                                                                      ! block        [03]
;    0590  2                          io$_readlblk,       !  logical read                        [03]
;    0591  2                          0,          !  physical addresses                       [03]
;    0592  2                          .cblock [ctl$l_rpb]);       !                           [03]
;    0593  2
;    0594  2          if not .stat
;    0595  2          then
```

```
;   0596  3            begin
;   0597  3            rab [rab$l_stv] = .stat;                  ! Save 'system' return
;   0598  3            return rms$_rer                           ! Return read error
;   0599  2            end;
;   0600  2
;   0601  2        rms$_normal
;   0602  1        end;



                                                  0004 00000          .ENTRY  RT11$READ, Save R2                          ; 0541
                                  52       04   A9  D0 00002          MOVL    4(CBLOCK), NEXT                             ; 0581
                    50       44   A9            52  C3 00006          SUBL3   NEXT, 68(CBLOCK), R0                        ; 0583
                    50            50            09  78 0000B          ASHL    #9, R0, R0                                  ;
                                  51       20   AA  3C 0000F          MOVZWL  32(RAB), R1                                 ;
                                  50            51  D1 00013          CMPL    R1, R0                                      ;
                                                03  15 00016          BLEQ    1$                                          ;
                                  51            50  D0 00018          MOVL    R0, R1                                      ;
                                  50            51  D0 0001B 1$:      MOVL    R1, SIZE                                    ; 0582
                             22   AA            51  B0 0001E          MOVW    R1, 34(RAB)                                 ;
                                  51     01FF   C0  9E 00022          MOVAB   511(R0), R1                                 ; 0584
                                  51 00000200   8F  C6 00027          DIVL2   #512, R1                                    ;
                04   A9           51            52  C1 0002E          ADDL3   NEXT, R1, 4(CBLOCK)                         ;
                                          38   A9  DD 00033          PUSHL   56(CBLOCK)                                  ; 0592
                                  7E            21  7D 00036          MOVQ    #33, -(SP)                                  ; 0585
                    51            52       50   A9  C1 00039          ADDL3   80(CBLOCK), NEXT, R1                        ; 0588
                                          FF   A1  9F 0003E          PUSHAB  -1(R1)                                      ;
                                                50  DD 00041          PUSHL   SIZE                                        ; 0587
                                          24   AA  DD 00043          PUSHL   36(RAB)                                     ; 0586
                             28   AA            6E  D0 00046          MOVL    (SP), 40(RAB)                               ;
                      00000000G   EF            06  FB 0004A          CALLS   #6, BOO$QIO                                 ;
                                  0C            50  E8 00051          BLBS    STAT, 2$                                    ; 0594
                             0C   AA            50  D0 00054          MOVL    STAT, 12(RAB)                               ; 0597
                                  50 0001C0F4   8F  D0 00058          MOVL    #114932, R0                                 ; 0598
                                                    04 0005F          RET                                                 ;
                                  50 00010001   8F  D0 00060 2$:      MOVL    #65537, R0                                  ; 0602
                                                    04 00067          RET                                                 ;

; Routine Size:  104 bytes,    Routine Base:  CODE + 02DB


;   0603  1
;   0604  1      end
;   0605  1
;   0606  0      eludom




;                                  PSECT SUMMARY
;
;       Name                    Bytes                        Attributes
;
```

; CODE 835 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON,NOPIC,ALIGN(2)

Symbol	Type	Defined	Referenced ...										
ACCESS	Routine	131	80f	465c	502c								
ADDRESS_FIRST	Local	269	335=	343=	343.								
ADVANCE	Routine	112	79f	436c	489c	496c							
BBLOCK	Structur	Lib02	155	156	267	268	275	277	343	418	419	573	574
BITE	Local	455	459=	475.	476.	477=	477.	479.					
BLOCK	ExtReg	153	188=	196.	203.	215=	215.	223=					
	GlobReg	422	471a.	475a.	486a.	508a.							
BOO$QIO	ExtRout	93	201c	322c	585c								
CACHE	Bind	159	201.	223.									
CALL_RMS	Linkage	Lib02	81	82	83	229	381	541					
CBLOCK	ExtReg	155	159aa	161a.	165a.	166a.	170a.	188a.	196a.	197a.	206a.	218a=	219a= 220a.
			223a.										
	ExtReg	275	280aa	282a=	283a=	284a.	285a.	327a.	366a=	367a=	368a=		
	ExtReg	418	431a.	432a.	436a.	438a=	439a=	440a=	511a=	512a.	517a.		
	ExtReg	573	581a.	583a.	584a=	588a.	592a.						
CHAR	Local	291	297=	299.	300.								
CR	Local	453	458=	475=	491.								
CTL$L_CACHE1	Macro	Lib02	159										
CTL$L_EOFVBN	Macro	Lib02	161	197	220	367	583						
CTL$L_FILE_PTR	Macro	Lib02	284										
CTL$L_GET	Macro	Lib02	282										
CTL$L_HVBN	Macro	Lib02	166	219									
CTL$L_LVBN	Macro	Lib02	165	218	223								
CTL$L_NBP	Macro	Lib02	581	584									
CTL$L_READ	Macro	Lib02	283										
CTL$L_RPB	Macro	Lib02	206	327	592								
CTL$L_STARTLBN	Macro	Lib02	188	196	368	588							
CTL$L_VBN	Macro	Lib02	431	438	512								
CTL$W_BYTE	Macro	Lib02	432	439	512								
CTL$W_CACHE_SIZE	Macro	Lib02	170										
CTL$W_FILE_LEN	Macro	Lib02	285										
CTL$W_OFFBYTE	Macro	Lib02	366										
CTL$W_REC_SIZE	Macro	Lib02	436	440	511	517							
CTL$W_UNIT	Macro	Lib02	280										
DSC$A_POINTER	Macro	Lib03	306										
DSC$C_S_BLN	Literal	Lib03	267										
DSC$W_LENGTH	Macro	Lib03	305										
ENTRY	Bind	343	345.	347.	350.	351.	352.						
EXTRA_LENGTH	Local	270	334=	335.	343.								
FAB	ExtReg	277	329a=	369a=	370a=	371a=	372a=	373a=	374a=				
FAB$B_FSZ	Macro	Lib03	371										
FAB$B_RAT	Macro	Lib03	372										
FAB$B_RFM	Macro	Lib03	373										
FAB$C_RT11	Literal	Lib02	373										
FAB$L_ALQ	Macro	Lib03	369										
FAB$L_FOP	Macro	Lib03	370										
FAB$L_STV	Macro	Lib03	329										
FAB$M_CR	Literal	Lib03	372										
FAB$M_CTG	Literal	Lib03	370										

Symbol	Type	Defined	Referenced ...
FAB$W_MRS	Macro	Lib03	374
FIL$CVTFILNAM	ExtRout	91	308c
FILE_LENGTH	Local	272	347= 348. 362. 367. 369.
I	Local	190	201.
IO$_READLBLK	Literal	Lib03	204 325 590
JSB_ACC	Linkage	71	80 131
JSB_ADV	Linkage	72	79 112
JSB_CBLOCK	Linkage	Lib02	92
LEN	Local	264	285= 293= 293. 305.
	Local	456	480= 486. 487. 488.
LENGTH	RoutForm	112	119.
NEXT	Local	579	581= 583. 584. 588.
OFFSET	ExtReg	117	119= 119. 121. 124. 125= 125.
	GlobReg	424	432= 439. 459. 471. 479. 486. 508. 512.
OUTLIM	Local	427	443= 480. 482. 487= 487. 524. 529= 529.
OUTPTR	Local	428	444= 486= 486a= 531a= 531= 531.
PTR	Local	263	284= 297. 297= 306.
RAB	ExtReg	156	163a. 182a= 211a=
	ExtReg	276	
	ExtReg	419	434a. 442a= 443a. 444a= 444a. 473a. 488a= 488a. 517a= 530a= 530a.
	ExtReg	574	582a= 582a. 586a= 586a. 597a=
RAB$B_RAC	Macro	Lib03	163 434 473
RAB$C_RFA	Literal	Lib03	163 434 473
RAB$L_RBF	Macro	Lib03	444 586
RAB$L_STV	Macro	Lib03	182 211 517 597
RAB$L_UBF	Macro	Lib03	444 586
RAB$W_RSZ	Macro	Lib03	442 488 530 582
RAB$W_USZ	Macro	Lib03	443 582
RAD50DESC	Local	267	305= 306= 308a
RAD50NAM	Local	266	308a 351. 352.
RMS$ALLOC_CACHE	ExtRout	92	177c
RMS$_ACC	Literal	Lib03	330
RMS$_DME	Literal	Lib03	183
RMS$_EOF	Literal	Lib03	163 473
RMS$_FNF	Literal	Lib03	320
RMS$_FNM	Literal	Lib03	295 310
RMS$_NORMAL	Literal	Lib03	375 520 601
RMS$_RER	Literal	Lib03	212 598
RMS$_RFA	Literal	Lib03	163 473
RMS$_RTB	Literal	Lib03	518
RT$L_FILENAME	Macro	Lib02	351
RT$V_ENDSEG	Macro	Lib02	345
RT$V_PERM	Macro	Lib02	350
RT$W_EXTRALEN	Macro	Lib02	334
RT$W_FILELENGTH	Macro	Lib02	347
RT$W_FILETYPE	Macro	Lib02	352
RT$W_FIRST	Macro	Lib02	335
RT$W_SEGMENT	Macro	Lib02	332
RT$W_STARTBLOCK	Macro	Lib02	333
RT$_FIXED	Literal	Lib02	335 343
RT11$GET	GlobRout	381	82f 282a
RT11$OPEN	GlobRout	229	81f
RT11$READ	GlobRout	541	83f 283a
RTB	Local	429	441= 484= 514. 526=
SEGBLK	Local	268	322a 332. 333. 334. 335a

Symbol	Type	Defined	Referenced ...						
SEGMENT	Local	265	312=	320.	324.	332=			
SIZE	Local	577	582=	584.	587.				
START_BLOCK	Local	271	333=	348=	348.	362=	362.	368.	
STAT	Local	175	177=	179.	182.				
	Local	194	201=	208.	211.				
	Local	318	322=	329.					
	Local	463	465=	467.					
	Local	500	502=	504.					
	Local	578	585=	594.	597.				
STRING_SIZE	Local	454	479=	480.	482.	489.			
UNIT	Bind	280							
VBN	ExtReg	116	124=	124.					
	ExtReg	154	161.	165.	166.	188.	218.	219.	223.
	GlobReg	423	431=	438.	512.				

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]RT11.B32;1
5	Module RT11	
60	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
62	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
64	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
606	Eludom RT11	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
a	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	0	0	96	00:00.0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	31	4	47	00:00.0
SYS$COMMON:[SYSLIB]LIB.L32;2	20450	29	0	1101	00:00.8

```
;                                       COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:RT11.B32

; Size:         835 code + 0 data bytes
; Run Time:        00:03.9
; Elapsed Time:    00:06.2
; Lines/CPU Min:    9299
; Lexemes/CPU-Min: 67074
; Memory Used:  146 pages
; Compilation Complete
```

Table of contents

(1) 157 DECLARATIONS
(1) 219 INITIAL SYSTEM CONTROL BLOCK IMAGE.
(1) 292 DSX$INITSCB INITIALIZE SYSTEM CONTROL BLOCK ROUTINE
(1) 389 DSX$SETVEC SET VECTOR ROUTINE
(1) 522 DSX$CLRVEC RESTORE VECTOR ROUTINE
(1) 600 CHECK_VECTOR Verify vector range
(1) 644 DSX$SETIPL SET INTERRUPT PRIORITY LEVEL SERVICE PROCEDURE.
(1) 677 SOFT$INT Software interrupt vectors

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element SCB.MAR
0000     2 ; *4    26-MAR-1987 10:09:04 MEIER "REMOVED REFERENCES TO LLL & JJJ"
0000     3 ; *3    19-JAN-1987 13:46:32 MEIER "LLL INTERVAL TIMER CHANGE IMPLEMENTED"
0000     4 ; *2     5-MAY-1986 11:35:52 SALEM "DONE"
0000     5 ; *1     6-FEB-1986 15:22:24 DS ""
0000     6 ; DEC/CMS REPLACEMENT HISTORY, Element SCB.MAR
0000     7         .TITLE  SCB     SYSTEM CONTROL BLOCK
0000     8         .IDENT  /07-34/                                                ;G:2
0000     9         .NoShow Conditionals
0000    10
0000    11 ;
0000    12 ; COPYRIGHT (C) 1977, 1981, 1983, 1984, 1985, 1986                     ;G:3
0000    13 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    14 ;
0000    15 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    16 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    17 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    18 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    19 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    20 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    21 ; REMAIN IN DEC.
0000    22 ;
0000    23 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    24 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    25 ; CORPORATION.
0000    26 ;
0000    27 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    28 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    29
0000    30 ;++
0000    31 ; FACILITY:
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;
0000    35 ; ENVIRONMENT:
0000    36 ;
0000    37 ; AUTHOR:        KEN CHAPMAN     20-SEP-77       VERSION 01.
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;                NICK HOWGATE    05-DEC-77       VERSION 02
0000    41 ;        01      ADDED EXCEPTION HANDLER ENHANCEMENT.
0000    42 ;
0000    43 ;                KEN CHAPMAN     01-DEC-77       VERSION 03
0000    44 ;        02      ADD TIMRON FLAG SET AND CLEAR TO $DS_SETVEC_L AND $DS_CLRVEC_L,
0000    45 ;                RESPECTIVELY, FOR INTERVAL TIMER VECTOR.
0000    46 ;
0000    47 ;                KEN CHAPMAN     16-FEB-78       VERSION 04 (ESSAA-3.07).
0000    48 ;        03      GENERAL CODE ENHANCEMENTS.
0000    49 ;        04      ADDED GENERALIZED EXCEPTION HANDLER FROM DEBUG.
0000    50 ;        05      "REI" INSTEAD OF "HALT" ON UNEXPECTED INTERRUPT.
0000    51 ;
0000    52 ;                N. HOWGATE      21-FEB-78       VERSION 05 (ESSAA-4.00).
0000    53 ;        06      ADD QIO SUPPORT - SOFTWARE LEVEL 8 VECTOR, AND MBA0 VECTOR
0000    54 ;                TOM SOUTTER     27-FEB-78
0000    55 ;        07      FIX TO UNEXPECTED VECTOR HANDLER TO CALL CLI
0000    56 ;                TOM SOUTTER     28-FEB-78
0000    57 ;        08      TURNED ON AST DELIVERY VECTOR (SOFTWARE LEVEL 2 INTERRUPT)
```

```
                     0000     58 ;
                     0000     59 ;                  TOM SOUTTER     28-MAR-78       VERSION 06 (ESSAA-4.01).
                     0000     60 ;          09      INSTALLED HOOK TO INITIALIZE SYSTEM TIME IN $DS_CLRVEC_L (TIMER)
                     0000     61 ;          10      SET TIMER VECTOR TO DSI$TIMSRV & TR9/BR5 TO MBA$INTDIS
                     0000     62 ;                  ROGER RIGGS     20-APR-78
                     0000     63 ;          11      CORRECTLY REMOVE UPPER 2 BITS OF SBA ADDRESS ON DISPLAY
                     0000     64 ;          12      Get PC correct and do DMP_SBI and DMP_SILO on SBI errors
                     0000     65 ;                  ROGER RIGGS     15-FEB-79
                     0000     66 ;          13      MODIFIED SETVEC SO LOW 2 BITS OF SCB REFLECT SETVEC BITS
                     0000     67 ;                  EVEN FOR 'SOFT' VECTORS.
                     0000     68 ;                  REMOVED 'IS' BIT FROM MOST EXCEPTIONS
                     0000     69 ;                  Roger Riggs     12-Jun-1979
                     0000     70 ;          14      Made software interrupt vectors soft, like BPT, T-bit...
                     0000     71 ;                  Roger Riggs     1-OCT-1979
                     0000     72 ;          15      Changes to have only 1 SCB instead of 1 per machine
                     0000     73 ;                  Space for SCB is allocated in kernel, initialization
                     0000     74 ;                  is done here.  SCB_IMAGE is reduced to only 1 page.
                     0000     75 ;                  Roger Riggs     6-NOV-1979
                     0000     76 ;          16      Made SETIPL change mode to KERNEL before setting the IPL
                     0000     77 ;                  Roger Riggs, 26-Jan-1980, Version 5.2
                     0000     78 ;          17      Set supervisor vector for software IPL interrupts on
                     0000     79 ;                  levels 6,9,10,11, to dispatch to EXE$FORKDSPTH
                     0000     80 ;                  John Ciukaj, 5-feb-80.    Version 5.2
                     0000     81 ;          18      Delete 'MOVAB   IS(R3),(R2)' causing exception while
                     0000     82 ;                  running diagnostic ECKAX, Exception depended on value
                     0000     83 ;                  of R2 on entry to DS$INITSCB
                     0000     84 ;                  Roger Riggs, 21-Mar-1980
                     0000     85 ;          19      Added EXE$GL_SCB to hold address of current SCB.
                     0000     86 ;                  Roger Riggs, 18-SEP-1980, Version 6.0
                     0000     87 ;          20      Added code to DS$INITSCB to restart the clock
                     0000     88 ;                  Dave Butenhof, 18-feb-1981, version 6.3
                     0000     89 ;          21      Also allow SETVEC w/out changing address
                     0000     90 ;          22      Fix several truncation errors
                     0000     91 ;
                     0000     92 ;          23      - Jack Stansbury, 22-Oct-1981, Version 6.5
                     0000     93 ;                  Fixed several truncation errors. Changed SEP Psects to
                     0000     94 ;                  what they should be. Added .LIBRARY statements for
                     0000     95 ;                  $DS and $DIAG.
                     0000     96 ;
                     0000     97 ;          24      Jack Stansbury, 15-September-1982, Version 6.9
                     0000     98 ;                  Added code in the SetVec routine that will take care of a race
                     0000     99 ;                  condition with the interval clock interrupts and an MTPR
                     0000    100 ;                  instruction that turns off the clock. Also changed the ICCS$M_ERR
                     0000    101 ;                  to be bit 15, instead of bit 31.
                     0000    102 ;
                     0000    103 ;          25      Jack Stansbury, 16-September-1982, Version 6.9
                     0000    104 ;                  Changed the ICCS$M_Err bit back to 31. The VAX Hardware Handbook
                     0000    105 ;                  has a picture of the ICCS longword in the back of the book. This
                     0000    106 ;                  picture states that the ERR bit is bit 15. However, in other
                     0000    107 ;                  sections of the book (and in the SRM), the ERR bit is 15. The
                     0000    108 ;                  NEBULA hardware thinks it is 15 also.
                     0000    109 ;
                     0000    110 ;          26      Bob Bergazzi    21-Oct-1982     version 6.9
                     0000    111 ;                  Changed comments in DSX$SETVEC routine. Also see CLOCK.MAR [25]
                     0000    112 ;
                     0000    113 ;          27      Bob Bergazzi    April 26, 1983  Version 6.11
                     0000    114 ;                  Added call to ONLY_SCB in DSX$INITSCB which fixes the SCB for VENUS.
```

```
0000    115 ;
0000    116 ;       28      Bob Bergazzi    June 7, 1983    Version 6.12
0000    117 ;               Changed the CLRVEC routine to reset the clock vector before
0000    118 ;               resetting the clock. This compensates for the TOK hardware bug
0000    119 ;               on 11/750s, which caused interrupts once in a blue moon when
0000    120 ;               resetting the clock.
0000    121 ;
0000    122 ;       29      Domenic Andella 1-May-1984      Version 7.0
0000    123 ;               Included FAULT_CLEAR selection code macro, and added
0000    124 ;               push/pop of selection code around call to DSR$FAULT_CLEAR
0000    125 ;               in routine DSX$INITSCB. Specific action relating to
0000    126 ;               CPU type and calling routine can then be taken in
0000    127 ;               MCHK___ module.
0000    128 ;
0000    129 ;       30      M. Baggett      27-Aug-1984     Version 7.1
0000    130 ;               Do not access the interval timer registers for AURORA.
0000    131 ;
0000    132 ;       31      Bob Bergazzi    Oct. 15, 1984   Version 7.1
0000    133 ;               Change SETVEC so that vector 0 is legal.
0000    134 ;
0000    135 ;       32      Bob Bergazzi    Nov. 8, 1984    Version 7.1
0000    136 ;               Put UNKNOWN_INT into psect SEP, in order to make it closer
0000    137 ;               to the SCB. The word displacement of the BSBW in SCB_UNKINT
0000    138 ;               had grown larger than a word, since it is a negative displacement.
0000    139 ;
0000    140 ;       33      Bob Bergazzi    Sep. 18, 1985   Version 9.0
0000    141 ;               Changed success return for $DS_CLRVEC from SS$_NORMAL to DS$_NORMAL
0000    142 ;               which is what SETVEC does, and what the documentation says. Symptom
0000    143 ;               was that the channel service for DSINT was returning SS$_NORMAL from
0000    144 ;               the $DS_CLRVEC call when DSINT is also documented as returning
0000    145 ;               DS$_NORMAL.
0000    146 ;                                                                       ;G:2
0000    147 ;       34      Ted Salem       May 4, 1986     Version 10.0            ;G:2
0000    148 ;               Changes fix #30 to check for MicroVax based systems     ;G:2
0000    149 ;                                                                       ;G:2
0000    150 ;       35      Stephen Meier   Sept 22, 1986   Version 10.2            ;G:3
0000    151 ;               Updated reference to the interval clock for LLL;        ;G:4
0000    152 ;               LLL's interval clock is implemented in I/O space rather ;G:4
0000    153 ;               than as IPR's.                                          ;G:3
0000    154 ;                                                                       ;G:3
0000    155 ;--
```

```
                     0000   157          .SBTTL  DECLARATIONS
                     0000   158 ;
                     0000   159 ; INCLUDE FILES:
                     0000   160 ;
                     0000   161          .LIBRARY        /SYS$LIBRARY:LIB/       ;                                       [23]
                     0000   162          .LIBRARY        /$DS/           ;                                               [23]
                     0000   163          .LIBRARY        /$DIAG/         ;                                               [23]
                     0000   164
                     0000   165 ;
                     0000   166 ; MACROS:
                     0000   167 ;
                     0000   168
                     0000   169 ;
                     0000   170 ; EQUATED SYMBOLS:
                     0000   171 ;
                     0000   172
                     0000   173          $DS_CVTREG_DEF
                     0000   174          $DS_DSDEF
                     0000   175          $PRDEF
                     0000               .SAVE   LOCAL_BLOCK
                     0000               .RESTORE
                     0000   176          $PSLDEF
                     0000               .SAVE   LOCAL_BLOCK
                     0000               .RESTORE
                     0000   177          $DS_SCBDEF
                     0000   178          $DS_SETVEC_DEF
                     0000   179          CMKDEF
                     0000   180          DSFDEF
                     0000   181          FLTCLR_SEL                                                                      ;[29
00000001             0000   182 IS        = 1
                     0000   183
80000000             0000   184 ICCS$M_ERR       = 1 @ 31                ; ERROR INDICATOR IN PR$_ICCS                   [25]
00000080             0000   185 ICCS$M_INT       = 1 @ 7                 ; INTERRUPT PENDING IN $PR$_ICCS        ;G:3
                     0000   186                                                                                     ;G:3
                     0000   187 .WEAK    CLK$K_TCR0                      ; Location of LLL clock register        ;G:4
                     0000   188                                         ; defined in CLOCKLLL [35]              ;G:3
```

```
                        00000000   190          .PSECT  Work, NoExe, NoShr, Wrt, Long   ;                                  [23]
                            0000   191
                            0000   192 ;
                            0000   193 ; OWN STORAGE:
                            0000   194 ;
                            0000   195
                            0000   196 DS$GB_Stopping_The_Timer::      ; If = 1, we are doing an MTPR to stop the      [24]
                        00  0000   197          .Byte   0               ; . . . interval clock.                         [24]
                            0001   198
                            0001   199 DS$GA_CHMKVEC::
                   00000005 0001   200          .BLKL   1               ; USER CHANGE MODE TO KERNEL VECTOR.
                            0005   201
                            0005   202 DS$GA_BREAKVEC::
                   00000009 0005   203          .BLKL   1               ; USER BREAK INSTRUCTION FAULT VECTOR.      ;G:3
                            0009   204
                            0009   205 DS$GA_TBITVEC::
                   0000000D 0009   206          .BLKL   1               ; USER T-BIT TRAP VECTOR.
                            000D   207
                            000D   208 EXE$GL_SCB::
                   00000000' 000D  209          .LONG   SCB_IMAGE       ; Virtual address of SCB
                            0011   210
                            0011   211 DS$GA_SOFTINT::
                   00000000 0011   212          .LONG   0               ; 1 Bit per level 1-15(D)
00000000'00000000'00000000'00000000' 0015   213          .LONG   0[15]           ; Soft interrupt vectors, 0=not assigned
00000000'00000000'00000000'00000000' 0025
00000000'00000000'00000000'00000000' 0035
          00000000'00000000'00000000' 0045
```

```
                        00000000   215          .PSECT  Data, NoExe, Shr, NoWrt, Long   ;                              [23]
                            0000   216
                            0000   217          MODNAM  SCB
                  42 43 53 00'  0000       $MODULE:        .ASCIC  \SCB\
                              03  0000
```

```
                     0004   219          .SBTTL  INITIAL SYSTEM CONTROL BLOCK IMAGE.
                     0004   220 ;+
                     0004   221 ; IMAGE OF THE SYSTEM CONTROL BLOCK USED TO REINITIALIZE THE ACTUAL EXE.
                     0004   222 ;-
                     0004   223
                 00000000   224          .PSECT  $SCB, SHR, EXE, WRT, PAGE
                     0000   225
                     0000   226 SCB_IMAGE::
00000000'            0000   227                  .LONG   SCBVECTOR_000   ; UNSPECIFIED. RESERVED TO DEC.
00000001'            0004   228 SCB_MCHK:        .LONG   EXE_MCHK+IS     ; MACHINE CHECK ABORT.
00000001'            0008   229 SCB_KRNLSTK:     .LONG   EXE_KRNLSTK+IS  ; KERNEL STACK NOT VALID ABORT.
00000001'            000C   230 SCB_POWER:       .LONG   EXE_POWER+IS    ; POWER FAIL INTERRUPT.
00000000'            0010   231 SCB_OPCDEC:      .LONG   EXE_OPCDEC      ; RESERVED/PRIVILEGED INSTRUCTIONS FAULT.
00000000'            0014   232 SCB_OPCCUS:      .LONG   EXE_OPCCUS      ; CUSTOMER RESERVED INSTRUCTION FAULT.
00000000'            0018   233 SCB_ROPRAND:     .LONG   EXE_ROPRAND     ; RESERVED OPERAND FAULT/ABORT.
00000000'            001C   234 SCB_RADRMOD:     .LONG   EXE_RADRMOD     ; RESERVED ADDRESSING MODE FAULT.
00000000'            0020   235 SCB_ACCVIO:      .LONG   EXE_ACCVIO      ; ACCESS CONTROL VIOLATION FAULT.
00000000'            0024   236 SCB_TRANSL:      .LONG   EXE_TRANSL      ; TRANSLATION NOT VALID FAULT.
00000000'            0028   237 SCB_TBIT:        .LONG   EXE_TBIT        ; TRACE TRAP.
00000000'            002C   238 SCB_BREAK:       .LONG   EXE_BREAK       ; BREAKPOINT FAULT.
00000000'            0030   239 SCB_COMPAT:      .LONG   EXE_COMPAT      ; COMPATIBILITY TRAP.
00000000'            0034   240 SCB_ARITH:       .LONG   EXE_ARITH       ; ARITHMETIC TRAP.
00000000'            0038   241                  .LONG   SCBVECTOR_038   ; UNSPECIFIED.
00000000'            003C   242                  .LONG   SCBVECTOR_03C   ; UNSPECIFIED.
00000000'            0040   243 SCB_CHMK:        .LONG   EXE_CMODKRNL    ; CHMK TRAP.
00000000'            0044   244 SCB_CHME:        .LONG   EXE_CHME        ; CHME TRAP.
00000000'            0048   245 SCB_CHMS:        .LONG   EXE_CHMS        ; CHMS TRAP.
00000000'            004C   246 SCB_CHMU:        .LONG   EXE_CHMU        ; CHMU TRAP.
00000000'            0050   247                  .LONG   SCBVECTOR_050   ; UNSPECIFIED.
00000000'            0054   248                  .LONG   SCBVECTOR_054   ; UNSPECIFIED.
00000000'            0058   249                  .LONG   SCBVECTOR_058   ; UNSPECIFIED.
00000000'            005C   250                  .LONG   SCBVECTOR_05C   ; UNSPECIFIED.
00000000'            0060   251                  .LONG   SCBVECTOR_060   ; UNSPECIFIED.
00000000'            0064   252                  .LONG   SCBVECTOR_064   ; UNSPECIFIED.
00000000'            0068   253                  .LONG   SCBVECTOR_068   ; UNSPECIFIED.
00000000'            006C   254                  .LONG   SCBVECTOR_06C   ; UNSPECIFIED.
00000000'            0070   255                  .LONG   SCBVECTOR_070   ; UNSPECIFIED.
00000000'            0074   256                  .LONG   SCBVECTOR_074   ; UNSPECIFIED.
00000000'            0078   257                  .LONG   SCBVECTOR_078   ; UNSPECIFIED.
00000000'            007C   258                  .LONG   SCBVECTOR_07C   ; UNSPECIFIED.
00000000'            0080   259                  .LONG   SCBVECTOR_080   ; UNSPECIFIED.
000002B4'            0084   260 SCB_SFTLVL1:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 1 INTERRUPT.
000002B4'            0088   261 SCB_SFTLVL2:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 2 INTERRUPT.
000002B4'            008C   262 SCB_SFTLVL3:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 3 INTERRUPT.
000002B4'            0090   263 SCB_SFTLVL4:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 4 INTERRUPT.
000002B4'            0094   264 SCB_SFTLVL5:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 5 INTERRUPT.
000002B4'            0098   265 SCB_SFTLVL6:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 6 INTERRUPT.
000002B4'            009C   266 SCB_SFTLVL7:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 7 INTERRUPT.
000002B4'            00A0   267 SCB_SFTLVL8:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 8 INTERRUPT.
000002B4'            00A4   268 SCB_SFTLVL9:     .LONG   SOFT$INT        ; SOFTWARE LEVEL 9 INTERRUPT.
000002B4'            00A8   269 SCB_SFTLVL10:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 10 INTERRUPT.
000002B4'            00AC   270 SCB_SFTLVL11:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 11 INTERRUPT.
000002B4'            00B0   271 SCB_SFTLVL12:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 12 INTERRUPT.
000002B4'            00B4   272 SCB_SFTLVL13:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 13 INTERRUPT.
000002B4'            00B8   273 SCB_SFTLVL14:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 14 INTERRUPT.
000002B4'            00BC   274 SCB_SFTLVL15:    .LONG   SOFT$INT        ; SOFTWARE LEVEL 15 INTERRUPT.
00000001'            00C0   275 SCB_TIMER:       .LONG   DSI$TIMSRV+IS   ; INTERVAL TIMER INTERRUPT.
```

```
00000000' 00C4   276                .LONG   SCBVECTOR_0C4   ; UNSPECIFIED.
00000000' 00C8   277                .LONG   SCBVECTOR_0C8   ; UNSPECIFIED.
00000000' 00CC   278                .LONG   SCBVECTOR_0CC   ; UNSPECIFIED.
00000000' 00D0   279                .LONG   SCBVECTOR_0D0   ; UNSPECIFIED.
00000000' 00D4   280                .LONG   SCBVECTOR_0D4   ; UNSPECIFIED.
00000000' 00D8   281                .LONG   SCBVECTOR_0D8   ; UNSPECIFIED.
00000000' 00DC   282                .LONG   SCBVECTOR_0DC   ; UNSPECIFIED.
00000000' 00E0   283                .LONG   SCBVECTOR_0E0   ; UNSPECIFIED.
00000000' 00E4   284                .LONG   SCBVECTOR_0E4   ; UNSPECIFIED.
00000000' 00E8   285                .LONG   SCBVECTOR_0E8   ; UNSPECIFIED.
00000000' 00EC   286                .LONG   SCBVECTOR_0EC   ; UNSPECIFIED.
00000000' 00F0   287                .LONG   SCBVECTOR_0F0   ; UNSPECIFIED.
00000000' 00F4   288                .LONG   SCBVECTOR_0F4   ; UNSPECIFIED.
00000000' 00F8   289                .LONG   SCBVECTOR_0F8   ; UNSPECIFIED.
00000000' 00FC   290                .LONG   SCBVECTOR_0FC   ; UNSPECIFIED.
```

```
                     0100   292          .SBTTL  DSX$INITSCB     INITIALIZE SYSTEM CONTROL BLOCK ROUTINE
                 00000000   293          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                          [23]
                     0000   294
                     0000   295 ;++
                     0000   296 ; FUNCTIONAL DESCRIPTION:
                     0000   297 ;
                     0000   298 ;       This routine generates the necessary SCB.
                     0000   299 ;       The 1st half page contains vectors for exceptions and traps
                     0000   300 ;       The rest of the 4 pages is filled with trap catchers.
                     0000   301 ;       Interrupts are disabled while the SCB is being updated.
                     0000   302 ;       Calls the ONLY_SCB routine to do CPU specific SCB setup.
                     0000   303 ;
                     0000   304 ; CALLING SEQUENCE:
                     0000   305 ;
                     0000   306 ;       DS$INITSCB()
                     0000   307 ;
                     0000   308 ; INPUT PARAMETERS:     NONE
                     0000   309 ;
                     0000   310 ; IMPLICIT INPUTS:      NONE
                     0000   311 ;
                     0000   312 ; OUTPUT PARAMETERS:    NONE
                     0000   313 ;
                     0000   314 ; IMPLICIT OUTPUTS:     NONE
                     0000   315 ;
                     0000   316 ; COMPLETION CODES:     NONE
                     0000   317 ;
                     0000   318 ; SIDE EFFECTS:         NONE
                     0000   319 ;--
```

```
                     000C  0000    321 .ENTRY  DSX$INITSCB,^M<R2,R3>
                           0002    322
                           0002    323         CMK     INITSCB                 ; CHANGE MODE TO KERNEL.
                 7E   DC   0002                MOVPSL  -(SP)
              06 6E   1A   E0   0004            BBS     #PSL$V_IS, (SP), 30000$
                 8E   D4   0008                CLRL    (SP)+
                 08   BC   000A                CHMK    #CMK$_INITSCB
                 06   11   000C                BRB     30001$
        00000026'EF   16   000E        30000$: JSB     CMK$INITSCB
                           0014        30001$:
                           0014    324
           50    00'  D0   0014    325         MOVL    S^#SS$_NORMAL,R0        ; Set success
     00000000'EF   11   E5   0017    326         BBCC    #DS$V_TIMRON,DS$GL_FLAGS,- ; Clear interval timer in use
                 00         001E    327                 10$
        00000000'EF   16   001F    328 10$:    JSB     KB_CHECK                ; CHECK KEYBOARD STATUS.
                      04   0025    329         RET                             ; RETURN TO CALLING ROUTINE
                           0026    330
                           0026    331 ;
                           0026    332 ; Reset the whole SCB and reset the SCBB
                           0026    333 ;
                           0026    334
                           0026    335 CMK$INITSCB::
                 0F   BB   0026    336         PUSHR   #^M<R0,R1,R2,R3>        ; Save registers
                           0028    337         DSBINT                          ; Disable interrupts
              7E   12   DB   0028                MFPR    S^#PR$_IPL,-(SP)
              12   1F   DA   002B                MTPR    #31,S^#PR$_IPL
                 51   D4   002E    338         CLRL    R1                      ; Clear count of quads to be copied
        52 00000000'EF   9E   0030    339         MOVAB   L^SCB_BASE,R2           ; Point to where to build SCB
        53 00000000'EF   9E   0037    340         MOVAB   SCB_IMAGE,R3            ; 1/2 page of SCB image
                           003E    341
              82   83   7D   003E    342 10$:    MOVQ    (R3)+,(R2)+             ; Copy quadword of SCB
           F9 51   1F   F3   0041    343         AOBLEQ  #31,R1,10$              ; Copy 1/2 page
                           0045    344
        53 00000100'EF   9E   0045    345         MOVAB   SCB_UNKINT+^X100,R3     ; Point to unknown interrupt field
        51 00000000'EF   9E   004C    346         MOVAB   L^UNKNOWN_INT,R1        ; Address of intermediate handler       [23]
                           0053    347
           50    03 A3   9E   0053    348 20$:    MOVAB   3(R3),R0                ; Address after BSBW inst
        50   51   50   C3   0057    349         SUBL3   R0,R1,R0                ; make offset to handler
        50   50   08   78   005B    350         ASHL    #8,R0,R0                ; Position offset
             50   30   90   005F    351         MOVB    #^X30,R0                ; Insert opcode for BSBW
             83   50   D0   0062    352         MOVL    R0,(R3)+                ; Insert BSBW UNKNOWN_INT
           62   FD A3   9E   0065    353         MOVAB   IS-4(R3),(R2)           ; Point SCB vector to BSBW+IS
FFE0 52    04   FFFFFFFC'8F   F1   0069    354         ACBL    #SCB_UNKINT-4,#4,R2,20$ ; For every vector in the SCB
                           0073    355
        00000000'EF   16   0073    356         JSB     ONLY_SCB                ; Do CPU specific SCB setup             [27]
                           0079    357
                           0079    358 ;+
                           0079    359 ; Re-init clock and flag
                           0079    360 ;-
                           0079    361
        00000000'EF   16   0079    362         JSB     L^CLK$RE_INIT           ; Initialize the system timer
     00000000'EF   11   E5   007F    363         BBCC    #DS$V_TIMRON, -         ; RELEASE THE INTERVAL CLOCK.
                      00   0086    364                   DS$GL_FLAGS, 25$
                           0087    365
        00000001'EF   D4   0087    366 25$:    CLRL    DS$GA_CHMKVEC           ; CLEAR USER CHMK VECTOR.
        00000005'EF   D4   008D    367         CLRL    DS$GA_BREAKVEC          ; CLEAR USER BREAK VECTOR.
        00000009'EF   D4   0093    368         CLRL    DS$GA_TBITVEC           ; CLEAR USER TBIT VECTOR.
```

```
                50   07   D0  0099   369          MOVL    #7,R0                      ; Max software interrupt level
                                009C   370
        00000011'EF40    7C  009C   371 30$:     CLRQ    DS$GA_SOFTINT[R0]          ; Clear soft vector
                  F6 50  F5  00A3   372          SOBGTR  R0,30$                     ; Clear all vectors including 0 (mask)
   50   00000000'EF      DE  00A6   373          MOVAL   SCB_BASE,R0                ; Point to SCB
                11   50  DA  00AD   374          MTPR    R0,#PR$_SCBB               ; Set SCBB
   0000000D'EF       50  D0  00B0   375          MOVL    R0,EXE$GL_SCB              ; Set Virtual address of SCB
                     02  DD  00B7   376          PUSHL   #DS$_GK_INITSCB            ; Pass fault clear selection code        ;[29
        00000000'EF      16  00B9   377          JSB     L^DSR$FAULT_CLEAR          ; Reset fault regs and enables
                     8E  D5  00BF   378          TSTL    (SP)+                      ; Reset the stack
                                00C1   379          ENBINT                             ; Re-enable interrupts
                12   8E  DA  00C1                       MTPR    (SP)+,S^#PR$_IPL
                     0F  BA  00C4   380          POPR    #^M<R0,R1,R2,R3>           ; Restore registers
                         02  00C6   381          REI
                                00C7   382
                         00000000   383 .PSECT   SEP,    SHR,    EXE,    WRT,    LONG                                ;        [32]
                                0000   384 UNKNOWN_INT:
   6E   00000003'8F      C2  0000   385          SUBL    #SCB_UNKINT+3,(SP)         ; BSBW addr-base is SCB offset
        00000000'EF      16  0007   386          JSB     DSI$CHANNEL_VEC            ; Enter channel services w/ vector       [23]
        00000000'EF      17  000D   387          JMP     EXE_UNEXPINT               ; With offset on stack flag unexpected   [23]
```

```
                        0013   389          .SBTTL  DSX$SETVEC      SET VECTOR ROUTINE
                        0013   390 ;++
                        0013   391 ; FUNCTIONAL DESCRIPTION:
                        0013   392 ;
                        0013   393 ;       ROUTINE TO LOAD THE ADDRESS OF AN EXCEPTION OR INTERRUPT SERVICE ROUTINE
                        0013   394 ;       INTO THE SYSTEM CONTROL BLOCK (SCB).
                        0013   395 ;
                        0013   396 ; CALLING SEQUENCE:
                        0013   397 ;
                        0013   398 ;       STANDARD CALL PROCEDURE.
                        0013   399 ;
                        0013   400 ; INPUT PARAMETERS:
                        0013   401 ;
                        0013   402 ;       SETVEC$_VECTOR(AP)      = OFFSET INTO SCB.
                        0013   403 ;       SETVEC$_SRVADR(AP)      = ADDRESS OF INTERRUPT SERVICE ROUTINE
                        0013   404 ;       SETVEC$_CODE(AP)        = VECTOR CODE IN BITS 1:0.
                        0013   405 ;
                        0013   406 ; IMPLICIT INPUTS:      NONE
                        0013   407 ;
                        0013   408 ; OUTPUT PARAMETERS:
                        0013   409 ;
                        0013   410 ;       R0      = COMPLETION CODE.
                        0013   411 ;       R1      = PREVIOUS SERVICE ADDRESS.
                        0013   412 ;
                        0013   413 ; IMPLICIT OUTPUTS:      NONE
                        0013   414 ;
                        0013   415 ; COMPLETION CODES:
                        0013   416 ;
                        0013   417 ;       DS$_NORMAL      = VECTOR MODIFIED.
                        0013   418 ;       DS$_IVADDR      = SERVICE ADDRESS <1:0> NOT ZERO.
                        0013   419 ;       DS$_IVVECT      = INVALID VECTOR NUMBER.
                        0013   420 ;       DS$_ICBUSY      = INTERVAL CLOCK BUSY.
                        0013   421 ;
                        0013   422 ; SIDE EFFECTS:         NONE
                        0013   423 ;--
```

```
                          000000C7          .PSECT  Code, Exe, Shr, NoWrt, Long     ;                                  [23]
                        000C  00C7  .ENTRY  DSX$SETVEC,^M<R2,R3>              ; Save R2 and R3                          [24]
                              00C9
                  0163    30  00C9          BSBW    CHECK_VECTOR              ; Check vector for validity
                              00CC
      50   00660040 8F    D0  00CC          MOVL    #DS$_IVADDR, R0           ; SET ERROR RETURN CODE.
            08 AC   03    D3  00D3          BITL    #3,8(AP)                  ; CHECK FOR LONGWORD ALIGNED ADR.
                    4A    12  00D7          BNEQ    53$                       ; Exit if not longword aligned
                              00D9
      51   00000000'E2    DE  00D9          MOVAL   L^SCB_BASE(R2), R1        ; GET ADR OF VECTOR AS I KNOW IT.
                              00E0                                                                                        ;G:3
          0040 8F   52    B1  00E0          CMPW    R2, #SCB$L_CHMK           ; CHECK FOR CHMK VECTOR.
                    0A    12  00E5          BNEQ    30$
      51   00000001'EF    DE  00E7          MOVAL   DS$GA_CHMKVEC, R1         ; SET SOFT VECTOR INSTEAD.
                  008D    31  00EE          BRW     100$                                                                ;G:3
                              00F1
              2C   52    B1  00F1  30$:    CMPW    R2, #SCB$L_BREAK          ; CHECK FOR BREAK VECTOR.
                    09    12  00F4          BNEQ    40$
      51   00000005'EF    DE  00F6          MOVAL   DS$GA_BREAKVEC, R1        ; SET SOFT VECTOR INSTEAD.
                    7F    11  00FD          BRB     100$
                              00FF
              28   52    B1  00FF  40$:    CMPW    R2, #SCB$L_TBIT           ; CHECK FOR TBIT VECTOR.
                    09    12  0102          BNEQ    50$
      51   00000009'EF    DE  0104          MOVAL   DS$GA_TBITVEC, R1         ; SET SOFT VECTOR INSTEAD.
                    71    11  010B          BRB     100$
                              010D
          00C0 8F   52    B1  010D  50$:    CMPW    R2, #SCB$L_TIMER          ; CHECK FOR TIMER VECTOR.
                    4D    12  0112          BNEQ    60$
      00000000'EF   11    E3  0114          BBCS    #DS$V_TIMRON, -           ; SET TIMER FLAG.
                    09        011B                    DS$GL_FLAGS, 55$
      50   006600C8 8F    D0  011C          MOVL    #DS$_ICBUSY, R0           ; INTERVAL CLOCK BUSY.
                              0123
                    7A    11  0123  53$:    BRB     DSX$SETVEC_X
                              0125
      00000000'EF   08    91  0125  55$:    CMPB    #PR$_SID_TYPUV2, EXE$GB_CPUTYPE ; Is it A MicroVax chip?[30][xx] ;G:
                    50    13  012C          BEQL    100$                            ; If yes do not access registers[30]
      00000000'EF   01    90  012E          MovB    #1, L^DS$GB_Stopping_The_Timer  ; Set flag to signal stopping   [24]
                              0135          ;+                                                                          ;G:3
                              0135          ; If this is an LLL machine, the interval clock is mapped into I/O        ;G:4
                              0135          ; space, hence, the clock must be disabled there.                          ;G:3
                              0135          ;-                                                                          ;G:3
              50   3E    DB  0135          MFPR    #PR$_SID,R0                     ; get system ID          [35]  ;G:3
      50   50   E8 8F    78  0138          ASHL    #-24,R0,R0                      ; get upper byte         [35]  ;G:3
              50   0A    91  013D          CMPB    #10,R0                          ; Is this an LLL??       [35]  ;G:4
                    0D    12  0140          BNEQ    57$                             ; not LLL                [35]  ;G:4
  00000000'EF   80000080 8F  D0  0142      MOVL    #ICCS$M_ERR + -                 ; CLEAR ERROR &                ;G:3
                              014D                  ICCS$M_INT , -                  ; INTERRUPT & STOP THE CLOCK   ;G:3
                              014D                  CLK$K_TCR0                      ; IN THE INTERVAL TIMER [35]   ;G:3
                    0A    11  014D          BRB     58$                             ; continue               [35]  ;G:3
                              014F          ;+
                              014F          ; Do an Mtpr and then an Mfpr because the interval clock may interrupt in
                              014F          ; between the time the MTPR console code starts executing and when the MTPR
                              014F          ; console code is actually finished. That is, the Mtpr console code may
                              014F          ; actually be executing at the same time as the MFPR macro code starts.
                              014F          ;-
                              014F
      18   80000080 8F    DA  014F  57$:    MTPR    #ICCS$M_ERR + -           ; CLEAR ERROR &                           ;G:3
```

```
                                 0156   482                  ICCS$M_INT , -          ; INTERRUPT & STOP THE CLOCK
                                 0156   483                #PR$_ICCS                 ; IN THE INTERVAL TIMER
                                 0156   484
                  53   18   DB  0156   485          Mfpr    #PR$_ICCS, R3           ; This ensures that the preceding      [25]
                                 0159   486                                          ; MTPR is really finished (at the      [25]
                                 0159   487                                          ; console level)                       [25]
                                 0159   488
        00000000'EF   94   DB    0159   489 58$:     ClrB    L^DS$GB_Stopping_The_Timer      ; Clear flag to signal done    [24]
                      1D   11    015F   490          BRB     100$
                                 0161   491
              0084 8F   52   B1  0161   492 60$:     CMPW    R2,#SCB$L_SFTLVL1       ; Possible software interrupt vector
                      16   19    0166   493          BLSS    70$                     ; No, desired vector is too low
              00BC 8F   52   B1  0168   494          CMPW    R2,#SCB$L_SFTLVL15      ; Possible software interrupt vector
                      0F   14    016D   495          BGTR    70$                     ; No, desired vector is too high
           51   52   04   02    EF   016F   496          EXTZV   #2,#4,R2,R1             ; Isolate software interrupt level
        51   00000011'EF41    DE   0174   497          MOVAL   DS$GA_SOFTINT[R1],R1    ; Point to true vector
                      00   11    017C   498          BRB     100$
                                 017E   499
                                 017E   500 70$:
                                 017E   501
                                 017E   502 ;+
                                 017E   503 ; Vector offset in R2
                                 017E   504 ; Address of vector in R1
                                 017E   505 ;-
                                 017E   506
                      61   DD    017E   507 100$:    PUSHL   (R1)                    ; SAVE OLD SERVICE ADDRESS.
                   08 AC   D5    0180   508          TSTL    8(AP)                   ; See if address is 0                  [21]
                      04   13    0183   509          BEQL    110$                    ; If so, don't change                  [21]
                61    08 AC   D0    0185   510          MOVL    8(AP), (R1)             ; LOAD THE VECTOR REQUESTED
                                 0189   511
                                 0189   512 110$:                                   ;                                      [21]
           51   00000000'E2   DE    0189   513          MOVAL   L^SCB_BASE(R2), R1      ; Set back to SCB vec (soft vector)    [21]
     61    02    00    0C AC   F0    0190   514          INSV    12(AP), #0, #2, (R1)    ; SET VECTOR CODE BITS 1:0.
                      02   BA    0196   515          POPR    #^M<R1>                 ; PUT OLD SERVICE ADDRESS INTO R1.
           50   00660001 8F   D0    0198   516          MOVL    #DS$_NORMAL, R0         ; SET SUCCESSFUL RETURN
                                 019F   517
                                 019F   518 DSX$SETVEC_X:
                00000000'EF   16    019F   519          JSB     KB_CHECK                ; CHECK KEYBOARD STATUS.
                      04   01A5   520          RET                                     ; RETURN TO THE DIAGNOSTIC
```

```
                        01A6    522          .SBTTL  DSX$CLRVEC      RESTORE VECTOR ROUTINE
                        01A6    523 ;++
                        01A6    524 ; FUNCTIONAL DESCRIPTION:
                        01A6    525 ;
                        01A6    526 ; THIS ROUTINE LOADS THE VECTOR IN THE 'SCB' WITH THE CONTENTS
                        01A6    527 ; OF THE CORRESPONDING VECTOR IN 'SCB_IMAGE' (A PAGE WHICH HOLDS THE
                        01A6    528 ; INITIAL CONTENTS OF EACH VECTOR).
                        01A6    529 ; Interrupts are disabled while the SCB is being changed.
                        01A6    530 ;
                        01A6    531 ; CALLING SEQUENCE:
                        01A6    532 ;
                        01A6    533 ;       DS$CLRVEC(VECTOR OFFSET)
                        01A6    534 ;
                        01A6    535 ; INPUT PARAMETERS:
                        01A6    536 ;
                        01A6    537 ;       4(AP) OFFSET INTO S.C.B.
                        01A6    538 ;
                        01A6    539 ; IMPLICIT INPUTS:
                        01A6    540 ;
                        01A6    541 ;       SCB_IMAGE - INITIAL SYSTEM CONTROL BLOCK
                        01A6    542 ;
                        01A6    543 ; OUTPUT PARAMETERS:
                        01A6    544 ;
                        01A6    545 ;       R0 = 1 > SUCCESSFUL
                        01A6    546 ;       R0 = 0 > FAILURE
                        01A6    547 ;
                        01A6    548 ; IMPLICIT OUTPUTS:     NONE
                        01A6    549 ;
                        01A6    550 ; COMPLETION CODES:     NONE
                        01A6    551 ;
                        01A6    552 ; SIDE EFFECTS:         NONE
                        01A6    553 ;--
```

```
                     0004  01A6   555 .ENTRY  DSX$CLRVEC,^M<R2>
                           01A8   556
                     0084   30  01A8   557         BSBW    CHECK_VECTOR            ; Verify vector
                           01AB   558
           0040 8F    52   B1  01AB   559 10$:     CMPW    R2, #SCB$L_CHMK         ; CHECK FOR CHMK VECTOR.
                        08   12  01B0   560         BNEQ    20$
            00000001'EF     D4  01B2   561         CLRL    DS$GA_CHMKVEC           ; CLEAR SOFT VECTOR.
                        36   11  01B8   562         BRB     100$
                           01BA   563
                2C    52   B1  01BA   564 20$:     CMPW    R2, #SCB$L_BREAK        ; CHECK FOR BREAK VECTOR.
                        08   12  01BD   565         BNEQ    30$
            00000005'EF     D4  01BF   566         CLRL    DS$GA_BREAKVEC          ; CLEAR SOFT VECTOR.
                        29   11  01C5   567         BRB     100$
                           01C7   568
                28    52   B1  01C7   569 30$:     CMPW    R2, #SCB$L_TBIT         ; CHECK FOR TBIT VECTOR.
                        08   12  01CA   570         BNEQ    50$
            00000009'EF     D4  01CC   571         CLRL    DS$GA_TBITVEC           ; CLEAR SOFT VECTOR.
                        1C   11  01D2   572         BRB     100$
                           01D4   573
           0084 8F    52   B1  01D4   574 50$:     CMPW    R2,#SCB$L_SFTLVL1       ; Possible software interrupt vector
                        15   19  01D9   575         BLSS    60$                     ; No, desired vector is too low
           00BC 8F    52   B1  01DB   576         CMPW    R2,#SCB$L_SFTLVL15      ; Possible software interrupt vector
                        0E   14  01E0   577         BGTR    60$                     ; No, desired vector is too high
      50   52   04    02   EF  01E2   578         EXTZV   #2,#4,R2,R0             ; Isolate software interrupt level
          00000011'EF40   D4  01E7   579         CLRL    DS$GA_SOFTINT[R0]       ; Clear vector
                        00   11  01EE   580         BRB     100$
                           01F0   581
                           01F0   582 60$:
      50    00000001'E2   9E  01F0   583 100$:    MOVAB   L^SCB_UNKINT+IS(R2),R0  ; Restore unexpected handler
          0100 8F    52   B1  01F7   584         CMPW    R2,#^X100               ; 1st 1/2 page?
                        07   18  01FC   585         BGEQ    120$                    ; Branch if not
      50    00000000'E2   D0  01FE   586         MOVL    L^SCB_IMAGE(R2),R0      ; Get static vector
                           0205   587
00000000'E2    50    D0  0205   588 120$:    MOVL    R0,L^SCB_BASE(R2)       ; RESTORE THE VECTOR.
          00C0 8F    52   B1  020C   589         CMPW    R2, #SCB$L_TIMER        ; TIMER VECTOR?                 [28]
                        0E   12  0211   590         BNEQ    150$                    ; No                            [28]
            00000000'EF     16  0213   591         JSB     L^CLK$RE_INIT           ; Yes, initialize the system timer [28]
00000000'EF    11   E5  0219   592         BBCC    #DS$V_TIMRON, -         ; RELEASE THE INTERVAL CLOCK.   [28]
                        00   0220   593                   DS$GL_FLAGS, 150$ ;                               [28]
      50    00660001 8F   D0  0221   594 150$:    MOVL    #DS$_NORMAL, R0         ; SET SUCCESSFUL RETURN         [33]
                           0228   595
                           0228   596 DSX$CLRVEC_X:
            00000000'EF     16  0228   597         JSB     KB_CHECK                ; CHECK KEYBOARD.
                        04   022E   598         RET                             ; RETURN TO DIAGNOSTIC PROGRAM
```

```
                        022F    600          .SBTTL  CHECK_VECTOR    Verify vector range
                        022F    601 ;++
                        022F    602 ; FUNCTIONAL DESCRIPTION:
                        022F    603 ;
                        022F    604 ;       This routine is used to verify that the vector specified
                        022F    605 ;       falls within the SCB and is aligned properly
                        022F    606 ;
                        022F    607 ; CALLING SEQUENCE:
                        022F    608 ;
                        022F    609 ;       BSBW    CHECK_VECTOR
                        022F    610 ;
                        022F    611 ; INPUT PARAMETERS:
                        022F    612 ;
                        022F    613 ;       4(AP)   Vector
                        022F    614 ;
                        022F    615 ; IMPLICIT INPUTS:      NONE
                        022F    616 ;
                        022F    617 ; IMPLICIT INPUTS:      NONE
                        022F    618 ;
                        022F    619 ; OUTPUT PARAMETERS:    NONE
                        022F    620 ;
                        022F    621 ; IMPLICIT OUTPUTS:
                        022F    622 ;
                        022F    623 ;       R2      correct vector
                        022F    624 ;
                        022F    625 ; SIDE EFFECTS:
                        022F    626 ;
                        022F    627 ;       Returns to caller if vector bad
                        022F    628 ;
                        022F    629 ; COMPLETION CODES:
                        022F    630 ;
                        022F    631 ;--
```

```
                                 022F   633 CHECK_VECTOR:
          52    04 AC    D0     022F   634          MOVL    4(AP),R2                  ; Get vector
    52    00000000'8F    D1     0233   635          CMPL    #SCB_UNKINT-SCB_BASE,R2   ; Within SCB?
                   06    15     023A   636          BLEQ    10$                       ; Branch if not
              52   03    D3     023C   637          BITL    #3,R2                     ; Longword aligned?
                   01    12     023F   638          BNEQ    10$                       ; Branch if not
                         05     0241   639          RSB                               ; Return since vector ok
                                0242   640
    50    00660038 8F    D0     0242   641 10$:     MOVL    #DS$_IVVECT,R0            ; Sorry bad vector, return
                         04     0249   642          RET
```

```
                     024A   644          .SBTTL  DSX$SETIPL      SET INTERRUPT PRIORITY LEVEL SERVICE PROCEDURE.
                     024A   645 ;++
                     024A   646 ; FUNCTIONAL DESCRIPTION:
                     024A   647 ;
                     024A   648 ;       THIS ROUTINE SETS THE CURRENT IPL
                     024A   649 ;
                     024A   650 ; CALLING SEQUENCE:     NONE
                     024A   651 ;
                     024A   652 ; INPUT PARAMETERS:     NONE
                     024A   653 ;
                     024A   654 ; IMPLICIT INPUTS:      NONE
                     024A   655 ;
                     024A   656 ; OUTPUT PARAMETERS:    NONE
                     024A   657 ;
                     024A   658 ; IMPLICIT OUTPUTS:     NONE
                     024A   659 ;
                     024A   660 ; COMPLETION CODES:     NONE
                     024A   661 ;
                     024A   662 ; SIDE EFFECTS:         NONE
                     024A   663 ;--
```

```
                                0000  024A   665 .ENTRY  DSX$SETIPL,^M<>
                                      024C   666
                                      024C   667         CMK     SETIPL                      ; Change mode to kernel
                                7E    DC     024C         MOVPSL  -(SP)
                             06 6E    1A     E0  024E         BBS     #PSL$V_IS, (SP), 30002$
                                8E    D4     0252         CLRL    (SP)+
                                09    BC     0254         CHMK    #CMK$_SETIPL
                                06    11     0256         BRB     30003$
                          00000266'EF   16     0258     30002$: JSB     CMK$SETIPL
                                      025E   669     30003$:
                       50 00000000'8F   D0     025E   668         MOVL    #SS$_NORMAL,R0          ; INDICATE SUCCESS
                                04    0265   669         RET
                                      0266   670
                                      0266   671 CMK$SETIPL::
                             12 04 AC   DA     0266   672         MTPR    4(AP), #PR$_IPL
                          04 AE 05 10 04 AC   F0     026A   673         INSV    4(AP),#PSL$V_IPL, -
                                      0271   674                 #PSL$S_IPL,4(SP)            ; Set IPL also in PSL about to be.
                                02    0271   675         REI                                 ; Exit CHMK
```

```
                        0272   677          .SBTTL  SOFT$INT        Software interrupt vectors
                        0272   678 ;++
                        0272   679 ; FUNCTIONAL DESCRIPTION:
                        0272   680 ;
                        0272   681 ;       These vectors are the software interrupt vector.
                        0272   682 ;
                        0272   683 ;--
                        0272   684          .ALIGN  LONG
                        0274   685 SOFT$INT_A:
                00000000' 0274   686          .ADDRESS        0               ; IPL 0 impossible / filler
                00000000' 0278   687          .ADDRESS        0               ; IPL 1 usused
                00000000' 027C   688          .ADDRESS        SCH$ASTDEL      ; IPL 2 is AST delivery
                00000000' 0280   689          .ADDRESS        0               ; IPL 3 unused
                00000000' 0284   690          .ADDRESS        IOC$IOPOST      ; IPL 4 QIO post processing
                00000000' 0288   691          .ADDRESS        DS$SOFTIPL5     ; IPL 5 Enter special handler
                00000000' 028C   692          .ADDRESS        EXE$FORKDSPTH   ; IPL 6 Fork dispatch
                00000000' 0290   693          .ADDRESS        DSI$TIMER       ; IPL 7 Timer interrupt service
                00000000' 0294   694          .ADDRESS        EXE$FORKDSPTH   ; IPL 8 Fork dispatch
                00000000' 0298   695          .ADDRESS        EXE$FORKDSPTH   ; IPL 9 Fork dispatch
                00000000' 029C   696          .ADDRESS        EXE$FORKDSPTH   ; IPL 10 Fork dispatch
                00000000' 02A0   697          .ADDRESS        EXE$FORKDSPTH   ; IPL 11 Fork dispatch
                00000000' 02A4   698          .ADDRESS        0               ; IPL 12 unused
                00000000' 02A8   699          .ADDRESS        0               ; IPL 13 unused
                00000000' 02AC   700          .ADDRESS        0               ; IPL 14 unused
                00000000' 02B0   701          .ADDRESS        0               ; IPL 15 unused
                          02B4   702
                          02B4   703
                          02B4   704          .ALIGN  LONG                    ; Software interrupt vector (all IPL's)
                          02B4   705 SOFT$INT:
                      03  BB    02B4   706          PUSHR   #^M<R0,R1>              ; Save register
                   50 12  DB    02B6   707          MFPR    #PR$_IPL,R0             ; Get IPL
0A 00000011'EF     50  E4    02B9   708          BBSC    R0,DS$GA_SOFTINT,10$    ; Branch if supervisor expected this
 51    00000011'EF40  D0    02C1   709          MOVL    DS$GA_SOFTINT[R0],R1    ; User handler address?
                   1B  12    02C9   710          BNEQ    20$                     ; Branch if user handler
                          02CB   711
      51    A5 AF40  D0    02CB   712 10$:     MOVL    SOFT$INT_A[R0],R1       ; Get handler address
                   25  12    02D0   713          BNEQ    30$                     ; Branch if handler specified
        51    04 AE  D0    02D2   714          MOVL    4(SP),R1                ; Restore R1
           50 20  C0    02D6   715          ADDL    #^X20,R0                ; Generate SCB offset
   04 AE   50 02  78    02D9   716          ASHL    #2,R0,4(SP)             ; SCB vector of unexpected interrupt
                   01  BA    02DE   717          POPR    #^M<R0>                 ; Restore register
     00000000'EF   17  02E0   718          JMP     EXE_UNEXPINT            ; Process unexpected interrupt          [23]
                          02E6   719
         7E    6E  7D    02E6   720 20$:     MOVQ    (SP),-(SP)              ; Make space for PC/PSL
      08 AE    51  D0    02E9   721          MOVL    R1,8(SP)                ; Set PC
            0C AE  DC    02ED   722          MOVPSL  12(SP)                  ; Current PSL for interrupt routine
      51    80 AF40  D0    02F0   723          MOVL    SOFT$INT_A[R0],R1       ; Get supervisor handler
                   0A  13    02F5   724          BEQL    40$                     ; Branch if none
                          02F7   725
         7E    6E  7D    02F7   726 30$:     MOVQ    (SP),-(SP)              ; make space for PC/PSL
      08 AE    51  D0    02FA   727          MOVL    R1,8(SP)                ; Set PC
            0C AE  DC    02FE   728          MOVPSL  12(SP)                  ; Current PSL for interrupt routine
                          0301   729
                      03  BA    0301   730 40$:     POPR    #^M<R0,R1>              ; Restore
                          02  0303   731          REI                             ; Enter interrupt handler
                          0304   732
                          0304   733          .END
```

```
$$ARGS                          = 00000003     D
$$T1                            = 00000010     D
$ER                             = 00000001     D
$MODULE                           00000000 R   D   03
BIT...                          = 00000004     D
CHECK_VECTOR                      0000022F R   D   05
CLK$K_TCR0                        ********W GX     00
CLK$RE_INIT                       ********    X    05
CMK$INITSCB                       00000026 RG  D   05
CMK$SETIPL                        00000266 RG  D   05
CMK$_APBOOT                     = 00000004     D
CMK$_ASTEXIT                    = 00000000     D
CMK$_CANTIM                     = 0000000B     D
CMK$_HIBER                      = 00000007     D
CMK$_INITSCB                    = 00000008     D
CMK$_LAST                       = 0000000E     D
CMK$_MMENABLE                   = 00000003     D
CMK$_RCONSOLE                   = 00000001     D
CMK$_REFRESH                    = 00000005     D
CMK$_SCHDWK                     = 00000006     D
CMK$_SETIMR                     = 0000000C     D
CMK$_SETIPL                     = 00000009     D
CMK$_SETPRT                     = 0000000D     D
CMK$_SGIPR                      = 0000000A     D
CMK$_TCONSOLE                   = 00000002     D
CVTREG$_DATA                    = 00000008     D
CVTREG$_MAXLEN                  = 00000014     D
CVTREG$_MNEADR                  = 0000000C     D
CVTREG$_MSB                     = 00000004     D
CVTREG$_NARGS                   = 0000000B     D
CVTREG$_STRBUF                  = 00000010     D
CVTREG$_V1                      = 00000018     D
CVTREG$_V2                      = 0000001C     D
CVTREG$_V3                      = 00000020     D
CVTREG$_V4                      = 00000024     D
CVTREG$_V5                      = 00000028     D
CVTREG$_V6                      = 0000002C     D
DS$GA_BREAKVEC                    00000005 RG  D   02
DS$GA_CHMKVEC                     00000001 RG  D   02
DS$GA_SOFTINT                     00000011 RG  D   02
DS$GA_TBITVEC                     00000009 RG  D   02
DS$GB_STOPPING_THE_TIMER          00000000 RG  D   02
DS$GL_FLAGS                       ********   X   05
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_WARNING                    = 00000000     D
DS$M_ABRTFLG                    = 00000040     D
DS$M_BADTIME                    = 00100000     D
DS$M_BATCH                      = 00400000     D
DS$M_BRKCLR                     = 00001000     D
DS$M_BRKPT                      = 00000800     D
DS$M_CHARFLG                    = 00000100     D
DS$M_CMDFLG                     = 00000080     D
DS$M_CTRLC                      = 00000001     D
DS$M_CTRLO                      = 00010000     D
DS$M_DEVFLG                     = 00000200     D
DS$M_DISABLCC                   = 01000000     D
DS$M_DONFLG                     = 00002000     D
DS$M_ERRFLG                     = 00000010     D
DS$M_EXCEPT                     = 00080000     D
DS$M_EXETST                     = 00040000     D
DS$M_HLTFLG                     = 00000008     D
DS$M_LODFLG                     = 00000002     D
DS$M_MEMMGT                     = 00008000     D
DS$M_NI                         = 04000000     D
DS$M_OUTPUT                     = 00800000     D
DS$M_RUBFLG                     = 00000020     D
DS$M_SCRIPT                     = 00200000     D
DS$M_SETIMR                     = 02000000     D
DS$M_STRFLG                     = 00000004     D
DS$M_SUBT                       = 00004000     D
DS$M_SYSFLG                     = 00000400     D
DS$M_TIMED                      = 02000000     D
DS$M_TIMED_OUT                  = 08000000     D
DS$M_TIMRON                     = 00020000     D
DS$SOFTIPL5                       ********   X   05
DS$V_ABRTFLG                    = 00000006     D
DS$V_BADTIME                    = 00000014     D
DS$V_BATCH                      = 00000016     D
DS$V_BRKCLR                     = 0000000C     D
DS$V_BRKPT                      = 0000000B     D
DS$V_CHARFLG                    = 00000008     D
DS$V_CMDFLG                     = 00000007     D
DS$V_CTRLC                      = 00000000     D
DS$V_CTRLO                      = 00000010     D
DS$V_DEVFLG                     = 00000009     D
DS$V_DISABLCC                   = 00000018     D
DS$V_DONFLG                     = 0000000D     D
DS$V_ERRFLG                     = 00000004     D
DS$V_EXCEPT                     = 00000013     D
DS$V_EXETST                     = 00000012     D
DS$V_HLTFLG                     = 00000003     D
DS$V_LODFLG                     = 00000001     D
DS$V_MEMMGT                     = 0000000F     D
DS$V_NI                         = 0000001A     D
DS$V_OUTPUT                     = 00000017     D
DS$V_RUBFLG                     = 00000005     D
DS$V_SCRIPT                     = 00000015     D
DS$V_SETIMR                     = 00000019     D
DS$V_STRFLG                     = 00000002     D
DS$V_SUBT                       = 0000000E     D
DS$V_SYSFLG                     = 0000000A     D
DS$V_TIMED                      = 00000019     D
DS$V_TIMED_OUT                  = 0000001B     D
DS$V_TIMRON                     = 00000011     D
DS$_AP_NORMAL_BREAK             = 00660150     D
DS$_ARITH                       = 006600D0     D
DS$_ASBE                        = 00660118     D
DS$_BADLINK                     = 006600F0     D
DS$_BADTYPE                     = 006600E8     D
DS$_BIIC                        = 00660120     D
DS$_CHME                        = 006600A8     D
```

```
DS$_CHMK                        = 006600E0     D        EXE$GL_SCB                      0000000D RG D  02
DS$_DEVNAME                     = 00660108     D        EXE_ACCVIO                      ********   X   04
DS$_ERROR                       = 00660002     D        EXE_ARITH                       ********   X   04
DS$_FHWE                        = 00660068     D        EXE_BREAK                       ********   X   04
DS$_FRAGBUF                     = 00660080     D        EXE_CHME                        ********   X   04
DS$_GK_EXE_MCHK                 = 00000000     D        EXE_CHMS                        ********   X   04
DS$_GK_INITSCB                  = 00000002     D        EXE_CHMU                        ********   X   04
DS$_GK_INT_WORK                 = 00000003     D        EXE_CMODKRNL                    ********   X   04
DS$_GK_TST$MCHK                 = 00000001     D        EXE_COMPAT                      ********   X   04
DS$_ICBUSY                      = 006600C8     D        EXE_KRNLSTK                     ********   X   04
DS$_ICERR                       = 006600C0     D        EXE_MCHK                        ********   X   04
DS$_IHWE                        = 00660060     D        EXE_OPCCUS                      ********   X   04
DS$_ILLCHAR                     = 00660018     D        EXE_OPCDEC                      ********   X   04
DS$_ILLPAGCNT                   = 00660078     D        EXE_POWER                       ********   X   04
DS$_ILLUNIT                     = 00660100     D        EXE_RADRMOD                     ********   X   04
DS$_INIT_FAIL                   = 00660140     D        EXE_ROPRAND                     ********   X   04
DS$_INSFMEM                     = 00660050     D        EXE_TBIT                        ********   X   04
DS$_INVCPU                      = 00660130     D        EXE_TRANSL                      ********   X   04
DS$_IPL2HI                      = 006600B8     D        EXE_UNEXPINT                    ********   X   06
DS$_IVADDR                      = 00660040     D        ICCS$M_ERR                    = 80000000     D
DS$_IVVECT                      = 00660038     D        ICCS$M_INT                    = 00000080     D
DS$_KRNLSTK                     = 00660090     D        IOC$IOPOST                      ********   X   05
DS$_LOGIC                       = 00660070     D        IS                            = 00000001     D
DS$_MCHK                        = 00660088     D        KB_CHECK                        ********   X   05
DS$_MEM_ALLOC_ERR               = 00660138     D        ONLY_SCB                        ********   X   05
DS$_MMOFF                       = 00660058     D        PR$_ICCS                      = 00000018     D
DS$_NEEDUNIT                    = 006600F8     D        PR$_IPL                       = 00000012     D
DS$_NODE                        = 00660128     D        PR$_SCBB                      = 00000011     D
DS$_NOPCS                       = 00660110     D        PR$_SID                       = 0000003E     D
DS$_NORMAL                      = 00660001     D        PR$_SID_TYPUV2                = 00000008     D
DS$_NOSUPPORT                   = 006600B1     D        PSL$S_IPL                     = 00000005     D
DS$_NOTALLOWMP                  = 00660148     D        PSL$V_IPL                     = 00000010     D
DS$_NOTDON                      = 00660030     D        PSL$V_IS                      = 0000001A     D
DS$_NOTIMP                      = 006600B0     D        SCB$L_ACCESS                    00000020     D
DS$_NULLSTR                     = 00660010     D        SCB$L_ARITH                     00000034     D
DS$_OVERFLOW                    = 00660008     D        SCB$L_BREAK                     0000002C     D
DS$_POWER                       = 00660098     D        SCB$L_CHME                      00000044     D
DS$_PROGERR                     = 00660020     D        SCB$L_CHMK                      00000040     D
DS$_SEVERE                      = 00660004     D        SCB$L_CHMS                      00000048     D
DS$_TRANSL                      = 006600A0     D        SCB$L_CHMU                      0000004C     D
DS$_TRUNCATE                    = 00660028     D        SCB$L_COMPAT                    00000030     D
DS$_UNEXPINT                    = 006600D8     D        SCB$L_KNLSTK                    00000008     D
DS$_UNSUPPDEV                   = 00660158     D        SCB$L_MACHCHK                   00000004     D
DS$_VASFULL                     = 00660048     D        SCB$L_OPCCUS                    00000014     D
DS$_WARNING                     = 00660000     D        SCB$L_OPCDEC                    00000010     D
DSI$CHANNEL_VEC                   ********     X   06   SCB$L_POWER                     0000000C     D
DSI$TIMER                         ********     X   05   SCB$L_RADRMOD                  0000001C     D
DSI$TIMSRV                        ********     X   04   SCB$L_ROPRAND                  00000018     D
DSR$FAULT_CLEAR                   ********     X   05   SCB$L_RXDB                     000000F8     D
DSX$CLRVEC                        000001A6 RG  D   05   SCB$L_SFTLVL1                  00000084     D
DSX$CLRVEC_X                      00000228 R   D   05   SCB$L_SFTLVL10                 000000A8     D
DSX$INITSCB                       00000000 RG  D   05   SCB$L_SFTLVL11                 000000AC     D
DSX$SETIPL                        0000024A RG  D   05   SCB$L_SFTLVL12                 000000B0     D
DSX$SETVEC                        000000C7 RG  D   05   SCB$L_SFTLVL13                 000000B4     D
DSX$SETVEC_X                      0000019F R   D   05   SCB$L_SFTLVL14                 000000B8     D
EXE$FORKDSPTH                     ********     X   05   SCB$L_SFTLVL15                 000000BC     D
EXE$GB_CPUTYPE                    ********     X   05   SCB$L_SFTLVL2                  00000088     D
```

```
SCB$L_SFTLVL3                   0000008C    D
SCB$L_SFTLVL4                   00000090    D
SCB$L_SFTLVL5                   00000094    D
SCB$L_SFTLVL6                   00000098    D
SCB$L_SFTLVL7                   0000009C    D
SCB$L_SFTLVL8                   000000A0    D
SCB$L_SFTLVL9                   000000A4    D
SCB$L_TBIT                      00000028    D
SCB$L_TIMER                     000000C0    D
SCB$L_TRANSL                    00000024    D
SCB$L_TXDB                      000000FC    D
SCB$L_VM                        00000038    D
SCB$L_ZERO                      00000000    D
SCBVECTOR_000                   ********  X   04
SCBVECTOR_038                   ********  X   04
SCBVECTOR_03C                   ********  X   04
SCBVECTOR_050                   ********  X   04
SCBVECTOR_054                   ********  X   04
SCBVECTOR_058                   ********  X   04
SCBVECTOR_05C                   ********  X   04
SCBVECTOR_060                   ********  X   04
SCBVECTOR_064                   ********  X   04
SCBVECTOR_068                   ********  X   04
SCBVECTOR_06C                   ********  X   04
SCBVECTOR_070                   ********  X   04
SCBVECTOR_074                   ********  X   04
SCBVECTOR_078                   ********  X   04
SCBVECTOR_07C                   ********  X   04
SCBVECTOR_080                   ********  X   04
SCBVECTOR_0C4                   ********  X   04
SCBVECTOR_0C8                   ********  X   04
SCBVECTOR_0CC                   ********  X   04
SCBVECTOR_0D0                   ********  X   04
SCBVECTOR_0D4                   ********  X   04
SCBVECTOR_0D8                   ********  X   04
SCBVECTOR_0DC                   ********  X   04
SCBVECTOR_0E0                   ********  X   04
SCBVECTOR_0E4                   ********  X   04
SCBVECTOR_0E8                   ********  X   04
SCBVECTOR_0EC                   ********  X   04
SCBVECTOR_0F0                   ********  X   04
SCBVECTOR_0F4                   ********  X   04
SCBVECTOR_0F8                   ********  X   04
SCBVECTOR_0FC                   ********  X   04
SCB_ACCVIO                      00000020 R  D   04
SCB_ARITH                       00000034 R  D   04
SCB_BASE                        ********    X   05
SCB_BREAK                       0000002C R  D   04
SCB_CHME                        00000044 R  D   04
SCB_CHMK                        00000040 R  D   04
SCB_CHMS                        00000048 R  D   04
SCB_CHMU                        0000004C R  D   04
SCB_COMPAT                      00000030 R  D   04
SCB_IMAGE                       00000000 RG D   04
SCB_KRNLSTK                     00000008 R  D   04
SCB_MCHK                        00000004 R  D   04
SCB_OPCCUS                      00000014 R  D   04
SCB_OPCDEC                      00000010 R  D   04
SCB_POWER                       0000000C R  D   04
SCB_RADRMOD                     0000001C R  D   04
SCB_ROPRAND                     00000018 R  D   04
SCB_SFTLVL1                     00000084 R  D   04
SCB_SFTLVL10                    000000A8 R  D   04
SCB_SFTLVL11                    000000AC R  D   04
SCB_SFTLVL12                    000000B0 R  D   04
SCB_SFTLVL13                    000000B4 R  D   04
SCB_SFTLVL14                    000000B8 R  D   04
SCB_SFTLVL15                    000000BC R  D   04
SCB_SFTLVL2                     00000088 R  D   04
SCB_SFTLVL3                     0000008C R  D   04
SCB_SFTLVL4                     00000090 R  D   04
SCB_SFTLVL5                     00000094 R  D   04
SCB_SFTLVL6                     00000098 R  D   04
SCB_SFTLVL7                     0000009C R  D   04
SCB_SFTLVL8                     000000A0 R  D   04
SCB_SFTLVL9                     000000A4 R  D   04
SCB_TBIT                        00000028 R  D   04
SCB_TIMER                       000000C0 R  D   04
SCB_TRANSL                      00000024 R  D   04
SCB_UNKINT                      ********    X   05
SCH$ASTDEL                      ********    X   05
SETVEC$_CODE                  = 0000000C      D
SETVEC$_NARGS                 = 00000003      D
SETVEC$_SRVADR                = 00000008      D
SETVEC$_VECTOR                = 00000004      D
SIZ...                        = 00000001      D
SOFT$INT                        000002B4 R  D   05
SOFT$INT_A                      00000274 R  D   05
SS$_NORMAL                      ********    X   05
UNKNOWN_INT                     00000000 R  D   06
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000200 (512.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WORK	00000051 (81.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DATA	00000004 (4.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
$SCB	00000100 (256.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC PAGE
CODE	00000304 (772.)	05 (5.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG
SEP	00000013 (19.)	06 (6.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL                          VALUE        DEFINITION      REFERENCES...
------                          -----        ----------      -------------
$$ARGS                         =00000003     178    (1)      173    (1)      178    (1)
$$T1                           =00000010     178    (1)      173    (1)      178    (1)
$ER                            =00000001     217    (1)
$MODULE                         00000000-R   217    (1)
BIT...                         =00000004     181    (1)      174    (1)      179    (1)      180    (1)      181    (1)
CHECK_VECTOR                    0000022F-R   633    (1)    #-428    (1)    #-557    (1)
CLK$K_TCR0                      00000000-XR                  187    (1)    #-472    (1)
CLK$RE_INIT                     00000000-XR                  362    (1)      591    (1)
CMK$INITSCB                     00000026-R   335    (1)      323    (1)
CMK$SETIPL                      00000266-R   671    (1)      667    (1)
CMK$_APBOOT                    =00000004     179    (1)
CMK$_ASTEXIT                   =00000000     179    (1)
CMK$_CANTIM                    =0000000B     179    (1)
CMK$_HIBER                     =00000007     179    (1)
CMK$_INITSCB                   =00000008     179    (1)    #-323    (1)
CMK$_LAST                      =0000000E     179    (1)
CMK$_MMENABLE                  =00000003     179    (1)
CMK$_RCONSOLE                  =00000001     179    (1)
CMK$_REFRESH                   =00000005     179    (1)
CMK$_SCHDWK                    =00000006     179    (1)
CMK$_SETIMR                    =0000000C     179    (1)
CMK$_SETIPL                    =00000009     179    (1)    #-667    (1)
CMK$_SETPRT                    =0000000D     179    (1)
CMK$_SGIPR                     =0000000A     179    (1)
CMK$_TCONSOLE                  =00000002     179    (1)
CVTREG$_DATA                   =00000008     173    (1)
CVTREG$_MAXLEN                 =00000014     173    (1)
CVTREG$_MNEADR                 =0000000C     173    (1)
CVTREG$_MSB                    =00000004     173    (1)
CVTREG$_NARGS                  =0000000B     173    (1)
CVTREG$_STRBUF                 =00000010     173    (1)
CVTREG$_V1                     =00000018     173    (1)
CVTREG$_V2                     =0000001C     173    (1)
CVTREG$_V3                     =00000020     173    (1)
CVTREG$_V4                     =00000024     173    (1)
CVTREG$_V5                     =00000028     173    (1)
CVTREG$_V6                     =0000002C     173    (1)
DS$GA_BREAKVEC                  00000005-R   202    (1)    #-367    (1)      443    (1)    #-566    (1)
DS$GA_CHMKVEC                   00000001-R   199    (1)    #-366    (1)      438    (1)    #-561    (1)
DS$GA_SOFTINT                   00000011-R   211    (1)    #-371    (1)      497    (1)    #-579    (1)      708    (1)
                                                           #-709    (1)
DS$GA_TBITVEC                   00000009-R   205    (1)    #-368    (1)      448    (1)    #-571    (1)
DS$GB_STOPPING_THE_TIMER        00000000-R   196    (1)    #-461    (1)    #-489    (1)
DS$GL_FLAGS                     00000000-XR                  326    (1)      364    (1)      454    (1)      593    (1)
DS$K_ERROR                     =00000002     174    (1)
DS$K_NORMAL                    =00000001     174    (1)
DS$K_SEVERE                    =00000004     174    (1)
DS$K_SUBSYS                    =00000066     174    (1)      174    (1)
DS$K_WARNING                   =00000000     174    (1)
DS$M_ABRTFLG                   =00000040     180    (1)
```

```
DS$M_BADTIME                =00100000     180    (1)
DS$M_BATCH                  =00400000     180    (1)
DS$M_BRKCLR                 =00001000     180    (1)
DS$M_BRKPT                  =00000800     180    (1)
DS$M_CHARFLG                =00000100     180    (1)
DS$M_CMDFLG                 =00000080     180    (1)
DS$M_CTRLC                  =00000001     180    (1)
DS$M_CTRLO                  =00010000     180    (1)
DS$M_DEVFLG                 =00000200     180    (1)
DS$M_DISABLCC               =01000000     180    (1)
DS$M_DONFLG                 =00002000     180    (1)
DS$M_ERRFLG                 =00000010     180    (1)
DS$M_EXCEPT                 =00080000     180    (1)
DS$M_EXETST                 =00040000     180    (1)
DS$M_HLTFLG                 =00000008     180    (1)
DS$M_LODFLG                 =00000002     180    (1)
DS$M_MEMMGT                 =00008000     180    (1)
DS$M_NI                     =04000000     180    (1)
DS$M_OUTPUT                 =00800000     180    (1)
DS$M_RUBFLG                 =00000020     180    (1)
DS$M_SCRIPT                 =00200000     180    (1)
DS$M_SETIMR                 =02000000     180    (1)
DS$M_STRFLG                 =00000004     180    (1)
DS$M_SUBT                   =00004000     180    (1)
DS$M_SYSFLG                 =00000400     180    (1)
DS$M_TIMED                  =02000000     180    (1)
DS$M_TIMED_OUT              =08000000     180    (1)
DS$M_TIMRON                 =00020000     180    (1)
DS$SOFTIPL5                  00000000-XR                  691    (1)
DS$V_ABRTFLG                =00000006     180    (1)
DS$V_BADTIME                =00000014     180    (1)
DS$V_BATCH                  =00000016     180    (1)
DS$V_BRKCLR                 =0000000C     180    (1)
DS$V_BRKPT                  =0000000B     180    (1)
DS$V_CHARFLG                =00000008     180    (1)
DS$V_CMDFLG                 =00000007     180    (1)
DS$V_CTRLC                  =00000000     180    (1)
DS$V_CTRLO                  =00000010     180    (1)
DS$V_DEVFLG                 =00000009     180    (1)
DS$V_DISABLCC               =00000018     180    (1)
DS$V_DONFLG                 =0000000D     180    (1)
DS$V_ERRFLG                 =00000004     180    (1)
DS$V_EXCEPT                 =00000013     180    (1)
DS$V_EXETST                 =00000012     180    (1)
DS$V_HLTFLG                 =00000003     180    (1)
DS$V_LODFLG                 =00000001     180    (1)
DS$V_MEMMGT                 =0000000F     180    (1)
DS$V_NI                     =0000001A     180    (1)
DS$V_OUTPUT                 =00000017     180    (1)
DS$V_RUBFLG                 =00000005     180    (1)
DS$V_SCRIPT                 =00000015     180    (1)
DS$V_SETIMR                 =00000019     180    (1)
DS$V_STRFLG                 =00000002     180    (1)
DS$V_SUBT                   =0000000E     180    (1)
DS$V_SYSFLG                 =0000000A     180    (1)
DS$V_TIMED                  =00000019     180    (1)
DS$V_TIMED_OUT              =0000001B     180    (1)
```

```
DS$V_TIMRON                =00000011     180    (1)    #-326    (1)    #-363    (1)    #-453    (1)    #-592    (1)
DS$_AP_NORMAL_BREAK        =00660150     174    (1)
DS$_ARITH                  =006600D0     174    (1)
DS$_ASBE                   =00660118     174    (1)
DS$_BADLINK                =006600F0     174    (1)
DS$_BADTYPE                =006600E8     174    (1)
DS$_BIIC                   =00660120     174    (1)
DS$_CHME                   =006600A8     174    (1)
DS$_CHMK                   =006600E0     174    (1)
DS$_DEVNAME                =00660108     174    (1)
DS$_ERROR                  =00660002     174    (1)
DS$_FHWE                   =00660068     174    (1)
DS$_FRAGBUF                =00660080     174    (1)
DS$_GK_EXE_MCHK            =00000000     181    (1)
DS$_GK_INITSCB             =00000002     181    (1)    #-376    (1)
DS$_GK_INT_WORK            =00000003     181    (1)
DS$_GK_TST$MCHK            =00000001     181    (1)
DS$_ICBUSY                 =006600C8     174    (1)    #-455    (1)
DS$_ICERR                  =006600C0     174    (1)
DS$_IHWE                   =00660060     174    (1)
DS$_ILLCHAR                =00660018     174    (1)
DS$_ILLPAGCNT              =00660078     174    (1)
DS$_ILLUNIT                =00660100     174    (1)
DS$_INIT_FAIL              =00660140     174    (1)
DS$_INSFMEM                =00660050     174    (1)
DS$_INVCPU                 =00660130     174    (1)
DS$_IPL2HI                 =006600B8     174    (1)
DS$_IVADDR                 =00660040     174    (1)    #-430    (1)
DS$_IVVECT                 =00660038     174    (1)    #-641    (1)
DS$_KRNLSTK                =00660090     174    (1)
DS$_LOGIC                  =00660070     174    (1)
DS$_MCHK                   =00660088     174    (1)
DS$_MEM_ALLOC_ERR          =00660138     174    (1)
DS$_MMOFF                  =00660058     174    (1)
DS$_NEEDUNIT               =006600F8     174    (1)
DS$_NODE                   =00660128     174    (1)
DS$_NOPCS                  =00660110     174    (1)
DS$_NORMAL                 =00660001     174    (1)    #-516    (1)    #-594    (1)
DS$_NOSUPPORT              =006600B1     174    (1)
DS$_NOTALLOWMP             =00660148     174    (1)
DS$_NOTDON                 =00660030     174    (1)
DS$_NOTIMP                 =006600B0     174    (1)      174    (1)
DS$_NULLSTR                =00660010     174    (1)
DS$_OVERFLOW               =00660008     174    (1)
DS$_POWER                  =00660098     174    (1)
DS$_PROGERR                =00660020     174    (1)
DS$_SEVERE                 =00660004     174    (1)
DS$_TRANSL                 =006600A0     174    (1)
DS$_TRUNCATE               =00660028     174    (1)
DS$_UNEXPINT               =006600D8     174    (1)
DS$_UNSUPPDEV              =00660158     174    (1)
DS$_VASFULL                =00660048     174    (1)
DS$_WARNING                =00660000     174    (1)      174    (1)
DSI$CHANNEL_VEC             00000000-XR                 386    (1)
DSI$TIMER                   00000000-XR                 693    (1)
DSI$TIMSRV                  00000000-XR                 275    (1)
DSR$FAULT_CLEAR             00000000-XR                 377    (1)
```

```
DSX$CLRVEC                      000001A6-R   555    (1)
DSX$CLRVEC_X                    00000228-R   596    (1)
DSX$INITSCB                     00000000-R   321    (1)
DSX$SETIPL                      0000024A-R   665    (1)
DSX$SETVEC                      000000C7-R   426    (1)
DSX$SETVEC_X                    0000019F-R   518    (1)    #-457    (1)
EXE$FORKDSPTH                   00000000-XR            692    (1)      694    (1)      695    (1)      696    (1)
                                                       697    (1)
EXE$GB_CPUTYPE                  00000000-XR          #-459    (1)
EXE$GL_SCB                      0000000D-R   208    (1)    #-375    (1)
EXE_ACCVIO                      00000000-XR            235    (1)
EXE_ARITH                       00000000-XR            240    (1)
EXE_BREAK                       00000000-XR            238    (1)
EXE_CHME                        00000000-XR            244    (1)
EXE_CHMS                        00000000-XR            245    (1)
EXE_CHMU                        00000000-XR            246    (1)
EXE_CMODKRNL                    00000000-XR            243    (1)
EXE_COMPAT                      00000000-XR            239    (1)
EXE_KRNLSTK                     00000000-XR            229    (1)
EXE_MCHK                        00000000-XR            228    (1)
EXE_OPCCUS                      00000000-XR            232    (1)
EXE_OPCDEC                      00000000-XR            231    (1)
EXE_POWER                       00000000-XR            230    (1)
EXE_RADRMOD                     00000000-XR            234    (1)
EXE_ROPRAND                     00000000-XR            233    (1)
EXE_TBIT                        00000000-XR            237    (1)
EXE_TRANSL                      00000000-XR            236    (1)
EXE_UNEXPINT                    00000000-XR            387    (1)      718    (1)
ICCS$M_ERR                      =80000000    184    (1)    #-470    (1)    #-481    (1)
ICCS$M_INT                      =00000080    185    (1)    #-471    (1)    #-482    (1)
IOC$IOPOST                       00000000-XR           690    (1)
IS                              =00000001    182    (1)    228    (1)      229    (1)      230    (1)      275    (1)
                                                       353    (1)      583    (1)
KB_CHECK                         00000000-XR           328    (1)      519    (1)      597    (1)
ONLY_SCB                         00000000-XR           356    (1)
PR$_ICCS                        =00000018            #-483    (1)    #-485    (1)
PR$_IPL                         =00000012            #-337    (1)    #-379    (1)    #-672    (1)    #-707    (1)
PR$_SCBB                        =00000011            #-374    (1)
PR$_SID                         =0000003E            #-466    (1)
PR$_SID_TYPUV2                  =00000008            #-459    (1)
PSL$S_IPL                       =00000005            #-674    (1)
PSL$V_IPL                       =00000010            #-673    (1)
PSL$V_IS                        =0000001A    179    (1)    #-323    (1)    #-667    (1)
SCB$L_BREAK                      0000002C            #-441    (1)    #-564    (1)
SCB$L_CHMK                       00000040            #-436    (1)    #-559    (1)
SCB$L_SFTLVL1                    00000084            #-492    (1)    #-574    (1)
SCB$L_SFTLVL15                   000000BC            #-494    (1)    #-576    (1)
SCB$L_TBIT                       00000028            #-446    (1)    #-569    (1)
SCB$L_TIMER                      000000C0            #-451    (1)    #-589    (1)
SCBVECTOR_000                    00000000-XR           227    (1)
SCBVECTOR_038                    00000000-XR           241    (1)
SCBVECTOR_03C                    00000000-XR           242    (1)
SCBVECTOR_050                    00000000-XR           247    (1)
SCBVECTOR_054                    00000000-XR           248    (1)
SCBVECTOR_058                    00000000-XR           249    (1)
SCBVECTOR_05C                    00000000-XR           250    (1)
SCBVECTOR_060                    00000000-XR           251    (1)
```

```
SCBVECTOR_064                   00000000-XR                     252     (1)
SCBVECTOR_068                   00000000-XR                     253     (1)
SCBVECTOR_06C                   00000000-XR                     254     (1)
SCBVECTOR_070                   00000000-XR                     255     (1)
SCBVECTOR_074                   00000000-XR                     256     (1)
SCBVECTOR_078                   00000000-XR                     257     (1)
SCBVECTOR_07C                   00000000-XR                     258     (1)
SCBVECTOR_080                   00000000-XR                     259     (1)
SCBVECTOR_0C4                   00000000-XR                     276     (1)
SCBVECTOR_0C8                   00000000-XR                     277     (1)
SCBVECTOR_0CC                   00000000-XR                     278     (1)
SCBVECTOR_0D0                   00000000-XR                     279     (1)
SCBVECTOR_0D4                   00000000-XR                     280     (1)
SCBVECTOR_0D8                   00000000-XR                     281     (1)
SCBVECTOR_0DC                   00000000-XR                     282     (1)
SCBVECTOR_0E0                   00000000-XR                     283     (1)
SCBVECTOR_0E4                   00000000-XR                     284     (1)
SCBVECTOR_0E8                   00000000-XR                     285     (1)
SCBVECTOR_0EC                   00000000-XR                     286     (1)
SCBVECTOR_0F0                   00000000-XR                     287     (1)
SCBVECTOR_0F4                   00000000-XR                     288     (1)
SCBVECTOR_0F8                   00000000-XR                     289     (1)
SCBVECTOR_0FC                   00000000-XR                     290     (1)
SCB_ACCVIO                      00000020-R    235     (1)
SCB_ARITH                       00000034-R    240     (1)
SCB_BASE                        00000000-XR                     339     (1)     373     (1)     434     (1)     513     (1)
                                                              #-588     (1)   #-635     (1)
SCB_BREAK                       0000002C-R    238     (1)
SCB_CHME                        00000044-R    244     (1)
SCB_CHMK                        00000040-R    243     (1)
SCB_CHMS                        00000048-R    245     (1)
SCB_CHMU                        0000004C-R    246     (1)
SCB_COMPAT                      00000030-R    239     (1)
SCB_IMAGE                       00000000-R    226     (1)     209     (1)     340     (1)   #-586     (1)
SCB_KRNLSTK                     00000008-R    229     (1)
SCB_MCHK                        00000004-R    228     (1)
SCB_OPCCUS                      00000014-R    232     (1)
SCB_OPCDEC                      00000010-R    231     (1)
SCB_POWER                       0000000C-R    230     (1)
SCB_RADRMOD                     0000001C-R    234     (1)
SCB_ROPRAND                     00000018-R    233     (1)
SCB_SFTLVL1                     00000084-R    260     (1)
SCB_SFTLVL10                    000000A8-R    269     (1)
SCB_SFTLVL11                    000000AC-R    270     (1)
SCB_SFTLVL12                    000000B0-R    271     (1)
SCB_SFTLVL13                    000000B4-R    272     (1)
SCB_SFTLVL14                    000000B8-R    273     (1)
SCB_SFTLVL15                    000000BC-R    274     (1)
SCB_SFTLVL2                     00000088-R    261     (1)
SCB_SFTLVL3                     0000008C-R    262     (1)
SCB_SFTLVL4                     00000090-R    263     (1)
SCB_SFTLVL5                     00000094-R    264     (1)
SCB_SFTLVL6                     00000098-R    265     (1)
SCB_SFTLVL7                     0000009C-R    266     (1)
SCB_SFTLVL8                     000000A0-R    267     (1)
SCB_SFTLVL9                     000000A4-R    268     (1)
SCB_TBIT                        00000028-R    237     (1)
```

```
SCB_TIMER                       000000C0-R   275     (1)
SCB_TRANSL                      00000024-R   236     (1)
SCB_UNKINT                      00000000-XR                  345     (1)     #-354    (1)     #-385    (1)      583     (1)
                                                           #-635    (1)
SCH$ASTDEL                      00000000-XR                  688     (1)
SETVEC$_CODE                    =0000000C    178     (1)
SETVEC$_NARGS                   =00000003    178     (1)
SETVEC$_SRVADR                  =00000008    178     (1)
SETVEC$_VECTOR                  =00000004    178     (1)
SIZ...                          =00000001    180     (1)     180     (1)
SOFT$INT                        000002B4-R   705     (1)     260     (1)      261     (1)      262     (1)      263     (1)
                                                             264     (1)      265     (1)      266     (1)      267     (1)
                                                             268     (1)      269     (1)      270     (1)      271     (1)
                                                             272     (1)      273     (1)      274     (1)
SOFT$INT_A                      00000274-R   685     (1)   #-712     (1)    #-723     (1)
SS$_NORMAL                      00000000-XR                #-325     (1)    #-668     (1)
UNKNOWN_INT                     00000000-R   384     (1)     346     (1)
```

```
                                    +-------------------------+
                                    ! Macros Cross Reference !
                                    +-------------------------+

MACRO            SIZE        DEFINITION      REFERENCES...
-----            ----        ----------      -------------
$DEF             1           181    (1)
$DEFINI          1           175    (1)      175    (1)      176    (1)      177    (1)
$DS_CVTREG_DEF   1           173    (1)      173    (1)
$DS_DSDEF        3           174    (1)      174    (1)
$DS_SCBDEF       3           177    (1)      177    (1)
$DS_SETVEC_DEF   1           178    (1)      178    (1)
$EQU             1           181    (1)      174    (1)      179    (1)      181    (1)
$EQULS1          1           181    (1)      174    (1)      179    (1)      181    (1)
$EQULST          1           174    (1)      174    (1)      179    (1)      181    (1)
$GBLINI          2           174    (1)      174    (1)      179    (1)      180    (1)      181    (1)
$OFFDEF          1           173    (1)      173    (1)      178    (1)
$PRDEF           5           175    (1)      175    (1)
$PSLDEF          2           176    (1)      176    (1)
$VIELD           1           180    (1)      180    (1)
$VIELD1          1           181    (1)      180    (1)
CMK              1           323    (1)      323    (1)      667    (1)
CMKDEF           1           179    (1)      179    (1)
DSBINT           1           337    (1)      337    (1)
DSFDEF           3           180    (1)      180    (1)
ENBINT           1           379    (1)      379    (1)
FLTCLR_SEL       1           181    (1)      181    (1)
MODNAM           1           217    (1)      217    (1)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ACBL	00F1	354 (1)
ADDL	00C0	715 (1)
AOBLEQ	00F3	343 (1)
ASHL	0078	350 (1) 467 (1) 716 (1)
BBCC	00E5	326 (1) 363 (1) 592 (1)
BBCS	00E3	453 (1)
BBS	00E0	323 (1) 667 (1)
BBSC	00E4	708 (1)
BEQL	0013	460 (1) 509 (1) 724 (1)
BGEQ	0018	585 (1)
BGTR	0014	495 (1) 577 (1)
BITL	00D3	431 (1) 637 (1)
BLEQ	0015	636 (1)
BLSS	0019	493 (1) 575 (1)
BNEQ	0012	432 (1) 437 (1) 442 (1) 447 (1) 452 (1) 469 (1) 560 (1)
		565 (1) 570 (1) 590 (1) 638 (1) 710 (1) 713 (1)
BRB	0011	323 (1) 444 (1) 449 (1) 457 (1) 473 (1) 490 (1) 498 (1)
		562 (1) 567 (1) 572 (1) 580 (1) 667 (1)
BRW	0031	439 (1)
BSBW	0030	428 (1) 557 (1)
CHMK	00BC	323 (1) 667 (1)
CLRB	0094	489 (1)
CLRL	00D4	323 (1) 338 (1) 366 (1) 367 (1) 368 (1) 561 (1) 566 (1)
		571 (1) 579 (1) 667 (1)
CLRQ	007C	371 (1)
CMPB	0091	459 (1) 468 (1)
CMPL	00D1	635 (1)
CMPW	00B1	436 (1) 441 (1) 446 (1) 451 (1) 492 (1) 494 (1) 559 (1)
		564 (1) 569 (1) 574 (1) 576 (1) 584 (1) 589 (1)
EXTZV	00EF	496 (1) 578 (1)
INSV	00F0	514 (1) 673 (1)
JMP	0017	387 (1) 718 (1)
JSB	0016	323 (1) 328 (1) 356 (1) 362 (1) 377 (1) 386 (1) 519 (1)
		591 (1) 597 (1) 667 (1)
MFPR	00DB	337 (1) 466 (1) 485 (1) 707 (1)
MOVAB	009E	339 (1) 340 (1) 345 (1) 346 (1) 348 (1) 353 (1) 583 (1)
MOVAL	00DE	373 (1) 434 (1) 438 (1) 443 (1) 448 (1) 497 (1) 513 (1)
MOVB	0090	351 (1) 461 (1)
MOVL	00D0	325 (1) 352 (1) 369 (1) 375 (1) 430 (1) 455 (1) 470 (1)
		510 (1) 516 (1) 586 (1) 588 (1) 594 (1) 634 (1) 641 (1)
		668 (1) 709 (1) 712 (1) 714 (1) 721 (1) 723 (1) 727 (1)
MOVPSL	00DC	323 (1) 667 (1) 722 (1) 728 (1)
MOVQ	007D	342 (1) 720 (1) 726 (1)
MTPR	00DA	337 (1) 374 (1) 379 (1) 481 (1) 672 (1)
POPR	00BA	380 (1) 515 (1) 717 (1) 730 (1)
PUSHL	00DD	376 (1) 507 (1)
PUSHR	00BB	336 (1) 706 (1)
REI	0002	381 (1) 675 (1) 731 (1)
RET	0004	329 (1) 520 (1) 598 (1) 642 (1) 669 (1)
RSB	0005	639 (1)

```
SOBGTR          00F5      372    (1)
SUBL            00C2      385    (1)
SUBL3           00C3      349    (1)
TSTL            00D5      378    (1)     508     (1)
```

! Directives Cross Reference !

DIRECTIVE	REFERENCES...															
.ADDRESS	686	(1)	687	(1)	688	(1)	689	(1)	690	(1)	691	(1)	692	(1)	693	(1)
	694	(1)	695	(1)	696	(1)	697	(1)	698	(1)	699	(1)	700	(1)	701	(1)
.ALIGN	684	(1)	704	(1)												
.ASCIC	217	(1)														
.BLKL	200	(1)	203	(1)	206	(1)										
.BYTE	197	(1)														
.END	733	(1)														
.ENDC	174	(1)	179	(1)	180	(1)	181	(1)	337	(1)	379	(1)				
.ENDM	173	(1)	174	(1)	175	(1)	176	(1)	177	(1)	178	(1)	179	(1)	180	(1)
	181	(1)	217	(1)	323	(1)	337	(1)	379	(1)						
.ENDR	174	(1)	179	(1)	180	(1)	181	(1)								
.ENTRY	321	(1)	426	(1)	555	(1)	665	(1)								
.IDENT	8	(1)														
.IF	174	(1)	179	(1)	180	(1)	181	(1)	337	(1)	379	(1)				
.IFF	174	(1)	179	(1)	180	(1)	181	(1)	337	(1)	379	(1)				
.IIF	174	(1)	179	(1)	180	(1)	181	(1)								
.IRP	173	(1)	174	(1)	178	(1)	179	(1)	180	(1)	181	(1)				
.LIBRARY	161	(1)	162	(1)	163	(1)										
.LIST	173	(1)	177	(1)	178	(1)										
.LONG	209	(1)	212	(1)	213	(1)	227	(1)	228	(1)	229	(1)	230	(1)	231	(1)
	232	(1)	233	(1)	234	(1)	235	(1)	236	(1)	237	(1)	238	(1)	239	(1)
	240	(1)	241	(1)	242	(1)	243	(1)	244	(1)	245	(1)	246	(1)	247	(1)
	248	(1)	249	(1)	250	(1)	251	(1)	252	(1)	253	(1)	254	(1)	255	(1)
	256	(1)	257	(1)	258	(1)	259	(1)	260	(1)	261	(1)	262	(1)	263	(1)
	264	(1)	265	(1)	266	(1)	267	(1)	268	(1)	269	(1)	270	(1)	271	(1)
	272	(1)	273	(1)	274	(1)	275	(1)	276	(1)	277	(1)	278	(1)	279	(1)
	280	(1)	281	(1)	282	(1)	283	(1)	284	(1)	285	(1)	286	(1)	287	(1)
	288	(1)	289	(1)	290	(1)										
.MACRO	174	(1)	179	(1)	180	(1)	181	(1)								
.MDELETE	173	(1)	174	(1)	178	(1)	179	(1)								
.NLIST	173	(1)	177	(1)	178	(1)										
.NOCROSS	175	(1)	176	(1)	177	(1)										
.NOSHOW	9	(1)														
.PAGE	156	(1)	189	(1)	214	(1)	218	(1)	291	(1)	320	(1)	388	(1)	424	(1)
	521	(1)	554	(1)	599	(1)	632	(1)	643	(1)	664	(1)	676	(1)		
.PSECT	190	(1)	215	(1)	224	(1)	293	(1)	383	(1)	425	(1)				
.RESTORE	175	(1)	176	(1)	177	(1)										
.SAVE	175	(1)	176	(1)	177	(1)										
.SBTTL	157	(1)	219	(1)	292	(1)	389	(1)	522	(1)	600	(1)	644	(1)	677	(1)
.TITLE	7	(1)														
.WEAK	187	(1)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	7	#-431	(1)	#-508	(1)	#-510	(1)	#-514	(1)	#-634	(1)	#-672	(1)
		#-673	(1)										
R0	41	#-325	(1)	#-336	(1)	#-348	(1)	#-349	(1)	#-350	(1)	#-351	(1)
		#-352	(1)	#-369	(1)	#-371	(1)	#-372	(1)	#-373	(1)	#-374	(1)
		#-375	(1)	#-380	(1)	#-430	(1)	#-455	(1)	#-466	(1)	#-467	(1)
		#-468	(1)	#-516	(1)	#-578	(1)	#-579	(1)	#-583	(1)	#-586	(1)
		#-588	(1)	#-594	(1)	#-641	(1)	#-668	(1)	#-706	(1)	#-707	(1)
		#-708	(1)	#-709	(1)	#-712	(1)	#-715	(1)	#-716	(1)	#-717	(1)
		#-723	(1)	#-730	(1)								
R1	26	#-336	(1)	#-338	(1)	#-343	(1)	#-346	(1)	#-349	(1)	#-380	(1)
		#-434	(1)	#-438	(1)	#-443	(1)	#-448	(1)	#-496	(1)	497	(1)
		#-507	(1)	#-510	(1)	#-513	(1)	514	(1)	#-515	(1)	#-706	(1)
		#-709	(1)	#-712	(1)	#-714	(1)	#-721	(1)	#-723	(1)	#-727	(1)
		#-730	(1)										
R2	32	321	(1)	#-336	(1)	#-339	(1)	#-342	(1)	#-353	(1)	#-354	(1)
		#-380	(1)	426	(1)	434	(1)	#-436	(1)	#-441	(1)	#-446	(1)
		#-451	(1)	#-492	(1)	#-494	(1)	496	(1)	513	(1)	555	(1)
		#-559	(1)	#-564	(1)	#-569	(1)	#-574	(1)	#-576	(1)	578	(1)
		583	(1)	#-584	(1)	#-586	(1)	#-588	(1)	#-589	(1)	#-634	(1)
		#-635	(1)	#-637	(1)								
R3	11	321	(1)	#-336	(1)	#-340	(1)	#-342	(1)	#-345	(1)	348	(1)
		#-352	(1)	353	(1)	#-380	(1)	426	(1)	#-485	(1)		
SP	21	#-323	(1)	#-337	(1)	#-378	(1)	#-379	(1)	#-385	(1)	#-667	(1)
		674	(1)	#-714	(1)	#-716	(1)	#-720	(1)	#-721	(1)	#-722	(1)
		#-726	(1)	#-727	(1)	#-728	(1)						

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.17
Command processing	880	00:00:00.25	00:00:00.80
Pass 1	591	00:00:01.87	00:00:03.59
Symbol table sort	0	00:00:00.13	00:00:00.12
Pass 2	36	00:00:00.57	00:00:01.15
Symbol table output	2	00:00:00.05	00:00:00.12
Psect synopsis output	2	00:00:00.02	00:00:00.08
Cross-reference output	7	00:00:00.21	00:00:00.40
Assembler run totals	1541	00:00:03.13	00:00:06.44

The working set limit was 3012 pages.
84505 bytes (166 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 454 non-local and 33 local symbols.
733 source lines were read in Pass 1, producing 55 object records in Pass 2.
63 pages of virtual memory were used to define 19 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                       Macros defined
------------------                       --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              4
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                5
SYS$COMMON:[SYSLIB]LIB.MLB;2                           2
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       8
TOTALS (all libraries)                              19

455 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:SCB.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:DIA
```

Table of contents

(1) 149 Libraries and External Symbols ; [12]
(1) 159 External Symbol Declarations
(1) 184 Macros and Equated Symbols
(1) 205 Data Psect Declarations
(1) 242 Work Psect Declarations
(1) 266 Absolute Psect Declarations
(1) 280 SCRIPT$INIT - Initialize script
(1) 322 SCRIPT$OPEN - Initialize a script file
(1) 479 SCRIPT$FLUSH, STOP, CONT - Terminate all scripts open
(1) 553 SCRIPT$FINISH - Terminate script input file
(1) 608 DS_GETLINE - Get program data input line
(1) 1057 COPY_IT - Copy and translate
(1) 1114 ADVANCE - Advance to next record in buffer

```
                                0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element SCRIPT.MAR
                                0000     2 ; *9    15-APR-1987 16:06:31 FORTUNA "REMOVE EXTRA CALL TO CLK$RE_INIT FROM FLUSH RO
                                0000     3 ; *8    14-APR-1987 16:00:38 MARTIN "PSTAR UPDATE"
                                0000     4 ; *7    15-OCT-1986 11:08:37 BERGAZZI "FIX DS$GT_PROMPT BUG FOR VSDP"
                                0000     5 ; *6    23-JUN-1986 14:22:18 MARTIN "SAME"
                                0000     6 ; *5    23-JUN-1986 14:19:07 MARTIN "DONE"
                                0000     7 ; *4    23-JUN-1986 10:59:50 MARTIN "DONE"
                                0000     8 ; *3    23-JUN-1986 10:56:59 MARTIN "DONE"
                                0000     9 ; *2    23-JUN-1986 09:45:00 MARTIN "DONE"
                                0000    10 ; *1     6-FEB-1986 15:22:27 DS ""
                                0000    11 ; DEC/CMS REPLACEMENT HISTORY, Element SCRIPT.MAR
                                0000    12         .Title  SCRIPT  *** SCRIPT Read from command stream
                                0000    13         .Ident  /10-27/                                                    ;G:9
                                0000    14         .NoShow Conditionals    ; For all the conditionals in the RMS macros
                                0000    15         .DSABL  GBL
                                0000    16
                                0000    17 ;
                                0000    18 ; Copyright (c) 1977, 1981, 1985, 1986, 1987                            ;G:8
                                0000    19 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                                0000    20 ;
                                0000    21 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                                0000    22 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                                0000    23 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                                0000    24 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                                0000    25 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                                0000    26 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                                0000    27 ; REMAIN IN DEC.
                                0000    28 ;
                                0000    29 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                                0000    30 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                                0000    31 ; CORPORATION.
                                0000    32 ;
                                0000    33 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                                0000    34 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                0000    35
                                0000    36 ;++
                                0000    37 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
                                0000    38 ;
                                0000    39 ; ABSTRACT:
                                0000    40 ;
                                0000    41 ; ENVIRONMENT:
                                0000    42 ;
                                0000    43 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
                                0000    44 ;
                                0000    45 ; MODIFICATIONS:
                                0000    46 ;                Roger Riggs     20-Jun-78       Version 02.
                                0000    47 ;                True implementation of scripting in USER mode.
                                0000    48 ;       01       Added code to implement "skipping... " for mismatched scripts
                                0000    49 ;                Roger Riggs     12-Jun-1979
                                0000    50 ;       02       changed W^ to "" on DSA$GL_FLAGS references
                                0000    51 ;                Roger Riggs, 17-Mar-1980, Version 5.3
                                0000    52 ;       03       DS$LOAD returns changed, reflected by getting SS$_NORMAL
                                0000    53 ;                Even if file was truncated.
                                0000    54 ;                Dave Butenhof   28-apr-80
                                0000    55 ;       04       Use $GET and $PUT for input/output if DS run from command file
                                0000    56 ;                Roger Riggs, 3-Sep-1980, Version 6.0
                                0000    57 ;       05       Modified SCRIPT$OPEN to use RMS instead of load
```

```
0000    58 ;
0000    59 ;                Dave Butenhof, 10-sep-1980, Version 6.0
0000    60 ;        06      Fix user mode RMS GETLINE code to perform case translation
```

```
0000     62 ;       07      - Jack Stansbury, 22-Oct-1981, Version 6.5
0000     63 ;               Fixed truncation errors. Also added .LIBRARY statements for
0000     64 ;               $DS and $DIAG.
0000     65 ;
0000     66 ;       08      - Dave Butenhof, 12-Nov-1981, version 6.5
0000     67 ;               Implement SET VERIFY/CLEAR VERIFY function.  When cleared,
0000     68 ;               this flag will cause command script input not to be echoed.
0000     69 ;               This is basically for the System Functional Text (EVXBB).
0000     70 ;                                                                    ;G:9
0000     71 ;       09      - Dave Butenhof, 17-Nov-1981, version 6.5
0000     72 ;               Implement type code bytes for all prompts.  The type code
0000     73 ;               will not be used in comparison of script file prompts,
0000     74 ;               but will be typed out to console or log file, if BINARY
0000     75 ;               flag is set.
0000     76 ;
0000     77 ;       10      -Jack Stansbury, 18-Dec-1981, Version 6.6
0000     78 ;               Added a little code in SCRIPT$FLUSH that will clear out the
0000     79 ;               QA bit in the DSA flags. This is to insure that QA gets turned
0000     80 ;               off when a Control-C interrupts a diagnostic in a script file,
0000     81 ;               and a non-continuable command (i.e., one that causes the script
0000     82 ;               file to be flushed) is issued. Before, if a START command was
0000     83 ;               issued, and a Control-C was typed, and a subsequent START
0000     84 ;               command was typed, the second one could be screwed up.
0000     85 ;
0000     86 ;       11      -Jack Stansbury, 26-Feb-1982, Version 6.6
0000     87 ;               Added comments to the DS_GETLINE routine to aid in
0000     88 ;               deciphering it.
0000     89 ;
0000     90 ;       12      - Jack Stansbury, 18-Mar-1982, Version 6.?
0000     91 ;               Took out any reference to the TypMsg routine. This is so
0000     92 ;               that type-coding can be added. Also added some new DS macros
0000     93 ;               that should make this more understandable.
0000     94 ;
0000     95 ;       13      Marion Baggett, 31-Mar-1982, Version 6.7
0000     96 ;               Added SYS$LIBRARY:LIB/
0000     97 ;
0000     98 ;       14      Marion Baggett, 8-Apr-1982, Version 6.7
0000     99 ;               Added type coding.
0000    100 ;
0000    101 ;       15      Jack Stansbury, 12-Apr-1982, Version 6.7
0000    102 ;               Added support for the Customer Runnable Diagnostics package.
0000    103 ;
0000    104 ;       16      John Ciukaj,    4-Apr-1983,     version 6.11
0000    105 ;               - Redefined reference to CRD bits in DSA Flags to
0000    106 ;                 distinguish between online and offline
0000    107 ;
0000    108 ;       17      Peter Green,    20-Jul-1983,    version 6.12
0000    109 ;               Added code to support scripting where prompts are
0000    110 ;               printed to the terminal and an answer is given by
0000    111 ;               the user. This is only done when the script prompt
0000    112 ;               ends with "\\".
0000    113 ;
0000    114 ;       18      M. Baggett      25-Jan-1984     Version 6.14
0000    115 ;               Changed dynamic allocation of XAB, FAB, RAB to static form
0000    116 ;               so script files will work.
0000    117 ;
0000    118 ;       19      M. Baggett      8-Feb_1985      Version 8.1
```

```
0000   119 ;               Changed COPY_IT to not change apha characters to uppers
0000   120 ;               case if ULTRIX.
0000   121 ;
0000   122 ;       20      Ted Salem       March 22, 1985  Version 8.2
0000   123 ;               Added interlock code to DS_GETLINE.
0000   124 ;
0000   125 ;       21      M. Baggett      20-June-1985    Version 8.3
0000   126 ;               Change some ULTRIX names.
0000   127 ;
0000   128 ;       22      Amy Iorio       10-July-1985    Version 8.3
0000   129 ;               Put ASK> prompt in for $DSASKxxx macros.
0000   130 ;
0000   131 ;       23      Domenic Andella 25-Sept-1985    Version 9.0
0000   132 ;               Fixed two truncation errors, BSBW's to JSB's
0000   133 ;                                                                       ;G:5
0000   134 ;       24      I. Martin       23-June-1986    Version 10.2            ;G:5
0000   135 ;               Changed literal to symbolic string.                     ;G:3
0000   136 ;                                                                       ;G:7
0000   137 ;       25      Bob Bergazzi    9-Oct-1986      Version 10.4            ;G:7
0000   138 ;               Fixed previous edit.                                    ;G:7
0000   139 ;                                                                       ;G:8
0000   140 ;       26      Ingrid Martin   1-Apr-1987      Version 10.8            ;G:9
0000   141 ;               RRR updates (do like nautilus).                         ;G:8
0000   142 ;                                                                       ;G:9
0000   143 ;       27      Bob Bergazzi    15-apr-1987     Version 10.8            ;G:9
0000   144 ;               Removed extra JSB to CLK$RE_INIT because it is already called ;G:9
0000   145 ;               once in the preceding $DS_CLRVEC on the timer vector.  No     ;G:9
0000   146 ;               symptom observed.                                       ;G:9
0000   147 ;--
```

```
0000    149          .SBTTL  Libraries and External Symbols  ;                          [12]
0000    150 ;
0000    151 ; INCLUDE FILES:
0000    152 ;
0000    153          .LIBRARY        /SYS$LIBRARY:LIB/       ;                          [13]
0000    154          .Library        /$CRD/                  ;                          [15]
0000    155          .LIBRARY        /$DS/                   ;                          [07]
0000    156          .LIBRARY        /$DIAG/                 ;                          [07]
0000    157
```

```
0000   159          .SBTTL  External Symbol Declarations
0000   160
0000   161 ;
0000   162 ; EXTERNAL SYMBOLS
0000   163 ;
0000   164          .EXTRN  DS$GT_PROMPT                                                    ;G:3
0000   165          .EXTRN  DS$CLI,DSX$PRINT , DS_ERRSUP, DS$LOAD          ;                 [12]
0000   166          .EXTRN  DSR$COMPLETION, DS$ABORT,CLK$RE_INIT,DS$CLRVEC
0000   167          .EXTRN  SCAN$COMPARE, SCAN$SPACES
0000   168          .EXTRN  DS$GW_TTOUT, DS$GW_TTIN
0000   169          .EXTRN  DS$GL_CLIBASE
0000   170          .EXTRN  SYS$FAO, SYS$QIOW, SYS$QIO, SYS$WAKE, SYS$HIBER, SYS$READEF
0000   171          .EXTRN  SS$_NORMAL, SS$_ENDOFFILE, SS$_CONTROLC, SS$_ILLIOFUNC
0000   172          .EXTRN  IO$_READPROMPT ,IO$_WRITEVBLK, IO$_READVBLK
0000   173          .EXTRN  DS$GL_FLAGS
0000   174          .EXTRN  EXE$ALONONPAGED, EXE$DEANONPAGED
0000   175          .EXTRN  DS$K_BUFSIZ, DS$GT_BUFFER
0000   176          .EXTRN  QIO$CLEANUP, INIT_CONTEXT, BEGIN
0000   177          .EXTRN  DS$RAB_INPUT, DS$RAB_OUTPUT
0000   178          .EXTRN  DSX$TYPE_OUT                ;                                 [09]
0000   179          .EXTRN  DEBUG_THE_SUCKER            ;                                 [99]
0000   180          .Extrn  DS$GA_CRD_DS_Interface  ; The address of the routine to ...   [15]
0000   181          .EXTRN  ULTRIX_V_MODE,ULTRIX_FLAGS      ; Ultrix flag
0000   182          .EXTRN  PRNT_PRMT
```

```
                        0000   184          .SBTTL Macros and Equated Symbols
                        0000   185 ;
                        0000   186 ; EQUATED SYMBOLS:
                        0000   187 ;
                        0000   188
                00000400  0000   189          SCRIPT$MINCORE == 2*512          ; NEEDS AT LEAST TWO PAGES FOR SCRIPT   [12]
                00000006  0000   190          pr$_sid_typ8nn  = 6              ; nautilus [26]                         ;G:8
                00000011  0000   191          pr$_sid_typ8rr  = 17             ; RRR [26]                              ;G:8
                        0000   192          $DS_SCBDEF
                        0000   193          $DS_DSADEF
                        0000   194          $DS_DSDEF
                        0000   195          DSFDEF
                        0000   196          CLIDEF
                        0000   197          $FABDEF
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000  FE70               .=0
                        0000                 .RESTORE
                        0000   198          $RMSDEF
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000  FE70               .=0
                        0000                 .RESTORE
                        0000   199          $RABDEF
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000  FE70               .=0
                        0000                 .RESTORE
                        0000   200          $XABDEF
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000  FE70               .=0
                        0000                 .RESTORE
                        0000   201          $XABFHCDEF
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000  FE70               .=0
                        0000                 .RESTORE
                        0000   202          $DS_TypeDef                      ; Define DS error codes                 [09]
                        0000   203          $CRD_Literals                    ; define some CRD equated symbols       [15]
```

```
                                    0000          .SBTTL Data Psect Declarations
                                00000000          .Psect  Data, Shr, Noexe, Nowrt, Byte
                                    0000    ;
                                    0000    ; STORAGE:
                                    0000    ;
                                    0000          MODNAM  SCRIPT
          54 50 49 52 43 53 00'     0000    $MODULE:        .ASCIC  \SCRIPT\
                                06  0000
                                    0007
                                    0007    FMT_PRMT:                                ; Prompt for $DSASKxxx macros [22]
          20 3E 4B 53 41 20 00'     0007          .ASCIC  " ASK> "
                                06  0007
                                    000E
                                    000E    T_DEF_COM:
                   4D 4F 43 2E      000E          .ASCII  ".COM"
                                    0012
                                    0012    T_NULL:
                                00' 0012          .ASCIC  ""
                                00  0012
                                    0013
                                    0013    T_EOF:
       3E 46 4F 45 3C 20 40 00'     0013          .ASCIC  "@ <EOF>"
                                07  0013
                                    001B
                                    001B    T_PROMPT:
    2F 21 44 41 21 43 41 21 00'     001B          .ASCIC  "!AC!AD!/"
                                08  001B
                                    0024
                                    0024    T_PROMPT2:
          43 41 21 43 41 21 00'     0024          .ASCIC  "!AC!AC"
                                06  0024
                                    002B
                                    002B    T_PROMPT3:
             2F 21 53 41 21 00'     002B          .ASCIC  "!AS!/"
                                05  002B
                                    0031
                                    0031    T_SKIP:
2E 2E 20 67 6E 69 70 70 69 6B 53 00'  0031          .ASCIC  'Skipping ..."!AD"!/'
       2F 21 22 44 41 21 22 2E      003D
                                13  0031
                                    0045
                                    0045    T_ECHO:
    2F 21 44 41 21 53 41 21 00'     0045          .ASCIC  '!AS!AD!/'      ; Format line echo to log file
                                08  0045
                                    004E
                                    004E    T_PNF:
43 41 21 22 20 74 70 6D 6F 72 50 00'  004E          .ASCIC  'Prompt "!AC" not found in script!/'
20 64 6E 75 6F 66 20 74 6F 6E 20 22   005A
   2F 21 74 70 69 72 63 73 20 6E 69   0066
                                22  004E
```

```
                     0071   242          .SBTTL  Work Psect Declarations
                 00000000   243          .PSECT  Work, Noshr, Noexe, Wrt, Long
                     0000   244 SCRIPT_XAB:     $XABFHC          ; XAB                                  [18]
                       1D   0000                .BYTE XAB$C_FHC
                       2C   0001                .BYTE XAB$C_FHCLEN
                 0000002C   0002                .BLKB XAB$C_FHCLEN-2
                 00000004   002C            .=$$.TAB+XAB$L_NXT
                 00000000'  0004                .ADDRESS        <0>
                 0000002C   0008            .=$$.TABEND
                     002C   245 SCRIPT_FAB:     $FAB    XAB=SCRIPT_XAB  ; FAB                          [18]
                       03   002C                .BYTE FAB$C_BID
                       50   002D                .BYTE FAB$C_BLN
                 0000007C   002E                .BLKB FAB$C_BLN-2
                 00000030   007C                .=$$.TAB+FAB$L_FOP
                 00000000'  0030                .ADDRESS        $$.TMP
                 0000003C   0034                .=$$.TAB+FAB$L_ALQ
                 00000000'  003C                .ADDRESS        0
                     0000   0040                .WORD   0
                       00   0042                .BYTE   $$.TMP
                       00   0043                .BYTE   $$.TMP
                 00000000'  0044                .ADDRESS        0
                       00   0048                .BYTE   0
                       00   0049                .BYTE   FAB$C_SEQ
                       00   004A                .BYTE   $$.TMP
                       02   004B                .BYTE   FAB$C_VAR
                       00   004C                .BYTE   $$.TMP
                       00   004D                .BYTE   0
                     0000   004E                .WORD
                 00000000'  0050                .ADDRESS        SCRIPT_XAB
                 00000000'  0054                .ADDRESS        0
                 00000000'  0058                .ADDRESS        0
                 00000000'  005C                .ADDRESS        0
                       00   0060                .BYTE   0
                       00   0061                .BYTE   0
                     0000   0062                .WORD   0
                 00000000'  0064                .ADDRESS        0
                     0000   0068                .WORD   0
                       00   006A                .BYTE   0
                       00   006B                .BYTE   0
                 00000074   006C                .=$$.TAB+FAB$W_GBC
                     0000   0074                .WORD   0
                       00   0076                .BYTE   <<0aFAB$V_LNM_MODE> + <0aFAB$V_CHAN_MODE> + -
                 0000007C   0077            .=$$.TABEND
                     007C   246 SCRIPT_RAB:     $RAB    FAB=SCRIPT_FAB  ;                              [18]
                       01   007C                .BYTE RAB$C_BID
                       44   007D                .BYTE RAB$C_BLN
                 000000C0   007E                .BLKB RAB$C_BLN-2
                 00000080   00C0                .=$$.TAB+RAB$L_ROP
                 00000000'  0080                .ADDRESS        $$.TMP
                 00000094   0084                .=$$.TAB+RAB$L_CTX
                 00000000'  0094                .ADDRESS        0
                 0000009A   0098                .=$$.TAB+RAB$B_RAC
                       00   009A                .BYTE   RAB$C_SEQ
                       00   009B                .BYTE   0
                     0000   009C                .WORD   0
                     0000   009E                .WORD   0
                 00000000'  00A0                .ADDRESS        0
```

```
00000000' 00A4                  .ADDRESS        0
00000000' 00A8                  .ADDRESS        0
00000000' 00AC                  .ADDRESS        0
       00 00B0                  .BYTE   0
       00 00B1                  .BYTE   0
       00 00B2                  .BYTE   0
       00 00B3                  .BYTE   0
00000000' 00B4                  .ADDRESS        0
0000002C' 00B8                  .ADDRESS        SCRIPT_FAB
00000000' 00BC                  .ADDRESS        0
                00C0    247
                00C0    248 PRMT_ADR:
00000000        00C0    249             .LONG   0               ; HOLDS SOURCE ADDRESS FOR USER PROMPT          [22]
                00C4    250 BLD_PRMT:
00000000        00C4    251             .LONG   0               ; USED TO BUILD ON TO USER PROMPT 'ASK>'        [22]
                00C8    252 PRMT_LEN:
00000000        00C8    253             .LONG   0               ; HOLDS COUNT OF ASCIC STRING                   [22]
                00CC    254 L_LINK:
000000D0        00CC    255             .BLKL   1               ; ADDRESS (IF ANY) OF CURRENT SCRIPT DESCRIPTOR
                00D0    256
                00D0    257 L_FLAGS:
000000D4        00D0    258             .BLKL   1               ; FLAG BITS
                00D4    259
00000001        00D4    260             V_ECHOED=1              ; SET IF LINE HAS BEEN READ BUT NOT ECHOED
                00D4    261
                00D4    262 DS$GB_QIOACT::
       00       00D4    263             .BYTE   0               ; Low Bit set if input QIO active
                00D5    264                                     ; Used by PRINTx in user mode to cancel input
```

```
                     00D5   266          .SBTTL Absolute Psect Declarations
                     00D5   267          .SAVE
                     00D5   268          .PSECT  $ABS$, ABS
00000000             FE70   269 .=0
00000000  00000000   0000   270 SC$L_LINK:       .BLKL   1       ; ADDRESS (IF ANY) OF NEXT STACKED SCRIPT
00000004  00000004   0004   271                  .BLKL   1       ; UNUSED
00000008  00000008   0008   272 SC$W_SIZE:       .BLKW   1       ; SIZE OF ALLOCATED SPACE
0000000A  0000000A   000A   273 SC$W_LEN:        .BLKW   1       ; Length of current record SC$W_CURR
0000000C  0000000C   000C   274 SC$W_CURR:       .BLKW   1       ; OFFSET TO CURRENT RECORD IN BUFFER
0000000E  0000000E   000E   275 SC$W_END:        .BLKW   1       ; NUMBER OF BYTES IN THIS BUFFER
00000010  00000010   0010   276 SC$A_BUF:        .BLKB   0       ; OFFSET TO START OF TEXT
00000010  00000010   0010   277 SC$C_BLN=.
                  000000D5   278          .RESTORE
```

```
                    00D5   280          .SBTTL  SCRIPT$INIT -   Initialize script
                00000000   281          .Psect  Code, Shr, Exe, Nowrt, Byte
                    0000   282 ;++
                    0000   283 ; FUNCTIONAL DESCRIPTION:
                    0000   284 ;
                    0000   285 ;       This routine is called from KERNEL during initialization to
                    0000   286 ;       initialize the SCRIPT functionality.
                    0000   287 ;
                    0000   288 ; CALLING SEQUENCE:
                    0000   289 ;
                    0000   290 ;       BSBW    SCRIPT$INIT
                    0000   291 ;
                    0000   292 ; INPUT PARAMETERS:
                    0000   293 ;
                    0000   294 ;       NONE
                    0000   295 ;
                    0000   296 ; IMPLICIT INPUTS:
                    0000   297 ;
                    0000   298 ;       NONE
                    0000   299 ;
                    0000   300 ; OUTPUT PARAMETERS:
                    0000   301 ;
                    0000   302 ;       NONE
                    0000   303 ;
                    0000   304 ; IMPLICIT OUTPUTS:
                    0000   305 ;
                    0000   306 ;       NONE
                    0000   307 ;
                    0000   308 ; COMPLETION CODES:
                    0000   309 ;
                    0000   310 ;       NONE
                    0000   311 ;
                    0000   312 ; SIDE EFFECTS:
                    0000   313 ;
                    0000   314 ;       NONE
                    0000   315 ;--
```

```
                    0000   317 SCRIPT$INIT::
000000CC'EF   D4    0000   318         CLRL    L_LINK                  ; Clear link to first script
       0190   30    0006   319         BSBW    SCRIPT$STOP             ; Indicate no scripts active      [11]
              05    0009   320         RSB                             ; Return
```

```
                    000A   322          .SBTTL  SCRIPT$OPEN -   Initialize a script file
                0000000A   323          .Psect  Code, Shr, Exe, Nowrt, Byte
                    000A   324 ;++
                    000A   325 ; FUNCTIONAL DESCRIPTION:
                    000A   326 ;
                    000A   327 ;       This routine is called from CLI to process a afilespec command.
                    000A   328 ;       It allocates space for the necessary internal data structures,
                    000A   329 ;       Open the file specified and sets a flag that input should be
                    000A   330 ;       taken from the script file instead of the console.
                    000A   331 ;
                    000A   332 ; CALLING SEQUENCE:
                    000A   333 ;
                    000A   334 ;       BSBW    SCRIPT$OPEN
                    000A   335 ;
                    000A   336 ; INPUT PARAMETERS:
                    000A   337 ;
                    000A   338 ;       NONE
                    000A   339 ;
                    000A   340 ; IMPLICIT INPUTS:
                    000A   341 ;
                    000A   342 ;       CLI$GL_BASE + CLI$Q_FILE        Length and Address of filespec
                    000A   343 ;
                    000A   344 ; OUTPUT PARAMETERS:
                    000A   345 ;
                    000A   346 ;       NONE
                    000A   347 ;
                    000A   348 ; IMPLICIT OUTPUTS:
                    000A   349 ;
                    000A   350 ;       DS$V_SCRIPT                     May be set in DS$GL_FLAGS
                    000A   351 ;
                    000A   352 ; COMPLETION CODES:
                    000A   353 ;
                    000A   354 ;       RMS Standard completion codes
                    000A   355 ;
                    000A   356 ; SIDE EFFECTS:
                    000A   357 ;
                    000A   358 ;       NONE
                    000A   359 ;--
```

```
                                000A   SCRIPT$OPEN::
                    014A   30   000A           BSBW    SCRIPT$FLUSH              ; Flush scripts if console command      [11]
          00000000'EF      16   000D           JSB     INIT_CONTEXT              ; Refresh Stacks, AST,PCB,PHD etc       [07]
                                0013           Br_If_User      10$               ; If user mode, branch                  [12]
     0000FE00'EF    1C   E0   0013           BBS     #DSA$V_User, -                    ; If bit set, branch            [28]
                    07        001A                     L^DSA$GL_Flags, 10$             ;                               [28]
     00000000'EF    6C   FA   001B           CALLG   (AP),QIO$CLEANUP          ; Flush current QIO database
                                0022
                                0022   ;
                                0022   ; Set pointer to XAB
                                0022   ;
                                0022
     57   00000000'EF      9E   0022   10$:    MOVAB   SCRIPT_XAB,R7             ; SAVE ADDRESS OF XAB                   [18]
                                0029   ;
                                0029   ; Fill in FAB
                                0029   ;
     59   0000002C'EF      9E   0029           MOVAB   SCRIPT_FAB,R9             ; SAVE ADDRESS OF FAB                   [18]
     52   00000000'EF      9E   0030           MOVAB   DS$GL_CLIBASE, R2         ; Address of CLI database               [11]
        34 A9   08 A2      90   0037           MOVB    CLI$Q_FILE (R2), -        ; File name size                        [11]
                                003C                     FAB$B_FNS (R9)         ;                                       [11]
        2C A9   0C A2      D0   003C           MOVL    CLI$Q_FILE+4 (R2), -      ; File name address                     [11]
                                0041                     FAB$L_FNA (R9)         ;                                       [11]
           35 A9   04      90   0041           MOVB    #4, FAB$B_DNS (R9)        ; Default name size                     [11]
  30 A9   0000000E'EF      9E   0045           MOVAB   L^T_DEF_COM, -            ; Default name address                  [07]
                                004D                     FAB$L_DNA (R9)         ;                                       [11]
```

```
                        004D   385 ;+
                        004D   386 ; Set pointer to RAB
                        004D   387 ;-
          58  0000007C'EF   9E  004D   388          MOVAB   SCRIPT_RAB,R8           ; SAVE ADDRESS OF RAB                   [18]
                        0054   389 ;+
                        0054   390 ; $OPEN the file
                        0054   391 ;-
                        0054   392
                        0054   393          $OPEN   FAB = (R9)
                     69   DF  0054                  PUSHAL  (R9)
          00000000'GF    01   FB  0056                  CALLS   #$$.TMP1,G^SYS$OPEN
               00000000'EF   16  005D   394          JSB     DSR$COMPLETION          ; Print errors if any
                     03 50   E8  0063   395          BLBS    R0,70$                  ; Branch to continue
                      00E8   31  0066   396          BRW     SCRIPT$OPEN_X           ; Exit w/error
                        0069   397
                        0069   398 ;+
                        0069   399 ; Allocate buffer for script
                        0069   400 ;-
                        0069   401
       50    10 A7    09   78  0069   402 70$:     ASHL    #9,XAB$L_EBK(R7),R0     ; Get length of file
            51   14 A7   3C  006E   403          MOVZWL  XAB$W_FFB(R7),R1        ; Get first free byte
          51   10 A041   9E  0072   404          MOVAB   SC$C_BLN(R0)[R1],R1     ; Length required for script + a little
                   FF86'   30  0077   405          BSBW    EXE$ALONONPAGED         ; Allocate space for buffer
                     03 50   E8  007A   406          BLBS    R0, 71$                 ; Branch around if no error             [12]
                      0081   31  007D   407          BRW     160$                    ; Branch if error, Close and exit       [12]
                        0080   408
                     62   7C  0080   409 71$:     CLRQ    (R2)                    ; Clear first part of SC$...            [12]
                  08 A2   7C  0082   410          CLRQ    8(R2)                   ; Clear second part of SC$...
                     5B   52  D0  0085   411          MOVL    R2,R11                  ; base script control area
                08 AB   51  B0  0088   412          MOVW    R1,SC$W_SIZE(R11)       ; Set the length allocated
       6B   000000CC'EF   D0  008C   413          MOVL    L^L_LINK,SC$L_LINK(R11) ; Make current script the previous     [07]
          000000CC'EF   6B  DE  0093   414          MOVAL   (R11),L^L_LINK          ; Make this the current script          [07]
                        009A   415          Clear_StrFlg                    ; Clear started flag                    [12]
          00000000'EF   02  E4  009A                  BBSC    #DS$V_StrFlg, -         ; Branch if STRFLG bit is set to        [29]
                        00  00A1                            L^DS$GL_Flags, 30000$ ; ...                           [29]
                        00A2       30000$:                                      ; ... here. Clear bit regardless.       [29]
                        00A2   416          Clear_CtrlC                     ; Clear ^C flag                         [12]
          00000000'EF   00  E4  00A2                  BBSC    #DS$V_CtrlC, -          ; Branch if CTRLC bit is set to         [29]
                        00  00A9                            L^DS$GL_Flags, 30001$ ; ...                           [29]
                        00AA       30001$:                                      ; ... here. Clear bit regardless.       [29]
                  53   10 AB   9E  00AA   417          MOVAB   SC$A_BUF(R11),R3        ; Address of free space in buffer
               54   51   10   C3  00AE   418          SUBL3   #SC$A_BUF,R1,R4         ; Length left in buffer
```

```
                                00B2    420 ;+
                                00B2    421 ; Connect to record stream
                                00B2    422 ;-
                                00B2    423
                                00B2    424         $CONNECT RAB = (R8)
                          68    DF    00B2                PUSHAL  (R8)
00000000'GF               01    FB    00B4                CALLS   #$$.TMP1,G^SYS$CONNECT
                       43 50    E9    00BB    425         BLBC    R0,160$                 ; Branch if error, Close and exit
                                00BE    426
   24 A8               01 A3    9E    00BE    427 100$:   MOVAB   1(R3), RAB$L_UBF(R8)    ; Set address to put data
   20 A8               FF 8F    9B    00C3    428         MOVZBW  #255, RAB$W_USZ(R8)     ; Limit size to 255 characters
54    00000100         8F    D1    00C8    429         CMPL    #256, R4                ; Room for count byte and 255 characters?
                          05    15    00CF    430         BLEQ    110$                    ; Branch if there is
 20 A8    54            01    A3    00D1    431         SUBW3   #1, R4, RAB$W_USZ(R8)   ; Shorten transfer size
                                00D6    432
                                00D6    433 110$:   $GET    RAB = (R8)              ; $GET a record
                          68    DF    00D6                PUSHAL  (R8)
00000000'GF               01    FB    00D8                CALLS   #$$.TMP1,G^SYS$GET
                       13 50    E9    00DF    434         BLBC    R0, 130$                ; Branch if error, includes EOF, RTB
         50            22 A8    3C    00E2    435         MOVZWL  RAB$W_RSZ(R8), R0       ; Get actual record length
               63      50    90    00E6    436         MOVB    R0, (R3)                ; Set length of record
      53       01 A340    9E    00E9    437         MOVAB   1(R3)[R0], R3           ; Set new free
               54      50    C2    00EE    438         SUBL    R0, R4                  ; Decrement length remaining
                       54    D7    00F1    439         DECL    R4                      ; Less one for length
                       C9    11    00F3    440         BRB     100$                    ; Try again
```

```
                                00F5   442 ;+
                                00F5   443 ; Error, EOF, or Record too big on GET
                                00F5   444 ;-
                                00F5   445
                     52   50    D0 00F5   446 130$:   MOVL    R0,R2                   ; Save return code
                                00F8   447            $DISCONNECT     RAB = (R8)
                          68    DF 00F8              PUSHAL  (R8)
              00000000'GF   01    FB 00FA              CALLS   #$$.TMP1,G^SYS$DISCONNECT
                                0101   448
                                0101   449 160$:   $CLOSE  FAB = (R9)
                          69    DF 0101              PUSHAL  (R9)
              00000000'GF   01    FB 0103              CALLS   #$$.TMP1,G^SYS$CLOSE
                     50   52    D0 010A   450         MOVL    R2,R0                   ; Restore error code
              0001827A 8F   50    D1 010D   451         CMPL    R0, #RMS$_EOF           ; End of file?
                           0B    13 0114   452         BEQL    180$                    ; Branch if legit EOF
                   00000000'EF   16 0116   453         JSB     DSR$COMPLETION          ; Report the error
                         0096    30 011C   454         BSBW    SCRIPT$FINISH           ; Error, flush the script
                                011F   455
                           30    11 011F   456 170$:   BRB     SCRIPT$OPEN_X           ; Exit
                                0121   457
                                0121   458 ;+
                                0121   459 ; EOF
                                0121   460 ;-
                                0121   461
                     50   53    D0 0121   462 180$:   MOVL    R3, R0                  ; Free address in buffer
                51   50   5B    C3 0124   463         SUBL3   R11, R0, R1             ; Length used
              0E AB  51   10    A3 0128   464         SUBW3   #SC$A_BUF, R1, SC$W_END(R11); Save length of data
                   51  0F  A1    9E 012D   465         MOVAB   15(R1), R1              ; Round up to block boundry
              08 AB  51   0F    AB 0131   466         BICW3   #15, R1, SC$W_SIZE(R11) ; Truncate to block boundry and save
                     54   0F    AA 0136   467         BICW    #15, R4                 ; Length remaining to block boundry
                           0E    13 0139   468         BEQL    190$                    ; None,  finish
                50   0F  A0    9E 013B   469         MOVAB   15(R0), R0              ; Round up to block boundry
                     50   0F    CA 013F   470         BICL    #15, R0                 ; Finish rounding
                  08 A0   54    B0 0142   471         MOVW    R4, 8(R0)               ; Set size of section to be released
                        FEB7'   30 0146   472         BSBW    EXE$DEANONPAGED         ; Release the unused portion
                                0149   473
                                0149   474 190$:   Set_Script                         ; Set to script mode and exit          [12]
              00000000'EF   15    E2 0149              BBSS    #DS$V_Script, -         ; Branch if SCRIPT bit is set to       [29]
                           00         0150                        L^DS$GL_Flags, 30002$ ; ...                              [29]
                                0151       30002$:                                   ; ... here. Set bit regardless.        [29]
                                0151   475
                                0151   476 SCRIPT$OPEN_X:
                   00000000'EF   17 0151   477         JMP     BEGIN                   ; And force reentry of CLI
```

```
                     0157   479          .SBTTL  SCRIPT$FLUSH, STOP, CONT -      Terminate all scripts open
                  00000157   480          .Psect  Code, Shr, Exe, Nowrt, Byte
                     0157   481 ;++
                     0157   482 ; FUNCTIONAL DESCRIPTION:
                     0157   483 ;
                     0157   484 ;       SCRIPT$FLUSH    Will SCRIPT$FINISH until L_LINK is zero.
                     0157   485 ;       SCRIPT$STOP     Clears DS$V_SCRIPT
                     0157   486 ;       SCRIPT$CONT     Sets DS$V_SCRIPT if L_LINK NEQ 0
                     0157   487 ;
                     0157   488 ; CALLING SEQUENCE:
                     0157   489 ;
                     0157   490 ;       BSBW    SCRIPT$FLUSH
                     0157   491 ;       BSBW    SCRIPT$STOP
                     0157   492 ;       BSBW    SCRIPT$CONT
                     0157   493 ;
                     0157   494 ; INPUT PARAMETERS:
                     0157   495 ;
                     0157   496 ;       NONE
                     0157   497 ;
                     0157   498 ; IMPLICIT INPUTS:
                     0157   499 ;
                     0157   500 ;       Current DS$GL_FLAGS and script OWN data
                     0157   501 ;
                     0157   502 ; OUTPUT PARAMETERS:
                     0157   503 ;
                     0157   504 ;       NONE
                     0157   505 ;
                     0157   506 ; IMPLICIT OUTPUTS:
                     0157   507 ;
                     0157   508 ;       NONE
                     0157   509 ;
                     0157   510 ; COMPLETION CODES:
                     0157   511 ;
                     0157   512 ;       SS$_NORMAL
                     0157   513 ;
                     0157   514 ; SIDE EFFECTS:
                     0157   515 ;
                     0157   516 ;       NONE
                     0157   517 ;--
```

```
                        0157   519 SCRIPT$FLUSH::
06   0000FE17'EF   91   0157   520          CMPB    DSA$GL_SID+3,#PR$_sid_typ8NN          ; if we are nautilus [26] ;G
                09   13   015E   521          BEQL    3$                                    ; then 3$                 ;G:8
11   0000FE17'EF   91   0160   522          CMPB    DSA$GL_SID+3,#PR$_sid_typ8RR          ; if we are not RRR [26] ;G:
                15   12   0167   523          BNEQ    5$                                    ; then 5$                 ;G:8
                        0169   524 3$:       BR_IF_USER      5$                            ; else if nautilus or     ;G:8
0000FE00'EF     1C   E0   0169                BBS     #DSA$V_User, -                ; If bit set, branch              [28]
                0D           0170                  L^DSA$GL_Flags, 5$                  ;                                 [28]
                        0171   525                                                        ; RRR, and user mode then 5$
                        0171   526          $DS_CLRVEC_S -                        ; clr interval timer vector
                        0171   527                  #SCB$L_TIMER
     000000C0   8F   DD   0171                  PUSHL   #SCB$L_TIMER
00000000'9F     01   FB   0177                  CALLS   #1, @#DS$CLRVEC
                        017E   528                                                ; Removed call to CLK$RE_INIT   [27]  ;G:9
                        017E   529 5$:
                        017E   530          Clear_QA                              ; Turn off the QA flag            [12]
0000FE00'EF     0F   E4   017E                BBSC    #DSA$V_QA, -                  ; Branch if QA bit is set to      [29]
                00   0185                    L^DSA$GL_Flags, 30003$                ; ...
                        0186       30003$:                                        ; ... here. Clear bit regardless. [29]
                        0186   531
                        0186   532 10$:      Br_If_Script    20$                   ; Branch if script mode           [12]
00000000'EF     15   E0   0186                BBS     #DS$V_Script, -               ; If bit set, branch              [28]
                0A           018D                  L^DS$GL_Flags, 20$                  ;                                 [28]
                25   10   018E   533          BSBB    SCRIPT$FINISH                 ; YES, FINISH TOP LEVEL           [07]
     000000CC'EF     D5   0190   534          TSTL    L^L_LINK                      ; ANY SCRIPTS ACTIVE?             [07]
                EE   12   0196   535          BNEQ    10$                           ; AND TRY AGAIN                   [10]
                        0198   536
                05   0198   537 20$:      RSB
                        0199   538
                        0199   539
                        0199   540 SCRIPT$STOP::
                        0199   541          Clear_Script                          ; Clear script flag               [12]
00000000'EF     15   E4   0199                BBSC    #DS$V_Script, -               ; Branch if SCRIPT bit is set to  [29]
                00   01A0                    L^DS$GL_Flags, 30004$               ; ...                             [29]
                        01A1       30004$:                                        ; ... here. Clear bit regardless. [29]
                05   01A1   542          RSB                                   ;                                 [12]
                        01A2   543
                        01A2   544
                        01A2   545 SCRIPT$CONT::
                F5   10   01A2   546          BSBB    SCRIPT$STOP                   ; Clear script flag
     000000CC'EF     D5   01A4   547          TSTL    L^L_LINK                      ; Any scripts present?            [07]
                08   13   01AA   548          BEQL    10$                           ; Branch if not
                        01AC   549          Set_Script                            ; Set script flag                 [12]
00000000'EF     15   E2   01AC                BBSS    #DS$V_Script, -               ; Branch if SCRIPT bit is set to  [29]
                00   01B3                    L^DS$GL_Flags, 30005$               ; ...                             [29]
                        01B4       30005$:                                        ; ... here. Set bit regardless.   [29]
                        01B4   550
                05   01B4   551 10$:      RSB
```

```
                        01B5    553          .SBTTL  SCRIPT$FINISH - Terminate script input file
                 000001B5    554          .Psect  Code, Shr, Exe, Nowrt, Byte
                        01B5    555 ;++
                        01B5    556 ; FUNCTIONAL DESCRIPTION:
                        01B5    557 ;
                        01B5    558 ;       This routine will $CLOSE the current script file (if any) and return the
                        01B5    559 ;       space allocated. If this is the last script file, it will clear the
                        01B5    560 ;       DS$V_SCRIPT flag in DS$GL_FLAGS.
                        01B5    561 ;
                        01B5    562 ; CALLING SEQUENCE:
                        01B5    563 ;
                        01B5    564 ;       BSBW    SCRIPT$FINISH
                        01B5    565 ;
                        01B5    566 ; INPUT PARAMETERS:
                        01B5    567 ;
                        01B5    568 ;       NONE
                        01B5    569 ;
                        01B5    570 ; IMPLICIT INPUTS:
                        01B5    571 ;
                        01B5    572 ;       Current DS$GL_FLAGS and Script's OWN data.
                        01B5    573 ;
                        01B5    574 ; OUTPUT PARAMETERS:
                        01B5    575 ;
                        01B5    576 ;       NONE
                        01B5    577 ;
                        01B5    578 ; IMPLICIT OUTPUTS:
                        01B5    579 ;
                        01B5    580 ;       DS$GL_SCRIPT and OWN DATA
                        01B5    581 ;
                        01B5    582 ; COMPLETION CODES:
                        01B5    583 ;
                        01B5    584 ;       SS$_NORMAL
                        01B5    585 ;
                        01B5    586 ; SIDE EFFECTS:
                        01B5    587 ;
                        01B5    588 ;       NONE
                        01B5    589 ;--
```

```
                                01B5   SCRIPT$FINISH:
                    3C    BB    01B5           PUSHR   #^M<R2, R3, R4, R5>      ; Save registers
                                01B7
      50   000000CC'EF    D0    01B7   10$:    MOVL    L^L_LINK, R0             ; GET POINTER TO CURRENT SCRIPT          [07]
                    12    13    01BE           BEQL    30$                      ; NONE, EXIT RIGHT AWAY
      000000CC'EF   60    D0    01C0           MOVL    SC$L_LINK(R0),L^L_LINK   ; MAKE NEXT CURRENT                      [07]
                 FE36'    30    01C7           BSBW    EXE$DEANONPAGED          ; DEALLOCATE IT
                                01CA
           000000CC'EF    D5    01CA   20$:    TSTL    L^L_LINK                 ; ANY CURRENT SCRIPT?                    [07]
                    02    12    01D0           BNEQ    40$                      ; YES, JUST EXIT
                                01D2
                    C5    10    01D2   30$:    BSBB    SCRIPT$STOP              ; Clear script active flag               [11]
                                01D4
               50   00'    D0    01D4   40$:    MOVL    S^#SS$_NORMAL, R0        ; IT WORKED!
                    3C    BA    01D7           POPR    #^M<R2, R3, R4, R5>      ; Restore registers
                          05    01D9           RSB                              ; RETURN
```

```
                    01DA   608          .SBTTL  DS_GETLINE -    Get program data input line
                000001DA   609          .Psect  Code, Shr, Exe, Nowrt, Byte
                    01DA   610 ;**
                    01DA   611 ; FUNCTIONAL DESCRIPTION:
                    01DA   612 ;
                    01DA   613 ;       This routine returns a line from the input stream. It may be from the consol
                    01DA   614 ;       from a script file. If the first character of the user prompt from ASK$????
                    01DA   615 ;       <NULL> the operator is prompted directly without consulting the script.
                    01DA   616 ;       Syncronization between the scripts and DS_??LINE is accomplished by matching
                    01DA   617 ;       prompt string with the line in the script file.  If it does not match and
                    01DA   618 ;       DS_GETLINE was called the program is aborted, If DS_SUPERLINE was called the
                    01DA   619 ;       is ignored and the next line is examined. The prompt string and leading spac
                    01DA   620 ;       tabs are removed from the script line before it is returned to the program.
                    01DA   621 ;       addition lower case characters in the script are translated to uppercase. If
                    01DA   622 ;       input line is expected from the operator and the OPERTOR flag is clear a nul
                    01DA   623 ;       line is returned.  Currently the ASK$???? routines apply the program specifi
                    01DA   624 ;       default value in the case of a null line.
                    01DA   625 ;
                    01DA   626 ; CALLING SEQUENCE:
                    01DA   627 ;
                    01DA   628 ;       CALL    DS_GETLINE
                    01DA   629 ;       CALL    DS_SUPERLINE
                    01DA   630 ;
                    01DA   631 ; INPUT PARAMETERS:
                    01DA   632 ;
                    01DA   633 ;        4(AP) -        STRING BUFFER ADDRESS
                    01DA   634 ;        8(AP) -        LONGWORD FOR RETURNED STRING LENGTH
                    01DA   635 ;       12(AP) -        STRING BUFFER SIZE
                    01DA   636 ;       16(AP) -        Address of ASCIC user prompt
                    01DA   637 ;       20(AP) -        Address of ASCIC extra prompt text
```

```
01DA    639 ; IMPLICIT INPUTS:
01DA    640 ;
01DA    641 ;       NONE
01DA    642 ;
01DA    643 ; OUTPUT PARAMETERS:
01DA    644 ;
01DA    645 ;       NONE
01DA    646 ;
01DA    647 ; IMPLICIT OUTPUTS:
01DA    648 ;
01DA    649 ;       NONE
01DA    650 ;
01DA    651 ; COMPLETION CODES:
01DA    652 ;
01DA    653 ;       SS$_NORMAL
01DA    654 ;
01DA    655 ; REGISTER USAGE:
01DA    656 ;
01DA    657 ;       R4      Length of current line in script
01DA    658 ;       R5      Address of current line in script
01DA    659 ;       R6      Flags
01DA    660 ;                 bit 0 set if supervisor request, else clear
01DA    661 ;                 bit 1 set if already printed "prompt not found text"
01DA    662 ;       R7      Address of ASCIC user prompt
01DA    663 ;       R8      address of informational prompt
01DA    664 ;       R11     Base of current script block
01DA    665 ;
01DA    666 ; SIDE EFFECTS:
01DA    667 ;
01DA    668 ;       R1 = RETURNED STRING LENGTH
01DA    669 ;--
```

```
                          0BFC  01DA   671          .ENTRY  DS_SUPERLINE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R11>
                                01DC   672
                    56    01    D0   01DC   673          MOVL    #1, R6                  ; Flag supervisor request
                          53    11   01DF   674          BRB     COMMON                  ; Join at common
                                01E1   675
                          0BFC  01E1   676          .ENTRY  DS_GETLINE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R11>
                                01E3   677
                00000000'EF     D5   01E3   678          TSTL    PRNT_PRMT               ; See if ASK> is to be printed
                          47    13   01E9   679          BEQL    10$                     ; If zero branch around
                                01EB   680
                                01EB   681
                   7E   10 AC   D0   01EB   682          MOVL    16(AP),-(SP)            ; Address of prompt descriptor
                          6E    D6   01EF   683          INCL    (SP)                    ; Pass count, get to address of prompt
                   7E   10 BC   9A   01F1   684          MOVZBL  a16(AP),-(SP)           ; Mov count of prompt to use as length
000000C5'EF     04 BE    6E    28   01F5   685          MOVC3   (SP),a4(SP),BLD_PRMT+1  ; Length # of char., moved to buffer
                                01FE   686
               04 AE    10 BC   9A   01FE   687          MOVZBL  a16(AP),4(SP)           ; Length of ASK prompt
          04 AE    000000C5'8F   C0   0203   688          ADDL2   #BLD_PRMT+1,4(SP)       ; Add count of prompt to start address
             6E    00000007'EF   9A   020B   689          MOVZBL  FMT_PRMT,(SP)           ; Count of $DSASKxxx prompt
04 BE        00000008'EF     6E   28   0212   690          MOVC3   (SP),FMT_PRMT+1,a4(SP)  ; Mov ASK> into buffer
                          8E    7C   021B   691          CLRQ    (SP)+                   ; Clear the data off stack
          00000007'EF   10 BC   81   021D   692          ADDB3   a16(AP),FMT_PRMT,BLD_PRMT; Put added counts into buffer
                000000C4'EF          0225
          10 AC 000000C4'EF     DE   022A   693          MOVAL   BLD_PRMT,16(AP)         ; Buffer address into expected position
                                0232   694
                                0232   695
                          56    D4   0232   696 10$:     CLRL    R6                      ; Flag user request
                                0234   697
                                0234   698
                                0234   699 COMMON:
                    59    56    D0   0234   700          MOVL    R6,R9                   ; Save r6                                [20]
57      00000012'EF     9E   0237   701          MOVAB   L^T_NULL, R7            ; Initially a null user prompt           [07]
                    58    57    D0   023E   702          MOVL    R7, R8                  ; Null information prompt
                    04    6C    D1   0241   703          CMPL    (AP), #4                ; Is user prompt address present?
                          0D    1F   0244   704          BLSSU   10$                     ; Branch if not
                   57   10 AC   D0   0246   705          MOVL    16(AP), R7              ; Yes use it
                       05    6C   D1   024A   706          CMPL    (AP), #5                ; Is user info prompt present?
                          04    1F   024D   707          BLSSU   10$                     ; Branch if not
                   58   14 AC   D0   024F   708          MOVL    20(AP), R8              ; Yes, use it
                                0253   709
                                0253   710 10$:     Br_If_Not_Script        20$     ; Branch if using console now            [12]
00000000'EF             15    E1   0253          BBC     #DS$V_Script, -                 ; If bit not set, branch         [28]
                          12         025A                    L^DS$GL_Flags, 20$      ;                                [28]
5B      000000CC'EF     D0   025B   711          MOVL    L^L_LINK, R11           ; Any script active?                     [07]
                          09    13   0262   712          BEQL    20$                     ; Branch if no script active
                                0264   713
                          67    95   0264   714          TSTB    (R7)                    ; null user prompt?
                          08    13   0266   715          BEQL    30$                     ; Yes, continue
                       01 A7    95   0268   716          TSTB    1(R7)                   ; Is first character of prompt null?
                          03    12   026B   717          BNEQ    30$                     ; Branch if can be scripted
                                026D   718
                        00B9    31   026D   719 20$:     BRW     CONSOLE                 ; Branch to use console
                                0270   720
                        035F    30   0270   721 30$:     BSBW    ADVANCE                 ; Point to next record
                      0F 50     E8   0273   722          BLBS    R0, 40$                 ; Branch if not <EOF>
55      00000013'EF     9E   0276   723          MOVAB   L^T_EOF, R5             ; Text for "a <EOF>"                     [07]
                    54    85    9A   027D   724          MOVZBL  (R5)+, R4               ; length of "a <EOF>"
```

```
                0085   30  0280   725          BSBW    PRINT_LINE              ; Type prompt and text--CH TO WORD DISP
                  AF   11  0283   726          BRB     COMMON                  ; And try again
```

```
                      54   D5  0285   728 40$:    TSTL    R4                          ; Any characters in line?
                      09   13  0287   729         BEQL    50$                         ; no, use it as is
                   65 21   91  0289   730         CMPB    #^A"!", (R5)                ; Comment line?
                      04   12  028C   731         BNEQ    50$                         ; No, use it
                      78   10  028E   732         BSBB    PRINT_LINE                  ; Type prompt and text
                      A2   11  0290   733         BRB     COMMON                      ; Try again
                           0292   734
                      53   57  D0   0292   735 50$:    MOVL    R7, R3                      ; Get address of user prompt
                      52   83  9A   0295   736         MOVZBL  (R3)+, R2                   ; Get length of ASCIC string
                         FD65' 30   0298   737         BSBW    SCAN$COMPARE                ; Compare the strings
                      52   D5  029B   738         TSTL    R2                          ; Check for exhausted prompt string
                      4A   13  029D   739         BEQL    80$                         ; Branch if prompt match
                   15 56   01  E2   029F   740         BBSS    #1, R6, 60$                 ; Branch if already told of error
                           02A3   741
                      57   DD  02A3   742         PUSHL   R7                          ; Address of prompt string
                0000004E'EF   9F  02A5   743         PUSHAB  L^T_PNF                     ; Edit string for "Prompt Not Found"    [07]
                   7E   01   9A  02AB   744         movzbl  #DS$K_PrintF,-(SP)          ; Routine name constant                 [14]
                   7E   1E   98  02AE   745         cvtbl   #DS$K_Type_Script_Pnf,-(SP)     ; The typecode constant             [14]
             00000000'GF   04   FB  02B1   746         CALLS   #4, G^DSX$Print             ; Type it                               [14]
                           02B8   747
                   22 56   E9  02B8   748 60$:    BLBC    R6, 70$                     ; Branch if user request
                50   0C AB   3C  02BB   749         MOVZWL  SC$W_CURR(R11), R0          ; Get current offset
                   10 AB40   9F  02BF   750         PUSHAB  SC$A_BUF(R11)[R0]           ; Address of string
                7E   0A AB   3C  02C3   751         MOVZWL  SC$W_LEN(R11), -(SP)        ; Length of string
                00000031'EF   9F  02C7   752         PUSHAB  L^T_SKIP                    ; Edit string for "skipping"            [07]
                   7E   01   9A  02CD   753         movzbl  #DS$K_printf,-(sp)          ; Routine                               [14]
                   7E   1F   98  02D0   754         cvtbl   #DS$K_type_script_skip,-(sp)    ; Print code.                       [14]
             00000000'GF   05   FB  02D3   755         CALLS   #5, G^DSX$PRINT             ; Print it                              [14]
                      FF57   31  02DA   756         BRW     COMMON                      ; Branch to try again
                           02DD   757
                0A AB   0A AB   B2  02DD   758 70$:    MCOMW   SC$W_LEN(R11), -
                           02E2   759                         SC$W_LEN(R11)           ; Complement len so record is used again
             00000000'9F   6C   FA  02E2   760         CALLG   (AP), @#DS$ABORT            ; Program request...abort program
                           02E9   761
                      FD14' 30   02E9   762 80$:    BSBW    SCAN$SPACES                 ; Remove spaces and tabs from input
                           02EC   763                                                 ;                                       [17]
                5C5C 8F   65   B1  02EC   764         CMPW    (R5),#^X5C5C                ; CHECK FOR // AT THE END OF PROMPT     [17]
                      03   12  02F1   765         BNEQ    90$                         ; IF NOT CONTINUE SCRIPT
                   0033   31  02F3   766         BRW     CONSOLE                     ; IF YES PRINT PROMPT AND WAIT FOR ANSWER
                           02F6   767 90$:                                            ;
                      10   10  02F6   768         BSBB    PRINT_LINE                  ; Type it
                   02A7   30  02F8   769         BSBW    COPY_IT                     ; Copy input to buffer
                           02FB   770
                51   08 BC   D0  02FB   771         MOVL    @8(AP), R1                  ; Get length of returned line
                   50   00'  D0  02FF   772         MOVL    S^#SS$_NORMAL, R0           ; It worked!
                           0302   773
                0000059B'EF   17  0302   774         JMP     DS_GETLINE_X_CHECK          ; Jump to exit :make sure interlock is clear
                           0308   775                                                    ; and then return to caller
```

```
                                0308   777 ;+
                                0308   778 ; Little subroutine to print a line iff the Verify bit is set.
                                0308   779 ;-
                                0308   780
                                0308   781 PRINT_LINE:
                                0308   782         Br_If_Not_Verify        10$     ; Skip output if VERIFY is clear        [12]
          0000FE00'EF   09   E1 0308           BBC     #DSA$V_Verify, -              ; If bit not set, branch                [29]
                        18      030F             L^DSA$GL_Flags, 10$         ;                                       [29]
                   7E   54   7D 0310   783      MOVQ    R4, -(SP)                   ; Push length/address of data
                        57   DD 0313   784      PUSHL   R7                          ; Push address of ASCIC user prompt
                0000001B'EF   9F 0315   785      PUSHAB  L^T_PROMPT                  ; Type prompt and returned line         [07]
                   7E   01   9A 031B   786      movzbl  #DS$K_printf,-(sp)          ; routine.                              [14]
                   7E   20   98 031E   787      cvtbl   #DS$K_type_script_prompt,-(sp)  ; code                              [14]
          00000000'GF   06   FB 0321   788      CALLS   #6, G^DSX$PRINT             ; Type prompt and returned line         [14]
                                0328   789
                                05 0328   790 10$:    RSB                         ; Return                                [08]
```

```
                          0329    792 ;+
                          0329    793 ; No DS script currently active; get input from SYS$INPUT
                          0329    794 ;-
                          0329    795
                          0329    796 CONSOLE:
     51  00000024'EF  9E  0329    797          MOVAB   L^T_PROMPT2, R1          ; Address of edit string                [07]
         50  81  9A      0330    798          MOVZBL  (R1)+, R0                ; Get length of string
         7E  50  7D      0333    799          MOVQ    R0, -(SP)                ; Push descriptor
  00000000'EF  9F        0336    800          PUSHAB  L^DS$GT_BUFFER           ; Address of buffer                     [07]
      7E  0000'8F  3C    033C    801          MOVZWL  #DS$K_BUFSIZ, -(SP)      ; Length of output buffer
         52  D4          0341    802          clrl    r2                       ; Indicator to save space               [09]
                          0343    803          Br_If_Not_Binary        20$      ; If not BINARY, branch around          [12]
0000FE00'EF  01  E1      0343                 BBC     #DSA$V_Binary, -                 ; If bit not set, branch        [29]
             23          034A                   L^DSA$GL_Flags, 20$            ;                               [29]
      50  04  AE  D0     034B    804          movl    4(sp), r0                ; Get address of buffer                 [09]
         60  02  90        034F    805          movb    #ds$k_type_user_prompt, -; Set first byte to the type code ...   [09]
                          0352    806                    (r0)                  ; ... for general prompt                [09]
             12  56  E9    0352    807          blbc    r6, 10$                  ; Branch if it's not DS>                [09]
         67  04  91        0355    808          cmpb    #4, (r7)                 ; Compare length to "DS> "              [09]
             0D  12        0358    809          bneq    10$                      ; If not, can't be DS>                  [09]
00000001'EF  01  A7      035A    810          CMPL    1(R7), DS$GT_PROMPT+1    ; is it really "DS>"                    [25]
             03  12        0362    811          bneq    10$                      ; Skip if not                           [09]
         60  01  90        0364    812          movb    #ds$k_type_ds_prompt, - ; Otherwise, load ...                   [09]
                          0367    813                    (r0)                  ; ... DS prompt code                    [09]
                          0367    814
             6E  D7       0367    815 10$:     decl    (sp)                     ; Don't let FAO overwrite the type      [09]
          04  AE  D6      0369    816          incl    4(sp)                    ;                                       [09]
             52  D6        036C    817          incl    r2                       ; Remember that BINARY is set           [09]
                          036E    818
         7E  57  7D       036E    819 20$:     MOVQ    R7, -(SP)                ; Push user prompt, info prompt         [09]
          08  AE  7F      0371    820          PUSHAQ  8(SP)                    ; Address of buffer descriptor
          0C  AE  3F      0374    821          PUSHAW  12(SP)                   ; Address to store final length
          18  AE  7F      0377    822          PUSHAQ  24(SP)                   ; Address of descriptor of edit string
00000000'9F  05  FB      037A    823          CALLS   #5, @#SYS$FAO            ; Edit prompt
         54  52  E9      0381    824          blbc    r2, 60$                  ; Branch if not BINARY                  [09]
             6E  D6      0384    825          incl    (sp)                     ; Now include type code in string       [09]
          04  AE  D7      0386    826          decl    4(sp)                    ;                                       [09]
```

```
                                0389    828 ;
                                0389    829 ; DEBUG ***
                                0389    830 ;
          48 00000000'EF   E9   0389    831          blbc    L^Debug_the_sucker, 50$ ; Skip debug stuff if flag clear        [99]
                        1F   BB   0390    832          pushr   #^M<r0, r1, r2, r3, r4> ; Make debug transparent                [99]
                     54 6E   9E   0392    833          movab   (sp), r4                ; Save current sp                       [99]
                        12   11   0395    834          brb     40$                     ; Skip format string                    [99]
                                0397    835
5A 21 3D 70 74 5B 0000039F'010E0000'    0397    836 30$:     .ascid  "[tp=!ZB]!/"            ; Format string                 [12]
                     2F 21 5D 42        03A5
                                03A9    837
                     5E 0A   C2   03A9    838 40$:     subl2   #10, sp                 ; Allocate space on stack               [99]
                        6E   9F   03AC    839          pushab  (sp)                    ; Set it's address                      [99]
                     7E 0A   9A   03AE    840          movzbl  #10, -(sp)              ; And it's length                       [99]
                     53 6E   9E   03B1    841          movab   (sp), r3                ; And remember it                       [99]
                  7E    18 B4   9A   03B4    842          movzbl  @24(r4), -(sp)          ; Final argument for FAO                [99]
                        18 B4   94   03B8    843          clrb    @24(r4)                 ; Clear it out                          [99]
                        63   9F   03BB    844          pushab  (r3)                    ; Where to put length                   [99]
                        63   9F   03BD    845          pushab  (r3)                    ; Address of result buffer              [99]
                     D5 AF   9F   03BF    846          pushab  30$                     ; Address of format string              [99]
             00000000'GF    04   FB   03C2    847          calls   #4, G^sys$fao           ; FAO it                                [99]
                     7E 63   7D   03C9    848          movq    (r3), -(sp)             ; Push address of result string         [99]
             00000000'EF    02   FB   03CC    849          calls   #2, dsx$type_out        ; Print it                              [99]
                     5E 64   9E   03D3    850          movab   (r4), sp                ; Restore initial stack                 [99]
                        1F   BA   03D6    851          popr    #^M<r0, r1, r2, r3, r4> ; Restore registers                     [99]
                                03D8    852
                                03D8    853 50$:                                    ;                                       [99]
                                03D8    854
                                03D8    855 ;
                                03D8    856 ; DEBUG ***
                                03D8    857 ;
                                03D8    858
                     7E 7C   03D8    859 60$:     CLRQ    -(SP)                   ; Space for IOSB                        [09]
                  21 56 E8   03DA    860          BLBS    R6, 70$                 ; Branch if command mode
                             03DD    861          Br_If_Oper      70$             ; If operator present, branch around    [12]
0000FE00'EF    0C   E0   03DD                    BBS     #DSA$V_Oper, -                  ; If bit set, branch            [29]
                  19        03E4                      L^DSA$GL_Flags, 70$           ;                               [29]
                     08 AE 7F   03E5    862          PUSHAQ  8(SP)                   ; Address of prompt descriptor
                0000002B'EF 9F   03E8    863          PUSHAB  L^T_PROMPT3             ; Edit string address                   [07]
                     7E 01 9A   03EE    864          movzbl  #DS$K_printf,-(sp)      ; Routine                               [14]
                     7E 20 98   03F1    865          cvtbl   #DS$K_type_script_prompt,-(sp) ; Code.                          [14]
00000000'GF    04   FB   03F4    866          CALLS   #4, G^DSX$PRINT         ; Type it ...                           [14]
                  0192 31   03FB    867          BRW     DS_GETLINE_OKAY_X       ; ... and exit w/null input line        [11]
```

```
                          03FE   869 ;
                          03FE   870 ; Separate flow into two primary cases: if DS$V_BATCH bit is set, then
                          03FE   871 ; SYS$INPUT is a file, and we are in user mode: utilize RMS routines to
                          03FE   872 ; manipulate file.  Otherwise, use QIO since it must run in standalone.
                          03FE   873 ;
                          03FE   874
                          03FE   875 70$:    Br_If_Not_Batch 100$              ; Branch if terminal or mailbox control [12]
        00000000'EF   16  E1    03FE               BBC     #DS$V_Batch, -       ; If bit not set, branch                [28]
                      61        0405                 L^DS$GL_Flags, 100$        ;                                       [28]
                          0406   876
                          0406   877 ;+
                          0406   878 ; RMS SYS$INPUT processing.  Use $GET to read record, and 'fake' echo
                          0406   879 ; via $PUT (PRINT call) to SYS$OUTPUT
                          0406   880 ;-
                          0406   881
    53  00000000'EF       9E    0406   882         MOVAB   DS$RAB_OUTPUT, R3       ; Point to output RAB
    52  00000000'EF       9E    040D   883         MOVAB   DS$RAB_INPUT, R2        ; Point to input RAB
          54    08 AE     9E    0414   884         MOVAB   8(SP), R4               ; Save pointer to prompt descriptor
       24 A2    04 AC     D0    0418   885         MOVL    4(AP), RAB$L_UBF(R2)    ; Load input buffer address
       20 A2    0C AC     B0    041D   886         MOVW    12(AP), RAB$W_USZ(R2)   ;  And size of same
                          0422   887         $GET    (R2)                    ; Read a record
                      62    DF    0422         PUSHAL  (R2)
        00000000'GF   01    FB    0424         CALLS   #$$.TMP1,G^SYS$GET
                   11 50    E8    042B   888         BLBS    R0, 90$                 ; Exit if error
        0001827A 8F   50    D1    042E   889         CMPL    R0, #RMS$_EOF           ; If this is end-of-file time
                      05    12    0435   890         BNEQ    80$                     ; :then
            50  0000'8F    3C    0437   891         MOVZWL  #SS$_ENDOFFILE, R0      ; ::pretend it was SS end-of-file
                          043C   892
                    0154  31    043C   893 80$:    BRW     DS_GETLINE_X            ; Exit routine                          [11]
                          043F   894
              50    22 A2   3C    043F   895 90$:    MOVZWL  RAB$W_RSZ(R2), R0       ; Get size of input line
                 6E    01    B0    0443   896         MOVW    #1, (SP)                ; Set success code in IOSB
              02 AE    50    B0    0446   897         MOVW    R0, 2(SP)               ; Set it into IOSB for common code
                    28 A2    DD    044A   898         PUSHL   RAB$L_RBF(R2)           ; Address of input record
                       50    DD    044D   899         PUSHL   R0                      ; Size of input record
                       54    DD    044F   900         PUSHL   R4                      ; Pointer to prompt descriptor
                00000045'EF  9F    0451   901         PUSHAB  L^T_ECHO                ; Pointer to ASCIC control string       [07]
                     7E    01    9A    0457   902         movzbl  #DS$K_printf,-(sp)      ; routine.                              [14]
                     7E    21    98    045A   903         cvtbl   #DS$K_type_script_echo,-(sp)    ; code                  [14]
        00000000'GF   06    FB    045D   904         CALLS   #6, G^DSX$PRINT         ; Output                                [14]
                    0109  31    0464   905         BRW     CHECK_STATUS            ; Skip ahead                            [11]
```

```
                                0467    907 ;+
                                0467    908 ; QIO SYS$INPUT.  Attempt QIO read with prompt--this will fail if SYS$INPUT
                                0467    909 ; is a mailbox.
                                0467    910 ;-
                                0467    911
                                0467    912 100$:   Br_If_CRD_AutoTest_Off  110$    ; If running CRD-AutoTest, branch       [16]
          0000FE00'EF    0B     E0      0467                BBS     #DSA$V_CRD_Autotest_Off, -      ; If bit set, branch    [36]
                         12             046E                  L^DSA$GL_Flags, 110$      ;                               [36]
                                046F    913         Br_If_CRD_MenuTest_On   110$    ; If running CRD-MenuTest, branch       [16]
          0000FE00'EF    12     E0      046F                BBS     #DSA$V_CRD_MenuTest_On, -       ; If bit set, branch    [36]
                         0A             0476                  L^DSA$GL_Flags, 110$      ;                               [36]
                                0477    914         Br_If_CRD_MenuTest_Off  110$    ; If running CRD Menu Test, branch      [16]
          0000FE00'EF    10     E0      0477                BBS     #DSA$V_CRD_MenuTest_Off, -      ; If bit set, branch    [36]
                         02             047E                  L^DSA$GL_Flags, 110$      ;                               [36]
                         45     11      047F    915         Brb     114$                    ; Else, skip all this CRD crap  [15]
                                0481    916
                    21   56    E9      0481    917 110$:   BlbC    R6, 112$                ; If user prompt, branch        [15]
                                0484    918
                                0484    919         ;+
                                0484    920         ; This is a DS prompt (i.e., "DS> "). Push the address and length of    [15]
                                0484    921         ; where to put the returned string as parameters 4 and 3.               [15]
                                0484    922         ;-
                                0484    923
                    04   BC    DF      0484    924         PushAL  a4 (AP)                 ; Push address of string as param-4     [15]
                    0C   AC    DD      0487    925         PushL   12 (AP)                 ; Push length of string as param-3      [15]
                         01    DD      048A    926         PushL   #DS$K_Type_DS_Prompt    ; Push typecode as param-2              [15]
                         01    DD      048C    927         PushL   #CRD$K_TypeCode         ; Push type of call as param-1          [15]
          00000000'FF    04    FB      048E    928         CallS   #4, @L^DS$GA_CRD_DS_Interface   ; And call the routine!       [15]
                                0495    929
                                0495    930         ;+
                                0495    931         ; If the routine returns:                                               [15]
                                0495    932         ;       Success => return the string that this routine returned.        [15]
                                0495    933         ;                  (High word of R0 = length of the returned string).   [15]
                                0495    934         ;       Failure => do the QIO and return that string.                   [15]
                                0495    935         ; Note the the length of the returned string is stored in the high word [15]
                                0495    936         ; of R0. Thus, the Extract-Zero-Extended-Field instruction below.       [15]
                                0495    937         ;-
                                0495    938
                        2E   50    E9      0495    939         BlbC    R0, 114$                ; Do the QIO if the routine fails       [15]
                                0498    940
          51    50    10     10    EF      0498    941         ExtZV   #16, #16, R0, R1        ; Return the length of the string ...   [15]
                      08   BC    51    D0      049D    942         MovL    R1, @8 (AP)             ; ... in both R1 and the 2nd parameter  [15]
                           50    00'   D0      04A1    943         MovL    S^#SS$_Normal, R0       ; Return success                        [15]
                                 04    04A4    944         Ret                             ; Return to the DS caller               [15]
                                04A5    945
                                04A5    946         ;+
                                04A5    947         ; This is a user prompt. Push the address and length of the user's      [15]
                                04A5    948         ; prompt as parameters 4 and 3.                                         [15]
                                04A5    949         ;-
                                04A5    950
                    0C   AE    DD      04A5    951 112$:   PushL   12 (SP)                 ; Push address of prompt as param-4     [15]
                    08   AE    DD      04A8    952         PushL   8  (SP)                 ; Push length of prompt as param-3      [15]
                         02    DD      04AB    953         PushL   #DS$K_Type_User_Prompt  ; Push typecode as param-2              [15]
                         01    DD      04AD    954         PushL   #CRD$K_TypeCode         ; Push type of call as param-1          [15]
          00000000'FF    04    FB      04AF    955         CallS   #4, @L^DS$GA_CRD_DS_Interface   ; And call the routine!       [15]
                                04B6    956
                                04B6    957         ;+
```

```
                                04B6          ; If the routine returns:                                          [15]
                                04B6          ;       Success => return the string that this routine returned.   [15]
                                04B6          ;                  (High word of R0 = length of the returned string).  [15]
                                04B6          ;       Failure => do the QIO and return that string.              [15]
                                04B6          ; Note the the length of the returned string is stored in the high word [15]
                                04B6          ; of R0. Thus, the Extract-Zero-Extended-Field instruction below.  [15]
                                04B6          ;-
                                04B6
                  0D 50    E9   04B6          BlbC    R0, 114$                ; Do the QIO if the routine fails  [15]
                                04B9
      51   50   10   10    EF   04B9          ExtZV   #16, #16, R0, R1        ; Return the length of the string ...  [15]
            08 BC   51    D0   04BE          MovL    R1, @8 (AP)             ; ... in both R1 and the 2nd parameter [15]
               50   00'   D0   04C2          MovL    S^#SS$_Normal, R0       ; Return success                   [15]
                          04   04C5          Ret                             ; Return to the DS caller          [15]
                                04C6
               50   6E    7E   04C6  114$:   MOVAQ   (SP), R0                ; Point to allocated IOSB          [15]
                                04C9          Br_If_Not_User  115$            ; Skip next if not user mode       [12]
      0000FE00'EF   1C    E1   04C9          BBC     #DSA$V_User, -                  ; If bit not set, branch
                    07         04D0                    L^DSA$GL_Flags, 115$            ;
      000000D4'EF   01    88   04D1          BISB    #1, DS$GB_QIOACT        ; Set input QIO active             [15]
                                04D8
                                04D8  115$:   $QIOW_S   , -                   ; No efn                           [07]
                                04D8                    L^DS$GW_TTIN, -       ; Channel number
                                04D8                    S^#IO$_READPROMPT, -  ; Function code
                                04D8                    (R0), -               ; IOSB
                                04D8                    ,, -                  ; No ASTadr or ASTprm
                                04D8                    P1=@4(AP), -          ; Buffer address
                                04D8                    P2=12(AP), -          ; Buffer size
                                04D8                    P5=12(R0), -          ; Prompt buffer address
                                04D8                    P6=8(R0)              ; Prompt buffer size
                  08 A0    DD   04D8                  PUSHL   8(R0)
                  0C A0    DD   04DB                  PUSHL   12(R0)
                     7E    7C   04DE                  CLRQ    -(SP)
                  0C AC    DD   04E0                  PUSHL   12(AP)
                  04 BC    DF   04E3                  PUSHAL  @4(AP)
                     7E    7C   04E6                  CLRQ    -(SP)
                     60    7F   04E8                  PUSHAQ  (R0)
               7E   00'   3C   04EA                  MOVZWL  S^#IO$_READPROMPT,-(SP)
      7E   00000000'EF    3C   04ED                  MOVZWL  L^DS$GW_TTIN,-(SP)
                     00    DD   04F4                  PUSHL   #0
      00000000'GF   0C    FB   04F6                  CALLS   #12,G^SYS$QIOW
                                04FD
      000000D4'EF   94         04FD  118$:   CLRB    DS$GB_QIOACT            ; Clear QIO active
      50   0000'8F   B1         0503          CMPW    #SS$_ILLIOFUNC, R0      ; Did readprompt fail?
                     03    13   0508          BEQL    120$                    ; Yes try w/ prompt
                   0063    31   050A          BRW     CHECK_STATUS            ; Branch to check the status       [11]
```

```
                          050D    992 ;+
                          050D    993 ; Mail box read: Fake "read with prompt" function via write followed by read.
                          050D    994 ;-
                          050D    995
                50   6E   7E    050D    996 120$:   MOVAQ   (SP), R0                ; Current stack level
                          0510    997
                          0510    998         $QIOW_S     , -                     ; No efn                        [07]
                          0510    999                 L^DS$GW_TTOUT, -            ; Channel number                [11]
                          0510   1000                 S^#IO$_WRITEVBLK, -         ; Function code                 [11]
                          0510   1001                 (R0), -                     ; IOSB                          [11]
                          0510   1002                 P1=a12(R0), -               ; Prompt buffer address         [11]
                          0510   1003                 P2=8(R0)                    ; Prompt buffer size            [11]
                   7E   7C  0510                   CLRQ    -(SP)
                   7E   7C  0512                   CLRQ    -(SP)
                08 A0   DD  0514                   PUSHL   8(R0)
                0C B0   DF  0517                   PUSHAL  a12(R0)
                   7E   7C  051A                   CLRQ    -(SP)
                   60   7F  051C                   PUSHAQ  (R0)
             7E    00'  3C  051E                   MOVZWL  S^#IO$_WRITEVBLK,-(SP)
7E   00000000'EF   3C  0521                   MOVZWL  L^DS$GW_TTOUT,-(SP)
                   00   DD  0528                   PUSHL   #0
00000000'GF        0C   FB  052A                   CALLS   #12,G^SYS$QIOW
          5F 50    E9  0531   1004         BLBC    R0, DS_GETLINE_X            ; Finish if error on prompt     [11]
          5C 6E    E9  0534   1005         BLBC    (SP), DS_GETLINE_X          ; Finish if error on prompt     [11]
                50   6E   7E  0537   1006         MOVAQ   (SP), R0                ; Current stack level
                          053A   1007         Br_If_Not_User  130$            ; Skip if not usermode          [12]
0000FE00'EF        1C   E1  053A                   BBC     #DSA$V_User, -              ; If bit not set, branch
                   07       0541                     L^DSA$GL_Flags, 130$        ;
000000D4'EF        01   88  0542   1008         BISB    #1, DS$GB_QIOACT        ; Flag input QIO active
                          0549   1009
                          0549   1010 130$:   $QIOW_S     , -                     ; No efn                        [07]
                          0549   1011                 L^DS$GW_TTIN, -             ; Channel number                [11]
                          0549   1012                 S^#IO$_READVBLK, -          ; Function code                 [11]
                          0549   1013                 (R0), -                     ; IOSB                          [11]
                          0549   1014                 P1=a4(AP), -                ; Buffer address                [11]
                          0549   1015                 P2=12(AP)                   ; Buffer size                   [11]
                   7E   7C  0549                   CLRQ    -(SP)
                   7E   7C  054B                   CLRQ    -(SP)
                0C AC   DD  054D                   PUSHL   12(AP)
                04 BC   DF  0550                   PUSHAL  a4(AP)
                   7E   7C  0553                   CLRQ    -(SP)
                   60   7F  0555                   PUSHAQ  (R0)
             7E    00'  3C  0557                   MOVZWL  S^#IO$_READVBLK,-(SP)
7E   00000000'EF   3C  055A                   MOVZWL  L^DS$GW_TTIN,-(SP)
                   00   DD  0561                   PUSHL   #0
00000000'GF        0C   FB  0563                   CALLS   #12,G^SYS$QIOW
     000000D4'EF   94       056A   1016         CLRB    DS$GB_QIOACT            ; Flag input QIO inactive
                          0570   1017
                          0570   1018 ;+
                          0570   1019 ; Fall into the common completion part.                                    [11]
                          0570   1020 ;-
```

```
                                0570  ;+
                                0570  ; Common completion section, joined from RMS section and both QIO sections
                                0570  ; via branch.  Check status and translate output.
                                0570  ;-
                                0570
                                0570  CHECK_STATUS:                                  ;                                        [11]
                  20 50    E9   0570          BLBC    R0, DS_GETLINE_X              ; RETURN IF QIO FAILED                   [11]
                  1D 6E    E9   0573          BLBC    (SP), DS_GETLINE_X            ; Return if QIO failed                   [11]
                                0576
            50    0000'8F  3C   0576          MOVZWL  #SS$_ENDOFFILE, R0            ; ASSUME END OF FILE
     04 AE    0001001A 8F  D1   057B          CMPL    #^X1001A, 4(SP)               ; SINGLE CHARACTER TERM=^Z?
                     0E    13   0583          BEQL    DS_GETLINE_X                  ; Exit with EOF                          [11]
                                0585
            54    02 AE    3C   0585          MOVZWL  2(SP), R4                     ; GET STRING LENGTH
            55    04 AC    D0   0589          MOVL    4(AP), R5                     ; Address of input string
                   0012    30   058D          BSBW    COPY_IT                       ; Copy and translate
                                0590
                                0590  ;+
                                0590  ; Here if no error
                                0590  ;-
                                0590
                                0590  DS_GETLINE_OKAY_X:                             ;                                        [11]
                50    00'  D0   0590          MOVL    S^#SS$_NORMAL, R0             ; Return success!
                                0593
                                0593  DS_GETLINE_X:                                  ;                                        [11]
            51    02 AE    3C   0593          MOVZWL  2(SP), R1                     ; Return the string length ...           [11]
            08 BC    51    D0   0597          MOVL    R1, @8(AP)                    ; ... in both places                     [11]
                                059B
                                059B  DS_GETLINE_X_CHECK:
                  03 59    E9   059B          BLBC    R9,80$                        ; Branch if we entered through GETline  [20]
                   0000    31   059E          BRW     EXIT                          ; Otherwise we entered through SUPERline[20]
                                05A1  80$:
                                05A1
                          04   05A1  EXIT :   RET                                   ; Return to caller
```

```
                05A2  1057          .SBTTL  COPY_IT -       Copy and translate
           000005A2  1058          .PSECT  Code, Shr, Exe, NoWrt, Byte
                05A2  1059 ;++
                05A2  1060 ; FUNCTIONAL DESCRIPTION:
                05A2  1061 ;
                05A2  1062 ;       This routine is used to copy a line to the user buffer
                05A2  1063 ;       It translates lower case alpha's to upper case if not ULTRIX.
                05A2  1064 ;
                05A2  1065 ; CALLING SEQUENCE:
                05A2  1066 ;
                05A2  1067 ;       BSBW    COPY_IT
                05A2  1068 ;
                05A2  1069 ; INPUT PARAMETERS:
                05A2  1070 ;
                05A2  1071 ;       NONE
                05A2  1072 ;
                05A2  1073 ; IMPLICIT INPUTS:
                05A2  1074 ;
                05A2  1075 ;       R4      Length of string to be copied
                05A2  1076 ;       R5      Address of string to be copied
                05A2  1077 ;       4(AP)   Address of user buffer
                05A2  1078 ;       8(AP)   Address to store length of line returned
                05A2  1079 ;       12(AP)  Length of user buffer
                05A2  1080 ;
                05A2  1081 ; OUTPUT PARAMETERS:
                05A2  1082 ;
                05A2  1083 ;       NONE
                05A2  1084 ;
                05A2  1085 ; IMPLICIT OUTPUTS:
                05A2  1086 ;
                05A2  1087 ;       4(AP)   buffer written
                05A2  1088 ;       @8(AP)  written with length of buffer
                05A2  1089 ;
                05A2  1090 ; SIDE EFFECTS:
                05A2  1091 ;
                05A2  1092 ;       NONE
                05A2  1093 ;
                05A2  1094 ; COMPLETION CODES:
                05A2  1095 ;
                05A2  1096 ;       NONE
                05A2  1097 ;--
```

```
                                  05A2  1099 COPY_IT:
                 0C AC    54   D1  05A2  1100          CMPL    R4, 12(AP)                ; Is buffer large enough?
                          04   15  05A6  1101          BLEQ    10$                       ; Branch if it is
                 54    0C AC   D0  05A8  1102          MOVL    12(AP), R4                ; Shorten record
                                  05AC  1103 10$:
                 08 BC    54   D0  05AC  1104          MOVL    R4, a8(AP)                ; Store length of record
                 50    04 AC   D0  05B0  1105          MOVL    4(AP), R0                 ; Get address of buffer
                          18   11  05B4  1106          BRB     30$                       ; Start with count of length
                    80    85   90  05B6  1107 20$:     MOVB    (R5)+, (R0)+              ; Copy a byte
              10 FF A0    06   E1  05B9  1108          BBC     #6, -1(R0), 30$           ; Branch if not alpha
 04 00000000'EF  00000000'8F   E0  05BE  1109          BBS     #ULTRIX_V_MODE, L^ULTRIX_FLAGS, 30$ ; Do not change if ULTRIX   [19]
                 FF A0    20   8A  05CA  1110          BICB2   #32, -1(R0)               ; force it to upper case
                       E5 54   F4  05CE  1111 30$:     SOBGEQ  R4, 20$                   ; Count and branch
                                05  05D1  1112          RSB                               ; Return to caller
```

```
                     05D2  1114          .SBTTL  ADVANCE -       Advance to next record in buffer
                000005D2   1115          .PSECT  Code, Shr, Exe, NoWrt, Byte
                     05D2  1116 ;++
                     05D2  1117 ; FUNCTIONAL DESCRIPTION:
                     05D2  1118 ;
                     05D2  1119 ;       Simulate the RMS $GET routine, deblocking records from disk blocks.
                     05D2  1120 ;       Note that .SC$W_CUR(R11) + SC$A_BUF(R11) is address of current record
                     05D2  1121 ;       Also .SC$W_LEN(R11) is length of that record.
                     05D2  1122 ;       If length is negative the same record is to be used again and length
                     05D2  1123 ;       is the complement of the record length.
                     05D2  1124 ;
                     05D2  1125 ; CALLING SEQUENCE:
                     05D2  1126 ;
                     05D2  1127 ;       BSBB    ADVANCE
                     05D2  1128 ;
                     05D2  1129 ; INPUT PARAMETERS:
                     05D2  1130 ;
                     05D2  1131 ;       NONE
                     05D2  1132 ;
                     05D2  1133 ; IMPLICIT INPUTS:
                     05D2  1134 ;
                     05D2  1135 ;       NONE
                     05D2  1136 ;
                     05D2  1137 ; OUTPUT PARAMETERS:
                     05D2  1138 ;
                     05D2  1139 ;       NONE
                     05D2  1140 ;
                     05D2  1141 ; IMPLICIT OUTPUTS:
                     05D2  1142 ;
                     05D2  1143 ;       SC$W_CURR       Offset to current record
                     05D2  1144 ;       SC$W_LEN        Length of current record
                     05D2  1145 ;       R4              Length of current record
                     05D2  1146 ;       R5              Address of current record
                     05D2  1147 ;
                     05D2  1148 ; COMPLETION CODES:
                     05D2  1149 ;
                     05D2  1150 ;       SS$_NORMAL
                     05D2  1151 ;       SS$_ENDOFFILE
                     05D2  1152 ;
                     05D2  1153 ; SIDE EFFECTS:
                     05D2  1154 ;
                     05D2  1155 ;       THE CURRENT SCRIPT DESCRIPTOR IS UPDATED.
                     05D2  1156 ;
                     05D2  1157 ; REGISTER USAGE:
                     05D2  1158 ;
                     05D2  1159 ;       R11     Assumed to contain the base for the current script block
                     05D2  1160 ;       R5      Current offset within buffer
                     05D2  1161 ;       R4      Length of current record
                     05D2  1162 ;--
```

```
                        05D2  1164 ADVANCE:
     55   0C AB   32    05D2  1165          CVTWL   SC$W_CURR(R11), R5       ; Get current offset
     54   0A AB   32    05D6  1166          CVTWL   SC$W_LEN(R11), R4        ; Get negated length
             12   19    05DA  1167          BLSS    60$                      ; Branch if record should be used again
                        05DC  1168
        55   54   C0    05DC  1169          ADDL    R4, R5                   ; Skip previous record
     55   0E AB   B1    05DF  1170          CMPW    SC$W_END(R11), R5        ; Past end of data?
             21   15    05E3  1171          BLEQ    80$                      ; Branch if end of file
  54   10 AB45   9A    05E5  1172          MOVZBL  SC$A_BUF(R11)[R5], R4    ; Get length of next record
             55   D6    05EA  1173          INCL    R5                       ; Point to data
             03   11    05EC  1174          BRB     70$                      ; Branch to finish
                        05EE  1175
        54   54   D2    05EE  1176 60$:     MCOML   R4, R4                   ; Fix length
                        05F1  1177
     0C AB   55   B0    05F1  1178 70$:     MOVW    R5, SC$W_CURR(R11)       ; Put current offset back
     0C AB   55   B0    05F5  1179          MOVW    R5, SC$W_CURR(R11)       ; Put current offset back
     0A AB   54   B0    05F9  1180          MOVW    R4, SC$W_LEN(R11)        ; Set current length
  55   10 AB45   9E    05FD  1181          MOVAB   SC$A_BUF(R11)[R5], R5    ; Absolute address of line
        50   00'  D0    0602  1182          MOVL    S^#SS$_NORMAL, R0        ; It worked!
                  05    0605  1183          RSB
                        0606  1184 80$:
           FBAC   30    0606  1185          BSBW    SCRIPT$FINISH            ; Remove current script file
  50   0000'8F   3C    0609  1186          MOVZWL  #SS$_ENDOFFILE, R0       ; Indicate the end
                  05    060E  1187          RSB                              ; Return
                        060F  1188          .END
```

```
$$.TAB                          = 0000007C R   D   03
$$.TABEND                       = 000000C0 R   D   03
$$.TMP                          = 00000000     D
$$.TMP1                         = 00000001     D
$$.TMP2                         = 00000062
$$T1                            = 00000001     D
$ER                             = 00000001     D
$MODULE                           00000000 R   D   02
ADVANCE                           000005D2 R   D   04
BEGIN                             ********     X   00
BIT...                          = 00000003     D
BLD_PRMT                          000000C4 R   D   03
CHECK_STATUS                      00000570 R   D   04
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_SIZE                        0000044C     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_FLAGS2                      00000444     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEMP_BASE                   00000448     D
CLI$L_TEST                        00000020     D
CLI$M_START_OR_RUN              = 00000008     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BASE_QUAL                 = 00000002     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NEXT                      = 00000001     D
CLI$V_NOALLOC                   = 00000000     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_START_OR_RUN              = 00000003     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
CLK$RE_INIT                       ********     X   00
COMMON                            00000234 R   D   04
CONSOLE                           00000329 R   D   04
COPY_IT                           000005A2 R   D   04
CRD$K_CRD_INITIALIZATION        = 00000000   G D
CRD$K_DS_CONTROL_C              = 00000001   G D
CRD$K_DS_FLAGS                  = 00000000   G D
CRD$K_FAILURE                   = 00000000     D
CRD$K_HOOK_POINT                = 00000000   G D
CRD$K_LAST_ACCESS_CODE          = 00000002   G D
CRD$K_LAST_DS_VARIABLE          = 00000002   G D
CRD$K_LAST_FUNCTION             = 00000002   G D
CRD$K_LAST_HOOK_POINT           = 00000001   G D
CRD$K_READ                      = 00000000   G D
CRD$K_SUCCESS                   = 00000001     D
CRD$K_TYPECODE                  = 00000001   G D
CRD$K_WRITE                     = 00000001   G D
DEBUG_THE_SUCKER                  ********     X   00
DS$ABORT                          ********     X   00
DS$CLI                            ********     X   00
DS$CLRVEC                         ********     X   00
DS$GA_CRD_DS_INTERFACE            ********     X   00
DS$GB_QIOACT                      000000D4 RG  D   03
DS$GL_CLIBASE                     ********     X   00
DS$GL_FLAGS                       ********     X   00
DS$GT_BUFFER                      ********     X   00
DS$GT_PROMPT                      ********     X   00
DS$GW_TTIN                        ********     X   00
DS$GW_TTOUT                       ********     X   00
DS$K_BUFSIZ                       ********     X   00
DS$K_ERROR                      = 00000002     D
DS$K_NORMAL                     = 00000001     D
DS$K_PRINTB                     = 00000002     D
DS$K_PRINTF                     = 00000001     D
DS$K_PRINTI                     = 00000000     D
DS$K_PRINTX                     = 00000003     D
DS$K_SEVERE                     = 00000004     D
DS$K_SUBSYS                     = 00000066     D
DS$K_TYPE_ABORT_PROGRAM         = 00000014     D
DS$K_TYPE_ABORT_TEST            = 00000013     D
DS$K_TYPE_COMMAND_ERR           = 00000015     D
DS$K_TYPE_COMMAND_OUT           = 00000016     D
DS$K_TYPE_CRD_AUTOTEST          = 0000001A     D
DS$K_TYPE_DS_PROMPT             = 00000001     D
DS$K_TYPE_DS_START              = 0000001D     D
```

```
DS$K_TYPE_ERRDEV                = 00000008    D
DS$K_TYPE_ERRHARD               = 00000006    D
DS$K_TYPE_ERROR_BODY            = 00000009    D
DS$K_TYPE_ERROR_END             = 0000000A    D
DS$K_TYPE_ERRPREP               = 0000001B    D
DS$K_TYPE_ERRSOFT               = 00000007    D
DS$K_TYPE_ERRSUP                = 00000004    D
DS$K_TYPE_ERRSYS                = 00000005    D
DS$K_TYPE_ERR_HALT              = 0000000D    D
DS$K_TYPE_EXCEPTION             = 0000000C    D
DS$K_TYPE_EXCEPTION_HEAD        = 0000000B    D
DS$K_TYPE_FIRST_PASS            = 00000011    D
DS$K_TYPE_GENERAL               = 00000000    D
DS$K_TYPE_GENERAL_ERROR         = 00000003    D
DS$K_TYPE_NO_TESTS              = 00000012    D
DS$K_TYPE_PARAM_ERROR           = 0000001C    D
DS$K_TYPE_PROGRAM_END           = 00000010    D
DS$K_TYPE_PROGRAM_INFO          = 00000017    D
DS$K_TYPE_PROGRAM_START         = 0000000F    D
DS$K_TYPE_QIO_INVADP            = 00000024    D
DS$K_TYPE_QIO_NODRIVER          = 00000022    D
DS$K_TYPE_QIO_WRONGVER          = 00000023    D
DS$K_TYPE_SCRIPT_ECHO           = 00000021    D
DS$K_TYPE_SCRIPT_PNF            = 0000001E    D
DS$K_TYPE_SCRIPT_PROMPT         = 00000020    D
DS$K_TYPE_SCRIPT_SKIP           = 0000001F    D
DS$K_TYPE_SEQUENCE_ERROR        = 00000019    D
DS$K_TYPE_START_ERR             = 00000018    D
DS$K_TYPE_START_LIST            = 00000025    D
DS$K_TYPE_SUMMARY               = 0000000E    D
DS$K_TYPE_USER_PROMPT           = 00000002    D
DS$K_WARNING                    = 00000000    D
DS$LOAD                           ********    X   00
DS$M_ABRTFLG                    = 00000040    D
DS$M_BADTIME                    = 00100000    D
DS$M_BATCH                      = 00400000    D
DS$M_BRKCLR                     = 00001000    D
DS$M_BRKPT                      = 00000800    D
DS$M_CHARFLG                    = 00000100    D
DS$M_CMDFLG                     = 00000080    D
DS$M_CTRLC                      = 00000001    D
DS$M_CTRLO                      = 00010000    D
DS$M_DEVFLG                     = 00000200    D
DS$M_DISABLCC                   = 01000000    D
DS$M_DONFLG                     = 00002000    D
DS$M_ERRFLG                     = 00000010    D
DS$M_EXCEPT                     = 00080000    D
DS$M_EXETST                     = 00040000    D
DS$M_HLTFLG                     = 00000008    D
DS$M_LODFLG                     = 00000002    D
DS$M_MEMMGT                     = 00008000    D
DS$M_NI                         = 04000000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$RAB_INPUT                      ********    X   00
DS$RAB_OUTPUT                     ********    X   00
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DS$_AP_NORMAL_BREAK             = 00660150    D
DS$_ARITH                       = 006600D0    D
DS$_ASBE                        = 00660118    D
DS$_BADLINK                     = 006600F0    D
DS$_BADTYPE                     = 006600E8    D
DS$_BIIC                        = 00660120    D
DS$_CHME                        = 006600A8    D
DS$_CHMK                        = 006600E0    D
DS$_DEVNAME                     = 00660108    D
DS$_ERROR                       = 00660002    D
DS$_FHWE                        = 00660068    D
DS$_FRAGBUF                     = 00660080    D
DS$_ICBUSY                      = 006600C8    D
DS$_ICERR                       = 006600C0    D
DS$_IHWE                        = 00660060    D
DS$_ILLCHAR                     = 00660018    D
DS$_ILLPAGCNT                   = 00660078    D
DS$_ILLUNIT                     = 00660100    D
DS$_INIT_FAIL                   = 00660140    D
DS$_INSFMEM                     = 00660050    D
DS$_INVCPU                      = 00660130    D
```

```
DS$_IPL2HI                      = 006600B8    D
DS$_IVADDR                      = 00660040    D
DS$_IVVECT                      = 00660038    D
DS$_KRNLSTK                     = 00660090    D
DS$_LOGIC                       = 00660070    D
DS$_MCHK                        = 00660088    D
DS$_MEM_ALLOC_ERR               = 00660138    D
DS$_MMOFF                       = 00660058    D
DS$_NEEDUNIT                    = 006600F8    D
DS$_NODE                        = 00660128    D
DS$_NOPCS                       = 00660110    D
DS$_NORMAL                      = 00660001    D
DS$_NOSUPPORT                   = 006600B1    D
DS$_NOTALLOWMP                  = 00660148    D
DS$_NOTDON                      = 00660030    D
DS$_NOTIMP                      = 006600B0    D
DS$_NULLSTR                     = 00660010    D
DS$_OVERFLOW                    = 00660008    D
DS$_POWER                       = 00660098    D
DS$_PROGERR                     = 00660020    D
DS$_SEVERE                      = 00660004    D
DS$_TRANSL                      = 006600A0    D
DS$_TRUNCATE                    = 00660028    D
DS$_UNEXPINT                    = 006600D8    D
DS$_UNSUPPDEV                   = 00660158    D
DS$_VASFULL                     = 00660048    D
DS$_WARNING                     = 00660000    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSA$V_BINARY                  = 00000001      D
DSA$V_CRD_AUTOTEST_OFF        = 0000000B      D
DSA$V_CRD_MENUTEST_OFF        = 00000010      D
DSA$V_CRD_MENUTEST_ON         = 00000012      D
DSA$V_OPER                    = 0000000C      D
DSA$V_QA                      = 0000000F      D
DSA$V_USER                    = 0000001C      D
DSA$V_VERIFY                  = 00000009      D
DSR$COMPLETION                  ********    X   00
DSX$PRINT                       ********    X   00
DSX$TYPE_OUT                    ********    X   00
DS_ERRSUP                       ********    X   00
DS_GETLINE                      000001E1 RG D   04
DS_GETLINE_OKAY_X                 00000590 R  D   04
DS_GETLINE_X                      00000593 R  D   04
DS_GETLINE_X_CHECK                0000059B R  D   04
DS_SUPERLINE                      000001DA RG D   04
EXE$ALONONPAGED                   ********    X   00
EXE$DEANONPAGED                   ********    X   00
EXIT                              000005A1 R  D   04
FAB$B_DNS                     = 00000035      D
FAB$B_FNS                     = 00000034      D
FAB$C_BID                     = 00000003      D
FAB$C_BLN                     = 00000050      D
FAB$C_SEQ                     = 00000000      D
FAB$C_VAR                     = 00000002      D
FAB$L_ALQ                     = 00000010      D
FAB$L_DNA                     = 00000030      D
FAB$L_FNA                     = 0000002C      D
FAB$L_FOP                     = 00000004      D
FAB$V_CHAN_MODE               = 00000002      D
FAB$V_FILE_MODE               = 00000004      D
FAB$V_LNM_MODE                = 00000000      D
FAB$W_GBC                     = 00000048      D
FALSE                         = 00000000      D
FMT_PRMT                          00000007 R  D   02
INIT_CONTEXT                      ********    X   00
IO$_READPROMPT                    ********    X   00
IO$_READVBLK                      ********    X   00
IO$_WRITEVBLK                     ********    X   00
L_FLAGS                           000000D0 R  D   03
L_LINK                            000000CC R  D   03
OFF                           = 00000000      D
ON                            = 00000001      D
PR$_SID_TYP8NN                = 00000006      D
PR$_SID_TYP8RR                = 00000011      D
PRINT_LINE                        00000308 R  D   04
PRMT_ADR                          000000C0 R  D   03
PRMT_LEN                          000000C8 R  D   03
PRNT_PRMT                         ********    X   00
QIO$CLEANUP                       ********    X   00
RAB$B_RAC                     = 0000001E      D
RAB$C_BID                     = 00000001      D
RAB$C_BLN                     = 00000044      D
RAB$C_SEQ                     = 00000000      D
RAB$L_CTX                     = 00000018      D
RAB$L_RBF                     = 00000028      D
RAB$L_ROP                     = 00000004      D
RAB$L_UBF                     = 00000024      D
RAB$W_RSZ                     = 00000022      D
RAB$W_USZ                     = 00000020      D
RMS$_EOF                      = 0001827A      D
SC$A_BUF                          00000010    D
SC$C_BLN                      = 00000010      D
SC$L_LINK                         00000000    D
SC$W_CURR                         0000000C    D
SC$W_END                          0000000E    D
SC$W_LEN                          0000000A    D
SC$W_SIZE                         00000008    D
SCAN$COMPARE                      ********    X   00
```

```
SCAN$SPACES                     ********   X   00
SCB$L_ACCESS                    00000020     D
SCB$L_ARITH                     00000034     D
SCB$L_BREAK                     0000002C     D
SCB$L_CHME                      00000044     D
SCB$L_CHMK                      00000040     D
SCB$L_CHMS                      00000048     D
SCB$L_CHMU                      0000004C     D
SCB$L_COMPAT                    00000030     D
SCB$L_KNLSTK                    00000008     D
SCB$L_MACHCHK                   00000004     D
SCB$L_OPCCUS                    00000014     D
SCB$L_OPCDEC                    00000010     D
SCB$L_POWER                     0000000C     D
SCB$L_RADRMOD                   0000001C     D
SCB$L_ROPRAND                   00000018     D
SCB$L_RXDB                      000000F8     D
SCB$L_SFTLVL1                   00000084     D
SCB$L_SFTLVL10                  000000A8     D
SCB$L_SFTLVL11                  000000AC     D
SCB$L_SFTLVL12                  000000B0     D
SCB$L_SFTLVL13                  000000B4     D
SCB$L_SFTLVL14                  000000B8     D
SCB$L_SFTLVL15                  000000BC     D
SCB$L_SFTLVL2                   00000088     D
SCB$L_SFTLVL3                   0000008C     D
SCB$L_SFTLVL4                   00000090     D
SCB$L_SFTLVL5                   00000094     D
SCB$L_SFTLVL6                   00000098     D
SCB$L_SFTLVL7                   0000009C     D
SCB$L_SFTLVL8                   000000A0     D
SCB$L_SFTLVL9                   000000A4     D
SCB$L_TBIT                      00000028     D
SCB$L_TIMER                     000000C0     D
SCB$L_TRANSL                    00000024     D
SCB$L_TXDB                      000000FC     D
SCB$L_VM                        00000038     D
SCB$L_ZERO                      00000000     D
SCRIPT$CONT                     000001A2  RG D   04
SCRIPT$FINISH                   000001B5  R  D   04
SCRIPT$FLUSH                    00000157  RG D   04
SCRIPT$INIT                     00000000  RG D   04
SCRIPT$MINCORE                =  00000400   G D
SCRIPT$OPEN                     0000000A  RG D   04
SCRIPT$OPEN_X                   00000151  R  D   04
SCRIPT$STOP                     00000199  RG D   04
SCRIPT_FAB                      0000002C  R  D   03
SCRIPT_RAB                      0000007C  R  D   03
SCRIPT_XAB                      00000000  R  D   03
SIZ...                        =  00000001     D
SS$_CONTROLC                    ********    X   00
SS$_ENDOFFILE                   ********    X   00
SS$_ILLIOFUNC                   ********    X   00
SS$_NORMAL                      ********    X   00
SYS$CLOSE                       ********  G     04
SYS$CONNECT                     ********  G     04
SYS$DISCONNECT                  ********  G     04
SYS$FAO                         ********    X   00
SYS$GET                         ********  G     04
SYS$HIBER                       ********    X   00
SYS$OPEN                        ********  G     04
SYS$QIO                         ********    X   00
SYS$QIOW                        ********  GX    00
SYS$READEF                      ********    X   00
SYS$WAKE                        ********    X   00
TRUE                          =  00000001     D
T_DEF_COM                       0000000E  R  D   02
T_ECHO                          00000045  R  D   02
T_EOF                           00000013  R  D   02
T_NULL                          00000012  R  D   02
T_PNF                           0000004E  R  D   02
T_PROMPT                        0000001B  R  D   02
T_PROMPT2                       00000024  R  D   02
T_PROMPT3                       0000002B  R  D   02
T_SKIP                          00000031  R  D   02
ULTRIX_FLAGS                    ********    X   00
ULTRIX_V_MODE                   ********    X   00
V_ECHOED                      =  00000001     D
XAB$C_FHC                     =  0000001D     D
XAB$C_FHCLEN                  =  0000002C     D
XAB$L_EBK                     =  00000010     D
XAB$L_NXT                     =  00000004     D
XAB$W_FFB                     =  00000014     D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000FE70 (65136.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000071 (113.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
WORK	000000D5 (213.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
CODE	0000060F (1551.)	04 (4.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$.TAB	=0000007C-R	246 (1)	244 (1) 245 (1) 246 (1)
$$.TABEND	=000000C0-R	246 (1)	244 (1) 245 (1) 246 (1)
$$.TMP	=00000000	246 (1)	245 (1) 246 (1)
$$.TMP1	=00000001	887 (1)	393 (1) 424 (1) 433 (1) 447 (1) 449 (1) 887 (1)
$$.TMP2	=00000062	887 (1)	393 (1) 424 (1) 433 (1) 447 (1) 449 (1) 887 (1)
$$T1	=00000001	1015 (1)	1003 (1) 1015 (1) 985 (1)
$ER	=00000001	210 (1)	
$MODULE	00000000-R	210 (1)	
ADVANCE	000005D2-R	1164 (1)	#-721 (1)
BEGIN	00000000-XR		176 (1) 477 (1)
BIT...	=00000003	203 (1)	194 (1) 195 (1) 202 (1) 203 (1)
BLD_PRMT	000000C4-R	250 (1)	685 (1) #-688 (1) #-692 (1) 693 (1)
CHECK_STATUS	00000570-R	1027 (1)	#-905 (1) #-990 (1)
CLI$K_BUFSIZ	=00000100	196 (1)	196 (1)
CLI$K_SIZE	0000044C	196 (1)	
CLI$L_ADDRESS	00000018	196 (1)	
CLI$L_COMMAND	00000004	196 (1)	
CLI$L_DATA	0000001C	196 (1)	
CLI$L_FLAGS	00000000	196 (1)	
CLI$L_FLAGS2	00000444	196 (1)	
CLI$L_LAST	00000024	196 (1)	
CLI$L_NEXT	00000030	196 (1)	
CLI$L_PASS	0000002C	196 (1)	
CLI$L_SUBT	00000028	196 (1)	
CLI$L_TEMP_BASE	00000448	196 (1)	
CLI$L_TEST	00000020	196 (1)	
CLI$M_START_OR_RUN	=00000008	196 (1)	
CLI$Q_BUFQWD	00000034	196 (1)	
CLI$Q_FILE	00000008	196 (1)	#-377 (1) #-379 (1)
CLI$Q_SECTION	00000010	196 (1)	
CLI$Q_TIME	0000043C	196 (1)	
CLI$T_BUFFER	0000003C	196 (1)	
CLI$V_ADAPTER	=00000018	196 (1)	
CLI$V_ADR	=0000000B	196 (1)	
CLI$V_ASCII	=00000013	196 (1)	
CLI$V_BASE_QUAL	=00000002	196 (1)	
CLI$V_BREAK	=0000000A	196 (1)	
CLI$V_BRIEF	=0000001B	196 (1)	
CLI$V_BYTE	=0000000D	196 (1)	
CLI$V_CLEAR	=00000002	196 (1)	
CLI$V_DEC	=00000010	196 (1)	
CLI$V_DEFAULT	=0000000C	196 (1)	
CLI$V_DEPOSIT	=00000019	196 (1)	
CLI$V_EVENT	=00000008	196 (1)	
CLI$V_EXAM	=00000005	196 (1)	
CLI$V_FLAGS	=00000009	196 (1)	
CLI$V_HEX	=00000012	196 (1)	
CLI$V_KERNEL	=00000017	196 (1)	

```
CLI$V_LOAD                      =00000006     196     (1)
CLI$V_LONG                      =0000000F     196     (1)
CLI$V_NEXT                      =00000001     196     (1)
CLI$V_NOALLOC                   =00000000     196     (1)
CLI$V_NOTNUF                    =00000001     196     (1)
CLI$V_OCT                       =00000011     196     (1)
CLI$V_PREG                      =0000001A     196     (1)
CLI$V_QA                        =00000007     196     (1)
CLI$V_QACKLOOPLOOPS             =0000001C     196     (1)
CLI$V_QAERRORPRINTS             =0000001B     196     (1)
CLI$V_QAMULTIPLEPASS            =0000001F     196     (1)
CLI$V_QASUBTESTLOOPS            =0000001E     196     (1)
CLI$V_QATESTLOOPS               =0000001D     196     (1)
CLI$V_REG                       =00000014     196     (1)
CLI$V_REQUIRED                  =00000000     196     (1)
CLI$V_RUN                       =00000015     196     (1)
CLI$V_SET                       =00000003     196     (1)
CLI$V_SHOW                      =00000004     196     (1)
CLI$V_START_OR_RUN              =00000003     196     (1)       196     (1)
CLI$V_VALSEC                    =00000016     196     (1)
CLI$V_WORD                      =0000000E     196     (1)
CLK$RE_INIT                      00000000-XR                    166     (1)
COMMON                           00000234-R   699     (1)     #-674     (1)     #-726     (1)     #-733     (1)     #-756     (1)
CONSOLE                          00000329-R   796     (1)     #-719     (1)     #-766     (1)
COPY_IT                          000005A2-R   1099    (1)     #-1037    (1)     #-769     (1)
CRD$K_CRD_INITIALIZATION        =00000000     203     (1)
CRD$K_DS_CONTROL_C              =00000001     203     (1)
CRD$K_DS_FLAGS                  =00000000     203     (1)
CRD$K_FAILURE                   =00000000     203     (1)
CRD$K_HOOK_POINT                =00000000     203     (1)
CRD$K_LAST_ACCESS_CODE          =00000002     203     (1)
CRD$K_LAST_DS_VARIABLE          =00000002     203     (1)
CRD$K_LAST_FUNCTION             =00000002     203     (1)
CRD$K_LAST_HOOK_POINT           =00000001     203     (1)
CRD$K_READ                      =00000000     203     (1)
CRD$K_SUCCESS                   =00000001     203     (1)
CRD$K_TYPECODE                  =00000001     203     (1)     #-927     (1)     #-954     (1)
CRD$K_WRITE                     =00000001     203     (1)
DEBUG_THE_SUCKER                 00000000-XR                    179     (1)     #-831     (1)
DS$ABORT                         00000000-XR                    166     (1)       760     (1)
DS$CLI                           00000000-XR                    165     (1)
DS$CLRVEC                        00000000-XR                    166     (1)       527     (1)
DS$GA_CRD_DS_INTERFACE           00000000-XR                    180     (1)       928     (1)       955     (1)
DS$GB_QIOACT                     000000D4-R   262     (1)     #-1008    (1)     #-1016    (1)     #-975     (1)     #-987     (1)
DS$GL_CLIBASE                    00000000-XR                    169     (1)       376     (1)
DS$GL_FLAGS                      00000000-XR                    173     (1)       415     (1)       416     (1)       474     (1)
                                                                532     (1)       541     (1)       549     (1)       710     (1)
                                                                875     (1)
DS$GT_BUFFER                     00000000-XR                    175     (1)       800     (1)
DS$GT_PROMPT                     00000000-XR                    164     (1)     #-810     (1)
DS$GW_TTIN                       00000000-XR                  #-1015    (1)       168     (1)     #-985     (1)
DS$GW_TTOUT                      00000000-XR                  #-1003    (1)       168     (1)
DS$K_BUFSIZ                      00000000-XR                    175     (1)     #-801     (1)
DS$K_ERROR                      =00000002     194     (1)
DS$K_NORMAL                     =00000001     194     (1)
DS$K_PRINTB                     =00000002     202     (1)
DS$K_PRINTF                     =00000001     202     (1)     #-744     (1)     #-753     (1)     #-786     (1)     #-864     (1)
```

```
                                                                  #-902    (1)
DS$K_PRINTI                    =00000000     202    (1)
DS$K_PRINTX                    =00000003     202    (1)
DS$K_SEVERE                    =00000004     194    (1)
DS$K_SUBSYS                    =00000066     194    (1)      194    (1)
DS$K_TYPE_ABORT_PROGRAM        =00000014     202    (1)
DS$K_TYPE_ABORT_TEST           =00000013     202    (1)
DS$K_TYPE_COMMAND_ERR          =00000015     202    (1)
DS$K_TYPE_COMMAND_OUT          =00000016     202    (1)
DS$K_TYPE_CRD_AUTOTEST         =0000001A     202    (1)
DS$K_TYPE_DS_PROMPT            =00000001     202    (1)    #-812    (1)    #-926    (1)
DS$K_TYPE_DS_START             =0000001D     202    (1)
DS$K_TYPE_ERRDEV               =00000008     202    (1)
DS$K_TYPE_ERRHARD              =00000006     202    (1)
DS$K_TYPE_ERROR_BODY           =00000009     202    (1)
DS$K_TYPE_ERROR_END            =0000000A     202    (1)
DS$K_TYPE_ERRPREP              =0000001B     202    (1)
DS$K_TYPE_ERRSOFT              =00000007     202    (1)
DS$K_TYPE_ERRSUP               =00000004     202    (1)
DS$K_TYPE_ERRSYS               =00000005     202    (1)
DS$K_TYPE_ERR_HALT             =0000000D     202    (1)
DS$K_TYPE_EXCEPTION            =0000000C     202    (1)
DS$K_TYPE_EXCEPTION_HEAD       =0000000B     202    (1)
DS$K_TYPE_FIRST_PASS           =00000011     202    (1)
DS$K_TYPE_GENERAL              =00000000     202    (1)
DS$K_TYPE_GENERAL_ERROR        =00000003     202    (1)
DS$K_TYPE_NO_TESTS             =00000012     202    (1)
DS$K_TYPE_PARAM_ERROR          =0000001C     202    (1)
DS$K_TYPE_PROGRAM_END          =00000010     202    (1)
DS$K_TYPE_PROGRAM_INFO         =00000017     202    (1)
DS$K_TYPE_PROGRAM_START        =0000000F     202    (1)
DS$K_TYPE_QIO_INVADP           =00000024     202    (1)
DS$K_TYPE_QIO_NODRIVER         =00000022     202    (1)
DS$K_TYPE_QIO_WRONGVER         =00000023     202    (1)
DS$K_TYPE_SCRIPT_ECHO          =00000021     202    (1)    #-903    (1)
DS$K_TYPE_SCRIPT_PNF           =0000001E     202    (1)    #-745    (1)
DS$K_TYPE_SCRIPT_PROMPT        =00000020     202    (1)    #-787    (1)    #-865    (1)
DS$K_TYPE_SCRIPT_SKIP          =0000001F     202    (1)    #-754    (1)
DS$K_TYPE_SEQUENCE_ERROR       =00000019     202    (1)
DS$K_TYPE_START_ERR            =00000018     202    (1)
DS$K_TYPE_START_LIST           =00000025     202    (1)
DS$K_TYPE_SUMMARY              =0000000E     202    (1)
DS$K_TYPE_USER_PROMPT          =00000002     202    (1)    #-805    (1)    #-953    (1)
DS$K_WARNING                   =00000000     194    (1)
DS$LOAD                         00000000-XR                 165    (1)
DS$M_ABRTFLG                   =00000040     195    (1)
DS$M_BADTIME                   =00100000     195    (1)
DS$M_BATCH                     =00400000     195    (1)
DS$M_BRKCLR                    =00001000     195    (1)
DS$M_BRKPT                     =00000800     195    (1)
DS$M_CHARFLG                   =00000100     195    (1)
DS$M_CMDFLG                    =00000080     195    (1)
DS$M_CTRLC                     =00000001     195    (1)
DS$M_CTRLO                     =00010000     195    (1)
DS$M_DEVFLG                    =00000200     195    (1)
DS$M_DISABLCC                  =01000000     195    (1)
DS$M_DONFLG                    =00002000     195    (1)
```

```
DS$M_ERRFLG                     =00000010      195     (1)
DS$M_EXCEPT                     =00080000      195     (1)
DS$M_EXETST                     =00040000      195     (1)
DS$M_HLTFLG                     =00000008      195     (1)
DS$M_LODFLG                     =00000002      195     (1)
DS$M_MEMMGT                     =00008000      195     (1)
DS$M_NI                         =04000000      195     (1)
DS$M_OUTPUT                     =00800000      195     (1)
DS$M_RUBFLG                     =00000020      195     (1)
DS$M_SCRIPT                     =00200000      195     (1)
DS$M_SETIMR                     =02000000      195     (1)
DS$M_STRFLG                     =00000004      195     (1)
DS$M_SUBT                       =00004000      195     (1)
DS$M_SYSFLG                     =00000400      195     (1)
DS$M_TIMED                      =02000000      195     (1)
DS$M_TIMED_OUT                  =08000000      195     (1)
DS$M_TIMRON                     =00020000      195     (1)
DS$RAB_INPUT                     00000000-XR                   177     (1)       883     (1)
DS$RAB_OUTPUT                    00000000-XR                   177     (1)       882     (1)
DS$V_ABRTFLG                    =00000006      195     (1)
DS$V_BADTIME                    =00000014      195     (1)
DS$V_BATCH                      =00000016      195     (1)     #-875     (1)
DS$V_BRKCLR                     =0000000C      195     (1)
DS$V_BRKPT                      =0000000B      195     (1)
DS$V_CHARFLG                    =00000008      195     (1)
DS$V_CMDFLG                     =00000007      195     (1)
DS$V_CTRLC                      =00000000      195     (1)     #-416     (1)
DS$V_CTRLO                      =00000010      195     (1)
DS$V_DEVFLG                     =00000009      195     (1)
DS$V_DISABLCC                   =00000018      195     (1)
DS$V_DONFLG                     =0000000D      195     (1)
DS$V_ERRFLG                     =00000004      195     (1)
DS$V_EXCEPT                     =00000013      195     (1)
DS$V_EXETST                     =00000012      195     (1)
DS$V_HLTFLG                     =00000003      195     (1)
DS$V_LODFLG                     =00000001      195     (1)
DS$V_MEMMGT                     =0000000F      195     (1)
DS$V_NI                         =0000001A      195     (1)
DS$V_OUTPUT                     =00000017      195     (1)
DS$V_RUBFLG                     =00000005      195     (1)
DS$V_SCRIPT                     =00000015      195     (1)     #-474     (1)     #-532     (1)     #-541     (1)     #-549     (1)
                                                               #-710     (1)
DS$V_SETIMR                     =00000019      195     (1)
DS$V_STRFLG                     =00000002      195     (1)     #-415     (1)
DS$V_SUBT                       =0000000E      195     (1)
DS$V_SYSFLG                     =0000000A      195     (1)
DS$V_TIMED                      =00000019      195     (1)
DS$V_TIMED_OUT                  =0000001B      195     (1)
DS$V_TIMRON                     =00000011      195     (1)
DS$_AP_NORMAL_BREAK             =00660150      194     (1)
DS$_ARITH                       =006600D0      194     (1)
DS$_ASBE                        =00660118      194     (1)
DS$_BADLINK                     =006600F0      194     (1)
DS$_BADTYPE                     =006600E8      194     (1)
DS$_BIIC                        =00660120      194     (1)
DS$_CHME                        =006600A8      194     (1)
DS$_CHMK                        =006600E0      194     (1)
```

```
DS$_DEVNAME                     =00660108     194    (1)
DS$_ERROR                       =00660002     194    (1)
DS$_FHWE                        =00660068     194    (1)
DS$_FRAGBUF                     =00660080     194    (1)
DS$_ICBUSY                      =006600C8     194    (1)
DS$_ICERR                       =006600C0     194    (1)
DS$_IHWE                        =00660060     194    (1)
DS$_ILLCHAR                     =00660018     194    (1)
DS$_ILLPAGCNT                   =00660078     194    (1)
DS$_ILLUNIT                     =00660100     194    (1)
DS$_INIT_FAIL                   =00660140     194    (1)
DS$_INSFMEM                     =00660050     194    (1)
DS$_INVCPU                      =00660130     194    (1)
DS$_IPL2HI                      =006600B8     194    (1)
DS$_IVADDR                      =00660040     194    (1)
DS$_IVVECT                      =00660038     194    (1)
DS$_KRNLSTK                     =00660090     194    (1)
DS$_LOGIC                       =00660070     194    (1)
DS$_MCHK                        =00660088     194    (1)
DS$_MEM_ALLOC_ERR               =00660138     194    (1)
DS$_MMOFF                       =00660058     194    (1)
DS$_NEEDUNIT                    =006600F8     194    (1)
DS$_NODE                        =00660128     194    (1)
DS$_NOPCS                       =00660110     194    (1)
DS$_NORMAL                      =00660001     194    (1)
DS$_NOSUPPORT                   =006600B1     194    (1)
DS$_NOTALLOWMP                  =00660148     194    (1)
DS$_NOTDON                      =00660030     194    (1)
DS$_NOTIMP                      =006600B0     194    (1)      194    (1)
DS$_NULLSTR                     =00660010     194    (1)
DS$_OVERFLOW                    =00660008     194    (1)
DS$_POWER                       =00660098     194    (1)
DS$_PROGERR                     =00660020     194    (1)
DS$_SEVERE                      =00660004     194    (1)
DS$_TRANSL                      =006600A0     194    (1)
DS$_TRUNCATE                    =00660028     194    (1)
DS$_UNEXPINT                    =006600D8     194    (1)
DS$_UNSUPPDEV                   =00660158     194    (1)
DS$_VASFULL                     =00660048     194    (1)
DS$_WARNING                     =00660000     194    (1)      194    (1)
DSA$GL_FLAGS                     0000FE00                     1007   (1)      364    (1)      524    (1)      530    (1)
                                                              782    (1)      803    (1)      861    (1)      912    (1)
                                                              913    (1)      914    (1)      974    (1)
DSA$GL_SID                       0000FE14                   #-520    (1)    #-522    (1)
DSA$V_BINARY                    =00000001                   #-803    (1)
DSA$V_CRD_AUTOTEST_OFF          =0000000B                   #-912    (1)
DSA$V_CRD_MENUTEST_OFF          =00000010                   #-914    (1)
DSA$V_CRD_MENUTEST_ON           =00000012                   #-913    (1)
DSA$V_OPER                      =0000000C                   #-861    (1)
DSA$V_QA                        =0000000F                   #-530    (1)
DSA$V_USER                      =0000001C                  #-1007    (1)    #-364    (1)    #-524    (1)    #-974    (1)
DSA$V_VERIFY                    =00000009                   #-782    (1)
DSR$COMPLETION                   00000000-XR                 166    (1)      394    (1)      453    (1)
DSX$PRINT                        00000000-XR                 165    (1)      746    (1)      755    (1)      788    (1)
                                                             866    (1)      904    (1)
DSX$TYPE_OUT                     00000000-XR                 178    (1)      849    (1)
DS_ERRSUP                        00000000-XR                 165    (1)
```

```
DS_GETLINE                      000001E1-R   676    (1)
DS_GETLINE_OKAY_X               00000590-R  1043    (1)    #-867    (1)
DS_GETLINE_X                    00000593-R  1046    (1)    #-1004   (1)    #-1005   (1)    #-1028   (1)    #-1029   (1)
                                                           #-1033   (1)    #-893    (1)
DS_GETLINE_X_CHECK              0000059B-R  1050    (1)      774    (1)
DS_SUPERLINE                    000001DA-R   671    (1)
EXE$ALONONPAGED                 00000000-XR                  174    (1)    #-405    (1)
EXE$DEANONPAGED                 00000000-XR                  174    (1)    #-472    (1)    #-597    (1)
EXIT                            000005A1-R  1055    (1)    #-1052   (1)
FAB$B_DNS                      =00000035                     #-381    (1)
FAB$B_FNS                      =00000034                     #-378    (1)
FAB$C_BID                      =00000003                      245    (1)
FAB$C_BLN                      =00000050                      245    (1)
FAB$C_SEQ                      =00000000                      245    (1)
FAB$C_VAR                      =00000002                      245    (1)
FAB$L_ALQ                      =00000010                      245    (1)
FAB$L_DNA                      =00000030                     #-383    (1)
FAB$L_FNA                      =0000002C                     #-380    (1)
FAB$L_FOP                      =00000004                      245    (1)
FAB$V_CHAN_MODE                =00000002                      245    (1)
FAB$V_FILE_MODE                =00000004                      245    (1)
FAB$V_LNM_MODE                 =00000000                      245    (1)
FAB$W_GBC                      =00000048                      245    (1)
FALSE                          =00000000    203    (1)
FMT_PRMT                        00000007-R  212    (1)    #-689    (1)      690    (1)    #-692    (1)
INIT_CONTEXT                    00000000-XR                  176    (1)      363    (1)
IO$_READPROMPT                  00000000-XR                  172    (1)    #-985    (1)
IO$_READVBLK                    00000000-XR                 #-1015  (1)      172    (1)
IO$_WRITEVBLK                   00000000-XR                 #-1003  (1)      172    (1)
L_FLAGS                         000000D0-R   257    (1)
L_LINK                          000000CC-R   254    (1)    #-318    (1)    #-413    (1)    #-414    (1)    #-534    (1)
                                                           #-547    (1)    #-594    (1)    #-596    (1)    #-599    (1)
                                                           #-711    (1)
OFF                            =00000000    203    (1)
ON                             =00000001    203    (1)
PR$_SID_TYP8NN                 =00000006    190    (1)    #-520    (1)
PR$_SID_TYP8RR                 =00000011    191    (1)    #-522    (1)
PRINT_LINE                      00000308-R   781    (1)    #-725    (1)    #-732    (1)    #-768    (1)
PRMT_ADR                        000000C0-R   248    (1)
PRMT_LEN                        000000C8-R   252    (1)
PRNT_PRMT                       00000000-XR                  182    (1)    #-678    (1)
QIO$CLEANUP                     00000000-XR                  176    (1)      365    (1)
RAB$B_RAC                      =0000001E                     246    (1)
RAB$C_BID                      =00000001                     246    (1)
RAB$C_BLN                      =00000044                     246    (1)
RAB$C_SEQ                      =00000000                     246    (1)
RAB$L_CTX                      =00000018                     246    (1)
RAB$L_RBF                      =00000028                    #-898    (1)
RAB$L_ROP                      =00000004                     246    (1)
RAB$L_UBF                      =00000024                    #-427    (1)    #-885    (1)
RAB$W_RSZ                      =00000022                    #-435    (1)    #-895    (1)
RAB$W_USZ                      =00000020                    #-428    (1)    #-431    (1)    #-886    (1)
RMS$_EOF                       =0001827A                    #-451    (1)    #-889    (1)
SC$A_BUF                        00000010    276    (1)    #-1172   (1)     1181    (1)      417    (1)    #-418    (1)
                                                           #-464    (1)      750    (1)
SC$C_BLN                       =00000010    277    (1)      404    (1)
SC$L_LINK                       00000000    270    (1)    #-413    (1)    #-596    (1)
```

```
SC$W_CURR                       0000000C     274    (1)    #-1165   (1)    #-1178   (1)    #-1179   (1)    #-749    (1)
SC$W_END                        0000000E     275    (1)    #-1170   (1)    #-464    (1)
SC$W_LEN                        0000000A     273    (1)    #-1166   (1)    #-1180   (1)    #-751    (1)    #-758    (1)
                                                           #-759    (1)
SC$W_SIZE                       00000008     272    (1)    #-412    (1)    #-466    (1)
SCAN$COMPARE                    00000000-XR                   167    (1)    #-737    (1)
SCAN$SPACES                     00000000-XR                   167    (1)    #-762    (1)
SCB$L_TIMER                     000000C0                   #-527    (1)
SCRIPT$CONT                     000001A2-R   545    (1)
SCRIPT$FINISH                   000001B5-R   591    (1)    #-1185   (1)    #-454    (1)    #-533    (1)
SCRIPT$FLUSH                    00000157-R   519    (1)    #-362    (1)
SCRIPT$INIT                     00000000-R   317    (1)
SCRIPT$MINCORE                  =00000400    189    (1)
SCRIPT$OPEN                     0000000A-R   361    (1)
SCRIPT$OPEN_X                   00000151-R   476    (1)    #-396    (1)    #-456    (1)
SCRIPT$STOP                     00000199-R   540    (1)    #-319    (1)    #-546    (1)    #-602    (1)
SCRIPT_FAB                      0000002C-R   245    (1)     246     (1)     375     (1)
SCRIPT_RAB                      0000007C-R   246    (1)     388     (1)
SCRIPT_XAB                      00000000-R   244    (1)     245     (1)     371     (1)
SIZ...                          =00000001    195    (1)     195     (1)
SS$_CONTROLC                    00000000-XR                   171     (1)
SS$_ENDOFFILE                   00000000-XR                #-1031   (1)    #-1186   (1)     171     (1)    #-891    (1)
SS$_ILLIOFUNC                   00000000-XR                   171     (1)    #-988    (1)
SS$_NORMAL                      00000000-XR                #-1044   (1)    #-1182   (1)     171     (1)    #-604    (1)
                                                           #-772    (1)    #-943    (1)    #-970    (1)
SYS$CLOSE                       00000000-XR                   449     (1)
SYS$CONNECT                     00000000-XR                   424     (1)
SYS$DISCONNECT                  00000000-XR                   447     (1)
SYS$FAO                         00000000-XR                   170     (1)     823     (1)     847     (1)
SYS$GET                         00000000-XR                   433     (1)     887     (1)
SYS$HIBER                       00000000-XR                   170     (1)
SYS$OPEN                        00000000-XR                   393     (1)
SYS$QIO                         00000000-XR                   170     (1)
SYS$QIOW                        00000000-XR                  1003     (1)    1015     (1)     170     (1)     985     (1)
SYS$READEF                      00000000-XR                   170     (1)
SYS$WAKE                        00000000-XR                   170     (1)
TRUE                            =00000001    203    (1)
T_DEF_COM                        0000000E-R  215    (1)     382     (1)
T_ECHO                           00000045-R  236    (1)     901     (1)
T_EOF                            00000013-R  221    (1)     723     (1)
T_NULL                           00000012-R  218    (1)     701     (1)
T_PNF                            0000004E-R  239    (1)     743     (1)
T_PROMPT                         0000001B-R  224    (1)     785     (1)
T_PROMPT2                        00000024-R  227    (1)     797     (1)
T_PROMPT3                        0000002B-R  230    (1)     863     (1)
T_SKIP                           00000031-R  233    (1)     752     (1)
ULTRIX_FLAGS                     00000000-XR                  1109     (1)     181     (1)
ULTRIX_V_MODE                    00000000-XR               #-1109     (1)     181     (1)
V_ECHOED                        =00000001    260    (1)
XAB$C_FHC                       =0000001D                     244     (1)
XAB$C_FHCLEN                    =0000002C                     244     (1)
XAB$L_EBK                       =00000010                  #-402     (1)
XAB$L_NXT                       =00000004                     244     (1)
XAB$W_FFB                       =00000014                  #-403     (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$$R_SET	1	244 (1)	244 (1)
$$R_TABINIT	1	244 (1)	244 (1) 245 (1) 246 (1)
$$R_VBFSET	1	245 (1)	245 (1) 246 (1)
$$R_XSET	1	244 (1)	244 (1)
$CLOSE	1	449 (1)	449 (1)
$CONNECT	1	424 (1)	424 (1)
$CRD_LITERALS	5	203 (1)	203 (1)
$DEF	1	203 (1)	
$DEFINI	1	192 (1)	192 (1) 193 (1) 197 (1) 198 (1)
			199 (1) 200 (1) 201 (1)
$DISCONNECT	1	447 (1)	447 (1)
$DS_CLRVEC_S	1	527 (1)	527 (1)
$DS_DSADEF	5	193 (1)	193 (1)
$DS_DSDEF	3	194 (1)	194 (1)
$DS_SCBDEF	3	192 (1)	192 (1)
$DS_TYPEDEF	4	202 (1)	202 (1)
$EQU	1	203 (1)	194 (1) 202 (1) 203 (1)
$EQULS1	1	203 (1)	194 (1) 202 (1) 203 (1)
$EQULST	1	194 (1)	194 (1) 202 (1) 203 (1)
$FAB	4	245 (1)	245 (1)
$FABDEF	1	197 (1)	197 (1) 245 (1)
$GBLINI	2		194 (1) 195 (1) 202 (1) 203 (1)
$GET	1	433 (1)	433 (1) 887 (1)
$OPEN	1	393 (1)	393 (1)
$PUSHADR	1	985 (1)	1003 (1) 1015 (1) 985 (1)
$PUSHTWO	1	985 (1)	1003 (1) 1015 (1) 985 (1)
$QIOPUSH	1	985 (1)	1003 (1) 1015 (1) 985 (1)
$QIOW_S	1	977 (1)	1010 (1) 977 (1) 998 (1)
$RAB	2	246 (1)	246 (1)
$RABDEF	1	199 (1)	199 (1) 246 (1)
$RMSCALL	2	393 (1)	393 (1) 424 (1) 433 (1) 447 (1)
			449 (1) 887 (1)
$RMSDEF	13	198 (1)	198 (1)
$VIELD	1		195 (1)
$VIELD1	1	203 (1)	195 (1)
$XABDEF	1	200 (1)	200 (1) 244 (1)
$XABFHC	1	244 (1)	244 (1)
$XABFHCDEF	1	201 (1)	201 (1) 244 (1)
BR_IF_CRD_AUTOTEST_OFF	1	912 (1)	912 (1)
BR_IF_CRD_MENUTEST_OFF	1	914 (1)	914 (1)
BR_IF_CRD_MENUTEST_ON	1	913 (1)	913 (1)
BR_IF_NOT_BATCH	1	875 (1)	875 (1)
BR_IF_NOT_BINARY	1	803 (1)	803 (1)
BR_IF_NOT_SCRIPT	1	710 (1)	710 (1)
BR_IF_NOT_USER	1	974 (1)	1007 (1) 974 (1)
BR_IF_NOT_VERIFY	1	782 (1)	782 (1)
BR_IF_OPER	1	861 (1)	861 (1)
BR_IF_SCRIPT	1	532 (1)	532 (1)
BR_IF_USER	1	364 (1)	364 (1) 524 (1)
CLEAR_CTRLC	1	416 (1)	416 (1)

```
CLEAR_QA                  1          530    (1)      530    (1)
CLEAR_SCRIPT              1          541    (1)      541    (1)
CLEAR_STRFLG              1          415    (1)      415    (1)
CLIDEF                    4          196    (1)      196    (1)
DSFDEF                    3          195    (1)      195    (1)
MODNAM                    1          210    (1)      210    (1)
SET_SCRIPT                1          474    (1)      474    (1)      549    (1)
```

```
                                    +-------------------------+
                                    ! Opcode Cross Reference !
                                    +-------------------------+

OPCODE          VALUE     REFERENCES...
------          -----     -------------
ADDB3           0081      692    (1)
ADDL            00C0      1169   (1)
ADDL2           00C0      688    (1)
ASHL            0078      402    (1)
BBC             00E1      1007   (1)    1108   (1)    710    (1)    782    (1)    803    (1)    875    (1)    974    (1)
BBS             00E0      1109   (1)    364    (1)    524    (1)    532    (1)    861    (1)    912    (1)    913    (1)
                          914    (1)
BBSC            00E4      415    (1)    416    (1)    530    (1)    541    (1)
BBSS            00E2      474    (1)    549    (1)    740    (1)
BEQL            0013      1033   (1)    452    (1)    468    (1)    521    (1)    548    (1)    595    (1)    679    (1)
                          712    (1)    715    (1)    729    (1)    739    (1)    989    (1)
BICB2           008A      1110   (1)
BICL            00CA      470    (1)
BICW            00AA      467    (1)
BICW3           00AB      466    (1)
BISB            0088      1008   (1)    975    (1)
BLBC            00E9      1004   (1)    1005   (1)    1028   (1)    1029   (1)    1051   (1)    425    (1)    434    (1)
                          748    (1)    807    (1)    824    (1)    831    (1)    917    (1)    939    (1)    966    (1)
BLBS            00E8      395    (1)    406    (1)    722    (1)    860    (1)    888    (1)
BLEQ            0015      1101   (1)    1171   (1)    430    (1)
BLSS            0019      1167   (1)
BLSSU           001F      704    (1)    707    (1)
BNEQ            0012      523    (1)    535    (1)    600    (1)    717    (1)    731    (1)    765    (1)    809    (1)
                          811    (1)    890    (1)
BRB             0011      1106   (1)    1174   (1)    440    (1)    456    (1)    674    (1)    726    (1)    733    (1)
                          834    (1)    915    (1)
BRW             0031      1052   (1)    396    (1)    407    (1)    719    (1)    756    (1)    766    (1)    867    (1)
                          893    (1)    905    (1)    990    (1)
BSBB            0010      533    (1)    546    (1)    602    (1)    732    (1)    768    (1)
BSBW            0030      1037   (1)    1185   (1)    319    (1)    362    (1)    405    (1)    454    (1)    472    (1)
                          597    (1)    721    (1)    725    (1)    737    (1)    762    (1)    769    (1)
CALLG           00FA      365    (1)    760    (1)
CALLS           00FB      1003   (1)    1015   (1)    393    (1)    424    (1)    433    (1)    447    (1)    449    (1)
                          527    (1)    746    (1)    755    (1)    788    (1)    823    (1)    847    (1)    849    (1)
                          866    (1)    887    (1)    904    (1)    928    (1)    955    (1)    985    (1)
CLRB            0094      1016   (1)    843    (1)    987    (1)
CLRL            00D4      318    (1)    696    (1)    802    (1)
CLRQ            007C      1003   (1)    1015   (1)    409    (1)    410    (1)    691    (1)    859    (1)    985    (1)
CMPB            0091      520    (1)    522    (1)    730    (1)    808    (1)
CMPL            00D1      1032   (1)    1100   (1)    429    (1)    451    (1)    703    (1)    706    (1)    810    (1)
                          889    (1)
CMPW            00B1      1170   (1)    764    (1)    988    (1)
CVTBL           0098      745    (1)    754    (1)    787    (1)    865    (1)    903    (1)
CVTWL           0032      1165   (1)    1166   (1)
DECL            00D7      439    (1)    815    (1)    826    (1)
EXTZV           00EF      941    (1)    968    (1)
INCL            00D6      1173   (1)    683    (1)    816    (1)    817    (1)    825    (1)
JMP             0017      477    (1)    774    (1)
JSB             0016      363    (1)    394    (1)    453    (1)
MCOML           00D2      1176   (1)
```

MCOMW	00B2	758	(1)												
MOVAB	009E	1181	(1)	371	(1)	375	(1)	376	(1)	382	(1)	388	(1)	404	(1)
		417	(1)	427	(1)	437	(1)	465	(1)	469	(1)	701	(1)	723	(1)
		797	(1)	833	(1)	841	(1)	850	(1)	882	(1)	883	(1)	884	(1)
MOVAL	00DE	414	(1)	693	(1)										
MOVAQ	007E	1006	(1)	973	(1)	996	(1)								
MOVB	0090	1107	(1)	377	(1)	381	(1)	436	(1)	805	(1)	812	(1)		
MOVC3	0028	685	(1)	690	(1)										
MOVL	00D0	1036	(1)	1044	(1)	1048	(1)	1102	(1)	1104	(1)	1105	(1)	1182	(1)
		379	(1)	411	(1)	413	(1)	446	(1)	450	(1)	462	(1)	594	(1)
		596	(1)	604	(1)	673	(1)	682	(1)	700	(1)	702	(1)	705	(1)
		708	(1)	711	(1)	735	(1)	771	(1)	772	(1)	804	(1)	885	(1)
		942	(1)	943	(1)	969	(1)	970	(1)						
MOVQ	007D	783	(1)	799	(1)	819	(1)	848	(1)						
MOVW	00B0	1178	(1)	1179	(1)	1180	(1)	412	(1)	471	(1)	886	(1)	896	(1)
		897	(1)												
MOVZBL	009A	1172	(1)	684	(1)	687	(1)	689	(1)	724	(1)	736	(1)	744	(1)
		753	(1)	786	(1)	798	(1)	840	(1)	842	(1)	864	(1)	902	(1)
MOVZBW	009B	428	(1)												
MOVZWL	003C	1003	(1)	1015	(1)	1031	(1)	1035	(1)	1047	(1)	1186	(1)	403	(1)
		435	(1)	749	(1)	751	(1)	801	(1)	891	(1)	895	(1)	985	(1)
POPR	00BA	605	(1)	851	(1)										
PUSHAB	009F	743	(1)	750	(1)	752	(1)	785	(1)	800	(1)	839	(1)	844	(1)
		845	(1)	846	(1)	863	(1)	901	(1)						
PUSHAL	00DF	1003	(1)	1015	(1)	393	(1)	424	(1)	433	(1)	447	(1)	449	(1)
		887	(1)	924	(1)	985	(1)								
PUSHAQ	007F	1003	(1)	1015	(1)	820	(1)	822	(1)	862	(1)	985	(1)		
PUSHAW	003F	821	(1)												
PUSHL	00DD	1003	(1)	1015	(1)	527	(1)	742	(1)	784	(1)	898	(1)	899	(1)
		900	(1)	925	(1)	926	(1)	927	(1)	951	(1)	952	(1)	953	(1)
		954	(1)	985	(1)										
PUSHR	00BB	592	(1)	832	(1)										
RET	0004	1055	(1)	944	(1)	971	(1)								
RSB	0005	1112	(1)	1183	(1)	1187	(1)	320	(1)	537	(1)	542	(1)	551	(1)
		606	(1)	790	(1)										
SOBGEQ	00F4	1111	(1)												
SUBL	00C2	438	(1)												
SUBL2	00C2	838	(1)												
SUBL3	00C3	418	(1)	463	(1)										
SUBW3	00A3	431	(1)	464	(1)										
TSTB	0095	714	(1)	716	(1)										
TSTL	00D5	534	(1)	547	(1)	599	(1)	678	(1)	728	(1)	738	(1)		

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ADDRESS	244	(1)	245	(1)	246	(1)										
.ASCIC	210	(1)	213	(1)	219	(1)	222	(1)	225	(1)	228	(1)	231	(1)	234	(1)
	237	(1)	240	(1)												
.ASCID	836	(1)														
.ASCII	216	(1)														
.BLKB	244	(1)	245	(1)	246	(1)	276	(1)								
.BLKL	196	(1)	255	(1)	258	(1)	270	(1)	271	(1)						
.BLKW	272	(1)	273	(1)	274	(1)	275	(1)								
.BYTE	244	(1)	245	(1)	246	(1)	263	(1)								
.DSABL	15	(1)														
.END	1188	(1)														
.ENDC	1003	(1)	1015	(1)	194	(1)	195	(1)	202	(1)	203	(1)	244	(1)	245	(1)
	246	(1)	393	(1)	424	(1)	433	(1)	447	(1)	449	(1)	887	(1)	985	(1)
.ENDM	192	(1)	193	(1)	194	(1)	195	(1)	196	(1)	197	(1)	198	(1)	199	(1)
	200	(1)	201	(1)	202	(1)	203	(1)	210	(1)	244	(1)	245	(1)	246	(1)
	364	(1)	393	(1)	415	(1)	416	(1)	424	(1)	433	(1)	447	(1)	449	(1)
	474	(1)	527	(1)	530	(1)	532	(1)	541	(1)	710	(1)	782	(1)	803	(1)
	861	(1)	875	(1)	912	(1)	913	(1)	914	(1)	974	(1)	977	(1)	985	(1)
.ENDR	194	(1)	195	(1)	202	(1)	203	(1)	245	(1)	246	(1)				
.ENTRY	671	(1)	676	(1)												
.EXTRN	164	(1)	165	(1)	166	(1)	167	(1)	168	(1)	169	(1)	170	(1)	171	(1)
	172	(1)	173	(1)	174	(1)	175	(1)	176	(1)	177	(1)	178	(1)	179	(1)
	180	(1)	181	(1)	182	(1)										
.GLOBL	1003	(1)	1015	(1)	393	(1)	424	(1)	433	(1)	447	(1)	449	(1)	887	(1)
	985	(1)														
.IDENT	13	(1)														
.IF	1003	(1)	1015	(1)	194	(1)	195	(1)	202	(1)	203	(1)	244	(1)	245	(1)
	246	(1)	393	(1)	424	(1)	433	(1)	447	(1)	449	(1)	887	(1)	985	(1)
.IFF	1003	(1)	1015	(1)	194	(1)	195	(1)	202	(1)	203	(1)	245	(1)	246	(1)
	393	(1)	424	(1)	433	(1)	447	(1)	449	(1)	887	(1)	985	(1)		
.IIF	194	(1)	195	(1)	202	(1)	203	(1)	244	(1)	245	(1)	246	(1)		
.IRP	194	(1)	195	(1)	202	(1)	203	(1)	245	(1)	246	(1)				
.LIBRARY	153	(1)	154	(1)	155	(1)	156	(1)								
.LIST	192	(1)	193	(1)	196	(1)										
.LONG	249	(1)	251	(1)	253	(1)										
.MACRO	194	(1)	195	(1)	202	(1)	203	(1)								
.MDELETE	194	(1)														
.MEXIT	244	(1)														
.NLIST	192	(1)	193	(1)	196	(1)										
.NOCROSS	192	(1)	193	(1)	197	(1)	198	(1)	199	(1)	200	(1)	201	(1)		
.NOSHOW	14	(1)														
.NTYPE	393	(1)	424	(1)	433	(1)	447	(1)	449	(1)	887	(1)				
.PAGE	1021	(1)	1056	(1)	1098	(1)	1113	(1)	1163	(1)	148	(1)	158	(1)	183	(1)
	204	(1)	241	(1)	265	(1)	279	(1)	316	(1)	321	(1)	360	(1)	384	(1)
	419	(1)	441	(1)	478	(1)	518	(1)	552	(1)	590	(1)	607	(1)	61	(1)
	638	(1)	670	(1)	727	(1)	776	(1)	791	(1)	827	(1)	868	(1)	906	(1)
	991	(1)														
.PSECT	1058	(1)	1115	(1)	196	(1)	206	(1)	243	(1)	268	(1)	281	(1)	323	(1)
	480	(1)	554	(1)	609	(1)										
.RESTORE	192	(1)	193	(1)	196	(1)	197	(1)	198	(1)	199	(1)	200	(1)	201	(1)

```
                278   (1)
.SAVE           192   (1)   193   (1)   196   (1)   197   (1)   198   (1)   199   (1)   200   (1)   201   (1)
                267   (1)
.SBTTL          1057  (1)   1114  (1)   149   (1)   159   (1)   184   (1)   205   (1)   242   (1)   266   (1)
                280   (1)   322   (1)   479   (1)   553   (1)   608   (1)
.TITLE          12    (1)
.WORD           245   (1)   246   (1)
```

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...
AP	28	#-1015 (1) #-1036 (1) #-1048 (1) #-1100 (1) #-1102 (1) #-1104 (1)
		#-1105 (1) 365 (1) #-682 (1) #-684 (1) #-687 (1) #-692 (1)
		#-693 (1) #-703 (1) #-705 (1) #-706 (1) #-708 (1) 760 (1)
		#-771 (1) #-885 (1) #-886 (1) 924 (1) #-925 (1) #-942 (1)
		#-969 (1) #-985 (1)
R0	66	#-1003 (1) #-1004 (1) #-1006 (1) 1015 (1) #-1028 (1) #-1031 (1)
		#-1044 (1) #-1105 (1) #-1107 (1) 1108 (1) #-1110 (1) #-1182 (1)
		#-1186 (1) #-395 (1) #-402 (1) 404 (1) #-406 (1) #-425 (1)
		#-434 (1) #-435 (1) #-436 (1) 437 (1) #-438 (1) #-446 (1)
		#-450 (1) #-451 (1) #-462 (1) #-463 (1) 469 (1) #-470 (1)
		#-471 (1) #-594 (1) #-596 (1) #-604 (1) #-722 (1) #-749 (1)
		750 (1) #-772 (1) #-798 (1) #-799 (1) #-804 (1) #-806 (1)
		#-813 (1) #-832 (1) #-851 (1) #-888 (1) #-889 (1) #-891 (1)
		#-895 (1) #-897 (1) #-899 (1) #-939 (1) 941 (1) #-943 (1)
		#-966 (1) 968 (1) #-970 (1) #-973 (1) #-985 (1) #-988 (1)
		#-996 (1)
R1	21	#-1047 (1) #-1048 (1) #-403 (1) 404 (1) #-412 (1) #-418 (1)
		#-463 (1) #-464 (1) 465 (1) #-466 (1) #-771 (1) #-797 (1)
		#-798 (1) #-832 (1) #-851 (1) #-941 (1) #-942 (1) #-968 (1)
		#-969 (1)
R11	24	#-1165 (1) #-1166 (1) #-1170 (1) #-1172 (1) #-1178 (1) #-1179 (1)
		#-1180 (1) 1181 (1) #-411 (1) #-412 (1) #-413 (1) 414 (1)
		417 (1) #-463 (1) #-464 (1) #-466 (1) 671 (1) 676 (1)
		#-711 (1) #-749 (1) 750 (1) #-751 (1) #-758 (1) #-759 (1)
R2	26	#-376 (1) #-377 (1) #-379 (1) #-409 (1) #-410 (1) #-411 (1)
		#-446 (1) #-450 (1) #-592 (1) #-605 (1) 671 (1) 676 (1)
		#-736 (1) #-738 (1) #-802 (1) #-817 (1) #-824 (1) #-832 (1)
		#-851 (1) #-883 (1) #-885 (1) #-886 (1) 887 (1) #-895 (1)
		#-898 (1)
R3	19	#-417 (1) 427 (1) #-436 (1) 437 (1) #-462 (1) #-592 (1)
		#-605 (1) 671 (1) 676 (1) #-735 (1) #-736 (1) #-832 (1)
		#-841 (1) 844 (1) 845 (1) #-848 (1) #-851 (1) #-882 (1)
R4	33	#-1035 (1) #-1100 (1) #-1102 (1) #-1104 (1) #-1111 (1) #-1166 (1)
		#-1169 (1) #-1172 (1) #-1176 (1) #-1180 (1) #-418 (1) #-429 (1)
		#-431 (1) #-438 (1) #-439 (1) #-467 (1) #-471 (1) #-592 (1)
		#-605 (1) 671 (1) 676 (1) #-724 (1) #-728 (1) #-783 (1)
		#-832 (1) #-833 (1) #-842 (1) #-843 (1) 850 (1) #-851 (1)
		#-884 (1) #-900 (1)
R5	19	#-1036 (1) #-1107 (1) #-1165 (1) #-1169 (1) #-1170 (1) #-1172 (1)
		#-1173 (1) #-1178 (1) #-1179 (1) 1181 (1) #-592 (1) #-605 (1)
		671 (1) 676 (1) #-723 (1) #-724 (1) #-730 (1) #-764 (1)
R6	10	671 (1) #-673 (1) 676 (1) #-696 (1) #-700 (1) 740 (1)
		#-748 (1) #-807 (1) #-860 (1) #-917 (1)
R7	16	#-371 (1) #-402 (1) #-403 (1) 671 (1) 676 (1) #-701 (1)
		#-702 (1) #-705 (1) #-714 (1) #-716 (1) #-735 (1) #-742 (1)
		#-784 (1) #-808 (1) #-810 (1) #-819 (1)
R8	15	#-388 (1) 424 (1) #-427 (1) #-428 (1) #-431 (1) 433 (1)
		#-435 (1) 447 (1) 671 (1) 676 (1) #-702 (1) #-708 (1)
R9	13	#-1051 (1) #-375 (1) #-378 (1) #-380 (1) #-381 (1) #-383 (1)
		393 (1) 449 (1) 671 (1) 676 (1) #-700 (1)

```
SP                 71        #-1003   (1)    #-1005   (1)      1006   (1)    #-1015   (1)    #-1029   (1)    #-1032   (1)
                             #-1035   (1)    #-1047   (1)    #-682    (1)    #-683    (1)    #-684    (1)    #-685    (1)
                             #-687    (1)    #-688    (1)    #-689    (1)    #-690    (1)    #-691    (1)    #-744    (1)
                             #-745    (1)    #-751    (1)    #-753    (1)    #-754    (1)    #-783    (1)    #-786    (1)
                             #-787    (1)    #-799    (1)    #-801    (1)    #-804    (1)    #-815    (1)    #-816    (1)
                             #-819    (1)      820    (1)      821    (1)      822    (1)    #-825    (1)    #-826    (1)
                               833    (1)    #-838    (1)      839    (1)    #-840    (1)      841    (1)    #-842    (1)
                             #-848    (1)    #-850    (1)    #-859    (1)      862    (1)    #-864    (1)    #-865    (1)
                               884    (1)    #-896    (1)    #-897    (1)    #-902    (1)    #-903    (1)    #-951    (1)
                             #-952    (1)      973    (1)    #-985    (1)      996    (1)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.33
Command processing	895	00:00:00.24	00:00:00.78
Pass 1	1088	00:00:03.91	00:00:07.43
Symbol table sort	0	00:00:00.26	00:00:00.27
Pass 2	57	00:00:01.06	00:00:02.16
Symbol table output	2	00:00:00.06	00:00:00.29
Psect synopsis output	2	00:00:00.00	00:00:00.00
Cross-reference output	7	00:00:00.46	00:00:00.85
Assembler run totals	2075	00:00:06.01	00:00:12.11

```
The working set limit was 3400 pages.
204011 bytes (399 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1060 non-local and 60 local symbols.
1188 source lines were read in Pass 1, producing 80 object records in Pass 2.
144 pages of virtual memory were used to define 56 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	5
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	19
DISK$DS:[DS.LOCAL.LIBRARY]CRD.MLB;4	1
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	27
TOTALS (all libraries)	52

```
1465 GETS were required to define 52 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:SCRIPT.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

```
;  0001  0     ! DEC/CMS REPLACEMENT HISTORY, Element SHOWMEMORY.B32
;  0002  0     ! *3    16-SEP-1987 15:33:18 GEARIN "CHANGED MFPR CALLS TO CALL CMK MACRO [SO MODE CAN BE CHANGED IF NECCESSARY] BEF
;  0003  0     !EXAMINED"
;  0004  0     ! *2    21-MAR-1986 15:27:48 BAGGETT "DISPLAY OF ULTRIX BUFFER ADDED"
;  0005  0     ! *1     6-FEB-1986 15:22:30 DS ""
;  0006  0     ! DEC/CMS REPLACEMENT HISTORY, Element SHOWMEMORY.B32
;  0007  0     %Title '*** SHOWMEM Handle Show Memory command'
;  0008  0     Module ShowMemory (                                 ! Supervisor SHOW MEMORY command
;  0009  0                        Ident = '10.10-3',                                          !G:3
;  0010  0                        Addressing_Mode (External = Long_Relative),
;  0011  0                        Debug,
;  0012  0                        OptLevel = 3,
;  0013  0                        NoZip
;  0014  0                        ) =
;  0015  0     !
;  0016  0     ! Copyright (c) 1982, 1983, 1984, 1986                                        !G:2
;  0017  0     ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;  0018  0     !
;  0019  0     ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;  0020  0     ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;  0021  0     ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;  0022  0     ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;  0023  0     ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;  0024  0     ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;  0025  0     ! REMAIN IN DEC.
;  0026  0     !
;  0027  0     ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;  0028  0     ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;  0029  0     ! CORPORATION.
;  0030  0     !
;  0031  0     ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;  0032  0     ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;  0033  0     !
;  0034  0
;  0035  0     !++
;  0036  0     ! Facility:
;  0037  0     !
;  0038  0     !       Diagnostic Supervisor
;  0039  0     !
;  0040  0     ! Abstract:
;  0041  0     !
;  0042  0     !       List physical memory map (virtual in user mode).
;  0043  0     !
;  0044  0     ! Environment:
;  0045  0     !
;  0046  0     !       Diagnostic Supervisor
;  0047  0     !
;  0048  0     ! Author:
;  0049  0     !
;  0050  0     !       Dave Butenhof           05-Feb-1982
;  0051  0     !
;  0052  0     ! Modified by:
;  0053  0     !
;  0054  0     ! [01] Dave Butenhof, 25-Mar-1982, version 6.7
;  0055  0     !       Fix order of non-paged pool and lookaside list typeouts,
;  0056  0     !       mask out region select bits (31, 32) on P0BR and P1BR addresses
;  0057  0     !       (to make them physical, not virtual), and change top-of-DS and
```

```
;    0058  0     !       physical-memory-size messages to show Kb, not bytes (easier to
;    0059  0     !       read).  Also, print amount of available buffer space on /Buffer,
;    0060  0     !       and cut down on code some by setting all option bits if /All is
;    0061  0     !       set, instead of checking it throughout.
;    0062  0     !
;    0063  0     ! [02] Richard Brown, 9-July-82, Version 6.8
;    0064  0     !       Add printing of debugger memory allocation.
;    0065  0     !
;    0066  0     ! [03] Jack Stansbury, 14-September-1982, Version 6.9
;    0067  0     !       Added showing the buffer count.
;    0068  0     !
;    0069  0     !       04      Bob Bergazzi    May 17, 1983    version 6.11
;    0070  0     !               Changed the order of searching libraries.
;    0071  0     !
;    0072  0     !       05      Bob Bergazzi    Aug 29,1983     Version 6.13
;    0073  0     !               Added the VRSETMEM routine which sets a new memory
;    0074  0     !               size and then starts VDS at 10000. Doesn't get
;    0075  0     !               here in user mode.
;    0076  0     !
;    0077  0     !       06      Bob Bergazzi    Feb 14, 1984    Version 6.14
;    0078  0     !               Fixed VRSETMEM to call INI$LOAD_DEVICE which
;    0079  0     !               makes ptables for the console and boot devices.
;    0080  0     !                                                                       !G:2
;    0081  0     !       07      M. Baggett      10-March-1986   Version 10              !G:2
;    0082  0     !               Added printout for ULTRIX buffer.                       !G:2
;    0083  0     !                                                                       !G:3
;    0084  0     !       ******* VERSION #'s CORRESPOND TO GENERATION #'S FOR NOW ON *******  !G:3
;    0085  0     !                                                                       !G:3
;    0086  0     !       03      David Gearin    15-Sept-1987    Version 10.10           !G:3
;    0087  0     !               Modified showmemory routine to call cmk macro instead   !G:3
;    0088  0     !               of merely moving ipr values via MFPR.  This allows for  !G:3
;    0089  0     !               showing of ipr's when the current mode is not kernel.   !G:3
;    0090  0     !--
;    0091  0
;    0092  1     begin
;    0093  1     !
;    0094  1     ! Table of contents:
;    0095  1     !
;    0096  1
;    0097  1     !
;    0098  1     ! Include files:
;    0099  1     !
;    0100  1
;    0101  1     Library '$Diag';                    !                               [04]
;    0102  1
;    0103  1     Library '$Ds';                      !                               [04]
;    0104  1
;    0105  1     Library 'Sys$Library:Lib';          !                               [04]
;    0106  1
;    0107  1     !
;    0108  1     ! Included macros and literals                                       !G:3
;    0109  1     !                                                                   !G:3
;    0110  1     !                                                                   !G:3
;    0111  1     ! Macros:
;    0112  1     !                                                                   !G:3
;    0113  1             CMKDEF;                                                     !G:3
;    0114  1
```

```
;    0115  1      !                                                                                          !G:3
;    0116  1      ! Psect Definitions:
;    0117  1      !
;    0118  1
;    0119  1      Psect Plit   = Data (NoExecute, Share, NoWrite, Addressing_Mode (Long_Relative));
;    0120  1      Psect Own    = Work (NoExecute, NoShare, Write, Addressing_Mode (Long_Relative));
;    0121  1      Psect Global = Work (NoExecute, NoShare, Write, Addressing_Mode (Long_Relative));
;    0122  1      Psect Code   = Code (Execute, Share, NoWrite);
;    0123  1
;    0124  1      !
;    0125  1      ! Equated Symbols:
;    0126  1      !
;    0127  1
;    0128  1      $Ds_DsaDef;                                         ! Define DSA locations
;    0129  1      $Ds_TypeDef;                                        ! Define type codes
;    0130  1
;    0131  1      !linkage                                                                                   !G:3
;    0132  1      !       jsb_mfpr = jsb: GLOBAL (r3 = 3, r5 = 5);                                           !G:3
;    0133  1      Literal                                             ! Bits in ShowMemoryFlags
;    0134  1          Sm_V_Map = 0,                                   ! /Map
;    0135  1          Sm_V_Buffer = 1,                                ! /Buffer
;    0136  1          Sm_V_Data = 2,                                  ! /Data_Structure
;    0137  1          Sm_V_All = 3,                                   ! /All
;    0138  1          CR = 13,
;    0139  1          LF = 10;
;    0140  1
;    0141  1      Bind                                                ! Format strings
;    0142  1          Format = $Ascic ('!22<!XL !-(%D!SL)!> : !AC!/'),
;    0143  1          Format2= $Ascic ('!22<!XL !-(%D!SL)!> : !AC!AC!/'),
;    0144  1          Format3= $Ascic ('!22<!XL !-(%D!SL)!> :  !AC!AC!/'),
;    0145  1          Format4= $Ascic ('!22<!XL (!SLKb)!> : !AC!/'),      !                              [01]
;    0146  1          Format5= $Ascic ('!22<!XL (!SLKb)!> : !AC!AC!/'),   !                              [01]
;    0147  1          Format6= $Ascic ('!22  : !AC!/');                   !              [02]
;    0148  1
;    0149  1      Bind                                                ! Common text parts
;    0150  1          T_Reserved  = $Ascic ('Reserved page'),
;    0151  1          T_DS        = $Ascic ('Diagnostic Supervisor'),
;    0152  1          T_Kernel    = $Ascic ('Kernel'),
;    0153  1          T_Interrupt = $Ascic ('Interrupt'),
;    0154  1          T_Executive = $Ascic ('Executive'),
;    0155  1          T_Supervisor= $Ascic ('Supervisor'),
;    0156  1          T_User      = $Ascic ('User'),
;    0157  1          T_Stack     = $Ascic (' stack base'),
;    0158  1          T_SP        = $Ascic (' SP'),
;    0159  1          T_P0        = $Ascic ('P0'),
;    0160  1          T_P1        = $Ascic ('P1'),
;    0161  1          T_System    = $Ascic ('System'),
;    0162  1          T_Base      = $Ascic (' Page Table base'),
;    0163  1          T_Length    = $Ascic (' Page Table length (longwords)'),
;    0164  1          T_P0_BufCnt = $Ascic ('Pages in P0 space in use for buffers'),      !              [03]
;    0165  1          T_P1_BufCnt = $Ascic ('Pages in P1 space in use for buffers'),      !              [03]
;    0166  1          T_S0_BufCnt = $Ascic ('Pages in System space in use for buffers'),  !              [03]
;    0167  1          T_BufferSpace = $Ascic ('Available  buffer space ');                !              [01]
;    0168  1
;    0169  1      !
;    0170  1      ! Own storage:
;    0171  1      !
```

```
;    0172  1
;    0173  1     Global
;    0174  1         ShowMemoryFlags : BitVector [4],                                    !G:3
;    0175  1         ds$gl_ipr_value: long;                                              !G:3
;    0176  1
;    0177  1     !
;    0178  1     ! External references:
;    0179  1     !
;    0180  1
;    0181  1     External
;    0182  1         Ds$GB_TypeCode : Byte,                  ! Type code
;    0183  1         Ds$GL_Flags,                            ! Common flags (LODFLG)
;    0184  1         DS$GL_BufCnt : VECTOR [3, LONG],        ! The three buffer counts                  [03]
;    0185  1         Ax_Buff,                                ! Actual end of DS image
;    0186  1         Phd_Base,                               ! Base of Process Header
;    0187  1         Pcb_Base,                               ! Base of Hardware PCB
;    0188  1         Ds$AX_Jib,                              ! Base of pseudo JIB
;    0189  1         Ds$AX_Arb,                              ! Base of pseudo ARB
;    0190  1         Ds$AX_SoftPCB,                          ! Base of pseudo software PCB
;    0191  1         Scb_Base,                               ! Base of SCB
;    0192  1         Stack_Base,                             ! Base of stack space
;    0193  1         KStkPtr,                                ! Beginning of Kernel stack
;    0194  1         IStkPtr,                                !
;    0195  1         EStkPtr,                                !
;    0196  1         SStkPtr,                                !
;    0197  1         UStkPtr,                                !
;    0198  1         Ds$A_PrgBgn,                            ! Where program starts
;    0199  1         POPT_Base,                              ! End of built-in storage (_LAST)
;    0200  1         Ds$GL_LookAside,                        ! Start of Look-aside IRP list
;    0201  1         Exe$GL_Splitadr,                        ! Address of non-paged pool split
;    0202  1         Ds$GA_LastAdr,                          ! End of run-time DS
;    0203  1         Ds$GL_MemSize,                          ! Size of physical memory
;    0204  1         DS$GL_SET_MEMSIZE,                      ! Size of physical memory desired          [05]
;    0205  1         debug_lo,                               ! Bottom of debugger                       [02]
;    0206  1         debug_hi,                               ! Top of debugger                          [02]
;    0207  1         ultrix_iob_add;                         ! Start of ULTRIX buffer space  [07]  !G:2
;    0208  1
;    0209  1     Forward Routine
;    0210  1         DSR$CheckLoad   : Jsb_None,
;    0211  1         DSV$ShowMemory  : Jsb_None NoValue,
;    0212  1         ShowMemory      : NoValue,
;    0213  1         VRSETMEM        : JSB_CLI ;             !                                          [05]
```

```
;   0215  1     %Sbttl 'Is a program loaded, really?'
;   0216  1
;   0217  1     Global Routine DSR$CheckLoad : Jsb_None =
;   0218  2     Begin                                                                                    ![03
;   0219  2         Return ((.Ds$GL_Flags <Ds$V_LodFlg, 0>) And (.L$L_Environ <Env$V_Super, Env$S_Super> Eql Env$_Super));  ![03
;   0220  1     End;                                                                                     ![03


                                                        .TITLE  SHOWMEMORY *** SHOWMEM Handle Show Memory comma
;                                                                   nd
                                                        .IDENT  \10.10-3\

                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

21 44 25 28 2D 21 20 4C 58 21 3C 32 32 21 1B 00000 P.AAA:  .ASCII  <27>\!22<!XL !-(%D!SL)!> : !AC!/\          ;
      2F 21 43 41 21 20 3A 20 3E 21 29 4C 53 0000F                                                        ;
21 44 25 28 2D 21 20 4C 58 21 3C 32 32 21 1E 0001C P.AAB:  .ASCII  <30>\!22<!XL !-(%D!SL)!> : !AC!AC!/\       ;
21 43 41 21 43 41 21 20 3A 20 3E 21 29 4C 53 0002B                                                        ;
                                        2F 0003A                                                          ;
21 44 25 28 2D 21 20 4C 58 21 3C 32 32 21 1F 0003B P.AAC:  .ASCII  <31>\!22<!XL !-(%D!SL)!> :  !AC!AC!/\      ;
43 41 21 43 41 21 20 20 3A 20 3E 21 29 4C 53 0004A                                                        ;
                                        2F 21 00059                                                       ;
62 4B 4C 53 21 28 20 4C 58 21 3C 32 32 21 19 0005B P.AAD:  .ASCII  <25>\!22<!XL (!SLKb)!> : !AC!/\          ;
      2F 21 43 41 21 20 3A 20 3E 21 29 0006A                                                              ;
62 4B 4C 53 21 28 20 4C 58 21 3C 32 32 21 1C 00075 P.AAE:  .ASCII  <28>\!22<!XL (!SLKb)!> : !AC!AC!/\       ;
   2F 21 43 41 21 43 41 21 20 3A 20 3E 21 29 00084                                                        ;
      2F 21 43 41 21 20 3A 20 20 32 32 21 0C 00092 P.AAF:  .ASCII  <12>\!22  : !AC!/\                       ;
   65 67 61 70 20 64 65 76 72 65 73 65 52 0D 0009F P.AAG:  .ASCII  <13>\Reserved page\                      ;
70 75 53 20 63 69 74 73 6F 6E 67 61 69 44 15 000AD P.AAH:  .ASCII  <21>\Diagnostic Supervisor\             ;
                        72 6F 73 69 76 72 65 000BC                                                        ;
                        6C 65 6E 72 65 4B 06 000C3 P.AAI:  .ASCII  <6>\Kernel\                            ;
                  74 70 75 72 72 65 74 6E 49 09 000CA P.AAJ:  .ASCII  <9>\Interrupt\                       ;
                  65 76 69 74 75 63 65 78 45 09 000D4 P.AAK:  .ASCII  <9>\Executive\                       ;
               72 6F 73 69 76 72 65 70 75 53 0A 000DE P.AAL:  .ASCII  <10>\Supervisor\                     ;
                              72 65 73 55 04 000E9 P.AAM:  .ASCII  <4>\User\                              ;
            65 73 61 62 20 6B 63 61 74 73 20 0B 000EE P.AAN:  .ASCII  <11>\ stack base\                    ;
                              50 53 20 03 000FA P.AAO:  .ASCII  <3>\ SP\                                  ;
                              30 50 02 000FE P.AAP:  .ASCII  <2>\P0\                                      ;
                              31 50 02 00101 P.AAQ:  .ASCII  <2>\P1\                                      ;
                        6D 65 74 73 79 53 06 00104 P.AAR:  .ASCII  <6>\System\                            ;
61 62 20 65 6C 62 61 54 20 65 67 61 50 20 10 0010B P.AAS:  .ASCII  <16>\ Page Table base\                   ;
                                          65 73 0011A                                                     ;
65 6C 20 65 6C 62 61 54 20 65 67 61 50 20 1E 0011C P.AAT:  .ASCII  <30>\ Page Table length (longwords)\    ;
73 64 72 6F 77 67 6E 6F 6C 28 20 68 74 67 6E 0012B                                                        ;
                                          29 0013A                                                        ;
70 73 20 30 50 20 6E 69 20 73 65 67 61 50 24 0013B P.AAU:  .ASCII  \$Pages in P0 space in use for buffers\  ;
20 72 6F 66 20 65 73 75 20 6E 69 20 65 63 61 0014A                                                        ;
                        73 72 65 66 66 75 62 00159                                                        ;
70 73 20 31 50 20 6E 69 20 73 65 67 61 50 24 00160 P.AAV:  .ASCII  \$Pages in P1 space in use for buffers\  ;
20 72 6F 66 20 65 73 75 20 6E 69 20 65 63 61 0016F                                                        ;
                        73 72 65 66 66 75 62 0017E                                                        ;
65 74 73 79 53 20 6E 69 20 73 65 67 61 50 28 00185 P.AAW:  .ASCII  \(Pages in System space in use for buffer\ ;
20 65 73 75 20 6E 69 20 65 63 61 70 73 20 6D 00194                                                        ;
                  72 65 66 66 75 62 20 72 6F 66 001A3                                                     ;
                                          73 001AD          .ASCII  \s\                                     ;
66 75 62 20 20 65 6C 62 61 6C 69 61 76 41 18 001AE P.AAX:  .ASCII  <24>\Available  buffer space \         ;
                  20 65 63 61 70 73 20 72 65 66 001BD                                                     ;
```

```
                        .PSECT  WORK,NOEXE,2

00000 SHOWMEMORYFLAGS::
                        .BLKB   1
00001                   .BLKB   3
00004 DS$GL_IPR_VALUE::
                        .BLKB   4

              DSA$AL_APTMAIL=     65024
              DSA$GL_FLAGS=       65024
              DSA$GL_APTCOM=      65028
              DSA$GL_PASSES=      65032
              DSA$GL_UNITS=       65036
              DSA$GL_SECTNO=      65040
              DSA$GL_SID=         65044
              DSA$GL_MSGTYP=      65088
              DSA$GL_ERRNO=       65092
              DSA$GL_EVENT=       65096
              DSA$GL_SUBTNO=      65100
              DSA$GL_TESTNO=      65104
              DSA$GL_PASSNO=      65108
              DSA$GL_DEVLEN=      65112
              DSA$GL_DEVNAM=      65116
              DSA$GL_MSGPTR=      65128
              FORMAT=             P.AAA
              FORMAT2=            P.AAB
              FORMAT3=            P.AAC
              FORMAT4=            P.AAD
              FORMAT5=            P.AAE
              FORMAT6=            P.AAF
              T_RESERVED=         P.AAG
              T_DS=               P.AAH
              T_KERNEL=           P.AAI
              T_INTERRUPT=        P.AAJ
              T_EXECUTIVE=        P.AAK
              T_SUPERVISOR=       P.AAL
              T_USER=             P.AAM
              T_STACK=            P.AAN
              T_SP=               P.AAO
              T_P0=               P.AAP
              T_P1=               P.AAQ
              T_SYSTEM=           P.AAR
              T_BASE=             P.AAS
              T_LENGTH=           P.AAT
              T_P0_BUFCNT=        P.AAU
              T_P1_BUFCNT=        P.AAV
              T_S0_BUFCNT=        P.AAW
              T_BUFFERSPACE=      P.AAX
                      .EXTRN  DS$GB_TYPECODE, DS$GL_FLAGS
                      .EXTRN  DS$GL_BUFCNT, AX_BUFF
                      .EXTRN  PHD_BASE, PCB_BASE
                      .EXTRN  DS$AX_JIB, DS$AX_ARB
                      .EXTRN  DS$AX_SOFTPCB, SCB_BASE
                      .EXTRN  STACK_BASE, KSTKPTR
                      .EXTRN  ISTKPTR, ESTKPTR
                      .EXTRN  SSTKPTR, USTKPTR
```

```
                                                        .EXTRN  DS$A_PRGBGN, POPT_BASE
                                                        .EXTRN  DS$GL_LOOKASIDE
                                                        .EXTRN  EXE$GL_SPLITADR
                                                        .EXTRN  DS$GA_LASTADR, DS$GL_MEMSIZE
                                                        .EXTRN  DS$GL_SET_MEMSIZE
                                                        .EXTRN  DEBUG_LO, DEBUG_HI
                                                        .EXTRN  ULTRIX_IOB_ADD

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                        50  D4 00000 DSR$CHECKLOAD::
                                                        CLRL    R0                                      ; 0219
        01 00000204   9F        08      02  ED 00002            CMPZV   #2, #8, @#^X00000204, #1        ;
                                        02  12 0000B            BNEQ    1$                              ;
                                        50  D6 0000D            INCL    R0                              ;
        51 00000000G  EF        01      01  EF 0000F 1$:        EXTZV   #1, #1, DS$GL_FLAGS, R1         ;
                                51      51  D2 00018            MCOML   R1, R1                          ;
                                50      51  CA 0001B            BICL2   R1, R0                          ;
                                        05  0001E              RSB                                     ; 0220

; Routine Size:  31 bytes,    Routine Base:  CODE + 0000
```

```
;   0222  1     %Sbttl 'DSV$ShowMemory routine'
;   0223  1
;   0224  1     Global Routine DSV$ShowMemory : Jsb_None NoValue =
;   0225  2         Begin
;   0226  2         ShowMemory ()                                   ! Just CALLS routine
;   0227  1         End;



                        0000V  CF          00  FB 00000 DSV$SHOWMEMORY::
                                                        CALLS   #0, SHOWMEMORY                  ; 0226
                                               05 00005 RSB                                     ; 0227

; Routine Size:  6 bytes,    Routine Base:  CODE + 001F
```

```
;    0229  1     %Sbttl 'ShowMemory Routine'
;    0230  1
;    0231  1     Routine ShowMemory : NoValue =
;    0232  2         Begin
;    0233  2
;    0234  2         Local
;    0235  2             temp_psl: BitVector [32],                                          !G:3
;    0236  2             Temp,
;    0237  2             MemoryFlags : BitVector [4],
;    0238  2             OnLine : Byte;
;    0239  2                                                                                !G:3
;    0240  2          global register                                                       !G:3
;    0241  2             r3 = 3, r5 = 5;                                                    !G:3
;    0242  2
;    0243  2         MemoryFlags = .ShowMemoryFlags<0, 4>;          ! Get flags locally
;    0244  2
;    0245  2         If .MemoryFlags Eql 0                          ! Default is /Map
;    0246  2         Then
;    0247  2             MemoryFlags [Sm_V_Map] = 1;
;    0248  2                                                                                !G:3
;    0249  2         If .MemoryFlags [Sm_V_All]                     ! Set all option bits if /All                    [01]
;    0250  2         Then                                           !                                                [01]
;    0251  3             Begin                                      !                                                [01]
;    0252  3             MemoryFlags [Sm_V_Map] = 1;                !                                                [01]
;    0253  3             MemoryFlags [Sm_V_Buffer] = 1;             !                                                [01]
;    0254  3             MemoryFlags [Sm_V_Data] = 1                !                                                [01]
;    0255  2             End;                                       !                                                [01]
;    0256  2
;    0257  2         OnLine = .Dsa$V_User;                          ! Get user mode bit locally
;    0258  2         Ds$GB_TypeCode = Ds$K_Type_Command_Out;
;    0259  2
;    0260  2         If .MemoryFlags [Sm_V_Map]                     !                                                [01]
;    0261  2         Then
; P  0262  2             $Ds_PrintF (                               ! First page of memory
; P  0263  2                 Format,
; P  0264  2                 0,
; P  0265  2                 T_Reserved
;    0266  2                 );
;    0267  2
;    0268  2         If .MemoryFlags [Sm_V_Map]                     !                                                [01]
;    0269  2         Then
; P  0270  2             $Ds_PrintF (                               ! Begin of diagnostic
; P  0271  2                 Format,
; P  0272  2                 Ds$A_PrgBgn,
; P  0273  2                 $Ascic ('Beginning of diagnostic region')
;    0274  2                 );
;    0275  2
;    0276  2         If .MemoryFlags [Sm_V_Map] And DSR$CheckLoad ()    !                                            [01]
;    0277  2         Then
; P  0278  2             $Ds_PrintF (                               ! End of diagnostic
; P  0279  2                 Format,
; P  0280  2                 .L$A_LastAdr,
; P  0281  2                 $Ascic ('End of loaded diagnostic')
;    0282  2                 );
;    0283  2
;    0284  2         If .MemoryFlags [Sm_V_Buffer]                  ! If /Buffer or /All                             [01]
;    0285  2          And Dsr$CheckLoad ()                          ! .. and diagnostic is loaded                    [01]
```

```
;   0286  2          Then                                                  !                                                  [01]
; P 0287  2              $Ds_PrintF (                                      ! Available buffer space                           [01]
; P 0288  2                  Format5,                                      !                                                  [01]
; P 0289  2                  %X'FA00' - .L$A_LastAdr,                      ! .. amount available                              [01]
; P 0290  2                  (%X'FA00' - .L$A_LastAdr)/1024,               ! .. same in Kb                                    [01]
; P 0291  2                  T_BufferSpace,                                !                                                  [01]
; P 0292  2                  $Ascic ('below Supervisor')                   !                                                  [01]
;   0293  2                  );                                            !                                                  [01]
;   0294  2
;   0295  2          If .MemoryFlags [Sm_V_Map] Or .MemoryFlags [Sm_V_Data]      !                                            [01]
;   0296  2          Then
; P 0297  2              $Ds_PrintF (                                      ! APT Ptext buffer
; P 0298  2                  Format,
; P 0299  2                  %X'FA00',
; P 0300  2                  $Ascic ('APT Ptext buffer')
;   0301  2                  );
;   0302  2
;   0303  2          If .MemoryFlags [Sm_V_Map]                            !                                                  [01]
;   0304  2           Or .MemoryFlags [Sm_V_Data]
;   0305  2          Then
; P 0306  2              $Ds_PrintF (                                      ! APT mailbox
; P 0307  2                  Format,
; P 0308  2                  %X'FE00',
; P 0309  2                  $Ascic ('APT Mailbox')
;   0310  2                  );
;   0311  2
;   0312  2          If .MemoryFlags [Sm_V_Map]                            !                                                  [01]
;   0313  2          Then
; P 0314  2              $Ds_PrintF (                                      ! Begin of Supervisor
; P 0315  2                  Format2,
; P 0316  2                  %X'10000',
; P 0317  2                  $Ascic ('Beginning of '),
; P 0318  2                  T_DS
;   0319  2                  );
;   0320  2
;   0321  2          If .MemoryFlags [Sm_V_Map]                            !                                                  [01]
;   0322  2          Then
; P 0323  2              $Ds_PrintF (                                      ! Top of DS image
; P 0324  2                  Format,
; P 0325  2                  Ax_Buff,
; P 0326  2                  $Ascic ('End of Supervisor load image')
;   0327  2                  );
;   0328  2
;   0329  2          If .MemoryFlags [Sm_V_Data] And Not .OnLine           !                                                  [01]
;   0330  2          Then
;   0331  3              Begin
; P 0332  3              $Ds_PrintF (                                      ! PHD
; P 0333  3                  Format,
; P 0334  3                  Phd_Base,
; P 0335  3                  $Ascic ('PHD')
;   0336  3                  );
; P 0337  3              $Ds_PrintF (                                      ! hardware PCB
; P 0338  3                  Format,
; P 0339  3                  Pcb_Base,
; P 0340  3                  $Ascic ('Hardware PCB')
;   0341  3                  );
; P 0342  3              $Ds_PrintF (                                      ! JIB
```

```
; P 0343  3                   Format,
; P 0344  3                   Ds$AX_Jib,
; P 0345  3                   $Ascic ('JIB')
;    0346  3                   );
; P 0347  3               $Ds_PrintF (                                ! ARB
; P 0348  3                   Format,
; P 0349  3                   Ds$AX_Arb,
; P 0350  3                   $Ascic ('ARB')
;    0351  3                   );
; P 0352  3               $Ds_PrintF (                                ! software PCB
; P 0353  3                   Format,
; P 0354  3                   Ds$AX_SoftPCB,
; P 0355  3                   $Ascic ('Software PCB')
;    0356  3                   );
; P 0357  3               $Ds_PrintF (                                ! SCB
; P 0358  3                   Format,
; P 0359  3                   SCB_Base,
; P 0360  3                   $Ascic ('SCB')
;    0361  3                   );                                                          !G:3
; P 0362  3               $Ds_PrintF (                                ! Reserved page
; P 0363  3                   Format,
; P 0364  3                   Stack_Base,
; P 0365  3                   T_Reserved
;    0366  3                   );
;    0367  3               r3 = pr$_ksp;                                                   !G:3
;    0368  3               r5 = 1;                                                         !G:3
;    0369  3               cmk (sgipr);                                                    !G:3
; P 0370  3               $Ds_PrintF (                                ! Kernel Stack pointer
; P 0371  3                   Format2,                                !                   [03]
; P 0372  3                   .ds$gl_ipr_value,                                           !G:3
; P 0373  3                   T_Kernel,
; P 0374  3                   T_SP
;    0375  3                   );
; P 0376  3               $Ds_PrintF (                                ! Kernel Stack pointer base
; P 0377  3                   Format3,                                !                   [03]
; P 0378  3                   KStkPtr,
; P 0379  3                   T_Kernel,
; P 0380  3                   T_Stack
;    0381  3                   );
;    0382  3               r3 = pr$_isp;                                                   !G:3
;    0383  3               r5 = 1;                                                         !G:3
;    0384  3               cmk (sgipr);                                                    !G:3
; P 0385  3               $Ds_PrintF (                                ! Interrupt Stack pointer
; P 0386  3                   Format2,                                !                   [03]
; P 0387  3                   .ds$gl_ipr_value,                                           !G:3
; P 0388  3                   T_Interrupt,
; P 0389  3                   T_SP
;    0390  3                   );
; P 0391  3               $Ds_PrintF (                                ! Interrupt Stack pointer base
; P 0392  3                   Format3,                                !                   [03]
; P 0393  3                   IStkPtr,
; P 0394  3                   T_Interrupt,
; P 0395  3                   T_Stack
;    0396  3                   );
;    0397  3               r3 = pr$_esp;                                                   !G:3
;    0398  3               r5 = 1;                                                         !G:3
;    0399  3               cmk (sgipr);                                                    !G:3
```

```
; P 0400  3          $Ds_PrintF (                          ! Executive Stack pointer
; P 0401  3              Format2,                          !                                        [03]
; P 0402  3              .ds$gl_ipr_value,                                                    !G:3
; P 0403  3              T_Executive,
; P 0404  3              T_SP
;   0405  3              );
; P 0406  3          $Ds_PrintF (                          ! Executive Stack pointer base
; P 0407  3              Format3,                          !                                        [03]
; P 0408  3              EStkPtr,
; P 0409  3              T_Executive,
; P 0410  3              T_Stack
;   0411  3              );
;   0412  3          r3 = pr$_ssp;                                                            !G:3
;   0413  3          r5 = 1;                                                                  !G:3
;   0414  3          cmk (sgipr);                                                             !G:3
; P 0415  3          $Ds_PrintF (                          ! Supervisor Stack pointer
; P 0416  3              Format2,                          !                                        [03]
; P 0417  3              .ds$gl_ipr_value,                                                    !G:3
; P 0418  3              T_Supervisor,
; P 0419  3              T_SP
;   0420  3              );
; P 0421  3          $Ds_PrintF (                          ! Supervisor Stack pointer base
; P 0422  3              Format3,                          !                                        [03]
; P 0423  3              SStkPtr,
; P 0424  3              T_Supervisor,
; P 0425  3              T_Stack
;   0426  3              );
;   0427  3          r3 = pr$_usp;                                                            !G:3
;   0428  3          r5 = 1;                                                                  !G:3
;   0429  3          cmk (sgipr);                                                             !G:3
; P 0430  3          $Ds_PrintF (                          ! User Stack pointer
; P 0431  3              Format2,                          !                                        [03]
; P 0432  3              .ds$gl_ipr_value,                                                    !G:3
; P 0433  3              T_User,
; P 0434  3              T_SP
;   0435  3              );
; P 0436  3          $Ds_PrintF (                          ! User Stack pointer base
; P 0437  3              Format3,                          !                                        [03]
; P 0438  3              UStkPtr,
; P 0439  3              T_User,
; P 0440  3              T_Stack
;   0441  3              );
;   0442  3          r3 = pr$_p0br;                                                           !G:3
;   0443  3          r5 = 1;                                                                  !G:3
;   0444  3          cmk (sgipr);                                                             !G:3
; P 0445  3          $Ds_PrintF (                          ! P0 Page table                    !G:3
; P 0446  3              Format2,
; P 0447  3              .ds$gl_ipr_value<0, 30>,                  !
; P 0448  3              T_P0,
; P 0449  3              T_Base
;   0450  3              );
;   0451  3          r3 = pr$_p0lr;                                                           !G:3
;   0452  3          r5 = 1;                                                                  !G:3
;   0453  3          cmk (sgipr);                                                             !G:3
; P 0454  3          $Ds_PrintF (                          ! P0 Page table
; P 0455  3              Format3,                                                             !G:3
; P 0456  3              .ds$gl_ipr_value,                                                    !G:3
```

```
; P 0457  3                   T_P0,
; P 0458  3                   T_Length
;    0459  3                   );
;    0460  3               r3 = pr$_p1br;                                                          !G:3
;    0461  3               r5 = 1;                                                                 !G:3
;    0462  3               cmk (sgipr);                                                            !G:3
; P 0463  3               $Ds_PrintF (                        ! P1 Page table                     !G:3
; P 0464  3                   Format2,
; P 0465  3                   .ds$gl_ipr_value<0, 30>,            !
; P 0466  3                   T_P1,
; P 0467  3                   T_Base
;    0468  3                   );
;    0469  3               r3 = pr$_p1lr;                                                          !G:3
;    0470  3               r5 = 1;                                                                 !G:3
;    0471  3               cmk (sgipr);                                                            !G:3
; P 0472  3               $Ds_PrintF (                        ! P1 Page table                     !G:3
; P 0473  3                   Format3,                                                            !G:3
; P 0474  3                   .ds$gl_ipr_value,                                                   !G:3
; P 0475  3                   T_P1,
; P 0476  3                   T_Length
;    0477  3                   );
;    0478  3               r3 = pr$_sbr;                                                           !G:3
;    0479  3               r5 = 1;                                                                 !G:3
;    0480  3               cmk (sgipr);                                                            !G:3
; P 0481  3               $Ds_PrintF (                        ! System Page table
; P 0482  3                   Format2,
; P 0483  3                   .ds$gl_ipr_value,                                                   !G:3
; P 0484  3                   T_System,
; P 0485  3                   T_Base
;    0486  3                   );                                                                  !G:3
;    0487  3               r3 = pr$_slr;                                                           !G:3
;    0488  3               r5 = 1;                                                                 !G:3
;    0489  3               cmk (sgipr);                                                            !G:3
; P 0490  3               $Ds_PrintF (                        ! System Page table
; P 0491  3                   Format3,
; P 0492  3                   .ds$gl_ipr_value,                                                   !G:3
; P 0493  3                   T_System,
; P 0494  3                   T_Length
;    0495  4                   )
;    0496  2               End;
;    0497  2           If .MemoryFlags [Sm_V_Map]              !                                                   [07]
;    0498  2            Then                                                                       !G:2
; P 0499  2               $Ds_PrintF (                        ! ULTRIX buffer space   [07]        !G:2
; P 0500  2                   Format,                                                             !G:2
; P 0501  2                   .ultrix_iob_add,                                                    !G:2
; P 0502  2                   $Ascic ('Start of ULTRIX buffer space')                             !G:2
;    0503  2                   );                                                                  !G:2
;    0504  2
;    0505  2            If .MemoryFlags [Sm_V_Map] Or .MemoryFlags [Sm_V_Data]      !                       [01]
;    0506  2            Then
;    0507  3               Begin
; P 0508  3               $Ds_PrintF (                        ! Non-Paged Pool
; P 0509  3                   Format,
; P 0510  3                   .Ds$GL_LookAside,
; P 0511  3                   $Ascic ('Start of Supervisor memory pool')
;    0512  3                   );
;    0513  3
```

```
;    0514  3              If .Ds$GL_LookAside Neq .Exe$GL_SplitAdr
;    0515  3              Then
; P  0516  3                  $Ds_PrintF (                                ! Lookaside list start
; P  0517  3                      Format,
; P  0518  3                      .Exe$GL_SplitAdr,
; P  0519  3                      $Ascic ('Start of IRP Look-aside list')
;    0520  3                      );
;    0521  3
;    0522  2              End;
;    0523  2
;    0524  2          If .MemoryFlags [Sm_V_Map]                          !                                   [01]
;    0525  2           Or .MemoryFlags [Sm_V_Data]
;    0526  2          Then
; P  0527  2              $Ds_PrintF (                                    ! Top of run-time DS
; P  0528  2                  Format5,
; P  0529  2                  .Ds$GA_LastAdr,
; P  0530  2                  (.Ds$GA_LastAdr + 1023)/1024,               ! # of Kb (next higher)             [01]
; P  0531  2                  $Ascic ('Top of run-time '),
; P  0532  2                  T_DS
;    0533  2                  );
;    0534  2
;    0535  2          If .MemoryFlags [Sm_V_Buffer]                       ! If /Buffer or /All                [01]
;    0536  3           And (.Ds$GL_MemSize Gtr .Ds$GA_LastAdr)            !                                   [01]
;    0537  2          Then                                                !                                   [01]
; P  0538  2              $Ds_PrintF (                                    ! Available buffer space            [01]
; P  0539  2                  Format5,                                    !                                   [01]
; P  0540  2                  .Ds$GL_MemSize - .Ds$GA_LastAdr,            ! .. amount available               [01]
; P  0541  2                  (.Ds$GL_MemSize - .Ds$GA_LastAdr)/1024,     ! .. same in Kb                     [01]
; P  0542  2                  T_BufferSpace,                              !                                   [01]
; P  0543  2                  $Ascic ('above Supervisor')                 !                                   [01]
;    0544  2                  );                                          !                                   [01]
;    0545  2
;    0546  2          If .MemoryFlags [Sm_V_Buffer] Then                  ! If /Buffer or /All                [03]
;    0547  3              Begin                                           !                                   [03]
; P  0548  3              $DS_PrintF (                                    ! Print the amount of buffer space being used in  [03]
; P  0549  3                  Format,                                     ! . . . P0 space.                   [03]
; P  0550  3                  .DS$GL_BufCnt [0],                          ! . . . This is the number of pages [03]
; P  0551  3                  T_P0_BufCnt                                 !                                   [03]
;    0552  3                  );                                          !                                   [03]
; P  0553  3              $DS_PrintF (                                    ! Print the amount of buffer space being used in  [03]
; P  0554  3                  Format,                                     ! . . . P1 space.                   [03]
; P  0555  3                  .DS$GL_BufCnt [1],                          ! . . . This is the number of pages [03]
; P  0556  3                  T_P1_BufCnt                                 !                                   [03]
;    0557  3                  );                                          !                                   [03]
; P  0558  3              $DS_PrintF (                                    ! Print the amount of buffer space being used in  [03]
; P  0559  3                  Format,                                     ! . . . S0 space.                   [03]
; P  0560  3                  .DS$GL_BufCnt [2],                          ! . . . This is the number of pages [03]
; P  0561  3                  T_S0_BufCnt                                 !                                   [03]
;    0562  3                  );                                          !                                   [03]
;    0563  2              End;                                            !                                   [03]
;    0564  2
;    0565  2          If .dsa$v_debug                                     !If debugger is loaded, print its limits.  [02]
;    0566  2          then
;    0567  3              begin
; P  0568  3              $ds_printf (format,
; P  0569  3                          .debug_lo,
;    0570  3                          $ascic ('Bottom of debugger'));
```

```
;    0571  3              if not .online
;    0572  3              then
; P  0573  3                  $ds_printf (format6,
;    0574  3                              $ascic (' (includes allocation for symbol tables)'));
; P  0575  3              $ds_printf (format,
; P  0576  3                          .debug_hi,
;    0577  3                          $ascic ('Top of debugger'));
;    0578  2              end;
;    0579  2
;    0580  2          If Not .OnLine
;    0581  2          Then
; P  0582  2              $Ds_PrintF (                                ! Top of physical memory
; P  0583  2                  Format4,
; P  0584  2                  .Ds$GL_MemSize,
; P  0585  2                  .Ds$GL_MemSize/1024,                    ! Kb of memory                    [01]
; P  0586  2                  $Ascic ('Top of physical memory')
;    0587  2                  );
;    0588  2          Ds$GB_TypeCode = 0;                             ! Clear sticky typecode
;    0589  1          End;


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

64 20 66 6F 20 67 6E 69 6E 6E 69 67 65 42 1E  001C7 P.AAY:  .ASCII  <30>\Beginning of diagnostic region\      ;
6F 69 67 65 72 20 63 69 74 73 6F 6E 67 61 69  001D6                                                        ;
                                              6E 001E5                                                      ;
20 64 65 64 61 6F 6C 20 66 6F 20 64 6E 45 18  001E6 P.AAZ:  .ASCII  <24>\End of loaded diagnostic\         ;
                  63 69 74 73 6F 6E 67 61 69  64 001F5                                                      ;
73 69 76 72 65 70 75 53 20 77 6F 6C 65 62 10  001FF P.ABA:  .ASCII  <16>\below Supervisor\                  ;
                                              72 6F 0020E                                                   ;
66 66 75 62 20 74 78 65 74 50 20 54 50 41 10  00210 P.ABB:  .ASCII  <16>\APT Ptext buffer\                 ;
                                              72 65 0021F                                                   ;
         78 6F 62 6C 69 61 4D 20 54 50 41 0B  00221 P.ABC:  .ASCII  <11>\APT Mailbox\                      ;
   20 66 6F 20 67 6E 69 6E 6E 69 67 65 42 0D  0022D P.ABD:  .ASCII  <13>\Beginning of \                    ;
69 76 72 65 70 75 53 20 66 6F 20 64 6E 45 1C  0023B P.ABE:  .ASCII  <28>\End of Supervisor load image\     ;
   65 67 61 6D 69 20 64 61 6F 6C 20 72 6F 73  0024A                                                        ;
                                 44 48 50 03  00258 P.ABF:  .ASCII  <3>\PHD\                             ;
      42 43 50 20 65 72 61 77 64 72 61 48 0C  0025C P.ABG:  .ASCII  <12>\Hardware PCB\                    ;
                                 42 49 4A 03  00269 P.ABH:  .ASCII  <3>\JIB\                             ;
                                 42 52 41 03  0026D P.ABI:  .ASCII  <3>\ARB\                             ;
      42 43 50 20 65 72 61 77 74 66 6F 53 0C  00271 P.ABJ:  .ASCII  <12>\Software PCB\                    ;
                                 42 43 53 03  0027E P.ABK:  .ASCII  <3>\SCB\                             ;
49 52 54 4C 55 20 66 6F 20 74 72 61 74 53 1C  00282 P.ABL:  .ASCII  <28>\Start of ULTRIX buffer space\    ;
   65 63 61 70 73 20 72 65 66 66 75 62 20 58  00291                                                        ;
72 65 70 75 53 20 66 6F 20 74 72 61 74 53 1F  0029F P.ABM:  .ASCII  <31>\Start of Supervisor memory pool\ ;
6F 70 20 79 72 6F 6D 65 6D 20 72 6F 73 69 76  002AE                                                        ;
                                              6C 6F 002BD                                                   ;
4C 20 50 52 49 20 66 6F 20 74 72 61 74 53 1C  002BF P.ABN:  .ASCII  <28>\Start of IRP Look-aside list\   ;
   74 73 69 6C 20 65 64 69 73 61 2D 6B 6F 6F  002CE                                                        ;
6D 69 74 2D 6E 75 72 20 66 6F 20 70 6F 54 10  002DC P.ABO:  .ASCII  <16>\Top of run-time \               ;
                                              20 65 002EB                                                   ;
73 69 76 72 65 70 75 53 20 65 76 6F 62 61 10  002ED P.ABP:  .ASCII  <16>\above Supervisor\               ;
                                              72 6F 002FC                                                   ;
75 62 65 64 20 66 6F 20 6D 6F 74 74 6F 42 12  002FE P.ABQ:  .ASCII  <18>\Bottom of debugger\             ;
                                 72 65 67 67 0030D                                                          ;
6C 6C 61 20 73 65 64 75 6C 63 6E 69 28 20 28  00311 P.ABR:  .ASCII  \( (includes allocation for symbol tables\  ;
```

```
6D 79 73 20 72 6F 66 20 6E 6F 69 74 61 63 6F  00320                                                    ;
                73 65 6C 62 61 74 20 6C 6F 62  0032F                                                    ;
                                            29  00339          .ASCII  \)\                                  ;
65 67 67 75 62 65 64 20 66 6F 20 70 6F 54 0F  0033A P.ABS:   .ASCII  <15>\Top of debugger\                 ;
                                            72  00349                                                    ;
61 63 69 73 79 68 70 20 66 6F 20 70 6F 54 16  0034A P.ABT:   .ASCII  <22>\Top of physical memory\         ;
                      79 72 6F 6D 65 6D 20 6C  00359                                                    ;

                                                             .EXTRN  DS$PRINTF, CMK$SGIPR

                                                             .PSECT  CODE,NOWRT,  SHR,2

                                          0FFC 00000 SHOWMEMORY:
                                                             .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11   ; 0231
                                 5B 00000000G  EF  9E 00002  MOVAB   DS$GL_MEMSIZE, R11                     ;
                                 5A 00000000G  EF  9E 00009  MOVAB   DS$GA_LASTADR, R10                     ;
                                 59 00000000G  EF  9E 00010  MOVAB   CMK$SGIPR, R9                          ;
                                 58 00000000'  EF  9E 00017  MOVAB   DS$GL_IPR_VALUE, R8                    ;
                                 57 00000000G  9F  9E 0001E  MOVAB   @#DS$PRINTF, R7                        ;
                                 56 00000000'  EF  9E 00025  MOVAB   FORMAT, R6                             ;
      50       FC   A8          04            00  EF 0002C  EXTZV   #0, #4, SHOWMEMORYFLAGS, R0           ; 0243
                                 52            50  90 00032  MOVB    R0, MEMORYFLAGS                        ;
                                               03  12 00035  BNEQ    1$                                     ; 0245
                                 52            01  88 00037  BISB2   #1, MEMORYFLAGS                        ; 0247
                      03         52            03  E1 0003A 1$:  BBC     #3, MEMORYFLAGS, 2$                    ; 0249
                                 52            07  88 0003E  BISB2   #7, MEMORYFLAGS                        ; 0252
      50 0000FE03   9F           01            04  EF 00041 2$:  EXTZV   #4, #1, @#^X0000FE03, R0              ; 0257
                                 54            50  90 0004A  MOVB    R0, ONLINE                             ;
                      00000000G  EF            16  90 0004D  MOVB    #22, DS$GB_TYPECODE                    ; 0258
                                 35            52  E9 00054  BLBC    MEMORYFLAGS, 3$                        ; 0260
                                        009F   C6  9F 00057  PUSHAB  T_RESERVED                             ; 0266
                                               7E  D4 0005B  CLRL    -(SP)                                  ;
                                               56  DD 0005D  PUSHL   R6                                     ;
                                 67            03  FB 0005F  CALLS   #3, DS$PRINTF                          ;
                                 27            52  E9 00062  BLBC    MEMORYFLAGS, 3$                        ; 0268
                                        01C7   C6  9F 00065  PUSHAB  P.AAY                                  ; 0274
                                    00000000G  EF  9F 00069  PUSHAB  DS$A_PRGBGN                            ;
                                               56  DD 0006F  PUSHL   R6                                     ;
                                 67            03  FB 00071  CALLS   #3, DS$PRINTF                          ;
                                 15            52  E9 00074  BLBC    MEMORYFLAGS, 3$                        ; 0276
                                            FF61  30 00077  BSBW    DSR$CHECKLOAD                          ;
                                 0F            50  E9 0007A  BLBC    R0, 3$                                 ;
                                        01E6   C6  9F 0007D  PUSHAB  P.AAZ                                  ; 0282
                                    00000214   9F  DD 00081  PUSHL   @#^X00000214                           ;
                                               56  DD 00087  PUSHL   R6                                     ;
                                 67            03  FB 00089  CALLS   #3, DS$PRINTF                          ;
                      2E         52            01  E1 0008C 3$:  BBC     #1, MEMORYFLAGS, 4$                    ; 0284
                                            FF48  30 00090  BSBW    DSR$CHECKLOAD                          ; 0285
                                 28            50  E9 00093  BLBC    R0, 4$                                 ;
                                        01FF   C6  9F 00096  PUSHAB  P.ABA                                  ; 0293
                                        01AE   C6  9F 0009A  PUSHAB  T_BUFFERSPACE                          ;
                      50 00000214  9F 0000FA00  8F  C3 0009E  SUBL3   #64000, @#^X00000214, R0              ;
                      51           50 00000400  8F  C7 000AA  DIVL3   #1024, R0, R1                          ;
                                 7E            51  CE 000B2  MNEGL   R1, -(SP)                              ;
                                 7E            50  CE 000B5  MNEGL   R0, -(SP)                              ;
                                          75   A6  9F 000B8  PUSHAB  FORMAT5                                ;
                                 67            05  FB 000BB  CALLS   #5, DS$PRINTF                          ;
```

```
                                  53  D4 000BE 4$:    CLRL    R3                                  ; 0301
                      05          52  93 000C0        BITB    MEMORYFLAGS, #5                      ;
                                  10  13 000C3        BEQL    5$                                  ;
                                  53  D6 000C5        INCL    R3                                  ;
                            0210  C6  9F 000C7        PUSHAB  P.ABB                               ;
                      7E    FA00  8F  3C 000CB        MOVZWL  #64000, -(SP)                       ;
                                  56  DD 000D0        PUSHL   R6                                  ;
                      67          03  FB 000D2        CALLS   #3, DS$PRINTF                       ;
                      0E          53  E9 000D5 5$:    BLBC    R3, 6$                              ; 0304
                            0221  C6  9F 000D8        PUSHAB  P.ABC                               ; 0310
                      7E    FE00  8F  3C 000DC        MOVZWL  #65024, -(SP)                       ;
                                  56  DD 000E1        PUSHL   R6                                  ;
                      67          03  FB 000E3        CALLS   #3, DS$PRINTF                       ;
                      26          52  E9 000E6 6$:    BLBC    MEMORYFLAGS, 7$                     ; 0312
                            00AD  C6  9F 000E9        PUSHAB  T_DS                                ; 0319
                            022D  C6  9F 000ED        PUSHAB  P.ABD                               ;
                        00010000  8F  DD 000F1        PUSHL   #65536                              ;
                              1C  A6  9F 000F7        PUSHAB  FORMAT2                             ;
                      67          04  FB 000FA        CALLS   #4, DS$PRINTF                       ;
                      0F          52  E9 000FD        BLBC    MEMORYFLAGS, 7$                     ; 0321
                            023B  C6  9F 00100        PUSHAB  P.ABE                               ; 0327
                        00000000G EF  9F 00104        PUSHAB  AX_BUFF                             ;
                                  56  DD 0010A        PUSHL   R6                                  ;
                      67          03  FB 0010C        CALLS   #3, DS$PRINTF                       ;
          03          52          02  E0 0010F 7$:    BBS     #2, MEMORYFLAGS, 9$                 ; 0329
                                02CF  31 00113 8$:    BRW     32$                                 ;
                      FA          54  E8 00116 9$:    BLBS    ONLINE, 8$                          ;
                            0258  C6  9F 00119        PUSHAB  P.ABF                               ; 0336
                        00000000G EF  9F 0011D        PUSHAB  PHD_BASE                            ;
                                  56  DD 00123        PUSHL   R6                                  ;
                      67          03  FB 00125        CALLS   #3, DS$PRINTF                       ;
                            025C  C6  9F 00128        PUSHAB  P.ABG                               ; 0341
                        00000000G EF  9F 0012C        PUSHAB  PCB_BASE                            ;
                                  56  DD 00132        PUSHL   R6                                  ;
                      67          03  FB 00134        CALLS   #3, DS$PRINTF                       ;
                            0269  C6  9F 00137        PUSHAB  P.ABH                               ; 0346
                        00000000G EF  9F 0013B        PUSHAB  DS$AX_JIB                           ;
                                  56  DD 00141        PUSHL   R6                                  ;
                      67          03  FB 00143        CALLS   #3, DS$PRINTF                       ;
                            026D  C6  9F 00146        PUSHAB  P.ABI                               ; 0351
                        00000000G EF  9F 0014A        PUSHAB  DS$AX_ARB                           ;
                                  56  DD 00150        PUSHL   R6                                  ;
                      67          03  FB 00152        CALLS   #3, DS$PRINTF                       ;
                            0271  C6  9F 00155        PUSHAB  P.ABJ                               ; 0356
                        00000000G EF  9F 00159        PUSHAB  DS$AX_SOFTPCB                       ;
                                  56  DD 0015F        PUSHL   R6                                  ;
                      67          03  FB 00161        CALLS   #3, DS$PRINTF                       ;
                            027E  C6  9F 00164        PUSHAB  P.ABK                               ; 0361
                        00000000G EF  9F 00168        PUSHAB  SCB_BASE                            ;
                                  56  DD 0016E        PUSHL   R6                                  ;
                      67          03  FB 00170        CALLS   #3, DS$PRINTF                       ;
                            009F  C6  9F 00173        PUSHAB  T_RESERVED                          ; 0366
                        00000000G EF  9F 00177        PUSHAB  STACK_BASE                          ;
                                  56  DD 0017D        PUSHL   R6                                  ;
                      67          03  FB 0017F        CALLS   #3, DS$PRINTF                       ;
                                  53  D4 00182        CLRL    R3                                  ; 0367
                      55          01  D0 00184        MOVL    #1, R5                              ; 0368
```

```
                                50  DC 00187          MOVPSL  TEMP_PSL                          ; 0369
      0A          50            1A  E1 00189          BBC     #26, TEMP_PSL, 10$                ;
                  5E            04  C2 0018D          SUBL2   #4, SP                            ;
                  6E            50  D0 00190          MOVL    TEMP_PSL, (SP)                    ;
                                69  16 00193          JSB     CMK$SGIPR                         ;
                                08  11 00195          BRB     11$                               ;
                  50            51  9A 00197 10$:     MOVZBL  TEMP_CHMK_CODE, R0                ;
                  60            0A  D0 0019A          MOVL    #10, (R0)                         ;
                                51  BC 0019D          CHMK    TEMP_CHMK_CODE                    ;
                        00FA    C6  9F 0019F 11$:     PUSHAB  T_SP                              ; 0375
                        00C3    C6  9F 001A3          PUSHAB  T_KERNEL                          ;
                                68  DD 001A7          PUSHL   DS$GL_IPR_VALUE                   ;
                          1C    A6  9F 001A9          PUSHAB  FORMAT2                           ;
                  67            04  FB 001AC          CALLS   #4, DS$PRINTF                     ;
                        00EE    C6  9F 001AF          PUSHAB  T_STACK                           ; 0381
                        00C3    C6  9F 001B3          PUSHAB  T_KERNEL                          ;
                    00000000G   EF  9F 001B7          PUSHAB  KSTKPTR                           ;
                          3B    A6  9F 001BD          PUSHAB  FORMAT3                           ;
                  67            04  FB 001C0          CALLS   #4, DS$PRINTF                     ;
                  53            04  D0 001C3          MOVL    #4, R3                            ; 0382
                  55            01  D0 001C6          MOVL    #1, R5                            ; 0383
                                50  DC 001C9          MOVPSL  TEMP_PSL                          ; 0384
      0A          50            1A  E1 001CB          BBC     #26, TEMP_PSL, 12$                ;
                  5E            04  C2 001CF          SUBL2   #4, SP                            ;
                  6E            50  D0 001D2          MOVL    TEMP_PSL, (SP)                    ;
                                69  16 001D5          JSB     CMK$SGIPR                         ;
                                08  11 001D7          BRB     13$                               ;
                  50            51  9A 001D9 12$:     MOVZBL  TEMP_CHMK_CODE, R0                ;
                  60            0A  D0 001DC          MOVL    #10, (R0)                         ;
                                51  BC 001DF          CHMK    TEMP_CHMK_CODE                    ;
                        00FA    C6  9F 001E1 13$:     PUSHAB  T_SP                              ; 0390
                        00CA    C6  9F 001E5          PUSHAB  T_INTERRUPT                       ;
                                68  DD 001E9          PUSHL   DS$GL_IPR_VALUE                   ;
                          1C    A6  9F 001EB          PUSHAB  FORMAT2                           ;
                  67            04  FB 001EE          CALLS   #4, DS$PRINTF                     ;
                        00EE    C6  9F 001F1          PUSHAB  T_STACK                           ; 0396
                        00CA    C6  9F 001F5          PUSHAB  T_INTERRUPT                       ;
                    00000000G   EF  9F 001F9          PUSHAB  ISTKPTR                           ;
                          3B    A6  9F 001FF          PUSHAB  FORMAT3                           ;
                  67            04  FB 00202          CALLS   #4, DS$PRINTF                     ;
                  53            01  D0 00205          MOVL    #1, R3                            ; 0397
                  55            01  D0 00208          MOVL    #1, R5                            ; 0398
                                50  DC 0020B          MOVPSL  TEMP_PSL                          ; 0399
      0A          50            1A  E1 0020D          BBC     #26, TEMP_PSL, 14$                ;
                  5E            04  C2 00211          SUBL2   #4, SP                            ;
                  6E            50  D0 00214          MOVL    TEMP_PSL, (SP)                    ;
                                69  16 00217          JSB     CMK$SGIPR                         ;
                                08  11 00219          BRB     15$                               ;
                  50            51  9A 0021B 14$:     MOVZBL  TEMP_CHMK_CODE, R0                ;
                  60            0A  D0 0021E          MOVL    #10, (R0)                         ;
                                51  BC 00221          CHMK    TEMP_CHMK_CODE                    ;
                        00FA    C6  9F 00223 15$:     PUSHAB  T_SP                              ; 0405
                        00D4    C6  9F 00227          PUSHAB  T_EXECUTIVE                       ;
                                68  DD 0022B          PUSHL   DS$GL_IPR_VALUE                   ;
                          1C    A6  9F 0022D          PUSHAB  FORMAT2                           ;
                  67            04  FB 00230          CALLS   #4, DS$PRINTF                     ;
                        00EE    C6  9F 00233          PUSHAB  T_STACK                           ; 0411
```

```
                    00D4   C6  9F 00237          PUSHAB  T_EXECUTIVE                    ;
                00000000G  EF  9F 0023B          PUSHAB  ESTKPTR                        ;
                      3B   A6  9F 00241          PUSHAB  FORMAT3                        ;
              67           04  FB 00244          CALLS   #4, DS$PRINTF                  ;
              53           02  D0 00247          MOVL    #2, R3                         ; 0412
              55           01  D0 0024A          MOVL    #1, R5                         ; 0413
                           50  DC 0024D          MOVPSL  TEMP_PSL                       ; 0414
      0A      50           1A  E1 0024F          BBC     #26, TEMP_PSL, 16$             ;
              5E           04  C2 00253          SUBL2   #4, SP                         ;
              6E           50  D0 00256          MOVL    TEMP_PSL, (SP)                 ;
                           69  16 00259          JSB     CMK$SGIPR                      ;
                           08  11 0025B          BRB     17$                            ;
              50           51  9A 0025D 16$:     MOVZBL  TEMP_CHMK_CODE, R0             ;
              60           0A  D0 00260          MOVL    #10, (R0)                      ;
                           51  BC 00263          CHMK    TEMP_CHMK_CODE                 ;
                    00FA   C6  9F 00265 17$:     PUSHAB  T_SP                           ; 0420
                    00DE   C6  9F 00269          PUSHAB  T_SUPERVISOR                   ;
                           68  DD 0026D          PUSHL   DS$GL_IPR_VALUE                ;
                      1C   A6  9F 0026F          PUSHAB  FORMAT2                        ;
              67           04  FB 00272          CALLS   #4, DS$PRINTF                  ;
                    00EE   C6  9F 00275          PUSHAB  T_STACK                        ; 0426
                    00DE   C6  9F 00279          PUSHAB  T_SUPERVISOR                   ;
                00000000G  EF  9F 0027D          PUSHAB  SSTKPTR                        ;
                      3B   A6  9F 00283          PUSHAB  FORMAT3                        ;
              67           04  FB 00286          CALLS   #4, DS$PRINTF                  ;
              53           03  D0 00289          MOVL    #3, R3                         ; 0427
              55           01  D0 0028C          MOVL    #1, R5                         ; 0428
                           50  DC 0028F          MOVPSL  TEMP_PSL                       ; 0429
      0A      50           1A  E1 00291          BBC     #26, TEMP_PSL, 18$             ;
              5E           04  C2 00295          SUBL2   #4, SP                         ;
              6E           50  D0 00298          MOVL    TEMP_PSL, (SP)                 ;
                           69  16 0029B          JSB     CMK$SGIPR                      ;
                           08  11 0029D          BRB     19$                            ;
              50           51  9A 0029F 18$:     MOVZBL  TEMP_CHMK_CODE, R0             ;
              60           0A  D0 002A2          MOVL    #10, (R0)                      ;
                           51  BC 002A5          CHMK    TEMP_CHMK_CODE                 ;
                    00FA   C6  9F 002A7 19$:     PUSHAB  T_SP                           ; 0435
                    00E9   C6  9F 002AB          PUSHAB  T_USER                         ;
                           68  DD 002AF          PUSHL   DS$GL_IPR_VALUE                ;
                      1C   A6  9F 002B1          PUSHAB  FORMAT2                        ;
              67           04  FB 002B4          CALLS   #4, DS$PRINTF                  ;
                    00EE   C6  9F 002B7          PUSHAB  T_STACK                        ; 0441
                    00E9   C6  9F 002BB          PUSHAB  T_USER                         ;
                00000000G  EF  9F 002BF          PUSHAB  USTKPTR                        ;
                      3B   A6  9F 002C5          PUSHAB  FORMAT3                        ;
              67           04  FB 002C8          CALLS   #4, DS$PRINTF                  ;
              53           08  D0 002CB          MOVL    #8, R3                         ; 0442
              55           01  D0 002CE          MOVL    #1, R5                         ; 0443
                           50  DC 002D1          MOVPSL  TEMP_PSL                       ; 0444
      0A      50           1A  E1 002D3          BBC     #26, TEMP_PSL, 20$             ;
              5E           04  C2 002D7          SUBL2   #4, SP                         ;
              6E           50  D0 002DA          MOVL    TEMP_PSL, (SP)                 ;
                           69  16 002DD          JSB     CMK$SGIPR                      ;
                           08  11 002DF          BRB     21$                            ;
              50           51  9A 002E1 20$:     MOVZBL  TEMP_CHMK_CODE, R0             ;
              60           0A  D0 002E4          MOVL    #10, (R0)                      ;
                           51  BC 002E7          CHMK    TEMP_CHMK_CODE                 ;
```

```
                                 010B   C6  9F 002E9 21$:    PUSHAB  T_BASE                                    ; 0450
                                 00FE   C6  9F 002ED         PUSHAB  T_P0                                      ;
      7E          68       1E          00  EF 002F1         EXTZV   #0, #30, DS$GL_IPR_VALUE, -(SP)           ;
                                   1C    A6  9F 002F6         PUSHAB  FORMAT2                                   ;
                            67           04  FB 002F9         CALLS   #4, DS$PRINTF                             ;
                            53           09  D0 002FC         MOVL    #9, R3                                    ; 0451
                            55           01  D0 002FF         MOVL    #1, R5                                    ; 0452
                                         50  DC 00302         MOVPSL  TEMP_PSL                                  ; 0453
                     0A     50           1A  E1 00304         BBC     #26, TEMP_PSL, 22$                        ;
                            5E           04  C2 00308         SUBL2   #4, SP                                    ;
                            6E           50  D0 0030B         MOVL    TEMP_PSL, (SP)                            ;
                                         69  16 0030E         JSB     CMK$SGIPR                                 ;
                                         08  11 00310         BRB     23$                                       ;
                            50           51  9A 00312 22$:    MOVZBL  TEMP_CHMK_CODE, R0                        ;
                            60           0A  D0 00315         MOVL    #10, (R0)                                 ;
                                         51  BC 00318         CHMK    TEMP_CHMK_CODE                            ;
                                 011C    C6  9F 0031A 23$:    PUSHAB  T_LENGTH                                  ; 0459
                                 00FE    C6  9F 0031E         PUSHAB  T_P0                                      ;
                                         68  DD 00322         PUSHL   DS$GL_IPR_VALUE                           ;
                                   3B    A6  9F 00324         PUSHAB  FORMAT3                                   ;
                            67           04  FB 00327         CALLS   #4, DS$PRINTF                             ;
                            53           0A  D0 0032A         MOVL    #10, R3                                   ; 0460
                            55           01  D0 0032D         MOVL    #1, R5                                    ; 0461
                                         50  DC 00330         MOVPSL  TEMP_PSL                                  ; 0462
                     0A     50           1A  E1 00332         BBC     #26, TEMP_PSL, 24$                        ;
                            5E           04  C2 00336         SUBL2   #4, SP                                    ;
                            6E           50  D0 00339         MOVL    TEMP_PSL, (SP)                            ;
                                         69  16 0033C         JSB     CMK$SGIPR                                 ;
                                         08  11 0033E         BRB     25$                                       ;
                            50           51  9A 00340 24$:    MOVZBL  TEMP_CHMK_CODE, R0                        ;
                            60           0A  D0 00343         MOVL    #10, (R0)                                 ;
                                         51  BC 00346         CHMK    TEMP_CHMK_CODE                            ;
                                 010B    C6  9F 00348 25$:    PUSHAB  T_BASE                                    ; 0468
                                 0101    C6  9F 0034C         PUSHAB  T_P1                                      ;
      7E          68       1E          00  EF 00350         EXTZV   #0, #30, DS$GL_IPR_VALUE, -(SP)           ;
                                   1C    A6  9F 00355         PUSHAB  FORMAT2                                   ;
                            67           04  FB 00358         CALLS   #4, DS$PRINTF                             ;
                            53           0B  D0 0035B         MOVL    #11, R3                                   ; 0469
                            55           01  D0 0035E         MOVL    #1, R5                                    ; 0470
                                         50  DC 00361         MOVPSL  TEMP_PSL                                  ; 0471
                     0A     50           1A  E1 00363         BBC     #26, TEMP_PSL, 26$                        ;
                            5E           04  C2 00367         SUBL2   #4, SP                                    ;
                            6E           50  D0 0036A         MOVL    TEMP_PSL, (SP)                            ;
                                         69  16 0036D         JSB     CMK$SGIPR                                 ;
                                         08  11 0036F         BRB     27$                                       ;
                            50           51  9A 00371 26$:    MOVZBL  TEMP_CHMK_CODE, R0                        ;
                            60           0A  D0 00374         MOVL    #10, (R0)                                 ;
                                         51  BC 00377         CHMK    TEMP_CHMK_CODE                            ;
                                 011C    C6  9F 00379 27$:    PUSHAB  T_LENGTH                                  ; 0477
                                 0101    C6  9F 0037D         PUSHAB  T_P1                                      ;
                                         68  DD 00381         PUSHL   DS$GL_IPR_VALUE                           ;
                                   3B    A6  9F 00383         PUSHAB  FORMAT3                                   ;
                            67           04  FB 00386         CALLS   #4, DS$PRINTF                             ;
                            53           0C  D0 00389         MOVL    #12, R3                                   ; 0478
                            55           01  D0 0038C         MOVL    #1, R5                                    ; 0479
                                         50  DC 0038F         MOVPSL  TEMP_PSL                                  ; 0480
                     0A     50           1A  E1 00391         BBC     #26, TEMP_PSL, 28$                        ;
```

```
                 5E          04  C2 00395        SUBL2   #4, SP                                  ;
                 6E          50  D0 00398        MOVL    TEMP_PSL, (SP)                          ;
                             69  16 0039B        JSB     CMK$SGIPR                               ;
                             08  11 0039D        BRB     29$                                     ;
                 50          51  9A 0039F 28$:   MOVZBL  TEMP_CHMK_CODE, R0                      ;
                 60          0A  D0 003A2        MOVL    #10, (R0)                               ;
                             51  BC 003A5        CHMK    TEMP_CHMK_CODE                          ;
                      010B   C6  9F 003A7 29$:   PUSHAB  T_BASE                                  ; 0486
                      0104   C6  9F 003AB        PUSHAB  T_SYSTEM                                ;
                             68  DD 003AF        PUSHL   DS$GL_IPR_VALUE                         ;
                        1C   A6  9F 003B1        PUSHAB  FORMAT2                                 ;
                 67          04  FB 003B4        CALLS   #4, DS$PRINTF                           ;
                 53          0D  D0 003B7        MOVL    #13, R3                                 ; 0487
                 55          01  D0 003BA        MOVL    #1, R5                                  ; 0488
                             50  DC 003BD        MOVPSL  TEMP_PSL                                ; 0489
        0A       50          1A  E1 003BF        BBC     #26, TEMP_PSL, 30$                      ;
                 5E          04  C2 003C3        SUBL2   #4, SP                                  ;
                 6E          50  D0 003C6        MOVL    TEMP_PSL, (SP)                          ;
                             69  16 003C9        JSB     CMK$SGIPR                               ;
                             08  11 003CB        BRB     31$                                     ;
                 50          51  9A 003CD 30$:   MOVZBL  TEMP_CHMK_CODE, R0                      ;
                 60          0A  D0 003D0        MOVL    #10, (R0)                               ;
                             51  BC 003D3        CHMK    TEMP_CHMK_CODE                          ;
                      011C   C6  9F 003D5 31$:   PUSHAB  T_LENGTH                                ; 0495
                      0104   C6  9F 003D9        PUSHAB  T_SYSTEM                                ;
                             68  DD 003DD        PUSHL   DS$GL_IPR_VALUE                         ;
                        3B   A6  9F 003DF        PUSHAB  FORMAT3                                 ;
                 67          04  FB 003E2        CALLS   #4, DS$PRINTF                           ;
                 0F          52  E9 003E5 32$:   BLBC    MEMORYFLAGS, 33$                        ; 0497
                      0282   C6  9F 003E8        PUSHAB  P.ABL                                   ; 0503
                 00000000G   EF  DD 003EC        PUSHL   ULTRIX_IOB_ADD                          ;
                             56  DD 003F2        PUSHL   R6                                      ;
                 67          03  FB 003F4        CALLS   #3, DS$PRINTF                           ;
                             53  D4 003F7 33$:   CLRL    R3                                      ; 0507
                 05          52  93 003F9        BITB    MEMORYFLAGS, #5                         ;
                             2D  13 003FC        BEQL    34$                                     ;
                             53  D6 003FE        INCL    R3                                      ;
                      029F   C6  9F 00400        PUSHAB  P.ABM                                   ; 0512
                 00000000G   EF  DD 00404        PUSHL   DS$GL_LOOKASIDE                         ;
                             56  DD 0040A        PUSHL   R6                                      ;
                 67          03  FB 0040C        CALLS   #3, DS$PRINTF                           ;
     00000000G   EF 00000000G EF  D1 0040F       CMPL    DS$GL_LOOKASIDE, EXE$GL_SPLITADR        ; 0514
                             0F  13 0041A        BEQL    34$                                     ;
                      02BF   C6  9F 0041C        PUSHAB  P.ABN                                   ; 0520
                 00000000G   EF  DD 00420        PUSHL   EXE$GL_SPLITADR                         ;
                             56  DD 00426        PUSHL   R6                                      ;
                 67          03  FB 00428        CALLS   #3, DS$PRINTF                           ;
                 20          53  E9 0042B 34$:   BLBC    R3, 35$                                 ; 0525
                      00AD   C6  9F 0042E        PUSHAB  T_DS                                    ; 0533
                      02DC   C6  9F 00432        PUSHAB  P.ABO                                   ;
        50       6A 000003FF 8F  C1 00436        ADDL3   #1023, DS$GA_LASTADR, R0                ;
        7E       50 00000400 8F  C7 0043E        DIVL3   #1024, R0, -(SP)                        ;
                             6A  DD 00446        PUSHL   DS$GA_LASTADR                           ;
                        75   A6  9F 00448        PUSHAB  FORMAT5                                 ;
                 67          05  FB 0044B        CALLS   #5, DS$PRINTF                           ;
        52       52          01  E1 0044E 35$:   BBC     #1, MEMORYFLAGS, 37$                    ; 0535
                 6A          6B  D1 00452        CMPL    DS$GL_MEMSIZE, DS$GA_LASTADR            ; 0536
```

```
                                  1C  15 00455          BLEQ    36$                                          ;
                          02ED    C6  9F 00457          PUSHAB  P.ABP                                        ; 0544
                          01AE    C6  9F 0045B          PUSHAB  T_BUFFERSPACE                                ;
        50          6B            6A  C3 0045F          SUBL3   DS$GA_LASTADR, DS$GL_MEMSIZE, R0             ;
        7E          50 00000400   8F  C7 00463          DIVL3   #1024, R0, -(SP)                             ;
                                  50  DD 0046B          PUSHL   R0                                           ;
                             75   A6  9F 0046D          PUSHAB  FORMAT5                                      ;
                    67            05  FB 00470          CALLS   #5, DS$PRINTF                                ;
        2D          52            01  E1 00473 36$:     BBC     #1, MEMORYFLAGS, 37$                         ; 0546
                          013B    C6  9F 00477          PUSHAB  T_P0_BUFCNT                                  ; 0552
                       00000000G  EF  DD 0047B          PUSHL   DS$GL_BUFCNT                                 ;
                                  56  DD 00481          PUSHL   R6                                           ;
                    67            03  FB 00483          CALLS   #3, DS$PRINTF                                ;
                          0160    C6  9F 00486          PUSHAB  T_P1_BUFCNT                                  ; 0557
                       00000000G  EF  DD 0048A          PUSHL   DS$GL_BUFCNT+4                               ;
                                  56  DD 00490          PUSHL   R6                                           ;
                    67            03  FB 00492          CALLS   #3, DS$PRINTF                                ;
                          0185    C6  9F 00495          PUSHAB  T_S0_BUFCNT                                  ; 0562
                       00000000G  EF  DD 00499          PUSHL   DS$GL_BUFCNT+8                               ;
                                  56  DD 0049F          PUSHL   R6                                           ;
                    67            03  FB 004A1          CALLS   #3, DS$PRINTF                                ;
        2C 0000FE03 9F            02  E1 004A4 37$:     BBC     #2, @#^X0000FE03, 39$                        ; 0565
                          02FE    C6  9F 004AC          PUSHAB  P.ABQ                                        ; 0570
                       00000000G  EF  DD 004B0          PUSHL   DEBUG_LO                                     ;
                                  56  DD 004B6          PUSHL   R6                                           ;
                    67            03  FB 004B8          CALLS   #3, DS$PRINTF                                ;
                    0B            54  E8 004BB          BLBS    ONLINE, 38$                                  ; 0571
                          0311    C6  9F 004BE          PUSHAB  P.ABR                                        ; 0574
                          0092    C6  9F 004C2          PUSHAB  FORMAT6                                      ;
                    67            02  FB 004C6          CALLS   #2, DS$PRINTF                                ;
                          033A    C6  9F 004C9 38$:     PUSHAB  P.ABS                                        ; 0577
                       00000000G  EF  DD 004CD          PUSHL   DEBUG_HI                                     ;
                                  56  DD 004D3          PUSHL   R6                                           ;
                    67            03  FB 004D5          CALLS   #3, DS$PRINTF                                ;
                    14            54  E8 004D8 39$:     BLBS    ONLINE, 40$                                  ; 0580
                          034A    C6  9F 004DB          PUSHAB  P.ABT                                        ; 0587
        7E          6B 00000400   8F  C7 004DF          DIVL3   #1024, DS$GL_MEMSIZE, -(SP)                  ;
                                  6B  DD 004E7          PUSHL   DS$GL_MEMSIZE                                ;
                             5B   A6  9F 004E9          PUSHAB  FORMAT4                                      ;
                    67            04  FB 004EC          CALLS   #4, DS$PRINTF                                ;
                       00000000G  EF  94 004EF 40$:     CLRB    DS$GB_TYPECODE                               ; 0588
                                      04 004F5          RET                                                  ; 0589

; Routine Size:  1270 bytes,    Routine Base:  CODE + 0025


;   0590  1
```

```
1       !                                                                                   BEGIN   [05]
1       %SBTTL 'VRSETMEM'
1
1       GLOBAL ROUTINE VRSETMEM (CLI_BASE) : JSB_CLI =
1
1       !++
1       ! FUNCTIONAL DESCRIPTION:
1       !
1       !       This routine is called from the CLI module. It sets the value of
1       !       DS$GL_SET_MEMSIZE to the value specified, remaps memory, and goes
1       !       to command level (DS>).
1       !
1       ! CALLER(S):
1       !
1       !       CLI
1       !
1       ! FORMAL PARAMETERS:
1       !
1       !       R2 - Points to the CLI data table base.
1       !
1       ! IMPLICIT INPUTS:
1       !
1       !       CLI_BASE [CLI$L_DATA]  - Indicates the new value of memory.
1       !
1       ! IMPLICIT OUTPUTS:
1       !
1       !       New value for DS$GL_SET_MEMSIZE.
1       !
1       ! COMPLETION CODES:
1       !
1       !       None
1       !
1       ! SIDE EFFECTS:
1       !
1       !       Executes user's cleanup code.
1       !--
1
2       BEGIN
2
2           EXTERNAL ROUTINE
2               BEGIN_BLISS : INT,
2               DSR$MMENABLE : JSB_0,
2               DS_CLEANUP : JSB_NONE,
2               INI$LOAD_DEVICE : JSB_NONE,                                 !                   [06]
2               INI$PTABLE : JSB_NONE,
2               MAPMEM : JSB_NONE,
2               SCRIPT$FLUSH : JSB_NONE;
2
2           LOCAL
2               WASSET;
2
2           MAP
2               CLI_BASE : REF BLOCK [, BYTE];
2
3           IF (.CLI_BASE [CLI$L_DATA] LSS 0)
3               THEN $DS_PRINTF ($ASCIC (%STRING ('?? Error, negative number of pages specified!!', %CHAR (CR, LF))))
3               ELSE BEGIN
```

```
;   0649  3                     DS$GL_SET_MEMSIZE = .CLI_BASE [CLI$L_DATA];          ! Set the new value of memory
;   0650  3                     SCRIPT$FLUSH ();                                     ! Flush scripts if console command
;   0651  3                     DS_CLEANUP ();                                       ! Execute user's cleanup code
;   0652  3                     WASSET = DSR$MMENABLE (0);                           ! Turn off memory management
;   0653  3                     MAPMEM ();                                           ! Remap memory with new value
;   0654  3                     IF .WASSET EQL SS$_WASSET THEN DSR$MMENABLE (1);     ! Reset memory management if on before
;   0655  3                     INI$PTABLE ();                                       ! Initialize P-tables
;   0656  3                     INI$LOAD_DEVICE ();                                  ! ATT console and boot devices          [06]
;   0657  3                     BEGIN_BLISS ()                                       ! Reset and go to DS>
;   0658  3                 END
;   0659  1     END;                                                                 !                                End    [05]


                                                        .PSECT  DATA,NOWRT,NOEXE,  SHR,2

61  67  65  6E  20  2C  72  6F  72  72  45  20  3F  3F  30  00361 P.ABU:  .ASCII  \0?? Error, negative number of pages spec\  ;
20  66  6F  20  72  65  62  6D  75  6E  20  65  76  69  74  00370                                                              ;
                    63  65  70  73  20  73  65  67  61  70  0037F                                                              ;
                        0A  0D  21  21  64  65  69  66  69  00389          .ASCII  \ified!!\<13><10>                           ;

                                                        .EXTRN  BEGIN_BLISS, DSR$MMENABLE
                                                        .EXTRN  DS_CLEANUP, INI$LOAD_DEVICE
                                                        .EXTRN  INI$PTABLE, MAPMEM
                                                        .EXTRN  SCRIPT$FLUSH

                                                        .PSECT  CODE,NOWRT,  SHR,2

                                    52  DD 00000 VRSETMEM::
                                                            PUSHL   R2                              ; 0595
                                1C  A2  D5 00002            TSTL    28(CLI_BASE)                    ; 0646
                                    0F  18 00005            BGEQ    1$                              ;
                        00000000'   EF  9F 00007            PUSHAB  P.ABU                           ; 0647
            00000000G   9F          01  FB 0000D            CALLS   #1, a#DS$PRINTF                 ;
                                    45  11 00014            BRB     3$                              ;
            00000000G   EF      1C  A2  D0 00016 1$:        MOVL    28(CLI_BASE), DS$GL_SET_MEMSIZE ; 0649
                    00000000G   EF  16  D0 0001E            JSB     SCRIPT$FLUSH                    ; 0650
                    00000000G   EF  16  00024               JSB     DS_CLEANUP                      ; 0651
                                50  D4 0002A                CLRL    R0                              ; 0652
                    00000000G   EF  16  0002C               JSB     DSR$MMENABLE                    ;
                52              50  D0 00032               MOVL    R0, WASSET                      ;
                    00000000G   EF  16  00035               JSB     MAPMEM                          ; 0653
                09              52  D1 0003B               CMPL    WASSET, #9                      ; 0654
                                09  12 0003E               BNEQ    2$                              ;
                50              01  D0 00040               MOVL    #1, R0                          ;
                    00000000G   EF  16  00043               JSB     DSR$MMENABLE                    ;
                    00000000G   EF  16  00049 2$:           JSB     INI$PTABLE                      ; 0655
                    00000000G   EF  16  0004F               JSB     INI$LOAD_DEVICE                 ; 0656
                    00000000G   EF  16  00055               JSB     BEGIN_BLISS                     ; 0657
                                04  BA 0005B 3$:           POPR    #^M<R2>                         ; 0659
                                    05  0005D               RSB                                     ;

; Routine Size:  94 bytes,    Routine Base:  CODE + 051B

;   0660  1
;   0661  1     End                                                                  ! End of module
```

```
;   0662  0     Eludom



;                                  PSECT SUMMARY
;
;       Name                    Bytes                          Attributes
;
;  DATA                           914  NOVEC,NOWRT,  RD ,NOEXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  WORK                             8  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                          1401  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



Symbol                    Type  Defined  Referenced ...
------------------------ -------- ----- ----------------
$ASCIC                   Macro    Lib01   142     143     144     145     146     147     150     151     152     153     154
                                          155     156     157     158     159     160     161     162     163     164     165
                                          166     167     273     281     292     300     309     317     326     335     340
                                          345     350     355     360     502     511     519     531     543     570     574
                                          577     586     647
$DS_DSADEF               Macro    Lib01   128
$DS_PRINTF               Macro    Lib01   262     270     278     287     297     306     314     323     332     337     342
                                          347     352     357     362     370     376     385     391     400     406     415
                                          421     430     436     445     454     463     472     481     490     499     508
                                          516     527     538     548     553     558     568     573     575     582     647
$DS_TYPEDEF              Macro    Lib01   129
$EQULST                  Macro    Lib03   113     129
AX_BUFF                  External   185   327a
BEGIN_BLISS              ExtRout    632   657c
CHMK                     Builtin    369   369
                         Builtin    384   384
                         Builtin    399   399
                         Builtin    414   414
                         Builtin    429   429
                         Builtin    444   444
                         Builtin    453   453
                         Builtin    462   462
                         Builtin    471   471
                         Builtin    480   480
                         Builtin    489   489
CLI$L_DATA               Macro    Lib02   646     649
CLI_BASE                 RoutForm   595   644m    646a.   649a.
CMK                      Macro    Lib02   369     384     399     414     429     444     453     462     471     480     489
CMK$SGIPR                ExtRout    369   369c    384e    384c    399e    399c    414e    414c    429e    429c    444e    444c
                                          453e    453c    462e    462c    471e    471c    480e    480c    489e    489c
CMK$_APBOOT              Literal    113
CMK$_ASTEXIT             Literal    113
CMK$_CANTIM              Literal    113
CMK$_HIBER               Literal    113
CMK$_INITSCB             Literal    113
CMK$_LAST                Literal    113
```

Symbol	Type	Defined	Referenced ...
CMK$_MMENABLE	Literal	113	
CMK$_RCONSOLE	Literal	113	
CMK$_REFRESH	Literal	113	
CMK$_SCHDWK	Literal	113	
CMK$_SETIMR	Literal	113	
CMK$_SETIPL	Literal	113	
CMK$_SETPRT	Literal	113	
CMK$_SGIPR	Literal	113	369 384 399 414 429 444 453 462 471 480 489
CMK$_TCONSOLE	Literal	113	
CMKDEF	Macro	Lib02	113
CR	Literal	138	647
DEBUG_HI	External	206	577.
DEBUG_LO	External	205	570.
DS$AX_ARB	External	189	351a
DS$AX_JIB	External	188	346a
DS$AX_SOFTPCB	External	190	356a
DS$A_PRGBGN	External	198	274a
DS$GA_LASTADR	External	202	533. 536. 544.
DS$GB_TYPECODE	External	182	258= 588=
DS$GL_BUFCNT	External	184	552. 557. 562.
DS$GL_FLAGS	External	183	219.
DS$GL_IPR_VALUE	Global	175	375. 390. 405. 420. 435. 450. 459. 468. 477. 486. 495.
DS$GL_LOOKASIDE	External	200	512. 514.
DS$GL_MEMSIZE	External	203	536. 544. 587.
DS$GL_SET_MEMSIZE	External	204	649=
DS$K_PRINTB	Literal	129	
DS$K_PRINTF	Literal	129	
DS$K_PRINTI	Literal	129	
DS$K_PRINTX	Literal	129	
DS$K_TYPE_ABORT_PROGRAM	Literal	129	
DS$K_TYPE_ABORT_TEST	Literal	129	
DS$K_TYPE_COMMAND_ERR	Literal	129	
DS$K_TYPE_COMMAND_OUT	Literal	129	258
DS$K_TYPE_CRD_AUTOTEST	Literal	129	
DS$K_TYPE_DS_PROMPT	Literal	129	
DS$K_TYPE_DS_START	Literal	129	
DS$K_TYPE_ERRDEV	Literal	129	
DS$K_TYPE_ERRHARD	Literal	129	
DS$K_TYPE_ERROR_BODY	Literal	129	
DS$K_TYPE_ERROR_END	Literal	129	
DS$K_TYPE_ERRPREP	Literal	129	
DS$K_TYPE_ERRSOFT	Literal	129	
DS$K_TYPE_ERRSUP	Literal	129	
DS$K_TYPE_ERRSYS	Literal	129	
DS$K_TYPE_ERR_HALT	Literal	129	
DS$K_TYPE_EXCEPTION	Literal	129	
DS$K_TYPE_EXCEPTION_HEAD	Literal	129	
DS$K_TYPE_FIRST_PASS	Literal	129	
DS$K_TYPE_GENERAL	Literal	129	
DS$K_TYPE_GENERAL_ERROR	Literal	129	
DS$K_TYPE_NO_TESTS	Literal	129	
DS$K_TYPE_PARAM_ERROR	Literal	129	
DS$K_TYPE_PROGRAM_END	Literal	129	
DS$K_TYPE_PROGRAM_INFO	Literal	129	
DS$K_TYPE_PROGRAM_START	Literal	129	

Symbol	Type	Defined	Referenced ...									
DS$K_TYPE_QIO_INVADP	Literal	129										
DS$K_TYPE_QIO_NODRIVER	Literal	129										
DS$K_TYPE_QIO_WRONGVER	Literal	129										
DS$K_TYPE_SCRIPT_ECHO	Literal	129										
DS$K_TYPE_SCRIPT_PNF	Literal	129										
DS$K_TYPE_SCRIPT_PROMPT	Literal	129										
DS$K_TYPE_SCRIPT_SKIP	Literal	129										
DS$K_TYPE_SEQUENCE_ERROR	Literal	129										
DS$K_TYPE_START_ERR	Literal	129										
DS$K_TYPE_START_LIST	Literal	129										
DS$K_TYPE_SUMMARY	Literal	129										
DS$K_TYPE_USER_PROMPT	Literal	129										
DS$PRINTF	ExtRout	266	266c	274e	274c	282e	282c	293e	293c	301e	301c	310e	310c
			319e	319c	327e	327c	336e	336c	341e	341c	346e	346c	351e
			351c	356e	356c	361e	361c	366e	366c	375e	375c	381e	381c
			390e	390c	396e	396c	405e	405c	411e	411c	420e	420c	426e
			426c	435e	435c	441e	441c	450e	450c	459e	459c	468e	468c
			477e	477c	486e	486c	495e	495c	503e	503c	512e	512c	520e
			520c	533e	533c	544e	544c	552e	552c	557e	557c	562e	562c
			570e	570c	574e	574c	577e	577c	587e	587c	647e	647c	
DS$V_LODFLG	Macro	Lib02	219										
DSA$AL_APTMAIL	Bind	128	128										
DSA$GL_APTCOM	Bind	128											
DSA$GL_DEVLEN	Bind	128											
DSA$GL_DEVNAM	Bind	128											
DSA$GL_ERRNO	Bind	128											
DSA$GL_EVENT	Bind	128											
DSA$GL_FLAGS	Bind	128	257	565									
DSA$GL_MSGPTR	Bind	128											
DSA$GL_MSGTYP	Bind	128											
DSA$GL_PASSES	Bind	128											
DSA$GL_PASSNO	Bind	128											
DSA$GL_SECTNO	Bind	128											
DSA$GL_SID	Bind	128											
DSA$GL_SUBTNO	Bind	128											
DSA$GL_TESTNO	Bind	128											
DSA$GL_UNITS	Bind	128											
DSA$V_DEBUG	Macro	Lib01	565										
DSA$V_USER	Macro	Lib01	257										
DSR$CHECKLOAD	GlobRout	217	210f	276c	285c								
DSR$MMENABLE	ExtRout	633	652c	654c									
DSV$SHOWMEMORY	GlobRout	224	211f										
DS_CLEANUP	ExtRout	634	651c										
ENV$S_SUPER	Literal	Lib01	219										
ENV$V_SUPER	Literal	Lib01	219										
ENV$_SUPER	Literal	Lib01	219										
ESTKPTR	External	195	411a										
EXE$GL_SPLITADR	External	201	514.	520.									
FORMAT	Bind	142	266a	274a	282a	301a	310a	327a	336a	341a	346a	351a	356a
			361a	366a	503a	512a	520a	552a	557a	562a	570a	577a	
FORMAT2	Bind	143	319a	375a	390a	405a	420a	435a	450a	468a	486a		
FORMAT3	Bind	144	381a	396a	411a	426a	441a	459a	477a	495a			
FORMAT4	Bind	145	587a										
FORMAT5	Bind	146	293a	533a	544a								
FORMAT6	Bind	147	574a										

Symbol	Type	Defined	Referenced ...
GET1ST_	Macro	Lib03	113 129
GET2ND_	Unbound		113 129
INI$LOAD_DEVICE	ExtRout	635	656c
INI$PTABLE	ExtRout	636	655c
INT	Linkage	Lib02	632
ISTKPTR	External	194	396a
JSB_0	Linkage	Lib02	633
JSB_CLI	Linkage	Lib02	213 595
JSB_MFPR	Linkage	132	369 384 399 414 429 444 453 462 471 480 489
JSB_NONE	Linkage	Lib02	210 211 217 224 634 635 636 637 638
KSTKPTR	External	193	381a
L$A_LASTADR	Literal	Lib01	282 293
L$L_ENVIRON	Literal	Lib01	219
LF	Literal	139	647
MAPMEM	ExtRout	637	653c
MEMORYFLAGS	Local	237	243= 245. 247= 249. 252= 253= 254= 260. 268. 276. 284. 295. 303. 304. 312. 321. 329. 497. 505. 524. 525. 535. 546.
MOVPSL	Builtin	369	369
	Builtin	384	384
	Builtin	399	399
	Builtin	414	414
	Builtin	429	429
	Builtin	444	444
	Builtin	453	453
	Builtin	462	462
	Builtin	471	471
	Builtin	480	480
	Builtin	489	489
NUL2ND_	Macro	Lib03	113 129
ONLINE	Local	238	257= 329. 571. 580.
POPT_BASE	External	199	
PCB_BASE	External	187	341a
PHD_BASE	External	186	336a
PR$_ESP	Literal	Lib03	397
PR$_ISP	Literal	Lib03	382
PR$_KSP	Literal	Lib03	367
PR$_P0BR	Literal	Lib03	442
PR$_P0LR	Literal	Lib03	451
PR$_P1BR	Literal	Lib03	460
PR$_P1LR	Literal	Lib03	469
PR$_SBR	Literal	Lib03	478
PR$_SLR	Literal	Lib03	487
PR$_SSP	Literal	Lib03	412
PR$_USP	Literal	Lib03	427
R3	GlobReg	241	367= 382= 397= 412= 427= 442= 451= 460= 469= 478= 487=
R5	GlobReg	241	368= 383= 398= 413= 428= 443= 452= 461= 470= 479= 488=
SCB_BASE	External	191	361a
SCRIPT$FLUSH	ExtRout	638	650c
SHOWMEMORY	Routine	231	212f 226c
SHOWMEMORYFLAGS	Global	174	243.
SM_V_ALL	Literal	137	249
SM_V_BUFFER	Literal	135	253 284 535 546
SM_V_DATA	Literal	136	254 295 304 329 505 525
SM_V_MAP	Literal	134	247 252 260 268 276 295 303 312 321 497 505

Symbol	Type	Defined	Referenced ...				
			524				
SP	Builtin	369	369=	369.	369a=		
	Builtin	384	384=	384.	384a=		
	Builtin	399	399=	399.	399a=		
	Builtin	414	414=	414.	414a=		
	Builtin	429	429=	429.	429a=		
	Builtin	444	444=	444.	444a=		
	Builtin	453	453=	453.	453a=		
	Builtin	462	462=	462.	462a=		
	Builtin	471	471=	471.	471a=		
	Builtin	480	480=	480.	480a=		
	Builtin	489	489=	489.	489a=		
SS$_WASSET	Literal	Lib03	654				
SSTKPTR	External	196	426a				
STACK_BASE	External	192	366a				
TEMP	Local	236					
TEMP_CHMK_CODE	Local	369	369a=	369.			
	Local	384	384a=	384.			
	Local	399	399a=	399.			
	Local	414	414a=	414.			
	Local	429	429a=	429.			
	Local	444	444a=	444.			
	Local	453	453a=	453.			
	Local	462	462a=	462.			
	Local	471	471a=	471.			
	Local	480	480a=	480.			
	Local	489	489a=	489.			
TEMP_PSL	Local	235					
	Local	369	369=	369.			
	Local	384	384=	384.			
	Local	399	399=	399.			
	Local	414	414=	414.			
	Local	429	429=	429.			
	Local	444	444=	444.			
	Local	453	453=	453.			
	Local	462	462=	462.			
	Local	471	471=	471.			
	Local	480	480=	480.			
	Local	489	489=	489.			
T_BASE	Bind	162	450a	468a	486a		
T_BUFFERSPACE	Bind	167	293a	544a			
T_DS	Bind	151	319a	533a			
T_EXECUTIVE	Bind	154	405a	411a			
T_INTERRUPT	Bind	153	390a	396a			
T_KERNEL	Bind	152	375a	381a			
T_LENGTH	Bind	163	459a	477a	495a		
T_P0	Bind	159	450a	459a			
T_P0_BUFCNT	Bind	164	552a				
T_P1	Bind	160	468a	477a			
T_P1_BUFCNT	Bind	165	557a				
T_RESERVED	Bind	150	266a	366a			
T_S0_BUFCNT	Bind	166	562a				
T_SP	Bind	158	375a	390a	405a	420a	435a
T_STACK	Bind	157	381a	396a	411a	426a	441a
T_SUPERVISOR	Bind	155	420a	426a			

Symbol	Type	Defined	Referenced ...	
T_SYSTEM	Bind	161	486a	495a
T_USER	Bind	156	435a	441a
ULTRIX_IOB_ADD	External	207	503.	
USTKPTR	External	197	441a	
VRSETMEM	GlobRout	595	213f	
WASSET	Local	641	652=	654.

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]SHOWMEMORY.B32;1
8	Module SHOWMEMORY	
101	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
103	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
105	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
662	Eludom SHOWMEMORY	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
a	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5	940	11	1	96	00:00.0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5	675	8	1	47	00:00.0
SYS$COMMON:[SYSLIB]LIB.L32;2	20450	15	0	1101	00:00.8

COMMAND QUALIFIERS

```
BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:SHOWMEMORY.B32
```

```
; Size:          1401 code + 922 data bytes
; Run Time:         00:06.1
; Elapsed Time:     00:08.8
; Lines/CPU Min:     6532
; Lexemes/CPU-Min:101279
; Memory Used:  402 pages
; Compilation Complete
```

Table of contents

(3)   241   Declarations
(3)   358   VRStart and VRReStart Routines
(3)   812   VRAbort and DSX$Abort Routines
(4)   949   VRSupport Routines
(4)   1010   DSV$SHOWTESTS
(4)   1049   VRShowSections Routine

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element START.MAR
0000     2 ; *10   27-AUG-1987 09:49:49 MEIER "CHANGES TO DS"
0000     3 ; *9     9-JUN-1986 10:36:00 BERGAZZI "CHANGE TYPECODE ON /START FOR VSDP"
0000     4 ; *8    28-MAY-1986 13:31:47 BERGAZZI "/TIME CHANGE"
0000     5 ; *7    20-MAY-1986 10:24:37 BERGAZZI "/TIME"
0000     6 ; *6    31-MAR-1986 16:37:36 BAGGETT "FINISHED"
0000     7 ; *5    28-MAR-1986 10:44:01 BAGGETT "DEVICE NAMES SEPARATED WHEN TESTING"
0000     8 ; *4    19-MAR-1986 15:57:35 MUSE "<no reason given>"
0000     9 ; *3    19-MAR-1986 15:44:57 MUSE "<no reason given>"
0000    10 ; *2    10-MAR-1986 10:48:25 ANDELLA "<no reason given>"
0000    11 ; *1     6-FEB-1986 15:22:34 DS ""
0000    12 ; DEC/CMS REPLACEMENT HISTORY, Element START.MAR
0000    13         .TITLE  START   *** START Handle START command
0000    14         .IDENT  /10.1-10/                                              ;G:10
0000    15         .NoShow Conditionals
0000    16
0000    17 ;
0000    18 ; Copyright (c) 1977, 1981, 1982, 1983, 1984, 1985, 1986, 1987          ;G:10
0000    19 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    20 ;
0000    21 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    22 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    23 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    24 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    25 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    26 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    27 ; REMAIN IN DEC.
0000    28 ;
0000    29 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    30 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    31 ; CORPORATION.
0000    32 ;
0000    33 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    34 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    35
0000    36 ;++
0000    37 ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    38 ;
0000    39 ; ABSTRACT:     CONTAINS ALL PROGRAM STARTING AND STOPPING FUNCTIONS:
0000    40 ;                        "START" COMMAND SUBROUTINE.
0000    41 ;                        "CONTINUE" COMMAND SUBROUTINE.
0000    42 ;                        "ABORT" COMMAND SUBROUTINE.
0000    43 ;                       $DS_ABORT DIAGNOSTIC SERVICE PROCEDURE.
0000    44 ;
0000    45 ; ENVIRONMENT:
0000    46 ;
0000    47 ; AUTHOR:       KEN CHAPMAN     10-NOV-77       VERSION 01.
```

```
0000    ; MODIFIED BY:
0000    ;                TOM SOUTTER     17-NOV-77       VERSION 02 (ESSAA-3.04).
0000    ;        01      DSPR #25 - CONTINUE FROM BREAKPOINT PROBLEMS.
0000    ;        02      DSPR #42 - APT FUNCTION UPDATE.
0000    ;        03      FIX TO AVOID "BUG CHECK" WHEN PROCEEDING FROM AN EXCEPTION.
0000    ;
0000    ;                KEN CHAPMAN     08-DEC-77       VERSION 03 (ESSAA-3.05).
0000    ;        04      ADD NEW-LINE TO END OF PROGRAM TITLE MESSAGE.
0000    ;        05      DSPR #4, #5 - ADDED CONTINUE AND ABORT MESSAGES.
0000    ;
0000    ;                N. HOWGATE      15-FEB-78       VERSION 04 (ESSAA-4.00).
0000    ;        06      QIO SUPPORT
0000    ;
0000    ;                KEN CHAPMAN     20-MAR-78       VERSION 05 (ESSAA-4.01).
0000    ;        07      ALLOW CONTINUE FROM ANY BREAKPOINT OR ^C.
0000    ;        08      ADD TEST COUNT TO PROGRAM TITLE MESSAGE.
0000    ;        09      CLEAR TIMER VECTOR IN STARTUP SEQUENCE.
0000    ;                Roger Riggs     12-Jun-78       VERSION 06 (ESSAA-4.03).
0000    ;        10      Standardizing program related messages.
0000    ;                NICK HOWGATE    23-JUN-78
0000    ;        11      CORRECT 'APT' ABORT PROBLEM
0000    ;                Roger Riggs             07-Aug-78
0000    ;        12      Install test number range checking, also update DS$GL_NUMTEST.
0000    ;                Fix so DS$M_DONFLG is set by DS_CLEANUP and not checked
0000    ;                before calling it.
0000    ;        13      Changes reflected by new structure of basic loop.
0000    ;                i.e. START and RESTART commands call test program directly
0000    ;                Roger Riggs             23-OCT-1978     VERSION 07 (ESSAA-5.01)
0000    ;        14      Changes to VRSTART to support new QIO initialization.
0000    ;        15      Added code to verify that this program is compatible with
0000    ;                this supervisor, via ENV$V_SUPER, Value must match
0000    ;                ENV$_SUPER, in DIAG. Updated when supervisor interface
0000    ;                changes REQUIRE diagnostic changes.
0000    ;                Roger Riggs     4-Sept-1979
0000    ;        17      Removed check of APT flag during start command
0000    ;                Roger Riggs, 14-Feb-1980, Vervsion 5.2
0000    ;        18      Changed so program will not be started if QIO initialization
0000    ;                fails, QIO has already printed the error message.
0000    ;                Roger Riggs, 12-Mar-1980, Version 5.3
0000    ;        19      In VRABORT or DSX$ABORT change mode back to kernel before
0000    ;                doing cleanup but after the abort message is printed
0000    ;                Roger Riggs, 5-Jun-1980, Version 5.4
0000    ;
0000    ;        20      Added code to change mode and stack back to kernel
0000    ;                before calling cleanup as a result of start.
0000
```

```
0000     95 ;        21      - dave butenhof, 09-Sep-1981, version 6.-
0000     96 ;                It is desirable, under APT, and if program error had
0000     97 ;                occured, to have ABORT leave machine state intact, for
0000     98 ;                purposes of isolating error.  Add special code to skip
0000     99 ;                INIT_CONTEXT and CLEANUP calls if in APT mode and error
0000    100 ;                flag set.
0000    101 ;
0000    102 ;        22      - Marion Baggett 80-oct-1981 Version 6.-
0000    103 ;                Added new entry point to support command SHOW SUPPORT
0000    104 ;
0000    105 ;        23      - Jack Stansbury, 21-Oct-1981, Version 6.-
```

```
                        0000   106 ;               Added calls to QA$MAIN for the QA enhancement.
                        0000   107 ;               Also added .LIBRARY statements for $DS and $DIAG.
                        0000   108 ;
                        0000   109 ;       24      - Jack Stansbury, 22-Oct-1981, Version 6.-
                        0000   110 ;               Fixed truncation errors.
                        0000   111 ;
                        0000   112 ;       25      - Jack Stansbury, 26-Oct-1981, Version 6.-
                        0000   113 ;               Changed most of the error messages to mixed case
                        0000   114 ;               (as per government affirmative action regulations)!
                        0000   115 ;
                        0000   116 ;       26      - Dave Butenhof, 19-Nov-1981, version 6.-
                        0000   117 ;               Add type codes to printouts
                        0000   118 ;
                        0000   119 ;       27      - Jack Stansbury, 16-Nov-1981, Version 6.5
                        0000   120 ;               Added the VRSHOWSECTIONS routine.
                        0000   121 ;
                        0000   122 ;       28      - Jack Stansbury, 21-Nov-1981, Version 6.5
                        0000   123 ;               Commented out three of the calls to QA_MAIN because it
                        0000   124 ;               appears they will not be needed for the 6.5 DS version.
                        0000   125 ;
                        0000   126 ;       29      - Jack Stansbury, 5-Jan-1982, Version 6.6
                        0000   127 ;               Added comments to the START routine in preparation for the
                        0000   128 ;               version 30 changes. Also took out the commented-out calls
                        0000   129 ;               to QA_MAIN since I will not be needing them.
                        0000   130 ;
                        0000   131 ;       30      - Jack Stansbury, 6-Jan-1982, Version 6.6
                        0000   132 ;               Added code and moved other code around in the START routine so
                        0000   133 ;               as to facilitate the "QA Start Loop".
                        0000   134 ;
                        0000   135 ;       31      - Jack Stansbury, 15-Jan-1982, Version 6.6
                        0000   136 ;               Changed the FMT_PROG header line so that it would print the
                        0000   137 ;               number of tests on the first line. Only the time appears on the
                        0000   138 ;               second line. This is in keeping with the EndPass message and
                        0000   139 ;               the End-Of-Looping message (in EndSub).
                        0000   140 ;
                        0000   141 ;       32      - Jack Stansbury, 20-Jan-1982, Version 6.6
                        0000   142 ;               Added code to not print the header line if the QA Loop on
                        0000   143 ;               Subtest check was executing and if a flag bit had been cleared.
                        0000   144 ;               This prevents the header line from being output once for every
                        0000   145 ;               subtest in the diagnostic when the Loop on Subtest check
                        0000   146 ;               routine is executing.
                        0000   147 ;
                        0000   148 ;       33      Marion Baggett, 9-Apr-1982, Version 6.7
                        0000   149 ;               Added .NoShow Conditionals after .IDENT , ordered .LIBRARY
                        0000   150 ;               statements. Change DS$PRINTF statement to use type codes
                        0000   151 ;
                        0000   152 ;       34      Richard Brown, 30-June-82, Version 6.8
                        0000   153 ;               Addition of call to MAP_DEBUGGER so repeated STARTs will
                        0000   154 ;               not wipe out the debugger once it has been loaded.
                        0000   155 ;               Also, addition of loading debugger before diagnostic is
                        0000   156 ;               started, if proper flag is set.
                        0000   157 ;
                        0000   158 ;       35      Marion Baggett, 24-Sept-82  Version 6.9
                        0000   159 ;               Fixed truncation error.
                        0000   160 ;
                        0000   161 ;       36      Bob Bergazzi    2-Nov-1982      Version 6.9
                        0000   162 ;               Changed VRSTART so that if we are loaded under APT do not
```

```
                        0000   163 ;               check the DSA$V_LOAD_DEBUGGER flag. APT sets this flag for
                        0000   164 ;               reasons long since lost. When running under APT, every
                        0000   165 ;               START would cause the supervisor to try and load the debugger
                        0000   166 ;               from the console device and cause APT to timeout.
                        0000   167 ;
                        0000   168 ;       37      Jack Stansbury, 8-Mar-1983, Version 6.11
                        0000   169 ;               Changed the abort routines to make sure they print the true
                        0000   170 ;               (if there is one) user PC.
                        0000   171 ;
                        0000   172 ;       38      John Ciukaj     3-May-1983      Version 6.11
                        0000   173 ;               - Alter Start routine so that an error does not occur
                        0000   174 ;                 when an ambiguous section name is found AND the section name
                        0000   175 ;                 is equivalent to the name entered.
                        0000   176 ;
                        0000   177 ;       39      Bob Bergazzi    May 16, 1983    Version 6.11
                        0000   178 ;               Changed the order of the .LIB statements.
                        0000   179 ;
                        0000   180 ;       40      Bob Bergazzi    Sep 21, 1983    Version 6.13
                        0000   181 ;               Added BIN_TIME quadword in PSECT WORK for START/TIME switch.
                        0000   182 ;               Just before call to DISPATCH, do a $SETIMR if /TIME was given.
                        0000   183 ;               Commented out for version 6.13 and 6.14, and 7.0
                        0000   184 ;
                        0000   185 ;       41      Bob Bergazzi    Jan. 26, 1984   Version 6.14
                        0000   186 ;               Fixed ABORT so it won't print out VMS system addresses as the
                        0000   187 ;               user PC.
                        0000   188 ;
                        0000   189 ;       42      Ted Salem       Aug. 3, 1984    Version 7.10
                        0000   190 ;               Added SHOW TESTS command.
                        0000   191 ;
                        0000   192 ;       43      Bob Bergazzi    Feb. 26, 1985   Version 8.1
                        0000   193 ;               Fixed edit [40] (/TIME).
                        0000   194 ;
                        0000   195 ;       44      Bob Bergazzi    Mar. 4, 1985    Version 8.1
                        0000   196 ;               Commented out edit 43.
                        0000   197 ;
                        0000   198 ;       45      Amy Iorio       June 17, 1985   Version 8.3
                        0000   199 ;               clear externals for Printrev
                        0000   200 ;
                        0000   201 ;       46      Ted Salem       Oct. 15, 1985   Version 9.0
                        0000   202 ;               Do above (#45) properly by clearing
                        0000   203 ;               entire date structure.
                        0000   204 ;
                        0000   205 ;       47      J. Baranski      1-Nov-1985     Version 9.1
                        0000   206 ;               Un Comment Out START/TIME.
                        0000   207 ;
                        0000   208 ;       48      M. Baggett      14-Jan-1986     Version 9.1
                        0000   209 ;               Stop truncating long device name when starting program
                        0000   210 ;                                                                       ;G:2
                        0000   211 ;       49      Domenic Andella 7-Feb-86        Version 10.0            ;G:2
                        0000   212 ;               For every start or run sequence in an mp system, the    ;G:2
                        0000   213 ;               primary processor should clear its own interlocking     ;G:2
                        0000   214 ;               data structures, therefore add jsb to MP$_CLEAR_LOCKS   ;G:2
                        0000   215 ;               in routine VRSTART. In addition, clear MP$GR_BOOTED_PROCESSORS, ;G:2
                        0000   216 ;               MP$GR_IDLING_PROCESSORS, MP$GL_FLAGS, and MP$GL_CONDITION_FLAGS. ;G:
                        0000   217 ;                                                                       ;G:2
                        0000   218 ;       50      M. Baggett      26-Mar-1986     Version 10              ;G:5
                        0000   219 ;               Separate device names in list printed after starting test   ;G:5
```

```
0000   220 ;                 with tabs.                                                    ;G:5
0000   221 ;                                                                               ;G:5
0000   222 ;        51       M. Baggett      31-Mar-1986     Version 10                   ;G:6
0000   223 ;                 Fixed so diagnostics run offline.                             ;G:6
0000   224 ;                                                                               ;G:6
0000   225 ;        52       Bob Bergazzi    16-May-1986     Version 10.1                 ;G:7
0000   226 ;                 Added /TIME changes.                                          ;G:7
0000   227 ;                                                                               ;G:8
0000   228 ;        53       Bob Bergazzi    28-May-1986     Version 10.1                 ;G:8
0000   229 ;                 Changed running under VMS from elapsed time to execution      ;G:8
0000   230 ;                 time.                                                         ;G:8
0000   231 ;                                                                               ;G:9
0000   232 ;        54       Bob Bergazzi    9-Jun-1986      Version 10.2                 ;G:9
0000   233 ;                 Changed typecode on /TIME abort to DS$K_TYPE_PROGRAM_END.     ;G:9
0000   234 ;                 It turns out that VSDP uses these typecodes.                  ;G:9
0000   235 ;                                                                               ;G:10
0000   236 ;        10       Stephen Meier   27-Aug-1987     Version 10.9                 ;G:10
0000   237 ;                 Recompile to included changes to BR_IF_NOT_MP in DS.          ;G:10
0000   238 ;                                                                               ;G:10
0000   239 ;--
```

```
                        0000    241          .SBTTL  Declarations
                        0000    242 ;
                        0000    243 ; INCLUDE FILES:
                        0000    244 ;
                        0000    245          .LIBRARY        /Sys$library:lib/       ;                       [33]
                        0000    246          .LIBRARY        /$DS/                   ;                       [39]
                        0000    247          .LIBRARY        /$DIAG/                 ;                       [39]
                        0000    248
                        0000    249 ;
                        0000    250 ; MACROS:
                        0000    251 ;
                        0000    252
                        0000    253 ;
                        0000    254 ; EXTERNALS:
                        0000    255 ;
                        0000    256          .EXTRN  DEVREV$T                        ; Begin Printrev structure [46]
                        0000    257          .EXTRN  DEVREV$T_x                      ; End Printrev structure   [46]
                        0000    258          .WEAK   MP$_CLEAR_LOCKS                         ;[49]                   ;G:2
                        0000    259          .WEAK   MP$GR_BOOTED_PROCESSORS                 ;[49]                   ;G:2
                        0000    260          .WEAK   MP$GR_IDLING_PROCESSORS                 ;[49]                   ;G:2
                        0000    261          .WEAK   MP$GL_FLAGS                             ;[49]                   ;G:2
                        0000    262          .WEAK   MP$GL_CONDITION_FLAGS                   ;[49]                   ;G:2
                        0000    263
                        0000    264 ;
                        0000    265 ; EQUATED SYMBOLS:
                        0000    266 ;
                        0000    270          $DS_CFDEF
                        0000    271          $DS_CLIDEF
                        0000    272          $DS_DSADEF
                        0000    273          $DS_ENVDEF
                        0000    274          $DS_HDRDEF
                        0000    275          $DS_SCBDEF
                        0000    276          $PRDEF
                        0000                 .SAVE   LOCAL_BLOCK
                        0000                 .PSECT  $ABS$,ABS
                00000000 FE70                .=0
                        0000                 .RESTORE
                        0000    277          APTDEF
                        0000    278          CLIDEF
                        0000    279          DSPDEF
                        0000    280          DSFDEF
                        0000    281          DSQA                            ; DSQA$K constant definitions      [32]
                        0000    282          DS_QADEFS                       ; Other QA$K constant definitions  [32]
                        0000    283          $ds_typedef                     ; Define type codes                [26]
                        0000    284
```

```
                00000000   289          .PSECT  WORK, NOSHR, NOEXE, WRT, LONG                          ;  [40]
                    0000   290
        00000008    0000   291 BIN_TIME::      .BLKQ 1                  ; Holds converted 64-bit time      [40]
                    0008   292                                          ; specified by user with /TIME     [40]
        00000010    0008   293 BEGIN_TIME::    .BLKQ 1                  ; Holds absolute time when diag was[40]
                    0010   294                                          ; started.                         [40]
        00000018    0010   295 POLL_TIME::     .BLKQ 1                  ; Holds abs time at KB_CHECK.  [40][52] ;G:7
                    0018   296
        0000001C    0018   297 DUE_TIME::      .BLKL   1                ; Holds absolute time when diag [52]  ;G:7
                    001C   298                                          ; is due                        [52]  ;G:7
```

```
                                00000000   300          .PSECT  Data, Shr, NoExe, NoWrt, Byte
                                    0000   301
                                    0000   302          MODNAM  START
                     54 52 41 54 53 00'     0000      $MODULE:          .ASCIC  \START\
                                    05      0000
                                    0006   303
                                    0006   304 FMT_PROG:                                                                ;  [25]
61 72 67 6F 72 50 20 2E 2E 2F 21 00'    0006   305          .ASCIC  "!/.. Program: !AC, revision !UL.!UL, !UL test!%S,!/"-  ;  [31]
69 76 65 72 20 2C 43 41 21 20 3A 6D     0012
4C 55 21 2E 4C 55 21 20 6E 6F 69 73     001E
25 21 74 73 65 74 20 4C 55 21 20 2C     002A
25 21 20 74 61 20 20 20 2F 21 2C 53     0036
                     2F 21 2E 54        0042
                                    3F      0006
                                    0046   306                  "   at !%T.!/"                                     ;  [31]
                                    0046   307
                                    0046   308 FMT_ABORT:                                                             ;  [25]
65 74 72 6F 62 41 20 2E 2E 2F 21 00'    0046   309          .ASCIC  \!/.. Aborted program at pass !UL, !AS, PC !XL.!/\      ;  [25]
74 61 20 6D 61 72 67 6F 72 70 20 64     0052
21 20 2C 4C 55 21 20 73 73 61 70 20     005E
21 2E 4C 58 21 20 43 50 20 2C 53 41     006A
                                    2F      0076
                                    30      0046
                                    0077   310
                                    0077   311 Fmt_Abort_No_PC:                                                       ;  [37]
65 74 72 6F 62 41 20 2E 2E 2F 21 00'    0077   312          .ASCIC  \!/.. Aborted program at pass !UL, !AS.!/\              ;  [37]
74 61 20 6D 61 72 67 6F 72 70 20 64     0083
21 20 2C 4C 55 21 20 73 73 61 70 20     008F
                     2F 21 2E 53 41     009B
                                    28      0077
                                    00A0   313
                                    00A0   314 FMTCTR:                                                                ;G:10
                        2F 21 00'   00A0   315          .ASCIC  \!/\                                                    ;  [42]
                                    02      00A0
                                    00A3   316
                                    00A3   317 FMTTRACE:                                                             ;  [42]
21 20 74 73 65 54 20 2E 2E 2F 21 00'    00A3   318          .ASCIC  \!/.. Test !UB: !AC \                                   ;  [42]
                  20 43 41 21 20 3A 42 55   00AF
                                    13      00A3
                                    00B7   319
                                    00B7   320 FMT_TIME_EXPIRED:
6C 20 65 6D 69 54 20 2E 2E 2F 21 00'    00B7   321          .ASCIC  "!/.. Time limit has expired at pass !UL, !AS, PC !XL.!/"; [40]
70 78 65 20 73 61 68 20 74 69 6D 69     00C3
73 73 61 70 20 74 61 20 64 65 72 69     00CF
50 20 2C 53 41 21 20 2C 4C 55 21 20     00DB
                  2F 21 2E 4C 58 21 20 43   00E7
                                    37      00B7
                                    00EF   322
                                    00EF   323 FMT_TIME_EXPIRED_NO_PC:
6C 20 65 6D 69 54 20 2E 2E 2F 21 00'    00EF   324          .ASCIC  "!/.. Time limit has expired at pass !UL, !AS.!/"       ;  [40]
70 78 65 20 73 61 68 20 74 69 6D 69     00FB
73 73 61 70 20 74 61 20 64 65 72 69     0107
2F 21 2E 53 41 21 20 2C 4C 55 21 20     0113
                                    2F      00EF
```

```
                                        011F   326 FMT_TESTRANGE:                                                    ;  [25]
21 20 79 6C 6E 4F 20 3F 3F 2F 21 00'    011F   327         .ASCIC  \!/?? Only !UL test!%S, check test range.!/\        ;  [25]
20 2C 53 25 21 74 73 65 74 20 4C 55     012B
72 20 74 73 65 74 20 6B 63 65 68 63     0137
                2F 21 2E 65 67 6E 61    0143
                                   2A   011F
                                        014A   328
                                        014A   329 FMT_NOUNITS:                                                      ;  [25]
73 74 69 6E 75 20 6F 4E 20 3F 3F 00'    014A   330         .ASCIC  "?? No units to test, none selected with device type(s)!/";[25]
6F 6E 20 2C 74 73 65 74 20 6F 74 20     0156
20 64 65 74 63 65 6C 65 73 20 65 6E     0162
20 65 63 69 76 65 64 20 68 74 69 77     016E
         2F 21 29 73 28 65 70 79 74     017A
                                   38   014A
                                        0183   331
                                        0183   332 FMT_START:                                                        ;  [25]
6F 72 70 20 6F 4E 20 3F 3F 2F 21 00'    0183   333         .ASCIC  \!/?? No program loaded.!/\                       ;  [25]
2E 64 65 64 61 6F 6C 20 6D 61 72 67     018F
                                2F 21   019B
                                   19   0183
                                        019D   334
                                        019D   335 FMT_ILLSEC:                                                       ;  [25]
61 67 65 6C 6C 49 20 3F 3F 2F 21 00'    019D   336         .ASCIC  \!/?? Illegal section name, should be one of!/\      ;  [25]
61 6E 20 6E 6F 69 74 63 65 73 20 6C     01A9
62 20 64 6C 75 6F 68 73 20 2C 65 6D     01B5
      2F 21 66 6F 20 65 6E 6F 20 65     01C1
                                   2D   019D
                                        01CB   337
                                        01CB   338 FMT_AMBSEC:                                                       ;  [25]
75 67 69 62 6D 41 20 3F 3F 2F 21 00'    01CB   339         .ASCIC  \!/?? Ambiguous section name, should be one of!/\    ;  [25]
20 6E 6F 69 74 63 65 73 20 73 75 6F     01D7
64 6C 75 6F 68 73 20 2C 65 6D 61 6E     01E3
2F 21 66 6F 20 65 6E 6F 20 65 62 20     01EF
                                   2F   01CB
                                        01FB   340
                                        01FB   341 FMT_SECTIONS:                                                     ;  [27]
6E 69 61 74 6E 6F 63 20 43 41 21 00'    01FB   342         .ASCIC  "!AC contains the following sections:!/"            ;  [27]
77 6F 6C 6C 6F 66 20 65 68 74 20 73     0207
73 6E 6F 69 74 63 65 73 20 67 6E 69     0213
                             2F 21 3A   021F
                                   26   01FB
                                        0222   343
                                        0222   344 FMT_TESTING:                                                      ;  [25]
      20 3A 67 6E 69 74 73 65 54 00'    0222   345         .ASCIC  \Testing: \                                  ;[25][48][50] ;G:5
                                   09   0222
                                        022C   346 FMT_NAMES:                                                                ;G:5
                      20 53 41 21 00'   022C   347         .ASCIC  \!AS \                                       ;[50]         ;G:5
                                   04   022C
                                        0231   348 FMT_CRLF:                                                                 ;G:5
                             2F 21 00'  0231   349         .ASCIC  \!/\                                         ;[50]         ;G:5
                                   02   0231
                                        0234   350
                                        0234   351 FMT_MISMATCH:                                             ;       [25]
76 65 72 20 73 69 68 54 20 3F 3F 00'    0234   352         .ASCIC  "?? This rev of diagnostic will not run with this"-  ;  [25]
74 73 6F 6E 67 61 69 64 20 66 6F 20     0240
20 74 6F 6E 20 6C 6C 69 77 20 63 69     024C
69 68 74 20 68 74 69 77 20 6E 75 72     0258
```

```
                                                 73  0264
21 72 6F 73 69 76 72 65 70 75 73                 20  0265    353                 " supervisor!/"                                 ;       [25]
                                                 2F  0271
                                                 3D  0234
                                                     0272    354
                                                     0272    355 FMT_NODEVICE:                                               ;       [22]
76 65 64 20 6F 4E 20 3F 3F 2F 21 00'             0272    356             .ASCIC  \!/?? No device support list in program.!/\     ;       [22]
20 74 72 6F 70 70 75 73 20 65 63 69              027E
67 6F 72 70 20 6E 69 20 74 73 69 6C              028A
                  2F 21 2E 6D 61 72              0296
                                                 29  0272
```

```
                     029C   358          .SBTTL  VRStart and VRReStart Routines
                00000000    359          .PSECT  Code, Shr, Exe, NoWrt, Byte
                    0000    360 ;++
                    0000    361 ; FUNCTIONAL DESCRIPTION:
                    0000    362 ;
                    0000    363 ;       This subroutine is called from CLI to start the program.  It verifies
                    0000    364 ;       command parameters and builds P-tables from the selected devices, and if
                    0000    365 ;       necessary, initializes the QIO database.
                    0000    366 ;
                    0000    367 ; CALLING SEQUENCE:
                    0000    368 ;
                    0000    369 ;       Subroutine call from the command line interpreter.
                    0000    370 ;
                    0000    371 ; INPUT PARAMETERS:
                    0000    372 ;
                    0000    373 ;       NONE
                    0000    374 ;
                    0000    375 ; IMPLICIT INPUTS:
                    0000    376 ;
                    0000    377 ;       R2              = BASE ADDRESS FOR CLI DATA BLOCK.
                    0000    378 ;       CLI$L_TEST(R2)  = FIRST TEST TO EXECUTE.
                    0000    379 ;       CLI$L_LAST(R2)  = LAST TEST TO EXECUTE.
                    0000    380 ;       CLI$L_PASS(R2)  = NUMBER OF PASSES TO RUN.
                    0000    381 ;       CLI$L_SUBT(R2)  = SUBTEST NUMBER FOR LOOP.
                    0000    382 ;       L$L_UNIT        = MAXIMUM NUMBER OF UNITS.
                    0000    383 ;
                    0000    384 ; OUTPUT PARAMETERS:    NONE
                    0000    385 ;
                    0000    386 ; IMPLICIT OUTPUTS:     NONE
                    0000    387 ;
                    0000    388 ; COMPLETION CODES:     NONE
                    0000    389 ;
                    0000    390 ; SIDE EFFECTS:
                    0000    391 ;
                    0000    392 ;       MAPFREE is called to initialize free memory Device for QIO may be
                    0000    393 ;       initialized
                    0000    394 ;
                    0000    395 ; REGISTER USAGE:
                    0000    396 ;
                    0000    397 ;       R2 = BASE ADDRESS FOR CLI DATA BLOCK.
                    0000    398 ;       R3 = UNIT COUNT.
                    0000    399 ;       R4 = SECTION NAME ADDRESS LIST POINTER.
                    0000    400 ;       R5 = SECTION NAME LIST COUNTER.
                    0000    401 ;       R6 = SECTION NAME ASCII POINTER.
                    0000    402 ;       R7 = INPUT CHARACTER COUNT.
                    0000    403 ;       R8 = INPUT ASCII POINTER.
                    0000    404 ;       R9 = Equivalent name flag                                   [38]
                    0000    405 ;       R10= Ambiguous name flag                                    [38]
                    0000    406 ;--
```

```
                                0000   408 ;+
                                0000   409 ;       This is the entry point for the CLI call. Check to see if a     [29]
                                0000   410 ;       program has been loaded.                                        [29]
                                0000   411 ;-
                                0000   412
                                0000   413 VRSTART::
                                0000   414             BR_IF_NOT_MP    5$              ; Branch if not an mp system    [49]  ;G:2
    05000000 8F   00000000'8F   D1   0000             CMPL    #DS$GK_SID,#^X05000000          ; Branch if Scorpio [61]
                        27      13   000B             BEQL    30000$                  ; Yes, branch
    06000000 8F   00000000'8F   D1   000D             CMPL    #DS$GK_SID,#^X06000000          ; Branch if Nautilus [61]
                        1A      13   0018             BEQL    30000$                  ; Yes, branch                   ;G:23
    11000000 8F   00000000'8F   D1   001A             CMPL    #DS$GK_SID,#^X11000000          ; Branch if RRR [69]            ;G:20
                        0D      13   0025             BEQL    30000$                                                  ;G:23
    0A000000 8F   00000000'8F   D1   0027             CMPL    #DS$GK_SID,#^X0A000000          ; Branch if LLL [69]            ;G:20
                        1E      12   0032             BNEQ    5$                                                      ;G:18
                                0034         30000$:
                00000000'EF     16   0034   415         JSB     MP$_CLEAR_LOCKS             ; Clear the Pp's intlk struc    [49]  ;G:2
                00000000'EF     D4   003A   416         CLRL    MP$GR_BOOTED_PROCESSORS     ; Clear mp flags                [49]  ;G:2
                00000000'EF     D4   0040   417         CLRL    MP$GR_IDLING_PROCESSORS     ;       "                       [49]  ;G:2
                00000000'EF     D4   0046   418         CLRL    MP$GL_FLAGS                 ;       "                       [49]  ;G:2
                00000000'EF     D4   004C   419         CLRL    MP$GL_CONDITION_FLAGS       ;       "                       [49]  ;G:2
            00000000'EF   01    E0   0052   420 5$:     BBS     #DS$V_LODFLG, -             ; CHECK LOAD FLAG.                      ;G:2
                        16          0059   421           L^DS$GL_FLAGS, 10$        ;                               [30]
                                    005A   422         $PRINT  #ds$k_type_command_err, -        ; Command error message [26]
                                    005A   423                 #ds$k_printf, -         ; .. use printf                 [26]
                                    005A   424                 L^FMT_START             ; "No program loaded"           [26]
                00000183'EF   9F    005A                PUSHAB  L^FMT_START
                    7E   01   9A    0060                movzbl  #ds$k_printf, -(sp)
                    7E   15   98    0063                cvtbl   #ds$k_type_command_err, -(sp)
            00000000'GF   03    FB  0066                CALLS   #$$N+2, G^DSX$PRINT
                      03B2      31  006D   425         BRW     VRSTART_X                   ; Exit routine                  [29]
                                    0070   426
                                    0070   427 10$:     Br_If_Apt       20$             ; APT sets this flag so don't check[36]
    0000FE00'EF   1F    E0          0070                BBS     #DSA$V_Apt, -                   ; If bit set, branch            [29]
                        14          0077                  L^DSA$GL_Flags, 20$           ;                               [29]
                        1E      E1  0078   428         BBC     #DSA$V_LOAD_DEBUGGER,-  ; it IF USER HAS TYPED "/DEBUG" [36]
                  0C 0000FE00'EF        007A   429             DSA$GL_FLAGS,20$        ;THEN                           [34]
                     00000000'EF  16    0080   430         JSB     ACT_DEFAULT_DBG         ;  GET DEFAULT FILENAME FOR DEBUGGER[34]
                     00000000'EF  16    0086   431         JSB     DSV$DEBUG               ;  LOAD AND START THE DEBUGGER  [34]
                     40000000 8F  CA    008C   432 20$:    BICL2   #DSA$M_LOAD_DEBUGGER,-  ; Set up for new /DEBUG command [36]
                     0000FE00'EF        0092   433             L^DSA$GL_FLAGS          ;                               [36]
                              0399      30   0097   434     BSBW    Check_QA_Bit            ; Set or clear QA DSA bit       [30]
                                        009A   435     QA_MAIN Start_VRStart           ; Call QA$Main                  [30]
                                16      DD   009A             PUSHL   #DSQA$K_Start_VRStart
                00000000'9F     01      FB   009C             CALLS   #1, a#QA$MAIN
```

```
                                  00A3   437 ;+
                                  00A3   438 ;       Check to make sure the diagnostic and this Supervisor match.    [29]
                                  00A3   439 ;       Also initialize context and call QA$Main.                       [29]
                                  00A3   440 ;-
                                  00A3   441
                                  00A3   442 VRRESTART::
                                  00A3   443 START1:
                        18   BB   00A3   444         Pushr   #^M<r3,r4>                ; --------------------          [46]
          53   00000000'EF   DE   00A5   445         MOVAL   DEVREV$T,R3               ;                               [46]
          54   00000000'EF   DE   00AC   446         MOVAL   DEVREV$T_X,R4             ; Clear the Data structure      [46]
                        63   94   00B3   447 5$:     CLRB    (R3)                      ; used by dsx$printrev          [46]
                FA 53   54   F2   00B5   448         AOBLSS  R4,R3,5$                  ;                               [46]
                        18   BA   00B9   449         Popr    #^M<r3,r4>                ; --------------------          [46]
                                  00BB   450
                00000000'EF   16   00BB   451         JSB     SCRIPT$FLUSH              ; Flush scripts if console cmd  [24]
  50   00000204 9F   08   02   EF   00C1   452         extzv   #env$v_super, -           ;                               [26]
                                  00CA   453                   #env$s_super, -         ;                               [26]
                                  00CA   454                   @#l$l_environ, r0       ; Fetch Diagnostic version      [26]
                    01   50   D1   00CA   455         cmpl    r0, #env$_super           ; Is it identical?              [26]
                        16   13   00CD   456         BEQL    10$                       ; Branch if valid
                                  00CF   457         $PRINT  #ds$k_type_start_err, -   ; Error starting diagnostic     [26]
                                  00CF   458                   #ds$k_printf, -         ; .. use PRINTF                 [26]
                                  00CF   459                   L^FMT_MISMATCH          ; Inform of mismatch            [26]
                00000234'EF   9F   00CF                       PUSHAB  L^FMT_MISMATCH
                      7E   01   9A   00D5                       movzbl  #ds$k_printf, -(sp)
                      7E   18   98   00D8                       cvtbl   #ds$k_type_start_err, -(sp)
          00000000'GF   03   FB   00DB                       CALLS   #$$N+2, G^DSX$PRINT
                      033D   31   00E2   460         BRW     VRSTART_X                 ; And Exit
                                  00E5   461
                00000000'EF   16   00E5   462 10$:    JSB     INIT_CONTEXT              ; Reset stacks and modes        [24]
                      0345   30   00EB   463         BSBW    Check_QA_Bit              ; Set or clear QA DSA bit. Note:[30]
                                  00EE   464                                           ; the QA bit was cleared by the [30]
                                  00EE   465                                           ; SCRIPT$FLUSH routine above.   [30]
                                  00EE   466         QA_MAIN Start_VRRestart           ; Call QA$Main                  [30]
                        12   DD   00EE               PUSHL   #DSQA$K_Start_VRRestart
          00000000'9F   01   FB   00F0               CALLS   #1, @#QA$MAIN
```

```
                                00F7   468 ;+
                                00F7   469 ;       Cleanup the previous diagnostic and QIO (if applicable).      [29]
                                00F7   470 ;-
                                00F7   471
          00000000'EF    16     00F7   472          JSB     DS_CLEANUP                ; DO CLEANUP IF NECESSARY      [24]
                                00FD   473
    0000FE00'EF    1C    E0     00FD   474          BBS     #DSA$V_USER, -
                   07           0104   475                  DSA$GL_FLAGS, 15$         ; SKIP QIO$CLEANUP IN USERMODE
    00000000'EF    00    FB     0105   476          CALLS   #0,QIO$CLEANUP            ; Remove traces of QIO database
```

```
                                010C   478 ;+
                                010C   479 ;       Check the requested section name.                                [29]
                                010C   480 ;-
                                010C   481
          52   00000000'EF   DE  010C   482 15$:   MOVAL   DS$GL_CLIBASE, R2        ; Adress the CLI vector.        [30]
               0000FE10'EF   D4  0113   483         CLRL    DSA$GL_SECTNO            ; INIT SECTION TO "DEFAULT".
                      10 A2  D5  0119   484         TSTL    CLI$Q_SECTION(R2)        ; CHECK FOR SECTION SWITCH.
                         03  12  011C   485         BNEQ    18$                      ; If found, continue            [26]
                       00A7  31  011E   486         brw     50$                      ; Branch                        [26]
                                0121   487
          54   00000250'EF   D0  0121   488 18$:   MOVL    L$A_SECNAM, R4           ; GET SECTION LIST ADDRESS.
                    55   84  D0  0128   489         MOVL    (R4)+, R5                ; GET SECTION COUNT.
                                012B   490
                         5A  D4  012B   491         CLRL    R10                      ; Ambiguous name Flag           [38]
                    56   84  D0  012D   492 20$:   MOVL    (R4)+, R6                ; GET SECTION NAME ASCII POINTER.
                         59  D4  0130   493         CLRL    R9                       ; Equivalent name Flag          [38]
                 57   10 A2  D0  0132   494         MOVL    CLI$Q_SECTION(R2), R7    ; GET REQUESTED SECTION ASCII CNT.
                 58   14 A2  D0  0136   495         MOVL    CLI$Q_SECTION+4(R2), R8  ; GET STRING POINTER.
                    57   66  91  013A   496         CMPB    (R6), R7                 ; CHECK LENGTHS.                [38]
                         57  19  013D   497         BLSS    40$                      ; BR IF INPUT TOO LONG.
                    57   86  91  013F   498         CMPB    (R6)+,R7                 ; Characters Equivalent         [38]
                         02  12  0142   499         BNEQ    30$                      ; No                            [38]
                         59  D6  0144   500         INCL    R9                       ; Yes                           [38]
                                0146   501
                    88   86  91  0146   502 30$:   CMPB    (R6)+, (R8)+             ; COMPARE CHARACTERS.
                         4B  12  0149   503         BNEQ    40$                      ; BR IF MISMATCH.
                      F8 57  F5  014B   504         SOBGTR  R7, 30$                  ; COUNT CHARACTERS.
0000FE10'EF   00000250'FF   55  C3  014E   505         SUBL3   R5, @L$A_SECNAM, -       ; COMPUTE SECTION NUMBER.
                                015A   506                   DSA$GL_SECTNO
                         59  D5  015A   507         TSTL    R9                       ; Equivalent name               [38]
                         07  13  015C   508         BEQL    32$                      ; No                            [38]
                 3B 62   16  E3  015E   509         BBCS    #CLI$V_VALSEC, -         ; SET  VALID SECTION FLG.
                                0162   510                   CLI$L_FLAGS(R2), 42$   ;                               [38]
                                0162   511                                            ; Set Valid Section Flag        [38]
                       0038  31  0162   512         BRW     42$                      ; All Done                      [38]
                                0165   513 32$:                                       ;                               [38]
                 2D 62   16  E3  0165   514         BBCS    #CLI$V_VALSEC, -         ; SET AND CHECK VALID SECTION FLG.
                                0169   515                   CLI$L_FLAGS(R2), 40$
                    5A   01  D0  0169   516         MOVL    #1, R10                  ; Indicate Ambiguous name       [38]
                       0027  31  016C   517         BRW     40$                      ; Continue with list            [38]
                                016F   518 35$:                                       ;                               [38]
                                016F   519         $PRINT  #-ds$k_type_command_err, -      ; command error (sticky)[26]
                                016F   520                   #ds$k_printf, -        ; .. use PRINTF                 [26]
                                016F   521                   L^FMT_AMBSEC           ; "?? Ambiguous section name."  [26]
               000001CB'EF   9F  016F                       PUSHAB  L^FMT_AMBSEC
                    7E   01  9A  0175                       movzbl  #ds$k_printf, -(sp)
                 7E   EB 8F  98  0178                       cvtbl   #-ds$k_type_command_err, -(sp)
          00000000'GF   03  FB  017C                       CALLS   #$$N+2, G^DSX$PRINT
          50   00000250 9F  D0  0183   522         MOVL    @#L$A_SECNAM,R0          ; Point to section names
                       02C3  30  018A   523         BSBW    PRLIST                   ; Print list
               00000000'EF   94  018D   524         clrb    L^ds$gb_typecode         ; Clear type code after done    [26]
                       028C  31  0193   525         BRW     VRSTART_X
```

```
                94 55   F5  0196   527 40$:    SOBGTR  R5, 20$                 ; COUNT SECTIONS.
                   5A   D5  0199   528          TSTL    R10                     ; Ambiguous Name found?          [38]
                   D2   12  019B   529          BNEQ    35$                     ; Yes                            [38]
                        019D   530 42$:                                         ;                                [38]
             27 62   16   E0  019D   531          BBS     #CLI$V_VALSEC, -        ; CHECK FOR A VALID SECTION.
                                                   CLI$L_FLAGS(R2), 50$
                        01A1   532
                        01A1   533          $PRINT  #-ds$k_type_command_err, -       ; Command error (sticky)[26]
                        01A1   534                    #ds$k_printf, -           ; .. use PRINTF                  [26]
                        01A1   535                    L^FMT_ILLSEC              ; "?? Illegal section name."     [26]
          0000019D'EF   9F  01A1                    PUSHAB  L^FMT_ILLSEC
            7E   01   9A  01A7                    movzbl  #ds$k_printf, -(sp)
         7E   EB 8F   98  01AA                    cvtbl   #-ds$k_type_command_err, -(sp)
00000000'GF   03   FB  01AE                    CALLS   #$$N+2, G^DSX$PRINT
50  00000250 9F   D0  01B5   536          MOVL    a#L$A_SECNAM,R0             ; Point to section names
              0291   30  01BC   537          BSBW    PRLIST                  ; Print section names
          00000000'EF   94  01BF   538          clrb    L^ds$gb_typecode        ; Clear type code after done    [26]
              025A   31  01C5   539          BRW     VRSTART_X               ; Exit                           [27]
```

```
                          01C8    541 ;+
                          01C8    542 ;       The section name checked out. Set the default flags.        [29]
                          01C8    543 ;-
                          01C8    544
                          01C8    545 50$:
                       CA 01C8    546         BICL2   # DS$M_DONFLG ! -       ; CLEAR CLEANUP DONE
                          01C9    547                   DS$M_STRFLG ! -       ; AND PROGRAM STARTED
                          01C9    548                   DS$M_HLTFLG ! -       ; AND HALTED ON ERROR
                          01C9    549                   DS$M_CTRLC  ! -       ; AND STOPPED BY ^C
                          01C9    550                   DS$M_BRKPT  ! -       ; AND STOPPED BY BREAKPOINT
                          01C9    551                   DS$M_EXCEPT   -       ; AND STOPPED BY EXCEPTION
00000000'EF   0008280D 8F         01C9    552                 , L^DS$GL_FLAGS         ; IN THE GLOBAL FLAGS       [24]
```

```
                                01D3    554 ;+
                                01D3    555 ;       Check the requested test numbers (i.e. TEST:x:y).                   [29]
                                01D3    556 ;-
                                01D3    557
          50   00000254'FF    D0  01D3    558           MOVL    @L$A_TSTCNT, R0             ; GET ACTUAL TEST COUNT.
      00000000'EF    50    D0  01DA    559           MOVL    R0,L^DS$GL_NUMTEST          ; SET HIGHEST TEST NUMBER       [24]
                     51    D4  01E1    560           CLRL    R1                          ; Address page zero
      0000FE08'EF    2C A2 D0  01E3    561           MOVL    CLI$L_PASS(R2), -           ; PASS COUNT
                                01EB    562                     DSA$GL_PASSES
      00000000'EF    20 A2 D0  01EB    563           MOVL    CLI$L_TEST(R2), -           ; FIRST TEST NUMBER             [24]
                                01F3    564                     L^DS$GL_FSTTEST
                     43    19  01F3    565           BLSS    95$                         ; MUST BE POSITIVE
                     06    12  01F5    566           BNEQ    80$                         ; SKIP IF NOT BLANK.
               00000000'EF    D6  01F7    567           INCL    L^DS$GL_FSTTEST             ; START AT TEST 1.              [24]
                                01FD    568
      00000000'EF    24 A2 D0  01FD    569 80$:      MOVL    CLI$L_LAST(R2), -           ; LAST TEST NUMBER.             [24]
                                0205    570                     L^DS$GL_LSTTEST             ;                               [24]
                     31    19  0205    571           BLSS    95$                         ; MUST BE POSITIVE
                     0B    12  0207    572           BNEQ    90$                         ; SKIP IF NOT 0.
  00000000'EF   00000000'EF    D0  0209    573           MOVL    L^DS$GL_NUMTEST, -          ; DEFAULT LAST TEST.
                                0214    574                     L^DS$GL_LSTTEST             ;                               [24]
                                0214    575
  00000000'EF   00000000'EF    D1  0214    576 90$:      CMPL    L^DS$GL_FSTTEST, -          ; FIRST TEST NUMBER IN RANGE?   [24]
                                021F    577                     L^DS$GL_NUMTEST             ;                               [24]
                     17    14  021F    578           BGTR    95$                         ; NO, ERROR AND EXIT TO COMMAND
  00000000'EF   00000000'EF    D1  0221    579           CMPL    L^DS$GL_LSTTEST, -          ; LAST TEST NUMBER IN RANGE?    [24]
                                022C    580                     L^DS$GL_NUMTEST             ;                               [24]
                     0A    14  022C    581           BGTR    95$                         ; NO, ERROR AND EXIT TO COMMAND
                                022E    582
  00000000'EF    28 A2 D0  022E    583           MOVL    CLI$L_SUBT(R2), -           ; SUBTEST NUMBER.               [24]
                                0236    584                     L^DS$GL_SUBTEST             ;                               [24]
                     1C    11  0236    585           BRB     100$                        ; CONTINUE
                                0238    586
                                0238    587 95$:      $PRINT  #ds$k_type_command_err, -       ; Command error         [26]
                                0238    588                     #ds$k_printf, -             ; .. use PRINTF             [26]
                                0238    589                     L^FMT_TESTRANGE, -          ; .. format                 [26]
                                0238    590                     L^DS$GL_NUMTEST             ; .. error in test numbers  [26]
               00000000'EF    DD  0238                        PUSHL   L^DS$GL_NUMTEST
               0000011F'EF    9F  023E                        PUSHAB  L^FMT_TESTRANGE
                     7E    01  9A  0244                        movzbl  #ds$k_printf, -(sp)
                     7E    15  98  0247                        cvtbl   #ds$k_type_command_err, -(sp)
          00000000'GF    04  FB  024A                        CALLS   #$$N+2, G^DSX$PRINT
                   01CE    31  0251    591           BRW     VRSTART_X                   ; EXIT
```

```
                                0254   593 ;+
                                0254   594 ;       Print the diagnostic header lines, except under special
                                0254   595 ;       circumstances.                                          [29]
                                0254   596 ;-
                                0254   597
                                0254   598 100$:   QA_MAIN Start_Before_Header     ; Call QA$Main                [30]
                     17   DD    0254              PUSHL   #DSQA$K_Start_Before_Header
        00000000'9F  01   FB    0256              CALLS   #1, @#QA$MAIN
                                025D   599
        0000FE00'EF  0F   E1    025D   600         BBC     #DSA$V_QA, -             ; If not running QA, branch.   [32]
                     1A         0264   601                   L^DSA$GL_FLAGS, 104$  ;                              [32]
                     03   E0    0265   602         BBS     #QA$K_LOOP_ON_SUBTEST,- ; If the Loop on Subtest check [32]
            00000000'EF         0267   603                   L^QA$AOB_CHECK_STATE,-; is executing, branch.        [32]
                     0A         026C   604                   103$                  ;                              [32]
                     05   E0    026D   605         BBS     #QA$K_ERROR_PHASE_ONE,- ; If the Error Phase One check [32]
            00000000'EF         026F   606                   L^QA$AOB_CHECK_STATE,-; is executing, branch.        [32]
                     02         0274   607                   103$                  ;                              [32]
                     08   11    0275   608         BRB     104$                    ; Otherwise, branch and print  [32]
                                0277   609                                         ; header.                      [32]
                                0277   610
        00000000'EF  06   E0    0277   611 103$:   BBS     #QA$K_NO_HEADER, -       ; If header should not be printed,[32]
                     27         027E   612                   L^QA$AOB_FLAGS, -      ; then branch around the print. [32]
                                027F   613                   105$                  ;                              [32]
                                027F   614
                                027F   615 104$:   $PRINT  #ds$k_type_program_start, -     ; Start program        [32]
                                027F   616                   #ds$k_printf, -        ; .. Use PRINTF                [26]
                                027F   617                   L^FMT_PROG, -          ; .. Format string             [26]
                                027F   618                   L$A_NAME(R1), -        ; .. Program name.             [26]
                                027F   619                   L$L_REV(R1), -         ; .. revision                  [26]
                                027F   620                   L$L_UPDATE(R1), -      ; .. update                    [26]
                                027F   621                   @l$a_tstcnt, -         ; .. number of tests           [26]
                                027F   622                   #0                     ; .. current time              [26]
                     00   DD    027F                      PUSHL   #0
            00000254'FF   DD    0281                      PUSHL   @l$a_tstcnt
                0210 C1   DD    0287                      PUSHL   L$L_UPDATE(R1)
                020C C1   DD    028B                      PUSHL   L$L_REV(R1)
                0208 C1   DD    028F                      PUSHL   L$A_NAME(R1)
            00000006'EF   9F    0293                      PUSHAB  L^FMT_PROG
                7E   01   9A    0299                      movzbl  #ds$k_printf, -(sp)
                7E   0F   98    029C                      cvtbl   #ds$k_type_program_start, -(sp)
        00000000'GF  08   FB    029F                      CALLS   #$$N+2, G^DSX$PRINT
                                02A6   623
                                02A6   624 105$:                                   ;                              [32]
```

```
                                02A6    626 ;+
                                02A6    627 ;       Make Ptables for all the units. Print what this diagnostic is   [29]
                                02A6    628 ;       testing.                                                        [29]
                                02A6    629 ;-
                                02A6    630
          0000FE0C'EF    D4     02A6    631         CLRL    DSA$GL_UNITS            ; Initialize unit counter
          00000220'EF    D5     02AC    632         TSTL    L$L_UNIT                ; Does program test units?
                  03     12     02B2    633         BNEQ    107$                    ; Yes, go generate Ptables for them.[32]
                 00AC    31     02B4    634         BRW     170$                    ; No, skip unit dependent setup.[32]
                                02B7    635
      00000000'EF    00     FB  02B7    636 107$:   CALLS   #0,DSP$GEN_PTABLES      ; Make P-tables for selected units
                                02BE    637
  54  0000FE0C'EF    01     C3  02BE    638         SUBL3   #1,DSA$GL_UNITS,R4      ; Highest unit number
                  27     18     02C6    639         BGEQ    110$                    ; Branch if one found           [29]
                                02C8    640
                                02C8    641         $PRINT  #-ds$k_type_start_err, -; Error starting                [26]
                                02C8    642                   #ds$k_printf, -       ; .. use PRINTF                 [26]
                                02C8    643                   L^FMT_NOUNITS         ; .. no units to test text      [26]
          0000014A'EF    9F     02C8                        PUSHAB  L^FMT_NOUNITS
             7E    01    9A     02CE                        movzbl  #ds$k_printf, -(sp)
           7E    E8 8F    98     02D1                        cvtbl   #-ds$k_type_start_err, -(sp)
      00000000'GF    03     FB  02D5                        CALLS   #$$N+2, G^DSX$PRINT
      50  0000021C 9F     D0     02DC    644         MOVL    a#L$A_DEVP,R0           ; Get address of DEVTYP list
                 016A    30     02E3    645         BSBW    PRLIST                  ; Print device types            [29]
          00000000'EF    94     02E6    646         clrb    L^ds$gb_typecode        ; Clear sticky type code        [26]
                 0133    31     02EC    647         BRW     VRSTART_X               ; RETURN TO COMMAND MODE
```

```
                                02EF    649 ;+
                                02EF    650 ; If QA is running, AND if either the Loop on Subtest check or the     [32]
                                02EF    651 ; Error Phase One check is executing, do not print the "Testing: ..."  [32]
                                02EF    652 ; message unless the No_Header bit is False. This stops the message    [32]
                                02EF    653 ; being printed too many times.                                        [32]
                                02EF    654 ;-
                                02EF    655
          0000FE00'EF    0F E1  02EF    656 110$:   BBC     #DSA$V_QA, -              ; If not running QA, branch.    [32]
                         1A     02F6    657                  L^DSA$GL_FLAGS, 120$   ;                               [32]
                         03 E0  02F7    658         BBS     #QA$K_LOOP_ON_SUBTEST,- ; If the Loop on Subtest check is [32]
                00000000'EF     02F9    659                  L^QA$AOB_CHECK_STATE,-; executing, branch.             [32]
                         0A     02FE    660                  115$                   ;                               [32]
                         05 E0  02FF    661         BBS     #QA$K_ERROR_PHASE_ONE,- ; If the Error Phase One check is [32]
                00000000'EF     0301    662                  L^QA$AOB_CHECK_STATE,-; executing, branch.             [32]
                         02     0306    663                  115$                   ;                               [32]
                         08 11  0307    664         BRB     120$                    ; Otherwise, branch and print   [32]
                                0309    665                                         ; message.                      [32]
                                0309    666
          00000000'EF    06 E0  0309    667 115$:   BBS     #QA$K_NO_HEADER, -        ; If header should not be printed, [32]
                         52     0310    668                  L^QA$AOB_FLAGS, -      ; then branch around the print. [32]
                                0311    669                  130$                   ;                               [32]
                                0311    670
                                0311    671 120$:   $DS_GPHARD_S    R4,-(SP)        ; Get P-table address           [32]
                         7E DF  0311                PUSHAL  -(SP)
                         54 DD  0313                PUSHL   R4
          00000000'9F    02 FB  0315                CALLS   #2, a#DS$GPHARD
                      F2 54 F4  031C    672         SOBGEQ  R4, 120$                ; Decrement and branch if more  [32]
                                031F    673
                   50    03 9A  031F    674         MOVZBL  #3,R0                   ; Total parameter count         [26][50] ;G:
                00000222'EF    9F  0322    675         PUSHAB  L^FMT_TESTING           ; ASCIC EDIT STRING             [24]
                         01    DD  0328    676         pushl   #ds$k_printf            ; Use printf                    [26]
                         17    DD  032A    677         pushl   #ds$k_type_program_info ; Program info type code        [26]
          00000000'GF    50    FB  032C    678         CALLS   R0, G^DSX$PRINT         ; TYPE IT                       [26]
54        0000FE0C'EF    01    C3  0333    679         SUBL3   #1,DSA$GL_UNITS,R4      ; Number of names to print      [51]  ;G:6
                   50    04    9A  033B    680 125$:   MOVZBL  #4,R0                   ; Total parameter count         [51]  ;G:6
                0000022C'EF    9F  033E    681         pushab  I^FMT_NAMES             ; Name control string           [50]  ;G:5
                         01    DD  0344    682         pushl   #ds$k_printf            ; Use printf                    [50]  ;G:5
                         17    DD  0346    683         pushl   #ds$k_type_program_info ; Program info type code        [50]  ;G:5
          00000000'GF    50    FB  0348    684         CALLS   R0, G^DSX$PRINT         ; TYPE IT                       [50]  ;G:5
                      E9 54 F4  034F    685         SOBGEQ  R4,125$                 ; Decrement and branch if more  [50]  ;G:5
                00000231'EF    9F  0352    686         pushab  I^FMT_CRLF              ; Terminate line control string [50]  ;G:5
                         01    DD  0358    687         pushl   #ds$k_printf            ; Use printf                    [50]  ;G:5
                         17    DD  035A    688         pushl   #ds$k_type_program_info ; Program info type code        [50]  ;G:5
          00000000'GF    03    FB  035C    689         CALLS   #3, G^DSX$PRINT         ; TYPE IT                       [50]  ;G:6
                                0363    690 130$:                                   ;                               [32]
```

```
                                  0363   692 ;+                                                                                  ;G:10
                                  0363   693 ;       Add to the QIO database.          ;                                  [29]
                                  0363   694 ;-
                                  0363   695
          0000FE00'EF   1C   E0   0363   696 170$:   BBS     #DSA$V_USER, -             ; BRANCH IF USER MODE
                        57        036A   697                  DSA$GL_FLAGS, 180$
          00000000'EF   00   FB   036B   698         CALLS   #0,L^MAPFREE               ; Map free memory                  [24]
          0000FE00'EF   1A   E1   0372   699         BBC     #DSA$V_DEBUG, -            ; IF DEBUGGER IS RESIDENT          [34]
                        06        0379   700                  DSA$GL_FLAGS,172$          ; THEN                             [34]
               00000000'EF   16   037A   701         JSB     MAP_DEBUGGER               ;    MAP DEBUGGER'S MEMORY SPACE[34]
                                  0380   702 172$:                                      ;                                  [34]
                                  0380   703         $DS_CLRVEC_S #SCB$L_TIMER          ; Reset timer vector
               000000C0 8F   DD   0380                      PUSHL   #SCB$L_TIMER
          00000000'9F   01   FB   0386                      CALLS   #1, a#DS$CLRVEC
                                  038D   704
   02   00000204'EF   02   00   ED   038D   705         CMPZV   #0,#2,L$L_ENVIRON, -
                                  0396   706                  #SEP_FUNCTIONAL            ; System and Functional?
                        2A   12   0396   707         BNEQ    180$                       ; Branch if should not add unit to QIO
                        54   D4   0398   708         CLRL    R4                         ; Start with unit zero
                                  039A   709
                                  039A   710 175$:   $DS_GPHARD_S    R4,-(SP)           ; Get P-table address
                        7E   DF   039A                      PUSHAL  -(SP)
                        54   DD   039C                      PUSHL   R4
          00000000'9F   02   FB   039E                      CALLS   #2, a#DS$GPHARD
                    51   8E   D0   03A5   711         MOVL    (SP)+,R1                   ; P-table address
                    17 50   E9   03A8   712         BLBC    R0,180$                    ; Exit if error
                        51   DD   03AB   713         PUSHL   R1                         ; Push for addunit
          00000000'EF   01   FB   03AD   714         CALLS   #1,QIO$ADDUNIT             ; Add this unit to QIO database
                    03 50   E8   03B4   715         BLBS    R0,177$                    ;                                  [50]  ;G:7
                     0068   31   03B7   716         BRW     VRSTART_X                  ; Return to CLI if errors                ;G:7
       D8 54   0000FE0C'EF   F2   03BA   717 177$:   AOBLSS  DSA$GL_UNITS, R4, 175$     ; Count and continue if more units       ;G:7
```

```
                                03C2    719 ;+
                                03C2    720 ;       Reinitialize the stacks, set the started flag, dispatch the     [29]
                                03C2    721 ;       tests, and go back to command mode. Note: the DISPATCH routine  [29]
                                03C2    722 ;       will return to here only if the EndPass routine is NOT done.    [29]
                                03C2    723 ;       Just before call to DISPATCH, do a GET_TIME if a TIME switch was[52]   ;G:8
                                03C2    724 ;       given on the START or RUN command.                                    ;G:7
                                03C2    725 ;       Load up BEGIN_TIME with the time at which we did the             [52]  ;G:8
                                03C2    726 ;       GET_TIME, in case the user does a ^C and a                       [52]  ;G:8
                                03C2    727 ;       CONTINUE, we can figure out how much more to run.                [40]  ;G:8
                                03C2    728 ;-
                                03C2    729
            00000000'EF   16    03C2    730 180$:   JSB     INIT_CONTEXT                ; REINITIALIZE STACKS ETC.       [24]  ;G:7
        00000000'EF    04   C8  03C8    731         BISL    #DS$M_STRFLG, -             ; SET PROGRAM STARTED            [24]
                                03CF    732           L^DS$GL_FLAGS                     ;                                [24]
00000000'EF   00000000'EF   7D  03CF    733         MOVQ    L^BIN_TIME,L^BIN_TIME       ; /TIME given? (test for zero)   [40]
                      03    12  03DA    734         BNEQ    182$                        ;                                [52]  ;G:7
                   003D     31  03DC    735         BRW     190$                        ; Branch if not                  [40]  ;G:7
                                03DF    736                                                                                ;G:7
                      03    BB  03DF    737 182$:   PUSHR   #^M<R0,R1>                  ; Used by $GETTIM                [52]  ;G:7
            00000008'EF   DF  03E1    738         PUSHAL  BEGIN_TIME                  ; Address to store result        [52][53] ;G:
        00000000'EF    01   FB  03E7    739         CALLS   #1,GET_TIME                 ; Use TODR to calculate time     [52]  ;G:7
                      03    BA  03EE    740         POPR    #^M<R0,R1>                  ; Used by $GETTIM                [52]  ;G:7
                                03F0    741                                                                                ;G:7
00000018'EF   00000008'EF   7D  03F0    742         MOVQ    L^BEGIN_TIME,L^DUE_TIME     ; Initialize DUE_TIME            [52]  ;G:7
00000018'EF   00000000'EF   C2  03FB    743         SUBL2   L^BIN_TIME,L^DUE_TIME       ; Low order                      [52]  ;G:7
0000001C'EF   00000004'EF   D9  0406    744         SBWC    L^BIN_TIME+4,L^DUE_TIME+4   ; High order                     [52]  ;G:7
00000000'EF   02000000 8F   C8  0411    745         BISL2   #DS$M_TIMED,DS$GL_FLAGS     ; This run is TIMED.             [52]  ;G:7
                                041C    746
            00000000'EF   16    041C    747 190$:   JSB     DISPATCH                    ; CALL PROGRAM                   [24]
                                0422    748
```

```
                                0422    750 ;+
                                0422    751 ;       End of the VRSTART routine. Return indirectly to the CLI routine.[29]
                                0422    752 ;-
                                0422    753
                                0422    754 VRSTART_X:
00000000'EF   02000000 8F   CA  0422    755             BICL2   #DS$M_TIMED,DS$GL_FLAGS ; Clear TIMED flag.            [52]  ;G:7
              00000000'EF   17  042D    756             JMP     BEGIN                   ; Return to command decoder     [24]  ;G:7
```

```
                        0433   758 ;+
                        0433   759 ;       Routine to turn the QA flag on or off depending on whether      [29]  ;G:7
                        0433   760 ;       or not the CLI QA flag was set. This routine destroys R0.       [30]  ;G:7
                        0433   761 ;       Usage:                                                          [30]  ;G:7
                        0433   762 ;               BSBB    Check_QA_Bit                                    [30]  ;G:7
                        0433   763 ;               BSBW    Check_QA_Bit                                    [30]  ;G:7
                        0433   764 ;               JSB     Check_QA_Bit                                    [30]  ;G:7
                        0433   765 ;-                                                                             ;G:10
                        0433   766 Check_QA_Bit:
50   00000000'EF   DE   0433   767           MOVAL   DS$GL_CLIBASE, R0       ; Readdress CLI command area    [30]  ;G:7
      09 60   07   E1   043A   768           BBC     #CLI$V_QA, -            ; If bit clear, go clear DSA bit[23]  ;G:7
                        043E   769             CLI$L_FLAGS (R0), 20$ ;                                       [30]  ;G:7
0000FE00'EF   0F   E3   043E   770           BBCS    #DSA$V_QA, -            ; Set DSA QA bit                [23]  ;G:7
              00        0445   771             DSA$GL_FLAGS, 10$     ;                                       [30]  ;G:7
                        0446   772                                                                                ;G:10
                   05   0446   773 10$:      RSB                             ; Return to caller              [30]  ;G:7
                        0447   774
0000FE00'EF   0F   E5   0447   775 20$:      BBCC    #DSA$V_QA, -            ; Clear DSA QA bit              [30]  ;G:7
              00        044E   776             DSA$GL_FLAGS, 30$     ;                                       [30]  ;G:7
                        044F   777                                   ;                                       [23]  ;G:7
                        044F   778                                                                                ;G:10
                   05   044F   779 30$:      RSB                             ; Return to caller               [30]
```

```
                        0450   781 ;+
                        0450   782 ; This routine is used to print a list of ascic NAMES. R0= address of list.
                        0450   783 ; Usage:
                        0450   784 ;       JSB     PRLIST
                        0450   785 ;       BSBW    PRLIST
                        0450   786 ;       BSBB    PRLIST
                        0450   787 ;-
                        0450   788
                        0450   789 PRLIST:
          51    60  D0  0450   790         MOVL    (R0),R1                 ; Get number of entries in list
       7E   2F21 8F  B0  0453   791         MOVW    #^A"!/",-(SP)           ; Create edit string starting with "!/"
             05   11  0458   792         BRB     20$                     ; Skip first ","
       7E   202C 8F  B0  045A   793 10$:   MOVW    #^A", ",-(SP)           ; append ", "
       7E   4341 8F  B0  045F   794 20$:   MOVW    #^A"AC",-(SP)           ; Append "AC"
       7E     21 90  0464   795         MOVB    #^A"!",-(SP)            ; Append "!"
         F0 51  F5  0467   796         SOBGTR  R1,10$                  ; one for each parameter in list
    51   60   05  C5  046A   797         MULL3   #5,(R0),R1              ; Get count back
       7E   51  90  046E   798         MOVB    R1,-(SP)                ; Make edit string ASCIC
       51   60  D0  0471   799         MOVL    (R0),R1                 ; Get count back
          6041  DD  0474   800 30$:    PUSHL   (R0)[R1]                ; Push address of argument
        FA 51  F5  0477   801         SOBGTR  R1,30$                  ; Each address now on stack
       51   60  D0  047A   802         MOVL    (R0),R1                 ; Get count back
          6E41  DF  047D   803         PUSHAL  (SP)[R1]                ; Address of ASCIC string
       51   03  C0  0480   804         ADDL2   #3,R1                   ; Total count of print args     [33]
       7E   01  9A  0483   805         movzbl  #ds$k_printf,-(sp)      ; Use PRINTF                    [33]
       7E   25  98  0486   806         cvtbl   #ds$k_type_start_list,-(sp)     ; Code                  [33]
00000000'GF   51  FB  0489   807         CALLS   R1, G^DSX$PRINT         ; Print the text                [33]
       50   8E  9A  0490   808         MOVZBL  (SP)+,R0                ; Get length of edit string
       5E   50  C0  0493   809         ADDL    R0,SP                   ; Remove edit string from stack
                05  0496   810         RSB                             ; Return
```

```
0497    812          .SBTTL  VRAbort and DSX$Abort Routines
0497    813 ;++
0497    814 ; FUNCTIONAL DESCRIPTION:
0497    815 ;
0497    816 ;       ABORTS CURRENT USER PROGRAM AFTER ERROR HALT, ^C, BREAKPOINT, OR
0497    817 ;       EXCEPTION, OR /TIME HAS EXPIRED.                                    [40]
0497    818 ;
0497    819 ; CALLING SEQUENCE:
0497    820 ;
0497    821 ;       BSBW    VRABORT
0497    822 ; -or-  CALLG   (SP),DS$ABORT
0497    823 ; -or-  from DSI$TIME_AST, fix up the argument list to return to VRABORT [40]
0497    824 ;
0497    825 ; INPUT PARAMETERS:
0497    826 ;
0497    827 ;       NONE
0497    828 ;
0497    829 ; IMPLICIT INPUTS:
0497    830 ;
0497    831 ;       (AP)    - USER REGISTER LIST.
0497    832 ;       DS$GL_FLAGS:
0497    833 ;               DS$V_BRKPT      - BREAKPOINT FLAG.
0497    834 ;               DS$V_EXCEPT     - EXCEPTION FLAG.
0497    835 ;               DS$V_HLTFLG     - ERROR HALT FLAG.
0497    836 ;               DS$V_CMDFLG     - COMMAND MODE FLAG.
0497    837 ;               DS$V_DONFLG     - PROGRAM COMPLETE FLAG.
0497    838 ;
0497    839 ; OUTPUT PARAMETERS:    NONE
0497    840 ;
0497    841 ; IMPLICIT OUTPUTS:     NONE
0497    842 ;
0497    843 ; COMPLETION CODES:     NONE
0497    844 ;
0497    845 ; SIDE EFFECTS:
0497    846 ;
0497    847 ;       ALL USER PROGRAM CONTEXT CLEARED.
0497    848 ;--
```

```
                          0000  0497   850 .ENTRY  DSX$ABORT,^M<>
                                0499   851
          0000FE00'EF      0F    E1    0499   852          BBC     #DSA$V_QA, -                ; If not running QA, branch.      [32]
                           16          04A0   853            L^DSA$GL_FLAGS, 10$       ;                                 [32]
                           05    E1    04A1   854          BBC     #QA$K_ERROR_PHASE_ONE, -; If not running the Error Phase [32]
                                       04A3   855            L^QA$AOB_CHECK_STATE,-; One check, branch.               [32]
            0E 00000000'EF             04A3   856            10$                       ;                                 [32]
                           CA    04A9   857          BICL2   #DS$M_HLTFLG ! -            ; Clear halt on error.            [32]
                                       04AA   858           DS$M_BRKPT  ! -            ; Clear "breakpoint" flag.        [32]
                                       04AA   859           DS$M_EXCEPT   -            ; Clear "exception" flag.         [32]
    00000000'EF   00080808 8F          04AA   860          ,L^DS$GL_FLAGS              ;                                 [32]
                      FBEC   31        04B4   861          BRW     VRRESTART           ; Go back and Restart the diag    [32]
                                       04B7   862
                50   10 AD   D0        04B7   863 10$:     MOVL    16(FP),R0           ; Get callers PC                  [32]
                        04   11        04BB   864          BRB     IS_IT_USER_PC       ; Is this really a user PC?       [37]
                                       04BD   865
                                       04BD   866 VRABORT::
                50   3C AC   D0        04BD   867          MOVL    60(AP),R0           ; Get PC from CLI args
                                       04C1   868
                                       04C1   869          ;+
                                       04C1   870          ; Now, check to make sure this is a user's PC (i.e., less than   [37]
                                       04C1   871          ; %X'10000'). If R0 is GTR 10000, then look for the user's PC on [37]
                                       04C1   872          ; the call frame stack.                                          [37]
                                       04C1   873          ;-
                                       04C1   874 DSI$ABORT::                          ; Entry point for /TIME abort     [43]
                                       04C1   875 IS_IT_USER_PC:                       ; Is this really a user PC?       [37]
          00010000'EF      50    D1    04C1   876          CMPL    R0, ^X00010000      ; Compare the start of VDS code   [37]
                           07    1B    04C8   877          BLEQU   VRABORT_COM         ; If < 10000, it's user PC        [41]
          00000000'EF      00    FB    04CA   878          CALLS   #0, L^DSR$FIND_USER_PC  ; Search on the stack for it  [37]
                                       04D1   879
                                       04D1   880 VRABORT_COM:
                                       04D1   881          ;+
                                       04D1   882          ; R0 now contains either a user PC or 0.                         [37]
                                       04D1   883          ;-
                           01    BB    04D1   884          PUSHR   #^MR0               ; Save PC at error for PRINTF below
                                       04D3   885
                00000000'EF      16    04D3   886          JSB     SCRIPT$FLUSH        ; Flush scripts if console command[24]
                                 CA    04D9   887          BICL2   # DS$M_HLTFLG ! -   ; CLEAR HALT ON ERROR.
                                       04DA   888            DS$M_BRKPT  ! -         ; CLEAR "BREAKPOINT" FLAG.
                                       04DA   889            DS$M_EXCEPT   -         ; CLEAR "EXCEPTION" FLAG.
    00000000'EF   00080808 8F          04DA   890          , L^DS$GL_FLAGS             ;                                 [24]
          00000000'EF      02    E0    04E4   891          BBS     #DS$V_STRFLG, -     ; Started?                        [24]
                           03          04EB   892            L^DS$GL_FLAGS,5$        ;                                 [24]
                      00C6   31        04EC   893          BRW     100$                ; Branch if not started           [40]
                00000000'EF      16    04EF   894 5$:     JSB     DS_TESTID           ; GET TEST/SUBTEST STRING.        [35]
                                       04F5   895
                           01    BA    04F5   896          POPR    #^MR0               ; Restore PC at error
                     50    00    D1    04F7   897          CmpL    #0, R0              ; Did we find a real user PC?     [37]
                           4E    13    04FA   898          Beql    15$                 ; Branch if we did not            [37]
          00000000'EF      1B    E4    04FC   899          BBSC    #DS$V_TIMED_OUT, -  ; ABORT or /TIME ran out?        [52] ;G:7
                           23          0503   900            L^DS$GL_FLAGS,10$       ;                                 [40]
                                       0504   901
                                       0504   902          $PRINT  #DS$K_TYPE_ABORT_PROGRAM, -; Abort program message   [26]
                                       0504   903                    #DS$K_PRINTF, -   ; ... use PrintF                [26]
                                       0504   904                    L^FMT_ABORT, -    ; ... abort message             [26]
                                       0504   905                    DSA$GL_PASSNO, -  ; ... current pass number      [37]
                                       0504   906                    #DS$GQ_TESTID, -  ; ... test id                  [37]
```

```
                                0504   907                      R0                          ; ... user PC                   [37]
                     50   DD    0504                      PUSHL   R0
          00000000'8F     DD    0506                      PUSHL   #DS$GQ_TESTID
          0000FE54'EF     DD    050C                      PUSHL   DSA$GL_PASSNO
          00000046'EF     9F    0512                      PUSHAB  L^FMT_ABORT
              7E   01     9A    0518                      movzbl  #DS$K_PRINTF, -(sp)
              7E   14     98    051B                      cvtbl   #DS$K_TYPE_ABORT_PROGRAM, -(sp)
     00000000'GF   06     FB    051E                      CALLS   #$$N+2, G^DSX$PRINT
                   6B     11    0525   908          BRB     25$                         ; Do rest of abort              [37]
                                0527   909
                                0527   910 10$:     $PRINT  #DS$K_TYPE_PROGRAM_END, - ; CRD should never see this.[40][54] ;G:9
                                0527   911                  #DS$K_PRINTF, -             ; PRINTF                        [40]
                                0527   912                  L^FMT_TIME_EXPIRED, -       ; the message                   [40]
                                0527   913                  DSA$GL_PASSNO, -            ; current pass number           [40]
                                0527   914                  #DS$GQ_TESTID, -            ; test and subtest              [40]
                                0527   915                  R0
                     50   DD    0527                      PUSHL   R0
          00000000'8F     DD    0529                      PUSHL   #DS$GQ_TESTID
          0000FE54'EF     DD    052F                      PUSHL   DSA$GL_PASSNO
          000000B7'EF     9F    0535                      PUSHAB  L^FMT_TIME_EXPIRED
              7E   01     9A    053B                      movzbl  #DS$K_PRINTF, -(sp)
              7E   10     98    053E                      cvtbl   #DS$K_TYPE_PROGRAM_END, -(sp)
     00000000'GF   06     FB    0541                      CALLS   #$$N+2, G^DSX$PRINT
                   48     11    0548   916          BRB     25$                         ; Do rest of /TIME abort        [40]
                                054A   917
     00000000'EF   1B     E4    054A   918 15$:     BBSC    #DS$V_TIMED_OUT, -          ; ABORT or /TIME ran out?       [52] ;G:7
                   21           0551   919                  L^DS$GL_FLAGS,20$           ;                               [40]
                                0552   920
                                0552   921          $Print  #DS$K_Type_Abort_Program, -; Abort program typecode      [37]
                                0552   922                    #DS$K_PrintF, -         ; ... use PrintF                [37]
                                0552   923                    L^Fmt_Abort_No_PC, -    ; ... abort message             [37]
                                0552   924                    DSA$GL_PassNo, -        ; ... the current pass number  [37]
                                0552   925                    #DS$GQ_TestId           ; ... test/subtest              [37]
          00000000'8F     DD    0552                      PUSHL   #DS$GQ_TestId
          0000FE54'EF     DD    0558                      PUSHL   DSA$GL_PassNo
          00000077'EF     9F    055E                      PUSHAB  L^Fmt_Abort_No_PC
              7E   01     9A    0564                      movzbl  #DS$K_PrintF, -(sp)
              7E   14     98    0567                      cvtbl   #DS$K_Type_Abort_Program, -(sp)
     00000000'GF   05     FB    056A                      CALLS   #$$N+2, G^DSX$PRINT
                   1F     11    0571   926          BRB     25$                         ; Finish the ABORT              [40]
                                0573   927
                                0573   928 20$:     $PRINT  #DS$K_TYPE_PROGRAM_END, - ; CRD should never see this.[40][54] ;G:9
                                0573   929                  #DS$K_PRINTF, -             ; PRINTF                        [40]
                                0573   930                  L^FMT_TIME_EXPIRED_NO_PC, -     ; the message           [40]
                                0573   931                  DSA$GL_PASSNO, -            ; current pass number           [40]
                                0573   932                  #DS$GQ_TESTID               ; test and subtest              [40]
          00000000'8F     DD    0573                      PUSHL   #DS$GQ_TESTID
          0000FE54'EF     DD    0579                      PUSHL   DSA$GL_PASSNO
          000000EF'EF     9F    057F                      PUSHAB  L^FMT_TIME_EXPIRED_NO_PC
              7E   01     9A    0585                      movzbl  #DS$K_PRINTF, -(sp)
              7E   10     98    0588                      cvtbl   #DS$K_TYPE_PROGRAM_END, -(sp)
     00000000'GF   05     FB    058B                      CALLS   #$$N+2, G^DSX$PRINT
                                0592   933
     0000FE00'EF   1F     E1    0592   934 25$:     Bbc     #DSA$V_Apt, -               ; Clean up if not in APT mode   [37]
                   08           0599   935                   DSA$GL_Flags, 50$          ;                               [21]
     00000000'EF   04     E0    059A   936          Bbs     #DS$V_ErrFlg, -             ; Skip cleanup if APT mode      [21]
                   0C           05A1   937                   DS$GL_Flags, 75$          ; ... and there was an error   [21]
```

```
                          05A2   938
     00000000'EF    16    05A2   939 50$:    Jsb     Init_Context            ; Reset stacks and all          [24]
     00000000'EF    16    05A8   940         Jsb     DS_Cleanup              ; Do user cleanup               [24]
                          05AE   941
00000000'EF    04    CA    05AE   942 75$:    BicL    #DS$M_StrFlg, -         ; Clear start flag              [21]
                          05B5   943                  DS$GL_Flags             ;                               [21]
                          05B5   944
0000FE40'EF    06    D0    05B5   945 100$:   MOVL    #APM$_ABTDON, -
                          05BC   946                  DSA$GL_MSGTYP           ; YES- INDICATE ABORT DONE
     00000000'EF    17    05BC   947         JMP     BEGIN                   ; TRANSFER TO SUPERVISOR CLEANUP. [24]
```

```
05C2    949          .SBTTL  VRSupport Routines
05C2    950 ;++
05C2    951 ; FUNCTIONAL DESCRIPTION:
05C2    952 ;
05C2    953 ;       This subroutine is called from CLI to show the devices a diagnostic
05C2    954 ;       supports.  It will issue messages for no diagnostics loaded, no
05C2    955 ;       devices supported.
05C2    956 ;
05C2    957 ; CALLING SEQUENCE:
05C2    958 ;
05C2    959 ;       SUBROUTINE CALL FROM THE COMMAND LINE INTERPRETER.
05C2    960 ;
05C2    961 ; INPUT PARAMETERS:
05C2    962 ;
05C2    963 ;       NONE
05C2    964 ;
05C2    965 ; IMPLICIT INPUTS:
05C2    966 ;
05C2    967 ;       NONE
05C2    968 ;
05C2    969 ; OUTPUT PARAMETERS:    NONE
05C2    970 ;
05C2    971 ; IMPLICIT OUTPUTS:     NONE
05C2    972 ;
05C2    973 ; COMPLETION CODES:     NONE
05C2    974 ;
05C2    975 ; SIDE EFFECTS:         NONE
05C2    976 ;
05C2    977 ;--
```

```
                                 05C2   979 VRSUPPORT::                                                    ;                                                  [22]
          00000000'EF   01   E0  05C2   980          BBS     #DS$V_LODFLG, -                               ; Check load flag.                                 [22]
                        14       05C9   981             L^DS$GL_FLAGS, 10$                                 ;                                                  [22]
                                 05CA   982 5$:      $PRINT  #ds$k_type_command_err,-                      ; Command error                                    [26]
                                 05CA   983                    #ds$k_printf, -                         ; .. use PRINTF                                    [26]
                                 05CA   984                    L^FMT_START                             ;                                                  [22]
                00000183'EF   9F   05CA                       PUSHAB  L^FMT_START
                     7E   01   9A  05D0                       movzbl  #ds$k_printf, -(sp)
                     7E   15   98  05D3                       cvtbl   #ds$k_type_command_err, -(sp)
          00000000'GF   03   FB  05D6                       CALLS   #$$N+2, G^DSX$PRINT
                                 05DD   985          RSB                                                   ; Return to caller.                                [22]
                                 05DE   986
01   00000204 9F   08   02   ED  05DE   987 10$:     cmpzv   #env$v_super, -                               ; Make sure a diagnostic is really                 [26]
                                 05E7   988                    #env$s_super, -                         ; .. loaded, and has the correct                   [26]
                                 05E7   989                    a#l$l_environ, -                        ; .. version for this                              [26]
                                 05E7   990                    #env$_super                             ; .. Supervisor                                    [26]
                        E1   12  05E7   991          bneq    5$                                            ; If not, error                                    [26]
          50    0000021C 9F   D0  05E9   992          MOVL    a#L$A_DEVP,R0                                 ; Get address of DEVTYP list                       [22]
                     51   60   D0  05F0   993          MOVL    (R0),R1                                       ; Get value pointed to                             [22]
                          51   D5  05F3   994          TSTL    R1                                            ; Check if device list exists                      [22]
                          14   12  05F5   995          BNEQ    20$                                           ; Branch if list exists                            [22]
                                 05F7   996
                                 05F7   997          $PRINT  #ds$k_type_command_out,-                      ; Command output (no devices)                      [26]
                                 05F7   998                    #ds$k_printf, -                         ; .. use PRINTF                                    [26]
                                 05F7   999                    L^FMT_NODEVICE                          ; "No device supported."                           [22]
                00000272'EF   9F   05F7                       PUSHAB  L^FMT_NODEVICE
                     7E   01   9A  05FD                       movzbl  #ds$k_printf, -(sp)
                     7E   16   98  0600                       cvtbl   #ds$k_type_command_out, -(sp)
          00000000'GF   03   FB  0603                       CALLS   #$$N+2, G^DSX$PRINT
                          05   10  060A  1000          RSB                                                   ; Return to caller                                 [20]
                                 060B  1001
                       16   90  060B  1002 20$:      movb    #ds$k_type_command_out,-                      ; Set command type code                            [26]
                00000000'EF        060D  1003                    L^ds$gb_typecode                        ; .. in global                                     [26]
                          FE3B   30  0612  1004          BSBW    PRLIST                                        ; Print device types                               [20]
                00000000'EF   94   0615  1005          clrb    L^ds$gb_typecode                              ; Clear type code                                  [26]
                          05   10  061B  1006          RSB                                                   ; Return to caller                                 [20]
                                 061C  1007 ;
                                 061C  1008 ;
```

```
                                061C  1010          .SBTTL  DSV$SHOWTESTS
                                061C  1011 ;
                                061C  1012 ;
                                061C  1013 DSV$SHOWTESTS::
                                061C  1014 ;                                                    ; test program
          00000000'EF    01    E0  061C  1015          BBS     #DS$V_LODFLG, -          ; Check the load flag     [42]
                         16        0623  1016            L^DS$GL_FLAGS, 10$     ;                         [42]
                                   0624  1017 5$:      $PRINT  #DS$K_TYPE_COMMAND_ERR,-                  ; [42]
                                   0624  1018              #DS$K_PRINTF, -        ; Print the command error [42]
                                   0624  1019              L^FMT_Start                               ; [42]
                    00000183'EF    9F  0624                  PUSHAB  L^FMT_Start
                          7E    01    9A  062A                  movzbl  #DS$K_PRINTF, -(sp)
                          7E    15    98  062D                  cvtbl   #DS$K_TYPE_COMMAND_ERR, -(sp)
          00000000'GF    03    FB  0630                  CALLS   #$$N+2, G^DSX$PRINT
                       004D    31  0637  1020          BRW     SHOWTESTS_X              ; Exit from section       [42]
                                   063A  1021
   01   00000204 9F    08    02    ED  063A  1022 10$:     CMPZV   #ENV$V_SUPER, -          ; Make sure a diagnostic  [42]
                                   0643  1023              #ENV$S_SUPER, -          ; is really loaded, and   [42]
                                   0643  1024              a#L$L_ENVIRON, -         ; Has the correct version [42]
                                   0643  1025              #ENV$_SUPER              ; for this supervisor     [42]
                         DF    12  0643  1026          BNEQ    5$                       ; If not then error       [42]
                                   0645  1027
                         30    BB  0645  1028          PUSHR   #^M<R4,R5>               ; Save Registers          [42]
                         54    D4  0647  1029          CLRL    R4                       ; and clear R1.           [42]
                   55    54    18    C5  0649  1030 20$:     MULL3   #DSP$K_SIZE, R4, R5                       ; [42]
                55    00000218'FF45    9E  064D  1031          MOVAB   @L$A_DTP[R5], R5         ; Get Address of table  ; [42]
                             0C A5    D5  0655  1032          TSTL    DSP$A_TEST(R5)           ; Test displ. in table    [42]
                                1E    13  0658  1033          BEQL    30$                      ; and exit if zero        [42]
                                   065A  1034
                                   065A  1035          $PRINT  #DS$K_TYPE_PROGRAM_INFO,- ; and print test msg  ; [42]
                                   065A  1036              #DS$K_PRINTF, -                                   ; [42]
                                   065A  1037              L^FMTTRACE, -                                     ; [42]
                                   065A  1038              @DSP$A_BASE(R5), -                                ; [42]
                                   065A  1039              DSP$A_MSG(R5)                                     ; [42]
                          04 A5    DD  065A                  PUSHL   DSP$A_MSG(R5)
                          00 B5    DD  065D                  PUSHL   @DSP$A_BASE(R5)
                000000A3'EF    9F  0660                  PUSHAB  L^FMTTRACE
                          7E    01    9A  0666                  movzbl  #DS$K_PRINTF, -(sp)
                          7E    17    98  0669                  cvtbl   #DS$K_TYPE_PROGRAM_INFO, -(sp)
          00000000'GF    05    FB  066C                  CALLS   #$$N+2, G^DSX$PRINT
                          54    D6  0673  1040          INCL    R4                       ; Increment index         [42]
                       D1    AF    17  0675  1041          JMP     20$                      ; and jump for next test  [42]
                          30    BA  0678  1042 30$:     POPR    #^M<R5,R4>               ; Restore registers       [42]
                                   067A  1043          $DS_PRINTF_S  FMTCTR             ; Carriage return         [42]
                000000A0'EF    9F  067A                  PUSHAB  FMTCTR
          00000000'9F    01    FB  0680                  CALLS   #$$N, a#DS$PRINTF
                                   0687  1044
                                   0687  1045 SHOWTESTS_X:                                             ; [42]
                                05  0687  1046          RSB                                           ; [42]
                                   0688  1047 ;
```

```
                        1049          .SBTTL  VRShowSections Routine
                        1050 ;++
                        1051 ; FUNCTIONAL DESCRIPTION:
                        1052 ;
                        1053 ;       This subroutine is called from CLI to show the sections that
                        1054 ;       a diagnostic supports.
                        1055 ;
                        1056 ; CALLING SEQUENCE:
                        1057 ;
                        1058 ;       JSB     VRSHOWSECTIONS
                        1059 ;
                        1060 ; INPUT PARAMETERS:
                        1061 ;
                        1062 ;       NONE
                        1063 ;
                        1064 ; IMPLICIT INPUTS:
                        1065 ;
                        1066 ;       L$A_SECNAM - Pointer to list of section names
                        1067 ;       L$A_NAME   - Pointer to diagnostic program name
                        1068 ;
                        1069 ; OUTPUT PARAMETERS:
                        1070 ;
                        1071 ;       NONE
                        1072 ;
                        1073 ; IMPLICIT OUTPUTS:
                        1074 ;
                        1075 ;       NONE
                        1076 ;
                        1077 ; COMPLETION CODES:
                        1078 ;
                        1079 ;       NONE
                        1080 ;
                        1081 ; SIDE EFFECTS:
                        1082 ;
                        1083 ;       NONE
                        1084 ;
                        1085 ;--
```

```
                                 0688  1087 VRSHOWSECTIONS::
                                 0688  1088
          00000000'EF   01   E0  0688  1089          BBS     #DS$V_LODFLG, -                 ; Check load flag.          [27]
                        14       068F  1090                    L^DS$GL_FLAGS, 10$              ;                           [27]
                                 0690  1091 5$:       $PRINT  #ds$k_type_command_err, -       ; Command error             [26]
                                 0690  1092                    #ds$k_printf, -                 ; .. use PRINTF             [26]
                                 0690  1093                    L^FMT_START                     ; .. No diagnostic running.[27]
                00000183'EF   9F   0690                        PUSHAB  L^FMT_START
                     7E   01   9A  0696                        movzbl  #ds$k_printf, -(sp)
                     7E   15   98  0699                        cvtbl   #ds$k_type_command_err, -(sp)
          00000000'GF   03   FB  069C                        CALLS   #$$N+2, G^DSX$PRINT
                                 05   06A3  1094          RSB                                     ; Return to caller.         [27]
                                 06A4  1095
01   00000204 9F   08   02   ED  06A4  1096 10$:     cmpzv   #env$v_super, -                 ; Make sure a diagnostic is really [26]
                                 06AD  1097                    #env$s_super, -                 ; .. loaded, and has the correct   [26]
                                 06AD  1098                    a#l$l_environ, -                ; .. version for this              [26]
                                 06AD  1099                    #env$_super                     ; .. Supervisor                    [26]
                        E1   12  06AD  1100          bneq    5$                              ; If not, error             [26]
                                 06AF  1101          $PRINT  #-ds$k_type_command_out, -      ; Command output (sticky)   [26]
                                 06AF  1102                    #ds$k_printf, -                 ; .. use PRINTF             [26]
                                 06AF  1103                    L^FMT_SECTIONS, -               ; .. format                 [26]
                                 06AF  1104                    L^L$A_NAME                      ; .. the sections message   [27]
                00000208'EF   DD   06AF                        PUSHL   L^L$A_NAME
                000001FB'EF   9F   06B5                        PUSHAB  L^FMT_SECTIONS
                     7E   01   9A  06BB                        movzbl  #ds$k_printf, -(sp)
                   7E   EA 8F   98  06BE                        cvtbl   #-ds$k_type_command_out, -(sp)
          00000000'GF   04   FB  06C2                        CALLS   #$$N+2, G^DSX$PRINT
          50   00000250 9F   D0  06C9  1105          MOVL    a#L$A_SECNAM,R0                 ; Point to section names    [27]
                     FD7D   30  06D0  1106          BSBW    PRLIST                          ; Print list of sections    [27]
                00000000'EF   94  06D3  1107          clrb    L^ds$gb_typecode                ; Clear sticky code         [26]
                                 05   06D9  1108          RSB                                     ;                           [27]
                                 06DA  1109          .END
```

```
$$N                             = 00000002     D
$ER                             = 00000001     D
$MODULE                           00000000 R   D   03
ACT_DEFAULT_DBG                   ********     X   04
APC$_ABORT                      = 00000006     D
APC$_CONTINUE                   = 00000009     D
APC$_DUMP                       = 00000003     D
APC$_NOP                        = 00000000     D
APC$_START                      = 00000005     D
APC$_ZERO                       = 00000004     D
APM$_ABTDON                     = 00000006     D
APM$_DEVERR                     = 00000002     D
APM$_DONE                       = 0000000A     D
APM$_EXCEPT                     = 0000000B     D
APM$_HARDERR                    = 00000003     D
APM$_MORE                       = 00000008     D
APM$_NOMES                      = 00000000     D
APM$_PREPERR                    = 0000000C     D
APM$_PRGERR                     = 00000005     D
APM$_SOFTERR                    = 00000004     D
APM$_SPOOL                      = 00000009     D
APM$_SYSERR                     = 00000001     D
BEGIN                             ********     X   04
BEGIN_TIME                        00000008 RG  D   02
BIN_TIME                          00000000 RG  D   02
BIT...                          = 00000004     D
CEP_FUNCTIONAL                  = 00000000     D
CEP_REPAIR                      = 00000001     D
CF$L_AP                           00000008     D
CF$L_FP                           0000000C     D
CF$L_ONCOND                       00000000     D
CF$L_PC                           00000010     D
CF$L_REG                          00000014     D
CF$W_MASK                         00000006     D
CF$W_PSW                          00000004     D
CHECK_QA_BIT                      00000433 R   D   04
CLI$K_ALNUM                     = 00000087     D
CLI$K_ALPHA                     = 00000086     D
CLI$K_BIF                       = 00000083     D
CLI$K_BIFS                      = 00000094     D
CLI$K_BR                        = 00000082     D
CLI$K_BUFSIZ                    = 00000100     D
CLI$K_CALL                      = 00000092     D
CLI$K_COMMA                     = 0000008E     D
CLI$K_DEC                       = 0000008A     D
CLI$K_EOL                       = 00000091     D
CLI$K_ERROR                     = 00000080     D
CLI$K_EXIT                      = 00000081     D
CLI$K_FILE                      = 00000095     D
CLI$K_HEX                       = 00000089     D
CLI$K_KEYWORD                   = 0000008C     D
CLI$K_NUM                       = 00000085     D
CLI$K_OCT                       = 00000088     D
CLI$K_RETURN                    = 00000093     D
CLI$K_SIZE                        0000044C     D
CLI$K_SLASH                     = 00000090     D
CLI$K_SPACE                     = 00000084     D
CLI$K_STRING                    = 0000008B     D
CLI$K_SYMBOL                    = 0000008D     D
CLI$K_VALUE                     = 0000008F     D
CLI$L_ADDRESS                     00000018     D
CLI$L_COMMAND                     00000004     D
CLI$L_DATA                        0000001C     D
CLI$L_FLAGS                       00000000     D
CLI$L_FLAGS2                      00000444     D
CLI$L_LAST                        00000024     D
CLI$L_NEXT                        00000030     D
CLI$L_PASS                        0000002C     D
CLI$L_SUBT                        00000028     D
CLI$L_TEMP_BASE                   00000448     D
CLI$L_TEST                        00000020     D
CLI$M_START_OR_RUN              = 00000008     D
CLI$Q_BUFQWD                      00000034     D
CLI$Q_FILE                        00000008     D
CLI$Q_SECTION                     00000010     D
CLI$Q_TIME                        0000043C     D
CLI$T_BUFFER                      0000003C     D
CLI$V_ADAPTER                   = 00000018     D
CLI$V_ADR                       = 0000000B     D
CLI$V_ASCII                     = 00000013     D
CLI$V_BASE_QUAL                 = 00000002     D
CLI$V_BREAK                     = 0000000A     D
CLI$V_BRIEF                     = 0000001B     D
CLI$V_BYTE                      = 0000000D     D
CLI$V_CLEAR                     = 00000002     D
CLI$V_DEC                       = 00000010     D
CLI$V_DEFAULT                   = 0000000C     D
CLI$V_DEPOSIT                   = 00000019     D
CLI$V_EVENT                     = 00000008     D
CLI$V_EXAM                      = 00000005     D
CLI$V_FLAGS                     = 00000009     D
CLI$V_HEX                       = 00000012     D
CLI$V_KERNEL                    = 00000017     D
CLI$V_LOAD                      = 00000006     D
CLI$V_LONG                      = 0000000F     D
CLI$V_NEXT                      = 00000001     D
CLI$V_NOALLOC                   = 00000000     D
CLI$V_NOTNUF                    = 00000001     D
CLI$V_OCT                       = 00000011     D
CLI$V_PREG                      = 0000001A     D
CLI$V_QA                        = 00000007     D
CLI$V_QACKLOOPLOOPS             = 0000001C     D
CLI$V_QAERRORPRINTS             = 0000001B     D
CLI$V_QAMULTIPLEPASS            = 0000001F     D
CLI$V_QASUBTESTLOOPS            = 0000001E     D
CLI$V_QATESTLOOPS               = 0000001D     D
CLI$V_REG                       = 00000014     D
CLI$V_REQUIRED                  = 00000000     D
CLI$V_RUN                       = 00000015     D
CLI$V_SET                       = 00000003     D
CLI$V_SHOW                      = 00000004     D
CLI$V_START_OR_RUN              = 00000003     D
CLI$V_VALSEC                    = 00000016     D
CLI$V_WORD                      = 0000000E     D
```

```
DEVREV$T                        ********  X   00
DEVREV$T_X                      ********  X   00
DISPATCH                        ********  X   04
DS$CLRVEC                       ********  X   04
DS$GB_TYPECODE                  ********  X   04
DS$GK_SID                       ********  X   04
DS$GL_CLIBASE                   ********  X   04
DS$GL_FLAGS                     ********  X   04
DS$GL_FSTTEST                   ********  X   04
DS$GL_LSTTEST                   ********  X   04
DS$GL_NUMTEST                   ********  X   04
DS$GL_SUBTEST                   ********  X   04
DS$GPHARD                       ********  X   04
DS$GQ_TESTID                    ********  X   04
DS$K_BBC                      = 0000001D  G D
DS$K_BBCC                     = 00000021  G D
DS$K_BBCCI                    = 00000023  G D
DS$K_BBCS                     = 0000001F  G D
DS$K_BBS                      = 0000001C  G D
DS$K_BBSC                     = 00000020  G D
DS$K_BBSS                     = 0000001E  G D
DS$K_BBSSI                    = 00000022  G D
DS$K_BCC_M                    = 0000001B  G D
DS$K_BCS_M                    = 0000001A  G D
DS$K_BEQLU_B                  = 00000005  G D
DS$K_BEQLU_M                  = 00000011  G D
DS$K_BEQL_B                   = 00000004  G D
DS$K_BEQL_M                   = 00000010  G D
DS$K_BERROR                   = 00000026  G D
DS$K_BGEQU_B                  = 00000007  G D
DS$K_BGEQU_M                  = 00000013  G D
DS$K_BGEQ_B                   = 00000006  G D
DS$K_BGEQ_M                   = 00000012  G D
DS$K_BGTRU_B                  = 00000009  G D
DS$K_BGTRU_M                  = 00000015  G D
DS$K_BGTR_B                   = 00000008  G D
DS$K_BGTR_M                   = 00000014  G D
DS$K_BLBC                     = 00000025  G D
DS$K_BLBS                     = 00000024  G D
DS$K_BLEQU_B                  = 00000003  G D
DS$K_BLEQU_M                  = 0000000F  G D
DS$K_BLEQ_B                   = 00000002  G D
DS$K_BLEQ_M                   = 0000000E  G D
DS$K_BLSSU_B                  = 00000001  G D
DS$K_BLSSU_M                  = 0000000D  G D
DS$K_BLSS_B                   = 00000000  G D
DS$K_BLSS_M                   = 0000000C  G D
DS$K_BNEQU_B                  = 0000000B  G D
DS$K_BNEQU_M                  = 00000017  G D
DS$K_BNEQ_B                   = 0000000A  G D
DS$K_BNEQ_M                   = 00000016  G D
DS$K_BNERROR                  = 00000027  G D
DS$K_BVC_M                    = 00000019  G D
DS$K_BVS_M                    = 00000018  G D
DS$K_DS_STARTING_LOCATION     = 00010000    D
DS$K_PRINTB                   = 00000002    D
DS$K_PRINTF                   = 00000001    D
DS$K_PRINTI                   = 00000000    D
DS$K_PRINTX                   = 00000003    D
DS$K_TYPE_ABORT_PROGRAM       = 00000014    D
DS$K_TYPE_ABORT_TEST          = 00000013    D
DS$K_TYPE_COMMAND_ERR         = 00000015    D
DS$K_TYPE_COMMAND_OUT         = 00000016    D
DS$K_TYPE_CRD_AUTOTEST        = 0000001A    D
DS$K_TYPE_DS_PROMPT           = 00000001    D
DS$K_TYPE_DS_START            = 0000001D    D
DS$K_TYPE_ERRDEV              = 00000008    D
DS$K_TYPE_ERRHARD             = 00000006    D
DS$K_TYPE_ERROR_BODY          = 00000009    D
DS$K_TYPE_ERROR_END           = 0000000A    D
DS$K_TYPE_ERRPREP             = 0000001B    D
DS$K_TYPE_ERRSOFT             = 00000007    D
DS$K_TYPE_ERRSUP              = 00000004    D
DS$K_TYPE_ERRSYS              = 00000005    D
DS$K_TYPE_ERR_HALT            = 0000000D    D
DS$K_TYPE_EXCEPTION           = 0000000C    D
DS$K_TYPE_EXCEPTION_HEAD      = 0000000B    D
DS$K_TYPE_FIRST_PASS          = 00000011    D
DS$K_TYPE_GENERAL             = 00000000    D
DS$K_TYPE_GENERAL_ERROR       = 00000003    D
DS$K_TYPE_NO_TESTS            = 00000012    D
DS$K_TYPE_PARAM_ERROR         = 0000001C    D
DS$K_TYPE_PROGRAM_END         = 00000010    D
DS$K_TYPE_PROGRAM_INFO        = 00000017    D
DS$K_TYPE_PROGRAM_START       = 0000000F    D
DS$K_TYPE_QIO_INVADP          = 00000024    D
DS$K_TYPE_QIO_NODRIVER        = 00000022    D
DS$K_TYPE_QIO_WRONGVER        = 00000023    D
DS$K_TYPE_SCRIPT_ECHO         = 00000021    D
DS$K_TYPE_SCRIPT_PNF          = 0000001E    D
DS$K_TYPE_SCRIPT_PROMPT       = 00000020    D
DS$K_TYPE_SCRIPT_SKIP         = 0000001F    D
DS$K_TYPE_SEQUENCE_ERROR      = 00000019    D
DS$K_TYPE_START_ERR           = 00000018    D
DS$K_TYPE_START_LIST          = 00000025    D
DS$K_TYPE_SUMMARY             = 0000000E    D
DS$K_TYPE_USER_PROMPT         = 00000002    D
DS$M_ABRTFLG                  = 00000040    D
DS$M_BADTIME                  = 00100000    D
DS$M_BATCH                    = 00400000    D
DS$M_BRKCLR                   = 00001000    D
DS$M_BRKPT                    = 00000800    D
DS$M_CHARFLG                  = 00000100    D
DS$M_CMDFLG                   = 00000080    D
DS$M_CTRLC                    = 00000001    D
DS$M_CTRLO                    = 00010000    D
DS$M_DEVFLG                   = 00000200    D
DS$M_DISABLCC                 = 01000000    D
DS$M_DONFLG                   = 00002000    D
DS$M_ERRFLG                   = 00000010    D
DS$M_EXCEPT                   = 00080000    D
DS$M_EXETST                   = 00040000    D
DS$M_HLTFLG                   = 00000008    D
DS$M_LODFLG                   = 00000002    D
```

```
DS$M_MEMMGT                     = 00008000    D
DS$M_NI                         = 04000000    D
DS$M_OUTPUT                     = 00800000    D
DS$M_RUBFLG                     = 00000020    D
DS$M_SCRIPT                     = 00200000    D
DS$M_SETIMR                     = 02000000    D
DS$M_STRFLG                     = 00000004    D
DS$M_SUBT                       = 00004000    D
DS$M_SYSFLG                     = 00000400    D
DS$M_TIMED                      = 02000000    D
DS$M_TIMED_OUT                  = 08000000    D
DS$M_TIMRON                     = 00020000    D
DS$PRINTF                         ********    X   04
DS$V_ABRTFLG                    = 00000006    D
DS$V_BADTIME                    = 00000014    D
DS$V_BATCH                      = 00000016    D
DS$V_BRKCLR                     = 0000000C    D
DS$V_BRKPT                      = 0000000B    D
DS$V_CHARFLG                    = 00000008    D
DS$V_CMDFLG                     = 00000007    D
DS$V_CTRLC                      = 00000000    D
DS$V_CTRLO                      = 00000010    D
DS$V_DEVFLG                     = 00000009    D
DS$V_DISABLCC                   = 00000018    D
DS$V_DONFLG                     = 0000000D    D
DS$V_ERRFLG                     = 00000004    D
DS$V_EXCEPT                     = 00000013    D
DS$V_EXETST                     = 00000012    D
DS$V_HLTFLG                     = 00000003    D
DS$V_LODFLG                     = 00000001    D
DS$V_MEMMGT                     = 0000000F    D
DS$V_NI                         = 0000001A    D
DS$V_OUTPUT                     = 00000017    D
DS$V_RUBFLG                     = 00000005    D
DS$V_SCRIPT                     = 00000015    D
DS$V_SETIMR                     = 00000019    D
DS$V_STRFLG                     = 00000002    D
DS$V_SUBT                       = 0000000E    D
DS$V_SYSFLG                     = 0000000A    D
DS$V_TIMED                      = 00000019    D
DS$V_TIMED_OUT                  = 0000001B    D
DS$V_TIMRON                     = 00000011    D
DSA$AL_APTMAIL                    0000FE00    D
DSA$AT_APTTXT                     0000FA00    D
DSA$GL_APTCOM                     0000FE04    D
DSA$GL_DEVLEN                     0000FE58    D
DSA$GL_ERRNO                      0000FE44    D
DSA$GL_EVENT                      0000FE48    D
DSA$GL_FLAGS                      0000FE00    D
DSA$GL_MSGTYP                     0000FE40    D
DSA$GL_PASSES                     0000FE08    D
DSA$GL_PASSNO                     0000FE54    D
DSA$GL_SECTNO                     0000FE10    D
DSA$GL_SID                        0000FE14    D
DSA$GL_SUBTNO                     0000FE4C    D
DSA$GL_TESTNO                     0000FE50    D
DSA$GL_UNITS                      0000FE0C    D
DSA$GQ_MSGPTR                     0000FE68    D
DSA$GT_DEVNAM                     0000FE5C    D
DSA$M_LOAD_DEBUGGER             = 40000000    D
DSA$V_APT                       = 0000001F    D
DSA$V_DEBUG                     = 0000001A    D
DSA$V_LOAD_DEBUGGER             = 0000001E    D
DSA$V_QA                        = 0000000F    D
DSA$V_USER                      = 0000001C    D
DSI$ABORT                         000004C1  RG D   04
DSP$A_BASE                        00000000    D
DSP$A_DATA                        00000008    D
DSP$A_MSG                         00000004    D
DSP$A_TEST                        0000000C    D
DSP$GEN_PTABLES                   ********      X   04
DSP$K_SIZE                      = 00000018    D
DSP$Q_SECTION                     00000010    D
DSQA$K_DISPAT_AFTER_INIT        = 00000014    D
DSQA$K_DISPAT_BEFORE_INIT       = 00000013    D
DSQA$K_DISPAT_CALLTEST          = 00000015    D
DSQA$K_DISPAT_DONE_TESTS        = 00000018    D
DSQA$K_DISPAT_DSX$ENDPASS_BGN   = 00000001    D
DSQA$K_DISPAT_DSX$ENDPASS_END   = 00000002    D
DSQA$K_DISPAT_DSX$ENDPASS_MID   = 00000000    D
DSQA$K_DISPAT_DS_CLEANUP_BGN    = 00000003    D
DSQA$K_DISPAT_DS_CLEANUP_END    = 00000004    D
DSQA$K_DUMMY_1                  = 00000007    D
DSQA$K_ERROR_ERROR_BGN          = 00000006    D
DSQA$K_ERROR_ERROR_END          = 00000005    D
DSQA$K_KERNEL_INIT_CONTEXT      = 00000008    D
DSQA$K_LOOP_DSX$BGNSUB          = 00000009    D
DSQA$K_LOOP_DSX$CKLOOP          = 0000000B    D
DSQA$K_LOOP_DSX$ENDSUB          = 0000000A    D
DSQA$K_PARAM_DSX$GETADDRESS     = 0000000E    D
DSQA$K_PARAM_DSX$GETDATA        = 0000000C    D
DSQA$K_PARAM_DSX$GETLOGICAL     = 0000000F    D
DSQA$K_PARAM_DSX$GETSTRING      = 00000010    D
DSQA$K_PARAM_DSX$GETVIELD       = 0000000D    D
DSQA$K_QA_DSX$BRANCH            = 00000011    D
DSQA$K_START_BEFORE_HEADER      = 00000017    D
DSQA$K_START_VRRESTART          = 00000012    D
DSQA$K_START_VRSTART            = 00000016    D
DSR$FIND_USER_PC                  ********      X   04
DSV$DEBUG                         ********      X   04
DSV$SHOWTESTS                     0000061C  RG D   04
DSX$ABORT                         00000497  RG D   04
DSX$PRINT                         ********      X   04
DS_CLEANUP                        ********      X   04
DS_TESTID                         ********      X   04
DUE_TIME                          00000018  RG D   02
ENV$M_DOMAIN                    = 00000002    D
ENV$M_LEVEL                     = 00000001    D
ENV$M_SUPER                     = 000003FC    D
ENV$S_DOMAIN                    = 00000001    D
ENV$S_LEVEL                     = 00000001    D
ENV$S_SUPER                     = 00000008    D
ENV$V_DOMAIN                    = 00000001    D
ENV$V_LEVEL                     = 00000000    D
```

```
ENV$V_SUPER                     =  00000002     D
ENV$_CPU                        =  00000000     D
ENV$_FUNCTIONAL                 =  00000000     D
ENV$_REPAIR                     =  00000001     D
ENV$_SUPER                      =  00000001     D
ENV$_SYSTEM                     =  00000001     D
FALSE                           =  00000000     D
FMTCTR                             000000A0 R   D   03
FMTTRACE                           000000A3 R   D   03
FMT_ABORT                          00000046 R   D   03
FMT_ABORT_NO_PC                    00000077 R   D   03
FMT_AMBSEC                         000001CB R   D   03
FMT_CRLF                           00000231 R   D   03
FMT_ILLSEC                         0000019D R   D   03
FMT_MISMATCH                       00000234 R   D   03
FMT_NAMES                          0000022C R   D   03
FMT_NODEVICE                       00000272 R   D   03
FMT_NOUNITS                        0000014A R   D   03
FMT_PROG                           00000006 R   D   03
FMT_SECTIONS                       000001FB R   D   03
FMT_START                          00000183 R   D   03
FMT_TESTING                        00000222 R   D   03
FMT_TESTRANGE                      0000011F R   D   03
FMT_TIME_EXPIRED                   000000B7 R   D   03
FMT_TIME_EXPIRED_NO_PC             000000EF R   D   03
GET_TIME                           ********       X   04
INIT_CONTEXT                       ********       X   04
IS_IT_USER_PC                      000004C1 R     D   04
L$A_CCP                            00000240       D
L$A_DEVP                           0000021C       D
L$A_DREG                           00000224       D
L$A_DTP                            00000218       D
L$A_ICP                            0000023C       D
L$A_LASTADR                        00000214       D
L$A_NAME                           00000208       D
L$A_REPP                           00000244       D
L$A_SECNAM                         00000250       D
L$A_STATAB                         00000248       D
L$A_TSTCNT                         00000254       D
L$L_ENVIRON                        00000204       D
L$L_ERRTYP                         0000024C       D
L$L_HEADLENGTH                     00000200       D
L$L_REV                            0000020C       D
L$L_UNIT                           00000220       D
L$L_UNUSED                         00000228       D
L$L_UPDATE                         00000210       D
MAPFREE                            ********       X   04
MAP_DEBUGGER                       ********       X   04
MP$GL_CONDITION_FLAGS              *********W GX      00
MP$GL_FLAGS                        *********W GX      00
MP$GR_BOOTED_PROCESSORS            *********W GX      00
MP$GR_IDLING_PROCESSORS            *********W GX      00
MP$_CLEAR_LOCKS                    *********W GX      00
OFF                             =  00000000     D
ON                              =  00000001     D
POLL_TIME                          00000010 RG  D   02
PRLIST                             00000450 R   D   04
QA$AOB_CHECK_STATE                 ********       X   04
QA$AOB_FLAGS                       ********       X   04
QA$K_ABORT_NOW                  =  00000007     D
QA$K_APT_PHASE_ONE              =  0000000B     D
QA$K_APT_PHASE_TWO              =  0000000C     D
QA$K_BAD_ADDRESS                =  00000009     D
QA$K_CONTROL_C_CONT             =  0000000A     D
QA$K_DEALLOCATION               =  0000000E     D
QA$K_ERROR_PHASE_ONE            =  00000005     D
QA$K_ERROR_PHASE_THREE          =  00000007     D
QA$K_ERROR_PHASE_TWO            =  00000006     D
QA$K_FAILURE                    =  00000000     D
QA$K_FIRST_BRANCH_CODE          =  00000000     D
QA$K_FIRST_TIME                 =  00000005     D
QA$K_INIT_CONTEXT_DONE          =  00000003     D
QA$K_LAST_BRANCH_CODE           =  00000025     D
QA$K_LOOP_ON_SUBTEST            =  00000003     D
QA$K_LOOP_ON_TEST               =  00000002     D
QA$K_LOOP_TEST_DONE             =  00000004     D
QA$K_MEMORY_MANAGE              =  0000000D     D
QA$K_MULTIPLE_PASS              =  00000001     D
QA$K_NODEF                      =  00000000     D
QA$K_NORMAL_START               =  00000000     D
QA$K_NO_HEADER                  =  00000006     D
QA$K_NUMBER_OF_CHECKS           =  0000000F     D
QA$K_NUMBER_QA_FLAGS            =  00000008     D
QA$K_NUMB_ROUTINE_STATES        =  00000004     D
QA$K_QA_DEBUG                   =  00000001     D
QA$K_QA_DONE                    =  00000002     D
QA$K_RUN_BACKWARDS              =  00000004     D
QA$K_SECTION                    =  00000008     D
QA$K_SUCCESS                    =  00000001     D
QA$MAIN                            ********       X   04
QA$V_CHECK_DONE                 =  00000003     D
QA$V_DOING_CLEANUP              =  00000002     D
QA$V_ERROR_FOUND                =  00000001     D
QA$V_INIT_DONE                  =  00000000     D
QIO$ADDUNIT                        ********       X   04
QIO$CLEANUP                        ********       X   04
SCB$L_ACCESS                       00000020       D
SCB$L_ARITH                        00000034       D
SCB$L_BREAK                        0000002C       D
SCB$L_CHME                         00000044       D
SCB$L_CHMK                         00000040       D
SCB$L_CHMS                         00000048       D
SCB$L_CHMU                         0000004C       D
SCB$L_COMPAT                       00000030       D
SCB$L_KNLSTK                       00000008       D
SCB$L_MACHCHK                      00000004       D
SCB$L_OPCCUS                       00000014       D
SCB$L_OPCDEC                       00000010       D
SCB$L_POWER                        0000000C       D
SCB$L_RADRMOD                      0000001C       D
SCB$L_ROPRAND                      00000018       D
SCB$L_RXDB                         000000F8       D
SCB$L_SFTLVL1                      00000084       D
SCB$L_SFTLVL10                     000000A8       D
```

```
SCB$L_SFTLVL11                 000000AC    D
SCB$L_SFTLVL12                 000000B0    D
SCB$L_SFTLVL13                 000000B4    D
SCB$L_SFTLVL14                 000000B8    D
SCB$L_SFTLVL15                 000000BC    D
SCB$L_SFTLVL2                  00000088    D
SCB$L_SFTLVL3                  0000008C    D
SCB$L_SFTLVL4                  00000090    D
SCB$L_SFTLVL5                  00000094    D
SCB$L_SFTLVL6                  00000098    D
SCB$L_SFTLVL7                  0000009C    D
SCB$L_SFTLVL8                  000000A0    D
SCB$L_SFTLVL9                  000000A4    D
SCB$L_TBIT                     00000028    D
SCB$L_TIMER                    000000C0    D
SCB$L_TRANSL                   00000024    D
SCB$L_TXDB                     000000FC    D
SCB$L_VM                       00000038    D
SCB$L_ZERO                     00000000    D
SCRIPT$FLUSH                   ********   X   04
SEP_FUNCTIONAL               = 00000002    D
SEP_REPAIR                   = 00000003    D
SHOWTESTS_X                    00000687 R  D  04
SIZ...                       = 00000001    D
START1                         000000A3 R  D  04
TRUE                         = 00000001    D
VRABORT                        000004BD RG D  04
VRABORT_COM                    000004D1 R  D  04
VRRESTART                      000000A3 RG D  04
VRSHOWSECTIONS                 00000688 RG D  04
VRSTART                        00000000 RG D  04
VRSTART_X                      00000422 R  D  04
VRSUPPORT                      000005C2 RG D  04

                                          +----------------+
                                          ! Psect synopsis !
                                          +----------------+

PSECT name                    Allocation          PSECT No.  Attributes
----------                    ----------          ---------  ----------
.  ABS  .                     00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                         0000FE70  (65136.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
WORK                          0000001C  (   28.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
DATA                          0000029C  (  668.)  03 (  3.)  NOPIC   USR   CON   REL   LCL   SHR NOEXE   RD  NOWRT NOVEC BYTE
CODE                          000006DA  ( 1754.)  04 (  4.)  NOPIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC BYTE
```

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$N	=00000002	1104 (4)	#-1019 (4) 1039 (4) #-1043 (4) #-1093 (4)
			1104 (4) #-424 (3) #-459 (3) #-521 (3)
			#-535 (3) 590 (3) 622 (3) #-643 (3)
			907 (4) 915 (4) 925 (4) 932 (4)
			#-984 (4) #-999 (4)
$ER	=00000001	302 (3)	
$MODULE	00000000-R	302 (3)	
ACT_DEFAULT_DBG	00000000-XR		430 (3)
APC$_ABORT	=00000006	277 (3)	
APC$_CONTINUE	=00000009	277 (3)	
APC$_DUMP	=00000003	277 (3)	
APC$_NOP	=00000000	277 (3)	
APC$_START	=00000005	277 (3)	
APC$_ZERO	=00000004	277 (3)	
APM$_ABTDON	=00000006	277 (3)	#-945 (4)
APM$_DEVERR	=00000002	277 (3)	
APM$_DONE	=0000000A	277 (3)	
APM$_EXCEPT	=0000000B	277 (3)	
APM$_HARDERR	=00000003	277 (3)	
APM$_MORE	=00000008	277 (3)	
APM$_NOMES	=00000000	277 (3)	
APM$_PREPERR	=0000000C	277 (3)	
APM$_PRGERR	=00000005	277 (3)	
APM$_SOFTERR	=00000004	277 (3)	
APM$_SPOOL	=00000009	277 (3)	
APM$_SYSERR	=00000001	277 (3)	
BEGIN	00000000-XR		756 (3) 947 (4)
BEGIN_TIME	00000008-R	293 (3)	738 (3) #-742 (3)
BIN_TIME	00000000-R	291 (3)	#-733 (3) #-743 (3) #-744 (3)
BIT...	=00000004	283 (3)	271 (3) 273 (3) 280 (3) 281 (3)
			282 (3) 283 (3)
CEP_FUNCTIONAL	=00000000	273 (3)	
CEP_REPAIR	=00000001	273 (3)	
CHECK_QA_BIT	00000433-R	766 (3)	#-434 (3) #-463 (3)
CLI$K_ALNUM	=00000087	271 (3)	
CLI$K_ALPHA	=00000086	271 (3)	
CLI$K_BIF	=00000083	271 (3)	
CLI$K_BIFS	=00000094	271 (3)	
CLI$K_BR	=00000082	271 (3)	
CLI$K_BUFSIZ	=00000100	278 (3)	278 (3)
CLI$K_CALL	=00000092	271 (3)	
CLI$K_COMMA	=0000008E	271 (3)	
CLI$K_DEC	=0000008A	271 (3)	
CLI$K_EOL	=00000091	271 (3)	
CLI$K_ERROR	=00000080	271 (3)	
CLI$K_EXIT	=00000081	271 (3)	
CLI$K_FILE	=00000095	271 (3)	
CLI$K_HEX	=00000089	271 (3)	
CLI$K_KEYWORD	=0000008C	271 (3)	
CLI$K_NUM	=00000085	271 (3)	

```
CLI$K_OCT                       =00000088     271     (3)
CLI$K_RETURN                    =00000093     271     (3)
CLI$K_SIZE                       0000044C     278     (3)
CLI$K_SLASH                     =00000090     271     (3)
CLI$K_SPACE                     =00000084     271     (3)
CLI$K_STRING                    =0000008B     271     (3)
CLI$K_SYMBOL                    =0000008D     271     (3)
CLI$K_VALUE                     =0000008F     271     (3)
CLI$L_ADDRESS                    00000018     278     (3)
CLI$L_COMMAND                    00000004     278     (3)
CLI$L_DATA                       0000001C     278     (3)
CLI$L_FLAGS                      00000000     278     (3)      510    (3)      515    (3)      532    (3)      769    (3)
CLI$L_FLAGS2                     00000444     278     (3)
CLI$L_LAST                       00000024     278     (3)   #-569    (3)
CLI$L_NEXT                       00000030     278     (3)
CLI$L_PASS                       0000002C     278     (3)   #-561    (3)
CLI$L_SUBT                       00000028     278     (3)   #-583    (3)
CLI$L_TEMP_BASE                  00000448     278     (3)
CLI$L_TEST                       00000020     278     (3)   #-563    (3)
CLI$M_START_OR_RUN              =00000008     278     (3)
CLI$Q_BUFQWD                     00000034     278     (3)
CLI$Q_FILE                       00000008     278     (3)
CLI$Q_SECTION                    00000010     278     (3)   #-484    (3)    #-494    (3)    #-495    (3)
CLI$Q_TIME                       0000043C     278     (3)
CLI$T_BUFFER                     0000003C     278     (3)
CLI$V_ADAPTER                   =00000018     278     (3)
CLI$V_ADR                       =0000000B     278     (3)
CLI$V_ASCII                     =00000013     278     (3)
CLI$V_BASE_QUAL                 =00000002     278     (3)
CLI$V_BREAK                     =0000000A     278     (3)
CLI$V_BRIEF                     =0000001B     278     (3)
CLI$V_BYTE                      =0000000D     278     (3)
CLI$V_CLEAR                     =00000002     278     (3)
CLI$V_DEC                       =00000010     278     (3)
CLI$V_DEFAULT                   =0000000C     278     (3)
CLI$V_DEPOSIT                   =00000019     278     (3)
CLI$V_EVENT                     =00000008     278     (3)
CLI$V_EXAM                      =00000005     278     (3)
CLI$V_FLAGS                     =00000009     278     (3)
CLI$V_HEX                       =00000012     278     (3)
CLI$V_KERNEL                    =00000017     278     (3)
CLI$V_LOAD                      =00000006     278     (3)
CLI$V_LONG                      =0000000F     278     (3)
CLI$V_NEXT                      =00000001     278     (3)
CLI$V_NOALLOC                   =00000000     278     (3)
CLI$V_NOTNUF                    =00000001     278     (3)
CLI$V_OCT                       =00000011     278     (3)
CLI$V_PREG                      =0000001A     278     (3)
CLI$V_QA                        =00000007     278     (3)   #-768    (3)
CLI$V_QACKLOOPLOOPS             =0000001C     278     (3)
CLI$V_QAERRORPRINTS             =0000001B     278     (3)
CLI$V_QAMULTIPLEPASS            =0000001F     278     (3)
CLI$V_QASUBTESTLOOPS            =0000001E     278     (3)
CLI$V_QATESTLOOPS               =0000001D     278     (3)
CLI$V_REG                       =00000014     278     (3)
CLI$V_REQUIRED                  =00000000     278     (3)
CLI$V_RUN                       =00000015     278     (3)
```

```
CLI$V_SET                       =00000003     278     (3)
CLI$V_SHOW                      =00000004     278     (3)
CLI$V_START_OR_RUN              =00000003     278     (3)      278     (3)
CLI$V_VALSEC                    =00000016     278     (3)    #-509     (3)    #-514     (3)    #-531     (3)
CLI$V_WORD                      =0000000E     278     (3)
DEVREV$T                         00000000-XR                    256     (3)      445     (3)
DEVREV$T_X                       00000000-XR                    257     (3)      446     (3)
DISPATCH                         00000000-XR                    747     (3)
DS$CLRVEC                        00000000-XR                    703     (3)
DS$GB_TYPECODE                   00000000-XR                  #-1003    (4)    #-1005    (4)    #-1107    (4)    #-524     (3)
                                                              #-538     (3)    #-646     (3)
DS$GK_SID                        00000000-XR                  #-414     (3)
DS$GL_CLIBASE                    00000000-XR                    482     (3)      767     (3)
DS$GL_FLAGS                      00000000-XR                   1016     (4)     1090     (4)      421     (3)    #-552     (3)
                                                              #-732     (3)    #-745     (3)    #-755     (3)    #-860     (4)
                                                              #-890     (4)      892     (4)      900     (4)      919     (4)
                                                               937     (4)    #-943     (4)      981     (4)
DS$GL_FSTTEST                    00000000-XR                  #-564     (3)    #-567     (3)    #-576     (3)
DS$GL_LSTTEST                    00000000-XR                  #-570     (3)    #-574     (3)    #-579     (3)
DS$GL_NUMTEST                    00000000-XR                  #-559     (3)    #-573     (3)    #-577     (3)    #-580     (3)
                                                              #-590     (3)
DS$GL_SUBTEST                    00000000-XR                  #-584     (3)
DS$GPHARD                        00000000-XR                    671     (3)      710     (3)
DS$GQ_TESTID                     00000000-XR                  #-907     (4)    #-915     (4)    #-925     (4)    #-932     (4)
DS$K_BBC                        =0000001D     282     (3)
DS$K_BBCC                       =00000021     282     (3)
DS$K_BBCCI                      =00000023     282     (3)
DS$K_BBCS                       =0000001F     282     (3)
DS$K_BBS                        =0000001C     282     (3)
DS$K_BBSC                       =00000020     282     (3)
DS$K_BBSS                       =0000001E     282     (3)
DS$K_BBSSI                      =00000022     282     (3)
DS$K_BCC_M                      =0000001B     282     (3)
DS$K_BCS_M                      =0000001A     282     (3)
DS$K_BEQLU_B                    =00000005     282     (3)
DS$K_BEQLU_M                    =00000011     282     (3)
DS$K_BEQL_B                     =00000004     282     (3)
DS$K_BEQL_M                     =00000010     282     (3)
DS$K_BERROR                     =00000026     282     (3)
DS$K_BGEQU_B                    =00000007     282     (3)
DS$K_BGEQU_M                    =00000013     282     (3)
DS$K_BGEQ_B                     =00000006     282     (3)
DS$K_BGEQ_M                     =00000012     282     (3)
DS$K_BGTRU_B                    =00000009     282     (3)
DS$K_BGTRU_M                    =00000015     282     (3)
DS$K_BGTR_B                     =00000008     282     (3)
DS$K_BGTR_M                     =00000014     282     (3)
DS$K_BLBC                       =00000025     282     (3)      282     (3)
DS$K_BLBS                       =00000024     282     (3)
DS$K_BLEQU_B                    =00000003     282     (3)
DS$K_BLEQU_M                    =0000000F     282     (3)
DS$K_BLEQ_B                     =00000002     282     (3)
DS$K_BLEQ_M                     =0000000E     282     (3)
DS$K_BLSSU_B                    =00000001     282     (3)
DS$K_BLSSU_M                    =0000000D     282     (3)
DS$K_BLSS_B                     =00000000     282     (3)      282     (3)
DS$K_BLSS_M                     =0000000C     282     (3)
```

```
DS$K_BNEQU_B                    =0000000B      282     (3)
DS$K_BNEQU_M                    =00000017      282     (3)
DS$K_BNEQ_B                     =0000000A      282     (3)
DS$K_BNEQ_M                     =00000016      282     (3)
DS$K_BNERROR                    =00000027      282     (3)
DS$K_BVC_M                      =00000019      282     (3)
DS$K_BVS_M                      =00000018      282     (3)
DS$K_DS_STARTING_LOCATION       =00010000      282     (3)
DS$K_PRINTB                     =00000002      283     (3)
DS$K_PRINTF                     =00000001      283     (3)    #-1019   (4)    #-1039   (4)    #-1093   (4)    #-1104   (4)
                                                              #-424    (3)    #-459    (3)    #-521    (3)    #-535    (3)
                                                              #-590    (3)    #-622    (3)    #-643    (3)    #-676    (3)
                                                              #-682    (3)    #-687    (3)    #-805    (3)    #-907    (4)
                                                              #-915    (4)    #-925    (4)    #-932    (4)    #-984    (4)
                                                              #-999    (4)
DS$K_PRINTI                     =00000000      283     (3)
DS$K_PRINTX                     =00000003      283     (3)
DS$K_TYPE_ABORT_PROGRAM         =00000014      283     (3)    #-907    (4)    #-925    (4)
DS$K_TYPE_ABORT_TEST            =00000013      283     (3)
DS$K_TYPE_COMMAND_ERR           =00000015      283     (3)    #-1019   (4)    #-1093   (4)    #-424    (3)    #-521    (3)
                                                              #-535    (3)    #-590    (3)    #-984    (4)
DS$K_TYPE_COMMAND_OUT           =00000016      283     (3)    #-1002   (4)    #-1104   (4)    #-999    (4)
DS$K_TYPE_CRD_AUTOTEST          =0000001A      283     (3)
DS$K_TYPE_DS_PROMPT             =00000001      283     (3)
DS$K_TYPE_DS_START              =0000001D      283     (3)
DS$K_TYPE_ERRDEV                =00000008      283     (3)
DS$K_TYPE_ERRHARD               =00000006      283     (3)
DS$K_TYPE_ERROR_BODY            =00000009      283     (3)
DS$K_TYPE_ERROR_END             =0000000A      283     (3)
DS$K_TYPE_ERRPREP               =0000001B      283     (3)
DS$K_TYPE_ERRSOFT               =00000007      283     (3)
DS$K_TYPE_ERRSUP                =00000004      283     (3)
DS$K_TYPE_ERRSYS                =00000005      283     (3)
DS$K_TYPE_ERR_HALT              =0000000D      283     (3)
DS$K_TYPE_EXCEPTION             =0000000C      283     (3)
DS$K_TYPE_EXCEPTION_HEAD        =0000000B      283     (3)
DS$K_TYPE_FIRST_PASS            =00000011      283     (3)
DS$K_TYPE_GENERAL               =00000000      283     (3)
DS$K_TYPE_GENERAL_ERROR         =00000003      283     (3)
DS$K_TYPE_NO_TESTS              =00000012      283     (3)
DS$K_TYPE_PARAM_ERROR           =0000001C      283     (3)
DS$K_TYPE_PROGRAM_END           =00000010      283     (3)    #-915    (4)    #-932    (4)
DS$K_TYPE_PROGRAM_INFO          =00000017      283     (3)    #-1039   (4)    #-677    (3)    #-683    (3)    #-688    (3)
DS$K_TYPE_PROGRAM_START         =0000000F      283     (3)    #-622    (3)
DS$K_TYPE_QIO_INVADP            =00000024      283     (3)
DS$K_TYPE_QIO_NODRIVER          =00000022      283     (3)
DS$K_TYPE_QIO_WRONGVER          =00000023      283     (3)
DS$K_TYPE_SCRIPT_ECHO           =00000021      283     (3)
DS$K_TYPE_SCRIPT_PNF            =0000001E      283     (3)
DS$K_TYPE_SCRIPT_PROMPT         =00000020      283     (3)
DS$K_TYPE_SCRIPT_SKIP           =0000001F      283     (3)
DS$K_TYPE_SEQUENCE_ERROR        =00000019      283     (3)
DS$K_TYPE_START_ERR             =00000018      283     (3)    #-459    (3)    #-643    (3)
DS$K_TYPE_START_LIST            =00000025      283     (3)    #-806    (3)
DS$K_TYPE_SUMMARY               =0000000E      283     (3)
DS$K_TYPE_USER_PROMPT           =00000002      283     (3)
DS$M_ABRTFLG                    =00000040      280     (3)
```

```
DS$M_BADTIME                    =00100000     280     (3)
DS$M_BATCH                      =00400000     280     (3)
DS$M_BRKCLR                     =00001000     280     (3)
DS$M_BRKPT                      =00000800     280     (3)    #-550    (3)    #-858    (4)    #-888    (4)
DS$M_CHARFLG                    =00000100     280     (3)
DS$M_CMDFLG                     =00000080     280     (3)
DS$M_CTRLC                      =00000001     280     (3)    #-549    (3)
DS$M_CTRLO                      =00010000     280     (3)
DS$M_DEVFLG                     =00000200     280     (3)
DS$M_DISABLCC                   =01000000     280     (3)
DS$M_DONFLG                     =00002000     280     (3)    #-546    (3)
DS$M_ERRFLG                     =00000010     280     (3)
DS$M_EXCEPT                     =00080000     280     (3)    #-551    (3)    #-859    (4)    #-889    (4)
DS$M_EXETST                     =00040000     280     (3)
DS$M_HLTFLG                     =00000008     280     (3)    #-548    (3)    #-857    (4)    #-887    (4)
DS$M_LODFLG                     =00000002     280     (3)
DS$M_MEMMGT                     =00008000     280     (3)
DS$M_NI                         =04000000     280     (3)
DS$M_OUTPUT                     =00800000     280     (3)
DS$M_RUBFLG                     =00000020     280     (3)
DS$M_SCRIPT                     =00200000     280     (3)
DS$M_SETIMR                     =02000000     280     (3)
DS$M_STRFLG                     =00000004     280     (3)    #-547    (3)    #-731    (3)    #-942    (4)
DS$M_SUBT                       =00004000     280     (3)
DS$M_SYSFLG                     =00000400     280     (3)
DS$M_TIMED                      =02000000     280     (3)    #-745    (3)    #-755    (3)
DS$M_TIMED_OUT                  =08000000     280     (3)
DS$M_TIMRON                     =00020000     280     (3)
DS$PRINTF                        00000000-XR                  1043    (4)
DS$V_ABRTFLG                    =00000006     280     (3)
DS$V_BADTIME                    =00000014     280     (3)
DS$V_BATCH                      =00000016     280     (3)
DS$V_BRKCLR                     =0000000C     280     (3)
DS$V_BRKPT                      =0000000B     280     (3)
DS$V_CHARFLG                    =00000008     280     (3)
DS$V_CMDFLG                     =00000007     280     (3)
DS$V_CTRLC                      =00000000     280     (3)
DS$V_CTRLO                      =00000010     280     (3)
DS$V_DEVFLG                     =00000009     280     (3)
DS$V_DISABLCC                   =00000018     280     (3)
DS$V_DONFLG                     =0000000D     280     (3)
DS$V_ERRFLG                     =00000004     280     (3)    #-936    (4)
DS$V_EXCEPT                     =00000013     280     (3)
DS$V_EXETST                     =00000012     280     (3)
DS$V_HLTFLG                     =00000003     280     (3)
DS$V_LODFLG                     =00000001     280     (3)    #-1015   (4)    #-1089   (4)    #-420    (3)    #-980    (4)
DS$V_MEMMGT                     =0000000F     280     (3)
DS$V_NI                         =0000001A     280     (3)
DS$V_OUTPUT                     =00000017     280     (3)
DS$V_RUBFLG                     =00000005     280     (3)
DS$V_SCRIPT                     =00000015     280     (3)
DS$V_SETIMR                     =00000019     280     (3)
DS$V_STRFLG                     =00000002     280     (3)    #-891    (4)
DS$V_SUBT                       =0000000E     280     (3)
DS$V_SYSFLG                     =0000000A     280     (3)
DS$V_TIMED                      =00000019     280     (3)
DS$V_TIMED_OUT                  =0000001B     280     (3)    #-899    (4)    #-918    (4)
```

```
DS$V_TIMRON                     =00000011      280    (3)
DSA$GL_FLAGS                     0000FE00                427    (3)      429    (3)     #-433    (3)      475    (3)
                                                         601    (3)      657    (3)      697    (3)      700    (3)
                                                         771    (3)      776    (3)      853    (4)      935    (4)
DSA$GL_MSGTYP                    0000FE40               #-946    (4)
DSA$GL_PASSES                    0000FE08               #-562    (3)
DSA$GL_PASSNO                    0000FE54               #-907    (4)     #-915    (4)     #-925    (4)     #-932    (4)
DSA$GL_SECTNO                    0000FE10               #-483    (3)     #-506    (3)
DSA$GL_UNITS                     0000FE0C               #-631    (3)     #-638    (3)     #-679    (3)     #-717    (3)
DSA$M_LOAD_DEBUGGER             =40000000               #-432    (3)
DSA$V_APT                       =0000001F               #-427    (3)     #-934    (4)
DSA$V_DEBUG                     =0000001A               #-699    (3)
DSA$V_LOAD_DEBUGGER             =0000001E               #-428    (3)
DSA$V_QA                        =0000000F               #-600    (3)     #-656    (3)     #-770    (3)     #-775    (3)
                                                        #-852    (4)
DSA$V_USER                      =0000001C               #-474    (3)     #-696    (3)
DSI$ABORT                        000004C1-R     874    (4)
DSP$A_BASE                       00000000       279    (3)    #-1039    (4)
DSP$A_DATA                       00000008       279    (3)
DSP$A_MSG                        00000004       279    (3)    #-1039    (4)
DSP$A_TEST                       0000000C       279    (3)    #-1032    (4)
DSP$GEN_PTABLES                  00000000-XR                     636    (3)
DSP$K_SIZE                      =00000018       279    (3)    #-1030    (4)
DSP$Q_SECTION                    00000010       279    (3)
DSQA$K_DISPAT_AFTER_INIT        =00000014       281    (3)
DSQA$K_DISPAT_BEFORE_INIT       =00000013       281    (3)
DSQA$K_DISPAT_CALLTEST          =00000015       281    (3)
DSQA$K_DISPAT_DONE_TESTS        =00000018       281    (3)
DSQA$K_DISPAT_DSX$ENDPASS_BGN   =00000001       281    (3)
DSQA$K_DISPAT_DSX$ENDPASS_END   =00000002       281    (3)
DSQA$K_DISPAT_DSX$ENDPASS_MID   =00000000       281    (3)
DSQA$K_DISPAT_DS_CLEANUP_BGN    =00000003       281    (3)
DSQA$K_DISPAT_DS_CLEANUP_END    =00000004       281    (3)
DSQA$K_DUMMY_1                  =00000007       281    (3)
DSQA$K_ERROR_ERROR_BGN          =00000006       281    (3)
DSQA$K_ERROR_ERROR_END          =00000005       281    (3)
DSQA$K_KERNEL_INIT_CONTEXT      =00000008       281    (3)
DSQA$K_LOOP_DSX$BGNSUB          =00000009       281    (3)
DSQA$K_LOOP_DSX$CKLOOP          =0000000B       281    (3)
DSQA$K_LOOP_DSX$ENDSUB          =0000000A       281    (3)
DSQA$K_PARAM_DSX$GETADDRESS     =0000000E       281    (3)
DSQA$K_PARAM_DSX$GETDATA        =0000000C       281    (3)
DSQA$K_PARAM_DSX$GETLOGICAL     =0000000F       281    (3)
DSQA$K_PARAM_DSX$GETSTRING      =00000010       281    (3)
DSQA$K_PARAM_DSX$GETVIELD       =0000000D       281    (3)
DSQA$K_QA_DSX$BRANCH            =00000011       281    (3)
DSQA$K_START_BEFORE_HEADER      =00000017       281    (3)    #-598    (3)
DSQA$K_START_VRRESTART          =00000012       281    (3)    #-466    (3)
DSQA$K_START_VRSTART            =00000016       281    (3)    #-435    (3)
DSR$FIND_USER_PC                 00000000-XR                     878    (4)
DSV$DEBUG                        00000000-XR                     431    (3)
DSV$SHOWTESTS                    0000061C-R     1013    (4)
DSX$ABORT                        00000497-R     850    (4)
DSX$PRINT                        00000000-XR                    1019    (4)     1039    (4)     1093    (4)     1104    (4)
                                                         424    (3)      459    (3)      521    (3)      535    (3)
                                                         590    (3)      622    (3)      643    (3)      678    (3)
                                                         684    (3)      689    (3)      807    (3)      907    (4)
```

```
                                                        915    (4)      925    (4)      932    (4)      984    (4)
                                                        999    (4)
DS_CLEANUP                  00000000-XR                 472    (3)      940    (4)
DS_TESTID                   00000000-XR                 894    (4)
DUE_TIME                    00000018-R   297    (3)   #-742    (3)    #-743    (3)    #-744    (3)
ENV$M_DOMAIN                =00000002    273    (3)
ENV$M_LEVEL                 =00000001    273    (3)
ENV$M_SUPER                 =000003FC    273    (3)
ENV$S_DOMAIN                =00000001    273    (3)
ENV$S_LEVEL                 =00000001    273    (3)
ENV$S_SUPER                 =00000008    273    (3)   #-1023   (4)    #-1097   (4)    #-453    (3)    #-988    (4)
ENV$V_DOMAIN                =00000001    273    (3)     273    (3)
ENV$V_LEVEL                 =00000000    273    (3)     273    (3)
ENV$V_SUPER                 =00000002    273    (3)   #-1022   (4)    #-1096   (4)    #-452    (3)    #-987    (4)
ENV$_CPU                    =00000000    273    (3)     273    (3)
ENV$_FUNCTIONAL             =00000000    273    (3)     273    (3)
ENV$_REPAIR                 =00000001    273    (3)     273    (3)
ENV$_SUPER                  =00000001    273    (3)   #-1025   (4)    #-1099   (4)    #-455    (3)    #-990    (4)
ENV$_SYSTEM                 =00000001    273    (3)     273    (3)
FALSE                       =00000000    282    (3)
FMTCTR                      000000A0-R   314    (3)     1043   (4)
FMTTRACE                    000000A3-R   317    (3)     1039   (4)
FMT_ABORT                   00000046-R   308    (3)     907    (4)
FMT_ABORT_NO_PC             00000077-R   311    (3)     925    (4)
FMT_AMBSEC                  000001CB-R   338    (3)     521    (3)
FMT_CRLF                    00000231-R   348    (3)     686    (3)
FMT_ILLSEC                  0000019D-R   335    (3)     535    (3)
FMT_MISMATCH                00000234-R   351    (3)     459    (3)
FMT_NAMES                   0000022C-R   346    (3)     681    (3)
FMT_NODEVICE                00000272-R   355    (3)     999    (4)
FMT_NOUNITS                 0000014A-R   329    (3)     643    (3)
FMT_PROG                    00000006-R   304    (3)     622    (3)
FMT_SECTIONS                000001FB-R   341    (3)     1104   (4)
FMT_START                   00000183-R   332    (3)     1019   (4)      1093   (4)      424    (3)      984    (4)
FMT_TESTING                 00000222-R   344    (3)     675    (3)
FMT_TESTRANGE               0000011F-R   326    (3)     590    (3)
FMT_TIME_EXPIRED            000000B7-R   320    (3)     915    (4)
FMT_TIME_EXPIRED_NO_PC      000000EF-R   323    (3)     932    (4)
GET_TIME                    00000000-XR                 739    (3)
INIT_CONTEXT                00000000-XR                 462    (3)      730    (3)      939    (4)
IS_IT_USER_PC               000004C1-R   875    (4)   #-864    (4)
L$A_DEVP                    0000021C                  #-644    (3)    #-992    (4)
L$A_DTP                     00000218                   1031    (4)
L$A_NAME                    00000208                  #-1104   (4)    #-622    (3)
L$A_SECNAM                  00000250                  #-1105   (4)    #-488    (3)    #-505    (3)    #-522    (3)
                                                      #-536    (3)
L$A_TSTCNT                  00000254                  #-558    (3)    #-622    (3)
L$L_ENVIRON                 00000204                   1024    (4)      1098   (4)      454    (3)      705    (3)
                                                        989    (4)
L$L_REV                     0000020C                  #-622    (3)
L$L_UNIT                    00000220                  #-632    (3)
L$L_UPDATE                  00000210                  #-622    (3)
MAPFREE                     00000000-XR                 698    (3)
MAP_DEBUGGER                00000000-XR                 701    (3)
MP$GL_CONDITION_FLAGS       00000000-XR                 262    (3)    #-419    (3)
MP$GL_FLAGS                 00000000-XR                 261    (3)    #-418    (3)
MP$GR_BOOTED_PROCESSORS     00000000-XR                 259    (3)    #-416    (3)
```

```
MP$GR_IDLING_PROCESSORS          00000000-XR                  260    (3)    #-417    (3)
MP$_CLEAR_LOCKS                  00000000-XR                  258    (3)      415    (3)
OFF                             =00000000      282    (3)
ON                              =00000001      282    (3)
POLL_TIME                        00000010-R    295    (3)
PRLIST                           00000450-R    789    (3)  #-1004    (4)  #-1106    (4)  #-523    (3)  #-537    (3)
                                                           #-645     (3)
QA$AOB_CHECK_STATE               00000000-XR                  603    (3)      606    (3)      659    (3)      662    (3)
                                                              855    (4)
QA$AOB_FLAGS                     00000000-XR                  612    (3)      668    (3)
QA$K_ABORT_NOW                  =00000007      282    (3)
QA$K_APT_PHASE_ONE              =0000000B      282    (3)
QA$K_APT_PHASE_TWO              =0000000C      282    (3)
QA$K_BAD_ADDRESS                =00000009      282    (3)
QA$K_CONTROL_C_CONT             =0000000A      282    (3)
QA$K_DEALLOCATION               =0000000E      282    (3)
QA$K_ERROR_PHASE_ONE            =00000005      282    (3)  #-605    (3)  #-661    (3)  #-854    (4)
QA$K_ERROR_PHASE_THREE          =00000007      282    (3)
QA$K_ERROR_PHASE_TWO            =00000006      282    (3)
QA$K_FAILURE                    =00000000      282    (3)
QA$K_FIRST_BRANCH_CODE          =00000000      282    (3)
QA$K_FIRST_TIME                 =00000005      282    (3)
QA$K_INIT_CONTEXT_DONE          =00000003      282    (3)
QA$K_LAST_BRANCH_CODE           =00000025      282    (3)
QA$K_LOOP_ON_SUBTEST            =00000003      282    (3)  #-602    (3)  #-658    (3)
QA$K_LOOP_ON_TEST               =00000002      282    (3)
QA$K_LOOP_TEST_DONE             =00000004      282    (3)
QA$K_MEMORY_MANAGE              =0000000D      282    (3)
QA$K_MULTIPLE_PASS              =00000001      282    (3)
QA$K_NODEF                      =00000000      282    (3)
QA$K_NORMAL_START               =00000000      282    (3)
QA$K_NO_HEADER                  =00000006      282    (3)  #-611    (3)  #-667    (3)
QA$K_NUMBER_OF_CHECKS           =0000000F      282    (3)
QA$K_NUMBER_QA_FLAGS            =00000008      282    (3)
QA$K_NUMB_ROUTINE_STATES        =00000004      282    (3)
QA$K_QA_DEBUG                   =00000001      282    (3)
QA$K_QA_DONE                    =00000002      282    (3)
QA$K_RUN_BACKWARDS              =00000004      282    (3)
QA$K_SECTION                    =00000008      282    (3)
QA$K_SUCCESS                    =00000001      282    (3)
QA$MAIN                          00000000-XR                  435    (3)      466    (3)      598    (3)
QA$V_CHECK_DONE                 =00000003      282    (3)
QA$V_DOING_CLEANUP              =00000002      282    (3)
QA$V_ERROR_FOUND                =00000001      282    (3)
QA$V_INIT_DONE                  =00000000      282    (3)
QIO$ADDUNIT                      00000000-XR                  714    (3)
QIO$CLEANUP                      00000000-XR                  476    (3)
SCB$L_TIMER                      000000C0                   #-703    (3)
SCRIPT$FLUSH                     00000000-XR                  451    (3)      886    (4)
SEP_FUNCTIONAL                  =00000002      273    (3)  #-706    (3)
SEP_REPAIR                      =00000003      273    (3)
SHOWTESTS_X                      00000687-R    1045   (4)  #-1020   (4)
SIZ...                          =00000001      280    (3)    273    (3)      280    (3)
START1                           000000A3-R    443    (3)
TRUE                            =00000001      282    (3)
VRABORT                          000004BD-R    866    (4)
VRABORT_COM                      000004D1-R    880    (4)  #-877    (4)
```

```
VRRESTART                    000000A3-R   442    (3)    #-861    (4)
VRSHOWSECTIONS               00000688-R   1087   (4)
VRSTART                      00000000-R   413    (3)
VRSTART_X                    00000422-R   754    (3)    #-425    (3)    #-460    (3)    #-525    (3)    #-539    (3)
                                                        #-591    (3)    #-647    (3)    #-716    (3)
VRSUPPORT                    000005C2-R   979    (4)
```

```
                                  +-------------------------+
                                  ! Macros Cross Reference !
                                  +-------------------------+

MACRO            SIZE        DEFINITION      REFERENCES...
-----            ----        ----------      -------------
$D1_PRINT_S      1           424    (3)      1019   (4)      1039   (4)      1043   (4)      1093   (4)      1104   (4)
                                            424    (3)      459    (3)      521    (3)      535    (3)      590    (3)
                                            622    (3)      643    (3)      907    (4)      915    (4)      925    (4)
                                            932    (4)      984    (4)      999    (4)
$DEF             1           283    (3)
$DEFINI          1           270    (3)      270    (3)      272    (3)      274    (3)      275    (3)      276    (3)
$DS_CFDEF        1           270    (3)      270    (3)
$DS_CLIDEF       1           271    (3)      271    (3)
$DS_CLRVEC_S     1           703    (3)      703    (3)
$DS_DSADEF       5           272    (3)      272    (3)
$DS_ENVDEF       2           273    (3)      273    (3)
$DS_GPHARD_S     1           671    (3)      671    (3)      710    (3)
$DS_HDRDEF       2           274    (3)      274    (3)
$DS_PRINTF_S     1           1043   (4)      1043   (4)
$DS_QADEF        2           282    (3)      282    (3)
$DS_SCBDEF       3           275    (3)      275    (3)
$DS_TYPEDEF      4           283    (3)      283    (3)
$EQU             1           283    (3)      271    (3)      273    (3)      281    (3)      282    (3)      283    (3)
$EQULS1          1           283    (3)      271    (3)      273    (3)      281    (3)      282    (3)      283    (3)
$EQULST          1           271    (3)      271    (3)      273    (3)      281    (3)      282    (3)      283    (3)
$GBLINI          2                           271    (3)      273    (3)      280    (3)      281    (3)      282    (3)
                                            283    (3)
$PRDEF           5           276    (3)      276    (3)
$PRINT           2           422    (3)      1017   (4)      1035   (4)      1091   (4)      1101   (4)      422    (3)
                                            457    (3)      519    (3)      533    (3)      587    (3)      615    (3)
                                            641    (3)      902    (4)      910    (4)      921    (4)      928    (4)
                                            982    (4)      997    (4)
$VIELD           1                           273    (3)      280    (3)
$VIELD1          1           283    (3)      273    (3)      280    (3)
APTDEF           1           277    (3)      277    (3)
BR_IF_APT        1           427    (3)      427    (3)
BR_IF_NOT_MP     1           414    (3)      414    (3)
CLIDEF           4           278    (3)      278    (3)
DSFDEF           3           280    (3)      280    (3)
DSPDEF           1           279    (3)      279    (3)
DSQA             2           281    (3)      281    (3)
DS_QADEFS        6           282    (3)      282    (3)
MODNAM           1           302    (3)      302    (3)
QA_MAIN          1           435    (3)      435    (3)      466    (3)      598    (3)
```

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	809	(3)												
ADDL2	00C0	804	(3)												
AOBLSS	00F2	448	(3)	717	(3)										
BBC	00E1	428	(3)	600	(3)	656	(3)	699	(3)	768	(3)	852	(4)	854	(4)
		934	(4)												
BBCC	00E5	775	(3)												
BBCS	00E3	509	(3)	514	(3)	770	(3)								
BBS	00E0	1015	(4)	1089	(4)	420	(3)	427	(3)	474	(3)	531	(3)	602	(3)
		605	(3)	611	(3)	658	(3)	661	(3)	667	(3)	696	(3)	891	(4)
		936	(4)	980	(4)										
BBSC	00E4	899	(4)	918	(4)										
BEQL	0013	1033	(4)	414	(3)	456	(3)	508	(3)	898	(4)				
BGEQ	0018	639	(3)												
BGTR	0014	578	(3)	581	(3)										
BICL	00CA	942	(4)												
BICL2	00CA	432	(3)	546	(3)	755	(3)	857	(4)	887	(4)				
BISL	00C8	731	(3)												
BISL2	00C8	745	(3)												
BLBC	00E9	712	(3)												
BLBS	00E8	715	(3)												
BLEQU	001B	877	(4)												
BLSS	0019	497	(3)	565	(3)	571	(3)								
BNEQ	0012	1026	(4)	1100	(4)	414	(3)	485	(3)	499	(3)	503	(3)	529	(3)
		566	(3)	572	(3)	633	(3)	707	(3)	734	(3)	991	(4)	995	(4)
BRB	0011	585	(3)	608	(3)	664	(3)	792	(3)	864	(4)	908	(4)	916	(4)
		926	(4)												
BRW	0031	1020	(4)	425	(3)	460	(3)	486	(3)	512	(3)	517	(3)	525	(3)
		539	(3)	591	(3)	634	(3)	647	(3)	716	(3)	735	(3)	861	(4)
		893	(4)												
BSBW	0030	1004	(4)	1106	(4)	434	(3)	463	(3)	523	(3)	537	(3)	645	(3)
CALLS	00FB	1019	(4)	1039	(4)	1043	(4)	1093	(4)	1104	(4)	424	(3)	435	(3)
		459	(3)	466	(3)	476	(3)	521	(3)	535	(3)	590	(3)	598	(3)
		622	(3)	636	(3)	643	(3)	671	(3)	678	(3)	684	(3)	689	(3)
		698	(3)	703	(3)	710	(3)	714	(3)	739	(3)	807	(3)	878	(4)
		907	(4)	915	(4)	925	(4)	932	(4)	984	(4)	999	(4)		
CLRB	0094	1005	(4)	1107	(4)	447	(3)	524	(3)	538	(3)	646	(3)		
CLRL	00D4	1029	(4)	416	(3)	417	(3)	418	(3)	419	(3)	483	(3)	491	(3)
		493	(3)	560	(3)	631	(3)	708	(3)						
CMPB	0091	496	(3)	498	(3)	502	(3)								
CMPL	00D1	414	(3)	455	(3)	576	(3)	579	(3)	876	(4)	897	(4)		
CMPZV	00ED	1022	(4)	1096	(4)	705	(3)	987	(4)						
CVTBL	0098	1019	(4)	1039	(4)	1093	(4)	1104	(4)	424	(3)	459	(3)	521	(3)
		535	(3)	590	(3)	622	(3)	643	(3)	806	(3)	907	(4)	915	(4)
		925	(4)	932	(4)	984	(4)	999	(4)						
EXTZV	00EF	452	(3)												
INCL	00D6	1040	(4)	500	(3)	567	(3)								
JMP	0017	1041	(4)	756	(3)	947	(4)								
JSB	0016	415	(3)	430	(3)	431	(3)	451	(3)	462	(3)	472	(3)	701	(3)
		730	(3)	747	(3)	886	(4)	894	(4)	939	(4)	940	(4)		
MOVAB	009E	1031	(4)												

MOVAL	00DE	445	(3)	446	(3)	482	(3)	767	(3)					
MOVB	0090	1002	(4)	795	(3)	798	(3)							
MOVL	00D0	1105	(4)	488	(3)	489	(3)	492	(3)	494	(3)	495	(3)	516	(3)
		522	(3)	536	(3)	558	(3)	559	(3)	561	(3)	563	(3)	569	(3)
		573	(3)	583	(3)	644	(3)	711	(3)	790	(3)	799	(3)	802	(3)
		863	(4)	867	(4)	945	(4)	992	(4)	993	(4)				
MOVQ	007D	733	(3)	742	(3)										
MOVW	00B0	791	(3)	793	(3)	794	(3)								
MOVZBL	009A	1019	(4)	1039	(4)	1093	(4)	1104	(4)	424	(3)	459	(3)	521	(3)
		535	(3)	590	(3)	622	(3)	643	(3)	674	(3)	680	(3)	805	(3)
		808	(3)	907	(4)	915	(4)	925	(4)	932	(4)	984	(4)	999	(4)
MULL3	00C5	1030	(4)	797	(3)										
POPR	00BA	1042	(4)	449	(3)	740	(3)	896	(4)						
PUSHAB	009F	1019	(4)	1039	(4)	1043	(4)	1093	(4)	1104	(4)	424	(3)	459	(3)
		521	(3)	535	(3)	590	(3)	622	(3)	643	(3)	675	(3)	681	(3)
		686	(3)	907	(4)	915	(4)	925	(4)	932	(4)	984	(4)	999	(4)
PUSHAL	00DF	671	(3)	710	(3)	738	(3)	803	(3)						
PUSHL	00DD	1039	(4)	1104	(4)	435	(3)	466	(3)	590	(3)	598	(3)	622	(3)
		671	(3)	676	(3)	677	(3)	682	(3)	683	(3)	687	(3)	688	(3)
		703	(3)	710	(3)	713	(3)	800	(3)	907	(4)	915	(4)	925	(4)
		932	(4)												
PUSHR	00BB	1028	(4)	444	(3)	737	(3)	884	(4)						
RSB	0005	1000	(4)	1006	(4)	1046	(4)	1094	(4)	1108	(4)	773	(3)	779	(3)
		810	(3)	985	(4)										
SBWC	00D9	744	(3)												
SOBGEQ	00F4	672	(3)	685	(3)										
SOBGTR	00F5	504	(3)	527	(3)	796	(3)	801	(3)						
SUBL2	00C2	743	(3)												
SUBL3	00C3	505	(3)	638	(3)	679	(3)								
TSTL	00D5	1032	(4)	484	(3)	507	(3)	528	(3)	632	(3)	994	(4)		

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	302	(3)	305	(3)	309	(3)	312	(3)	315	(3)	318	(3)	321	(3)	324	(3)
	327	(3)	330	(3)	333	(3)	336	(3)	339	(3)	342	(3)	345	(3)	347	(3)
	349	(3)	352	(3)	356	(3)										
.BLKL	278	(3)	279	(3)	297	(3)										
.BLKQ	291	(3)	293	(3)	295	(3)										
.END	1109	(4)														
.ENDC	1019	(4)	1039	(4)	1043	(4)	1093	(4)	1104	(4)	271	(3)	273	(3)	280	(3)
	281	(3)	282	(3)	283	(3)	424	(3)	435	(3)	459	(3)	466	(3)	521	(3)
	535	(3)	590	(3)	598	(3)	622	(3)	643	(3)	907	(4)	915	(4)	925	(4)
	932	(4)	984	(4)	999	(4)										
.ENDM	1043	(4)	270	(3)	271	(3)	272	(3)	273	(3)	274	(3)	275	(3)	276	(3)
	277	(3)	278	(3)	279	(3)	280	(3)	281	(3)	282	(3)	283	(3)	302	(3)
	414	(3)	422	(3)	424	(3)	427	(3)	435	(3)	671	(3)	703	(3)		
.ENDR	1019	(4)	1039	(4)	1043	(4)	1093	(4)	1104	(4)	271	(3)	273	(3)	280	(3)
	281	(3)	282	(3)	283	(3)	424	(3)	459	(3)	521	(3)	535	(3)	590	(3)
	622	(3)	643	(3)	907	(4)	915	(4)	925	(4)	932	(4)	984	(4)	999	(4)
.ENTRY	850	(4)														
.EXTRN	256	(3)	257	(3)												
.IDENT	14	(1)														
.IF	1019	(4)	1039	(4)	1043	(4)	1093	(4)	1104	(4)	271	(3)	273	(3)	280	(3)
	281	(3)	282	(3)	283	(3)	424	(3)	435	(3)	459	(3)	466	(3)	521	(3)
	535	(3)	590	(3)	598	(3)	622	(3)	643	(3)	907	(4)	915	(4)	925	(4)
	932	(4)	984	(4)	999	(4)										
.IFF	271	(3)	273	(3)	280	(3)	281	(3)	282	(3)	283	(3)				
.IF_FALSE	435	(3)	466	(3)	598	(3)										
.IIF	271	(3)	273	(3)	280	(3)	281	(3)	282	(3)	283	(3)				
.IRP	1019	(4)	1039	(4)	1043	(4)	1093	(4)	1104	(4)	271	(3)	273	(3)	280	(3)
	281	(3)	282	(3)	283	(3)	424	(3)	459	(3)	521	(3)	535	(3)	590	(3)
	622	(3)	643	(3)	907	(4)	915	(4)	925	(4)	932	(4)	984	(4)	999	(4)
.LIBRARY	245	(3)	246	(3)	247	(3)										
.LIST	269	(3)	270	(3)	272	(3)	274	(3)	275	(3)	278	(3)	279	(3)	286	(3)
	287	(3)														
.MACRO	271	(3)	273	(3)	280	(3)	281	(3)	282	(3)	283	(3)				
.MDELETE	271	(3)	273	(3)	282	(3)										
.NLIST	267	(3)	268	(3)	270	(3)	272	(3)	274	(3)	275	(3)	278	(3)	279	(3)
	285	(3)														
.NOCROSS	270	(3)	272	(3)	274	(3)	275	(3)	276	(3)						
.NOSHOW	15	(1)														
.PAGE	1009	(4)	1048	(4)	1086	(4)	240	(3)	288	(3)	299	(3)	325	(3)	357	(3)
	407	(3)	436	(3)	467	(3)	477	(3)	526	(3)	540	(3)	553	(3)	592	(3)
	625	(3)	648	(3)	691	(3)	718	(3)	749	(3)	757	(3)	780	(3)	811	(3)
	948	(4)	978	(4)												
.PSECT	278	(3)	279	(3)	289	(3)	300	(3)	359	(3)						
.RESTORE	270	(3)	272	(3)	274	(3)	275	(3)	276	(3)	278	(3)	279	(3)		
.SAVE	270	(3)	272	(3)	274	(3)	275	(3)	276	(3)	278	(3)	279	(3)		
.SBTTL	1010	(4)	1049	(4)	241	(3)	358	(3)	812	(3)	949	(4)				
.TITLE	13	(1)														
.WEAK	258	(3)	259	(3)	260	(3)	261	(3)	262	(3)						

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	1	#-867	(4)										
FP	1	#-863	(4)										
R0	35	#-1105	(4)	#-454	(3)	#-455	(3)	#-522	(3)	#-536	(3)	#-558	(3)
		#-559	(3)	#-644	(3)	#-674	(3)	#-678	(3)	#-680	(3)	#-684	(3)
		#-712	(3)	#-715	(3)	#-737	(3)	#-740	(3)	#-767	(3)	769	(3)
		#-790	(3)	#-797	(3)	#-799	(3)	#-800	(3)	#-802	(3)	#-808	(3)
		#-809	(3)	#-863	(4)	#-867	(4)	#-876	(4)	#-884	(4)	#-896	(4)
		#-897	(4)	#-907	(4)	#-915	(4)	#-992	(4)	#-993	(4)		
R1	21	#-560	(3)	#-622	(3)	#-711	(3)	#-713	(3)	#-737	(3)	#-740	(3)
		#-790	(3)	#-796	(3)	#-797	(3)	#-798	(3)	#-799	(3)	#-800	(3)
		#-801	(3)	#-802	(3)	803	(3)	#-804	(3)	#-807	(3)	#-993	(4)
		#-994	(4)										
R10	3	#-491	(3)	#-516	(3)	#-528	(3)						
R2	11	#-482	(3)	#-484	(3)	#-494	(3)	#-495	(3)	510	(3)	515	(3)
		532	(3)	#-561	(3)	#-563	(3)	#-569	(3)	#-583	(3)		
R3	5	#-444	(3)	#-445	(3)	#-447	(3)	#-448	(3)	#-449	(3)		
R4	20	#-1028	(4)	#-1029	(4)	#-1030	(4)	#-1040	(4)	#-1042	(4)	#-444	(3)
		#-446	(3)	#-448	(3)	#-449	(3)	#-488	(3)	#-489	(3)	#-492	(3)
		#-638	(3)	#-671	(3)	#-672	(3)	#-679	(3)	#-685	(3)	#-708	(3)
		#-710	(3)	#-717	(3)								
R5	11	#-1028	(4)	#-1030	(4)	1031	(4)	#-1032	(4)	#-1039	(4)	#-1042	(4)
		#-489	(3)	#-505	(3)	#-527	(3)						
R6	4	#-492	(3)	#-496	(3)	#-498	(3)	#-502	(3)				
R7	4	#-494	(3)	#-496	(3)	#-498	(3)	#-504	(3)				
R8	2	#-495	(3)	#-502	(3)								
R9	3	#-493	(3)	#-500	(3)	#-507	(3)						
SP	47	#-1019	(4)	#-1039	(4)	#-1093	(4)	#-1104	(4)	#-424	(3)	#-459	(3)
		#-521	(3)	#-535	(3)	#-590	(3)	#-622	(3)	#-643	(3)	671	(3)
		710	(3)	#-711	(3)	#-791	(3)	#-793	(3)	#-794	(3)	#-795	(3)
		#-798	(3)	803	(3)	#-805	(3)	#-806	(3)	#-808	(3)	#-809	(3)
		#-907	(4)	#-915	(4)	#-925	(4)	#-932	(4)	#-984	(4)	#-999	(4)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.02	00:00:00.21
Command processing	901	00:00:00.26	00:00:00.67
Pass 1	1019	00:00:03.51	00:00:05.93
Symbol table sort	0	00:00:00.18	00:00:00.17
Pass 2	73	00:00:00.92	00:00:01.76
Symbol table output	2	00:00:00.08	00:00:00.25
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	7	00:00:00.50	00:00:00.77
Assembler run totals	2028	00:00:05.48	00:00:09.78

The working set limit was 2900 pages.
202995 bytes (397 pages) of virtual memory were used to buffer the intermediate code.

```
There were 40 pages of symbol table space allocated to hold 645 non-local and 59 local symbols.
1109 source lines were read in Pass 1, producing 64 object records in Pass 2.
118 pages of virtual memory were used to define 31 macros.
```

Macro library statistics

Macro library name	Macros defined
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	12
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	11
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5	0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6	0
SYS$COMMON:[SYSLIB]LIB.MLB;2	0
SYS$COMMON:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	29

```
808 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:START.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:D
```

Table of contents

(2) 49 DECLARATIONS
(3) 315 TS11/TS04, TU80 Bootstrap driver

```
0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element TSBTDRIVR.MAR
0000     2 ; *1     6-FEB-1986 15:22:48 DS ""
0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element TSBTDRIVR.MAR
0000     4         .TITLE  TSBTDRIVR      *** TSBTDRIVR driver for TS11/TS04, TU80 magtape
0000     5         .IDENT  /V01-02/
0000     6         .DISABLE GLOBAL
0000     7
0000     8 ;
0000     9 ;                                COPYRIGHT (c) 1980,1982 BY
0000    10 ;                        DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
0000    11 ;
0000    12 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
0000    13 ; ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
0000    14 ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
0000    15 ; COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000    16 ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
0000    17 ; TRANSFERRED.
0000    18 ;
0000    19 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
0000    20 ; AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
0000    21 ; CORPORATION.
0000    22 ;
0000    23 ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
0000    24 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000    25 ;
0000    26
0000    27 ;++
0000    28 ; FACILITY:     DIAGNOSTIC SUPERVISOR
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;       This module contains the file load driver for
0000    32 ;       UNIBUS TS11/TS04 and TU80 magtape
0000    33 ;
0000    34 ; ENVIRONMENT:  kernel mode
0000    35 ;
0000    36 ; AUTHOR:  Roger Riggs, CREATION DATE:  2-OCT-1980
0000    37 ;
0000    38 ; MODIFIED BY:
0000    39 ;
0000    40 ;       01      Bob Bergazzi    Aug 30, 1982    v6.9
0000    41 ;               Changed MS$CMD to reside all on one page. Routine
0000    42 ;               MAP_VA won't work if MS$CMD is split over a page boundary
0000    43 ;
0000    44 ;       02      Marion Baggett  Dec 01, 1982    V6.10
0000    45 ;               Added comment changes for TU80.
0000    46
0000    47 ;--
```

```
                     0000     49          .SBTTL  DECLARATIONS
                     0000     50 ;
                     0000     51 ; INCLUDE FILES:
                     0000     52 ;
                     0000     53
                     0000     54          $BTDDEF                         ; Define boot device types
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     55          $IODEF                          ; Define I/O function codes
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     56          $NDTDEF                         ; Define NEXUS device types
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     57          $PTEDEF                         ; Define page table entry
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     58          $PRDEF                          ; Define processor registers
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     59          $RPBDEF                         ; Define RPB offsets
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     60          $SSDEF                          ; Define system status codes
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     61          $VADEF                          ; Define virtural address fields
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .RESTORE
                     0000     62
                     0000     63 ;
                     0000     64 ; MACROS:
                     0000     65 ;
                     0000     66
                     0000     67
                     0000     68 ;
                     0000     69 ; EQUATED SYMBOLS:
                     0000     70 ;
000000B1             0000     71 BTD$K_TM = 177 ; Temp
000000B2             0000     72 btd$k_TS = 178 ; Temp
                     0000     73
                     0000     74 ;
                     0000     75 ; LOCAL SYMBOLS
                     0000     76 ;
                     0000     77
                     0000     78 ;
                     0000     79 ; TS11/TS04, TU80 COMMAND PACKET DEFINITION
                     0000     80 ;
                     0000     81 ;
                     0000     82
                     0000     83          $DEFINI MS
                     0000                 .SAVE   LOCAL_BLOCK
                     0000     84
                     0000     85
00000000             0000     86 .=0                                      ;RESET PC??????
                     0000     87 $DEF     MS_CPHD .BLKW   1               ;COMMAND PACKET HEADER
                     0000                 .IIF    NB,MS_CPHD,     MS_CPHD:
```

```
                00000002  0000                .IIF    NB,..BLKW,      .BLKW   1
                          0002    88          _VIELD  MS_CPHD,0,<-            ;
                          0002    89                  <COD,5>,-               ;COMMAND CODE FIELD
                          0002    90                  <,2>,-                  ;B5-B6 ALWAYS 0 FOR TS04
                          0002    91                  <IE,,M>,-               ;INTERRUPT ENABLE
                          0002    92                  <MOD,4>,-               ;COMMAND MODE FIELD(B11-B8)
                          0002    93                      -                   ; B8=REVERSE & B9=RETRY
                          0002    94                  <SWB,,M>,-              ;SWAP BYTES BIT(B12)
                          0002    95                  <OPP,,M>,-              ;OPPOSITE BIT(B13)
                          0002    96                  <CVC,,M>,-              ;CLEAR VOLUME CHECK(B14)
                          0002    97                  <ACK,,M>,-              ;ACKNOWLEDGE BIT(B15)
                          0002    98                  >
                          0002    99 $DEF     MS_BACT .BLKW   1               ;BUS ADDRESS(B15-B0) OR COUNT
                          0002                .IIF    NB,MS_BACT,     MS_BACT:
                00000004  0002                .IIF    NB,..BLKW,      .BLKW   1
                          0004   100 $DEF     MS_BA1  .BLKW   1               ;BUS ADDRESS B17-B16(RIGHT JUST)
                          0004                .IIF    NB,MS_BA1,      MS_BA1:
                00000006  0004                .IIF    NB,..BLKW,      .BLKW   1
                          0006   101 $DEF     MS_CNT  .BLKW   1               ;BYTE COUNT
                          0006                .IIF    NB,MS_CNT,      MS_CNT:
                00000008  0006                .IIF    NB,..BLKW,      .BLKW   1
                          0008   102                                          ;FOR WRITE CHARACTERISTIC DATA
                          0008   103 $DEF     MS_MBA0 .BLKW   1               ;MESSAGE BUFFER ADDR. WRD 1
                          0008                .IIF    NB,MS_MBA0,     MS_MBA0:
                0000000A  0008                .IIF    NB,..BLKW,      .BLKW   1
                          000A   104 $DEF     MS_MBA1 .BLKW   1               ;MESSAGE BUFFER ADDR. WRD 2
                          000A                .IIF    NB,MS_MBA1,     MS_MBA1:
                0000000C  000A                .IIF    NB,..BLKW,      .BLKW   1
                          000C   105 $DEF     MS_LNTH .BLKW   1               ;MESSAGE BUFFER LENGTH(ALWAYS 14.)
                          000C                .IIF    NB,MS_LNTH,     MS_LNTH:
                0000000E  000C                .IIF    NB,..BLKW,      .BLKW   1
                          000E   106 $DEF     MS_CHWD .BLKW   1               ;CHARACTERISTIC WORD
                          000E                .IIF    NB,MS_CHWD,     MS_CHWD:
                00000010  000E                .IIF    NB,..BLKW,      .BLKW   1
                          0010   107          _VIELD  MS_CHWD,4,<-            ;
                          0010   108                  <ERI,,M>,-              ;ENABLE MESSAGE BUFFER RELEASE INTERRUPTS
                          0010   109                  <EAI,,M>,-              ;ENABLE ATTENTION INTERRUPTS
                          0010   110                  <ENB,,M>,-              ;USED WITH ESS BIT***
                          0010   111                  <ESS,,M>,-              ;ENABLE SKIP TAPE MARKS STOP
                          0010   112                  >
                          0010   113
                          0010   114 ;
                          0010   115 ; TS11/TS04,TU80  MESSAGE PACKET DEFINITION
                          0010   116 ;
                          0010   117
                          0010   118
                          0010   119 $DEF     MS_MHD  .BLKW   1               ;MESSAGE PACKET
                          0010                .IIF    NB,MS_MHD,      MS_MHD:
                00000012  0010                .IIF    NB,..BLKW,      .BLKW   1
                          0012   120          _VIELD  MS_MHD,0,<-             ;MESSAGE HEADER WORD
                          0012   121                  <COD,5>,-               ;MESSAGE CODE FIELD
                          0012   122                  <FMT,3>,-               ;FORMAT FIELD
                          0012   123                  <CLS,4>,-               ;CLASS CODE FIELD
                          0012   124                  <RSR,3>,-               ;RESERVED FIELD
                          0012   125                  <ACK,,M>,-              ;MESSAGE ACKNOWLEDGE BIT(B15)
                          0012   126                  >                       ;
                          0012   127 $DEF     MS_LNH  .BLKW   1               ;MESSAGE LENGTH WORD
```

```
                     0012                .IIF    NB,MS_LNH,      MS_LNH:
00000014             0012                .IIF    NB,,BLKW,       .BLKW   1
                     0014    128                                         ;HIGH BYTE=0,LOW BYTE=1010(LENGTH)
                     0014    129 $DEF    MS_RBPC .BLKW   1               ;RESIDUAL BYTE/POSITION COUNT
                     0014                .IIF    NB,MS_RBPC,     MS_RBPC:
00000016             0014                .IIF    NB,,BLKW,       .BLKW   1
                     0016    130 $DEF    MS_XSR0 .BLKW   1               ;EXTENDED STATUS REGISTER 0
                     0016                .IIF    NB,MS_XSR0,     MS_XSR0:
00000018             0016                .IIF    NB,,BLKW,       .BLKW   1
                     0018    131         _VIELD  MS_XSR0,0,<-            ;
                     0018    132                 <EOT,,M>,-              ;END OF TAPE DETECTED(B0)
                     0018    133                 <BOT,,M>,-              ;BEGINNING OF TAPE(B1)
                     0018    134                 <WLK,,M>,-              ;WRITE LOCKED(B2)
                     0018    135                 <PED,,M>,-              ;PHASE ENCODED DRIVE(B3)
                     0018    136                 <VCK,,M>,-              ;VOLUME CHECK(B4)
                     0018    137                 <IE,,M>,-               ;INTERRUPT WAS ENABLED(B5)
                     0018    138                 <ONL,,M>,-              ;DEVICE ON-LINE(B6)
                     0018    139                 <MOT,,M>,-              ;TAPE MOVING ON LAST COMMAND(B7)
                     0018    140                 <ILA,,M>,-              ;ILLEGAL ADDRESS(B8)
                     0018    141                 <ILC,,M>,-              ;ILLEGAL COMMAND(B9)
                     0018    142                 <NEF,,M>,-              ;NON-EXECUTALBE FUNCTION(B10)
                     0018    143                 <WLE,,M>,-              ;WRITE LOCK ERROR(B11)
                     0018    144                 <RLL,,M>,-              ;RECORD LENGTH LONG(B12)
                     0018    145                 <LET,,M>,-              ;LOGICAL END OF TAPE(B13)
                     0018    146                 <RLS,,M>,-              ;RECORD LENGTH SHORT(B14)
                     0018    147                 <TMK,,M>,-              ;TAPE MARK DETECTED(B15)
                     0018    148         >                               ;
                     0018    149 $DEF    MS_XSR1 .BLKW   1               ;EXTENDED STATUS REGISTER 1
                     0018                .IIF    NB,MS_XSR1,     MS_XSR1:
0000001A             0018                .IIF    NB,,BLKW,       .BLKW   1
                     001A    150         _VIELD  MS_XSR1,0,<-            ;
                     001A    151                 <MTE,,M>,-              ;(PE) MULTI-TRACK ERROR
                     001A    152                         -               ;(NRZ) VERTICAL PARITY ERROR
                     001A    153                 <UNC,,M>,-              ;(PE) UNCORRECTABLE DATA ERROR(B1)
                     001A    154                         -               ;(NRZ) CYCLIC REDUNDANCY CHECK ERROR
                     001A    155                 <POL,,M>,-              ;(PE) POSTAMBLE LONG(B2)
                     001A    156                         -               ;(NRZ) LONGITUDINAL REDUNDANCY CHECK ERROR
                     001A    157                 <POS,,M>,-              ;(PE) POSTAMBLE SHORT(B3)
                     001A    158                         -               ;(NRZ) NOISE RECORD
                     001A    159                 <IED,,M>,-              ;(PE) INVALID END DATA(B4)
                     001A    160                         -               ;(NRZ) LRC WAS 0.
                     001A    161                 <IPO,,M>,-              ;(PE) INVALID POSTAMBLE(B5)
                     001A    162                         -               ;(NRZ) ILLEGAL TAPE MARK
                     001A    163                 <SYN,,M>,-              ;(PE) SYNCH ERROR(B6)
                     001A    164                         -               ;(NRZ) FRAME DROPOUT
                     001A    165                 <IPR,,M>,-              ;(PE) INVALID PREAMBLE(B7)
                     001A    166                 <,1>,-                  ;RESERVED BIT
                     001A    167                 <SCK,,M>,-              ;SPEED CHECK(B9)
                     001A    168                 <DBF,,M>,-              ;(PE) DESKEW BUFFER FAIL(B10)
                     001A    169                         -               ;(NRZ) NRZ BOARD FIFO OVERFLOW
                     001A    170                 <TIG,,M>,-              ;TRASH IN GAP(B11)
                     001A    171                 <CRS,,M>,-              ;CREASE DETECTED(B12)
                     001A    172                 <COR,,M>,-              ;CORRCTABLE DATA(B13)
                     001A    173                 <,1>,-                  ;UNUSED BIT(B14)
                     001A    174                 <DLT,,M>,-              ;DATA LATE(B15)
                     001A    175         >
                     001A    176 $DEF    MS_XSR2 .BLKW   1               ;EXTENDED STATUS REGISTER 2
```

```
                    001A                .IIF    NB,MS_XSR2,     MS_XSR2:
          0000001C  001A                .IIF    NB,,BLKW,       .BLKW   1
                    001C   177          _VIELD  MS_XSR2,0,<-            ;
                    001C   178                  <DTP,8>,-               ;DEAD TRACK PARITY,B7-B0
                    001C   179                  <XSK,,M>,-              ;EXCESSIVE SKEW(B9)
                    001C   180                  <WCF,,M>,-              ;WRITE CLOCK FAIL(B10),BROKEN HARDWARE
                    001C   181                  <,1>,-                  ;B11 NOT USED
                    001C   182                  <CAF,,M>,-              ;CAPSTAN ACCELERATION FAIL(B12)
                    001C   183                  <BPE,,M>,-              ;SERIAL BUS PARITY ERROR AT DRIVE(B13)
                    001C   184                  <SIP,,M>,-              ;SILO PARITY ERROR(B14)
                    001C   185                  <OPM,,M>,-              ;OPERATION IN PROGRESS(B15)
                    001C   186          >
                    001C   187 $DEF     MS_XSR3 .BLKW   1               ;EXTENDED STATUS REGISER 3
                    001C                .IIF    NB,MS_XSR3,     MS_XSR3:
          0000001E  001C                .IIF    NB,,BLKW,       .BLKW   1
                    001E   188          _VIELD  MS_XSR3,0,<-            ;
                    001E   189                  <RIB,,M>,-              ;REVERSE INTO BOT(B0)
                    001E   190                  <LXS,,M>,-              ;LIMIT EXCEEDED STATICALLY(B1)
                    001E   191                  <NOI,,M>,-              ;NOISE RECORD(B2)
                    001E   192                  <DCK,,M>,-              ;DENSITY CHECK(B3)
                    001E   193                  <CRF,,M>,-              ;CAPSTAN RESPONSE FAIL(B4)
                    001E   194                  <REV,,M>,-              ;TAPE MOVED BACKWARDS(B5)
                    001E   195                  <OPI,,M>,-              ;OPERATION IN COMPLETE(B6)
                    001E   196                  <LMX,,M>,-              ;TAPE LIMIT EXCEEDED(B7)
                    001E   197                  <FEC,8>,-               ;B15-B8, FATAL ERROR CODE(U-DIAGNOSTIC)
                    001E   198          >
                    001E   199
                    001E   200          $DEFEND MS
                    0000                .RESTORE
                    0000   201
                    0000   202 ;
                    0000   203 ; TS11/TS04,TU80 TSSR TERMINATION CLASS CODES
                    0000   204 ;
                    0000   205
          00000000  0000   206 TCC_NML=0                                ;NORAML TERMINATION
          00000001  0000   207 TCC_ATN=1                                ;ATTENTION CONDITION
          00000002  0000   208 TCC_TSA=2                                ;TAPE STATUS ALERT
          00000003  0000   209 TCC_FNR=3                                ;FUNCTION REJECT
          00000004  0000   210 TCC_REM=4                                ;RECOVERABLE ERROR(TAPE MOVED)
          00000005  0000   211 TCC_REN=5                                ;RECOVERABLE ERROR(TAPE NOT MOVED)
          00000006  0000   212 TCC_UER=6                                ;UNRECOVERABLE ERROR(TAPE POSI. LOST)
          00000007  0000   213 TCC_FTL=7                                ;FATAL CONTROLLER ERROR
                    0000   214
                    0000   215 ;
                    0000   216 ; FATAL CLASS (FC) CODES IN TSSR
                    0000   217 ;
                    0000   218
          00000000  0000   219 FCC_IDF=0                                ;INTERNAL DIAG. FAILURE
          00000001  0000   220 FCC_CPE=1                                ;IO SEQUENCE CROM PARITY ERROR
          00000002  0000   221 FCC_UPE=2                                ;U-PROCESSOR CROM PARITY ERROR OR OTHER
          00000003  0000   222 FCC_LAP=3                                ;LOSS OF AC POWER DETECTED
                    0000   223
                    0000   224 ;
                    0000   225 ; TS11/TS04, TU80 MESSAGE CODES IN MS_MHD_COD
                    0000   226 ;
                    0000   227
          00000010  0000   228 MSG_END=^O20                             ;END
```

```
00000011  0000   229 MSG_FAL=^O021                                ;FAIL
00000012  0000   230 MSG_ERR=^O022                                ;ERROR
00000013  0000   231 MSG_ATN=^O023                                ;ATTENTION
00000014  0000   232 MSG_LOG=^O024                                ;LOG (NOT USED)
          0000   233
          0000   234 ;
          0000   235 ; CLASS CODE FOR MESSAGE CODES (MS_MHD_CLS VALUES)
          0000   236 ;
          0000   237
          0000   238                                        ;**WHEN MSG TYPE=ATTENTION**
00000000  0000   239 CLS_ONF=0                                      ;ON OR OFFLINE
00000001  0000   240 CLS_MDF=1                                      ;MICRO DIAG. FAILURE
          0000   241                                        ;**WHEN MSG TYPE=FAIL**
00000000  0000   242 CLS_PTB=0                                      ;PACKET BAD(SERIAL BUS PARITY ERROR)
00000001  0000   243 CLS_OTHER=1                                    ;OTHERS
00000002  0000   244 CLS_WLN=2                                      ;WRITE LOCK ERROR OR NON-EXECUTABLE FUNCTION
00000003  0000   245 CLS_MDE=3                                      ;MICRO DIAGNOSTIC ERROR
          0000   246
          0000   247 ;
          0000   248 ; TS11/TS04, TU80 HARDWARE COMMAND MODES/CODES
          0000   249 ;
          0000   250
          0000   251                                              ; INTERRUPT ENABLE & ACKNOWLEDGE
00000000  0000   252 HC_NOP=0                                ;SIMULATED NOP(REAL NO OPERATION)
00000000  0000   253 HC_PAK=HC_NOP                                ;SIMULATED PACK ACKNOWLEDGE
00000000  0000   254 HC_WCK=HC_NOP                                ;SIMULATED WRITE CHECK
00000000  0000   255 HC_WKR=HC_NOP                                ;SIMULATED WRITE CHECK REVERSE
00000000  0000   256 HC_RPS=HC_NOP                                ;SIMULATED READ IN PRESET
00000000  0000   257 HC_SCH=HC_NOP                                ;SIMULATED SET CHARACTERISTICS
          0000   258
          0000   259
0000C001  0000   260 HC_RDN=^O00001!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;* READ NEXT (FORWARD)
0000C004  0000   261 HC_WRC=^O00004!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;+ WRITE CHARACTERISTICS
0000C008  0000   262 HC_SRF=^O00010!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;$ SPACE RECORDS FORWARD
0000C108  0000   263 HC_SRR=^O00410!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;$ SPACE RECORDS REVERSE
0000C208  0000   264 HC_STF=^O01010!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;$ SKIP TAPE MARKS FORWARD
0000C308  0000   265 HC_STR=^O01410!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;$ SKIP TAPE MARKS REVERSE
0000C408  0000   266 HC_RWD=^O02010!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;$ REWIND
0000C10A  0000   267 HC_UNL=^O00412!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;- REWIND AND UNLOAD
0000C00B  0000   268 HC_DRI=^O00013!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;- DRIVER INITIALIZE
0000C00F  0000   269 HC_GST=^O00017!MS_CPHD_M_CVC!MS_CPHD_M_ACK       ;- GET STATUS IMMEDIATE
          0000   270                                        ;**NOTE**
          0000   271                                        ; * => DATA XFR
          0000   272                                        ; + => (SPECIAL)
          0000   273                                        ; $ => POSITION
          0000   274                                        ; - => FORMAT,CONTROL,INITIALIZE,& STATUS
          0000   275 ;+
          0000   276 ; TSSR defs
          0000   277 ;-
          0000   278
00000000  0000   279 TSDB    = 0                           ; Offset to TSDB
00000000  0000   280 TSBA    = 0                           ; Offset to TSBA
00000002  0000   281 TSSR    = 2                           ; Offset to TSSR
          0000   282
          0000   283 $VIELD  MS_TSSR,1,<-                  ;TS11/TS04, TU80 STATUS REGISTER(B0 UNUSED)
          0000   284         <TCC,3>,-                     ;TERMINATION CLASS CODE FIELD
          0000   285         <FC,2>,-                      ;FATAL ERROR CLASS CODE FIELD
```

```
                0000   286          <OFL,,M>,-                  ;DEVICE IS OFF-LINE(B6)
                0000   287          <SSR,,M>,-                  ;SUBSYSTEM READY(B7)
                0000   288          <A16,,M>,-                  ;BUFFER ADDRESS BIT 16
                0000   289          <A17,,M>,-                  ;BUFFER ADDRESS BIT 17
                0000   290          <NBA,,M>,-                  ;NEED BUFFER ADDRESS(B10)
                0000   291          <NXM,,M>,-                  ;NON-EXISTENT MEMORY(B11)
                0000   292          <RMR,,M>,-                  ;REGISTER MODIFICATION REFUSED(B12)
                0000   293          <SPE,,M>,-                  ;SERIAL BUS PARITY ERROR(B13)
                0000   294          <UPE,,M>,-                  ;UNIBUS PARITY ERROR(B14)
                0000   295          <SC,,M>,-                   ;SPECIAL CONDITION(B15)
                0000   296          >                           ;
                0000   297
                0000   298 ;
                0000   299 ; OWN
                0000   300 ;
            00000000   301          .PSECT  DATA, SHR, NOEXE, NOWRT, 5
        00000020  0000   302 MS$CMD::       .BLKW   16          ; Command packet/char data/message area
                0020   303
                0020   304
                0020   305 ;
                0020   306 ; BOOT DRIVER TABLE ENTRY
                0020   307 ;
                0020   308
                0020   309          $BOOT_DRIVER    DEVTYPE = BTD$K_TS,-    ; Device type (UNIBUS)
                0020   310                          SIZE = TS11_DRVSIZ,-    ; Driver size
                0020   311                          ADDR = TS11_DRIVER,-    ; Driver address
                0020   312                          ENTRY = TS11_DRIVER,-   ; Entry point
                0020   313                          DRIVRNAME = DRIVERNAME  ; Driver name
            00000000            .PSECT  BOOTDRIVR_4
                FFFF  0000      .WORD   -1
                00B2  0002      .WORD   BTD$K_TS
        00000000  0004          .LONG   0
        000001D2' 0008          .LONG   TS11_DRVSIZ
        00000000' 000C          .LONG   TS11_DRIVER-$TABLE
        00000000' 0010          .LONG   TS11_DRIVER-TS11_DRIVER
        000001C5' 0014          .LONG   DRIVERNAME-TS11_DRIVER
            00000000            .PSECT  BOOTDRIVR_2
```

```
0000    315          .SBTTL  TS11/TS04, TU80 Bootstrap driver
0000    316
0000    317 ;++
0000    318 ; Functional description:
0000    319 ;
0000    320 ;       This routine performs basic magnetic tape positioning.
0000    321 ;       functions are supported for read, rewind, skip files (forward
0000    322 ;       backward) and skip records (forward and backward). It also
0000    323 ;       support drive clear.  Drive clear must be called before the first
0000    324 ;       tape operation as it is necessary to initialize the tape subsystem
0000    325 ;       prior to the first position or transfer operation.
0000    326 ;       Drive clear initialization clears the Need Buffer Address (NBA)
0000    327 ;       bit in the drive. This bit can be reset by a UNIBUS reset/init
0000    328 ;       which in the TS11 causes the tape to be rewound. If NBA is set
0000    329 ;       the function is rejected with tape position lost.
0000    330 ;
0000    331 ; Inputs:
0000    332 ;
0000    333 ;       R3      - base address of adapter's register space
0000    334 ;       R5      - LBN for current piece of transfer
0000    335 ;       R6      - contains 0
0000    336 ;       R7      - Address of devices CSR
0000    337 ;       R8      - size of transfer in bytes
0000    338 ;       R9      - address of the RPB
0000    339 ;       R10     - starting address of transfer (byte offset in first
0000    340 ;                 page ORed with starting map register number)
0000    341 ;
0000    342 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_REWIND only)
0000    343 ;       MODE(AP)- Address interpretation mode (0 = physical, 1 = virtual)
0000    344 ;       SIZE(AP)- Size of buffer in bytes
0000    345 ;       BUF(AP) - Address of buffer
0000    346 ;
0000    347 ; Implicit inputs:
0000    348 ;
0000    349 ;       RPB$B_SLAVE     - Slave number of drive
0000    350 ;       RPB$W_UNIT      - RPB field containing boot device unit number
0000    351 ;
0000    352 ;
0000    353 ; Outputs:
0000    354 ;
0000    355 ;       R0 - status code
0000    356 ;
0000    357 ;               SS$_NORMAL      - successful transfer
0000    358 ;               SS$_TAPEPOSLOST - Tape position lost
0000    359 ;               SS$_TIMEOUT     - Device timeout
0000    360 ;               SS$_DATAOVERUN  - Record too long
0000    361 ;               SS$_ILLIOFUNC   - Illegal I/O function
0000    362 ;               SS$_CTRLERR     - Controller error
0000    363 ;               SS$_ENDOFFILE   - Tape mark was read
0000    364 ;               SS$_DEVOFFLINE  - Device is offline
0000    365 ;
0000    366 ;       R1 - Number of bytes transferred during a READ operation
0000    367 ; Register usage:
0000    368 ;
0000    369 ;               R5      - Map number of map for command area
0000    370 ;               R11     - Address of command buffer
0000    371 ;
```

```
                        0000   372 ;--
                        0000   373
                00000004 0000   374 BUF = 4
                00000008 0000   375 SIZE = 8
                0000000C 0000   376 LBN = 12
                00000010 0000   377 FUNC = 16
                00000014 0000   378 MODE = 20
```

```
                        0000   380 TS11_DRIVER::                           ; TS11/TS04, TU80  driver
                        0000   381
              04 AE   D4 0000   382         CLRL    4(SP)                   ; Clear retry count ****************
           52   02 A7   B0 0003   383         MOVW    TSSR(R7), R2            ; Get current status
           0C 52    0A   E1 0007   384         BBC     #MS_TSSR_V_NBA, R2, 10$ ; Branch if not reset since last op
           04   10 AC   D1 000B   385         CMPL    FUNC(AP),#IO$_DRVCLR    ; Drive clear?
                 06   13 000F   386         BEQL    10$                     ; If so continue
        50   0224 8F   3C 0011   387         MOVZWL  #SS$_TAPEPOSLOST, R0    ; Assume tape position lost
                     05 0016   388         RSB
                        0017   389
              09   00   EF 0017   390 10$:    EXTZV   #VA$V_BYTE, #VA$S_BYTE,-
              55   5A      001A   391                 R10, R5                 ; Byte offset from start
        55   03FF C845   9E 001C   392         MOVAB   ^X3FF(R8)[R5],R5        ; Length of transfer + extra page
     55   55   F7 8F   78 0022   393         ASHL    #-9, R5, R5             ; Number of pages to mapped + 1
                        0027   394
  5B   00000000'8F   D0 0027   395         MOVL    #MS$CMD, R11            ; Address of command area
              15   09   EF 002E   396         EXTZV   #VA$V_VPN, #VA$S_VPN,-
              50   5B      0031   397                 R11, R0                 ; Get PFN to map
        51   0800 C345   DE 0033   398         MOVAL   ^X800(R3)[R5], R1       ; Address of map register
     61   90000000 E0   9E 0039   399         MOVAB   <PTE$M_VALID!PTE$C_KW>(R0), -
                        0040   400                 (R1)                    ; Map first page
        81   81   01   C1 0040   401         ADDL3   #1, (R1)+, (R1)+        ; Map 2nd page
                  61   D4 0044   402         CLRL    (R1)                    ; And clear 3rd
                        0046   403
     10 AB   8000 8F   B0 0046   404         MOVW    #MS_MHD_M_ACK,MS_MHD(R11); Set ACK in message buffer
           50   10 AB   3E 004C   405         MOVAW   MS_MHD(R11), R0         ; Copy message buffer address
                013C   30 0050   406         BSBW    MAP_VA                  ; Get UNIBUS address of this VA
                        0053   407
           08 AB   50   D0 0053   408         MOVL    R0, MS_MBA0(R11)        ; Set address of message buffer
           0C AB   0E   B0 0057   409         MOVW    #14, MS_LNTH(R11)       ; Length of message buffer
              0E AB   B4 005B   410         CLRW    MS_CHWD(R11)            ; Characteristics word
                        005E   411
     6B   C004 8F   B0 005E   412         MOVW    #HC_WRC, MS_CPHD(R11)   ; Write characteristics command
        50   08 AB   9E 0063   413         MOVAB   MS_MBA0(R11), R0        ; Address of characteristic data
                0125   30 0067   414         BSBW    MAP_VA                  ; Get UNIBUS address of this VA
        02 AB   50   D0 006A   415         MOVL    R0, MS_BACT(R11)        ; Set low and high address
        06 AB   08   B0 006E   416         MOVW    #8, MS_CNT(R11)         ; Set length of area
                        0072   417
                010C   30 0072   418         BSBW    EXECUTE                 ; Execute function
                00A5   30 0075   419         BSBW    WAIT                    ; Wait for Subsystem Ready to set
              25 50   E9 0078   420         BLBC    R0,RETURN               ; Exit if subsystem ready does not set
                        007B   421
                        007B   422
                        007B   423 ;+
                        007B   424 ; branch on function type
                        007B   425 ;-
        21   10 AC   D1 007B   426         CMPL    FUNC(AP),#IO$_READLBLK  ; Read logical block?
                  2F   13 007F   427         BEQL    E_READ                  ; Execute read function
        24   10 AC   D1 0081   428         CMPL    FUNC(AP),#IO$_REWIND    ; Rewind?
                  1A   13 0085   429         BEQL    E_REWIND                ; Do a rewind
        25   10 AC   D1 0087   430         CMPL    FUNC(AP),#IO$_SKIPFILE  ; Skip past eof
                  4C   13 008B   431         BEQL    E_SKIPFILE              ; Do it
        26   10 AC   D1 008D   432         CMPL    FUNC(AP),#IO$_SKIPRECORD; Skip records?
                  68   13 0091   433         BEQL    E_SKIPRECORD            ; Do it
           50   01   D0 0093   434         MOVL    #SS$_NORMAL, R0         ; Assume normal completion
        04   10 AC   D1 0096   435         CMPL    FUNC(AP),#IO$_DRVCLR    ; Drive clear?
                  04   13 009A   436         BEQL    RETURN                  ; If so have already done it
```

```
                50   F4 8F    9A  009C   437          MOVZBL  #SS$_ILLIOFUNC,R0        ; illegal I/O function
                                  00A0   438 RETURN:
                             05   00A0   439          RSB                             ; Return to caller
                                  00A1   440 ;+
                                  00A1   441 ; Rewind slave drive, wait for completion
                                  00A1   442 ;-
                                  00A1   443 E_REWIND:
            6B   C408 8F     B0   00A1   444          MOVW    #HC_RWD, MS_CPHD(R11)   ; Set function to rewind
                 50   5B     D0   00A6   445          MOVL    R11, R0                 ; VA to map
                    00D5     30   00A9   446          BSBW    EXECUTE                 ; Execute function
                    006E     30   00AC   447          BSBW    WAIT                    ; Wait for ready to set
                             05   00AF   448          RSB                             ; Exit
                                  00B0   449 ;+
                                  00B0   450 ; Read block of data
                                  00B0   451 ;-
                                  00B0   452 E_READ:
            6B   C001 8F     B0   00B0   453          MOVW    #HC_RDN, MS_CPHD(R11)   ; Set function to read next
              02 AB   5A     D0   00B5   454          MOVL    R10, MS_BACT(R11)       ; Set address to read to
              06 AB   58     B0   00B9   455          MOVW    R8, MS_CNT(R11)         ; Set length to read
                    00C1     30   00BD   456          BSBW    EXECUTE                 ; Execute function
                    005A     30   00C0   457          BSBW    WAIT                    ; Wait for completion
              51    14 AB    3C   00C3   458          MOVZWL  MS_RBPC(R11), R1        ; Bytes residual byte count
            51    58   51    C3   00C7   459          SUBL3   R1, R8, R1              ; From desired length is transfered
                 0A 50     E9   00CB   460          BLBC    R0, 20$                 ; Branch if complete failure
         05 16 AB    0C     E1   00CE   461          BBC     #MS_XSR0_V_RLL, MS_XSR0(R11), -
                                  00D3   462                  20$                     ; Branch if record not longer than buf
            50   0838 8F     3C   00D3   463          MOVZWL  #SS$_DATAOVERUN, R0     ; Longer, change return
                             05   00D8   464 20$:      RSB
                                  00D9   465
                                  00D9   466 ;+
                                  00D9   467 ; Skip to end of file
                                  00D9   468 ; P1 is negative if backwards else forwards
                                  00D9   469 ;-
                                  00D9   470 E_SKIPFILE:
                50    01    D0   00D9   471          MOVL    #SS$_NORMAL,R0          ; Assume it works
             6B   C208 8F     B0   00DC   472          MOVW    #HC_STF, MS_CPHD(R11)   ; Set function to space tape marks
         02 AB    04 AC     F7   00E1   473          CVTLW   BUF(AP), MS_BACT(R11)   ; Set count of marks to skip
                    0C     14   00E6   474          BGTR    10$                     ; Branch if positive
                    10     13   00E8   475          BEQL    20$                     ; Branch if count 0
             6B   C308 8F     B0   00EA   476          MOVW    #HC_STR, MS_CPHD(R11)   ; Set function to space marks reverse
         02 AB    02 AB     AE   00EF   477          MNEGW   MS_BACT(R11),MS_BACT(R11); Make count positive
                                  00F4   478
                    008A     30   00F4   479 10$:     BSBW    EXECUTE                 ; Start function
                                  00F7   480
                    0023     30   00F7   481          BSBW    WAIT                    ; Wait for it to finish
                             05   00FA   482 20$:     RSB
                                  00FB   483
                                  00FB   484 ;+
                                  00FB   485 ; Skip to end of record, read w/o buffer
                                  00FB   486 ; P1 is negative if backwards else forwards
                                  00FB   487 ;-
                                  00FB   488 E_SKIPRECORD:
                50    01    D0   00FB   489          MOVL    #SS$_NORMAL,R0          ; Assume it works
             6B   C008 8F     B0   00FE   490          MOVW    #HC_SRF, MS_CPHD(R11)   ; Assume forward
         02 AB    04 AC     F7   0103   491          CVTLW   BUF(AP), MS_BACT(R11)   ; Set record count
                    0C     14   0108   492          BGTR    30$                     ; All set, execute
                    10     13   010A   493          BEQL    50$                     ; Finish if space = 0
```

```
                 6B   C108 8F    B0  010C   494          MOVW    #HC_SRR, MS_CPHD(R11)   ; Function = space reverse
               02 AB    02 AB    AE  0111   495          MNEGW   MS_BACT(R11),MS_BACT(R11); Set record count
                                    0116   496
                        0068      30  0116   497 30$:     BSBW    EXECUTE                 ; Translate R11 command and execute
                        0001      30  0119   498          BSBW    WAIT                    ; Wait for it to finish
                                    011C   499
                                 05 011C   500 50$:     RSB
                                    011D   501
                                    011D   502
                                    011D   503 ;+
                                    011D   504 ; Wait for slave to quiesce
                                    011D   505 ;-
                                    011D   506 WAIT:
                50   3B9ACA00 8F    D0  011D   507          MOVL    #1000000000, R0         ; time out counter
                 02 A7    0080 8F    B3  0124   508 10$:     BITW    #MS_TSSR_M_SSR, TSSR(R7); Subsystem ready?
                                 09 12  012A   509          BNEQ    20$                     ; Exit if subsystem ready
                                F5 50   F5  012C   510          SOBGTR  R0, 10$                 ; Count and try again
                   50    022C 8F    3C  012F   511          MOVZWL  #SS$_TIMEOUT, R0        ; Set return code
                                 05 0134   512          RSB                             ; return
                                    0135   513
                   50    0054 8F    3C  0135   514 20$:     MOVZWL  #SS$_CTRLERR, R0        ; Assume controller error
                   51    7C00 8F    B3  013A   515          BITW    #MS_TSSR_M_UPE ! MS_TSSR_M_SPE ! -
                                    013F   516                  MS_TSSR_M_RMR ! MS_TSSR_M_NXM ! -
                                    013F   517                  MS_TSSR_M_NBA, R1       ; Any fatal errors?
                                 3F 12  013F   518          BNEQ    90$                     ; Exit if any
                                    0141   519
                     51    02 A7    32  0141   520          CVTWL   TSSR(R7), R1            ; Get TSSR
                                 36 18  0145   521          BGEQ    80$                     ; Branch if SC clear, normal
                                    0147   522
                   51    03    01    EF  0147   523          EXTZV   #MS_TSSR_V_TCC, #MS_TSSR_S_TCC, -
                                 50       014B   524                  R1, R0                  ; Get termination class
                                    014C   525          CASE    R0, LIMIT = #TCC_NML, DISPLIST = <80$, 80$, 50$, 60$>
                03'   00    50    AF  014C               CASEW   R0,#TCC_NML,S^#<<30001$-30000$>/2>-1
                                    0150       30000$:
                                 002D' 0150               .SIGNED_WORD    80$-30000$
                                 002D' 0152               .SIGNED_WORD    80$-30000$
                                 000F' 0154               .SIGNED_WORD    50$-30000$
                                 001E' 0156               .SIGNED_WORD    60$-30000$
                                    0158       30001$:
                   50    0054 8F    3C  0158   526          MOVZWL  #SS$_CTRLERR, R0        ; Assume controller error
                                 21 11  015D   527          BRB     90$                     ; And exit
                                    015F   528
                                    015F   529 ;+
                                    015F   530 ; Tape status condition
                                    015F   531 ;-
             16 AB    8003 8F    B3  015F   532 50$:     BITW    #MS_XSR0_M_TMK ! -
                                    0165   533                  MS_XSR0_M_BOT ! -
                                    0165   534                  MS_XSR0_M_EOT, -
                                    0165   535                  MS_XSR0(R11)            ;Tape mark detected?
                                 16 13  0165   536          BEQL    80$                     ; No, exit normal
                   50    0870 8F    3C  0167   537          MOVZWL  #SS$_ENDOFFILE,R0       ; Assume EOF sensed
                                 12 11  016C   538          BRB     90$                     ; Exit
                                    016E   539
                                    016E   540 ;+
                                    016E   541 ; Function reject
                                    016E   542 ;-
                                    016E   543 60$:
```

```
                50   84 8F    9A  016E   544          MOVZBL  #SS$_DEVOFFLINE,R0       ; Assume offline
             09 16 AB    06    E1  0172   545          BBC     #MS_XSR0_V_ONL, -
                                  0177   546                  MS_XSR0(R11), 90$       ; Branch if offline
                                  0177   547
                50   54 8F    9A  0177   548          MOVZBL  #SS$_CTRLERR,R0          ; Assume error
                         03    11  017B   549          BRB     90$                      ; Exit
                                  017D   550 ;+
                                  017D   551 ; Success
                                  017D   552 ;-
                    50   01    D0  017D   553 80$:     MOVL    #SS$_NORMAL, R0          ; Success
                         05    0180   554 90$:     RSB                              ; Return from WAIT
                               0181   555
                               0181   556
                               0181   557 ;+
                               0181   558 ; Execute command pointer aR11
                               0181   559 ;-
                               0181   560 EXECUTE:
                    50   5B    D0  0181   561          MOVL    R11, R0                  ; VA to map
                         09    10  0184   562          BSBB    MAP_VA                   ; Map to UNIBUS space
             51     50   10    9C  0186   563          ROTL    #16, R0, R1              ; Position bits 16 & 17
             67     51   50    A9  018A   564          BISW3   R0, R1, TSBA(R7)         ; Execute the function
                         05    018E   565          RSB
                               018F   566
                               018F   567 ;+
                               018F   568 ; Map and translate VA to UNIBUS address
                               018F   569 ; R0 is VA to be mapped to UNIBUS space
                               018F   570 ;       R0      VA to be mapped
                               018F   571 ;       R3      Base of adapter space
                               018F   572 ;       R5      Map register number assigned for command/message area
                               018F   573 ;               may require two maps
                               018F   574 ;-
                               018F   575 MAP_VA:
          51   00000000'8F    D0  018F   576          MOVL    #MS$CMD, R1              ; Get command address
           51    51    F7 8F    78  0196   577          ASHL    #-9, R1,R1               ; PFN only
                               019B   578
                50   15   09    ED  019B   579          CMPZV   #VA$V_VPN, #VA$S_VPN, -
                               51      019F   580                  R0, R1           ; Does VA map same page?
                               07    12  01A0   581          BNEQ    10$              ; Branch if should be other page
          50   15   09   55    F0  01A2   582          INSV    R5, #VA$V_VPN, #VA$S_VPN, -
                               01A7   583                  R0                       ; Insert UNIBUS PFN
                               15    11  01A7   584          BRB     20$              ; Exit
                               01A9   585
                               51    D6  01A9   586 10$:     INCL    R1               ; Next map maps next PFN
                54   15   09    ED  01AB   587          CMPZV   #VA$V_VPN, #VA$S_VPN, -
                               51      01AF   588                  R4, R1           ; Does it map this page?
                               0D    12  01B0   589          BNEQ    30$              ; Abort if page not mapped by either
          50   15   09   55    F0  01B2   590          INSV    R5, #VA$V_VPN, #VA$S_VPN, -
                               01B7   591                  R0                       ; Insert UNIBUS PFN
           50   00000200 8F    C0  01B7   592          ADDL    #^X200, R0               ; Map to next page
                               01BE   593
                               05  01BE   594 20$:     RSB                              ; Return
                               01BF   595
                    50   2C    3C  01BF   596 30$:     MOVZWL  #SS$_ABORT, R0           ; Set return code
                         8E    D5  01C2   597          TSTL    (SP)+                    ; Return extra return address
                               05  01C4   598          RSB                              ; And return from bootdriver
                               01C5   599
                               01C5   600
```

```
                                  01C5   601 DRIVERNAME:                          ; Boot device driver name
58 45 2E 52 45 56 49 52 44 53 54 00' 01C5   602         .ASCIC  /TSDRIVER.EXE/        ; TSDRIVER
                              45   01D1
                              0C   01C5
                                  01D2   603
                        000001D2  01D2   604 TS11_DRVSIZ = .-TS11_DRIVER
                                  01D2   605
                                  01D2   606         .END
```

```
$TABLE                          = 00000000 R   D  03        MS_CPHD_M_ACK                   = 00008000     D
BIT...                          = 00000010     D          MS_CPHD_M_CVC                   = 00004000     D
BTD$K_TM                        = 000000B1     D          MS_LNH                            00000012     D
BTD$K_TS                        = 000000B2     D          MS_LNTH                           0000000C     D
BUF                             = 00000004     D          MS_MBA0                           00000008     D
CLS_MDE                         = 00000003     D          MS_MBA1                           0000000A     D
CLS_MDF                         = 00000001     D          MS_MHD                            00000010     D
CLS_ONF                         = 00000000     D          MS_MHD_M_ACK                    = 00008000     D
CLS_OTHER                       = 00000001     D          MS_RBPC                           00000014     D
CLS_PTB                         = 00000000     D          MS_TSSR_M_A16                   = 00000100     D
CLS_WLN                         = 00000002     D          MS_TSSR_M_A17                   = 00000200     D
DRIVERNAME                        000001C5 R   D  04        MS_TSSR_M_NBA                   = 00000400     D
EXECUTE                           00000181 R   D  04        MS_TSSR_M_NXM                   = 00000800     D
E_READ                            000000B0 R   D  04        MS_TSSR_M_OFL                   = 00000040     D
E_REWIND                          000000A1 R   D  04        MS_TSSR_M_RMR                   = 00001000     D
E_SKIPFILE                        000000D9 R   D  04        MS_TSSR_M_SC                    = 00008000     D
E_SKIPRECORD                      000000FB R   D  04        MS_TSSR_M_SPE                   = 00002000     D
FCC_CPE                         = 00000001     D          MS_TSSR_M_SSR                   = 00000080     D
FCC_IDF                         = 00000000     D          MS_TSSR_M_UPE                   = 00004000     D
FCC_LAP                         = 00000003     D          MS_TSSR_S_FC                    = 00000002     D
FCC_UPE                         = 00000002     D          MS_TSSR_S_TCC                   = 00000003     D
FUNC                            = 00000010     D          MS_TSSR_V_A16                   = 00000008     D
HC_DRI                          = 0000C00B     D          MS_TSSR_V_A17                   = 00000009     D
HC_GST                          = 0000C00F     D          MS_TSSR_V_FC                    = 00000004     D
HC_NOP                          = 00000000     D          MS_TSSR_V_NBA                   = 0000000A     D
HC_PAK                          = 00000000     D          MS_TSSR_V_NXM                   = 0000000B     D
HC_RDN                          = 0000C001     D          MS_TSSR_V_OFL                   = 00000006     D
HC_RPS                          = 00000000     D          MS_TSSR_V_RMR                   = 0000000C     D
HC_RWD                          = 0000C408     D          MS_TSSR_V_SC                    = 0000000F     D
HC_SCH                          = 00000000     D          MS_TSSR_V_SPE                   = 0000000D     D
HC_SRF                          = 0000C008     D          MS_TSSR_V_SSR                   = 00000007     D
HC_SRR                          = 0000C108     D          MS_TSSR_V_TCC                   = 00000001     D
HC_STF                          = 0000C208     D          MS_TSSR_V_UPE                   = 0000000E     D
HC_STR                          = 0000C308     D          MS_XSR0                           00000016     D
HC_UNL                          = 0000C10A     D          MS_XSR0_M_BOT                   = 00000002     D
HC_WCK                          = 00000000     D          MS_XSR0_M_EOT                   = 00000001     D
HC_WKR                          = 00000000     D          MS_XSR0_M_TMK                   = 00008000     D
HC_WRC                          = 0000C004     D          MS_XSR0_V_ONL                   = 00000006     D
IO$_DRVCLR                      = 00000004     D          MS_XSR0_V_RLL                   = 0000000C     D
IO$_READLBLK                    = 00000021     D          MS_XSR1                           00000018     D
IO$_REWIND                      = 00000024     D          MS_XSR2                           0000001A     D
IO$_SKIPFILE                    = 00000025     D          MS_XSR3                           0000001C     D
IO$_SKIPRECORD                  = 00000026     D          PTE$C_KW                        = 10000000     D
LBN                             = 0000000C     D          PTE$M_VALID                     = 80000000     D
MAP_VA                            0000018F R   D  04        RETURN                            000000A0 R   D  04
MODE                            = 00000014     D          SIZ...                          = 00000001     D
MS$CMD                            00000000 RG  D  02        SIZE                            = 00000008     D
MSG_ATN                         = 00000013     D          SS$_ABORT                       = 0000002C     D
MSG_END                         = 00000010     D          SS$_CTRLERR                     = 00000054     D
MSG_ERR                         = 00000012     D          SS$_DATAOVERUN                  = 00000838     D
MSG_FAL                         = 00000011     D          SS$_DEVOFFLINE                  = 00000084     D
MSG_LOG                         = 00000014     D          SS$_ENDOFFILE                   = 00000870     D
MS_BA1                            00000004     D          SS$_ILLIOFUNC                   = 000000F4     D
MS_BACT                           00000002     D          SS$_NORMAL                      = 00000001     D
MS_CHWD                           0000000E     D          SS$_TAPEPOSLOST                 = 00000224     D
MS_CNT                            00000006     D          SS$_TIMEOUT                     = 0000022C     D
MS_CPHD                           00000000     D          TCC_ATN                         = 00000001     D
```

```
TCC_FNR                         = 00000003    D
TCC_FTL                         = 00000007    D
TCC_NML                         = 00000000    D
TCC_REM                         = 00000004    D
TCC_REN                         = 00000005    D
TCC_TSA                         = 00000002    D
TCC_UER                         = 00000006    D
TS11_DRIVER                       00000000 RG D  04
TS11_DRVSIZ                     = 000001D2    D
TSBA                            = 00000000    D
TSDB                            = 00000000    D
TSSR                            = 00000002    D
VA$S_BYTE                       = 00000009    D
VA$S_VPN                        = 00000015    D
VA$V_BYTE                       = 00000000    D
VA$V_VPN                        = 00000009    D
WAIT                              0000011D R  D  04
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000001E (30.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000020 (32.)	02 (2.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC 21
BOOTDRIVR_4	00000018 (24.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	000001D2 (466.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$TABLE	=00000000-R	313 (2)	313 (2)
BIT...	=00000010	296 (2)	296 (2)
BTD$K_TM	=000000B1	71 (2)	
BTD$K_TS	=000000B2	72 (2)	313 (2)
BUF	=00000004	374 (3)	#-473 (4) #-491 (4)
CLS_MDE	=00000003	245 (2)	
CLS_MDF	=00000001	240 (2)	
CLS_ONF	=00000000	239 (2)	
CLS_OTHER	=00000001	243 (2)	
CLS_PTB	=00000000	242 (2)	
CLS_WLN	=00000002	244 (2)	
DRIVERNAME	000001C5-R	601 (4)	313 (2)
EXECUTE	00000181-R	560 (4)	#-418 (4) #-446 (4) #-456 (4) #-479 (4) #-497 (4)
E_READ	000000B0-R	452 (4)	#-427 (4)
E_REWIND	000000A1-R	443 (4)	#-429 (4)
E_SKIPFILE	000000D9-R	470 (4)	#-431 (4)
E_SKIPRECORD	000000FB-R	488 (4)	#-433 (4)
FCC_CPE	=00000001	220 (2)	
FCC_IDF	=00000000	219 (2)	
FCC_LAP	=00000003	222 (2)	
FCC_UPE	=00000002	221 (2)	
FUNC	=00000010	377 (3)	#-385 (4) #-426 (4) #-428 (4) #-430 (4) #-432 (4) #-435 (4)
HC_DRI	=0000C00B	268 (2)	
HC_GST	=0000C00F	269 (2)	
HC_NOP	=00000000	252 (2)	253 (2) 254 (2) 255 (2) 256 (2) 257 (2)
HC_PAK	=00000000	253 (2)	
HC_RDN	=0000C001	260 (2)	#-453 (4)
HC_RPS	=00000000	256 (2)	
HC_RWD	=0000C408	266 (2)	#-444 (4)
HC_SCH	=00000000	257 (2)	
HC_SRF	=0000C008	262 (2)	#-490 (4)
HC_SRR	=0000C108	263 (2)	#-494 (4)
HC_STF	=0000C208	264 (2)	#-472 (4)
HC_STR	=0000C308	265 (2)	#-476 (4)
HC_UNL	=0000C10A	267 (2)	
HC_WCK	=00000000	254 (2)	
HC_WKR	=00000000	255 (2)	
HC_WRC	=0000C004	261 (2)	#-412 (4)
IO$_DRVCLR	=00000004		#-385 (4) #-435 (4)
IO$_READLBLK	=00000021		#-426 (4)
IO$_REWIND	=00000024		#-428 (4)
IO$_SKIPFILE	=00000025		#-430 (4)
IO$_SKIPRECORD	=00000026		#-432 (4)
LBN	=0000000C	376 (3)	
MAP_VA	0000018F-R	575 (4)	#-406 (4) #-414 (4) #-562 (4)
MODE	=00000014	378 (3)	
MS$CMD	00000000-R	302 (2)	#-395 (4) #-576 (4)

```
MSG_ATN                         =00000013     231     (2)
MSG_END                         =00000010     228     (2)
MSG_ERR                         =00000012     230     (2)
MSG_FAL                         =00000011     229     (2)
MSG_LOG                         =00000014     232     (2)
MS_BACT                          00000002                       #-415     (4)     #-454     (4)     #-473     (4)     #-477     (4)
                                                                #-491     (4)     #-495     (4)
MS_CHWD                          0000000E                       #-410     (4)
MS_CNT                           00000006                       #-416     (4)     #-455     (4)
MS_CPHD                          00000000                       #-412     (4)     #-444     (4)     #-453     (4)     #-472     (4)
                                                                #-476     (4)     #-490     (4)     #-494     (4)
MS_CPHD_M_ACK                   =00008000                         260     (2)       261     (2)       262     (2)       263     (2)
                                                                  264     (2)       265     (2)       266     (2)       267     (2)
                                                                  268     (2)       269     (2)
MS_CPHD_M_CVC                   =00004000                         260     (2)       261     (2)       262     (2)       263     (2)
                                                                  264     (2)       265     (2)       266     (2)       267     (2)
                                                                  268     (2)       269     (2)
MS_LNTH                          0000000C                       #-409     (4)
MS_MBA0                          00000008                       #-408     (4)       413     (4)
MS_MHD                           00000010                       #-404     (4)       405     (4)
MS_MHD_M_ACK                    =00008000                       #-404     (4)
MS_RBPC                          00000014                       #-458     (4)
MS_TSSR_M_A16                   =00000100     296     (2)
MS_TSSR_M_A17                   =00000200     296     (2)
MS_TSSR_M_NBA                   =00000400     296     (2)       #-517     (4)
MS_TSSR_M_NXM                   =00000800     296     (2)       #-516     (4)
MS_TSSR_M_OFL                   =00000040     296     (2)
MS_TSSR_M_RMR                   =00001000     296     (2)       #-516     (4)
MS_TSSR_M_SC                    =00008000     296     (2)
MS_TSSR_M_SPE                   =00002000     296     (2)       #-515     (4)
MS_TSSR_M_SSR                   =00000080     296     (2)       #-508     (4)
MS_TSSR_M_UPE                   =00004000     296     (2)       #-515     (4)
MS_TSSR_S_FC                    =00000002     296     (2)
MS_TSSR_S_TCC                   =00000003     296     (2)       #-523     (4)
MS_TSSR_V_A16                   =00000008     296     (2)
MS_TSSR_V_A17                   =00000009     296     (2)
MS_TSSR_V_FC                    =00000004     296     (2)
MS_TSSR_V_NBA                   =0000000A     296     (2)       #-384     (4)
MS_TSSR_V_NXM                   =0000000B     296     (2)
MS_TSSR_V_OFL                   =00000006     296     (2)
MS_TSSR_V_RMR                   =0000000C     296     (2)
MS_TSSR_V_SC                    =0000000F     296     (2)
MS_TSSR_V_SPE                   =0000000D     296     (2)
MS_TSSR_V_SSR                   =00000007     296     (2)
MS_TSSR_V_TCC                   =00000001     296     (2)       #-523     (4)
MS_TSSR_V_UPE                   =0000000E     296     (2)
MS_XSR0                          00000016                         461     (4)     #-535     (4)       546     (4)
MS_XSR0_M_BOT                   =00000002                       #-533     (4)
MS_XSR0_M_EOT                   =00000001                       #-534     (4)
MS_XSR0_M_TMK                   =00008000                       #-532     (4)
MS_XSR0_V_ONL                   =00000006                       #-545     (4)
MS_XSR0_V_RLL                   =0000000C                       #-461     (4)
PTE$C_KW                        =10000000                         399     (4)
PTE$M_VALID                     =80000000                         399     (4)
RETURN                           000000A0-R    438     (4)       #-420     (4)     #-436     (4)
SIZ...                          =00000001     296     (2)         296     (2)
SIZE                            =00000008     375     (3)
```

```
SS$_ABORT                =0000002C                  #-596    (4)
SS$_CTRLERR              =00000054                  #-514    (4)    #-526    (4)    #-548    (4)
SS$_DATAOVERUN           =00000838                  #-463    (4)
SS$_DEVOFFLINE           =00000084                  #-544    (4)
SS$_ENDOFFILE            =00000870                  #-537    (4)
SS$_ILLIOFUNC            =000000F4                  #-437    (4)
SS$_NORMAL               =00000001                  #-434    (4)    #-471    (4)    #-489    (4)    #-553    (4)
SS$_TAPEPOSLOST          =00000224                  #-387    (4)
SS$_TIMEOUT              =0000022C                  #-511    (4)
TCC_ATN                  =00000001     207    (2)
TCC_FNR                  =00000003     209    (2)
TCC_FTL                  =00000007     213    (2)
TCC_NML                  =00000000     206    (2)    #-525    (4)
TCC_REM                  =00000004     210    (2)
TCC_REN                  =00000005     211    (2)
TCC_TSA                  =00000002     208    (2)
TCC_UER                  =00000006     212    (2)
TS11_DRIVER               00000000-R   380    (4)      313    (2)      604    (4)
TS11_DRVSIZ              =000001D2     604    (4)      313    (2)
TSBA                     =00000000     280    (2)    #-564    (4)
TSDB                     =00000000     279    (2)
TSSR                     =00000002     281    (2)    #-383    (4)    #-508    (4)    #-520    (4)
VA$S_BYTE                =00000009                  #-390    (4)
VA$S_VPN                 =00000015                  #-396    (4)    #-579    (4)    #-582    (4)    #-587    (4)
                                                    #-590    (4)
VA$V_BYTE                =00000000                  #-390    (4)
VA$V_VPN                 =00000009                  #-396    (4)    #-579    (4)    #-582    (4)    #-587    (4)
                                                    #-590    (4)
WAIT                      0000011D-R   506    (4)    #-419    (4)    #-447    (4)    #-457    (4)    #-481    (4)
                                                    #-498    (4)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$BOOT_DRIVER	1	309 (2)	309 (2)
$BTDDEF	2	54 (2)	54 (2)
$DEFINI	1	54 (2)	54 (2) 55 (2) 56 (2) 57 (2) 58 (2) 59 (2) 60 (2) 61 (2) 83 (2)
$IODEF	15	55 (2)	55 (2)
$NDTDEF	3	56 (2)	56 (2)
$PRDEF	5	58 (2)	58 (2)
$PTEDEF	3	57 (2)	57 (2)
$RPBDEF	5	59 (2)	59 (2)
$SSDEF	21	60 (2)	60 (2)
$VADEF	1	61 (2)	61 (2)
$VIELD1	1		296 (2)
CASE	1	525 (4)	525 (4)
_VIELD	1		283 (2)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...
ADDL	00C0	592 (4)
ADDL3	00C1	401 (4)
ASHL	0078	393 (4) 577 (4)
BBC	00E1	384 (4) 461 (4) 545 (4)
BEQL	0013	386 (4) 427 (4) 429 (4) 431 (4) 433 (4) 436 (4) 475 (4)
		493 (4) 536 (4)
BGEQ	0018	521 (4)
BGTR	0014	474 (4) 492 (4)
BISW3	00A9	564 (4)
BITW	00B3	508 (4) 515 (4) 532 (4)
BLBC	00E9	420 (4) 460 (4)
BNEQ	0012	509 (4) 518 (4) 581 (4) 589 (4)
BRB	0011	527 (4) 538 (4) 549 (4) 584 (4)
BSBB	0010	562 (4)
BSBW	0030	406 (4) 414 (4) 418 (4) 419 (4) 446 (4) 447 (4) 456 (4)
		457 (4) 479 (4) 481 (4) 497 (4) 498 (4)
CASEW	00AF	525 (4)
CLRL	00D4	382 (4) 402 (4)
CLRW	00B4	410 (4)
CMPL	00D1	385 (4) 426 (4) 428 (4) 430 (4) 432 (4) 435 (4)
CMPZV	00ED	579 (4) 587 (4)
CVTLW	00F7	473 (4) 491 (4)
CVTWL	0032	520 (4)
EXTZV	00EF	390 (4) 396 (4) 523 (4)
INCL	00D6	586 (4)
INSV	00F0	582 (4) 590 (4)
MNEGW	00AE	477 (4) 495 (4)
MOVAB	009E	392 (4) 399 (4) 413 (4)
MOVAL	00DE	398 (4)
MOVAW	003E	405 (4)
MOVL	00D0	395 (4) 408 (4) 415 (4) 434 (4) 445 (4) 454 (4) 471 (4)
		489 (4) 507 (4) 553 (4) 561 (4) 576 (4)
MOVW	00B0	383 (4) 404 (4) 409 (4) 412 (4) 416 (4) 444 (4) 453 (4)
		455 (4) 472 (4) 476 (4) 490 (4) 494 (4)
MOVZBL	009A	437 (4) 544 (4) 548 (4)
MOVZWL	003C	387 (4) 458 (4) 463 (4) 511 (4) 514 (4) 526 (4) 537 (4)
		596 (4)
ROTL	009C	563 (4)
RSB	0005	388 (4) 439 (4) 448 (4) 464 (4) 482 (4) 500 (4) 512 (4)
		554 (4) 565 (4) 594 (4) 598 (4)
SOBGTR	00F5	510 (4)
SUBL3	00C3	459 (4)
TSTL	00D5	597 (4)

Directives Cross Reference

DIRECTIVE	REFERENCES...															
.ASCIC	602	(4)														
.BLKW	302	(2)														
.DISABLE	6	(1)														
.END	606	(4)														
.ENDC	296	(2)	313	(2)												
.ENDM	309	(2)	525	(4)	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)
	60	(2)	61	(2)												
.ENDR	296	(2)	525	(4)												
.IDENT	5	(1)														
.IF	296	(2)	313	(2)												
.IFF	313	(2)														
.IIF	296	(2)														
.IRP	296	(2)	525	(4)												
.LONG	313	(2)														
.NOCROSS	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	60	(2)	61	(2)
	83	(2)														
.PSECT	301	(2)	313	(2)												
.RESTORE	200	(2)	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	60	(2)
	61	(2)														
.SAVE	54	(2)	55	(2)	56	(2)	57	(2)	58	(2)	59	(2)	60	(2)	61	(2)
	83	(2)														
.SBTTL	315	(3)	49	(2)												
.SIGNED_WORD	525	(4)														
.TITLE	4	(1)														
.WORD	313	(2)														

Register Cross Reference

REGISTER	NO. REFERENCES	REFERENCES...											
AP	8	#-385	(4)	#-426	(4)	#-428	(4)	#-430	(4)	#-432	(4)	#-435	(4)
		#-473	(4)	#-491	(4)								
R0	34	#-387	(4)	#-397	(4)	399	(4)	#-405	(4)	#-408	(4)	#-413	(4)
		#-415	(4)	#-420	(4)	#-434	(4)	#-437	(4)	#-445	(4)	#-460	(4)
		#-463	(4)	#-471	(4)	#-489	(4)	#-507	(4)	#-510	(4)	#-511	(4)
		#-514	(4)	#-524	(4)	#-525	(4)	#-526	(4)	#-537	(4)	#-544	(4)
		#-548	(4)	#-553	(4)	#-561	(4)	#-563	(4)	#-564	(4)	580	(4)
		583	(4)	591	(4)	#-592	(4)	#-596	(4)				
R1	19	#-398	(4)	#-400	(4)	#-401	(4)	#-402	(4)	#-458	(4)	#-459	(4)
		#-517	(4)	#-520	(4)	524	(4)	#-563	(4)	#-564	(4)	#-576	(4)
		#-577	(4)	#-580	(4)	#-586	(4)	#-588	(4)				
R10	2	391	(4)	#-454	(4)								
R11	31	#-395	(4)	397	(4)	#-404	(4)	405	(4)	#-408	(4)	#-409	(4)
		#-410	(4)	#-412	(4)	413	(4)	#-415	(4)	#-416	(4)	#-444	(4)
		#-445	(4)	#-453	(4)	#-454	(4)	#-455	(4)	#-458	(4)	461	(4)
		#-472	(4)	#-473	(4)	#-476	(4)	#-477	(4)	#-490	(4)	#-491	(4)
		#-494	(4)	#-495	(4)	#-535	(4)	546	(4)	#-561	(4)		
R2	2	#-383	(4)	384	(4)								
R3	1	398	(4)										
R4	1	588	(4)										
R5	8	#-391	(4)	392	(4)	#-393	(4)	398	(4)	#-582	(4)	#-590	(4)
R7	4	#-383	(4)	#-508	(4)	#-520	(4)	#-564	(4)				
R8	3	392	(4)	#-455	(4)	#-459	(4)						
SP	2	#-382	(4)	#-597	(4)								

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.01	00:00:00.18
Command processing	882	00:00:00.23	00:00:00.81
Pass 1	624	00:00:02.98	00:00:04.37
Symbol table sort	0	00:00:00.37	00:00:00.36
Pass 2	6	00:00:00.73	00:00:01.45
Symbol table output	0	00:00:00.02	00:00:00.06
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	2	00:00:00.16	00:00:00.38
Assembler run totals	1537	00:00:04.51	00:00:07.62

The working set limit was 2900 pages.
149901 bytes (293 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1303 non-local and 17 local symbols.
606 source lines were read in Pass 1, producing 85 object records in Pass 2.
51 pages of virtual memory were used to define 18 macros.

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              1
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                1
SYS$COMMON:[SYSLIB]LIB.MLB;2                           5
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       7
TOTALS (all libraries)                                14

1201 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:TSBTDRIVR.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R
```

```
;    0001  0      ! DEC/CMS REPLACEMENT HISTORY, Element ULTRIX.B32
;    0002  0      ! *4    28-JUL-1986 10:35:51 BAGGETT "ULTRIX LOAD COMMAND FINISH."
;    0003  0      ! *3    25-JUL-1986 15:06:01 BAGGETT "LOAD COMMAND WORKS ON ULTRIX DISKS"
;    0004  0      ! *2    21-MAR-1986 15:26:29 BAGGETT "STATIC MEMORY REMOVE FOR MODULE"
;    0005  0      ! *1     6-FEB-1986 15:22:50 DS ""
;    0006  0      ! DEC/CMS REPLACEMENT HISTORY, Element ULTRIX.B32
;    0007  0       %title '*** ULTRIX Disk RMS routines'
;    0008  0       module ULTRIX (                                    ! RMS$OPEN, RMS$GET, RMS$READ services
;    0009  0                      ident = '01-05',                                                    !G:3
;    0010  0                      ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE,
;    0011  0                      NONEXTERNAL = LONG_RELATIVE)
;    0012  0                      ) =
;    0013  0       !
;    0014  0       ! Copyright (c) 1985, 1986                                                         !G:2
;    0015  0       ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
;    0016  0       !
;    0017  0       ! THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
;    0018  0       ! COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
;    0019  0       ! ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
;    0020  0       ! MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
;    0021  0       ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
;    0022  0       ! TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
;    0023  0       ! REMAIN IN DEC.
;    0024  0       !
;    0025  0       ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
;    0026  0       ! AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
;    0027  0       ! CORPORATION.
;    0028  0       !
;    0029  0       ! DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
;    0030  0       ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
;    0031  0
;    0032  0       !++
;    0033  0       ! FACILITY: Diagnostic Supervisor
;    0034  0       !
;    0035  0       ! ABSTRACT:
;    0036  0       !
;    0037  0       !     This module emulates RMS functions on ULTRIX media.
;    0038  0       !
;    0039  0       ! AUTHOR:
;    0040  0       !     M. BAGGETT              22-May-1985
;    0041  0       !
;    0042  0       ! MODIFIED BY:
;    0043  0       !     05      M. Baggett      24-Jul-1986     VDS 10                              !G:3
;    0044  0       !             Fix LOAD command and add documentation.                             !G:3
;    0045  0       !
;    0046  0       !     04      M. Baggett      10-March-1986   VDS 10                              !G:2
;    0047  0       !             Remove Static buffers and allow partition                           !G:2
;    0048  0       !             specifier to be number or letter.                                   !G:2
;    0049  0       !                                                                                 !G:2
;    0050  0       !     03      M. Baggett      4-Dec-1985      VDS 9.0
;    0051  0       !             Modified READ to allow starting at a given block
;    0052  0       !             within the file and only reading a specified
;    0053  0       !             number of bytes.
;    0054  0       !
;    0055  0       !     02      M. Baggett      8-Nov-1985      VDS 9.0
;    0056  0       !             Allow large files to load.
;    0057  0       !
```

```
;   0058  0       !      01      M. Baggett      22-Aug-1985     VDS 8.3
;   0059  0       !              Fix directory command to always give
;   0060  0       !              directory not found for bad directories.
;   0061  0       !--
;   0062  0
```

```
;    0063  0
;    0064  0       %sbttl 'declarations'
;    0065  1       begin
;    0066  1       !
;    0067  1       ! include files:
;    0068  1       !
;    0069  1
;    0070  1       library '$diag';                !                                                       [03]
;    0071  1
;    0072  1       library '$ds';                  !                                                       [03]
;    0073  1
;    0074  1       library 'sys$library:lib.l32';
;    0075  1       builtin
;    0076  1             rot;
;    0077  1       !
;    0078  1       ! local declarations
;    0079  1       !
;    0080  1
;    0081  1       linkage
;    0082  1           jsb_ultrix = jsb (register = 0) : global (block = 6, vbn = 7, offset = 8,  cblock = 9, rab = 10, fab =11);
;    0083  1
;    0084  1       !
;    0085  1       ! table of contents:
;    0086  1       !
;    0087  1
;    0088  1       forward routine
;    0089  1           ultrix$advance : jsb_ultrix novalue,       ! Advance RFA
;    0090  1           ultrix$access : jsb_ultrix,               ! Access a VBN
;    0091  1           ultrix$open : call_rms,                   ! RMS$OPEN emulator--open file
;    0092  1           ultrix$get : call_rms,                    ! RMS$GET emulator--deblock record
;    0093  1           ultrix$openi: jsb_ultrix,                 ! Read in specified data
;    0094  1           ultrix$sbmap: jsb_ultrix,                 ! Map to blk # to Superblock
;    0095  1           ultrix$find:  jsb_ultrix,                 ! Find file
;    0096  1           ultrix$dlook: jsb_ultrix,                 ! Look for directory
;    0097  1           ultrix$readdir: Jsb_ultrix,               ! Get directory inode #
;    0098  1           ultrix$getc:  jsb_ultrix,                 ! Get characters from file
;    0099  1           ultrix$ptinfo: jsb_ultrix,                ! Get partition information
;    0100  1           ultrix$devread: jsb_ultrix,               ! Read a block from device
;    0101  1           ultrix$read: call_rms;                    ! Read the file.
;    0102  1       !
;    0103  1       ! Local definitions
;    0104  1       !
;    0105  1       !
;    0106  1       ! External references
;    0107  1       !
;    0108  1
;    0109  1       external
;    0110  1          end_size: byte addressing_mode (long_relative); ! Directory string len
;    0111  1
;    0112  1       external routine
;    0113  1           rms$alloc_cache : jsb_cblock,      ! Allocate cache blocks
;    0114  1           boo$qio : addressing_mode (long_relative), ! standalone 'level-3' QIO
;    0115  1           exe$alononpaged : jsb_alloc addressing_mode (long_relative);       ! Allocate static buffer space
;    0116  1
;    0117  1       !
;    0118  1       ! psect definitions:
;    0119  1       !
```

```
;    0120  1
;    0121  1       psect
;    0122  1           plit = data ( share, addressing_mode (long_relative));
;    0123  1
;    0124  1       psect
;    0125  1           own = work ( read, write);
;    0126  1
;    0127  1       psect
;    0128  1           global = work ( read, write);
;    0129  1
;    0130  1       psect
;    0131  1           code = code (share);
;    0132  1      OWN
;    0133  1      cc: LONG,
;    0134  1      transfer_size,
;    0135  1      N,
;    0136  1      i,
;    0137  1      j,
;    0138  1      sh,
;    0139  1      rlen,
;    0140  1      len,
;    0141  1      d,
;    0142  1      lbn,
;    0143  1      off,
;    0144  1      state,
;    0145  1      frag_size,                                  ! Size of fragment in bytes.
;    0146  1      str_size,
;    0147  1      str_path,
;    0148  1      diff,
;    0149  1      tmp_boff,                                   ! 001
;    0150  1      boff,
;    0151  1      qio_status,                                 ! RETURN STATUS FROM READ FUNCTION
;    0152  1      start_lbn,                                  ! STARTING LBN
;    0153  1      byte_coun,                                  ! SIZE OF TRANSFER
;    0154  1      adrs_mode,                                  ! ADDRESSING MODE
;    0155  1      io_functn,                                  ! FUNCTION REQUESTED
;    0156  1      buff_addr,                                  ! BUFFER TO PUB DATA
;    0157  1      q,
;    0158  1      bn,
;    0159  1      nb,
;    0160  1      blknos,                                     ! Fileblock number for indirect address
;    0161  1      dp;
;    0162  1
;    0163  1
;    0164  1
;    0165  1      LITERAL
;    0166  1
;    0167  1      !
;    0168  1       ! Cylinder group related limits.
;    0169  1       !
;    0170  1       ! For each cylinder we keep track of the availability of blocks at different
;    0171  1       ! rotational positions, so that we can lay out the data to be picked
;    0172  1       ! up with minimum rotational latency.  NRPOS is the number of rotational
;    0173  1       ! positions which we distinguish.  With NRPOS 8 the resolution of our
;    0174  1       ! summary information is 2ms for a typical 3600 rpm drive.
;    0175  1       !
;    0176  1      NRPOS    =       8,      ! number distinct rotational positions
```

```
;    0177  1
;    0178  1      !
;    0179  1      ! File system parameters and macros.
;    0180  1      !
;    0181  1      ! The file system is made out of blocks of at most MAXBSIZE units,
;    0182  1      ! with smaller units (fragments) only in the last direct block.
;    0183  1      ! MAXBSIZE primarily determines the size of buffers in the buffer
;    0184  1      ! pool. It may be made larger without any effect on existing
;    0185  1      ! file systems; however making it smaller make make some file
;    0186  1      ! systems unmountable.
;    0187  1      !
;    0188  1      ! Note that the blocked devices are assumed to have DEV_BSIZE
;    0189  1      ! "sectors" and that fragments must be some multiple of this size.
;    0190  1      ! Block devices are read in BLKDEV_IOSIZE units. This number must
;    0191  1      ! be a power of two and in the range of
;    0192  1      !      DEV_BSIZE <= BLKDEV_IOSIZE <= MAXBSIZE
;    0193  1
;    0194  1      MAXBSIZE = 8192,
;    0195  1
;    0196  1
;    0197  1      !
;    0198  1      ! The I node is the focus of all file activity in UNIX.
;    0199  1      ! There is a unique inode allocated for each active file,
;    0200  1      ! each current directory, each mounted-on file, text file, and the root.
;    0201  1      ! An inode is 'named' by its dev/inumber pair. (iget/iget.c)
;    0202  1      ! Data in icommon is read in from permanent inode on volume.
;    0203  1      !
;    0204  1
;    0205  1      NDADDR  = 12,           ! direct addresses in inode
;    0206  1      NIADDR  =  3,           ! indirect addresses in inode
;    0207  1
;    0208  1      !
;    0209  1      ! MINBSIZE is the smallest allowable block size.
;    0210  1      ! In order to insure that it is possible to create files of size
;    0211  1      ! 2^32 with only two levels of indirection, MINBSIZE is set to 4096.
;    0212  1      ! MINBSIZE must be big enough to hold a cylinder group block,
;    0213  1      ! thus changes to (struct cg) must keep its size within MINBSIZE.
;    0214  1      ! MAXCPG is limited only to dimension an array in (struct cg);
;    0215  1      ! it can be made larger as long as that structures size remains
;    0216  1      ! within the bounds dictated by MINBSIZE.
;    0217  1      ! Note that super blocks are always of size MAXBSIZE,
;    0218  1      ! and that MAXBSIZE must be >= MINBSIZE.
;    0219  1
;    0220  1      MAXCPG  =       32,     ! maximum fs_cpg
;    0221  1
;    0222  1      !
;    0223  1      ! The path name on which the file system is mounted is maintained
;    0224  1      ! in fs_fsmnt. MAXMNTLEN defines the amount of space allocated in
;    0225  1      ! the super block for this name.
;    0226  1      ! The limit on the amount of summary information per file system
;    0227  1      ! is defined by MAXCSBUFS. It is currently parameterized for a
;    0228  1      ! maximum of two million cylinders.
;    0229  1      !
;    0230  1      MAXMNTLEN  =   512,
;    0231  1      MAXCSBUFS  =   32,
;    0232  1      SBSIZE     =   8192,
;    0233  1      dinode_size = 128,              ! Size of dinode structure
```

```
;    0234  1      pt_size = 72,                   ! Size of partition table
;    0235  1
;    0236  1      !
;    0237  1      !
;    0238  1      ! BLOCK STRUCTURE OFFSET VALUES
;    0239  1      !
;    0240  1      chain_base  =  4,
;    0241  1      lastr_base  =  chain_base + 34,
;    0242  1      mode_base   =  chain_base + 42,
;    0243  1      fs_base     =  MAXBSIZE + 214,
;    0244  1      ndir_base   =  fs_base + 192,
;    0245  1      buf_start   =  214,
;    0246  1      ROOTINO  = 2,   ! i number of all roots
;    0247  1
;    0248  1      NBUFS   =   1,  ! one buffer to read to.
;    0249  1      MAXNAMLEN  =   255,
;    0250  1
;    0251  1      !
;    0252  1      ! The first boot and super blocks are given in absolute disk addresses.
;    0253  1      !
;    0254  1
;    0255  1      ! Super block for a file system.
;    0256  1      !
;    0257  1      FS_MAGIC    =   %X'011954',
;    0258  1
;    0259  1      PT_MAGIC    =   %X'032957',     ! Partition magic number
;    0260  1      B_TAPE      =   1,
;    0261  1      BBSIZE          =  8192,
;    0262  1      DEV_BSIZE       =  512,
;    0263  1      F_FILE          =  %X'8',       ! file instead of device !
;    0264  1      SBLOCK     =    (BBSIZE / DEV_BSIZE),
;    0265  1
;    0266  1      NFILES          =  4,
;    0267  1
;    0268  1
;    0269  1      READVBLK =  %X'21',
;    0270  1      PHYSMODE  = %X'0',
;    0271  1
;    0272  1      READ    = 0,
;    0273  1
;    0274  1      ! io type
;    0275  1      F_RDDATA     =  %X'0100',               ! read data
;    0276  1
;    0277  1      F_TYPEMASK   =  %X'ff00',
;    0278  1      F_ALLOC = %X'4',        ! buffer allocated
;    0279  1
;    0280  1      !  modes
;    0281  1      IFMT    =       %o'0170000',    ! type of file
;    0282  1      IFDIR   =       %o'0040000',    ! directory
;    0283  1      NULL    =       0;
;    0284  1
;    0285  1                                                                          !G:2
;    0286  1      GLOBAL LITERAL                                                      !G:2
;    0287  1      ULTRIX$IOB = FS_BASE + SBSIZE,  ! Buffer size for IOB Buffer  [04]  !G:2
;    0288  1      ULTRIX$B   = NBUFS * MAXBSIZE;  ! Buffer size for B buffer    [04]  !G:2
;    0289  1                                                                          !G:2
;    0290  1      !
```

```
;    0291  1      ! I/O block
;    0292  1      !
;    0293  1      FIELD
;    0294  1       iob_fields=
;    0295  1             SET                                          ! STRUCTURE
;    0296  1        iob$l_i_flgs            =  [0,0,32,0],
;    0297  1        iob$l_i_ino             =  [4,0,32,0],                  ! inode, if file
;    0298  1         iob$z_i_chain          =  [chain_base,0,32,0],         ! array [2] of pointer to struct inode
;    0299  1         iob$w_i_flag           =  [8+chain_base,0,16,0],       !
;    0300  1         iob$w_i_count          =  [10+chain_base,0,16,0],      ! reference count
;    0301  1         iob$w_i_dev            =  [12+chain_base,0,16,0],      ! device where inode resides
;    0302  1         iob$w_i_shlockc        =  [14+chain_base,0,16,0],            ! count of shared locks on inode
;    0303  1         iob$w_i_exlockc        =  [16+chain_base,0,16,0],            ! count of exclusive locks on inode
;    0304  1         iob$l_i_number         =  [18+chain_base,0,32,0],            ! i number, 1-to-1 with device address
;    0305  1         iob$l_i_id             =  [22+chain_base,0,32,0],      ! unique identifier
;    0306  1         iob$l_i_fs             =  [26+chain_base,0,32,0],            ! pointer to struct fs
;    0307  1         iob$l_i_dquot          =  [30+chain_base,0,32,0],      ! pointer to struct dquot
;    0308  1         iob$z_i_unx            =  [34+chain_base,0,32,0],      !
;    0309  1          iob$l_if_lastr        =  [lastr_base,0,32,0],               ! last read [read-ahead]
;    0310  1          iob$l_is_socket       =  [lastr_base,0,32,0],               ! pointer to struct socket
;    0311  1          iob$z_i_fr            =  [lastr_base,0,32,0],               ! file sys associated with this inode
;    0312  1           iob$l_if_freef       =  [lastr_base,0,32,0],               ! pointer to struct inode
;    0313  1           iob$l_if_freeb       =  [4+lastr_base,0,32,0],             ! pointer to pointer to struct inode
;    0314  1         iob$z_i_ic             =  [4+42,0,32,0],               ! inode stucture
;    0315  1      !
;    0316  1      ! inode structure offsets (128 bytes long)
;    0317  1      !
;    0318  1          iob$w_ic_mode          =  [mode_base,0,16,0],                ! mode and type of file
;    0319  1          iob$w_ic_nlink         =  [2+mode_base,0,16,0],              ! number of links to file
;    0320  1          iob$w_ic_uid           =  [4+mode_base,0,16,0],              ! owner's user id
;    0321  1          iob$w_ic_gid           =  [6+mode_base,0,16,0],              ! owner's group id
;    0322  1          iob$z_ic_size          =  [8+mode_base,0,32,0],              ! number of bytes in file
;    0323  1           iob$z_val             =  [8+mode_base,0,32,0],              ! array [2] of long int
;    0324  1         iob$l_ic_atime          =  [16+mode_base,0,32,0],             ! time last accessed
;    0325  1         iob$l_ic_atspare        =  [20+mode_base,0,32,0],             !
;    0326  1         iob$l_ic_mtime          =  [24+mode_base,0,32,0],             ! time last modified
;    0327  1         iob$l_ic_mtspare        =  [28+mode_base,0,32,0],             !
;    0328  1         iob$l_ic_ctime          =  [32+mode_base,0,32,0],             ! last time inode changed
;    0329  1         iob$l_ic_ctspare        =  [36+mode_base,0,32,0],             !
;    0330  1         iob$z_ic_db             =  [40+mode_base,0,32,0],             ! disk block addresses
;    0331  1         iob$z_ic_ib             =  [40+4*NDADDR+mode_base,0,32,0],    ! indirect blocks
;    0332  1         iob$l_ic_flags          =  [40+4*NDADDR+4*NIADDR+mode_base,0,32,0],   ! status (unused)
;    0333  1         iob$l_ic_blocks         =  [104+mode_base,0,32,0],      ! blocks actually held
;    0334  1         iob$z_ic_spare          =  [108+mode_base,0,32,0],      ! reserved (unused) 5 longwords
;    0335  1
;    0336  1
;    0337  1        iob$l_i_unit             =  [174,0,32,0],          ! pseudo device unit
;    0338  1        iob$l_i_boff             =  [178,0,32,0],          ! block offset on device
;    0339  1        iob$l_i_cyloff           =  [182,0,32,0],          ! cylinder offset on device
;    0340  1        iob$l_i_offset           =  [186,0,32,0],          ! seek offset in file
;    0341  1        iob$l_i_bn               =  [190,0,32,0],          ! 1st block # of next read
;    0342  1        iob$l_i_ma               =  [194,0,32,0],          ! memory address of i/o buffer
;    0343  1        iob$l_i_cc               =  [198,0,32,0],          ! character counter of transfer
;    0344  1        iob$l_i_error            =  [202,0,32,0],          ! error # return
;    0345  1        iob$l_i_errcnt           =  [206,0,32,0],          ! error count for driver retries
;    0346  1        iob$l_i_errblk           =  [210,0,32,0],          ! block # in error for error reporting
;    0347  1        iob$z_i_buf              =  [214,0,32,0],          ! array  of MAXBSIZE char
```

```
;   0348  1       iob$z_i_un              =  [214*MAXBSIZE,0,32,0],           !
;   0349  1
;   0350  1     ! Partition SUPERBLOCK
;   0351  1     !
;   0352  1       iob$z_dummy           =  [fs_base,0,32,0],                    ! array of SBSIZE characters
;   0353  1        iob$z_ui_fs           =  [fs_base,0,32,0],                   ! struct fs (overlaps with above buffer)
;   0354  1         iob$l_fs_link         =  [fs_base,0,32,0],                  ! pointer to struct fs
;   0355  1         iob$l_fs_rlink        =  [4+fs_base,0,32,0],                ! pointer to struct fs. used for incore supe
;   0356  1         iob$l_fs_sblkno       =  [8+fs_base,0,32,0],                ! address of super-block in filesys
;   0357  1         iob$l_fs_cblkno       =  [12+fs_base,0,32,0],               ! offset of cylinder in filesys
;   0358  1         iob$l_fs_iblkno       =  [16+fs_base,0,32,0],               ! offset of inode-blocks in filesys
;   0359  1         iob$l_fs_dblkno       =  [20+fs_base,0,32,0],               ! offset of first data after cg
;   0360  1         iob$l_fs_cgoffset     =  [24+fs_base,0,32,0],               ! cylinder group offset in cylinder
;   0361  1         iob$l_fs_cgmask       =  [28+fs_base,0,32,0],               ! used to calc mod fs_ntrak
;   0362  1         iob$l_fs_time         =  [32+fs_base,0,32,0],               ! last time written
;   0363  1         iob$l_fs_size         =  [36+fs_base,0,32,0],               ! number of blocks in fs
;   0364  1         iob$l_fs_dsize        =  [40+fs_base,0,32,0],               ! number of data blocks in fs
;   0365  1         iob$l_fs_ncg          =  [44+fs_base,0,32,0],               ! number of cylinder groups
;   0366  1         iob$l_fs_bsize        =  [48+fs_base,0,32,0],               ! size of basic blocks in fs
;   0367  1         iob$l_fs_fsize        =  [52+fs_base,0,32,0],               ! size of frag blocks in fs
;   0368  1         iob$l_fs_frag         =  [56+fs_base,0,32,0],               ! number of frags in a blocks in fs
;   0369  1         iob$l_fs_minfree      =  [60+fs_base,0,32,0],               ! min % of free blocks. Configuration parame
;   0370  1         iob$l_fs_rotdelay     =  [64+fs_base,0,32,0],               ! # of ms for optimal next block
;   0371  1         iob$l_fs_rpss         =  [68+fs_base,0,32,0],               ! disk revolutions per second
;   0372  1         iob$l_fs_bmask        =  [72+fs_base,0,32,0],               ! "blkoff"  calc of blk offsets. Fields can
;   0373  1         iob$l_fs_fmask        =  [76+fs_base,0,32,0],               ! "fragoff" calc of frag offsets
;   0374  1         iob$l_fs_bshift       =  [80+fs_base,0,32,0],               ! "lblkno" calc of logical blkno
;   0375  1         iob$l_fs_fshift       =  [84+fs_base,0,32,0],               ! "numfrags" calc number of frags
;   0376  1         iob$l_fs_maxcontig    =  [88+fs_base,0,32,0],               ! max # of contiguous blks. Configuration pa
;   0377  1         iob$l_fs_maxbpg       =  [92+fs_base,0,32,0],               ! max # of blks per cyl group
;   0378  1         iob$l_fs_fragshift    =  [96+fs_base,0,32,0],               ! block to frag shift. Fields can be compute
;   0379  1         iob$l_fs_fsbtodb      =  [100+fs_base,0,32,0],          ! fsbtodb and dbtofsb shift constant
;   0380  1         iob$l_fs_sbsize       =  [104+fs_base,0,32,0],          ! actual size of super blocks
;   0381  1         iob$l_fs_csmask       =  [108+fs_base,0,32,0],          ! csum block offset
;   0382  1         iob$l_fs_csshift      =  [112+fs_base,0,32,0],          ! csum block number
;   0383  1         iob$l_fs_nindir       =  [116+fs_base,0,32,0],          ! value of NINDIR
;   0384  1         iob$l_fs_inopb        =  [120+fs_base,0,32,0],          ! value of IOPB
;   0385  1         iob$l_fs_nspf         =  [124+fs_base,0,32,0],          ! value of NSPF
;   0386  1         iob$z_fs_sparecon     =  [128+fs_base,0,32,0],          ! array  of 6 longword
;   0387  1         iob$l_fs_csaddr       =  [152+fs_base,0,32,0],          ! blk addr of cyl grp summary area.Sizes determined
;   0388  1         iob$l_fs_cssize       =  [156+fs_base,0,32,0],          ! size of cyl grp summary area
;   0389  1         iob$l_fs_cgsize       =  [160+fs_base,0,32,0],          ! cylinder group size
;   0390  1         iob$l_fs_ntrak        =  [164+fs_base,0,32,0],          ! track per cylinder. Fields derived from hardware
;   0391  1         iob$l_fs_nsect        =  [168+fs_base,0,32,0],          ! sectors per track
;   0392  1         iob$l_fs_spc          =  [172+fs_base,0,32,0],          ! sectors per cylinder
;   0393  1         iob$l_fs_ncyl         =  [176+fs_base,0,32,0],          ! cylinders in file system. From driver partitioning
;   0394  1         iob$l_fs_cpg          =  [180+fs_base,0,32,0],          ! cylinders per group. Fields computed
;   0395  1         iob$l_fs_ipg          =  [184+fs_base,0,32,0],          ! inodes per group
;   0396  1         iob$l_fs_fpg          =  [188+fs_base,0,32,0],          ! lblocks per group. fs_frag
;   0397  1         iob$z_fs_cstotal      =  [192+fs_base,0,32,0],          ! struct csum. Re-computed after crashes
;   0398  1          iob$l_cs_ndir         =  [ndir_base,0,32,0],                ! number of directories
;   0399  1          iob$l_cs_nbfree       =  [4+ndir_base,0,32,0],              ! number of free blocks
;   0400  1          iob$l_cs_nifree       =  [8+ndir_base,0,32,0],              ! number of free inodes
;   0401  1          iob$l_cs_nffree       =  [12+ndir_base,0,32,0],             ! number of free frags
;   0402  1         iob$b_fs_fmod         =  [208+fs_base,0,8,0],           ! super block modified flag
;   0403  1         iob$b_fs_clean        =  [209+fs_base,0,8,0],           ! file system is clean flag
;   0404  1         iob$b_fs_ronly        =  [210+fs_base,0,8,0],           ! mounted read-only flag
```

```
;    0405  1          iob$b_fs_flags          =  [211+fs_base,0,8,0],                ! currently unused flag
;    0406  1          iob$z_fs_fsmnt          =  [212+fs_base,0,32,0],               ! array of chars. Fields keep current blk allo info
;    0407  1          iob$l_fs_cgrotor        =  [212+fs_base+MAXMNTLEN,0,32,0],          ! last cg searched
;    0408  1          iob$z_fs_csp            =  [212+4+fs_base+MAXMNTLEN,0,32,0],        ! array of fs_cs info buffers
;    0409  1          iob$l_fs_cpc            =  [212+4+fs_base+MAXMNTLEN+4*MAXCSBUFS,0,32,0],     ! cyl per cycle in postbl
;    0410  1          iob$z_fs_postbl         =  [212+8+fs_base+MAXMNTLEN+4*MAXCSBUFS,0,32,0],     ! array of head of blocks of
;    0411  1          iob$l_fs_magic          =  [212+8+fs_base+MAXMNTLEN+4*MAXCSBUFS+2*(MAXCPG*NRPOS),0,32,0],    ! magic number
;    0412  1          iob$b_fs_rotbl          =  [212+12+fs_base+MAXMNTLEN+4*MAXCSBUFS+2*(MAXCPG*NRPOS),0,16,0],     ! list of bl
;    0413  2        iob$l_pt_magic            =  [fs_base+(sbsize
;    0414  2                                          - pt_size                      ! subtract size of partition table since this table
;    0415  2                                                                         ! justified to aligned with the end of the iob super
;    0416  1                                                  ),0,32,0],             ! magic #. Indicating partition info exits.
;    0417  1          iob$l_pt_valid          =  [fs_base+(sbsize-pt_size)+4,0,32,0],     ! Set by driver if pt is current.
;    0418  1          iob$l_pt_part           =  [fs_base+(sbsize-pt_size)+8,0,32,0],     !
;    0419  1
;    0420  1      ! the following two fields are replicated 8 times as a pair. One for each partition.
;    0421  1          iob$l_pi_nblocks        =  [fs_base+(sbsize-pt_size)+8,0,32,0],     ! No. of sectors for the partition  (start o
;    0422  1          iob$l_pi_blkoff         =  [fs_base+(sbsize-pt_size)+12,0,32,0]     ! block offset for start of partition  (star
;    0423  1                     TES;
;    0424  1
;    0425  1
;    0426  1      FIELD
;    0427  1       DIRECT_fields =
;    0428  1              SET                                                   ! STRUCTURE
;    0429  1       direct$l_d_ino             =     [0,0,32,0],                ! inode number of entry
;    0430  1       direct$w_d_reclen          =     [4,0,16,0],                ! length of this record
;    0431  1       direct$w_d_namlen          =     [6,0,16,0],                ! length of string in d_name
;    0432  1       direct$z_d_name            =     [8,0,32,0]                 ! area of MAXNAMLEN+1 of chars
;    0433  1                     TES;
;    0434  1      FIELD
;    0435  1       DIRSTUFF_fields =
;    0436  1              SET                                                   ! STRUCTURE
;    0437  1       DIRSTUFF$l_loc             =     [0,0,32,0],                !
;    0438  1       DIRSTUFF$l_io              =     [4,0,32,0]                 ! pointer to iob
;    0439  1                     TES;
;    0440  1
;    0441  1
;    0442  1      FIELD
;    0443  1       DEVSW_fields =
;    0444  1              SET                                                   ! STRUCTURE
;    0445  1         devsw$l_dv_name          =     [0,0,32,0],                        ! pointer to char
;    0446  1         devsw$l_dv_strategy      =     [4,0,32,0],                        ! pointer to function
;    0447  1         devsw$l_dv_open          =     [8,0,32,0],                        ! pointer to function
;    0448  1         devsw$l_dv_close         =     [12,0,32,0],                       ! pointer to function
;    0449  1         devsw$l_dv_ioctl         =     [16,0,32,0],                       ! pointer to function
;    0450  1         devsw$l_dv_flags         =     [20,0,32,0]                        !
;    0451  1              TES;
;    0452  1
;    0453  1      FIELD
;    0454  1       PT_fields =
;    0455  1              SET                                                   ! STRUCTURE
;    0456  1              pt$l_magic         =     [0,0,32,0],                        ! magic # of sectors for the partition
;    0457  1              pt$l_valid         =     [4,0,32,0],                        ! set by driver if pt is current
;    0458  1              pt$l_part          =     [8,0,32,0],                        ! array or 8
;    0459  1               pt$l_nblocks      =     [8,0,32,0],                        ! # of sectors for the partition
;    0460  1               pt$l_blkoff       =     [12,0,32,0]                        ! block offset for start of partition
;    0461  1              TES;
```

```
;    0462  1
;    0463  1      OWN
;    0464  1      IOB: REF BLOCK[,BYTE] FIELD(IOB_FIELDS), ! Byte block storage [04]                    !G:2
;    0465  1      b: REF BLOCK[,BYTE],                                                                 !G:2
;    0466  1
;    0467  1
;    0468  1      !! DIRECT:      BLOCK[8 + MAXNAMLEN + 1, BYTE]  FIELD(DIRECT_fields),   ! Byte block storage
;    0469  1
;    0470  1      DIRSTUFF:       BLOCK[8, BYTE]  FIELD(DIRSTUFF_fields), ! Byte block storage
;    0471  1
;    0472  1
;    0473  1      !! PT:  BLOCK[pt_size, BYTE]  FIELD(PT_fields), ! Byte block storage
;    0474  1
;    0475  1      DEVSW:  BLOCK[24, BYTE]  FIELD(DEVSW_fields);   ! Byte block storage
;    0476  1
;    0477  1
;    0478  1      GLOBAL BIND                                                                          !G:2
;    0479  1      ULTRIX_IOB_ADD = IOB,                          ! Address of IOB buffer [04]          !G:2
;    0480  1      ULTRIX_B_ADD = B;                              ! Address of B buffer   [04]          !G:2
;    0481  1                                                                                           !G:2
;    0482  1      !
;    0483  1       ! Turn file system block numbers into disk block addresses.
;    0484  1       ! This maps file system blocks to device size blocks.
;    0485  1       !
;    0486  1
;    0487  1      !
;    0488  1      ! INOPB is the number of inodes in a secondary storage block.
;    0489  1
```

```
;    0490  1
;    0491  1      %sbttl 'Advance RFA'
;    0492  1      routine ultrix$advance (len) : jsb_ultrix novalue =
;    0493  1
;    0494  1      !++
;    0495  1      ! Functional description:
;    0496  1      !       Increment the internal 'RFA' (offset and vbn parts) by the
;    0497  1      !       specified number of bytes.
;    0498  1      !
;    0499  1      ! Inputs:
;    0500  1      !       len                     number of bytes to increment by
;    0501  1      !
;    0502  1      ! Implicit inputs:
;    0503  1      !       R7                      vbn
;    0504  1      !
;    0505  1      ! Outputs:
;    0506  1      !       none
;    0507  1      !
;    0508  1      ! Implicit outputs:
;    0509  1      !       R7                      new vbn
;    0510  1      !
;    0511  1      ! Side effects:
;    0512  1      !       none
;    0513  1      !
;    0514  1      ! Return codes:
;    0515  1      !       none
;    0516  1      !--
;    0517  1
;    0518  2          begin
;    0519  2
;    0520  2          external register
;    0521  2             vbn = 7,                        ! Virtual block number
;    0522  2             offset = 8;                     ! offset in VBN
;    0523  2
;    0524  2          vbn = .vbn + ((.offset+.len)/512);  ! Calculate new page
;    0525  2          offset = (.offset + .len) and 511   ! Truncate to offset in page
;    0526  1          end;                                ! of ultrix$advance

                                .TITLE  ULTRIX *** ULTRIX Disk RMS routines
                                .IDENT  \01-05\

                                .PSECT  WORK,NOEXE,2

00000 CC:               .BLKB   4
00004 TRANSFER_SIZE:
                        .BLKB   4
00008 N:                .BLKB   4
0000C I:                .BLKB   4
00010 J:                .BLKB   4
00014 SH:               .BLKB   4
00018 RLEN:             .BLKB   4
0001C LEN:              .BLKB   4
00020 D:                .BLKB   4
00024 LBN:              .BLKB   4
00028 OFF:              .BLKB   4
0002C STATE:            .BLKB   4
```

```
00030 FRAG_SIZE:
                .BLKB   4
00034 STR_SIZE:
                .BLKB   4
00038 STR_PATH:
                .BLKB   4
0003C DIFF:     .BLKB   4
00040 TMP_BOFF:
                .BLKB   4
00044 BOFF:     .BLKB   4
00048 QIO_STATUS:
                .BLKB   4
0004C START_LBN:
                .BLKB   4
00050 BYTE_COUN:
                .BLKB   4
00054 ADRS_MODE:
                .BLKB   4
00058 IO_FUNCTN:
                .BLKB   4
0005C BUFF_ADDR:
                .BLKB   4
00060 Q:        .BLKB   4
00064 BN:       .BLKB   4
00068 NB:       .BLKB   4
0006C BLKNOS:   .BLKB   4
00070 DP:       .BLKB   4
00074 IOB:      .BLKB   4
00078 B:        .BLKB   4
0007C DIRSTUFF:
                .BLKB   8
00084 DEVSW:    .BLKB   24

                ULTRIX$IOB==            16598
                ULTRIX$B==              8192
                ULTRIX_IOB_ADD==        IOB
                ULTRIX_B_ADD==          B
                        .EXTRN  END_SIZE, RMS$ALLOC_CACHE
                        .EXTRN  BOO$QIO, EXE$ALONONPAGED

                        .PSECT  CODE,NOWRT,  SHR,2

                51      58      50  C1 00000 ULTRIX$ADVANCE:
                                             ADDL3   LEN, OFFSET, R1                 ; 0524
                50      51 00000200 8F  C7 00004 DIVL3   #512, R1, R0                ;
                        57      50  C0 0000C  ADDL2   R0, VBN                       ;
        58      51      09      00  EF 0000F  EXTZV   #0, #9, R1, OFFSET            ; 0525
                                    05 00014  RSB                                   ; 0526

; Routine Size:  21 bytes,    Routine Base:  CODE + 0000
```

```
;    0527  1
;    0528  1        %sbttl 'Access file block'
;    0529  1        routine ultrix$access : jsb_ultrix =
;    0530  1
;    0531  1        !++
;    0532  1        ! Functional description:
;    0533  1        !       If the desired VBN is not currently cached, update cache to
;    0534  1        !       contain it.
;    0535  1        !
;    0536  1        ! Inputs:
;    0537  1        !       none
;    0538  1        !
;    0539  1        ! Implicit inputs:
;    0540  1        !       R7                      VBN to cache/access
;    0541  1        !       R9                      pointer to CBLOCK control structure
;    0542  1        !       R10                     pointer to RAB
;    0543  1        !
;    0544  1        ! Outputs:
;    0545  1        !       none
;    0546  1        !
;    0547  1        ! Implicit outputs:
;    0548  1        !       R6                      pointer to 'memory image' of file block
;    0549  1        !
;    0550  1        ! Side effects:
;    0551  1        !       may alter cache control structure of CBLOCK
;    0552  1        !
;    0553  1        ! Return codes:
;    0554  1        !       RMS$_NORMAL             OK
;    0555  1        !       RMS$_RFA                random RFA is illegal
;    0556  1        !       RMS$_EOF                VBN is past end of file
;    0557  1        !       RMS$_RER                File read error
;    0558  1        !--
;    0559  1
;    0560  2            begin
;    0561  2
;    0562  2            external register
;    0563  2               block = 6,
;    0564  2               vbn = 7,
;    0565  2               offset = 8,
;    0566  2               cblock = 9 : ref bblock,
;    0567  2               rab = 10 : ref bblock,
;    0568  2               fab = 11 : ref bblock;
;    0569  2
;    0570  2            local
;    0571  2               read_blocks;                                 ! Blocks to EOF
;    0572  2
;    0573  2            bind
;    0574  2               cache = cblock [ctl$l_cache1] : vector; ! Map for convenience
;    0575  2
;    0576  2            read_blocks = .cblock [ctl$l_eofvbn] - .vbn;
;    0577  2
;    0578  2            if .read_blocks lss 0                        ! If no more blocks
;    0579  3               or (.read_blocks eql 0                    !  or at EOF block
;    0580  3               and .cblock [ctl$w_offbyte] eql 0)        !  and no bytes there
;    0581  2            then
;    0582  2               return
;    0583  2
```

```
;    0584  2                 if .rab [rab$b_rac] eql rab$c_rfa then rms$_rfa else rms$_eof;      ! Check for illegal vbn
;    0585  2
;    0586  2             if .vbn lss .cblock [ctl$l_lvbn]                ! If VBN isn't
;    0587  2                or .vbn geq .cblock [ctl$l_hvbn]             !  cached now
;    0588  2             then
;    0589  3                begin                                        ! then cache it
;    0590  3
;    0591  3                if .cblock [ctl$w_cache_size] eql 0          ! If no cache allocated
;    0592  3                then
;    0593  4                    begin
;    0594  4
;    0595  4                    local
;    0596  4                        stat;
;    0597  4
;    0598  4                    stat = rms$alloc_cache (1);              ! Allocate cache
;    0599  4
;    0600  4                    if not .stat
;    0601  4                    then
;    0602  4                        return rms$_dme                      ! Error if can't
;    0603  3                    end;
;    0604  3
;    0605  3                cblock [ctl$l_lvbn] = 0;                     ! Clear start vbn
;    0606  3                cblock [ctl$l_hvbn] = 0;                     ! Clear high vbn
;    0607  3                read_blocks = min (2,                        ! 3 blocks or to EOF
;    0608  3                    (.read_blocks*512 + .cblock [ctl$w_offbyte] + 511)/512 - 1);
;    0609  3                incr i from 0 to .read_blocks do             ! Read cache buffers
;    0610  4                    begin
;    0611  4
;    0612  4                    local
;    0613  4                        stat;
;    0614  4
;    0615  4                        iob[iob$l_i_ma] = .cache[.i];            ! Buffer to read to
;    0616  4                        iob[iob$l_i_offset] = (.vbn-1 + .i) * 512 ;     ! offset into file in bytes
;    0617  4                        iob[iob$l_i_cc]  =   512 ;               ! # of bytes to read
;    0618  4                        stat = ultrix$getc() ;                   ! Read in data
;    0619  4
;    0620  4                    if not .stat                             ! If it failed
;    0621  4                    then
;    0622  5                        begin
;    0623  5                        rab [rab$l_stv] = .stat;             ! Load QIO return
;    0624  5                        return rms$_rer
;    0625  5                        end
;    0626  5
;    0627  3                    end;
;    0628  3
;    0629  3                cblock [ctl$l_lvbn] = .vbn;                  ! Set new low vbn
;    0630  3                cblock [ctl$l_hvbn] = .vbn + .read_blocks + 1    ! Set new high vbn
;    0631  2                end;
;    0632  2
;    0633  2             block = .cache [.vbn - .cblock [ctl$l_lvbn]];       ! Get buffer pointer
;    0634  2             1                                               ! Return success
;    0635  1             end;                                            ! of ultrix$access
```

```
                                        1C  BB 00000 ULTRIX$ACCESS:
                                                          PUSHR   #^M<R2,R3,R4>                                  ; 0529
                        54       2C     A9  9E 00002     MOVAB   44(CBLOCK), R4                                 ; 0574
          52       44   A9             57  C3 00006     SUBL3   VBN, 68(CBLOCK), READ_BLOCKS                   ; 0576
                                        07  19 0000B     BLSS    1$                                             ; 0578
                                        1D  12 0000D     BNEQ    3$                                             ; 0579
                                 42     A9  B5 0000F     TSTW    66(CBLOCK)                                     ; 0580
                                        18  12 00012     BNEQ    3$                                             ;
                        02       1E     AA  91 00014 1$: CMPB    30(RAB), #2                                    ; 0584
                                        09  12 00018     BNEQ    2$                                             ;
                        50 0001865C     8F  D0 0001A     MOVL    #99932, R0                                     ;
                                        30  11 00021     BRB     5$                                             ;
                        50 0001827A     8F  D0 00023 2$: MOVL    #98938, R0                                     ;
                                        27  11 0002A     BRB     5$                                             ;
                   28   A9             57  D1 0002C 3$: CMPL    VBN, 40(CBLOCK)                                ; 0586
                                        09  19 00030     BLSS    4$                                             ;
                   24   A9             57  D1 00032     CMPL    VBN, 36(CBLOCK)                                ; 0587
                                        03  18 00036     BGEQ    4$                                             ;
                                      0087  31 00038     BRW     10$                                            ;
                                 0A     A9  B5 0003B 4$: TSTW    10(CBLOCK)                                     ; 0591
                                        15  12 0003E     BNEQ    6$                                             ;
                        50              01  D0 00040     MOVL    #1, R0                                         ; 0598
                           00000000G    EF  16 00043     JSB     RMS$ALLOC_CACHE                                ;
                        09              50  E8 00049     BLBS    STAT, 6$                                       ; 0600
                        50 000184D4     8F  D0 0004C     MOVL    #99540, R0                                     ; 0602
                                        79  11 00053 5$: BRB     11$                                            ;
                                 24     A9  7C 00055 6$: CLRQ    36(CBLOCK)                                     ; 0606
          50            52              09  78 00058     ASHL    #9, READ_BLOCKS, R0                            ; 0608
                        51       42     A9  3C 0005C     MOVZWL  66(CBLOCK), R1                                 ;
                        50     01FF C140  9E 00060     MOVAB   511(R1)[R0], R0                                ;
                        50 00000200     8F  C6 00066     DIVL2   #512, R0                                       ;
                                        50  D7 0006D     DECL    R0                                             ;
                        02              50  D1 0006F     CMPL    R0, #2                                         ; 0607
                                        03  15 00072     BLEQ    7$                                             ;
                        50              02  D0 00074     MOVL    #2, R0                                         ;
                        52              50  D0 00077 7$: MOVL    R0, READ_BLOCKS                                ;
                        53              01  CE 0007A     MNEGL   #1, I                                          ; 0609
                                        35  11 0007D     BRB     9$                                             ;
                        50 00000000'    EF  D0 0007F 8$: MOVL    IOB, R0                                        ; 0615
                   00C2 C0            6443  D0 00086     MOVL    (R4)[I], 194(R0)                               ;
                        51       FF A347  9E 0008C     MOVAB   -1(I)[VBN], R1                                 ; 0616
     00BA   C0          51              09  78 00091     ASHL    #9, R1, 186(R0)                                ;
                   00C6 C0     0200     8F  3C 00097     MOVZWL  #512, 198(R0)                                  ; 0617
                           00000000V    EF  16 0009E     JSB     ULTRIX$GETC                                    ; 0618
                        0D              50  E8 000A4     BLBS    STAT, 9$                                       ; 0620
                   0C   AA              50  D0 000A7     MOVL    STAT, 12(RAB)                                  ; 0623
                        50 0001C0F4     8F  D0 000AB     MOVL    #114932, R0                                    ; 0624
                                        1A  11 000B2     BRB     11$                                            ;
          C7            53              52  F3 000B4 9$: AOBLEQ  READ_BLOCKS, I, 8$                             ; 0622
                   28   A9              57  D0 000B8     MOVL    VBN, 40(CBLOCK)                                ; 0629
                   24   A9       01 A247  9E 000BC     MOVAB   1(READ_BLOCKS)[VBN], 36(CBLOCK)                ; 0630
          50            57       28     A9  C3 000C2 10$: SUBL3   40(CBLOCK), VBN, R0                            ; 0633
                        56            6440  D0 000C7     MOVL    (R4)[R0], BLOCK                                ;
                        50              01  D0 000CB     MOVL    #1, R0                                         ; 0635
                                        1C  BA 000CE 11$: POPR    #^M<R2,R3,R4>                                  ;
                                            05 000D0     RSB                                                    ;
```

```
; Routine Size:  209 bytes,    Routine Base:  CODE + 0015
```

```
;    0636  1
;    0637  1       %sbttl 'ULTRIX OPEN service'
;    0638  1
;    0639  1       global routine ultrix$open : call_rms =
;    0640  1
;    0641  1       !++
;    0642  1       ! Functional description:
;    0643  1       !       Access a file on ULTRIX files-11 disk media.  Return file parameters
;    0644  1       !       in the FAB and internal control block associated with the file
;    0645  1       !
;    0646  1       ! Inputs:
;    0647  1       !       none
;    0648  1       !
;    0649  1       ! Implicit inputs:
;    0650  1       !       R11     points to the FAB
;    0651  1       !       R9      points to the CBLOCK (internal RMS control block)
;    0652  1       !
;    0653  1       ! Outputs:
;    0654  1       ! str_path    - Pointer to start of directory "---/---/---..." following ')'.
;    0655  1       ! str_size    - length of directory string
;    0656  1       !
;    0657  1       !
;    0658  1       ! Implicit outputs:
;    0659  1       !       Several fields in the FAB and in the static file control block
;    0660  1       !       are altered to describe the file.  Also, if an XAB (XABFHC is
;    0661  1       !       the only XAB currently supported) exists, it will be filled in.
;    0662  1       !
;    0663  1       ! Side effects:
;    0664  1       !
;    0665  1       !
;    0666  1       ! Return status
;    0667  1       !       RMS$_DME        insufficient memory for buffer allocation
;    0668  1       !       RMS$_FNF        file not found
;    0669  1       !       RMS$_FNM        Bad file name
;    0670  1       !       RMS$_RER        QIO error on file access
;    0671  1       !       RMS$_NORMAL     success
;    0672  1       !--
;    0673  1
;    0674  2           begin
;    0675  2
;    0676  2           local
;    0677  2              stat;
;    0678  2
;    0679  2           external register
;    0680  2              cblock = 9 : ref bblock,        ! Pointer to control block
;    0681  2              rab = 10 : ref bblock,
;    0682  2              fab = 11 : ref bblock;          ! Pointer to FAB
;    0683  2           global register
;    0684  2              block = 6 : ref vector [,byte], ! Pointer to cache buffer
;    0685  2              vbn = 7,                        ! Current VBN in file
;    0686  2              offset = 8;                     ! Current offset in VBN
;    0687  2
;    0688  2
;    0689  2
;    0690  2           cblock [ctl$l_get] = ultrix$get;          ! Set GET routine address
;    0691  2           cblock [ctl$l_read] = ultrix$read;        ! Set READ routine address
;    0692  2           cblock [ctl$l_startlbn] = 0 ;             ! First LBN (if contiguous)
```

```
;    0693  2                                                    ! 0 if not
;    0694  2          fab [fab$b_org] = FAB$C_SEQ;          ! File organization
;    0695  2          fab [fab$b_rfm] = FAB$C_fix;          ! Record format, assume .exe file.
;    0696  2          fab [fab$w_mrs] = 512;                ! Max. rec size
;    0697  2
;    0698  2                                                ! initialize parameters
;    0699  2        iob[iob$l_i_flgs] = .iob[iob$l_i_flgs] OR F_ALLOC;  ! Set as being allocated
;    0700  2        iob[iob$l_i_unit] = 0 ;                 ! Initialize counter.
;    0701  2            iob[iob$l_i_boff] = CH$FIND_CH(.CBLOCK[CTL$W_FILE_LEN], .CBLOCK[CTL$L_FILE_PTR],%C'(')+1;
;    0702  2           str_size  = .CBLOCK[CTL$W_FILE_LEN] - (.iob[iob$l_i_boff] -.CBLOCK[CTL$L_FILE_PTR]) - 1; ! update length
;    0703  2           str_path  = .iob[iob$l_i_boff]+1;     ! update pointer to include pass ')'
;    0704  2
;    0705  2                                                    ! Check partition indicator
;    0706  2                                                                                          !G:2
;    0707  2          IF  .(.iob[iob$l_i_boff])<0,8> GEQ %C'A' AND .(.iob[iob$l_i_boff])<0,8> LEQ %C'G' ! Check for letter only. [0
;    0708  2           THEN                                                                           !G:2
;    0709  2                iob[iob$l_i_unit] =  .(.iob[iob$l_i_boff])<0,8> -%c'A'   ! Change from ascii.  !G:2
;    0710  2           ELSE                                                                           !G:2
;    0711  3             BEGIN                                                                        !G:2
;    0712  3               IF  .(.iob[iob$l_i_boff])<0,8> GEQ %C'a' AND .(.iob[iob$l_i_boff])<0,8> LEQ %C'g' ! Check for letter on
;    0713  3                THEN                                                                      !G:2
;    0714  3                 iob[iob$l_i_unit] =  .(.iob[iob$l_i_boff])<0,8> -%c'a'   ! Change from ascii.  !G:2
;    0715  3                ELSE                                                                      !G:2
;    0716  4                 BEGIN                                                                    !G:2
;    0717  4                  IF  .(.iob[iob$l_i_boff])<0,8> GEQ %C'0' AND .(.iob[iob$l_i_boff])<0,8> LEQ %C'9' ! Check for numbe
;    0718  4                   THEN                                                                   !G:2
;    0719  4                    iob[iob$l_i_unit] =  .(.iob[iob$l_i_boff])<0,8> -%c'0'   ! Change from ascii.  !G:2
;    0720  4                   ELSE                                                                   !G:2
;    0721  4                    RETURN (RMS$_RER) ;                ! Exit with error. Bad unit number.  !G:2
;    0722  3                 END;                                                                     !G:2
;    0723  2             END;                                    !                        [04]        !G:2
;    0724  2                                                                                          !G:2
;    0725  2          str_path = .str_path + 1;          ! Point to next character.                   !G:2
;    0726  2          str_size = .str_size - 1;          ! Reduce length.                             !G:2
;    0727  2          iob[iob$l_i_boff] = .iob[iob$l_i_unit] ;          ! Set in proper field.
;    0728  2           !
;    0729  2           !      If we are opening a tape we don't do the disk partitioning
;    0730  2           !      stuff
;    0731  2           !
;    0732  2          IF  .devsw[devsw$l_dv_flags]  NEQ  B_TAPE
;    0733  2          THEN
;    0734  3                BEGIN
;    0735  3
;    0736  3                !
;    0737  3                !       Before we can actually open up the device we first must
;    0738  3                !       try to read the partition table if  exists
;    0739  3                !
;    0740  3
;    0741  3                tmp_boff = .iob[iob$l_i_boff];
;    0742  3                iob[iob$l_i_boff] = 0;
;    0743  3
;    0744  3                !
;    0745  3                !       Open up the "A" partition
;    0746  3                !
;    0747  3                iob[iob$l_i_ma] = iob[iob$z_ui_fs];                ! set pointer to buffer
;    0748  3                iob[iob$l_i_cc] = SBSIZE;                  ! set size of transfer
;    0749  3                iob[iob$l_i_bn] =  SBLOCK;                 ! set 1st block to read superblock
```

```
;    0750  3                    iob[iob$l_i_offset] = 0;                    ! set offset for seek in file
;    0751  3
;    0752  3                    !
;    0753  3                    !       Read the superblock that may contain the partition
;    0754  3                    !       tables
;    0755  3                    !
;    0756  3                    IF  ultrix$devread()                        ! Read in Superblock, Cylinder group block and I-list.
;    0757  3                    THEN
;    0758  4                            BEGIN                               !
;    0759  4                            !       Now check to see if partition table exists
;    0760  4                            !
;    0761  4                            IF  ( state =ultrix$ptinfo(.tmp_boff) )  EQL DS$_NOSUPPORT
;    0762  4                            THEN
;    0763  4                                    !       Partition table does not exist so just
;    0764  4                                    !       open the device that user has specified
;    0765  4                                    !
;    0766  4                            ELSE
;    0767  4                                    IF .state EQL  DS$_ERROR
;    0768  4                                    THEN RETURN (RMS$_RER)  ;       ! Exit on error.
;    0769  3                            END;
;    0770  3
;    0771  2                    END;
;    0772  2
;    0773  2                    !
;    0774  2                    !       We come here if we are a tape device
;    0775  2                    !       No special work for tape device
;    0776  2                    !       Just fall through as if disk
;    0777  2                    !
;    0778  2                    IF      .str_size LEQ  0
;    0779  2                            THEN
;    0780  3                               BEGIN
;    0781  3                               iob[iob$l_i_flgs]= .iob[iob$l_i_flgs] OR  1 ;
;    0782  3                                    iob[iob$l_i_cc] = 0;
;    0783  3                                    iob[iob$l_i_offset] = 0;
;    0784  3                                    return (RMS$_FNM);
;    0785  2                               END;
;    0786  2                    iob[iob$l_i_ma] = iob[iob$z_ui_fs];     ! Buffer to read to.
;    0787  2                    iob[iob$l_i_cc] = SBSIZE;               ! Character count to read
;    0788  2                    iob[iob$l_i_bn] = SBLOCK + .iob[iob$l_i_boff]; ! Starting block # to read in
;    0789  2                    iob[iob$l_i_offset] = 0;
;    0790  2                    IF  NOT ultrix$devread()                ! Read in superblock for this operation
;    0791  2                    THEN
;    0792  3                            BEGIN
;    0793  3                            iob[iob$l_i_flgs] = 0;
;    0794  3                                                            ! super block read error
;    0795  3                            return (RMS$_RER);
;    0796  2                            END;
;    0797  2
;    0798  2                    IF ( i = ultrix$find())  EQL 0          ! send address of start of "---/---/--..."
;    0799  2                    THEN
;    0800  3                            BEGIN
;    0801  3                            iob[iob$l_i_flgs] = 0;
;    0802  3                            return (RMS$_FNF);              ! File not found
;    0803  2                            END;
;    0804  2                    IF ultrix$openi(.i)  LSS  0             ! Insure current file is set up in IOB structure
;    0805  2                    THEN
;    0806  3                            BEGIN
```

```
;   0807  3                 iob[iob$l_i_flgs] = 0;
;   0808  3                 return (RMS$_FNF);
;   0809  2                 END;
;   0810  2            iob[iob$l_i_flgs] = .iob[iob$l_i_flgs]  OR  (F_FILE OR 1);
;   0811  2                                                    ! Get EOF block, which is
;   0812  2         cblock [ctl$l_eofvbn] =                    ! The EOF block VBN is
;   0813  2            1 + .iob[iob$z_ic_size]/512;            ! Convert bytes to  virtual blocks
;   0814  2         cblock [ctl$w_offbyte] = .iob[iob$z_ic_size] MOD 512; ! Get residual bytes.
;   0815  2            blknos  = 0;                            ! Initialize value flag.
;   0816  2            return  (RMS$_NORMAL);
;   0817  2
;   0818  1         end;                                       ! of ULTRIX$OPEN



                                        01FC 00000          .ENTRY  ULTRIX$OPEN, Save R2,R3,R4,R5,R6,R7,R8      ; 0639
                     54 00000000V  EF   9E 00002          MOVAB   ULTRIX$DEVREAD, R4                          ;
                     53 00000000'  EF   9E 00009          MOVAB   IOB, R3                                     ;
                 0C  A9 00000000V  EF   9E 00010          MOVAB   ULTRIX$GET, 12(CBLOCK)                      ; 0690
                 10  A9 00000000V  EF   9E 00018          MOVAB   ULTRIX$READ, 16(CBLOCK)                     ; 0691
                              50   A9   D4 00020          CLRL    80(CBLOCK)                                  ; 0692
                              1D   AB   94 00023          CLRB    29(FAB)                                     ; 0694
                 1F  AB            01   90 00026          MOVB    #1, 31(FAB)                                 ; 0695
                 36  AB     0200   8F   B0 0002A          MOVW    #512, 54(FAB)                               ; 0696
                     57            63   D0 00030          MOVL    IOB, R7                                     ; 0699
                     67            04   88 00033          BISB2   #4, (R7)                                    ;
                     58     00AE   C7   9E 00036          MOVAB   174(R7), R8                                 ; 0700
                                   68   D4 0003B          CLRL    (R8)                                        ;
                     52     00B2   C7   9E 0003D          MOVAB   178(R7), R2                                 ; 0701
                     56       48   A9   3C 00042          MOVZWL  72(CBLOCK), R6                              ;
         4C  B9      56            2C   3A 00046          LOCC    #44, R6, @76(CBLOCK)                        ;
                                   02   12 0004B          BNEQ    1$                                          ;
                                   51   D4 0004D          CLRL    R1                                          ;
                     62       01   A1   9E 0004F 1$:      MOVAB   1(R1), (R2)                                 ;
                     50            62   D0 00053          MOVL    (R2), R0                                    ; 0702
             51  4C  A9            50   C3 00056          SUBL3   R0, 76(CBLOCK), R1                          ;
                 C0  A3       FF A146   9E 0005B          MOVAB   -1(R1)[R6], STR_SIZE                        ;
                 C4  A3       01   A0   9E 00061          MOVAB   1(R0), STR_PATH                             ; 0703
                 41  8F            60   91 00066          CMPB    (R0), #65                                   ; 0707
                                   12   1F 0006A          BLSSU   2$                                          ;
                 47  8F            60   91 0006C          CMPB    (R0), #71                                   ;
                                   0C   1A 00070          BGTRU   2$                                          ;
                     68            60   9A 00072          MOVZBL  (R0), (R8)                                  ; 0709
                     68 00000041   8F   C2 00075          SUBL2   #65, (R8)                                   ;
                                   2B   11 0007C          BRB     6$                                          ;
                 61  8F            60   91 0007E 2$:      CMPB    (R0), #97                                   ; 0712
                                   12   1F 00082          BLSSU   3$                                          ;
                 67  8F            60   91 00084          CMPB    (R0), #103                                  ;
                                   0C   1A 00088          BGTRU   3$                                          ;
                     68            60   9A 0008A          MOVZBL  (R0), (R8)                                  ; 0714
                     68 00000061   8F   C2 0008D          SUBL2   #97, (R8)                                   ;
                                   13   11 00094          BRB     6$                                          ;
                     30            60   91 00096 3$:      CMPB    (R0), #48                                   ; 0717
                                   03   1E 00099          BGEQU   5$                                          ;
                                009D   31 0009B 4$:      BRW     9$                                          ;
```

```
                                39          60  91 0009E 5$:    CMPB    (R0), #57                          ;
                                            F8  1A 000A1        BGTRU   4$                                 ;
                                68          60  9A 000A3        MOVZBL  (R0), (R8)                         ; 0719
                                68          30  C2 000A6        SUBL2   #48, (R8)                          ;
                                      C4    A3  D6 000A9 6$:    INCL    STR_PATH                           ; 0725
                                      C0    A3  D7 000AC        DECL    STR_SIZE                           ; 0726
                                62          68  D0 000AF        MOVL    (R8), (R2)                         ; 0727
                                01    24    A3  D1 000B2        CMPL    DEVSW+20, #1                       ; 0732
                                            43  13 000B6        BEQL    7$                                 ;
                          CC    A3          62  D0 000B8        MOVL    (R2), TMP_BOFF                     ; 0741
                                            62  D4 000BC        CLRL    (R2)                               ; 0742
                        00C2    C7    20D6  C7  9E 000BE        MOVAB   8406(R7), 194(R7)                  ; 0747
                        00C6    C7    2000  8F  3C 000C5        MOVZWL  #8192, 198(R7)                     ; 0748
                        00BE    C7          10  D0 000CC        MOVL    #16, 190(R7)                       ; 0749
                                      00BA  C7  D4 000D1        CLRL    186(R7)                            ; 0750
                                            64  16 000D5        JSB     ULTRIX$DEVREAD                     ; 0756
                                21          50  E9 000D7        BLBC    R0, 7$                             ;
                                50    CC    A3  D0 000DA        MOVL    TMP_BOFF, R0                       ; 0761
                                   00000000V  EF  16 000DE        JSB     ULTRIX$PTINFO                      ;
                          B8    A3          50  D0 000E4        MOVL    R0, STATE                          ;
                    006600B1    8F          50  D1 000E8        CMPL    R0, #6684849                       ;
                                            0A  13 000EF        BEQL    7$                                 ;
                    00660002    8F    B8    A3  D1 000F1        CMPL    STATE, #6684674                    ; 0767
                                            40  13 000F9        BEQL    9$                                 ;
                                      C0    A3  D5 000FB 7$:    TSTL    STR_SIZE                           ; 0778
                                            16  14 000FE        BGTR    8$                                 ;
                                50          63  D0 00100        MOVL    IOB, R0                            ; 0781
                                60          01  88 00103        BISB2   #1, (R0)                           ;
                                      00C6  C0  D4 00106        CLRL    198(R0)                            ; 0782
                                      00BA  C0  D4 0010A        CLRL    186(R0)                            ; 0783
                                50 0001852C 8F  D0 0010E        MOVL    #99628, R0                         ; 0784
                                            04  00115           RET                                        ;
                                50          63  D0 00116 8$:    MOVL    IOB, R0                            ; 0786
                        00C2    C0    20D6  C0  9E 00119        MOVAB   8406(R0), 194(R0)                  ;
                        00C6    C0    2000  8F  3C 00120        MOVZWL  #8192, 198(R0)                     ; 0787
          00BE    C0    00B2    C0          10  C1 00127        ADDL3   #16, 178(R0), 190(R0)              ; 0788
                                      00BA  C0  D4 0012F        CLRL    186(R0)                            ; 0789
                                            64  16 00133        JSB     ULTRIX$DEVREAD                     ; 0790
                                0B          50  E8 00135        BLBS    R0, 10$                            ;
                                      00    B3  D4 00138        CLRL    @IOB                               ; 0793
                                50 0001C0F4 8F  D0 0013B 9$:    MOVL    #114932, R0                        ; 0795
                                            04  00142           RET                                        ;
                                   00000000V  EF  16 00143 10$:   JSB     ULTRIX$FIND                        ; 0798
                          98    A3          50  D0 00149        MOVL    R0, I                              ;
                                            0E  13 0014D        BEQL    11$                                ;
                                50    98    A3  D0 0014F        MOVL    I, R0                              ; 0804
                                   00000000V  EF  16 00153        JSB     ULTRIX$OPENI                       ;
                                            50  D5 00159        TSTL    R0                                 ;
                                            0B  18 0015B        BGEQ    12$                                ;
                                      00    B3  D4 0015D 11$:   CLRL    @IOB                               ; 0807
                                50 00018292 8F  D0 00160        MOVL    #98962, R0                         ; 0808
                                            04  00167           RET                                        ;
                                50          63  D0 00168 12$:   MOVL    IOB, R0                            ; 0810
                                60          09  88 0016B        BISB2   #9, (R0)                           ;
                    51    36    A0 00000200 8F  C7 0016E        DIVL3   #512, 54(R0), R1                   ; 0813
                          44    A9    01    A1  9E 00177        MOVAB   1(R1), 68(CBLOCK)                  ;
      7E          00    36    A0          01  7A 0017C        EMUL    #1, 54(R0), #0, -(SP)              ; 0814
```

```
          50            50          8E 00000200   8F  7B 00182        EDIV    #512, (SP)+, R0, R0                     ;
                                 42   A9           50  B0 0018B        MOVW    R0, 66(CBLOCK)                          ;
                                       F8          A3  D4 0018F        CLRL    BLKNOS                                  ; 0815
                                      50 00010001  8F  D0 00192        MOVL    #65537, R0                              ; 0816
                                                   04  00199        RET                                             ; 0818

; Routine Size:  410 bytes,    Routine Base:  CODE + 00E6

;   0819  1
;   0820  1
;   0821  1
```

```
;    0823  1      %sbttl 'ultrix$find'                                                    !G:3
;    0824  1      routine ultrix$find  : jsb_ultrix  =
;    0825  1
;    0826  1      !++
;    0827  1      ! Functional description:
;    0828  1      !
;    0829  1      ! Addresses stored in inodes are capable of addressing fragments
;    0830  1      ! of several blocks. File system blocks of at most size MAXBSIZE can
;    0831  1      ! be optionally broken into 2, 4, or 8 pieces, each of which is
;    0832  1      ! addressible; these pieces may be DEV_BSIZE, or some multiple of
;    0833  1      ! a DEV_BSIZE unit.
;    0834  1      !
;    0835  1      ! Large files consist of exclusively large data blocks.  To avoid
;    0836  1      ! undue wasted disk space, the last data block of a small file may be
;    0837  1      ! allocated as only as many fragments of a large block as are
;    0838  1      ! necessary.  The file system format retains only a single pointer
;    0839  1      ! to such a fragment, which is a piece of a single large block that
;    0840  1      ! has been divided.  The size of such a fragment is determinable from
;    0841  1      ! information in the inode.  This field  is filled in when the disk
;    0842  1      ! is created.
;    0843  1      !
;    0844  1      ! The file system shows file size at the fragment level;
;    0845  1      ! to determine number of blocks, aligned fragments are examined.
;    0846  1      !
;    0847  1      ! The root inode is the root of the file system.
;    0848  1      ! Inode 0 can't be used for normal purposes and
;    0849  1      ! historically bad blocks were linked to inode 1,
;    0850  1      ! thus the root inode is 2. (inode 1 is no longer used for
;    0851  1      ! this purpose, however numerous dump tapes make this
;    0852  1      ! assumption, so we are stuck with it).
;    0853  1      !
;    0854  1      ! Inputs:
;    0855  1      !       none
;    0856  1      !
;    0857  1      ! Implicit inputs:
;    0858  1      ! str_path      - Pointer to start of directory "---/---/---..."
;    0859  1      ! str_size      - length of directory string
;    0860  1      ! CBLOCK        - Various fields
;    0861  1      !
;    0862  1      ! Outputs:
;    0863  1      ! inode # or zero if none.
;    0864  1      !
;    0865  1      ! Implicit outputs:
;    0866  1      !
;    0867  1      ! Side effects:
;    0868  1      !
;    0869  1      ! Return status
;    0870  1      !       none
;    0871  1      !
;    0872  1      ! --
;    0873  2      BEGIN
;    0874  2
;    0875  2          Local
;    0876  2              dn_ext : long,                                  ! Default name extension
;    0877  2              prev_n;                                         ! Previous inode #
;    0878  2          external register
;    0879  2              block = 6,
```

```
;    0880  2              vbn = 7,
;    0881  2              offset = 8,
;    0882  2              cblock = 9 : ref bblock,
;    0883  2              rab = 10 : ref bblock,
;    0884  2              fab = 11 : ref bblock;
;    0885  2
;    0886  2
;    0887  2              If ultrix$openi(ROOTINO) LSS 0                  ! ROOTINO = 2
;    0888  2              THEN
;    0889  2                      return (0);
;    0890  2              IF    .Fab[fab$l_dna] NEQ 0                     ! If present Convert to LC
;    0891  2                 THEN
;    0892  2                      dn_ext =  .(.Fab[fab$l_dna] +.Fab[fab$b_dns]-4) OR  '.   ';
;    0893  2              rlen = .str_size ;                 ! initialize remain length
;    0894  2              while .rlen NEQ 0 DO
;    0895  3              BEGIN
;    0896  3                      while  .(.str_path)<0,8>  EQL %C'/' DO
;    0897  4                              BEGIN
;    0898  4                              str_size = .str_size - 1;               ! Go pass "/"
;    0899  4                              str_path = .str_path + 1;
;    0900  3                              END;
;    0901  3                      q = .str_path;
;    0902  3                      rlen = .str_size;                               ! save remaining length in string
;    0903  3                      while .(.q)<0,8> NEQ  %C'/' AND  .str_size NEQ  0  DO
;    0904  4                              BEGIN
;    0905  4                              str_size = .str_size - 1;
;    0906  4                              q = .q + 1;                     ! Find next "/" or EOL
;    0907  3                              END;
;    0908  3                      IF      .str_size eql 0
;    0909  3                      THEN
;    0910  4                      BEGIN
;    0911  4                              str_size = .rlen;                       ! reset to remaing length of string
;    0912  4                              rlen = 0;                       ! signal end of string
;    0913  3                      END;
;    0914  3                      IF (n = ultrix$dlook()) NEQ 0           ! Get inode # for this link
;    0915  3                      THEN
;    0916  4                      BEGIN
;    0917  4                      IF .rlen EQL  0                         ! exit if end of string
;    0918  4                         THEN                                 ! If this is the end of the string and not a directory
;    0919  4                                                              ! and a directory command was given, return previous inode
;    0920  5                              BEGIN
;    0921  6                              IF  (.Fab[fab$l_dna] NEQ 0 AND  .dn_ext  EQL '.dir')
;    0922  5                                THEN
;    0923  6                                BEGIN
;    0924  6                                  IF ultrix$openi(.n) LSS 0  ! Read in to check if this is a directory
;    0925  6                                    THEN
;    0926  7                                      RETURN (0)              ! File not found
;    0927  6                                    ELSE
;    0928  7                                      BEGIN
;    0929  7                                      IF  (.iob[iob$w_ic_mode] AND IFMT)  NEQ  IFDIR  ! Check if directory
;    0930  7                                       THEN
;    0931  8                                       BEGIN
;    0932  8                                        end_size = .Fab[fab$b_fns] -.str_size -1; ! Reflect true file
;    0933  9                                        RETURN (.prev_n)       ! Return previous inode # to be opened
;    0934  8                                       END
;    0935  7                                       ELSE RETURN(.n);       ! Return last inode #
;    0936  7                                      END
```

```
;   0937  6                                    END
;   0938  5                                  ELSE
;   0939  4                                END;
;   0940  4                          IF ultrix$openi(.n) LSS 0        ! Set up IOB to current inode
;   0941  4                          THEN
;   0942  4                                RETURN (0);                ! File not found
;   0943  4                          str_path = .q;                   ! Update pointer in pathname
;   0944  4                          prev_n = .n;                     ! Update previous n value.
;   0945  4                          END
;   0946  3                    ELSE
;   0947  4                          BEGIN
;   0948  4                          IF .rlen EQL  0                  ! exit if end of string
;   0949  4                             THEN                          ! If this is the end of the string and not a directory
;   0950  4                                                           ! and a directory command was given, return previous inode
;   0951  5                            BEGIN
;   0952  5                               IF  (.Fab[fab$l_dna] NEQ 0 AND  .dn_ext  EQL '.dir') AND
;   0953  6                                                  ((.iob[iob$w_ic_mode] AND IFMT)  EQL  IFDIR)
;   0954  5                                 THEN
;   0955  6                                   BEGIN
;   0956  6
;   0957  6                             end_size = .Fab[fab$b_fns] -.str_size ; ! Reflect true file
;   0958  6                                   IF .(.fab[fab$l_fna])<(.end_size-1)*8,8> EQL %C')' ! If only one name,error.
;   0959  7                                     THEN   RETURN (0) ! Error if no string left
;   0960  6                                     ELSE
;   0961  6                                    RETURN (.prev_n);     ! Return previous inode # to be opened
;   0962  6                                   END
;   0963  5                                 ELSE RETURN(0); ! Return error if not directory command
;   0964  5                             END
;   0965  4                              ELSE RETURN (0);
;   0966  3                          END;
;   0967  2                END;
;   0968  2                return (.n);
;   0969  1          END;



                                        0C  BB 00000 ULTRIX$FIND:
                                                          PUSHR   #^M<R2,R3>                                  ; 0824
                        50              02  D0 00002      MOVL    #2, R0                                      ; 0887
                           00000000V    EF  16 00005      JSB     ULTRIX$OPENI                                ;
                                        50  D5 0000B      TSTL    R0                                          ;
                                        03  18 0000D      BGEQ    1$                                          ;
                                      0186  31 0000F      BRW     16$                                         ;
                                  30    AB  D5 00012 1$:  TSTL    48(FAB)                                     ; 0890
                                        11  13 00015      BEQL    2$                                          ;
                        50        35    AB  9A 00017      MOVZBL  53(FAB), R0                                 ; 0892
                        50        30    AB  C0 0001B      ADDL2   48(FAB), R0                                 ;
                 53        FC   A0 2020202E  8F  C9 0001F  BISL3   #538976302, -4(R0), DN_EXT                  ;
                    00000000'   EF 00000000'  EF  D0 00028 2$:  MOVL    STR_SIZE, RLEN                              ; 0893
                                00000000'  EF  D5 00033 3$:  TSTL    RLEN                                        ; 0894
                                        03  12 00039      BNEQ    4$                                          ;
                                      0151  31 0003B      BRW     15$                                         ;
                        2F 00000000'    FF  91 0003E 4$:  CMPB    @STR_PATH, #47                              ; 0896
                                        0E  12 00045      BNEQ    5$                                          ;
                           00000000'    EF  D7 00047      DECL    STR_SIZE                                    ; 0898
```

```
                                 00000000'  EF  D6 0004D          INCL    STR_PATH                                  ; 0899
                                            E9  11 00053          BRB     4$                                        ;
                     00000000'  EF 00000000'  EF  D0 00055 5$:     MOVL    STR_PATH, Q                               ; 0901
                     00000000'  EF 00000000'  EF  D0 00060          MOVL    STR_SIZE, RLEN                            ; 0902
                                2F 00000000'  FF  91 0006B 6$:     CMPB    aQ, #47                                   ; 0903
                                            16  13 00072          BEQL    7$                                        ;
                                 00000000'  EF  D5 00074          TSTL    STR_SIZE                                  ;
                                            0E  13 0007A          BEQL    7$                                        ;
                                 00000000'  EF  D7 0007C          DECL    STR_SIZE                                  ; 0905
                                 00000000'  EF  D6 00082          INCL    Q                                         ; 0906
                                            E1  11 00088          BRB     6$                                        ;
                                 00000000'  EF  D5 0008A 7$:     TSTL    STR_SIZE                                  ; 0908
                                            11  12 00090          BNEQ    8$                                        ;
                     00000000'  EF 00000000'  EF  D0 00092          MOVL    RLEN, STR_SIZE                            ; 0911
                                 00000000'  EF  D4 0009D          CLRL    RLEN                                      ; 0912
                                 00000000V  EF  16 000A3 8$:     JSB     ULTRIX$DLOOK                              ; 0914
                     00000000'  EF            50  D0 000A9          MOVL    R0, N                                     ;
                                            03  12 000B0          BNEQ    9$                                        ;
                                          0080  31 000B2          BRW     13$                                       ;
                                 00000000'  EF  D5 000B5 9$:     TSTL    RLEN                                      ; 0917
                                            52  12 000BB          BNEQ    11$                                       ;
                                       30   AB  D5 000BD          TSTL    48(FAB)                                   ; 0921
                                            4D  13 000C0          BEQL    11$                                       ;
                     7269642E   8F            53  D1 000C2          CMPL    DN_EXT, #1919509550                       ;
                                            44  12 000C9          BNEQ    11$                                       ;
                                50 00000000'  EF  D0 000CB          MOVL    N, R0                                     ; 0924
                                 00000000V  EF  16 000D2          JSB     ULTRIX$OPENI                              ;
                                            50  D5 000D8          TSTL    R0                                        ;
                                            42  19 000DA          BLSS    12$                                       ;
                                50 00000000'  EF  D0 000DC          MOVL    IOB, R0                                   ; 0929
                                50       2E   A0  3C 000E3          MOVZWL  46(R0), R0                                ;
                                50 FFFF0FFF   8F  CA 000E7          BICL2   #-61441, R0                               ;
                     00004000   8F            50  D1 000EE          CMPL    R0, #16384                                ;
                                            03  12 000F5          BNEQ    10$                                       ;
                                          0095  31 000F7          BRW     15$                                       ;
                                50       34   AB  9A 000FA 10$:    MOVZBL  52(FAB), R0                               ; 0932
                                50 00000000'  EF  C2 000FE          SUBL2   STR_SIZE, R0                              ;
          00000000G  EF         50            01  83 00105          SUBB3   #1, R0, END_SIZE                          ;
                                            7B  11 0010D          BRB     14$                                       ; 0935
                                50 00000000'  EF  D0 0010F 11$:    MOVL    N, R0                                     ; 0940
                                 00000000V  EF  16 00116          JSB     ULTRIX$OPENI                              ;
                                            50  D5 0011C          TSTL    R0                                        ;
                                            78  19 0011E 12$:    BLSS    16$                                       ;
                     00000000'  EF 00000000'  EF  D0 00120          MOVL    Q, STR_PATH                               ; 0943
                                52 00000000'  EF  D0 0012B          MOVL    N, PREV_N                                 ; 0944
                                          FEFE  31 00132          BRW     3$                                        ;
                                 00000000'  EF  D5 00135 13$:    TSTL    RLEN                                      ; 0948
                                            5B  12 0013B          BNEQ    16$                                       ;
                                       30   AB  D5 0013D          TSTL    48(FAB)                                   ; 0952
                                            56  13 00140          BEQL    16$                                       ;
                     7269642E   8F            53  D1 00142          CMPL    DN_EXT, #1919509550                       ;
                                            4D  12 00149          BNEQ    16$                                       ;
                                50 00000000'  EF  D0 0014B          MOVL    IOB, R0                                   ; 0953
                                50       2E   A0  3C 00152          MOVZWL  46(R0), R0                                ;
                                50 FFFF0FFF   8F  CA 00156          BICL2   #-61441, R0                               ;
                     00004000   8F            50  D1 0015D          CMPL    R0, #16384                                ;
                                            32  12 00164          BNEQ    16$                                       ;
```

```
        00000000G  EF      34   AB 00000000'  EF  83 00166          SUBB3   STR_SIZE, 52(FAB), END_SIZE                ; 0957
                                50 00000000G  EF  9A 00173          MOVZBL  END_SIZE, R0                               ; 0958
                                              50  D7 0017A          DECL    R0                                         ;
                                50            08  C4 0017C          MULL2   #8, R0                                     ;
     51       2C   BB           08            50  EF 0017F          EXTZV   R0, #8, @44(FAB), R1                       ;
                                29            51  D1 00185          CMPL    R1, #41                                    ;
                                              0E  13 00188          BEQL    16$                                        ;
                                50            52  D0 0018A 14$:     MOVL    PREV_N, R0                                 ; 0961
                                              0B  11 0018D          BRB     17$                                        ; 0963
                                50 00000000'  EF  D0 0018F 15$:     MOVL    N, R0                                      ; 0968
                                              02  11 00196          BRB     17$                                        ;
                                              50  D4 00198 16$:     CLRL    R0                                         ; 0969
                                              0C  BA 0019A 17$:     POPR    #^M<R2,R3>                                 ;
                                                  05 0019C          RSB                                                ;

; Routine Size:  413 bytes,    Routine Base:  CODE + 0280
```

```
;    0970  1
;    0971  1       %sbttl 'ultrix$dlook'
;    0972  1       routine ultrix$dlook  : jsb_ultrix  =
;    0973  1
;    0974  1       !++
;    0975  1       ! Functional description:
;    0976  1       !
;    0977  1       ! Look at each inode in directory
;    0978  1       !
;    0979  1       ! A directory consists of some number of blocks of DIRBLKSIZ(currently 255)
;    0980  1       ! bytes, where DIRBLKSIZ is chosen such that it can be transferred
;    0981  1       ! to disk in a single atomic operation (e.g. 512 bytes on most machines).
;    0982  1       !
;    0983  1       ! Each DIRBLKSIZ byte block contains some number of directory entry
;    0984  1       ! structures, which are of variable length.  Each directory entry has
;    0985  1       ! a struct direct at the front of it, containing its inode number,
;    0986  1       ! the length of the entry, and the length of the name contained in
;    0987  1       ! the entry.  These are followed by the name padded to a 4 byte boundary
;    0988  1       ! with null bytes.  All names are guaranteed null terminated.
;    0989  1       ! The maximum length of a name in a directory is MAXNAMLEN.
;    0990  1       !
;    0991  1       !  All DIRBLKSIZ bytes
;    0992  1       ! in a directory block are claimed by the directory entries.  This
;    0993  1       ! usually results in the last entry in a directory having a large
;    0994  1       ! direct$w_d_reclen.   If the first entry of  a directory block is free,
;    0995  1       !  then its direct$l_d_ino is set to 0.  Entries other than the first
;    0996  1       ! in a directory do not normally have direct$l_d_ino set to 0.
;    0997  1       !
;    0998  1       ! Inputs:
;    0999  1       !       none
;    1000  1       !
;    1001  1       !  Implicit inputs:
;    1002  1       ! str_path      - Pointer to start of directory "---/---/---..."
;    1003  1       ! str_size      - length of directory string
;    1004  1       ! IOB           - Various fields
;    1005  1       !
;    1006  1       ! Outputs:
;    1007  1       !  Number for file inode  or zero if file not found.
;    1008  1       !
;    1009  1       ! Implicit outputs:
;    1010  1       !
;    1011  1       ! Side effects:
;    1012  1       !
;    1013  1       !
;    1014  1       ! Return status
;    1015  1       !       none
;    1016  1       !
;    1017  1       ! --
;    1018  1
;    1019  2      BEGIN
;    1020  2
;    1021  2           external register
;    1022  2              block = 6,
;    1023  2              vbn = 7,
;    1024  2              offset = 8,
;    1025  2              cblock = 9 : ref bblock,
;    1026  2              rab = 10 : ref bblock,
```

```
;   1027  2             fab = 11 : ref bblock;
;   1028  2
;   1029  2
;   1030  2             IF  .str_size EQL NULL                          ! exit if null string
;   1031  2             THEN
;   1032  2                     return (0);
;   1033  2             IF  (.iob[iob$w_ic_mode] AND IFMT)  NEQ  IFDIR          ! check for directory file type.
;   1034  2             THEN
;   1035  2                                                             ! not a directory
;   1036  2                     return (0);
;   1037  2             if  .iob[iob$z_val] EQL 0
;   1038  2             THEN
;   1039  2                     return (0);                             ! zero length directory
;   1040  2
;   1041  2             len = CH$FIND_CH(.str_size, .str_path, %C'/');
;   1042  2             IF .len eql 0                                   ! no delimiter found
;   1043  2             THEN
;   1044  2                     len = .str_size                         ! use remaining length
;   1045  2             ELSE
;   1046  2                     len = .len - .str_path;                 ! get difference if '/' found
;   1047  2             dirstuff[dirstuff$l_loc] = 0;                   ! Initialize accumulator.
;   1048  2
;   1049  2             While (dp = ultrix$readdir() ) NEQ NULL DO      ! get pointer to directory name structure.
;   1050  3             BEGIN
;   1051  3             IF  .(.dp)<0,32>  EQL 0                         ! Check if inode number for this name is zero.
;   1052  3             THEN
;   1053  3
;   1054  3             ELSE                                            ! Test if name matches returned name
;   1055  3                     IF  .(.dp)<48,16>  EQL  .len  AND  CH$COMPARE(.len, .str_path, .(.dp)<48,16>, (.dp)<64,32>) EQL 0
;   1056  3                     THEN
;   1057  3
;   1058  3
;   1059  3                             RETURN (.(.dp)<0,32>);          ! return inode number.
;   1060  2             END;
;   1061  2             RETURN (0);
;   1062  1     END;


                                                3C  BB 00000 ULTRIX$DLOOK:
                                                                    PUSHR   #^M<R2,R3,R4,R5>                        ; 0972
                                52 00000000'    EF  D0 00002        MOVL    STR_SIZE, R2                            ; 1030
                                                5C  13 00009        BEQL    5$                                      ;
                                50 00000000'    EF  D0 0000B        MOVL    IOB, R0                                 ; 1033
                                51        2E    A0  3C 00012        MOVZWL  46(R0), R1                              ;
                                51 FFFF0FFF     8F  CA 00016        BICL2   #-61441, R1                             ;
                    00004000    8F              51  D1 0001D        CMPL    R1, #16384                              ;
                                                7B  12 00024        BNEQ    7$                                      ;
                                          36    A0  D5 00026        TSTL    54(R0)                                  ; 1037
                                                76  13 00029        BEQL    7$                                      ;
        00000000'  FF           52              2F  3A 0002B        LOCC    #47, R2, @STR_PATH                      ; 1041
                                                02  12 00033        BNEQ    1$                                      ;
                                                51  D4 00035        CLRL    R1                                      ;
                    00000000'   EF              51  D0 00037 1$:    MOVL    R1, LEN                                 ;
                                                09  12 0003E        BNEQ    2$                                      ; 1042
```

```
                        00000000'  EF            52  D0 00040          MOVL    R2, LEN                                  ; 1044
                                                 0B  11 00047          BRB     3$                                       ;
                        00000000'  EF 00000000'  EF  C2 00049 2$:      SUBL2   STR_PATH, LEN                            ; 1046
                                      00000000'  EF  D4 00054 3$:      CLRL    DIRSTUFF                                 ; 1047
                                      00000000V  EF  16 0005A 4$:      JSB     ULTRIX$READDIR                           ; 1049
                        00000000'  EF            50  D0 00060          MOVL    R0, DP                                   ;
                                                 38  13 00067 5$:      BEQL    7$                                       ;
                                   54 00000000'  EF  D0 00069          MOVL    DP, R4                                   ; 1051
                                                 64  D5 00070          TSTL    (R4)                                     ;
                                                 E6  13 00072          BEQL    4$                                       ;
00000000'  EF       06   A4        10            00  ED 00074          CMPZV   #0, #16, 6(R4), LEN                      ; 1055
                                                 DA  12 0007E          BNEQ    4$                                       ;
                                   55            01  D0 00080          MOVL    #1, R5                                   ;
      06   A4            00 00000000'  FF 00000000'  EF  2D 00083          CMPC5   LEN, @STR_PATH, #0, 6(R4), 8(R4)         ;
                                            08   A4     00091                                                            ;
                                                 03  1A 00093          BGTRU   6$                                       ;
                                   55            01  D9 00095          SBWC    #1, R5                                   ;
                                                 55  D5 00098 6$:      TSTL    R5                                       ;
                                                 BE  12 0009A          BNEQ    4$                                       ;
                                   50            64  D0 0009C          MOVL    (R4), R0                                 ; 1059
                                                 02  11 0009F          BRB     8$                                       ;
                                                 50  D4 000A1 7$:      CLRL    R0                                       ; 1062
                                                 3C  BA 000A3 8$:      POPR    #^M<R2,R3,R4,R5>                         ;
                                                     05 000A5          RSB                                              ;

; Routine Size:  166 bytes,    Routine Base:  CODE + 041D
```

```
;    1063  1
;    1064  1       %sbttl 'read directory'
;    1065  1       routine ultrix$readdir  : jsb_ultrix  =
;    1066  1
;    1067  1       !++
;    1068  1       ! Functional description:
;    1069  1       !
;    1070  1       ! get next entry in a directory structure.
;    1071  1       !
;    1072  1       ! Inputs:
;    1073  1       !       none
;    1074  1       !
;    1075  1       !  Implicit inputs:
;    1076  1       ! IOB           - Various fields
;    1077  1       !
;    1078  1       ! Outputs:
;    1079  1       ! Pointer to directory name structure, or zero if EOF.
;    1080  1       !
;    1081  1       ! Implicit outputs:
;    1082  1       !
;    1083  1       !
;    1084  1       ! Side effects:
;    1085  1       !
;    1086  1       ! Return status
;    1087  1       !       none
;    1088  1       !
;    1089  1       ! --
;    1090  1
;    1091  2       BEGIN
;    1092  2
;    1093  2            external register
;    1094  2               block = 6,
;    1095  2               vbn = 7,
;    1096  2               offset = 8,
;    1097  2               cblock = 9 : ref bblock,
;    1098  2               rab = 10 : ref bblock,
;    1099  2               fab = 11 : ref bblock;
;    1100  2
;    1101  2               WHILE   1   DO
;    1102  3               BEGIN
;    1103  3                       IF .dirstuff[dirstuff$l_loc]  GEQ  .iob[iob$z_val]      ! Check against # of bytes in files
;    1104  3                            THEN
;    1105  3                            return (NULL);
;    1106  3                       off = .dirstuff[dirstuff$l_loc] AND NOT .iob[iob$l_fs_bmask];  ! find block offset
;    1107  3                       IF  .off EQL 0
;    1108  3                       THEN
;    1109  4                            BEGIN
;    1110  4                            lbn =  (.dirstuff[dirstuff$l_loc] ^ -.iob[iob$l_fs_bshift]); ! Calculate blk # in file to re
;    1111  4                            d = ultrix$sbmap(.lbn);          ! Get disk address
;    1112  4                            IF  .d EQL 0
;    1113  4                                 THEN
;    1114  4                                 return NULL;                ! Error if null
;    1115  5                            iob[iob$l_i_bn] =  (.d ^ .iob[iob$l_fs_fsbtodb]) ! Convert file segment to disk block
;    1116  4                                                             + .iob[iob$l_i_boff];
;    1117  4                            iob[iob$l_i_ma] = iob[iob$z_i_buf];             ! Set buffer address
;    1118  4
;    1119  4                            ! Get size of file segment in characters
```

```
;   1120  5                 IF .lbn GEQ NDADDR  OR  .iob[iob$z_val] GEQ  ((.lbn + 1) ^ .iob[iob$l_fs_bshift])
;   1121  4                  THEN
;   1122  4                      iob[iob$l_i_cc] = .iob[iob$l_fs_bsize]
;   1123  4                  ELSE
;   1124  4                      iob[iob$l_i_cc] = ((.iob[iob$z_val] MOD .iob[iob$l_fs_bmask]) + .iob[iob$l_fs_fsize] - 1) AND
;   1125  4                       .iob[iob$l_fs_fmask];
;   1126  4
;   1127  4                              IF NOT ultrix$devread()          ! Read in data
;   1128  4                              THEN
;   1129  4                                                               ! read error
;   1130  4                                      return (NULL);
;   1131  3                              END;
;   1132  3                      dp = iob[iob$z_i_buf] + .off;            ! Pointer to name direct structure.
;   1133  3                      dirstuff[dirstuff$l_loc] = .dirstuff[dirstuff$l_loc] + .(.dp)<32,16>; ! Set record length of this na
;   1134  3                      IF .(.dp)<0,32> NEQ 0                    ! Check if null inode number.
;   1135  3                      THEN
;   1136  3                      return (.dp);
;   1137  2                 END;
;   1138  2          return (.dp);
;   1139  1          END;



                                                  0C  BB 00000 ULTRIX$READDIR:
                                                                  PUSHR   #^M<R2,R3>                                  ; 1065
                                  52 00000000'  EF  D0 00002      MOVL    IOB, R2                                     ; 1103
                                  51 00000000'  EF  D0 00009 1$:  MOVL    DIRSTUFF, R1                                ;
                          36      A2            51  D1 00010      CMPL    R1, 54(R2)                                  ;
                                                03  19 00014      BLSS    3$                                          ;
                                              00D4  31 00016 2$:  BRW     9$                                          ;
        00000000'  EF             51     211E   C2  CB 00019 3$:  BICL3   8478(R2), R1, OFF                           ; 1106
                                                03  13 00023      BEQL    4$                                          ; 1107
                                              0094  31 00025      BRW     8$                                          ;
                                  50     2126   C2  CE 00028 4$:  MNEGL   8486(R2), R0                                ; 1110
        00000000'  EF             51            50  78 0002D      ASHL    R0, R1, LBN                                 ;
                                  50 00000000'  EF  D0 00035      MOVL    LBN, R0                                     ; 1111
                                     00000000V  EF  16 0003C      JSB     ULTRIX$SBMAP                                ;
                      00000000'   EF            50  D0 00042      MOVL    R0, D                                       ;
                                  51 00000000'  EF  D0 00049      MOVL    D, R1                                       ; 1112
                                                C4  13 00050      BEQL    2$                                          ;
                                  50 00000000'  EF  D0 00052      MOVL    IOB, R0                                     ; 1115
                  51              51     213A   C0  78 00059      ASHL    8506(R0), R1, R1                            ;
                          00BE    C0     00B2 D041  9E 0005F      MOVAB   @178(R0)[R1], 190(R0)                       ; 1116
                          00C2    C0     00D6   C0  9E 00067      MOVAB   214(R0), 194(R0)                            ; 1117
                                  0C 00000000'  EF  D1 0006E      CMPL    LBN, #12                                    ; 1120
                                                14  18 00075      BGEQ    5$                                          ;
                  51 00000000'    EF            01  C1 00077      ADDL3   #1, LBN, R1                                 ;
                  51              51     2126   C0  78 0007F      ASHL    8486(R0), R1, R1                            ;
                                  51       36   A0  D1 00085      CMPL    54(R0), R1                                  ;
                                                09  19 00089      BLSS    6$                                          ;
                          00C6    C0     2106   C0  D0 0008B 5$:  MOVL    8454(R0), 198(R0)                           ; 1122
                                                1F  11 00092      BRB     7$                                          ;
      7E          00        36    A0            01  7A 00094 6$:  EMUL    #1, 54(R0), #0, -(SP)                       ; 1124
      51          51              8E     211E   C0  7B 0009A      EDIV    8478(R0), (SP)+, R1, R1                     ;
                                  51     210A   C0  C0 000A1      ADDL2   8458(R0), R1                                ;
```

```
                                                  51  D7 000A6          DECL    R1                                  ;
                                  53      2122    C0  D2 000A8          MCOML   8482(R0), R3                        ; 1125
          00C6   C0               51              53  CB 000AD          BICL3   R3, R1, 198(R0)                     ;
                                     00000000V    EF  16 000B3 7$:      JSB     ULTRIX$DEVREAD                      ; 1127
                                  31              50  E9 000B9          BLBC    R0, 9$                              ;
                                  52 00000000'    EF  D0 000BC 8$:      MOVL    IOB, R2                             ; 1132
                  50              52 00000000'    EF  C1 000C3          ADDL3   OFF, R2, R0                         ;
                     00000000'    EF      00D6    C0  9E 000CB          MOVAB   214(R0), DP                         ;
                                  50 00000000'    EF  D0 000D4          MOVL    DP, R0                              ; 1133
                                  51        04    A0  3C 000DB          MOVZWL  4(R0), R1                           ;
                     00000000'    EF              51  C0 000DF          ADDL2   R1, DIRSTUFF                        ;
                                                  60  D5 000E6          TSTL    (R0)                                ; 1134
                                                  05  12 000E8          BNEQ    10$                                 ;
                                                FF1C  31 000EA          BRW     1$                                  ;
                                                  50  D4 000ED 9$:      CLRL    R0                                  ; 1139
                                                  0C  BA 000EF 10$:     POPR    #^M<R2,R3>                          ;
                                                      05 000F1          RSB                                         ;

; Routine Size:  242 bytes,    Routine Base:  CODE + 04C3


;   1140  1
```

```
;    1141  1
;    1142  1       %sbttl 'open inode'
;    1143  1       routine ultrix$openi (n) : jsb_ultrix  =
;    1144  1
;    1145  1       !++
;    1146  1       ! Functional description:
;    1147  1       !
;    1148  1       ! n is the inode number of the entry
;    1149  1       !
;    1150  1       ! Directory structure with file lookup and inode. The inode values
;    1151  1       ! are filled into the IOB structure for this file.
;    1152  1       !
;    1153  1       ! Inputs:
;    1154  1       !
;    1155  1       !
;    1156  1       ! Implicit inputs:
;    1157  1       !       R7                          vbn
;    1158  1       !
;    1159  1       ! Outputs:
;    1160  1       !
;    1161  1       ! Character count of inode read in
;    1162  1       !
;    1163  1       ! Implicit outputs:
;    1164  1       !       R7                          new vbn
;    1165  1       !
;    1166  1       ! Side effects:
;    1167  1       !       Some fields in IOB are changed to reflect this inode
;    1168  1       !
;    1169  1       ! Return codes:
;    1170  1       !       none
;    1171  1       !--
;    1172  1
;    1173  2           begin
;    1174  2
;    1175  2           external register
;    1176  2              block = 6,
;    1177  2              vbn = 7,
;    1178  2              offset = 8,
;    1179  2              cblock = 9 : ref bblock,
;    1180  2              rab = 10 : ref bblock,
;    1181  2              fab = 11 : ref bblock;
;    1182  2
;    1183  2
;    1184  2              iob[iob$l_i_offset] = 0;                          ! no offset in file
;    1185  2                                                                 ! Turn file system block numbers into disk block addresses.
;    1186  2                                                                 ! This maps file system blocks to device size blocks.
;    1187  2              iob[iob$l_i_bn] =                                 ! first block # to read
;    1188  4                 ((    .iob[iob$l_fs_fpg] * (.n/.iob[iob$l_fs_ipg]) + .iob[iob$l_fs_cgoffset] *
;    1189  4                       ((.n/.iob[iob$l_fs_ipg]) MOD .iob[iob$l_fs_cgmask]) + .iob[iob$l_fs_iblkno] +
;    1190  4                          (((.n MOD .iob[iob$l_fs_ipg])/.iob[iob$l_fs_inopb]) ^ .iob[iob$l_fs_fragshift] ))
;    1191  2                   ^ .iob[iob$l_fs_fsbtodb])  + .iob[iob$l_i_boff];
;    1192  2              iob[iob$l_i_cc] = .iob[iob$l_fs_bsize];           ! get character count of data to transfer
;    1193  2              iob[iob$l_i_ma] = iob[iob$z_i_buf];               ! get address of buffer to read into for this transfer
;    1194  2              ultrix$devread();                         ! Read this block for directory inode
;    1195  2              cc = .iob[iob$l_i_cc] ;                   ! Get character count
;    1196  2
;    1197  2              INCR    LOOP  FROM 0 TO 128 DO            ! Overlay dat in iob for inode
```

```
;   1198  3          IOB[MODE_BASE+.LOOP,0,8,0] = .(.iob[iob$l_i_ma]+                  ! start of buffer
;   1199  3                        dinode_size*(.n MOD .iob[iob$l_fs_inopb]) +  ! offset to directory inode
;   1200  2                                                        .loop) ; ! move 128 bytes of directory inode.
;   1201  2                  return (.cc);                    ! Return character count
;   1202  1      END;


                                                  0C  BB 00000 ULTRIX$OPENI:
                                                                        PUSHR   #^M<R2,R3>                          ; 1143
                                  53              50  D0 00002          MOVL    R0, R3                              ;
                                  51 00000000'    EF  D0 00005          MOVL    IOB, R1                             ; 1184
                                          00BA    C1  D4 0000C          CLRL    186(R1)                             ;
                  50              53      218E    C1  C7 00010          DIVL3   8590(R1), N, R0                     ; 1188
                  52      2192    C1              50  C5 00016          MULL3   R0, 8594(R1), R2                    ;
        7E        00              50              01  7A 0001C          EMUL    #1, R0, #0, -(SP)                   ; 1189
        50        50              8E      20F2    C1  7B 00021          EDIV    8434(R1), (SP)+, R0, R0             ;
                                  50      20EE    C1  C4 00028          MULL2   8430(R1), R0                        ;
                                  52              50  C0 0002D          ADDL2   R0, R2                              ; 1188
                                  52      20E6    C1  C0 00030          ADDL2   8422(R1), R2                        ; 1189
        7E        00              53              01  7A 00035          EMUL    #1, N, #0, -(SP)                    ; 1190
        50        50              8E      218E    C1  7B 0003A          EDIV    8590(R1), (SP)+, R0, R0             ;
                                  50      214E    C1  C6 00041          DIVL2   8526(R1), R0                        ;
                  50              50      2136    C1  78 00046          ASHL    8502(R1), R0, R0                    ;
                                  52              50  C0 0004C          ADDL2   R0, R2                              ;
                  52              52      213A    C1  78 0004F          ASHL    8506(R1), R2, R2                    ; 1191
                          00BE    C1      00B2 D142   9E 00055          MOVAB   @178(R1)[R2], 190(R1)               ;
                          00C6    C1      2106    C1  D0 0005D          MOVL    8454(R1), 198(R1)                   ; 1192
                          00C2    C1      00D6    C1  9E 00064          MOVAB   214(R1), 194(R1)                    ; 1193
                                      00000000V   EF  16 0006B          JSB     ULTRIX$DEVREAD                      ; 1194
                                  50 00000000'    EF  D0 00071          MOVL    IOB, R0                             ; 1195
                      00000000'   EF      00C6    C0  D0 00078          MOVL    198(R0), CC                         ;
                                                  52  D4 00081          CLRL    LOOP                                ; 1199
        7E        00              53              01  7A 00083 1$:      EMUL    #1, N, #0, -(SP)                    ;
        51        51              8E      214E    C0  7B 00088          EDIV    8526(R0), (SP)+, R1, R1             ;
                  51              51              07  78 0008F          ASHL    #7, R1, R1                          ;
                                  51      00C2    C0  C0 00093          ADDL2   194(R0), R1                         ;
                          2E A240             6241  90 00098          MOVB    (LOOP)[R1], 46(LOOP)[R0]            ; 1198
                  DD              52 00000080     8F  F3 0009E          AOBLEQ  #128, LOOP, 1$                      ;
                                  50 00000000'    EF  D0 000A6          MOVL    CC, R0                              ; 1201
                                                  0C  BA 000AD          POPR    #^M<R2,R3>                          ; 1202
                                                      05 000AF          RSB                                         ;

; Routine Size:  176 bytes,    Routine Base:  CODE + 05B5


;   1203  1
```

```
;    1204  1
;    1205  1       %sbttl 'ultrix$sbmap'                                                    !G:3
;    1206  1       routine ultrix$sbmap (bn) : jsb_ultrix  =
;    1207  1
;    1208  1       !++
;    1209  1       ! Functional description:
;    1210  1       !
;    1211  1       ! Superblock mapping of directory entries.
;    1212  1       !
;    1213  1       ! bn is block # in file.
;    1214  1       !
;    1215  1       ! Inputs:
;    1216  1       !       none
;    1217  1       !
;    1218  1       ! Implicit inputs:
;    1219  1       !       R7                      vbn
;    1220  1       !
;    1221  1       ! Output:
;    1222  1       !
;    1223  1       ! nb is address of block 'bn' on disk
;    1224  1       !
;    1225  1       ! Implicit outputs:
;    1226  1       !       none
;    1227  1       !
;    1228  1       ! Side effects:
;    1229  1       !
;    1230  1       ! Return codes:
;    1231  1       !       none
;    1232  1       ! --
;    1233  2       BEGIN
;    1234  2
;    1235  2           external register
;    1236  2               block = 6,
;    1237  2               vbn = 7,
;    1238  2               offset =8,
;    1239  2               cblock = 9 : ref bblock,
;    1240  2               rab = 10 : ref bblock,
;    1241  2               fab = 11 : ref bblock;
;    1242  2
;    1243  2
;    1244  2               if .bn LSS 0
;    1245  2               THEN
;    1246  3                       BEGIN
;    1247  3                                                               ! bn negative
;    1248  3                       return (0);                             ! Negative blk # not allowed
;    1249  2                       END;
;    1250  2
;    1251  2               !
;    1252  2               ! blocks 0<--->NDADDR are direct blocks
;    1253  2               !
;    1254  2               ! Direct address in inode
;    1255  2               !
;    1256  2               if .bn LSS NDADDR                               ! Check for mapping range of blk #
;    1257  2               THEN
;    1258  3                       BEGIN
;    1259  3                       nb = .iob[40+mode_base+(4*.bn),0,32,0]; ! Calculate disk address of blk #.
;    1260  3                       return (.nb);                           ! Return block address of data.
```

```
;    1261  2                    END;
;    1262  2
;    1263  2            !
;    1264  2            ! addresses NIADDR have single and double indirect blocks.
;    1265  2            ! the first step is to determine how many levels of indirection.
;    1266  2            !
;    1267  2            sh = 1;
;    1268  2            bn = .bn - NDADDR;          ! put in range of indirect pointer list
;    1269  2            j = NIADDR;             ! Initialize.
;    1270  2            WHILE   .j GTR  0 DO
;    1271  3                    BEGIN                       ! find indirect pointer to use
;    1272  3                    sh = .sh *  .iob[iob$l_fs_nindir];      ! Get # of indirects pointers in a file system blocks.
;    1273  3                    if .bn  LSS  .sh                        ! Check if within range.
;    1274  3                    THEN
;    1275  3                        EXITLOOP;
;    1276  3                    bn = .bn - .sh;   ! 1st loop : single indirect, 2nd loop: double indirect, 3rd loop: triple indirect
;    1277  3                    J = .J -1;
;    1278  2                    END;
;    1279  2            if  .j EQL 0
;    1280  2            THEN
;    1281  2                                    ! block number of file too big , error.
;    1282  2                    return (0);                     ! Error exit
;    1283  2
;    1284  2            !
;    1285  2            ! fetch the first indirect block address from the inode
;    1286  2            !
;    1287  2            nb = .iob[40+4*NDADDR+mode_base+(4*(NIADDR-.J)),0,32,0];
;    1288  2            if  .nb  EQL  0
;    1289  2            THEN                    ! block number of file not mapped to disk.
;    1290  2                    return (0);                     ! Error exit
;    1291  2
;    1292  2            !
;    1293  2            ! fetch through the indirect blocks
;    1294  2            !
;    1295  2            WHILE   .j LEQ NIADDR DO
;    1296  3            BEGIN
;    1297  3                    IF  .blknos  NEQ  .nb   ! If accessed before no need to re-read buffer
;    1298  3                    THEN
;    1299  4                        BEGIN
;    1300  4                            iob[iob$l_i_bn] =  (.nb ^ .iob[iob$l_fs_fsbtodb]) + .iob[iob$l_i_boff];
;    1301  4                            iob[iob$l_i_ma] = .b ;  ! buffer to read into.          [05]                    !G:3
;    1302  4                            iob[iob$l_i_cc] = .iob[iob$l_fs_bsize];
;    1303  4                            IF   NOT ultrix$devread()
;    1304  4                            THEN
;    1305  4                                                                                    ! read error
;    1306  4                                return (0);
;    1307  4                            blknos = .nb; ! flag as being currently in the buffer
;    1308  3                        END;
;    1309  3                    sh = .sh/.iob[iob$l_fs_nindir]; ! Set to base for computing offset index.
;    1310  3                    i = (.bn /.sh) MOD  .iob[iob$l_fs_nindir]; ! Compute index into buffer of pointers for this 'bn'.
;    1311  3                    nb = .(.b)<32*.i,32>;           ! Retrive pointer to next part of file or next level of indirect buf
;    1312  3                    IF .nb EQL 0                    ! If zero not mapped.
;    1313  3                    THEN
;    1314  3                                                    ! bn void
;    1315  3                        return (0);                 !  Error exit; requested block not mapped.
;    1316  3            j = .j + 1;
;    1317  2            END;
```

```
;   1318  2          return (.nb);                                  ! Return block address of data on disk.
;   1319  2
;   1320  1     END;



                                                  0C  BB 00000 ULTRIX$SBMAP:
                                                                  PUSHR   #^M<R2,R3>                                  ; 1206
                                  52              50  D0 00002    MOVL    R0, R2                                      ;
                                                  03  18 00005    BGEQ    2$                                          ; 1244
                                                011E  31 00007 1$:  BRW     10$                                         ;
                                  0C              52  D1 0000A 2$:  CMPL    BN, #12                                     ; 1256
                                                  13  18 0000D    BGEQ    4$                                          ;
                                  50 00000000'FF42  DE 0000F    MOVAL   @IOB[BN], R0                                ; 1259
                        00000000'  EF        56   A0  D0 00017    MOVL    86(R0), NB                                  ;
                                                00FD  31 0001F 3$:  BRW     9$                                          ; 1260
                        00000000'  EF              01  D0 00022 4$:  MOVL    #1, SH                                      ; 1267
                                  52              0C  C2 00029    SUBL2   #12, BN                                     ; 1268
                        00000000'  EF              03  D0 0002C    MOVL    #3, J                                       ; 1269
                                  51 00000000'     EF  D0 00033    MOVL    IOB, R1                                     ; 1272
                                     00000000'     EF  D5 0003A 5$:  TSTL    J                                           ; 1270
                                                  21  15 00040    BLEQ    6$                                          ;
                        00000000'  EF      214A    C1  C4 00042    MULL2   8522(R1), SH                                ; 1272
                        00000000'  EF              52  D1 0004B    CMPL    BN, SH                                      ; 1273
                                                  0F  19 00052    BLSS    6$                                          ;
                                  52 00000000'     EF  C2 00054    SUBL2   SH, BN                                      ; 1276
                                     00000000'     EF  D7 0005B    DECL    J                                           ; 1277
                                                  D7  11 00061    BRB     5$                                          ;
                                  50 00000000'     EF  D0 00063 6$:  MOVL    J, R0                                       ; 1279
                                                  9B  13 0006A    BEQL    1$                                          ;
                                  50              50  CE 0006C    MNEGL   R0, R0                                      ; 1287
                        00000000'  EF      0092 C140  D0 0006F    MOVL    146(R1)[R0], NB                             ;
                                                  8C  13 00079    BEQL    1$                                          ; 1288
                                  53 00000000'     EF  D0 0007B    MOVL    NB, R3                                      ; 1297
                                  03 00000000'     EF  D1 00082 7$:  CMPL    J, #3                                       ; 1295
                                                  94  14 00089    BGTR    3$                                          ;
                                  53 00000000'     EF  D1 0008B    CMPL    BLKNOS, R3                                  ; 1297
                                                  39  13 00092    BEQL    8$                                          ;
                                  50 00000000'     EF  D0 00094    MOVL    IOB, R0                                     ; 1300
                        51        53      213A    C0  78 0009B    ASHL    8506(R0), R3, R1                            ;
                            00BE  C0      00B2 D041  9E 000A1    MOVAB   @178(R0)[R1], 190(R0)                       ;
                            00C2  C0 00000000'     EF  D0 000A9    MOVL    B, 194(R0)                                  ; 1301
                            00C6  C0      2106    C0  D0 000B2    MOVL    8454(R0), 198(R0)                           ; 1302
                                     00000000V     EF  16 000B9    JSB     ULTRIX$DEVREAD                              ; 1303
                                  66              50  E9 000BF    BLBC    R0, 10$                                     ;
                        00000000'  EF 00000000'     EF  D0 000C2    MOVL    NB, BLKNOS                                  ; 1307
                                  51 00000000'     EF  D0 000CD 8$:  MOVL    IOB, R1                                     ; 1309
                        00000000'  EF      214A    C1  C6 000D4    DIVL2   8522(R1), SH                                ;
                        50        52 00000000'     EF  C7 000DD    DIVL3   SH, BN, R0                                  ; 1310
          7E            00        50              01  7A 000E5    EMUL    #1, R0, #0, -(SP)                           ;
          50            50        8E      214A    C1  7B 000EA    EDIV    8522(R1), (SP)+, R0, R0                     ;
                        00000000'  EF              50  D0 000F1    MOVL    R0, I                                       ;
                        50 00000000'  EF              05  78 000F8    ASHL    #5, I, R0                                   ; 1311
00000000'  EF 00000000'  FF        20              50  EF 00100    EXTZV   R0, #32, @B, NB                             ;
                                  53 00000000'     EF  D0 0010D    MOVL    NB, R3                                      ; 1312
```

```
                                  12  13 00114          BEQL    10$                                  ;
                       00000000'  EF  D6 00116          INCL    J                                    ; 1316
                                FF63  31 0011C          BRW     7$                                   ;
                    50 00000000'  EF  D0 0011F 9$:      MOVL    NB, R0                               ; 1318
                                  02  11 00126          BRB     11$                                  ;
                                  50  D4 00128 10$:     CLRL    R0                                   ; 1320
                                  0C  BA 0012A 11$:     POPR    #^M<R2,R3>                           ;
                                      05 0012C          RSB                                          ;

; Routine Size:  301 bytes,    Routine Base:  CODE + 0665
```

```
;    1321  1
;    1322  1       %sbttl 'ultrix$ptinfo'
;    1323  1       routine ultrix$ptinfo( boff) : jsb_ultrix  =
;    1324  1
;    1325  1       !++
;    1326  1       ! Functional description:
;    1327  1       !
;    1328  1       !       Called by ULTRIX$OPEN routine to get the partition table
;    1329  1       !       information to determine if another SUPERBLOCK should be
;    1330  1       !       read in to find the specified partition.
;    1331  1       ! Inputs:
;    1332  1       !       Boff    - Partition specifier for partition A -- H. (0 -- 7)
;    1333  1       !
;    1334  1       ! Implicit inputs:
;    1335  1       !       R7                          vbn
;    1336  1       !
;    1337  1       ! Output:
;    1338  1       !
;    1339  1       !
;    1340  1       ! Implicit outputs:
;    1341  1       !  IOB various fields in this structure
;    1342  1       !
;    1343  1       ! Side effects:
;    1344  1       !
;    1345  1       ! Return codes:
;    1346  1       !       RMS$_NORMAL             Partition table found.
;    1347  1       !       RMS$_NOSUPPORT          Partition table not support in this disk
;    1348  1       !       RMS$_ERROR              This is not a valid SUPERBLOCK
;    1349  1       ! --
;    1350  2       BEGIN
;    1351  2
;    1352  2            external register
;    1353  2               block = 6,
;    1354  2               vbn = 7,
;    1355  2               offset = 8,
;    1356  2               cblock = 9 : ref bblock,
;    1357  2               rab = 10 : ref bblock,
;    1358  2               fab = 11 : ref bblock;
;    1359  2
;    1360  2
;    1361  2               !
;    1362  2               !       Check to see if the block that was read in is a superblock
;    1363  2               !
;    1364  2               IF    .iob[iob$l_fs_magic]  EQL FS_MAGIC
;    1365  2               THEN
;    1366  3                       BEGIN
;    1367  3                       !
;    1368  3                       !       Get the possible partition table
;    1369  3                       !
;    1370  3                       !
;    1371  3                       !       Id the A real partition table
;    1372  3                       !
;    1373  3                       IF   .iob[iob$l_PT_MAGIC]  EQL  PT_MAGIC
;    1374  3                       THEN
;    1375  4                               BEGIN
;    1376  4                               !
;    1377  4                               !
```

```
;   1378  4                         !       We have a real partition table so now
;   1379  4                         !       set the driver's partition table index
;   1380  4                         !
;   1381  4                         iob[iob$l_i_boff] = .iob[(fs_base+(sbsize-pt_size)+12+(8*.boff)),0,32,0];
;   1382  4                         return(DS$_NORMAL);             ! Partition tables are supported
;   1383  4                         END
;   1384  3                 ELSE
;   1385  3                         return(DS$_NOSUPPORT);          ! No partition table on this disk. Use current superblock.
;   1386  3                 END
;   1387  2             ELSE
;   1388  2                 return(DS$_ERROR);                      ! Invalid superblock data read; abort operation.
;   1389  2
;   1390  1     END;



                        51 00000000'  EF  D0 00000 ULTRIX$PTINFO:
                                                          MOVL    IOB, R1                                 ; 1364
                00011954  8F     2632  C1  D1 00007       CMPL    9778(R1), #72020                        ;
                                       25  12 00010       BNEQ    2$                                      ;
                00032957  8F     408E  C1  D1 00012       CMPL    16526(R1), #207191                      ; 1373
                                       12  12 0001B       BNEQ    1$                                      ;
                                 409A C140  7F 0001D      PUSHAQ  16538(R1)[BOFF]                         ; 1381
                    00B2  C1           9E  D0 00022       MOVL    @(SP)+, 178(R1)                         ;
                        50 00660001   8F  D0 00027       MOVL    #6684673, R0                            ; 1385
                                          05 0002E       RSB                                             ;
                        50 006600B1   8F  D0 0002F 1$:   MOVL    #6684849, R0                            ;
                                          05 00036       RSB                                             ; 1388
                        50 00660002   8F  D0 00037 2$:   MOVL    #6684674, R0                            ;
                                          05 0003E       RSB                                             ; 1390

; Routine Size:  63 bytes,    Routine Base:  CODE + 0792


;   1391  1
```

```
;    1392  1
;    1393  1       %sbttl 'ultrix$devread'
;    1394  1       routine ultrix$devread : jsb_ultrix  =
;    1395  1
;    1396  1       !++
;    1397  1       ! Functional description:
;    1398  1       !
;    1399  1       ! Read the specified number of bytes from the disk.
;    1400  1       !
;    1401  1       ! Inputs:
;    1402  1       !       none
;    1403  1       !
;    1404  1       ! Implicit inputs:
;    1405  1       !       iob[iob$l_i_ma]         Address of buffer to write data to.
;    1406  1       !       iob[iob$l_i_cc]         Character count of data to read.
;    1407  1       !       iob[iob$l_i_bn]         Block number to start reading from.
;    1408  1       ! Outputs:
;    1409  1       ! Character count of data read in.
;    1410  1       !
;    1411  1       ! Implicit output:
;    1412  1       !       none
;    1413  1       !
;    1414  1       ! Side effects:
;    1415  1       ! The device is accessed.
;    1416  1       !
;    1417  1       ! Return codes:
;    1418  1       !       RMS$_NORMAL             if OK                                   !G:3
;    1419  1       !       RMS$_RER                if system read error (QIO)              !G:3
;    1420  1       !
;    1421  1       ! --
;    1422  2       BEGIN
;    1423  2
;    1424  2            external register
;    1425  2               block = 6,
;    1426  2               vbn = 7,
;    1427  2               offset = 8,
;    1428  2               cblock = 9 : ref bblock,
;    1429  2               rab = 10 : ref bblock,
;    1430  2               fab = 11: ref bblock;
;    1431  2
;    1432  2
;    1433  2               iob[iob$l_i_flgs] = .iob[iob$l_i_flgs] OR F_RDDATA;
;    1434  2               iob[iob$l_i_error] = 0;
;    1435  2               start_lbn = .iob[iob$l_i_bn];
;    1436  2               byte_coun = .iob[iob$l_i_cc];
;    1437  2               buff_addr = .iob[iob$l_i_ma];
;    1438  2               adrs_mode =  PHYSMODE;
;    1439  2               io_functn = READVBLK;
;    1440  2
;    1441  2               qio_status = boo$qio(.buff_addr, .byte_coun,  .start_lbn, .io_functn, .adrs_mode, .cblock[ctl$l_rpb]);
;    1442  2
;    1443  2
;    1444  2               cc = .iob[iob$l_i_cc];
;    1445  2               iob[iob$l_i_flgs] = .iob[iob$l_i_flgs] AND  (NOT F_TYPEMASK);   ! set flag
;    1446  2               IF NOT .qio_status
;    1447  2                 THEN
;    1448  2                    RETURN  RMS$_RER
```

```
;   1449  2                    ELSE
;   1450  2                RETURN  RMS$_NORMAL;
;   1451  1        END;



                        50 00000000'  EF  D0 00000 ULTRIX$DEVREAD:
                                                        MOVL    IOB, R0                          ; 1433
                  01    A0            01  88 00007      BISB2   #1, 1(R0)                        ;
                              00CA    C0  D4 0000B      CLRL    202(R0)                          ; 1434
            00000000'   EF    00BE    C0  D0 0000F      MOVL    190(R0), START_LBN               ; 1435
            00000000'   EF    00C6    C0  D0 00018      MOVL    198(R0), BYTE_COUN               ; 1436
            00000000'   EF    00C2    C0  D0 00021      MOVL    194(R0), BUFF_ADDR               ; 1437
                        00000000'     EF  D4 0002A      CLRL    ADRS_MODE                        ; 1438
            00000000'   EF            21  D0 00030      MOVL    #33, IO_FUNCTN                   ; 1439
                                38    A9  DD 00037      PUSHL   56(CBLOCK)                       ; 1441
                        00000000'     EF  DD 0003A      PUSHL   ADRS_MODE                        ;
                        00000000'     EF  DD 00040      PUSHL   IO_FUNCTN                        ;
                        00000000'     EF  DD 00046      PUSHL   START_LBN                        ;
                        00000000'     EF  DD 0004C      PUSHL   BYTE_COUN                        ;
                        00000000'     EF  DD 00052      PUSHL   BUFF_ADDR                        ;
            00000000G   EF            06  FB 00058      CALLS   #6, BOO$QIO                      ;
            00000000'   EF            50  D0 0005F      MOVL    R0, QIO_STATUS                   ;
                        50 00000000'  EF  D0 00066      MOVL    IOB, R0                          ; 1444
            00000000'   EF    00C6    C0  D0 0006D      MOVL    198(R0), CC                      ;
                                01    A0  94 00076      CLRB    1(R0)                            ; 1445
                        08 00000000'  EF  E8 00079      BLBS    QIO_STATUS, 1$                   ; 1446
                        50 0001C0F4   8F  D0 00080      MOVL    #114932, R0                      ; 1450
                                          05 00087      RSB                                      ;
                        50 00010001   8F  D0 00088 1$:  MOVL    #65537, R0                       ;
                                          05 0008F      RSB                                      ; 1451

; Routine Size:  144 bytes,    Routine Base:  CODE + 07D1


;   1452  1
```

```
;    1453  1
;    1454  1       %sbttl 'ULTRIX$GET routine'
;    1455  1
;    1456  1       global routine ultrix$get : call_rms =
;    1457  1
;    1458  1       !++
;    1459  1       ! Functional description:
;    1460  1       !       De-block and return a record from an ULTRIX file.
;    1461  1       !
;    1462  1       ! Inputs:
;    1463  1       !       none
;    1464  1       !
;    1465  1       ! Implicit inputs:
;    1466  1       !       R9                      CBLOCK pointer
;    1467  1       !       R10                     RAB pointer
;    1468  1       !       R11                     FAB pointer
;    1469  1       !       several fields in the above data structures
;    1470  1       !
;    1471  1       ! Outputs:
;    1472  1       !       none
;    1473  1       !
;    1474  1       ! Implicit outputs:
;    1475  1       !       RAB [rab$l_stv]         Set to error status of failing system service
;    1476  1       !       RAB [rab$l_rbf]         Set to user buffer
;    1477  1       !       RAB [rab$w_rsz]         Size of record copied to user buffer
;    1478  1       !
;    1479  1       ! Side effects:
;    1480  1       !       CBLOCK [ctl$w_rfa]      advanced one record if access was not
;    1481  1       !                               random-by-RFA
;    1482  1       !       CBLOCK                  cache control and cache buffers may be
;    1483  1       !                               altered if part or all of the record is
;    1484  1       !                               not within a block currently cached.
;    1485  1       !
;    1486  1       ! Return codes:
;    1487  1       !       RMS$_NORMAL             if OK
;    1488  1       !       RMS$_RER                if system read error (QIO)
;    1489  1       !       RMS$_RFA                if random RFA is beyond EOF
;    1490  1       !       RMS$_EOF                if RFA is beyond end of file
;    1491  1       !       RMS$_IRC                Illegal record
;    1492  1       !       RMS$_RTB                Record is too big for user buffer (truncated)
;    1493  1       !--
;    1494  1
;    1495  2           begin
;    1496  2
;    1497  2           external register
;    1498  2              cblock = 9 : ref bblock,                 ! Pointer to CBLOCK
;    1499  2              rab = 10 : ref bblock,                   ! Pointer to RAB
;    1500  2              fab = 11 : ref bblock;                   ! Pointer to FAB
;    1501  2
;    1502  2           global register
;    1503  2              block = 6 : ref vector [, byte],         ! Pointer to cache buffer
;    1504  2              vbn = 7,                                 ! Current VBN in file
;    1505  2              offset = 8;                              ! Current offset in VBN
;    1506  2
;    1507  2           local
;    1508  2              dn_ext : long,                           ! Default filename extension.
;    1509  2              LF_FLAG : byte,                          ! Line feed flag
```

```
;   1510  2              f_size : word,                          ! Full size of record
;   1511  2              size : long,                            ! Size of record
;   1512  2              lenlim : word,                          ! Size of user buffer
;   1513  2              ptr;                                    ! Address in user buffer
;   1514  2
;   1515  2           vbn = .cblock [ctl$l_vbn];                 ! Extract vbn from next RFA
;   1516  2           offset = .cblock [ctl$w_byte];             ! Extract byte offset
;   1517  2
;   1518  2              IF   .Fab[fab$l_dna] NEQ 0                      ! If present Convert to LC
;   1519  2                 THEN
;   1520  2                      dn_ext =  .(.Fab[fab$l_dna] +.Fab[fab$b_dns]-4) OR '.   ';
;   1521  2
;   1522  2              if .rab [rab$b_rac] neq rab$c_rfa       ! If not random access
;   1523  2              then
;   1524  2                  ultrix$advance (.cblock [ctl$w_rec_size]);  ! Else advance to next
;   1525  2
;   1526  2              cblock [ctl$l_vbn] = .vbn;              ! Update CBLOCK vbn
;   1527  2              cblock [ctl$w_byte] = .offset;          !  and it's offset
;   1528  2
;   1529  2              if .vbn gtr .cblock [ctl$l_eofvbn] or .vbn eql .cblock [ctl$l_eofvbn] and .offset geq
;   1530  2                                                      .cblock [ctl$w_offbyte]         ! If past file EOF
;   1531  2              then
;   1532  2                  return rms$_eof;                    ! can't read past EOF
;   1533  2
;   1534  3              (
;   1535  3
;   1536  3              local
;   1537  3                  stat;
;   1538  3
;   1539  3              stat = ultrix$access ();                        ! Load the block in cache
;   1540  2              if not .stat then return .stat);
;   1541  2              cblock [ctl$w_rec_size] =               ! Set size of current record
;   1542  4              (if .fab [fab$l_dna] neq 0 AND (.dn_ext  EQL '.dir')
;   1543  3              then
;   1544  4                  begin
;   1545  4                  size = .(block [.offset])<32, 16>;  ! then fetch record size
;   1546  4                  if .size EQL 0 Then RETURN rms$_eof;
;   1547  4                  .size
;   1548  4                  end
;   1549  2              else size = .fab [fab$w_mrs]);          ! else get fixed size from FAB
;   1550  2
;   1551  2           ptr = rab [rab$l_rbf] = .rab [rab$l_ubf];  ! Point to user buffer
;   1552  2           lenlim = .rab [rab$w_usz];                 ! Size limit
;   1553  2           rab [rab$w_rsz] = 0;                       ! Init output size
;   1554  2           f_size = .size;                            ! Store size, since SIZE moves
;   1555  2
;   1556  2           while 1 do                                      ! Copy over the record
;   1557  3              begin
;   1558  3
;   1559  3              local
;   1560  3                  length;
;   1561  3
;   1562  4              (
;   1563  4              local
;   1564  4                  stat;
;   1565  4
;   1566  4              stat = ultrix$access ();                    ! Be sure VBN is there
```

```
;    1567  3              if not .stat then return .stat);
;    1568  3              LF_FLAG = 0;                                    ! clear flag
;    1569  4              IF   .Fab[fab$l_dna]   NEQ 0  AND  ( .dn_ext  EQL '.hlp'
;    1570  4                                              OR  .dn_ext  EQL '.com')
;    1571  3              then
;    1572  4                      begin
;    1573  4                      IF (size = CH$FIND_CH(512-.offset , block[.offset], %x'0a') ) EQL 0 ! search for line feed character
;    1574  4
;    1575  4                      THEN
;    1576  5                              size =   (IF .VBN EQL .cblock[ctl$l_eofvbn]
;    1577  5                                        THEN  512 - .cblock[ctl$w_offbyte]- .offset   ! Set to EOF or
;    1578  5                                         ELSE 512 - .offset )                         ! remainder of this block
;    1579  5
;    1580  4                      ELSE
;    1581  5                              begin
;    1582  5                              size = .size - BLOCK[.offset]+1;        ! Character found.
;    1583  5                              LF_FLAG = 1;                            ! Set flag for char found.
;    1584  4                              end;
;    1585  3                      end;
;    1586  3              length = min ( .size, (512 - .offset), .lenlim);        ! Size to copy
;    1587  3              ptr = ch$move (.length, block [.offset], .ptr); ! Copy at least part of record
;    1588  3              rab [rab$w_rsz] = .rab [rab$w_rsz] + .length;   ! Update actual length so far
;    1589  3              size = .size - .length;                         ! Rest of record
;    1590  3              cblock [ctl$w_rec_size] = .rab[rab$w_rsz]  ;    ! Set to record size + LF.
;    1591  3
;    1592  3              if .size eql 0
;    1593  3              THEN
;    1594  4              BEGIN
;    1595  4              IF  (.LF_FLAG NEQ 0 AND .fab[fab$l_dna] NEQ 0) AND
;    1596  5                                      ( .dn_ext  EQL '.hlp'
;    1597  5                                        OR  .dn_ext  EQL '.com')
;    1598  5
;    1599  4              THEN                                                    ! If LF found on help or com file proceed.
;    1600  5                      BEGIN
;    1601  5                       rab [rab$w_rsz] = .rab [rab$w_rsz] -1;         !  No LF char send.
;    1602  5                      return rms$_normal;                             ! Return if done
;    1603  5                      END
;    1604  4              ELSE
;    1605  5              IF   .Fab[fab$l_dna]   NEQ 0  AND (.dn_ext  EQL '.hlp'
;    1606  5                                              OR .dn_ext  EQL '.com')
;    1607  4                      THEN                                            ! Continue searching for LF
;    1608  4                      ELSE
;    1609  4                      return rms$_normal
;    1610  4              ;                                                       ! Return if done
;    1611  3              END;
;    1612  3
;    1613  3              lenlim = .lenlim - .length;             ! Buffer size left
;    1614  3
;    1615  3              if .lenlim eql 0                        ! If no space left
;    1616  3              then
;    1617  4                  begin
;    1618  4                  rab [rab$l_stv] = .f_size;          ! Return true size
;    1619  4                  return rms$_rtb
;    1620  3                  end;
;    1621  3
;    1622  3              ultrix$advance (.length)                        ! Skip on to next block
;    1623  2              end;
```

```
;   1624  2
;   1625  2          rms$_normal
;   1626  1          end;



                                                  01FC 00000          .ENTRY  ULTRIX$GET, Save R2,R3,R4,R5,R6,R7,R8        ; 1456
                                 5E             1C  C2 00002          SUBL2   #28, SP                                      ;
                                 57        18   A9  D0 00005          MOVL    24(CBLOCK), VBN                              ; 1515
                                 58        1C   A9  3C 00009          MOVZWL  28(CBLOCK), OFFSET                           ; 1516
                                           30   AB  D5 0000D          TSTL    48(FAB)                                      ; 1518
                                                11  13 00010          BEQL    1$                                           ;
                                 50        35   AB  9A 00012          MOVZBL  53(FAB), R0                                  ; 1520
                                 50        30   AB  C0 00016          ADDL2   48(FAB), R0                                  ;
                     6E       FC   A0 2020202E  8F  C9 0001A          BISL3   #538976302, -4(R0), DN_EXT                   ;
                                 02        1E   AA  91 00023 1$:      CMPB    30(RAB), #2                                  ; 1522
                                                08  13 00027          BEQL    2$                                           ;
                                 50        1E   A9  3C 00029          MOVZWL  30(CBLOCK), R0                               ; 1524
                                         F76E   CF  16 0002D          JSB     ULTRIX$ADVANCE                               ;
                             18   A9            57  D0 00031 2$:      MOVL    VBN, 24(CBLOCK)                              ; 1526
                             1C   A9            58  B0 00035          MOVW    OFFSET, 28(CBLOCK)                           ; 1527
                             44   A9            57  D1 00039          CMPL    VBN, 68(CBLOCK)                              ; 1529
                                                29  14 0003D          BGTR    4$                                           ;
                                                08  12 0003F          BNEQ    3$                                           ;
        58       42   A9            10            00  ED 00041          CMPZV   #0, #16, 66(CBLOCK), OFFSET                  ; 1530
                                                1F  15 00047          BLEQ    4$                                           ;
                                         F767   CF  16 00049 3$:      JSB     ULTRIX$ACCESS                                ; 1539
                                 49            50  E9 0004D          BLBC    STAT, 8$                                     ; 1540
                                           30   AB  D5 00050          TSTL    48(FAB)                                      ; 1542
                                                1B  13 00053          BEQL    5$                                           ;
                         7269642E   8F            6E  D1 00055          CMPL    DN_EXT, #1919509550                          ;
                                                12  12 0005C          BNEQ    5$                                           ;
                                           04 A846  9F 0005E          PUSHAB  4(OFFSET)[BLOCK]                             ; 1545
                             08   AE            9E  3C 00062          MOVZWL  @(SP)+, SIZE                                 ;
                                                0D  12 00066          BNEQ    6$                                           ; 1546
                                 50 0001827A   8F  D0 00068 4$:      MOVL    #98938, R0                                   ;
                                                    04 0006F          RET                                                  ;
                             08   AE        36   AB  3C 00070 5$:      MOVZWL  54(FAB), SIZE                                ; 1549
                                 51        08   AE  D0 00075 6$:      MOVL    SIZE, R1                                     ;
                             1E   A9            51  B0 00079          MOVW    R1, 30(CBLOCK)                               ; 1542
                                 50        24   AA  D0 0007D          MOVL    36(RAB), R0                                  ; 1551
                             28   AA            50  D0 00081          MOVL    R0, 40(RAB)                                  ;
                                 53            50  D0 00085          MOVL    R0, PTR                                      ;
                             0C   AE        20   AA  B0 00088          MOVW    32(RAB), LENLIM                              ; 1552
                                           22   AA  B4 0008D          CLRW    34(RAB)                                      ; 1553
                             18   AE        08   AE  B0 00090          MOVW    SIZE, F_SIZE                                 ; 1554
                                         F71B   CF  16 00095 7$:      JSB     ULTRIX$ACCESS                                ; 1566
                                 01            50  E8 00099 8$:      BLBS    STAT, 9$                                     ; 1567
                                                    04 0009C          RET                                                  ;
                                           10   AE  94 0009D 9$:      CLRB    LF_FLAG                                      ; 1568
                                           14   AE  D4 000A0          CLRL    20(SP)                                       ; 1569
                                           30   AB  D5 000A3          TSTL    48(FAB)                                      ;
                                                5B  13 000A6          BEQL    14$                                          ;
                                           14   AE  D6 000A8          INCL    20(SP)                                       ;
                         706C682E   8F            6E  D1 000AB          CMPL    DN_EXT, #1886152750                          ;
```

```
                                                 09  13 000B2          BEQL    10$                                          ;
                    6D6F632E   8F                6E  D1 000B4          CMPL    DN_EXT, #1836016430                          ; 1570
                                                 46  12 000BB          BNEQ    14$                                          ;
                 52 00000200   8F                58  C3 000BD 10$:     SUBL3   OFFSET, #512, R2                             ; 1573
                 54            56                58  C1 000C5          ADDL3   OFFSET, BLOCK, R4                            ;
                 64            52                0A  3A 000C9          LOCC    #10, R2, (R4)                                ;
                                                 02  12 000CD          BNEQ    11$                                          ;
                                                 51  D4 000CF          CLRL    R1                                           ;
                         08   AE                51  D0 000D1 11$:     MOVL    R1, SIZE                                     ;
                                                 1E  12 000D5          BNEQ    13$                                          ;
                         44   A9                57  D1 000D7          CMPL    VBN, 68(CBLOCK)                              ; 1576
                                                 12  12 000DB          BNEQ    12$                                          ;
                              51          42    A9  3C 000DD          MOVZWL  66(CBLOCK), R1                               ; 1577
                              51                58  C0 000E1          ADDL2   OFFSET, R1                                   ;
           08   AE 00000200   8F                51  C3 000E4          SUBL3   R1, #512, SIZE                               ;
                                                 14  11 000ED          BRB     14$                                          ;
                         08   AE                52  D0 000EF 12$:     MOVL    R2, SIZE                                     ; 1578
                                                 0E  11 000F3          BRB     14$                                          ; 1576
                 51      08   AE                54  C3 000F5 13$:     SUBL3   R4, SIZE, R1                                 ; 1582
                         08   AE          01    A1  9E 000FA          MOVAB   1(R1), SIZE                                  ;
                         10   AE                01  90 000FF          MOVB    #1, LF_FLAG                                  ; 1583
                 50 00000200   8F                58  C3 00103 14$:     SUBL3   OFFSET, #512, R0                             ; 1586
                              51          08    AE  D0 0010B          MOVL    SIZE, R1                                     ;
                              50                51  D1 0010F          CMPL    R1, R0                                       ;
                                                 03  15 00112          BLEQ    15$                                          ;
                              51                50  D0 00114          MOVL    R0, R1                                       ;
     51      0C   AE          10                00  ED 00117 15$:     CMPZV   #0, #16, LENLIM, R1                          ;
                                                 04  18 0011D          BGEQ    16$                                          ;
                              51          0C    AE  3C 0011F          MOVZWL  LENLIM, R1                                   ;
                         04   AE                51  D0 00123 16$:     MOVL    R1, LENGTH                                   ;
                 63         6846          04    AE  28 00127          MOVC3   LENGTH, (OFFSET)[BLOCK], (PTR)               ; 1587
                         22   AA          04    AE  A0 0012D          ADDW2   LENGTH, 34(RAB)                              ; 1588
                         08   AE          04    AE  C2 00132          SUBL2   LENGTH, SIZE                                 ; 1589
                         1E   A9          22    AA  B0 00137          MOVW    34(RAB), 30(CBLOCK)                          ; 1590
                                          08    AE  D5 0013C          TSTL    SIZE                                         ; 1592
                                                 36  12 0013F          BNEQ    19$                                          ;
                                          10    AE  95 00141          TSTB    LF_FLAG                                      ; 1595
                                                 1B  13 00144          BEQL    18$                                          ;
                              4C          14    AE  E9 00146          BLBC    20(SP), 21$                                  ;
                    706C682E   8F                6E  D1 0014A          CMPL    DN_EXT, #1886152750                          ; 1596
                                                 09  13 00151          BEQL    17$                                          ;
                    6D6F632E   8F                6E  D1 00153          CMPL    DN_EXT, #1836016430                          ; 1597
                                                 05  12 0015A          BNEQ    18$                                          ;
                                          22    AA  B7 0015C 17$:     DECW    34(RAB)                                      ; 1601
                                                 35  11 0015F          BRB     21$                                          ; 1602
                              31          14    AE  E9 00161 18$:     BLBC    20(SP), 21$                                  ; 1605
                    706C682E   8F                6E  D1 00165          CMPL    DN_EXT, #1886152750                          ;
                                                 09  13 0016C          BEQL    19$                                          ;
                    6D6F632E   8F                6E  D1 0016E          CMPL    DN_EXT, #1836016430                          ; 1606
                                                 1F  12 00175          BNEQ    21$                                          ;
                         0C   AE          04    AE  A2 00177 19$:     SUBW2   LENGTH, LENLIM                               ; 1613
                                                 0D  12 0017C          BNEQ    20$                                          ; 1615
                         0C   AA          18    AE  3C 0017E          MOVZWL  F_SIZE, 12(RAB)                              ; 1618
                              50 000181A8        8F  D0 00183          MOVL    #98728, R0                                   ; 1619
                                                     04 0018A          RET                                                  ;
                              50          04    AE  D0 0018B 20$:     MOVL    LENGTH, R0                                   ; 1622
                                        F60C    CF  16 0018F          JSB     ULTRIX$ADVANCE                               ;
```

```
                                  FEFF  31 00193          BRW     7$                                  ;
                        50 00010001   8F  D0 00196 21$:    MOVL    #65537, R0                          ; 1626
                                          04 0019D          RET                                         ;

; Routine Size:  414 bytes,    Routine Base:  CODE + 0861


;   1627  1
```

```
;    1628  1
;    1629  1      %sbttl 'ultrix$getc'                                                    !G:3
;    1630  1      routine ultrix$getc   : jsb_ultrix  =
;    1631  1
;    1632  1      !++
;    1633  1      ! Functional description:
;    1634  1      ! Implicit inputs:
;    1635  1      !      R7                       vbn
;    1636  1      !
;    1637  1      ! Inputs:
;    1638  1      !
;    1639  1      ! iob[iob$l_i_ma]      - initial address of buffer to put data
;    1640  1      ! iob[iob$l_i_offset]  - initial offset into file in bytes
;    1641  1      ! iob[iob$l_i_cc]      - initial size of transfer in bytes
;    1642  1      !
;    1643  1      !
;    1644  1      ! implicit inputs:
;    1645  1      !
;    1646  1      ! Outputs:
;    1647  1      !
;    1648  1      ! Data is read into specified buffer.
;    1649  1      !
;    1650  1      ! Implicit outputs:
;    1651  1      !
;    1652  1      ! Side effects:
;    1653  1      !
;    1654  1      ! various fields in IOB are changed.
;    1655  1      !
;    1656  1      ! Return codes:
;    1657  1      !
;    1658  1      !      RMS$_RER         - File read error
;    1659  1      !      RMS$_NORMAL      - Success
;    1660  1      !      RMS$_EOF         - end of file encountered
;    1661  1      ! --
;    1662  2      BEGIN
;    1663  2
;    1664  2           external register
;    1665  2              block = 6,
;    1666  2              vbn = 7,
;    1667  2              offset = 8,
;    1668  2              cblock = 9 : ref bblock,
;    1669  2              rab = 10 : ref bblock,
;    1670  2              fab = 11: ref bblock;
;    1671  2           local
;    1672  2              write_buf_addr:long;    ! Current address of write buffer.                    [132]
;    1673  2
;    1674  2              IF     (.iob[iob$l_i_flgs]  AND  F_ALLOC) EQL 0
;    1675  2              THEN
;    1676  2                     return (RMS$_EOF);                                  ! error: bad file descriptor
;    1677  2              transfer_size = .iob[iob$l_i_cc];                           ! Get transfer size of this operation
;    1678  2                     write_buf_addr = .iob[iob$l_i_ma];      ! Initialize the buffer address  [02]
;    1679  2              WHILE .transfer_size GTR 0 DO
;    1680  3                     BEGIN
;    1681  3                     IF (.iob[iob$l_i_flgs]  AND   F_FILE) NEQ 0
;    1682  3                     THEN
;    1683  4                            BEGIN
;    1684  4                            diff = .iob[iob$z_ic_size] - .iob[iob$l_i_offset];          ! check if byte offset excee
```

```
;   1685  4                         IF .diff  LEQ  0
;   1686  4                           THEN
;   1687  4                               return (RMS$_EOF);
;   1688  4                       lbn =  (.iob[iob$l_i_offset] ^ -.iob[iob$l_fs_bshift]);          ! compute logical fileblock
;   1689  4                       iob[iob$l_i_bn] =  (ultrix$sbmap(.lbn) ^ .iob[iob$l_fs_fsbtodb]) + .iob[iob$l_i_boff]; !find
;   1690  4                                                   ! fileblock of file  from file system blk
;   1691  4                       off = .iob[iob$l_i_offset] AND ( NOT .iob[iob$l_fs_bmask]);      ! offset into block
;   1692  4                       iob[iob$l_i_bn] = .iob[iob$l_i_bn] + .off/512;           ! Compute logical block number to re
;   1693  4
;   1694  5                         IF .lbn GEQ NDADDR  OR  .iob[iob$z_val] GEQ  ((.lbn + 1) ^ .iob[iob$l_fs_bshift])
;   1695  4                           THEN
;   1696  4                               frag_size = .iob[iob$l_fs_bsize]        ! find size of fragment in bytes.
;   1697  4                           ELSE
;   1698  4                               frag_size = ((.iob[iob$z_val] AND ( NOT .iob[iob$l_fs_bmask])) + .iob[iob$l_fs_fsize
;   1699  4                                                                                  .iob[iob$l_fs_fmask];
;   1700  4
;   1701  4                       END
;   1702  3                 ELSE
;   1703  4                       BEGIN
;   1704  4                       iob[iob$l_i_bn] = .iob[iob$l_i_offset] / DEV_BSIZE;      ! next blk # to read
;   1705  4                       off = 0;
;   1706  4                       frag_size = DEV_BSIZE;
;   1707  3                       END;
;   1708  3                 iob[iob$l_i_cc] = MIN(.transfer_size, .frag_size);               ! set size of fragment of transfer
;   1709  3                 transfer_size = MAX(0, .transfer_size - .frag_size);             ! Reset remainder to transfer.
;   1710  3                 iob[iob$l_i_ma] = .write_buf_addr;                               ! restore correct buffer address
;   1711  3                 IF NOT ultrix$devread()
;   1712  3
;   1713  3                 THEN
;   1714  3                       return (RMS$_RER);
;   1715  3                 iob[iob$l_i_ma] = .iob[iob$l_i_ma] + .iob[iob$l_i_cc] ;          ! Reset buffer addr to transfer to.
;   1716  3                 write_buf_addr = .iob[iob$l_i_ma];       ! Save the buffer address  [02]
;   1717  3                 iob[iob$l_i_offset] = .iob[iob$l_i_offset] + .iob[iob$l_i_cc];  ! Adjust for next interation.
;   1718  2                 END;
;   1719  2           return (RMS$_NORMAL);                                   ! return pointer
;   1720  2
;   1721  1     END;



                                        3C  BB 00000 ULTRIX$GETC:
                                                                PUSHR   #^M<R2,R3,R4,R5>                        ; 1630
                          50 00000000'  EF  D0 00002            MOVL    IOB, R0                                 ; 1674
                37        60            02  E1 00009            BBC     #2, (R0), 4$                            ;
                   00000000'  EF    00C6  C0  D0 0000D            MOVL    198(R0), TRANSFER_SIZE                  ; 1677
                          54    00C2  C0  D0 00016            MOVL    194(R0), WRITE_BUF_ADDR                 ; 1678
                          52 00000000'  EF  D0 0001B            MOVL    IOB, R2                                 ; 1681
                          53    00BA  C2  9E 00022            MOVAB   186(R2), R3                             ; 1684
                             00000000'  EF  D5 00027 1$:        TSTL    TRANSFER_SIZE                           ; 1679
                                        03  14 0002D            BGTR    2$                                      ;
                                      0133  31 0002F            BRW     13$                                     ;
                03        62            03  E0 00032 2$:        BBS     #3, (R2), 3$                            ; 1681
                                      00A4  31 00036            BRW     8$                                      ;
        00000000'  EF      36  A2       63  C3 00039 3$:        SUBL3   (R3), 54(R2), DIFF                      ; 1684
                                        0A  14 00042            BGTR    5$                                      ; 1685
```

```
                                50 0001827A    8F   D0 00044 4$:     MOVL    #98938, R0                              ; 1687
                                          011E  31   0004B           BRW     14$                                     ;
                                50      2126   C2   CE 0004E 5$:     MNEGL   8486(R2), R0                            ; 1688
          00000000'  EF         63             50   78 00053         ASHL    R0, (R3), LBN                           ;
                                50 00000000'   EF   D0 0005B         MOVL    LBN, R0                                 ; 1689
                                     FC00      CF   16 00062         JSB     ULTRIX$SBMAP                            ;
                                51 00000000'   EF   D0 00066         MOVL    IOB, R1                                 ;
                   50           50      213A   C1   78 0006D         ASHL    8506(R1), R0, R0                        ;
                          00BE   C2      00B2 D140   9E 00073         MOVAB   @178(R1)[R0], 190(R2)                   ;
          00000000'  EF    00BA   C1      211E   C1   CB 0007B         BICL3   8478(R1), 186(R1), OFF                  ; 1691
                50 00000000'   EF 00000200    8F   C7 00087         DIVL3   #512, OFF, R0                           ; 1692
                          00BE   C1             50   C0 00093         ADDL2   R0, 190(R1)                             ;
                                 0C 00000000'   EF   D1 00098         CMPL    LBN, #12                                ; 1694
                                                14   18 0009F         BGEQ    6$                                      ;
                50 00000000'   EF             01   C1 000A1         ADDL3   #1, LBN, R0                             ;
                50             50      2126   C1   78 000A9         ASHL    8486(R1), R0, R0                        ;
                               50        36   A1   D1 000AF         CMPL    54(R1), R0                              ;
                                              0B   19 000B3         BLSS    7$                                      ;
                   00000000'   EF      2106   C1   D0 000B5 6$:     MOVL    8454(R1), FRAG_SIZE                     ; 1696
                                              36   11 000BE         BRB     9$                                      ;
                50        36   A1      211E   C1   CB 000C0 7$:     BICL3   8478(R1), 54(R1), R0                    ; 1698
                               50      210A   C1   C0 000C7         ADDL2   8458(R1), R0                            ;
                                              50   D7 000CC         DECL    R0                                      ;
                               55      2122   C1   D2 000CE         MCOML   8482(R1), R5                            ; 1699
          00000000'  EF        50             55   CB 000D3         BICL3   R5, R0, FRAG_SIZE                       ;
                                              19   11 000DB         BRB     9$                                      ; 1694
              00BE   C2         63 00000200    8F   C7 000DD 8$:     DIVL3   #512, (R3), 190(R2)                     ; 1704
                                 00000000'   EF   D4 000E7         CLRL    OFF                                     ; 1705
                   00000000'   EF      0200   8F   3C 000ED         MOVZWL  #512, FRAG_SIZE                         ; 1706
                               51 000000C0'   EF   D0 000F6 9$:     MOVL    IOB, R1                                 ; 1708
                               50 00000000'   EF   D0 000FD         MOVL    TRANSFER_SIZE, R0                       ;
                   00000000'   EF             50   D1 00104         CMPL    R0, FRAG_SIZE                           ;
                                              07   15 0010B         BLEQ    10$                                     ;
                               50 00000000'   EF   D0 0010D         MOVL    FRAG_SIZE, R0                           ;
                      00C6     C1             50   D0 00114 10$:     MOVL    R0, 198(R1)                             ;
                50 00000000'   EF 00000000'   EF   C3 00119         SUBL3   FRAG_SIZE, TRANSFER_SIZE, R0            ; 1709
                                              02   18 00125         BGEQ    11$                                     ;
                                              50   D4 00127         CLRL    R0                                      ;
                   00000000'   EF             50   D0 00129 11$:     MOVL    R0, TRANSFER_SIZE                       ;
                      00C2     C1             54   D0 00130         MOVL    WRITE_BUF_ADDR, 194(R1)                 ; 1710
                                     FC99      CF   16 00135         JSB     ULTRIX$DEVREAD                          ; 1711
                               09             50   E8 00139         BLBS    R0, 12$                                 ;
                               50 0001C0F4    8F   D0 0013C         MOVL    #114932, R0                             ; 1714
                                              27   11 00143         BRB     14$                                     ;
                               52 00000000'   EF   D0 00145 12$:     MOVL    IOB, R2                                 ; 1715
                      00C2     C2      00C6   C2   C0 0014C         ADDL2   198(R2), 194(R2)                        ;
                               54      00C2   C2   D0 00153         MOVL    194(R2), WRITE_BUF_ADDR                 ; 1716
                               53      00BA   C2   9E 00158         MOVAB   186(R2), R3                             ; 1717
                               63      00C6   C2   C0 0015D         ADDL2   198(R2), (R3)                           ;
                                            FEC2   31 00162         BRW     1$                                      ;
                               50 00010001    8F   D0 00165 13$:     MOVL    #65537, R0                              ; 1719
                                              3C   BA 0016C 14$:     POPR    #^M<R2,R3,R4,R5>                        ; 1721
                                                   05 0016E         RSB                                             ;

; Routine Size:  367 bytes,    Routine Base:  CODE + 09FF
```

```
;   1722  1
```

```
1     %sbttl 'ULTRIX$READ routine'
1
1     global routine ultrix$read : call_rms =
1
1     !++
1     ! Functional description:
1     !       Read virtual blocks of file
1     !
1     ! Inputs:
1     !       none
1     !
1     ! Implicit inputs:
1     !       R9                      CBLOCK pointer
1     !       R10                     RAB pointer
1     !       various fields in the above-listed data structures
1     !
1     ! Outputs:
1     !       none
1     !
1     ! Implicit outputs:
1     !       RAB [rab$l_rbf]         points to buffer
1     !       RAB [rab$w_rsz]         size of buffer
1     !
1     ! Side Effects:
1     !       CBLOCK [ctl$l_nbp]      points to next VBN for sequential READ.
1     !
1     ! Return codes:
1     !       RMS$_NORMAL             OK
1     !       RMS$_RER                Read error
1     !--
1
2         begin
2
2         external register
2             cblock = 9 : ref bblock,                ! Map the control block
2             rab = 10 : ref bblock,                  ! Map the RAB
2             fab = 11 : ref bblock;                  ! Map the FAB
2
2                 global register
2                     block = 6,                      ! Registers for ACCESS
2                     vbn = 7,
2                     offset = 8;
2
2
3     rab[rab$w_rsz]   = MIN(.rab[rab$w_usz], (.cblock[ctl$l_eofvbn] -
2                            .cblock[ctl$l_nbp])*512 + .cblock[ctl$w_offbyte]);
2     cblock [ctl$l_nbp] =.cblock [ctl$l_nbp] +(.rab[rab$w_rsz] + 511)/512 ;
2     iob[iob$l_i_ma]  = .rab[rab$l_ubf];             ! Buffer to read to
2     iob[iob$l_i_cc]  = .rab[rab$w_rsz]; ! Don't copy too much.
2     iob[iob$l_i_offset] = MAX((.rab[rab$l_bkt]-1)*512,0);   ! Start at correct offset.
2                                                     ! No negative value.
2
3     BEGIN
3             local
3                stat;
3
4      IF NOT (stat= ultrix$getc())
```

```
;   1781  3         THEN
;   1782  3             (rab[rab$l_stv] = .stat; return rms$_rer);
;   1783  2     END;
;   1784  2         rms$_normal
;   1785  1         end;                                        ! Of ultrix$read



                                                  01FC 00000          .ENTRY  ULTRIX$READ, Save R2,R3,R4,R5,R6,R7,R8      ; 1726
                  51        44   A9        04   A9  C3 00002          SUBL3   4(CBLOCK), 68(CBLOCK), R1                   ; 1769
                  51             51             09  78 00008          ASHL    #9, R1, R1                                  ;
                                 50        42   A9  3C 0000C          MOVZWL  66(CBLOCK), R0                              ;
                                 51             50  C0 00010          ADDL2   R0, R1                                      ;
                                 50        20   AA  3C 00013          MOVZWL  32(RAB), R0                                 ;
                                 51             50  D1 00017          CMPL    R0, R1                                      ;
                                                03  15 0001A          BLEQ    1$                                          ;
                                 50             51  D0 0001C          MOVL    R1, R0                                      ;
                            22   AA             50  B0 0001F 1$:      MOVW    R0, 34(RAB)                                 ; 1768
                                 50        22   AA  3C 00023          MOVZWL  34(RAB), R0                                 ; 1770
                                 50      01FF   C0  9E 00027          MOVAB   511(R0), R0                                 ;
                                 50 00000200    8F  C6 0002C          DIVL2   #512, R0                                    ;
                            04   A9             50  C0 00033          ADDL2   R0, 4(CBLOCK)                               ;
                                 51 00000000'   EF  D0 00037          MOVL    IOB, R1                                     ; 1771
                          00C2   C1        24   AA  D0 0003E          MOVL    36(RAB), 194(R1)                            ;
                          00C6   C1        22   AA  3C 00044          MOVZWL  34(RAB), 198(R1)                            ; 1772
                  50        38   AA             01  C3 0004A          SUBL3   #1, 56(RAB), R0                             ; 1773
                  50             50             09  78 0004F          ASHL    #9, R0, R0                                  ;
                                                02  18 00053          BGEQ    2$                                          ;
                                                50  D4 00055          CLRL    R0                                          ;
                          00BA   C1             50  D0 00057 2$:      MOVL    R0, 186(R1)                                 ;
                                        FE31   CF  16 0005C          JSB     ULTRIX$GETC                                 ; 1780
                                 0C             50  E8 00060          BLBS    STAT, 3$                                    ;
                            0C   AA             50  D0 00063          MOVL    STAT, 12(RAB)                               ; 1782
                                 50 0001C0F4    8F  D0 00067          MOVL    #114932, R0                                 ;
                                                    04 0006E          RET                                                 ;
                                 50 00010001    8F  D0 0006F 3$:      MOVL    #65537, R0                                  ; 1785
                                                    04 00076          RET                                                 ;

; Routine Size:  119 bytes,    Routine Base:  CODE + 0B6E


;   1786  1
;   1787  1     end
;   1788  1
;   1789  0     eludom




;                                 PSECT SUMMARY
;
;       Name                  Bytes                        Attributes
;
```

```
;  WORK                          156  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                         3045  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                       0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)
```

```
Symbol                 Type  Defined  Referenced ...
------------------ -------- ----- ----------------
ADRS_MODE            Own        154  1438=   1441.
B                    Own        465   480a   1301.   1311a.
BBLOCK               Structur Lib02   566     567     568     680     681     682     882     883     884    1025    1026    1027
                                     1097    1098    1099    1179    1180    1181    1239    1240    1241    1356    1357    1358
                                     1428    1429    1430    1498    1499    1500    1668    1669    1670    1758    1759    1760
BBSIZE               Literal    261   264
BLKNOS               Own        160   815=   1297.   1307=
BLOCK                ExtReg     563   633=
                     GlobReg    684
                     ExtReg     879
                     ExtReg    1022
                     ExtReg    1094
                     ExtReg    1176
                     ExtReg    1236
                     ExtReg    1353
                     ExtReg    1425
                     GlobReg   1503  1545.   1573a.  1582aa  1587a.
                     ExtReg    1665
                     GlobReg   1763
BN                   Own        158
                     RoutForm  1206  1244.   1256.   1259.   1268=   1268.   1273.   1276=   1276.   1310.
BOFF                 Own        150
                     RoutForm  1323  1381.
BOO$QIO              ExtRout    114  1441c
BUFF_ADDR            Own        156  1437=   1441.
BUF_START            Literal    245
BYTE_COUN            Own        153  1436=   1441.
B_TAPE               Literal    260   732
CACHE                Bind       574   615.    633.
CALL_RMS             Linkage  Lib02    91      92     101     639    1456    1726
CBLOCK               ExtReg     566   574aa   576a.   580a.   586a.   587a.   591a.   605a=   606a=   608a.   629a=   630a=   633a.
                     ExtReg     680   690a=   691a=   692a=   701a.   702a.   812a=   814a=
                     ExtReg     882
                     ExtReg    1025
                     ExtReg    1097
                     ExtReg    1179
                     ExtReg    1239
                     ExtReg    1356
                     ExtReg    1428  1441a.
                     ExtReg    1498  1515a.  1516a.  1524a.  1526a=  1527a=  1529a.  1530a.  1541a=  1576a.  1577a.  1590a=
                     ExtReg    1668
                     ExtReg    1758  1768a.  1769a.  1770a=  1770a.
CC                   Own        133  1195=   1201.   1444=
CHAIN_BASE           Literal    240   241     242     298     299     300     301     302     303     304     305     306     307
                                      308
CTL$L_CACHE1         Macro    Lib02   574
CTL$L_EOFVBN         Macro    Lib02   576     812    1529    1576    1768
```

Symbol	Type	Defined	Referenced ...							
CTL$L_FILE_PTR	Macro	Lib02	701	702						
CTL$L_GET	Macro	Lib02	690							
CTL$L_HVBN	Macro	Lib02	587	606	630					
CTL$L_LVBN	Macro	Lib02	586	605	629	633				
CTL$L_NBP	Macro	Lib02	1769	1770						
CTL$L_READ	Macro	Lib02	691							
CTL$L_RPB	Macro	Lib02	1441							
CTL$L_STARTLBN	Macro	Lib02	692							
CTL$L_VBN	Macro	Lib02	1515	1526						
CTL$W_BYTE	Macro	Lib02	1516	1527						
CTL$W_CACHE_SIZE	Macro	Lib02	591							
CTL$W_FILE_LEN	Macro	Lib02	701	702						
CTL$W_OFFBYTE	Macro	Lib02	580	608	814	1530	1577	1769		
CTL$W_REC_SIZE	Macro	Lib02	1524	1541	1590					
D	Own	141	1111=	1112.	1115.					
DEVSW	Own	475	732.							
DEVSW$L_DV_CLOSE	Field	448								
DEVSW$L_DV_FLAGS	Field	450	732.							
DEVSW$L_DV_IOCTL	Field	449								
DEVSW$L_DV_NAME	Field	445								
DEVSW$L_DV_OPEN	Field	447								
DEVSW$L_DV_STRATEGY	Field	446								
DEVSW_FIELDS	Fieldset	444	475							
DEV_BSIZE	Literal	262	264	1704	1706					
DIFF	Own	148	1684=	1685.						
DINODE_SIZE	Literal	233	1199							
DIRECT$L_D_INO	Field	429								
DIRECT$W_D_NAMLEN	Field	431								
DIRECT$W_D_RECLEN	Field	430								
DIRECT$Z_D_NAME	Field	432								
DIRECT_FIELDS	Fieldset	428								
DIRSTUFF	Own	470	1047=	1103.	1106.	1110.	1133=	1133.		
DIRSTUFF$L_IO	Field	438								
DIRSTUFF$L_LOC	Field	437	1047=	1103.	1106.	1110.	1133=	1133.		
DIRSTUFF_FIELDS	Fieldset	436	470							
DN_EXT	Local	876	892=	921.	952.					
	Local	1508	1520=	1542.	1569.	1570.	1596.	1597.	1605.	1606.
DP	Own	161	1049=	1051a.	1055.	1059a.	1132=	1133.	1134a.	1136. 1138.
DS$_ERROR	Literal	Lib01	767	1388						
DS$_NORMAL	Literal	Lib01	1382							
DS$_NOSUPPORT	Literal	Lib01	761	1385						
END_SIZE	External	110	932=	957=	958.					
EXE$ALONONPAGED	ExtRout	115								
FAB	ExtReg	568								
	ExtReg	682	694a=	695a=	696a=					
	ExtReg	884	890a.	892a.	921a.	932a.	952a.	957a.	958a.	
	ExtReg	1027								
	ExtReg	1099								
	ExtReg	1181								
	ExtReg	1241								
	ExtReg	1358								
	ExtReg	1430								
	ExtReg	1500	1518a.	1520a.	1542a.	1549a.	1569a.	1595a.	1605a.	
	ExtReg	1670								
	ExtReg	1760								

Symbol	Type	Defined	Referenced ...
FAB$B_DNS	Macro	Lib03	892 1520
FAB$B_FNS	Macro	Lib03	932 957
FAB$B_ORG	Macro	Lib03	694
FAB$B_RFM	Macro	Lib03	695
FAB$C_FIX	Literal	Lib03	695
FAB$C_SEQ	Literal	Lib03	694
FAB$L_DNA	Macro	Lib03	890 892 921 952 1518 1520 1542 1569 1595 1605
FAB$L_FNA	Macro	Lib03	958
FAB$W_MRS	Macro	Lib03	696 1549
FRAG_SIZE	Own	145	1696= 1698= 1706= 1708. 1709.
FS_BASE	Literal	243	244 287 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 402 403 404 405 406 407 408 409 410 411 412 413 417 418 421 422 1381
FS_MAGIC	Literal	257	1364
F_ALLOC	Literal	278	699 1674
F_FILE	Literal	263	810 1681
F_RDDATA	Literal	275	1433
F_SIZE	Local	1510	1554= 1618.
F_TYPEMASK	Literal	277	1445
I	Own	136	798= 804. 1310= 1311.
	Local	609	615. 616.
IFDIR	Literal	282	929 953 1033
IFMT	Literal	281	929 953 1033
IOB	Own	464	479a 615a= 616a= 617a= 699a= 699a. 700a= 701a= 702a. 703a. 707a. 709a= 709a. 712a. 714a= 714a. 717a. 719a= 719a. 727a= 727a. 741a. 742a= 747a= 747aa 748a= 749a= 750a= 781a= 781a. 782a= 783a= 786a= 786aa 787a= 788a= 788a. 789a= 793a= 801a= 807a= 810a= 810a. 813a. 814a. 929a. 953a. 1033a. 1037a. 1103a. 1106a. 1110a. 1115a= 1115a. 1116a. 1117a= 1117aa 1120a. 1122a= 1122a. 1124a= 1124a. 1125a. 1132. 1184a= 1187a= 1188a. 1189a. 1190a. 1191a. 1192a= 1192a. 1193a= 1193aa 1195a. 1198a= 1198a. 1199a. 1259a. 1272a. 1287a. 1300a= 1300a. 1301a= 1302a= 1302a. 1309a. 1310a. 1364a. 1373a. 1381a= 1381a. 1433a= 1433a. 1434a= 1435a. 1436a. 1437a. 1444a. 1445a= 1445a. 1674a. 1677a. 1678a. 1681a. 1684a. 1688a. 1689a= 1689a. 1691a. 1692a= 1692a. 1694a. 1696a. 1698a. 1699a. 1704a= 1704a. 1708a= 1710a= 1715a= 1715a. 1716a. 1717a= 1717a. 1771a= 1772a= 1773a=
IOB$B_FS_CLEAN	Field	403	
IOB$B_FS_FLAGS	Field	405	
IOB$B_FS_FMOD	Field	402	
IOB$B_FS_RONLY	Field	404	
IOB$B_FS_ROTBL	Field	412	
IOB$L_CS_NBFREE	Field	399	
IOB$L_CS_NDIR	Field	398	
IOB$L_CS_NFFREE	Field	401	
IOB$L_CS_NIFREE	Field	400	
IOB$L_FS_BMASK	Field	372	1106. 1124. 1691. 1698.
IOB$L_FS_BSHIFT	Field	374	1110. 1120. 1688. 1694.
IOB$L_FS_BSIZE	Field	366	1122. 1192. 1302. 1696.
IOB$L_FS_CBLKNO	Field	357	
IOB$L_FS_CGMASK	Field	361	1189.
IOB$L_FS_CGOFFSET	Field	360	1188.
IOB$L_FS_CGROTOR	Field	407	
IOB$L_FS_CGSIZE	Field	389	
IOB$L_FS_CPC	Field	409	

Symbol	Type	Defined	Referenced ...
IOB$L_FS_CPG	Field	394	
IOB$L_FS_CSADDR	Field	387	
IOB$L_FS_CSMASK	Field	381	
IOB$L_FS_CSSHIFT	Field	382	
IOB$L_FS_CSSIZE	Field	388	
IOB$L_FS_DBLKNO	Field	359	
IOB$L_FS_DSIZE	Field	364	
IOB$L_FS_FMASK	Field	373	1125. 1699.
IOB$L_FS_FPG	Field	396	1188.
IOB$L_FS_FRAG	Field	368	
IOB$L_FS_FRAGSHIFT	Field	378	1190.
IOB$L_FS_FSBTODB	Field	379	1115. 1191. 1300. 1689.
IOB$L_FS_FSHIFT	Field	375	
IOB$L_FS_FSIZE	Field	367	1124. 1698.
IOB$L_FS_IBLKNO	Field	358	1189.
IOB$L_FS_INOPB	Field	384	1190. 1199.
IOB$L_FS_IPG	Field	395	1188. 1189. 1190.
IOB$L_FS_LINK	Field	354	
IOB$L_FS_MAGIC	Field	411	1364.
IOB$L_FS_MAXBPG	Field	377	
IOB$L_FS_MAXCONTIG	Field	376	
IOB$L_FS_MINFREE	Field	369	
IOB$L_FS_NCG	Field	365	
IOB$L_FS_NCYL	Field	393	
IOB$L_FS_NINDIR	Field	383	1272. 1309. 1310.
IOB$L_FS_NSECT	Field	391	
IOB$L_FS_NSPF	Field	385	
IOB$L_FS_NTRAK	Field	390	
IOB$L_FS_RLINK	Field	355	
IOB$L_FS_ROTDELAY	Field	370	
IOB$L_FS_RPSS	Field	371	
IOB$L_FS_SBLKNO	Field	356	
IOB$L_FS_SBSIZE	Field	380	
IOB$L_FS_SIZE	Field	363	
IOB$L_FS_SPC	Field	392	
IOB$L_FS_TIME	Field	362	
IOB$L_IC_ATIME	Field	324	
IOB$L_IC_ATSPARE	Field	325	
IOB$L_IC_BLOCKS	Field	333	
IOB$L_IC_CTIME	Field	328	
IOB$L_IC_CTSPARE	Field	329	
IOB$L_IC_FLAGS	Field	332	
IOB$L_IC_MTIME	Field	326	
IOB$L_IC_MTSPARE	Field	327	
IOB$L_IF_FREEB	Field	313	
IOB$L_IF_FREEF	Field	312	
IOB$L_IF_LASTR	Field	309	
IOB$L_IS_SOCKET	Field	310	
IOB$L_I_BN	Field	341	749= 788= 1115= 1187= 1300= 1435. 1689= 1692= 1692. 1704=
IOB$L_I_BOFF	Field	338	701= 702. 703. 707a. 709a. 712a. 714a. 717a. 719a. 727= 741. 742= 788. 1116. 1191. 1300. 1381= 1689.
IOB$L_I_CC	Field	343	617= 748= 782= 787= 1122= 1124= 1192= 1195. 1302= 1436. 1444. 1677. 1708= 1715. 1717. 1772=
IOB$L_I_CYLOFF	Field	339	
IOB$L_I_DQUOT	Field	307	

Symbol	Type	Defined	Referenced ...
IOB$L_I_ERRBLK	Field	346	
IOB$L_I_ERRCNT	Field	345	
IOB$L_I_ERROR	Field	344	1434=
IOB$L_I_FLGS	Field	296	699= 699. 781= 781. 793= 801= 807= 810= 810. 1433= 1433. 1445= 1445. 1674. 1681.
IOB$L_I_FS	Field	306	
IOB$L_I_ID	Field	305	
IOB$L_I_INO	Field	297	
IOB$L_I_MA	Field	342	615= 747= 786= 1117= 1193= 1198. 1301= 1437. 1678. 1710= 1715= 1715. 1716. 1771=
IOB$L_I_NUMBER	Field	304	
IOB$L_I_OFFSET	Field	340	616= 750= 783= 789= 1184= 1684. 1688. 1691. 1704. 1717= 1717. 1773=
IOB$L_I_UNIT	Field	337	700= 709= 714= 719= 727.
IOB$L_PI_BLKOFF	Field	422	
IOB$L_PI_NBLOCKS	Field	421	
IOB$L_PT_MAGIC	Field	413	1373.
IOB$L_PT_PART	Field	418	
IOB$L_PT_VALID	Field	417	
IOB$W_IC_GID	Field	321	
IOB$W_IC_MODE	Field	318	929. 953. 1033.
IOB$W_IC_NLINK	Field	319	
IOB$W_IC_UID	Field	320	
IOB$W_I_COUNT	Field	300	
IOB$W_I_DEV	Field	301	
IOB$W_I_EXLOCKC	Field	303	
IOB$W_I_FLAG	Field	299	
IOB$W_I_SHLOCKC	Field	302	
IOB$Z_DUMMY	Field	352	
IOB$Z_FS_CSP	Field	408	
IOB$Z_FS_CSTOTAL	Field	397	
IOB$Z_FS_FSMNT	Field	406	
IOB$Z_FS_POSTBL	Field	410	
IOB$Z_FS_SPARECON	Field	386	
IOB$Z_IC_DB	Field	330	
IOB$Z_IC_IB	Field	331	
IOB$Z_IC_SIZE	Field	322	813. 814. 1684.
IOB$Z_IC_SPARE	Field	334	
IOB$Z_I_BUF	Field	347	1117a 1132 1193a
IOB$Z_I_CHAIN	Field	298	
IOB$Z_I_FR	Field	311	
IOB$Z_I_IC	Field	314	
IOB$Z_I_UN	Field	348	
IOB$Z_I_UNX	Field	308	
IOB$Z_UI_FS	Field	353	747a 786a
IOB$Z_VAL	Field	323	1037. 1103. 1120. 1124. 1694. 1698.
IOB_FIELDS	Fieldset	295	464
IO_FUNCTN	Own	155	1439= 1441.
J	Own	137	1269= 1270. 1277= 1277. 1279. 1287. 1295. 1316= 1316.
JSB_ALLOC	Linkage	Lib02	115
JSB_CBLOCK	Linkage	Lib02	113
JSB_ULTRIX	Linkage	82	89 90 93 94 95 96 97 98 99 100 492 529 824 972 1065 1143 1206 1323 1394 1630
LASTR_BASE	Literal	241	309 310 311 312 313
LBN	Own	142	1110= 1111. 1120. 1688= 1689. 1694.
LEN	Own	140	1041= 1042. 1044= 1046= 1046. 1055.

Symbol	Type	Defined	Referenced ...									
	RoutForm	492	524.	525.								
LENGTH	Local	1560	1586=	1587.	1588.	1589.	1613.	1622.				
LENLIM	Local	1512	1552=	1586.	1613=	1613.	1615.					
LF_FLAG	Local	1509	1568=	1583=	1595.							
LOOP	Local	1197	1198.	1200.								
MAXBSIZE	Literal	194	243	288	348							
MAXCPG	Literal	220	411	412								
MAXCSBUFS	Literal	231	409	410	411	412						
MAXMNTLEN	Literal	230	407	408	409	410	411	412				
MAXNAMLEN	Literal	249										
MODE_BASE	Literal	242	318	319	320	321	322	323	324	325	326	327	328	329
			330	331	332	333	334	1198	1259	1287				
N	Own	135	914=	924.	935.	940.	944.	968.						
	RoutForm	1143	1188.	1189.	1190.	1199.								
NB	Own	159	1259=	1260.	1287=	1288.	1297.	1300.	1307.	1311=	1312.	1318.		
NBUFS	Literal	248	288											
NDADDR	Literal	205	331	332	1120	1256	1268	1287	1694					
NDIR_BASE	Literal	244	398	399	400	401								
NFILES	Literal	266												
NIADDR	Literal	206	332	1269	1287	1295								
NRPOS	Literal	176	411	412										
NULL	Literal	283	1030	1049	1105	1114	1130							
OFF	Own	143	1106=	1107.	1132.	1691=	1692.	1705=						
OFFSET	ExtReg	522	524.	525=	525.									
	ExtReg	565												
	GlobReg	686												
	ExtReg	881												
	ExtReg	1024												
	ExtReg	1096												
	ExtReg	1178												
	ExtReg	1238												
	ExtReg	1355												
	ExtReg	1427												
	GlobReg	1505	1516=	1527.	1529.	1545.	1573.	1577.	1578.	1582.	1586.	1587.		
	ExtReg	1667												
	GlobReg	1765												
PHYSMODE	Literal	270	1438											
PREV_N	Local	877	933.	944=	961.									
PT$L_BLKOFF	Field	460												
PT$L_MAGIC	Field	456												
PT$L_NBLOCKS	Field	459												
PT$L_PART	Field	458												
PT$L_VALID	Field	457												
PTR	Local	1513	1551=	1587=	1587a=									
PT_FIELDS	Fieldset	455												
PT_MAGIC	Literal	259	1373											
PT_SIZE	Literal	234	414	417	418	421	422	1381						
Q	Own	157	901=	903a.	906=	906.	943.							
QIO_STATUS	Own	151	1441=	1446.										
RAB	ExtReg	567	584a.	623a=										
	ExtReg	681												
	ExtReg	883												
	ExtReg	1026												
	ExtReg	1098												
	ExtReg	1180												

Symbol	Type	Defined	Referenced ...
	ExtReg	1240	
	ExtReg	1357	
	ExtReg	1429	
	ExtReg	1499	1522a. 1551a= 1551a. 1552a. 1553a= 1588a= 1588a. 1590a. 1601a= 1601a. 1618a=
	ExtReg	1669	
	ExtReg	1759	1768a= 1768a. 1770a. 1771a. 1772a. 1773a. 1782a=
RAB$B_RAC	Macro	Lib03 584	1522
RAB$C_RFA	Literal	Lib03 584	1522
RAB$L_BKT	Macro	Lib03 1773	
RAB$L_RBF	Macro	Lib03 1551	
RAB$L_STV	Macro	Lib03 623	1618 1782
RAB$L_UBF	Macro	Lib03 1551	1771
RAB$W_RSZ	Macro	Lib03 1553	1588 1590 1601 1768 1770 1772
RAB$W_USZ	Macro	Lib03 1552	1768
READ	Literal	272	
READVBLK	Literal	269	1439
READ_BLOCKS	Local	571	576= 578. 579. 607= 608. 609. 630.
RLEN	Own	139	893= 894. 902= 911. 912= 917. 948.
RMS$ALLOC_CACHE	ExtRout	113	598c
RMS$_DME	Literal	Lib03 602	
RMS$_EOF	Literal	Lib03 584	1532 1546 1676 1687
RMS$_FNF	Literal	Lib03 802	808
RMS$_FNM	Literal	Lib03 784	
RMS$_NORMAL	Literal	Lib03 816	1450 1602 1609 1625 1719 1784
RMS$_RER	Literal	Lib03 624	721 768 795 1448 1714 1782
RMS$_RFA	Literal	Lib03 584	
RMS$_RTB	Literal	Lib03 1619	
ROOTINO	Literal	246	887
ROT	Builtin	76	
SBLOCK	Literal	264	749 788
SBSIZE	Literal	232	287 413 417 418 421 422 748 787 1381
SH	Own	138	1267= 1272= 1272. 1273. 1276. 1309= 1309. 1310.
SIZE	Local	1511	1545= 1546. 1547. 1549= 1554. 1573= 1576= 1582= 1582. 1586. 1589= 1589. 1592.
START_LBN	Own	152	1435= 1441.
STAT	Local	596	598= 600.
	Local	613	618= 620. 623.
	Local	677	
	Local	1537	1539= 1540.
	Local	1564	1566= 1567.
	Local	1778	1780= 1782.
STATE	Own	144	761= 767.
STR_PATH	Own	147	703= 725= 725. 896a. 899= 899. 901. 943= 1041a. 1046. 1055a.
STR_SIZE	Own	146	702= 726= 726. 778. 893. 898= 898. 902. 903. 905= 905. 908. 911= 932. 957. 1030. 1041. 1044.
TMP_BOFF	Own	149	741= 761.
TRANSFER_SIZE	Own	134	1677= 1679. 1708. 1709= 1709.
ULTRIX$ACCESS	Routine	529	90f 1539c 1566c
ULTRIX$ADVANCE	Routine	492	89f 1524c 1622c
ULTRIX$B	GlobLit	288	
ULTRIX$DEVREAD	Routine	1394	100f 756c 790c 1127c 1194c 1303c 1711c
ULTRIX$DLOOK	Routine	972	96f 914c
ULTRIX$FIND	Routine	824	95f 798c
ULTRIX$GET	GlobRout	1456	92f 690a
ULTRIX$GETC	Routine	1630	98f 618c 1780c

Symbol	Type	Defined	Referenced ...					
ULTRIX$IOB	GlobLit	287						
ULTRIX$OPEN	GlobRout	639	91f					
ULTRIX$OPENI	Routine	1143	93f	804c	887c	924c	940c	
ULTRIX$PTINFO	Routine	1323	99f	761c				
ULTRIX$READ	GlobRout	1726	101f	691a				
ULTRIX$READDIR	Routine	1065	97f	1049c				
ULTRIX$SBMAP	Routine	1206	94f	1111c	1689c			
ULTRIX_B_ADD	GlobBind	480						
ULTRIX_IOB_ADD	GlobBind	479						
VBN	ExtReg	521	524=	524.				
	ExtReg	564	576.	586.	587.	616.	629.	630. 633.
	GlobReg	685						
	ExtReg	880						
	ExtReg	1023						
	ExtReg	1095						
	ExtReg	1177						
	ExtReg	1237						
	ExtReg	1354						
	ExtReg	1426						
	GlobReg	1504	1515=	1526.	1529.	1576.		
	ExtReg	1666						
	GlobReg	1764						
WRITE_BUF_ADDR	Local	1672	1678=	1710.	1716=			

CROSS REFERENCE MAP

Line #	Event	File ...
1	Source (start)	DISK$DS:[DS.LOCAL.RELEASE.WORK]ULTRIX.B32;1
8	Module ULTRIX	
70	Library #1	DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
72	Library #2	DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
74	Library #3	SYS$COMMON:[SYSLIB]LIB.L32;2
1789	Eludom ULTRIX	

KEY TO REFERENCE TYPE FLAGS

.	Fetch
=	Store
c	Routine call
a	Address use
@	Indirect use
e	External, external routine, or external literal declaration
f	Forward or forward routine declaration
h	Condition handler enabling
m	Map declaration
u	Undeclare declaration

; Library Statistics
;
; -------- Symbols -------- Pages Processing

```
;       File                                    Total    Loaded   Percent      Mapped      Time
;
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.L32;5
;                                                940         3         0          96      00:00.0
;  DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.L32;5
;                                                675        20         2          47      00:00.0
;  SYS$COMMON:[SYSLIB]LIB.L32;2                20450        25         0        1101      00:00.8




;                                       COMMAND QUALIFIERS

;       BLISS/CROSS/DEBUG/TRACE/OPTIMIZE=(SPACE,LEVEL=3)/OBJECT=DS$R$W:/LIST=DS$R$W: DS$R$W:ULTRIX.B32

;    1790  0
; Size:         3045 code + 156 data bytes
; Run Time:        00:10.7
; Elapsed Time:    00:13.7
; Lines/CPU Min:   10037
; Lexemes/CPU-Min: 80422
; Memory Used:  235 pages
; Compilation Complete
```

Table of contents

(1) 65 UNWIND PROCEDURE CALL STACK
(1) 211 UNWIND CONDITION HANDLER
(1) 226 CALCULATE VALUE OF SP BEFORE CALL

```
                                0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element UNWIND.MAR
                                0000     2 ; *3    24-JUL-1986 10:23:08 BARANSKI "cms error recovery"
                                0000     3 ; *2    16-JUN-1986 13:00:52 MEIER "just looking"
                                0000     4 ; *1     6-FEB-1986 15:22:54 DS ""
                                0000     5 ; DEC/CMS REPLACEMENT HISTORY, Element UNWIND.MAR
                                0000     6         .TITLE  UNWIND - SYSTEM SERVICE UNWIND PROCEDURE CALL STACK
                                0000     7         .IDENT  /01-1/
                                0000     8
                                0000     9 ;
                                0000    10 ; COPYRIGHT (C) 1977, 1980                                          ;G:2
                                0000    11 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                                0000    12 ;
                                0000    13 ; THIS SOFTWARE IS FURNISHED UNDER  A LICENSE FOR USE ONLY  ON  A
                                0000    14 ; SINGLE COMPUTER SYSTEM AND MAY BE  COPIED ONLY WITH  THE INCLU-
                                0000    15 ; SION OF  THE  ABOVE  COPYRIGHT NOTICE.  THIS SOFTWARE,  OR  ANY
                                0000    16 ; OTHER COPIES THEREOF, MAY NOT BE  PROVIDED  OR  OTHERWISE  MADE
                                0000    17 ; AVAILABLE TO ANY OTHER PERSON EXCEPT  FOR  USE  ON  SUCH SYSTEM
                                0000    18 ; AND TO  ONE WHO AGREES  TO  THESE LICENSE  TERMS.  TITLE TO AND
                                0000    19 ; OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
                                0000    20 ;
                                0000    21 ; THE INFORMATION IN THIS SOFTWARE  IS  SUBJECT TO CHANGE WITHOUT
                                0000    22 ; NOTICE AND SHOULD NOT BE CONSTRUED  AS  A COMMITMENT BY DIGITAL
                                0000    23 ; EQUIPMENT CORPORATION.
                                0000    24 ;
                                0000    25 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
                                0000    26 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                                0000    27 ;
                                0000    28 ; D. N. CUTLER 16-DEC-76
                                0000    29 ;
                                0000    30 ; R. S. Riggs  9-April-1979
                                0000    31 ;        01      Added REI to STARTUNWIND to return to correct stack
                                0000    32 ;                after exception.
                                0000    33 ;
                                0000    34 ; SYSTEM SERVICE UNWIND PROCEDURE CALL STACK
                                0000    35 ;
                                0000    36 ; MACRO LIBRARY CALLS
                                0000    37 ;
                                0000    38
                                0000    39         $CHFDEF                          ;DEFINE CONDITION HANDLING ARGLIST OFFSETS
                                0000               .SAVE   LOCAL_BLOCK
                                0000               .RESTORE
                                0000    40         $SSDEF                           ;DEFINE SYSTEM STATUS VALUES
                                0000               .SAVE   LOCAL_BLOCK
                                0000               .RESTORE
                                0000    41
                                0000    42 ;
                                0000    43 ; LOCAL SYMBOLS
                                0000    44 ;
                                0000    45 ; ARGUMENT LIST OFFSET DEFINITIONS
                                0000    46 ;
                                0000    47
00000004                        0000    48 DEPADR=4                                 ;ADDRESS OF NUMBER OF FRAMES TO UNWIND
00000008                        0000    49 NEWPC=8                                  ;CHANGE OF FLOW FINAL RETURN ADDRESS
                                0000    50
                                0000    51 ;
                                0000    52 ; CALL FRAME OFFSET DEFINITIONS
                                0000    53 ;
```

```
                0000
        00000000 0000  HANDLER=0                          ;CONDITION HANDLER ADDRESS
        00000004 0000  SAVPSW=4                           ;SAVED PSW FROM CALL
        00000006 0000  SAVMSK=6                           ;REGISTER SAVE MASK
        00000008 0000  SAVAP=8                            ;SAVED AP REGISTER IMAGE
        0000000C 0000  SAVFP=12                           ;SAVED FP REGISTER IMAGE
        00000010 0000  SAVPC=16                           ;SAVED PC REGISTER IMAGE
        00000014 0000  SAVRG=20                           ;OTHER SAVED REGISTER IMAGES
                0000
            00000000          .PSECT  SEP, SHR, WRT, EXE, LONG
```

```
                                0000     65          .SBTTL  UNWIND PROCEDURE CALL STACK
                                0000     66 ;+
                                0000     67 ; EXE$UNWIND - UNWIND PROCEDURE CALL STACK
                                0000     68 ;
                                0000     69 ; THIS SERVICE PROVIDES THE CAPABILITY TO UNWIND THE PROCEDURE CALL STACK
                                0000     70 ; TO A SPECIFIED DEPTH AFTER A HARDWARE- OR SOFTWARE-DETECTED EXCEPTION
                                0000     71 ; CONDITION HAS BEEN SIGNALLED. OPTIONALLY A CHANGE OF FLOW RETURN ADDRESS
                                0000     72 ; MAY ALSO BE SPECIFIED. THE ACTUAL UNWIND IS NOT PERFORMED IMMEDIATELY BY
                                0000     73 ; THE SERVICE, BUT RATHER THE RETURN ADDRESSES IN THE CALL STACK ARE MODIFIED
                                0000     74 ; SUCH THAT WHEN THE CONDITION HANDLER RETURNS THE UNWIND OCCURS.
                                0000     75 ;
                                0000     76 ; INPUTS:
                                0000     77 ;
                                0000     78 ;       DEPADR(AP) = ADDRESS OF NUMBER OF FRAMES TO UNWIND.
                                0000     79 ;       NEWPC(AP) = CHANGE OF FLOW FINAL RETURN ADDRESS.
                                0000     80 ;
                                0000     81 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                0000     82 ;
                                0000     83 ; OUTPUTS:
                                0000     84 ;
                                0000     85 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO FULLY UNWIND CALL STACK.
                                0000     86 ;
                                0000     87 ;              R0 = SS$_ACCVIO - CALL STACK NOT ACCESSIBLE TO CALLING ACCESS
                                0000     88 ;                   MODE.
                                0000     89 ;
                                0000     90 ;              R0 = SS$_INSFRAME - INSUFFICIENT CALL FRAMES TO UNWIND TO
                                0000     91 ;                   SPECIFIED DEPTH.
                                0000     92 ;
                                0000     93 ;              R0 = SS$_NOSIGNAL - NO SIGNAL IS CURRENTLY ACTIVE TO UNWIND.
                                0000     94 ;
                                0000     95 ;              R0 = SS$_UNWINDING - UNWIND ALREADY IN PROGRESS.
                                0000     96 ;
                                0000     97 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                0000     98 ;
                                0000     99 ;              R0 = SS$_NORMAL - NORMAL COMPLETION.
                                0000    100 ;-
                                0000    101
                         003C   0000    102          .ENTRY  EXE$UNWIND,^M<R2,R3,R4,R5>
      6D    00EA'CF    DE     0002    103          MOVAL   W^CHANDL,(FP)           ;ESTABLISH CONDITION HANDLER
            54    5D   D0     0007    104          MOVL    FP,R4                   ;SET ADDRESS OF FIRST FRAME TO EXAMINE
                                000A    105
                                000A    106 ;
                                000A    107 ; SEARCH CALL STACK FOR A FRAME THAT WAS CREATED BY A CALL FROM THE SIGNAL
                                000A    108 ; DISPATCH VECTOR OR BY A CALL FROM THE UNWIND SIGNAL DISPATCHER.
                                000A    109 ;
                                000A    110
      50    0900 8F    3C     000A    111          MOVZWL  #SS$_NOSIGNAL,R0        ;ASSUME NO SIGNAL ACTIVE
        54    0C A4    D0     000F    112 10$:     MOVL    SAVFP(R4),R4            ;GET ADDRESS OF PREVIOUS FRAME
                     19   13     0013    113          BEQL    20$                     ;IF EQL END OF CALL STACK
10 A4    00000004'8F    D1     0015    114          CMPL    #SYS$CALL_HANDL+4,SAVPC(R4) ;CALL FROM CONDITION HANDLER DISPATCHER?
                     10   13     001D    115          BEQL    30$                     ;IF EQL YES
10 A4    000000D2'8F    D1     001F    116          CMPL    #CALLUNWIND+4,SAVPC(R4) ;CALL FROM UNWIND SIGNAL DISPATCHER?
                     E6   12     0027    117          BNEQ    10$                     ;IF NEQ NO
      50    0928 8F    3C     0029    118          MOVZWL  #SS$_UNWINDING,R0       ;SET ALREADY UNWINDING
                     04   002E    119 20$:     RET                             ;
                                002F    120
                                002F    121 ;
```

```
                                002F   122 ; SET TO UNWIND PROCEDURE CALL STACK TO SPECIFIED DEPTH
                                002F   123 ;
                                002F   124
            53   04 AC    D0  002F   125 30$:    MOVL    DEPADR(AP),R3           ;GET ADDRESS OF NUMBER OF FRAMES TO UNWIND
                    05    13  0033   126         BEQL    40$                     ;IF EQL NONE SPECIFIED
                53  63    D0  0035   127         MOVL    (R3),R3                 ;GET NUMBER OF FRAMES TO UNWIND
                    0F    11  0038   128         BRB     50$                     ;
                52  54    D0  003A   129 40$:    MOVL    R4,R2                   ;COPY CURRENT FRAME ADDRESS
                  00C9    30  003D   130         BSBW    OLDSP                   ;CALCULATE VALUE OF SP BEFORE CALL
            52   0C A2    D0  0040   131         MOVL    CHF$L_MCHARGLST+4(R2),R2 ;GET ADDRESS OF MECHANISM ARGUMENT LIST
        53  08 A2   01    C1  0044   132         ADDL3   #1,CHF$L_MCH_DEPTH(R2),R3 ;CALCULATE DEPTH OF ESTABLISHER'S CALLER
                    3E    15  0049   133 50$:    BLEQ    90$                     ;IF LEQ NO FRAMES TO REMOVE
          10 A4   A3'AF    DE  004B   134         MOVAL   B^STARTUNWIND,SAVPC(R4) ;SET CONDITION HANDLER UNWIND ADDRESS
                          0050   135
                          0050   136 ;
                          0050   137 ; SCAN THROUGH SPECIFIED NUMBER OF FRAMES SETTING EACH FRAME TO UNWIND ON RETURN
                          0050   138 ;
                          0050   139
            55   0C A4    D0  0050   140 60$:    MOVL    SAVFP(R4),R5            ;GET ADDRESS OF PREVIOUS FRAME
                    47    13  0054   141         BEQL    100$                    ;IF EQL INSUFICIENT FRAMES
                    53    D7  0056   142         DECL    R3                      ;ANY MORE FRAMES TO CONSIDER?
                    2F    15  0058   143         BLEQ    90$                     ;IF LEQ NO
        10 A5   00000004'8F   D1  005A   144         CMPL    #SYS$CALL_HANDL+4,SAVPC(R5) ;CALL FROM CONDITION DISPATCHER?
                    1B    12  0062   145         BNEQ    80$                     ;IF NEQ NO
                52  55    D0  0064   146         MOVL    R5,R2                   ;COPY ADDRESS OF CURRENT FRAME
                  009F    30  0067   147         BSBW    OLDSP                   ;CALCULATE VALUE OF SP BEFORE CALL
            52   0C A2    D0  006A   148         MOVL    CHF$L_MCHARGLST+4(R2),R2 ;GET ADDRESS OF MECHANISM ARGUMENT LIST
            04 A2   55    D1  006E   149 70$:    CMPL    R5,CHF$L_MCH_FRAME(R2)  ;UNWOUND TO ESTABLISHER FRAME?
                    0B    13  0072   150         BEQL    80$                     ;IF EQL YES
          10 A5   B6'AF    DE  0074   151         MOVAL   B^LOOPUNWIND,SAVPC(R5)  ;SET FRAME UNWIND ADDRESS
            55   0C A5    D0  0079   152         MOVL    SAVFP(R5),R5            ;GET ADDRESS OF PREVIOUS FRAME
                    EF    11  007D   153         BRB     70$                     ;
        10 A5   B6'AF    DE  007F   154 80$:    MOVAL   B^LOOPUNWIND,SAVPC(R5)  ;SET FRAME UNWIND ADDRESS
                54  55    D0  0084   155         MOVL    R5,R4                   ;SET ADDRESS OF CURRENT FRAME
                    C7    11  0087   156         BRB     60$                     ;
                          0089   157
                          0089   158 ;
                          0089   159 ; MODIFY CHANGE OF FLOW RETURN IF NEW ADDRESS SPECIFIED
                          0089   160 ;
                          0089   161
                50   01    3C  0089   162 90$:    MOVZWL  #SS$_NORMAL,R0          ;SET NORMAL COMPLETION
            51   08 AC    D0  008C   163         MOVL    NEWPC(AP),R1            ;GET CHANGE OF FLOW RETURN ADDRESS
                    10    13  0090   164         BEQL    110$                    ;IF EQL NONE SPECIFIED
            55   0C A4    D0  0092   165         MOVL    SAVFP(R4),R5            ;GET ADDRESS OF PREVIOUS FRAME
                    05    13  0096   166         BEQL    100$                    ;IF EQL INSUFFICIENT FRAMES
          10 A5   51    D0  0098   167         MOVL    R1,SAVPC(R5)            ;SET NEW FINAL RETURN ADDRESS
                    04    009C   168         RET                             ;
         50   012C 8F    3C  009D   169 100$:   MOVZWL  #SS$_INSFRAME,R0        ;SET INSUFFICIENT FRAMES
                    04    00A2   170 110$:   RET                             ;
                          00A3   171
                          00A3   172 ;
                          00A3   173 ; UNWIND HANDLER FRAME
                          00A3   174 ;
                          00A3   175
                          00A3   176 STARTUNWIND:                            ;START OF ACTUAL UNWIND
    6E    24 AE    09    C1  00A3   177         ADDL3   #9,36(SP),(SP)          ; Count of longwords to PSL
                6E   04    C4  00A8   178         MULL2   #4,(SP)                 ; Byte offset
```

```
                50   1C AE    7D  00AB   179          MOVQ    28(SP),R0                ; Restore R0/R1
                   5E   6E    C0  00AF   180          ADDL2   (SP),SP                  ; Remove all args down to PSL
                     B6'AF    9F  00B2   181          PUSHAB  B^LOOPUNWIND             ; Address to resume on correct stack
                              02  00B5   182          REI                              ; Resume below on correct stack
                                  00B6   183
                                  00B6   184 ;
                                  00B6   185 ; UNWIND CALL FRAME SIGNALING CONDITION HANDLER IF ONE IS SPECIFIED
                                  00B6   186 ;
                                  00B6   187
                                  00B6   188          .ENABL  LSB
                                  00B6   189 LOOPUNWIND:                               ;UNWIND CALL FRAME
                         6D       D5  00B6   190          TSTL    (FP)                 ;CONDITION HANDLER SPECIFIED?
                         1C       13  00B8   191          BEQL    10$                  ;IF EQL NO
                7E   0920 8F       3C  00BA   192          MOVZWL  #SS$_UNWIND,-(SP)    ;PUSH UNWIND SIGNAL CONDITION
                         01       DD  00BF   193          PUSHL   #1                   ;PUSH NUMBER OF SIGNAL ARGUMENTS
                         03       BB  00C1   194          PUSHR   #^M<R0,R1>           ;PUSH REGISTERS R0 AND R1
                         00       DD  00C3   195          PUSHL   #0                   ;PUSH FRAME DEPTH
                         5D       DD  00C5   196          PUSHL   FP                   ;PUSH FRAME ADDRESS
                         04       DD  00C7   197          PUSHL   #4                   ;PUSH NUMBER OF MECHANISM ARGUMENTS
                         6E       DF  00C9   198          PUSHAL  (SP)                 ;PUSH ADDRESS OF MECHANISM ARGUMENTS
                      18 AE       DF  00CB   199          PUSHAL  24(SP)               ;PUSH ADDRESS OF SIGNAL ARGUMENTS
                                  00CE   200 CALLUNWIND:                               ;SIGNAL UNWIND
                   00 BD    02     FB  00CE   201          CALLS   #2,@(FP)             ;CALL CONDITION HANDLER
                   50    0C AE     7D  00D2   202          MOVQ    CHF$L_MCH_SAVR0(SP),R0 ;RETRIEVE NEW VALUES FOR R0 AND R1
                   5C    14 AD     DE  00D6   203 10$:     MOVAL   SAVRG(FP),AP         ;GET ADDRESS OF REGISTER SAVE AREA
                      03 06 AD     E9  00DA   204          BLBC    SAVMSK(FP),20$       ;IF LBC R0 NOT SAVED
                      8C    50     D0  00DE   205          MOVL    R0,(AP)+             ;SAVE R0 FOR SUBSEQUENT RESTORATION
                03 06 AD    01     E1  00E1   206 20$:     BBC     #1,SAVMSK(FP),30$    ;IF CLR, R1 NOT SAVED
                      6C    51     D0  00E6   207          MOVL    R1,(AP)              ;SAVE R1 FOR SUBSEQUENT RESTORATION
                                  04  00E9   208 30$:     RET                          ;
                                  00EA   209          .DSABL  LSB
```

```
                                00EA          .SBTTL  UNWIND CONDITION HANDLER
                                00EA     ;
                                00EA     ; UNWIND CONDITION HANDLER
                                00EA     ;
                                00EA
                          0000  00EA     CHANDL: .WORD   0                          ;ENTRY MASK
          50    04 AC     D0    00EC             MOVL    CHF$L_SIGARGLST(AP),R0     ;GET ADDRESS OF SIGNAL ARGUMENT LIST
     04 A0    0920 8F     B1    00F0             CMPW    #SS$_UNWIND,CHF$L_SIG_NAME(R0) ;UNWINDING?
                   10     13    00F6             BEQL    10$                        ;IF EQL YES
          51    08 AC     D0    00F8             MOVL    CHF$L_MCHARGLST(AP),R1     ;GET ADDRESS OF MECHANISM ARGUMENT LIST
       0C A1    04 A0     D0    00FC             MOVL    CHF$L_SIG_NAME(R0),CHF$L_MCH_SAVR0(R1) ;SET FINAL RETURN STATUS
                   7E     7C    0101             CLRQ    -(SP)                      ;CLEAR DEPTH AND NEW PC ARGUMENTS
        FEF8 CF    02     FB    0103             CALLS   #2,W^EXE$UNWIND            ;UNWIND TO ESTABLISHER'S CALLER
                           04    0108     10$:    RET                                ;
```

```
                                0109   226          .SBTTL  CALCULATE VALUE OF SP BEFORE CALL
                                0109   227 ;
                                0109   228 ; SUBROUTINE TO CALCULATE VALUE OF SP BEFORE CALL
                                0109   229 ;
                                0109   230
      7E   06 A2   02   0E   EF 0109   231 OLDSP:  EXTZV   #14,#2,SAVMSK(R2),-(SP) ;GET STACK ALIGNMENT BIAS
      51   06 A2   0C   00   EF 010F   232         EXTZV   #0,#12,SAVMSK(R2),R1    ;GET REGISTER SAVE MASK
                   52   14   C0 0115   233         ADDL    #SAVRG,R2               ;ADD OFFSET TO REGISTER SAVE AREA
                   52   8E   C0 0118   234         ADDL    (SP)+,R2                ;ADD STACK ALIGNMENT BIAS
                     03 51   E9 011B   235 10$:    BLBC    R1,20$                  ;IF LBC CORRESPONDING REGISTER NOT SAVED
                   52   04   C0 011E   236         ADDL    #4,R2                   ;ADJUST FOR SAVED REGISTER
            51   51   FF 8F   78 0121   237 20$:    ASHL    #-1,R1,R1               ;ANY MORE REGISTERS SAVED?
                        F3   12 0126   238         BNEQ    10$                     ;IF NEQ YES
                             05 0128   239         RSB                             ;
                                0129   240
                                0129   241         .END
```

```
CALLUNWIND                      000000CE R  D  02
CHANDL                          000000EA R  D  02
CHF$L_MCHARGLST               = 00000008    D
CHF$L_MCH_DEPTH               = 00000008    D
CHF$L_MCH_FRAME               = 00000004    D
CHF$L_MCH_SAVR0               = 0000000C    D
CHF$L_SIGARGLST               = 00000004    D
CHF$L_SIG_NAME                = 00000004    D
DEPADR                        = 00000004    D
EXE$UNWIND                      00000000 RG D  02
HANDLER                       = 00000000    D
LOOPUNWIND                      000000B6 R  D  02
NEWPC                         = 00000008    D
OLDSP                           00000109 R  D  02
SAVAP                         = 00000008    D
SAVFP                         = 0000000C    D
SAVMSK                        = 00000006    D
SAVPC                         = 00000010    D
SAVPSW                        = 00000004    D
SAVRG                         = 00000014    D
SS$_INSFRAME                  = 0000012C    D
SS$_NORMAL                    = 00000001    D
SS$_NOSIGNAL                  = 00000900    D
SS$_UNWIND                    = 00000920    D
SS$_UNWINDING                 = 00000928    D
STARTUNWIND                     000000A3 R  D  02
SYS$CALL_HANDL                  ********  X   02
```

+-----------------+
! Psect synopsis !
+-----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SEP	00000129 (297.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD WRT NOVEC LONG

```
                                  +-------------------------+
                                  ! Symbol Cross Reference !
                                  +-------------------------+

SYMBOL                      VALUE        DEFINITION      REFERENCES...
------                      -----        ----------      -------------
CALLUNWIND                  000000CE-R   200    (1)    #-116    (1)
CHANDL                      000000EA-R   216    (1)      103    (1)
CHF$L_MCHARGLST            =00000008                   #-131    (1)    #-148    (1)    #-220    (1)
CHF$L_MCH_DEPTH            =00000008                   #-132    (1)
CHF$L_MCH_FRAME            =00000004                   #-149    (1)
CHF$L_MCH_SAVR0            =0000000C                   #-202    (1)    #-221    (1)
CHF$L_SIGARGLST            =00000004                   #-217    (1)
CHF$L_SIG_NAME             =00000004                   #-218    (1)    #-221    (1)
DEPADR                     =00000004     48     (1)    #-125    (1)
EXE$UNWIND                  00000000-R   102    (1)      223    (1)
HANDLER                    =00000000     55     (1)
LOOPUNWIND                  000000B6-R   189    (1)      151    (1)      154    (1)      181    (1)
NEWPC                      =00000008     49     (1)    #-163    (1)
OLDSP                       00000109-R   231    (1)    #-130    (1)    #-147    (1)
SAVAP                      =00000008     58     (1)
SAVFP                      =0000000C     59     (1)    #-112    (1)    #-140    (1)    #-152    (1)    #-165    (1)
SAVMSK                     =00000006     57     (1)    #-204    (1)      206    (1)      231    (1)      232    (1)
SAVPC                      =00000010     60     (1)    #-114    (1)    #-116    (1)    #-134    (1)    #-144    (1)
                                                      #-151    (1)    #-154    (1)    #-167    (1)
SAVPSW                     =00000004     56     (1)
SAVRG                      =00000014     61     (1)      203    (1)    #-233    (1)
SS$_INSFRAME               =0000012C                   #-169    (1)
SS$_NORMAL                 =00000001                   #-162    (1)
SS$_NOSIGNAL               =00000900                   #-111    (1)
SS$_UNWIND                 =00000920                   #-192    (1)    #-218    (1)
SS$_UNWINDING              =00000928                   #-118    (1)
STARTUNWIND                 000000A3-R   176    (1)      134    (1)
SYS$CALL_HANDL              00000000-XR                 #-114    (1)    #-144    (1)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$CHFDEF	1	39 (1)	39 (1)
$DEFINI	1	39 (1)	39 (1) 40 (1)
$SSDEF	21	40 (1)	40 (1)

Opcode Cross Reference

OPCODE	VALUE	REFERENCES...													
ADDL	00C0	233	(1)	234	(1)	236	(1)								
ADDL2	00C0	180	(1)												
ADDL3	00C1	132	(1)	177	(1)										
ASHL	0078	237	(1)												
BBC	00E1	206	(1)												
BEQL	0013	113	(1)	115	(1)	126	(1)	141	(1)	150	(1)	164	(1)	166	(1)
		191	(1)	219	(1)										
BLBC	00E9	204	(1)	235	(1)										
BLEQ	0015	133	(1)	143	(1)										
BNEQ	0012	117	(1)	145	(1)	238	(1)								
BRB	0011	128	(1)	153	(1)	156	(1)								
BSBW	0030	130	(1)	147	(1)										
CALLS	00FB	201	(1)	223	(1)										
CLRQ	007C	222	(1)												
CMPL	00D1	114	(1)	116	(1)	144	(1)	149	(1)						
CMPW	00B1	218	(1)												
DECL	00D7	142	(1)												
EXTZV	00EF	231	(1)	232	(1)										
MOVAL	00DE	103	(1)	134	(1)	151	(1)	154	(1)	203	(1)				
MOVL	00D0	104	(1)	112	(1)	125	(1)	127	(1)	129	(1)	131	(1)	140	(1)
		146	(1)	148	(1)	152	(1)	155	(1)	163	(1)	165	(1)	167	(1)
		205	(1)	207	(1)	217	(1)	220	(1)	221	(1)				
MOVQ	007D	179	(1)	202	(1)										
MOVZWL	003C	111	(1)	118	(1)	162	(1)	169	(1)	192	(1)				
MULL2	00C4	178	(1)												
PUSHAB	009F	181	(1)												
PUSHAL	00DF	198	(1)	199	(1)										
PUSHL	00DD	193	(1)	195	(1)	196	(1)	197	(1)						
PUSHR	00BB	194	(1)												
REI	0002	182	(1)												
RET	0004	119	(1)	168	(1)	170	(1)	208	(1)	224	(1)				
RSB	0005	239	(1)												
TSTL	00D5	190	(1)												

```
                                    +-----------------------------+
                                    ! Directives Cross Reference !
                                    +-----------------------------+

DIRECTIVE       REFERENCES...
---------       -------------
.DSABL          209     (1)
.ENABL          188     (1)
.END            241     (1)
.ENDM           39      (1)     40      (1)
.ENTRY          102     (1)
.IDENT          7       (1)
.NOCROSS        39      (1)     40      (1)
.PAGE           210     (1)     225     (1)     64      (1)
.PSECT          63      (1)
.RESTORE        39      (1)     40      (1)
.SAVE           39      (1)     40      (1)
.SBTTL          211     (1)     226     (1)     65      (1)
.TITLE          6       (1)
.WORD           216     (1)
```

```
                                    +-------------------------+
                                    ! Register Cross Reference !
                                    +-------------------------+

REGISTER  NO. REFERENCES     REFERENCES...
--------  --------------     -------------
AP                 7         #-125    (1)    #-163    (1)    #-203    (1)    #-205    (1)    #-207    (1)    #-217    (1)
                             #-220    (1)
FP                 8         #-103    (1)    #-104    (1)    #-190    (1)    #-196    (1)      201    (1)      203    (1)
                             #-204    (1)      206    (1)
R0                11         #-111    (1)    #-118    (1)    #-162    (1)    #-169    (1)    #-179    (1)    #-194    (1)
                             #-202    (1)    #-205    (1)    #-217    (1)    #-218    (1)    #-221    (1)
R1                10         #-163    (1)    #-167    (1)    #-194    (1)    #-207    (1)    #-220    (1)    #-221    (1)
                             #-232    (1)    #-235    (1)    #-237    (1)
R2                14           102    (1)    #-129    (1)    #-131    (1)    #-132    (1)    #-146    (1)    #-148    (1)
                             #-149    (1)      231    (1)      232    (1)    #-233    (1)    #-234    (1)    #-236    (1)
R3                 6           102    (1)    #-125    (1)    #-127    (1)    #-132    (1)    #-142    (1)
R4                11           102    (1)    #-104    (1)    #-112    (1)    #-114    (1)    #-116    (1)    #-129    (1)
                             #-134    (1)    #-140    (1)    #-155    (1)    #-165    (1)
R5                12           102    (1)    #-140    (1)    #-144    (1)    #-146    (1)    #-149    (1)    #-151    (1)
                             #-152    (1)    #-154    (1)    #-155    (1)    #-165    (1)    #-167    (1)
SP                13         #-177    (1)    #-178    (1)    #-179    (1)    #-180    (1)    #-192    (1)      198    (1)
                               199    (1)    #-202    (1)    #-222    (1)    #-231    (1)    #-234    (1)

                                    +-------------------------+
                                    ! Performance indicators !
                                    +-------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    22    00:00:00.03    00:00:00.24
Command processing               871    00:00:00.22    00:00:00.76
Pass 1                           324    00:00:00.91    00:00:01.69
Symbol table sort                  0    00:00:00.14    00:00:00.13
Pass 2                             9    00:00:00.27    00:00:00.56
Symbol table output                0    00:00:00.01    00:00:00.01
Psect synopsis output              0    00:00:00.00    00:00:00.00
Cross-reference output             2    00:00:00.06    00:00:00.09
Assembler run totals            1228    00:00:01.64    00:00:03.48

The working set limit was 2400 pages.
46032 bytes (90 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 439 non-local and 17 local symbols.
241 source lines were read in Pass 1, producing 40 object records in Pass 2.
14 pages of virtual memory were used to define 8 macros.
```

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.RELEASE.WORK]DIAG.MLB;5              0
DISK$DS:[DS.LOCAL.RELEASE.WORK]DS.MLB;6                0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       5
TOTALS (all libraries)                                 5

486 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/CROSS=ALL/SHOW=MEB/ENABLE=(TRACE,DEBUG)/OBJECT=DS$R$W:/LIST=DS$R$W DS$R$W:UNWIND.MAR+SYS$LIBRARY:LIB/LIB+DS$R$W:DS/LIB+DS$R$W:
```

Table of contents

(1) 36 DECLARATIONS

```
0000          .TITLE  VER730
0000          .IDENT  "10.10"
0000          .LIST   MEB
0000          .NLIST  CND
0000
0000    ;
0000    ; Copyright (c) 1977, 1982
0000    ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    ;
0000    ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    ; REMAIN IN DEC.
0000    ;
0000    ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    ; CORPORATION.
0000    ;
0000    ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    ;
0000
0000    ;++
0000    ; FACILITY:     VAX DIAGNOSTIC SUPERVISOR.
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ; ENVIRONMENT:
0000    ;
0000    ; AUTHOR:       KEN CHAPMAN     13-SEP-78       VERSION 01.
0000    ;--
```

```
                                    0000          .SBTTL  DECLARATIONS
                                    0000    ;
                                    0000    ; INCLUDE FILES:
                                    0000    ;
                                    0000
                                    0000    ;
                                    0000    ; MACROS:
                                    0000    ;
                                    0000
                                    0000    ;
                                    0000    ; EQUATED SYMBOLS:
                                    0000    ;
                                    0000
                                    0000
                                    0000    ;
                                    0000    ; OWN STORAGE:
                                    0000    ;
                                00000000          .PSECT  DATA, SHR, NOEXE, NOWRT, LONG
                                    0000    DS$GQ_DAYTIM::
43 45 44 2D 37 31 00000008'010E0000' 0000          .ASCID  "17-DEC-1987 15:30:16.15"
3A 30 33 3A 35 31 20 37 38 39 31 2D  000E
                     35 31 2E 36 31  001A
                                    001F    DS$GT_START::
                                35' 001F          .BYTE   10$-.-1
43 49 54 53 4F 4E 47 41 49 44 2F 21  0020          .ASCII  "!/DIAGNOSTIC SUPERVISOR.  ZZ-ENSAA-"
2E 52 4F 53 49 56 52 45 50 55 53 20  002C
   2D 41 41 53 4E 45 2D 5A 5A 20 20  0038
                 2D 30 31 2E 30 31  0043          .ASCII  "10.10-"
                          31 31 38  0049          .ASCII  "811"
         2F 21 44 25 30 32 21 20 20  004C          .ASCII  "  !20%D!/"
                                    0055    10$:
                                    0055    DS$GT_VDS_Version::                          ;                                [02]
      31 31 38 2D 30 31 2E 30 31 00' 0055          .ASCIC  "10.10-811"
                                09  0055
                                    005F    DS$GT_PROMPT::
                     20 3E 53 44 00' 005F          .ASCIC  \DS> \
                                04  005F
                                    0064
                                00000000          .PSECT  CODE, SHR, EXE, NOWRT, BYTE
                                    0000    DS$ENTRY::
                              0000' 0000          .VECTOR DS$USERMODE,^M<>
                  00000002'EF   17  0002          JMP     DS$USERMODE+2                    ; GO TO USER MODE INIT CODE.    [02]
                                    0008
                                    0008          .END    DS$ENTRY
```

```
DS$ENTRY                        00000000 RG    02
DS$GQ_DAYTIM                    00000000 RG    01
DS$GT_PROMPT                    0000005F RG    01
DS$GT_START                     0000001F RG    01
DS$GT_VDS_VERSION               00000055 RG    01
DS$USERMODE                     ******** X     02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
DATA	00000064 (100.)	01 (1.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC LONG
CODE	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.03	00:00:00.19
Command processing	841	00:00:00.15	00:00:00.46
Pass 1	62	00:00:00.07	00:00:00.23
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	7	00:00:00.04	00:00:00.12
Symbol table output	0	00:00:00.00	00:00:00.00
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	934	00:00:00.30	00:00:01.02

```
The working set limit was 2012 pages.
1109 bytes (3 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 6 non-local and 1 local symbols.
73 source lines were read in Pass 1, producing 13 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.
```

Macro library statistics

Macro library name	Macros defined
SYS$COMMON:[SYSLIB]STARLET.MLB;2	0

```
0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO VER730;811/LIS=VER730;811/OBJ=VER730;811
```

Table of contents

(1) 42 History
(1) 52 Declarations
(1) 66 Return-only Dummy Routines ; [02]
(1) 111 MUTEX Control Routines

```
                        0000     1 ; DEC/CMS REPLACEMENT HISTORY, Element VMSDUMMY.MAR
                        0000     2 ; *1     6-FEB-1986 15:23:43 DS ""
                        0000     3 ; DEC/CMS REPLACEMENT HISTORY, Element VMSDUMMY.MAR
                        0000     4         .TITLE  VMSDUMMY        *** VMSDUMMY Fake VMS driver entry points
                        0000     5         .IDENT  '05-02'
                        0000     6
                        0000     7
                        0000     8 ;
                        0000     9 ;*****************************************************************************
                        0000    10 ;*                                                                           *
                        0000    11 ;*  COPYRIGHT (c) 1977, 1978, 1979, 1980                                     *
                        0000    12 ;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
                        0000    13 ;*                                                                           *
                        0000    14 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED   *
                        0000    15 ;*  ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE  *
                        0000    16 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER   *
                        0000    17 ;*  COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
                        0000    18 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY   *
                        0000    19 ;*  TRANSFERRED.                                                             *
                        0000    20 ;*                                                                           *
                        0000    21 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE   *
                        0000    22 ;*  AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
                        0000    23 ;*  CORPORATION.                                                             *
                        0000    24 ;*                                                                           *
                        0000    25 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS   *
                        0000    26 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        0000    27 ;*                                                                           *
                        0000    28 ;*****************************************************************************
                        0000    29
                        0000    30 ;++
                        0000    31 ; FACILITY : Diagnostic Supervisor
                        0000    32 ;
                        0000    33 ; ABSTRACT:
                        0000    34 ;       This module includes dummies of VMS routines which are either
                        0000    35 ; unnecessary in the Supervisor, or exist only in drastically reduced form,
                        0000    36 ; and do not fit within any other Supervisor module.
                        0000    37 ;
                        0000    38 ; ENVIRONMENT:
                        0000    39 ;
                        0000    40 ;--
```

```
                     0000     42          .Sbttl  History
                     0000     43 ;
                     0000     44 ; AUTHOR:        Dave Butenhof   CREATION DATE: 18-jun-1980
                     0000     45 ;
                     0000     46 ; MODIFIED BY:
                     0000     47 ;        01      Jack Stansbury, Version 6.8, July 1, 1982
                     0000     48 ;                Took out the Exe$PwrTimChk routine label because I am adding
                     0000     49 ;                the real routine to the QIO module.
                     0000     50 ;-
```

```
0000          .Sbttl Declarations
0000    ;
0000    ; INCLUDE FILES
0000    ;
0000
0000    ;
0000    ; EQUATES
0000    ;
0000          $SSDEF          ; Define SS return codes
0000          .SAVE   LOCAL_BLOCK
0000          .RESTORE
0000
0000    ;
0000    ; OWN STORAGE
0000    ;
```

```
                    0000          .Sbttl  Return-only Dummy Routines      ;                                [02]
                00000000          .Psect  Code, Shr, Exe, NoWrt, Byte     ;                                [02]
                    0000
                    0000    ;++
                    0000    ; FUNCTIONAL DESCRIPTION:
                    0000    ;       Routines which require no Supervisor action.  Rather than edit the
                    0000    ; calls from sources, these dummy entry points are provided to return
                    0000    ; immediately.
                    0000    ;
                    0000    ; CALLING SEQUENCE:
                    0000    ;       JSB type.
                    0000    ;
                    0000    ; INPUT PARAMETERS:
                    0000    ;       Ignored.
                    0000    ;
                    0000    ; IMPLICIT INPUTS:
                    0000    ;       None
                    0000    ;
                    0000    ; OUTPUT PARAMETERS:
                    0000    ;       R0 may indicate completion status or fake function value.
                    0000    ;
                    0000    ; IMPLICIT OUTPUTS:
                    0000    ;       none
                    0000    ;
                    0000    ; COMPLETION CODES:
                    0000    ;
                    0000    ; SIDE EFFECTS:
                    0000    ;       none
                    0000    ;--
                    0000
                    0000    MT$CHECK_ACCESS::
          50    D4  0000          CLRL    R0
                    0002    EXE$SNDEVMSG::
                    0002    ERL$RELEASEMB::
                    0002    ERL$DEVICERR::
                    0002    ERL$DEVICTMO::
                    0002    ;       EXE$PWRTIMCHK::                 ; Put into QIO module                    [02]
                05  0002          RSB                             ; Return
                    0003
                    0003    EXE$BUFFRQUOTA::
                    0003    EXE$BUFQUOPRC::
                    0003    SCH$WAKE::
    50    01    D0  0003          MOVL    #SS$_NORMAL, R0         ; Success code
                05  0006          RSB
```

```
0007   111          .Sbttl  MUTEX Control Routines
0007   112
0007   113 ;++
0007   114 ; FUNCTIONAL DESCRIPTION:
0007   115 ;       Lock and unlock MUTEX database.  Since the Supervisor does not
0007   116 ; implement multi-programming, a "MUTEX"-type lock is implemented via
0007   117 ; disabling AST delivery.
0007   118 ;
0007   119 ; CALLING SEQUENCE:
0007   120 ;       SCH$LOCKW -
0007   121 ;                 JSB     SCH$LOCKW
0007   122 ;       SCH$UNLOCK -
0007   123 ;                 JSB     SCH$UNLOCK
0007   124 ;
0007   125 ; INPUT PARAMETERS:
0007   126 ;       Ignored.  In Supervisor implmentation, which MUTEX is targeted for
0007   127 ; lock or unlock is irrelevant, since no data is accessed.
0007   128 ;
0007   129 ; IMPLICIT INPUTS:
0007   130 ;       none
0007   131 ;
0007   132 ; OUTPUT PARAMETERS:
0007   133 ;       none
0007   134 ;
0007   135 ; IMPLICIT OUTPUTS:
0007   136 ;       none
0007   137 ;
0007   138 ; COMPLETION CODES:
0007   139 ;       SS$_NORMAL
0007   140 ;
0007   141 ; SIDE EFFECTS:
0007   142 ;       SCH$LOCKW results in AST delivery for the current mode to be
0007   143 ;       disabled.  SCH$UNLOCK results in AST delivery being re-enabled.
0007   144 ;--
```

```
                        0007   146 ;
                        0007   147 ; Lock common data block for write
                        0007   148 ;
                        0007   149         .ENABL  LSB
                        0007   150
                        0007   151 SCH$LOCKW::
                7E  50  7D  0007   152         MOVQ    R0,-(SP)                ; Save R0, R1
                    7E  D4  000A   153         CLRL    -(SP)                   ; Set to disable AST delivery
                    05  11  000C   154         BRB     10$                     ; Join common code
                        000E   155
                        000E   156 SCH$UNLOCK::
                7E  50  7D  000E   157         MOVQ    R0,-(SP)                ; Save R0, R1
                    01  DD  0011   158         PUSHL   #1                      ; Set to enable AST delivery
                        0013   159
      00000000'9F   01  FB  0013   160 10$:    CALLS   #1, a#EXE$SETAST        ; Enable/disable AST delivery
                50  8E  7D  001A   161         MOVQ    (SP)+,R0                ; Restore registers
                    05  001D   162         RSB                             ; Return with SETAST completion code
                        001E   163
                        001E   164         .DISABLE LSB
                        001E   165         .END
```

```
ERL$DEVICERR                      00000002 RG D  02
ERL$DEVICTMO                      00000002 RG D  02
ERL$RELEASEMB                     00000002 RG D  02
EXE$BUFFRQUOTA                    00000003 RG D  02
EXE$BUFQUOPRC                     00000003 RG D  02
EXE$SETAST                        ******** X    02
EXE$SNDEVMSG                      00000002 RG D  02
MT$CHECK_ACCESS                   00000000 RG D  02
SCH$LOCKW                         00000007 RG D  02
SCH$UNLOCK                        0000000E RG D  02
SCH$WAKE                          00000003 RG D  02
SS$_NORMAL                      = 00000001    D
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CODE	0000001E (30.)	02 (2.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL                          VALUE        DEFINITION      REFERENCES...
------                          -----        ----------      -------------
ERL$DEVICERR                    00000002-R   100    (1)
ERL$DEVICTMO                    00000002-R   101    (1)
ERL$RELEASEMB                   00000002-R   99     (1)
EXE$BUFFRQUOTA                  00000003-R   105    (1)
EXE$BUFQUOPRC                   00000003-R   106    (1)
EXE$SETAST                      00000000-XR                  160    (1)
EXE$SNDEVMSG                    00000002-R   98     (1)
MT$CHECK_ACCESS                 00000000-R   96     (1)
SCH$LOCKW                       00000007-R   151    (1)
SCH$UNLOCK                      0000000E-R   156    (1)
SCH$WAKE                        00000003-R   107    (1)
SS$_NORMAL                     =00000001                   #-108    (1)

                                        +-------------------------+
                                        ! Performance indicators !
                                        +-------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    25    00:00:00.02    00:00:00.21
Command processing               919    00:00:00.26    00:00:00.81
Pass 1                           396    00:00:00.77    00:00:01.39
Symbol table sort                  0    00:00:00.12    00:00:00.12
Pass 2                            32    00:00:00.26    00:00:00.43
Symbol table output                2    00:00:00.00    00:00:00.00
Psect synopsis output              2    00:00:00.00    00:00:00.00
Cross-reference output             4    00:00:00.01    00:00:00.01
Assembler run totals            1382    00:00:01.44    00:00:02.97

The working set limit was 1900 pages.
40786 bytes (80 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 421 non-local and 1 local symbols.
165 source lines were read in Pass 1, producing 36 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.

                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DS:[DS.LOCAL.LIBRARY]DS.MLB;1                     0
DISK$DS:[DS.LOCAL.LIBRARY]DIAG.MLB;1                   0
SYS$COMMON:[SYSLIB]LIB.MLB;2                           0
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       4
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       0
TOTALS (all libraries)                                 4

469 GETS were required to define 4 macros.
```

```
ZZ-ENSAA-10.10  VAX-11 Macro Run Statistics                    3-MAR-1988       Fiche 20 Frame G4          Sequence 3959
VMSDUMMY                    *** VMSDUMMY Fake VMS driver entry point 10-DEC-1987 18:29:13  VAX/VMS Macro V04-00          Page   9
VAX-11 Macro Run Statistics                                    3-JAN-1985 16:57:39  [DS.LOCAL.LIBRARY]VMSDUMMY.MAR;1  (1)

There were no errors, warnings or information messages.

MACRO/CROSS=SYM/ENABLE=(DEBUG,TRACE)/SHOW=MEB SYS$LIBRARY:STARLET.MLB/LIBRARY+SYS$LIBRARY:LIB.MLB/LIBRARY+DS$W:DIAG.MLB/LIBRARY+DS$W
```